Multimedia: Streaming Video & Audio

CS 352, Lecture 22, Spring 2020

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

Srinivas Narayana

RUTGERS
UNIVERSITY | NEW BRUNSWICK

Course announcements

- Quiz 8 (the last one!) will go online later today
  - Due Tuesday at 10 PM on Sakai
- Project 3 due next Friday

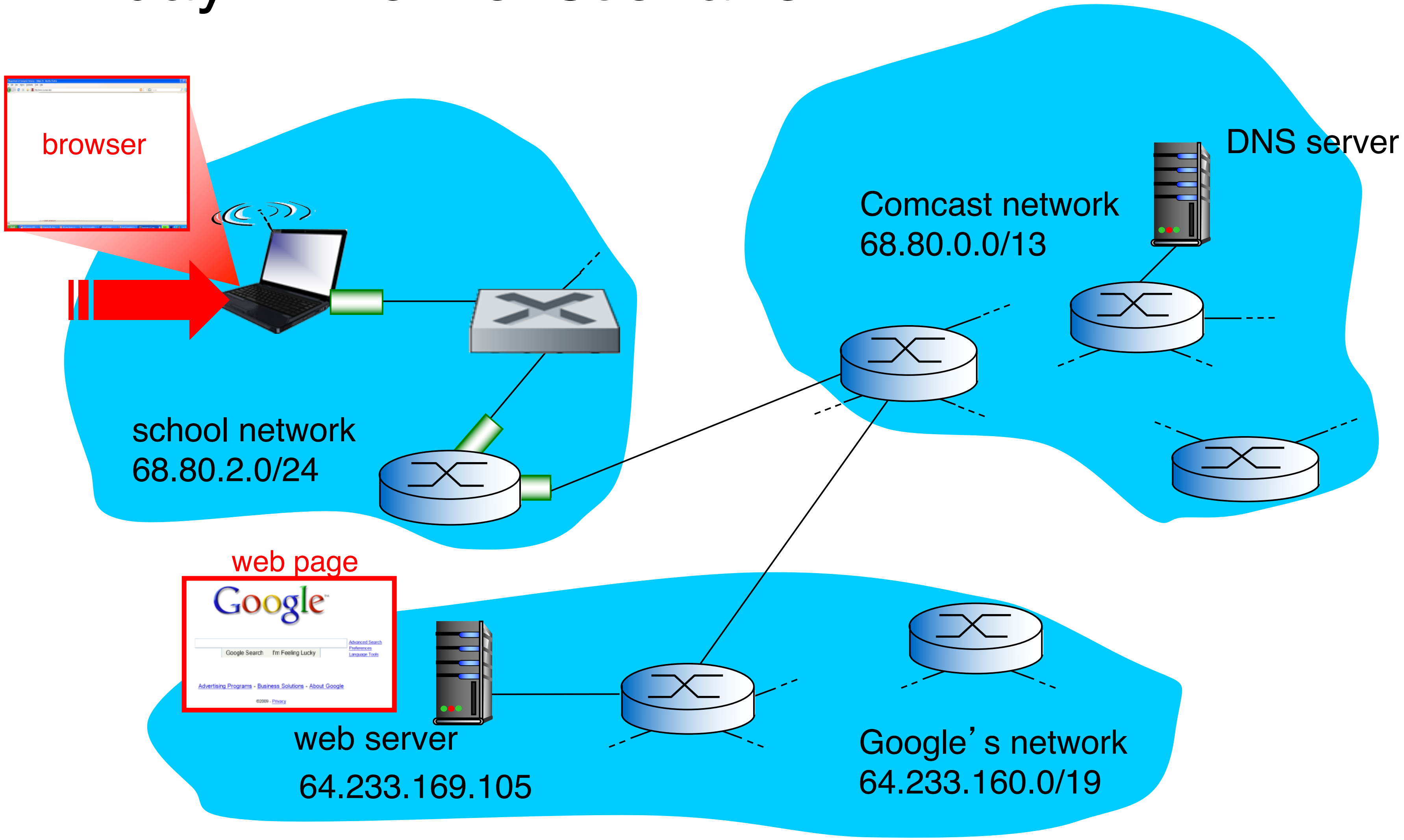
Synthesis of protocols

Synthesis: a day in the life of a web request

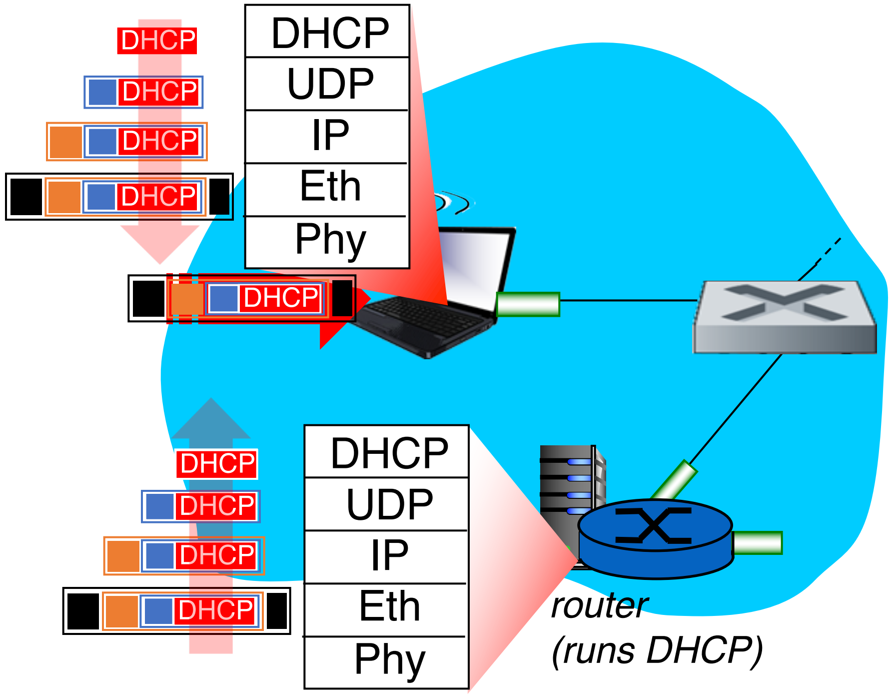
- Our journey down protocol stack complete!
  - application, transport, network, link

- putting-it-all-together: synthesis!
