The Application Layer: SMTP, Multimedia

Lecture 7

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

Srinivas Narayana

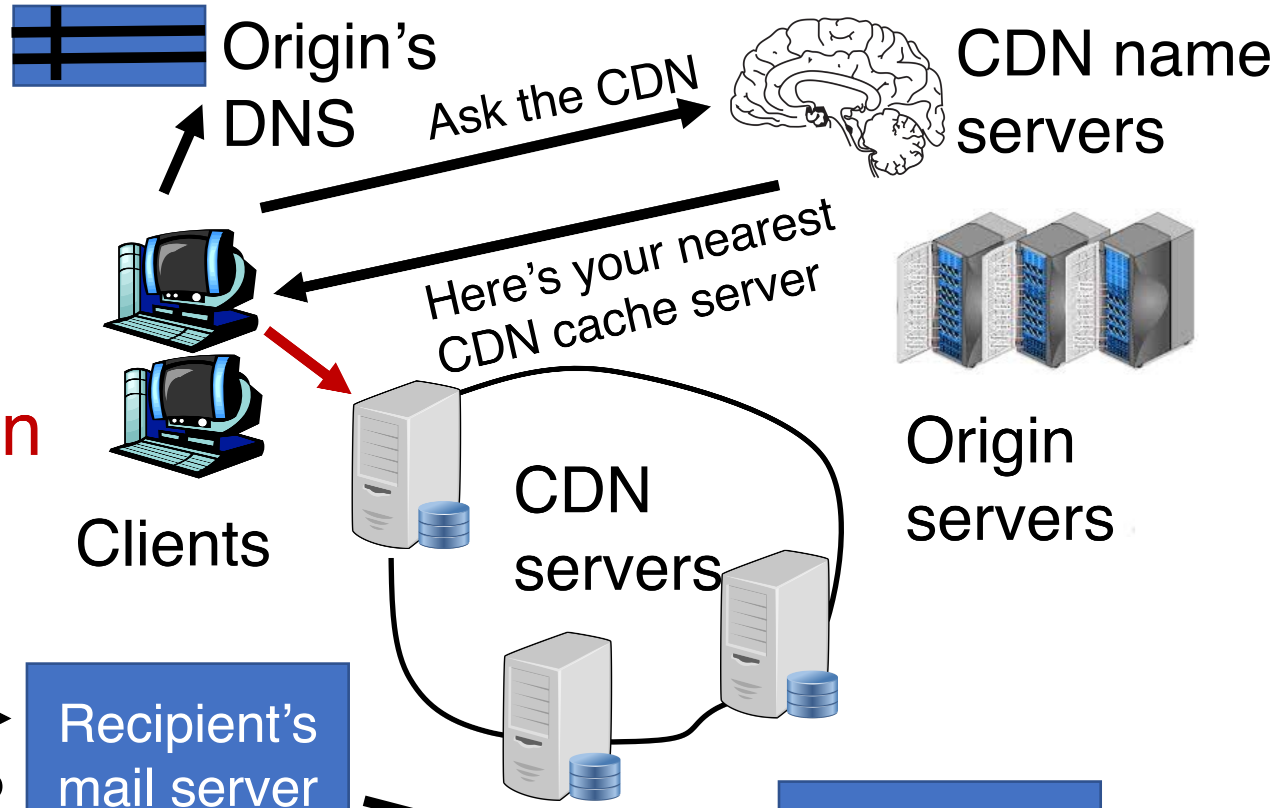
RUTGERS
UNIVERSITY | NEW BRUNSWICK

Quick recap of concepts

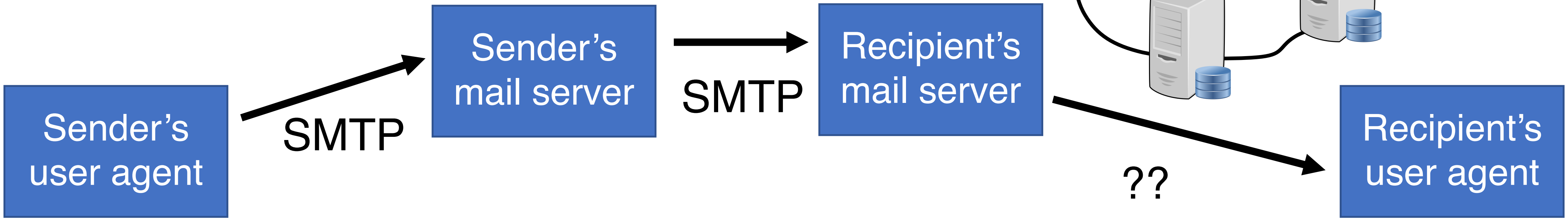
HyperText Transfer Protocol (HTTP)

Caching
e.g., proxy server

Content Distribution Networks

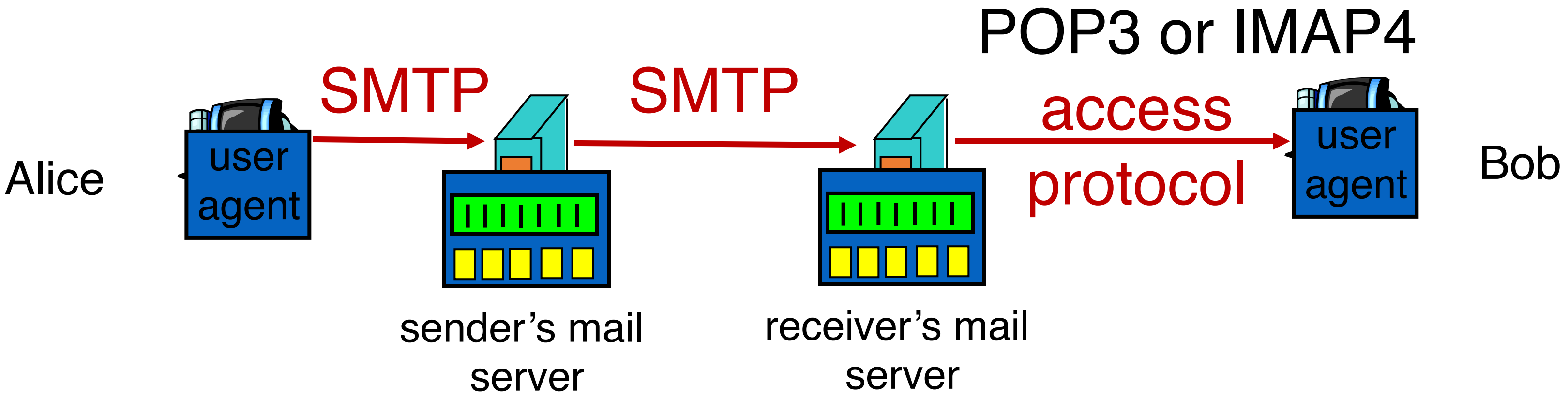


Simple Mail Transfer Protocol (SMTP)