  - *goal:* identify, review, understand protocols (at all layers) involved in seemingly simple scenario: requesting www page
  - *scenario:* student attaches laptop to campus network, requests/receives www.google.com

A day in the life: scenario



A day in the life… connecting to the Internet



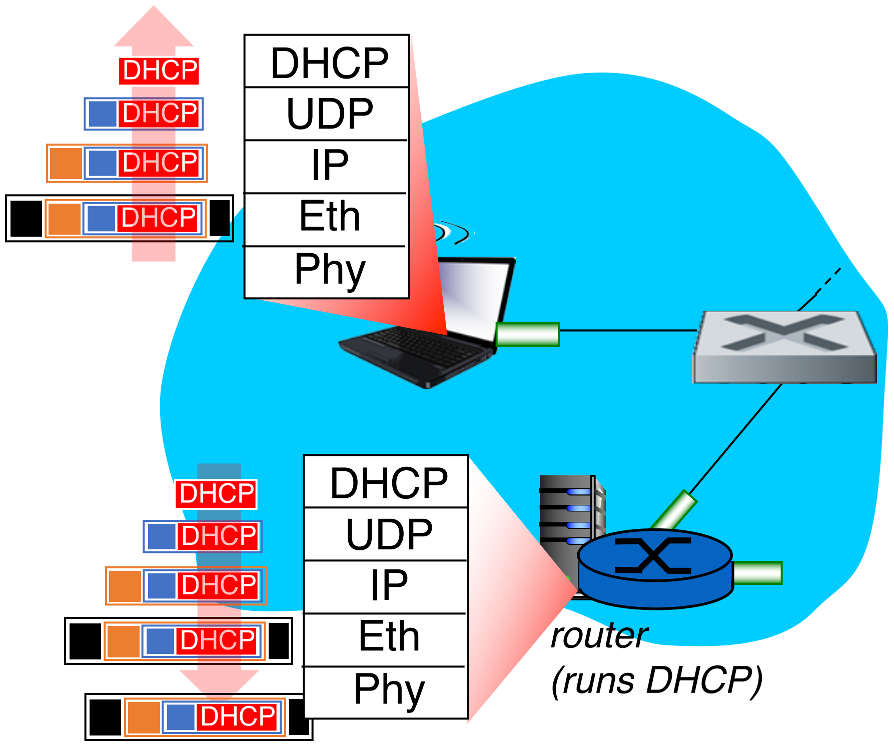
- connecting laptop needs to get its own IP address, addr of first-hop router, addr of DNS server: use *DHCP*

- DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in 802.3 Ethernet

- Ethernet frame broadcast (dest: FFFFFFFFFFFF) on LAN, received at router running DHCP server

- Ethernet demuxed to IP demuxed, UDP demuxed to DHCP

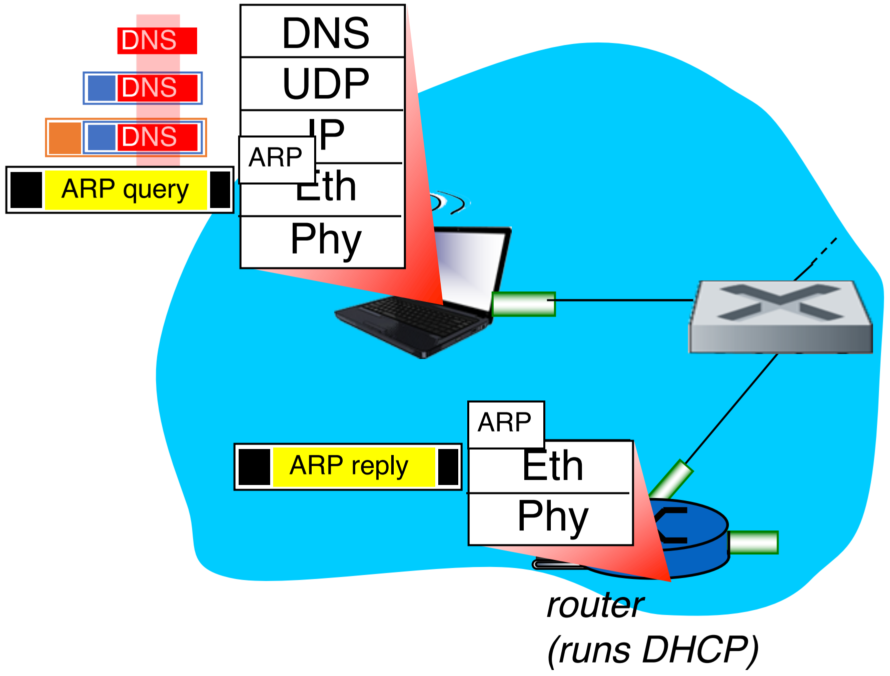
A day in the life… connecting to the Internet



- DHCP server formulates *DHCP ACK* containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- encapsulation at DHCP server, frame forwarded through LAN, demultiplexing at client
- DHCP client receives DHCP ACK reply

Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router

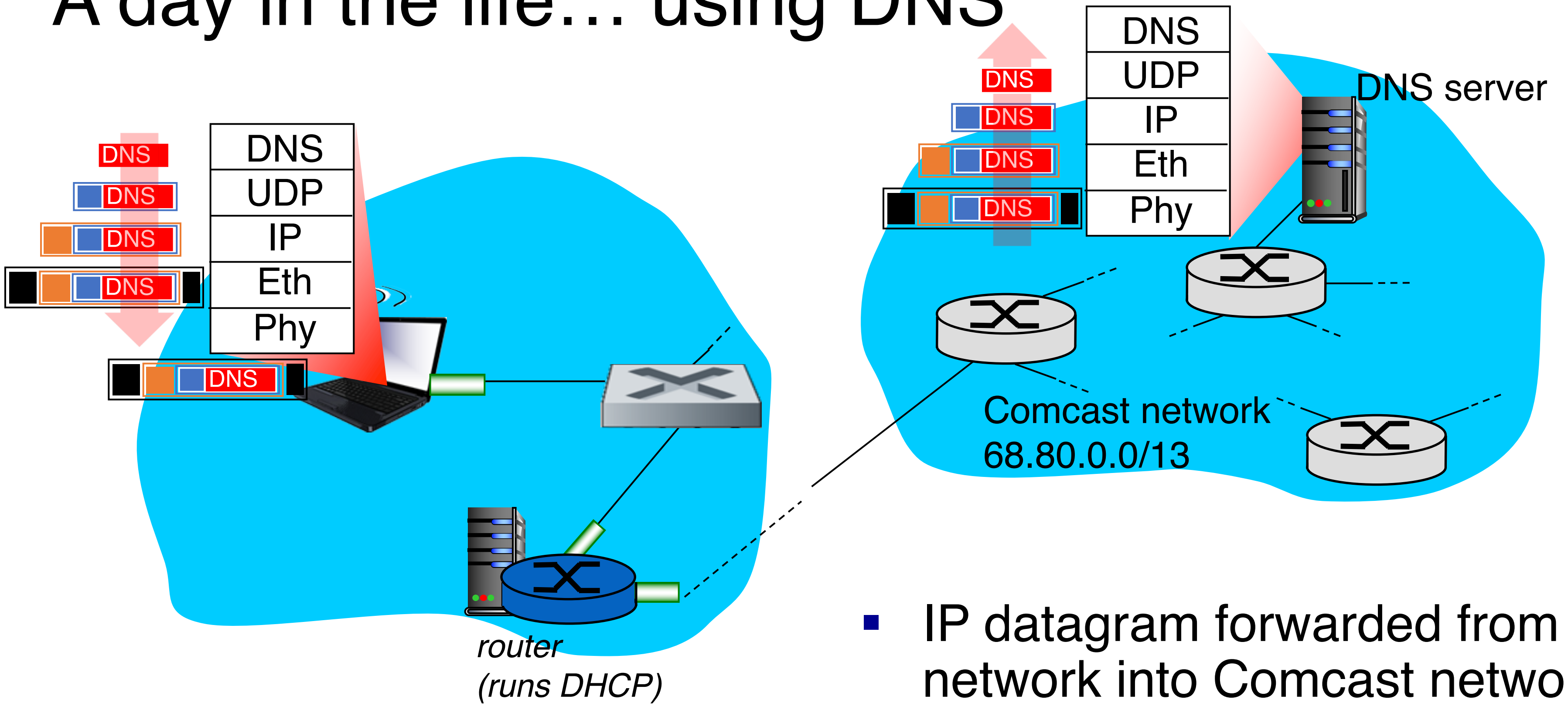
A day in the life… ARP (before DNS, before HTTP)