Mail Access Protocols

Mail access protocols



- SMTP: delivery/storage to receiver’s server. Focused on push
- Mail access protocol: pull from server
  - POP: Post Office Protocol [RFC 1939]
    - Client connects to POP3 server on TCP port 110
  - IMAP: Internet Mail Access Protocol [RFC 1730]
    - Client connects to TCP port 143
  - HTTP: gmail, outlook, etc.

POP vs IMAP

- POP3
- Stateless server
- UA-heavy processing
- UA retrieves email from server, then typically deleted from server
- Latest changes are at the UA
- Simple protocol (`list`, `retr`, `del` within a POP session)

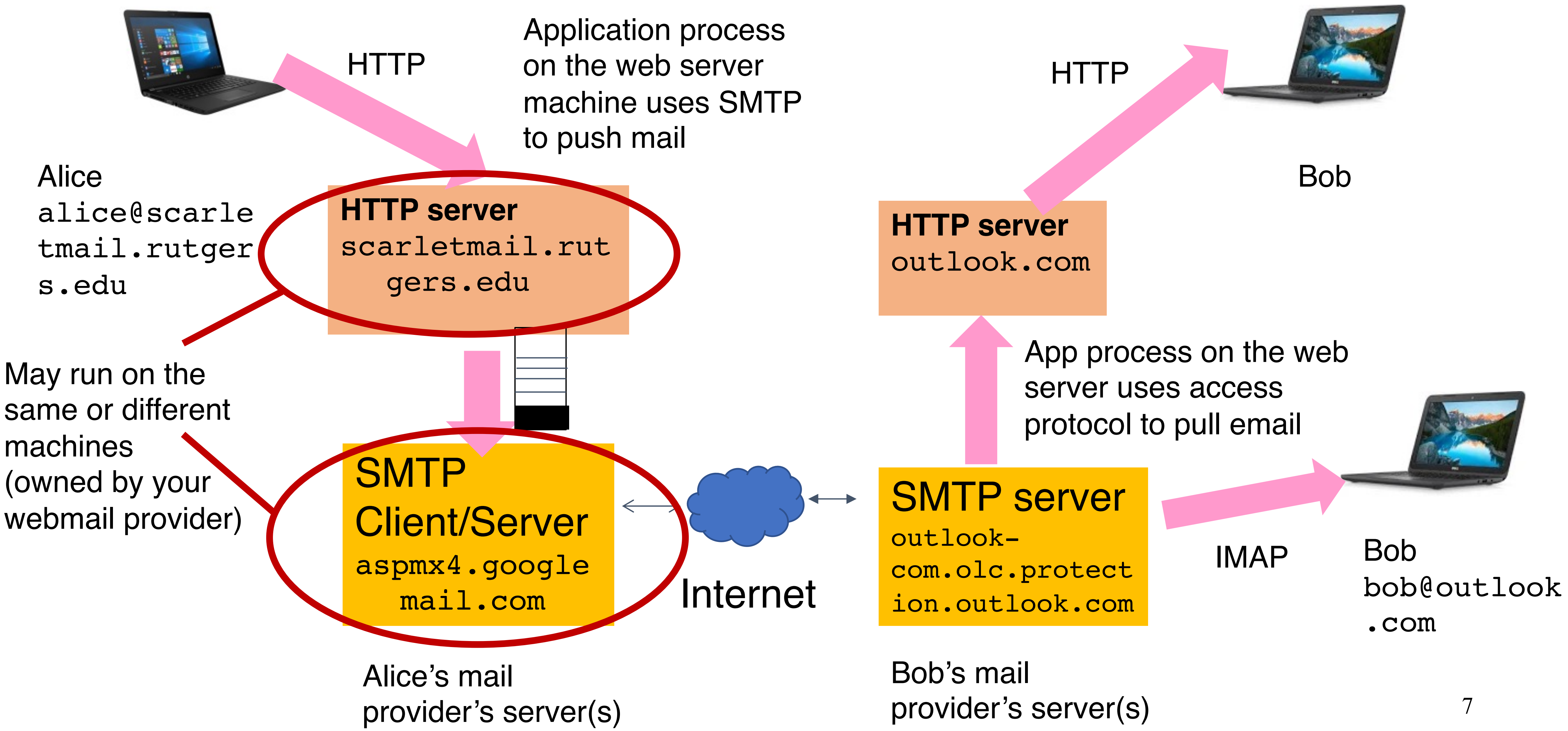
- IMAP4
- Stateful server
- UA and server processing
- Server sees folders, etc. which are visible to UAs
- Latest changes are at the server
- Complex protocol
- Heavily used: email sync across devices, reliable, …

What about web-based email?

- Connect to mail servers via web browser
  - Ex: gmail, scarletmail, etc.