- before sending *HTTP* request, need IP address of www.google.com: *DNS*

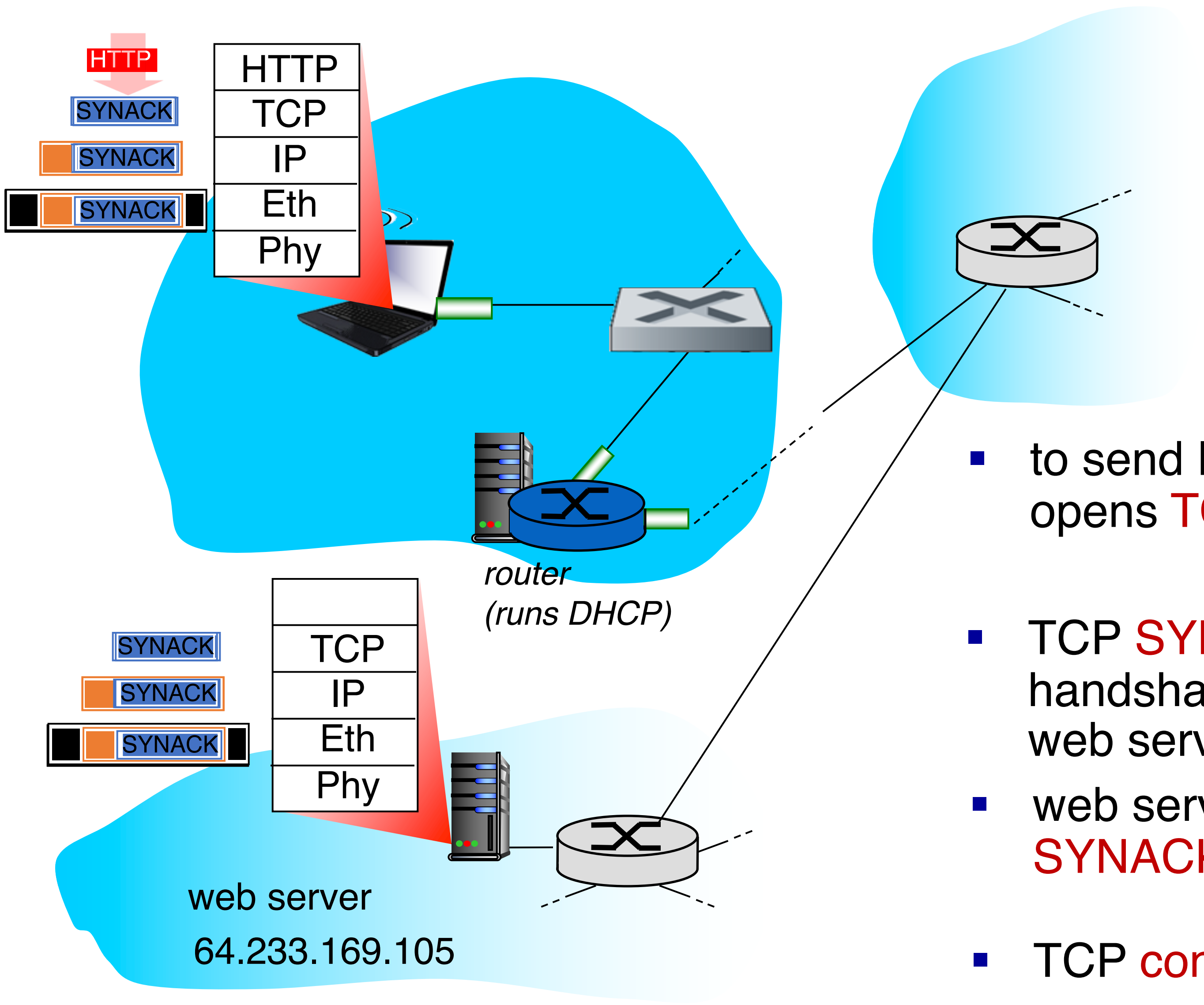
- DNS query created, encapsulated in UDP, encapsulated in IP, encapsulated in Eth. To send frame to router, need MAC address of router interface: ARP
- ARP query broadcast, received by router, which replies with ARP reply giving MAC address of router interface
- client now knows MAC address of first hop router, so can now send frame containing DNS query

A day in the life… using DNS



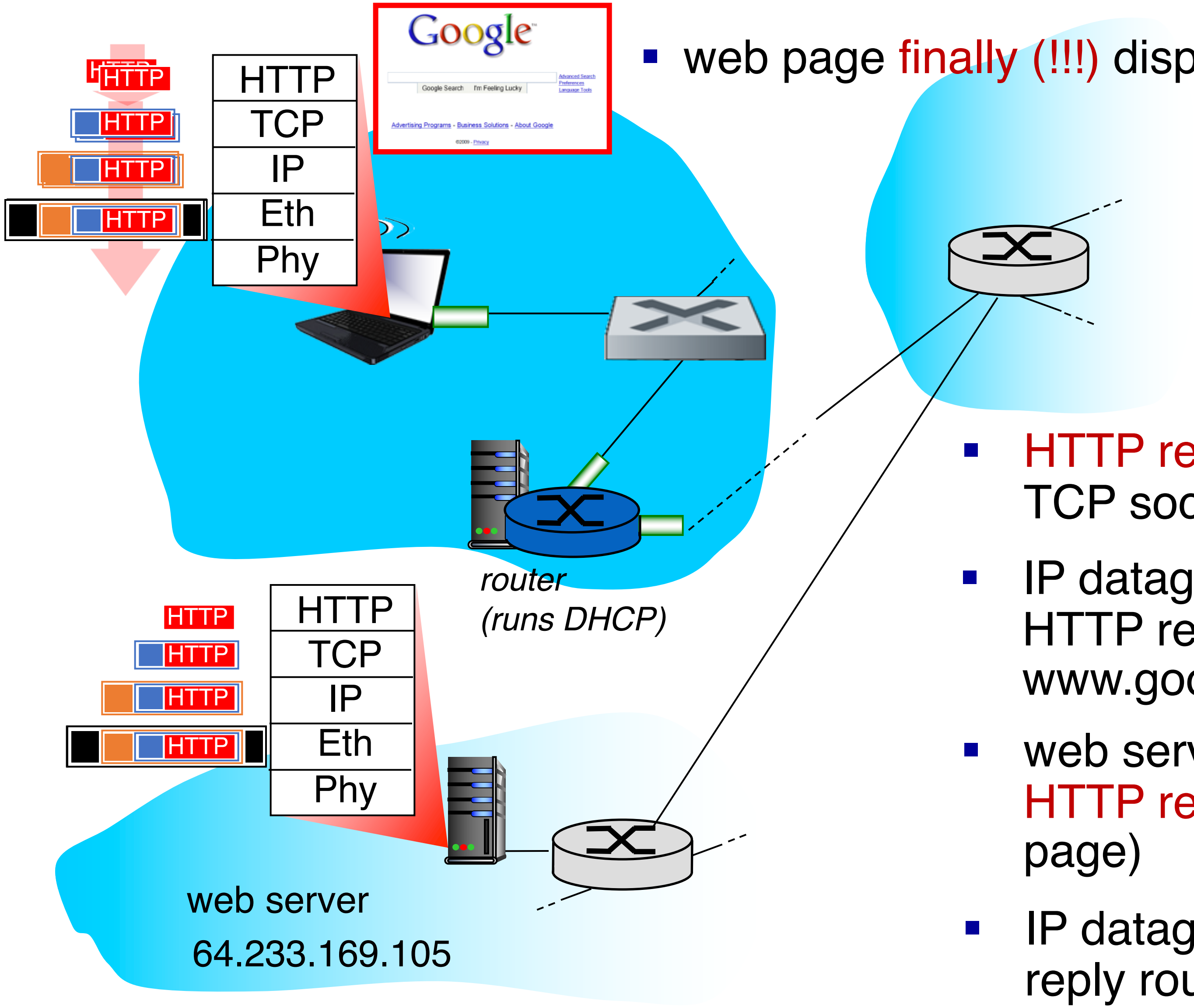
- IP datagram containing DNS query forwarded via LAN switch from client to 1st hop router
- IP datagram forwarded from campus network into Comcast network, routed (tables created by RIP, OSPF, IS-IS and/or BGP routing protocols) to DNS server
- demuxed to DNS server
- DNS server replies to client with IP address of www.google.com

A day in the life…TCP connection carrying HTTP



- to send HTTP request, client first opens TCP socket to web server
- TCP SYN segment (step 1 in 3-way handshake) inter-domain routed to web server
- web server responds with TCP SYNACK (step 2 in 3-way handshake)
- TCP connection established!

A day in the life… HTTP request/reply



- web page finally (!!!) displayed
- HTTP request sent into TCP socket
- IP datagram containing HTTP request routed to www.google.com
- web server responds with HTTP reply (containing web page)
- IP datagram containing HTTP reply routed back to client

Multimedia Networking

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
- Internet video surveillance traffic 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

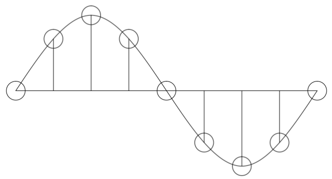
Source: Cisco visual networking index 2017--22

What's different about these applications?