- Browsers speak HTTP
- Email servers speak SMTP
- Need to bridge these two

Web based email



Comparing SMTP with HTTP

- HTTP: pull
- SMTP: push

- both have ASCII command/response interaction, status codes

- HTTP: each object encapsulated in its own response msg
- SMTP: multiple objects sent in multipart msg

- HTTP: can put non-ASCII data directly in response (dedicated entity body for binary data)
- SMTP: need ASCII-based encoding (base64)

More themes from app-layer protocols

- Keep it simple until you really need complexity
  - Start with ASCII-based design; stateless servers. Then introduce:
  - Cookies for HTTP state
  - Stateful mail (IMAP, folders, etc.) for email organization
  - Security extensions (e.g., HTTPS, IMAPS, SMTPS, …)
  - Performance optimizations: persistence, caching, indirection, …
  - Use headers as much as possible to non-intrusively evolve functionality
- Partition functions based on what's done best at the user (app), protocol, and infrastructure. Examples:
  - Content rendering for users (browser, UA) separate from protocol operations (mail server)
  - Mail UAs don't need to be always on to send or receive email reliably. That's the mail server's job (an "infrastructure" concern)

Multimedia:
Data Representations

RUTGERS

UNIVERSITY | NEW BRUNSWICK

Multimedia networking

- Many applications on the Internet use audio or video
- IP video traffic will be 82 percent of all IP traffic […] by 2022, up from 75 percent in 2017
- CCTV traffic over the Internet will increase sevenfold between 2017 to 2022
- Internet video to TV will increase threefold between 2017 to 2022.
- Consumer Video-on-Demand (VoD) traffic will nearly double by 2022

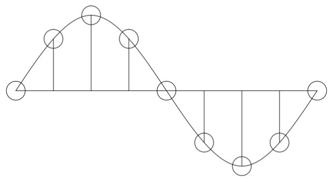
Source: Cisco visual networking index 2017--22

What's different about these applications?

- Traditional applications (HTTP(S), SMTP)
  - Delay tolerant but not loss tolerant
  - Data used *after* transfer complete
- Multimedia applications are often real time
  - Data delivery time *during transfer* matters for user experience
- Video/audio streaming
  - Delay-sensitive
- Real-time audio and video
  - Delays > 400 ms for audio is a bad user experience
  - Somewhat loss tolerant

Digital representation of audio and video

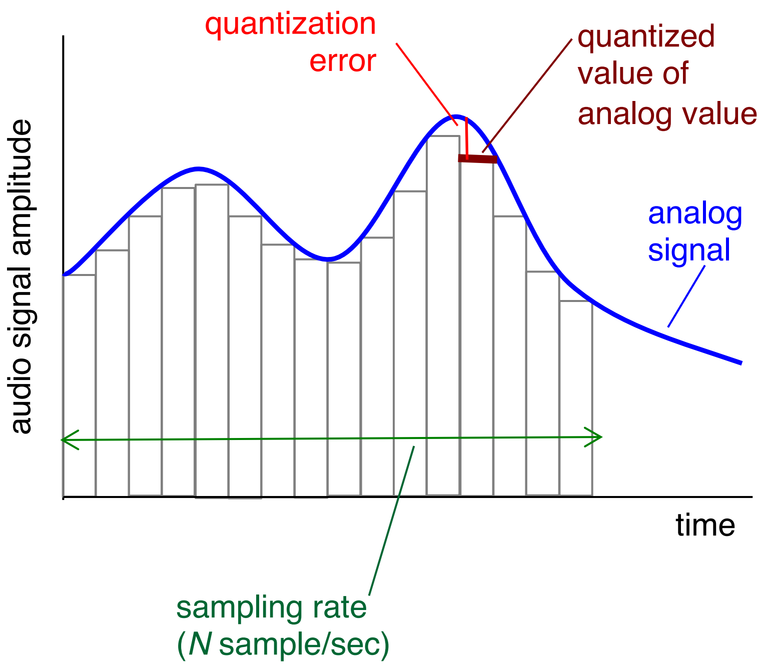
Digital representation of audio



- 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 accuracy
  - More levels ➔ more bits to store & need more bandwidth to transmit
- Compress
  - Compact representation of quantized values

Audio representation

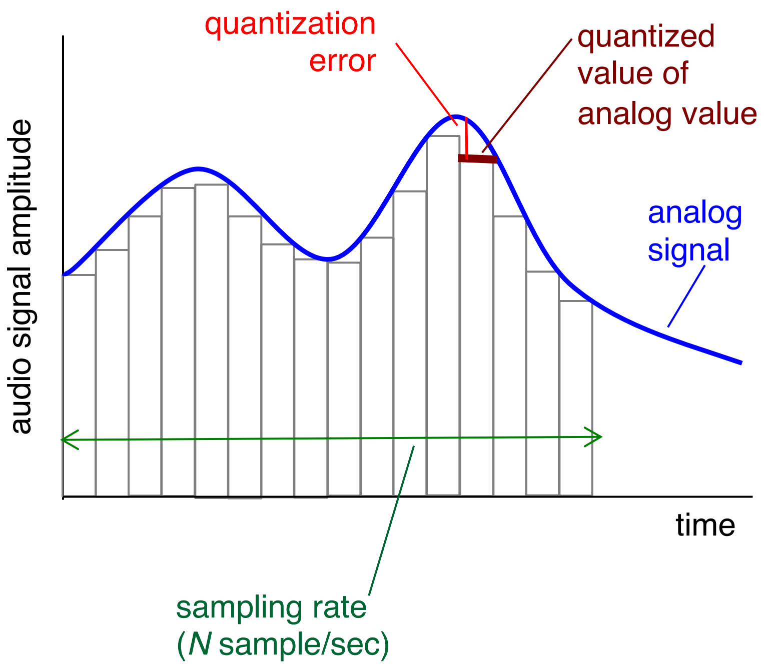
- analog audio signal sampled at constant rate
  - telephone: 8,000 samples/sec
  - CD music: 44,100 samples/sec
- each sample quantized, i.e., rounded
  - e.g., 2^8=256 possible quantized values
  - each quantized value represented by bits, e.g., 8 bits for 256 values



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

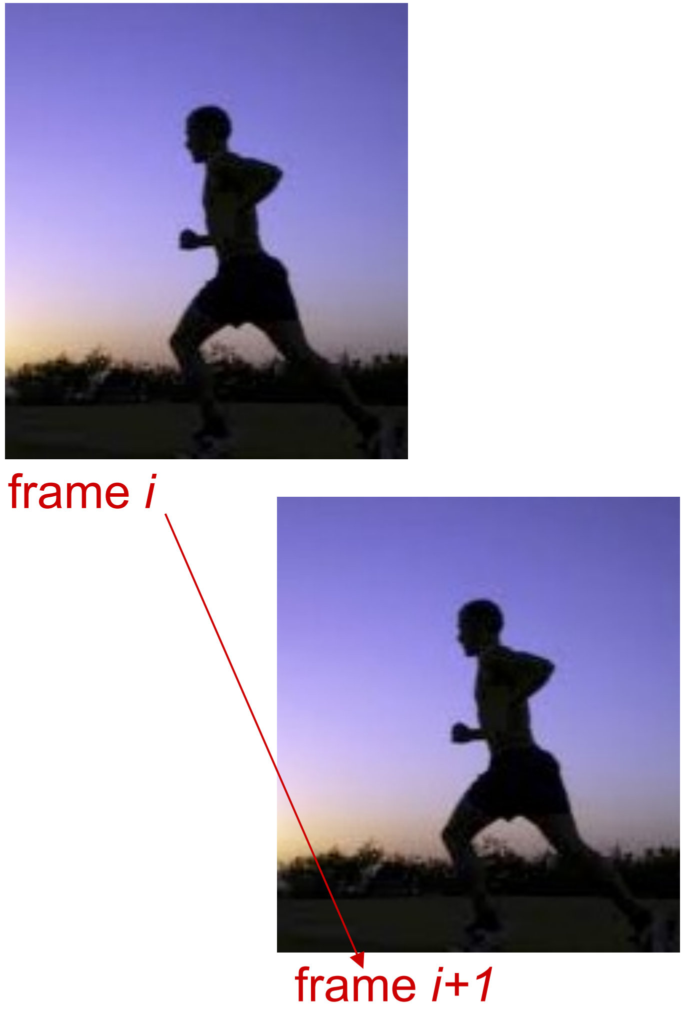


Example rates

- 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

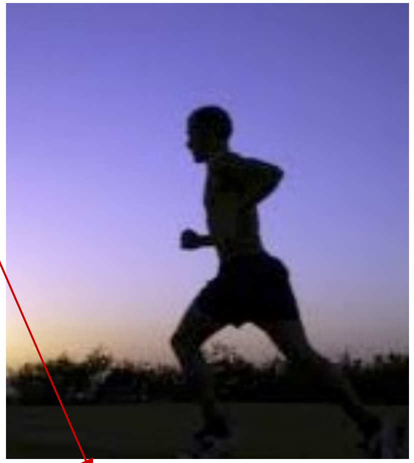


Video representation

- 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)
- Coding/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*



frame *i+1*

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

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: better 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 (unless HD)

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

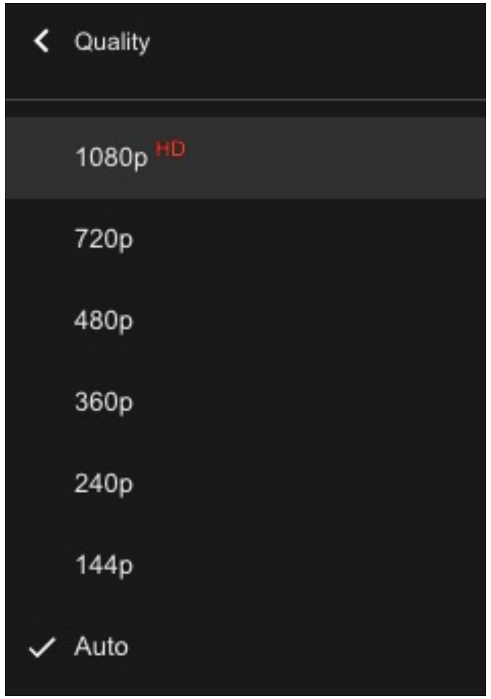
Networking multimedia: 3 types

- 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, 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