- Traditional applications (HTTP(S), SMTP)
  - Delay tolerant but not loss tolerant
  - Data used *after* transfer complete
- But multimedia applications are often "real time"
  - Data delivery time *during transfer* has implications
- 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

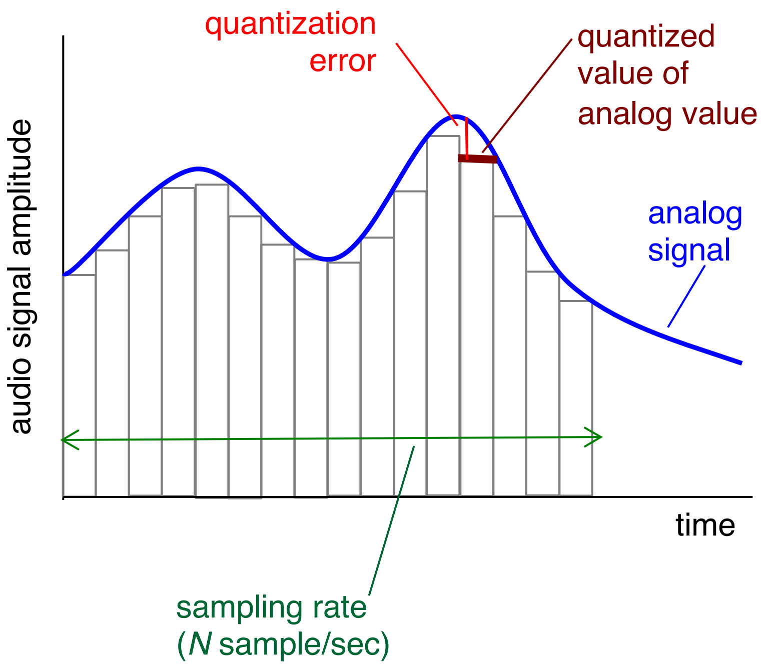
Digital representation of audio



- 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 & more bandwidth to transmit
- Compress
  - Compact representation of quantized values

Audio representation

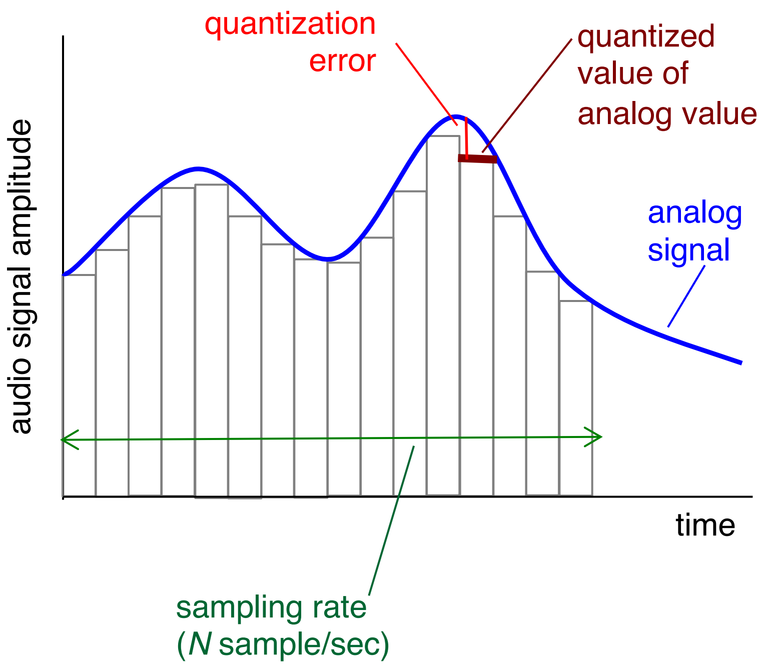
- analog audio signal sampled at constant rate
  - telephone: 8,000 samples/sec
  - CD music: 44,100 samples/sec
- each sample quantized, i.e., rounded
  - e.g., 2^8=256 possible quantized values
  - each quantized value represented by bits, e.g., 8 bits for 256 values



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

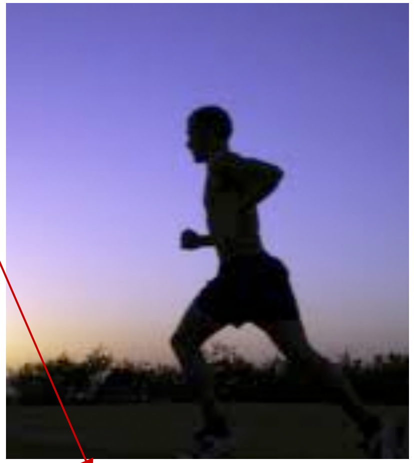
Video representation

- Video: sequence of images displayed at constant rate
  - e.g., 24 images/sec
- Digital image: array of pixels
  - each pixel represented by bits
- 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*