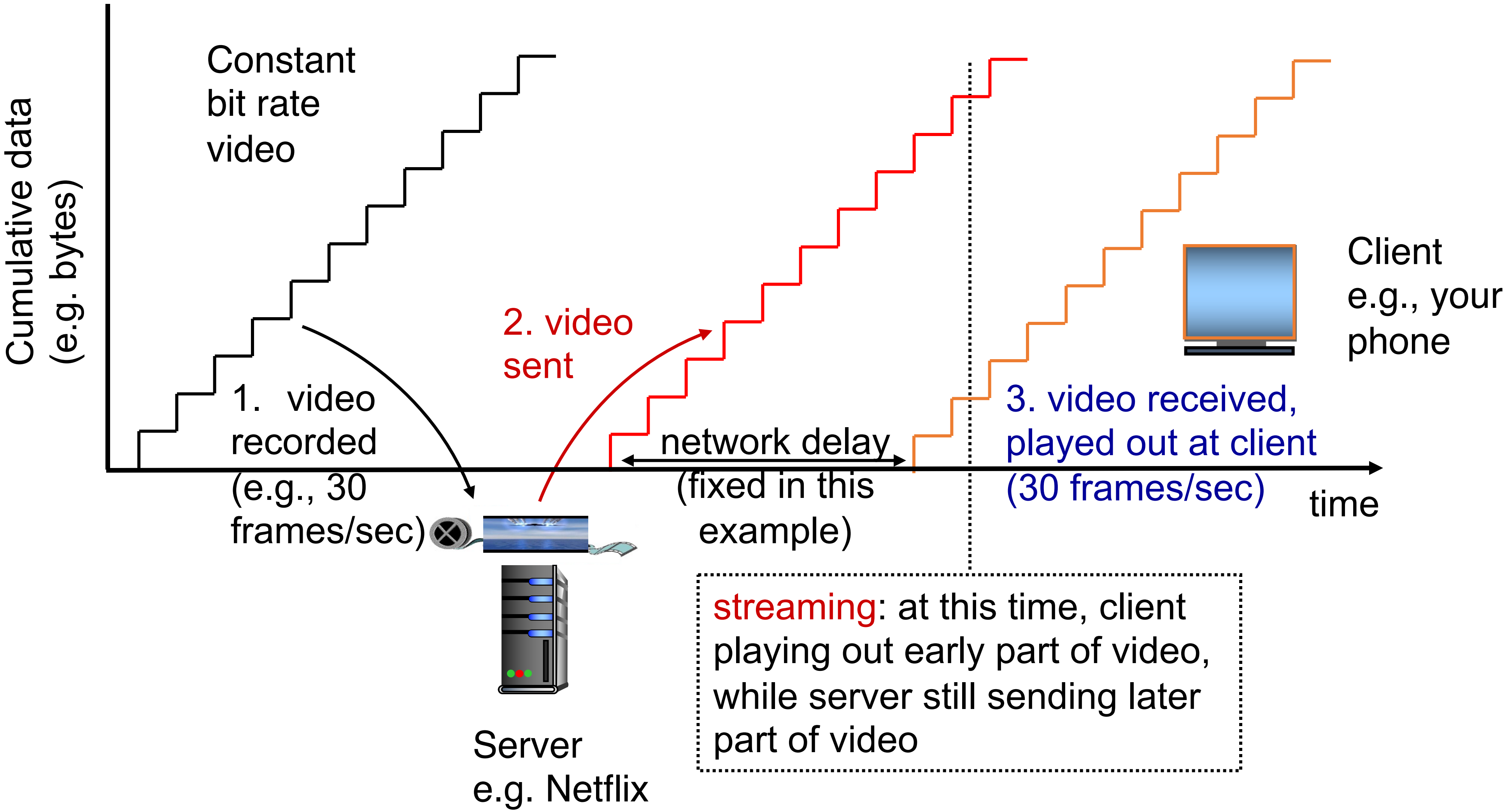
On-demand Video Streaming

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



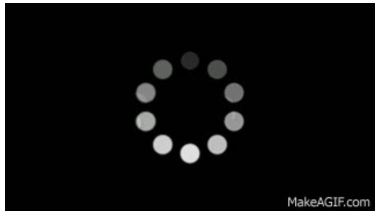
Streaming stored video



streaming: at this time, client playing out early part of video, while server still sending later part of video