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

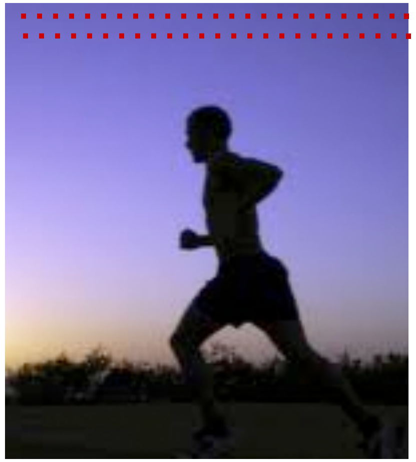


frame *i+1*

Video representation

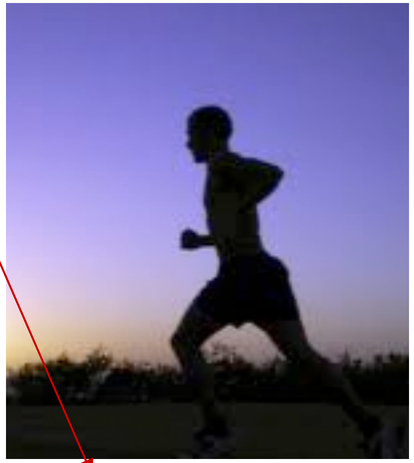
- Video *bit rate*: effective bits per second of the video after encoding
- CBR: (constant bit rate): video encoding rate fixed
- VBR: (variable bit rate): video encoding rate changes as amount of spatial, temporal coding changes
- examples:
  - MPEG 1 (CD-ROM) 1.5 Mbps
  - MPEG2 (DVD) 3-6 Mbps
  - MPEG4 (often used in Internet, < 1 Mbps)

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



frame *i+1*

Multimedia networking: 3 application types

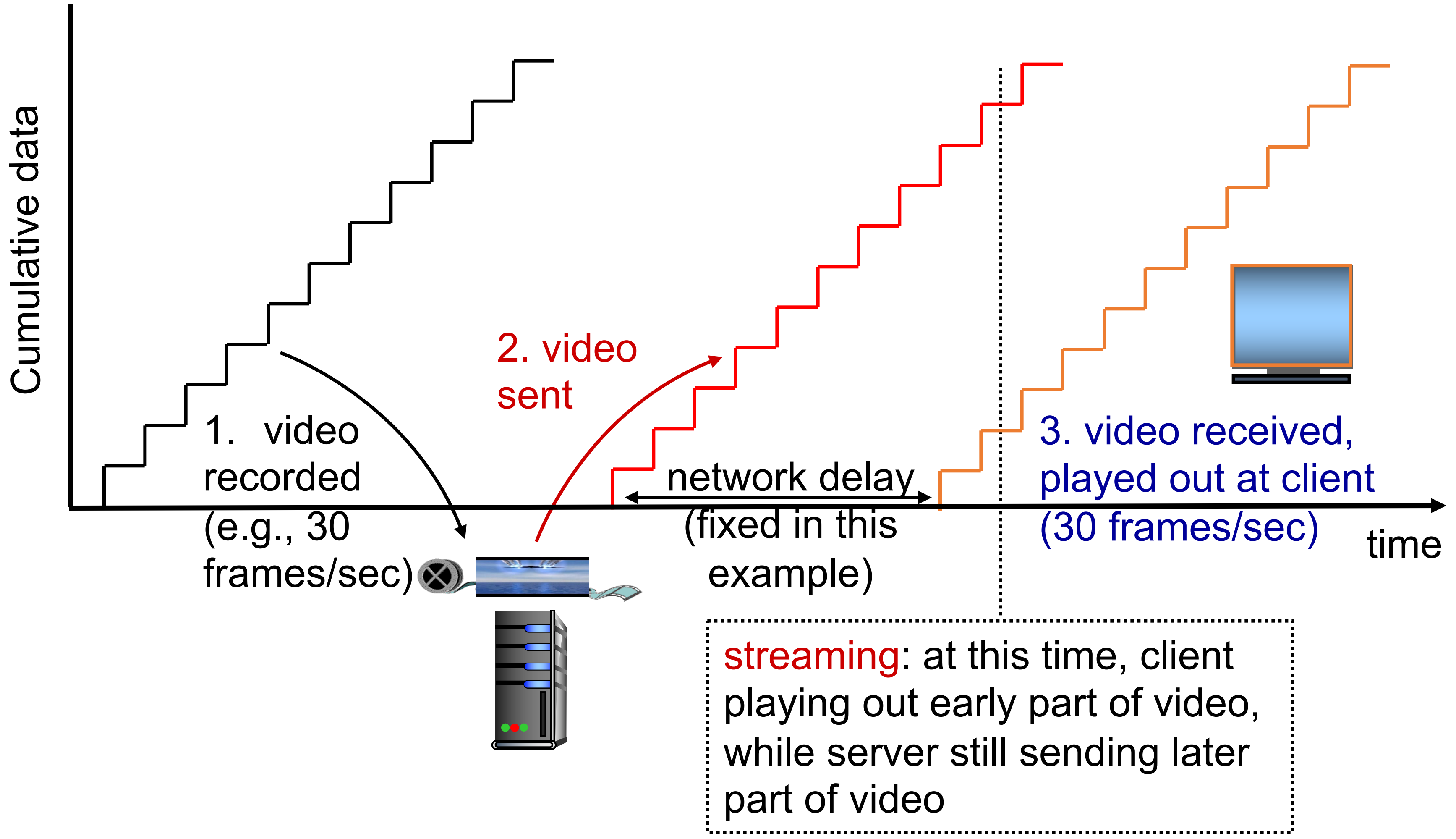
- *streaming, stored* audio, video
  - *streaming:* can begin playout before downloading entire file
  - *stored (at server):* can transmit faster than audio/video will be rendered (implies storing/buffering at client)
  - e.g., YouTube, AmazonPrime, Disney, Netflix, Hulu
- *conversational* voice/video over IP
  - interactive nature of human-to-human conversation limits delay tolerance
  - e.g., Skype
- *streaming live* audio, video
  - e.g., live sporting event