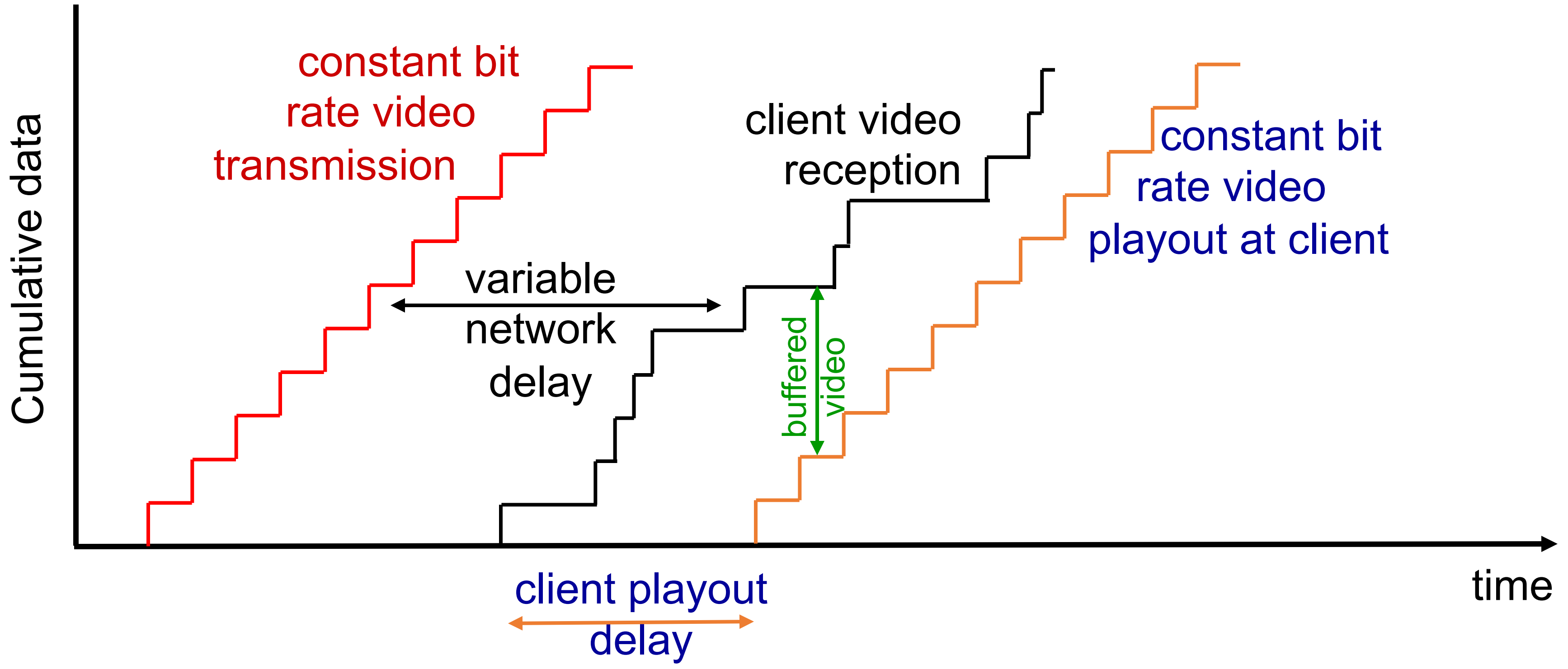
Streaming stored video: challenges

- 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

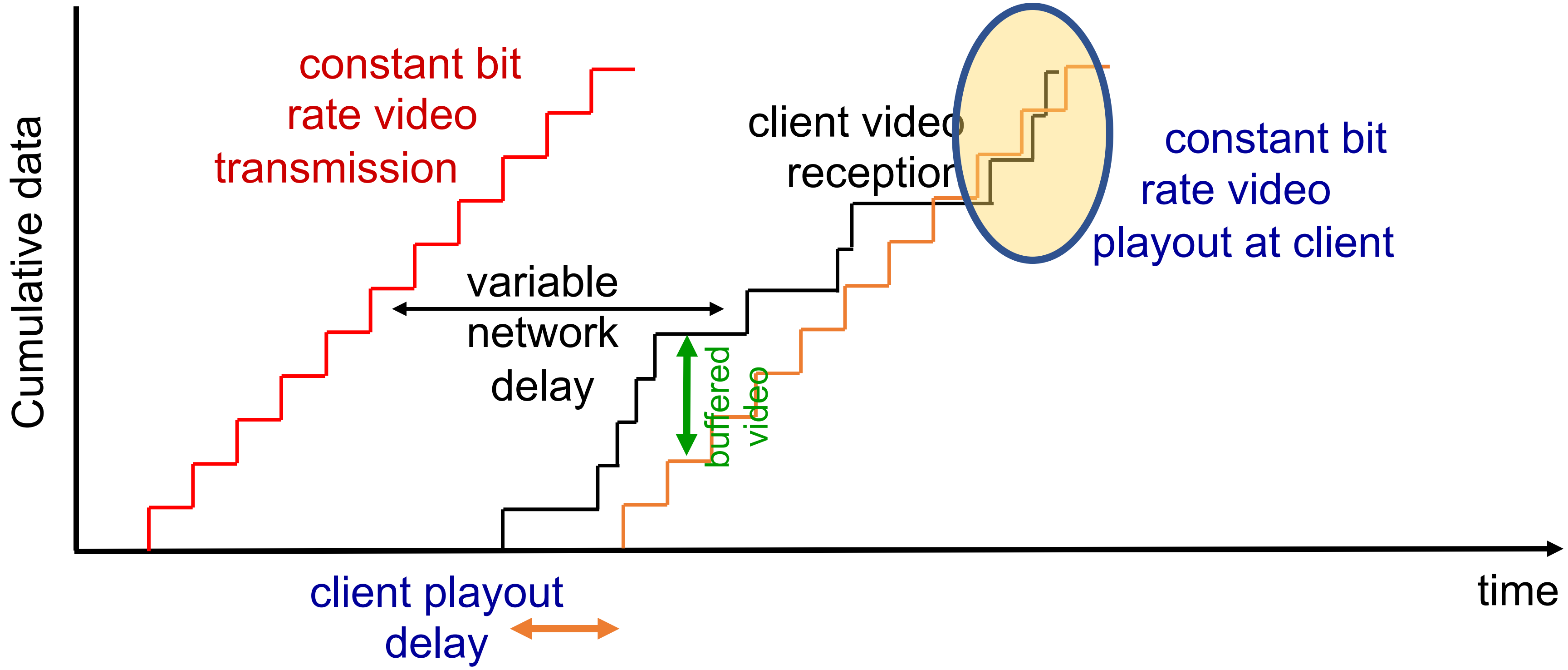
Scenario 1: Constant bit-rate video



Client-side buffering with playout delay:

compensate for network-added delays and variations in the delay

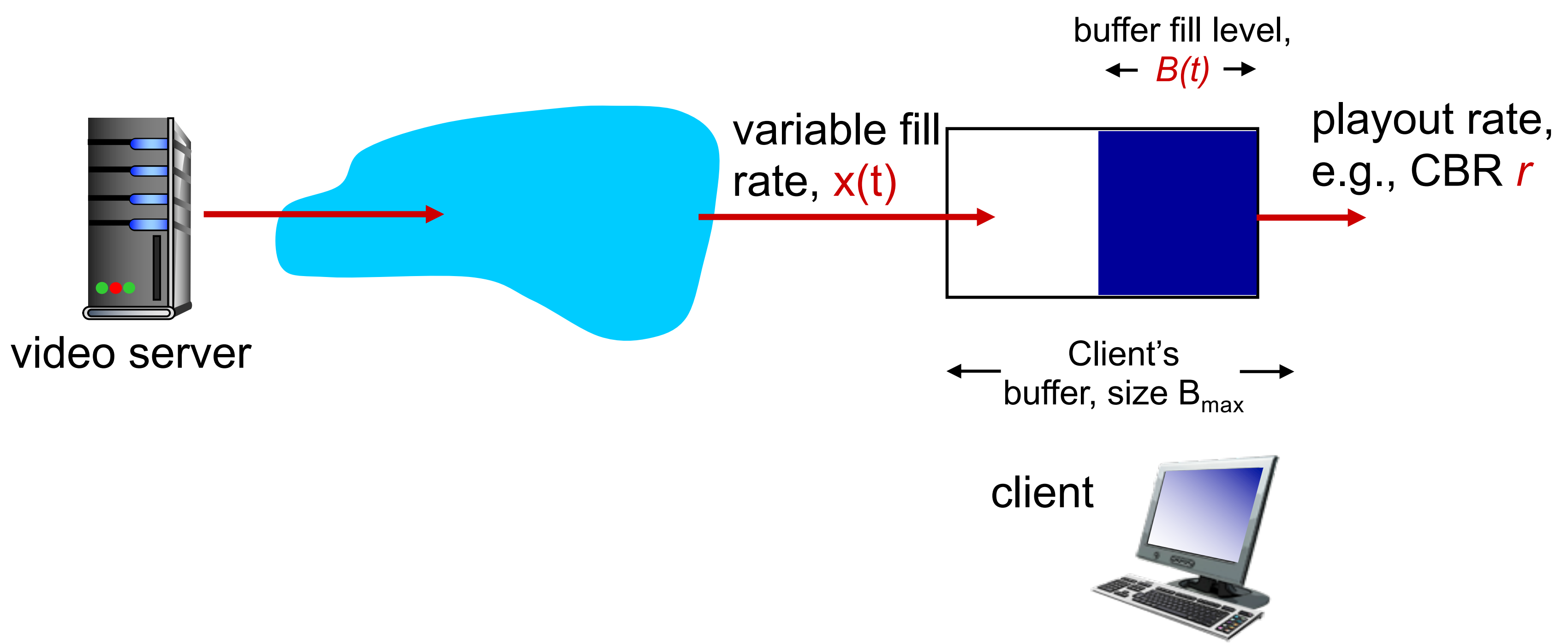
Scenario 2: Small playout delay



Playout delay that's too small can cause stalls

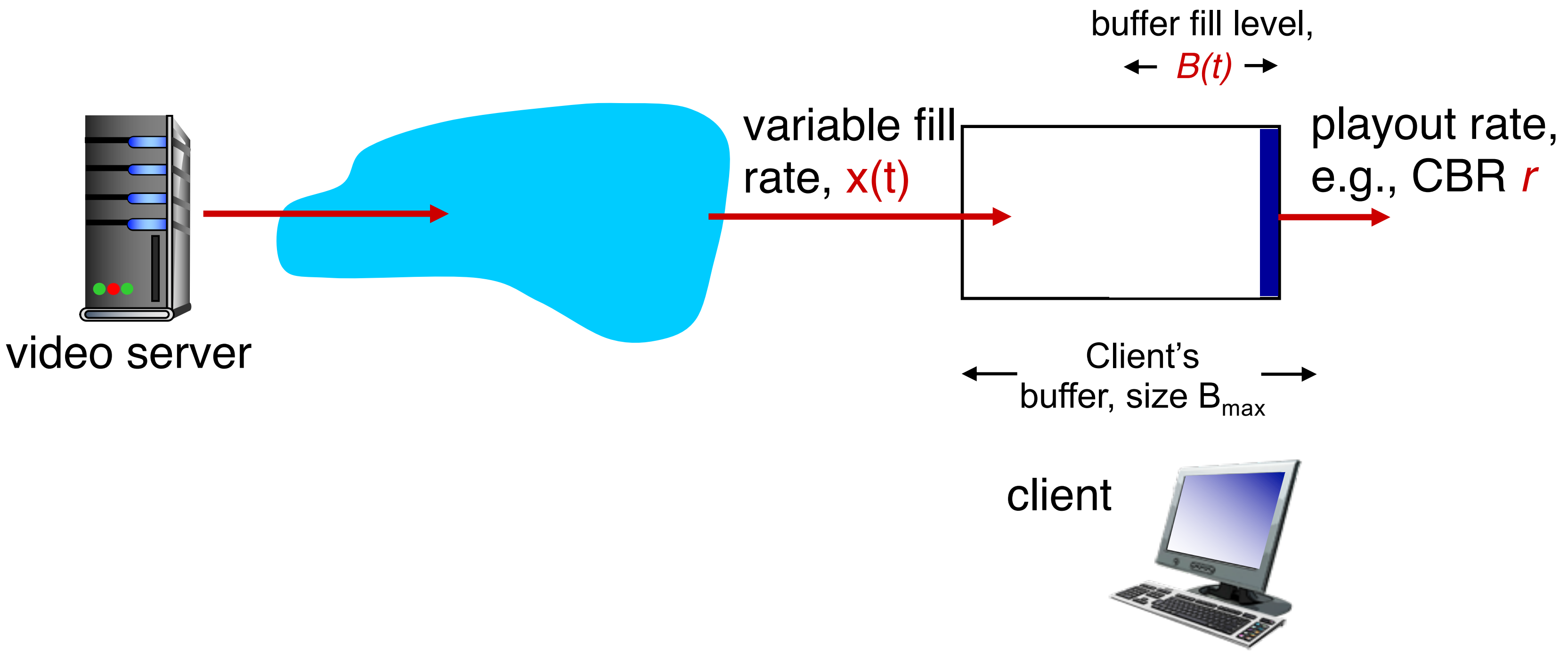
There's nothing in the buffer to show to the user

Client-side buffering, playout



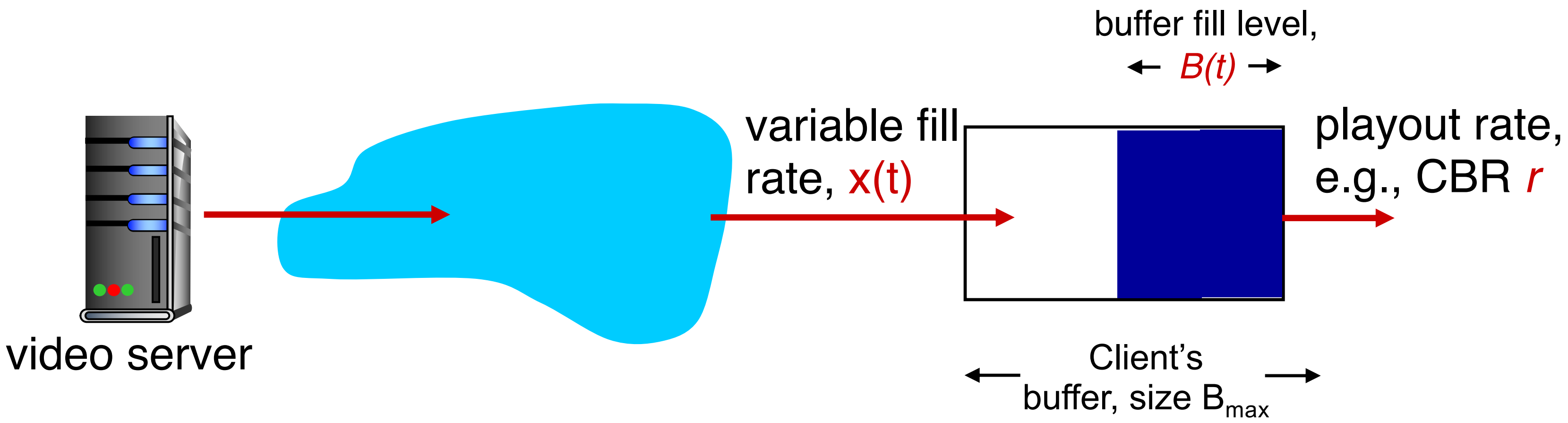
Most video is broken up in time into multiple segments
Client downloads video segment by segment
For example: a segment might be 4 seconds worth of video.

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)

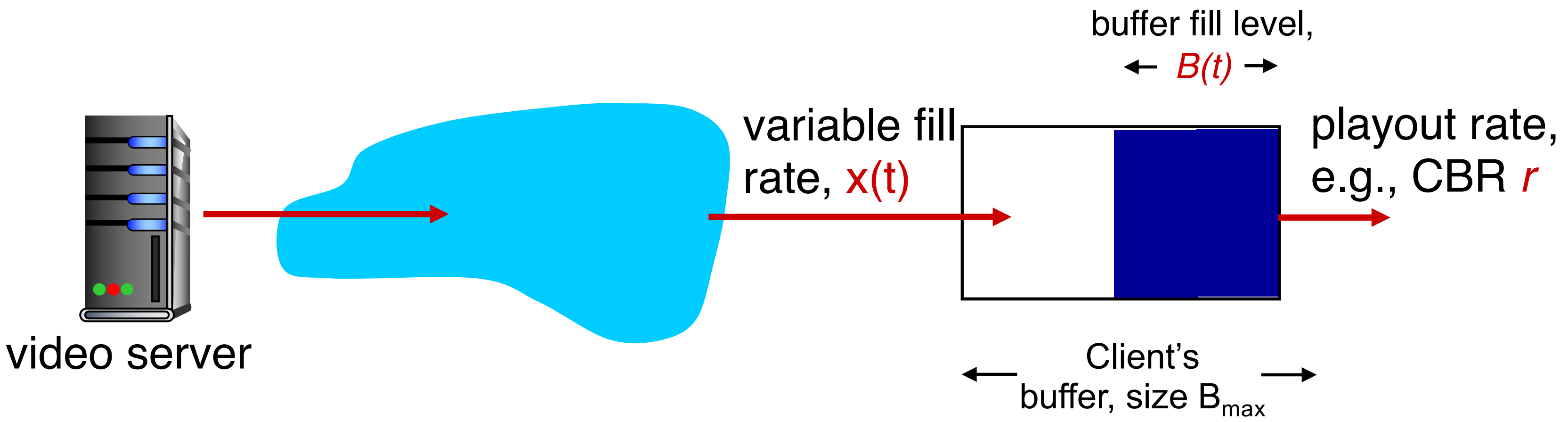
Client-side buffering, playout



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

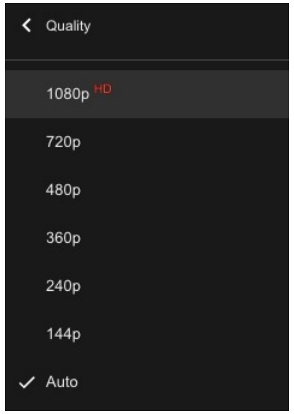


playout buffering: average fill rate ($\bar{x}$), playout rate (r):

- is $\bar{x} < r$ or $\bar{x} > r$ 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

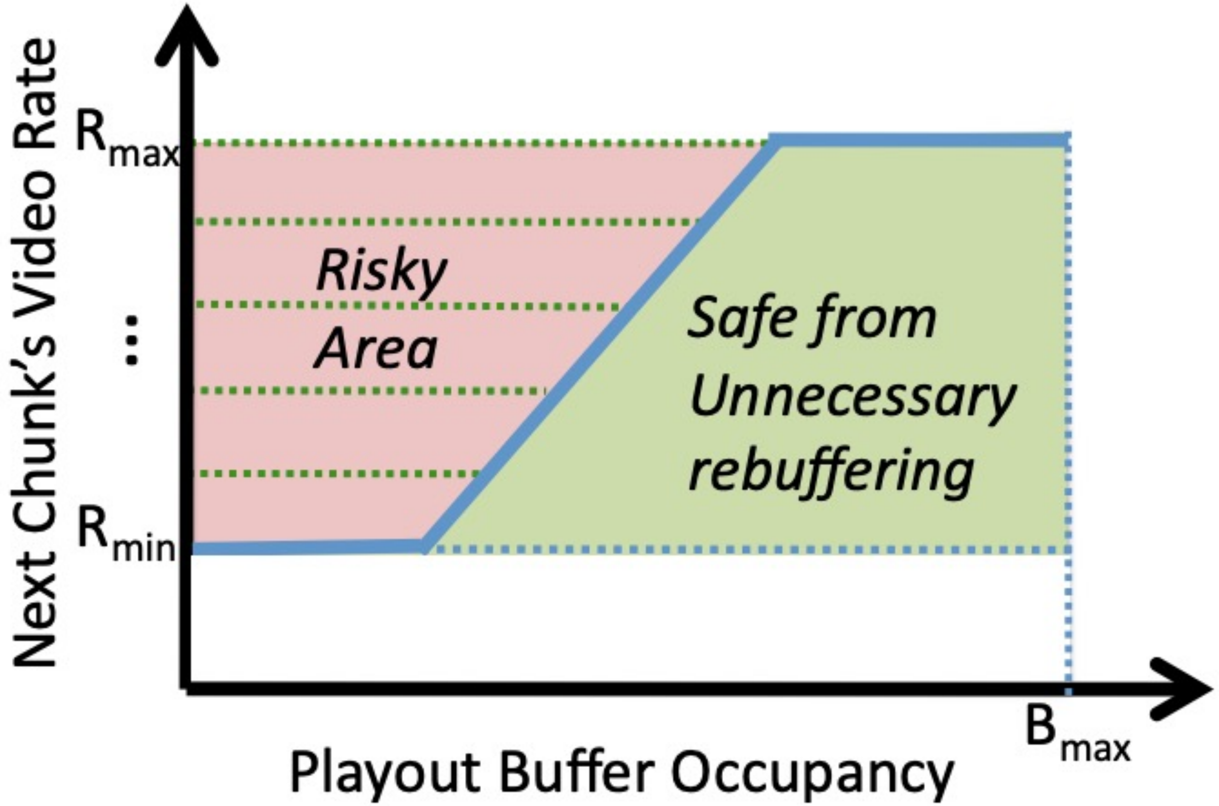


- Adapt bit rate per segment through collaboration between the video client (e.g., your browser) and the server (e.g., @ Netflix)
- Adaptive bit-rate (ABR) video: change the bit-rate (quality) of next video segment based on network and client conditions
- A typical strategy: Buffer-based rate adaptation

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