Streaming video

Streaming stored content

- Media is prerecorded
- Client downloads an initial portion and starts viewing
- Rest downloaded as time progresses
- No need to wait for entire content to be downloaded
- Can change content sites mid-stream based on network conditions

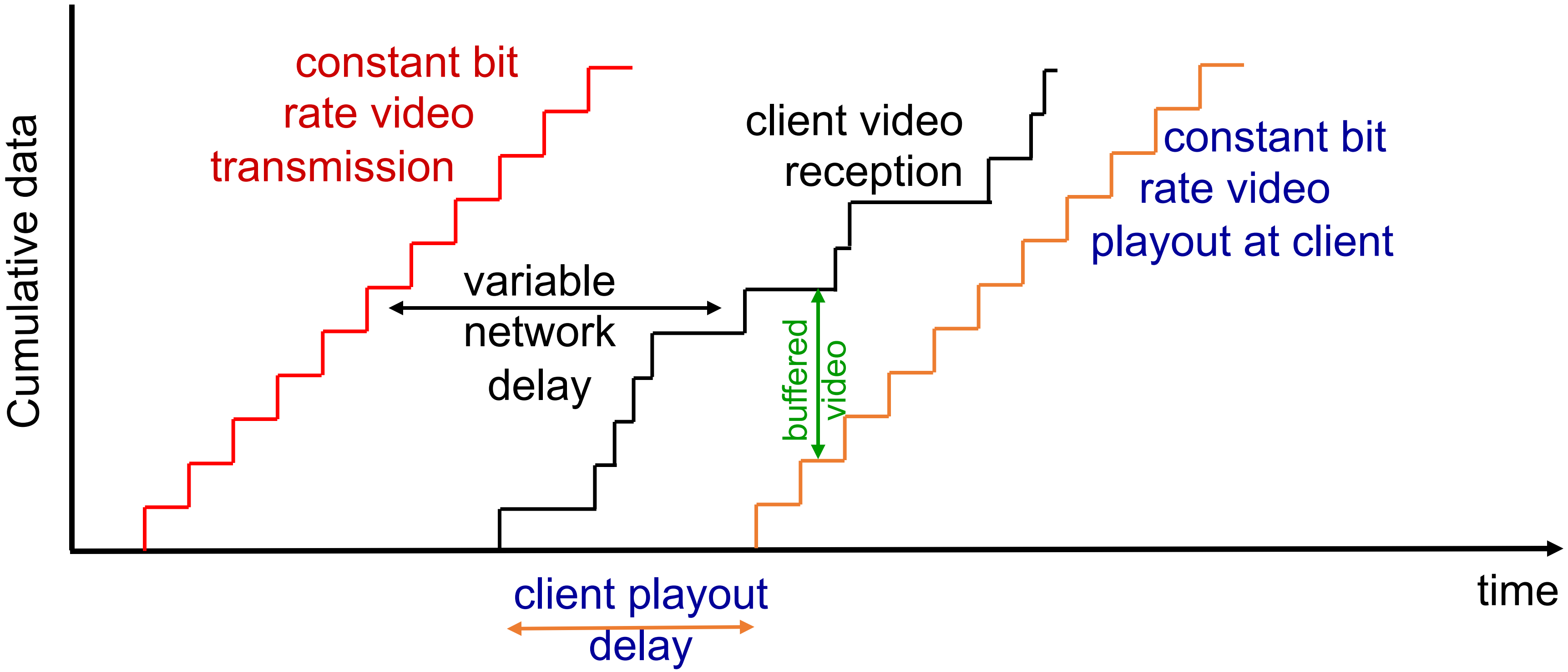
Streaming stored video:



Streaming stored video: challenges

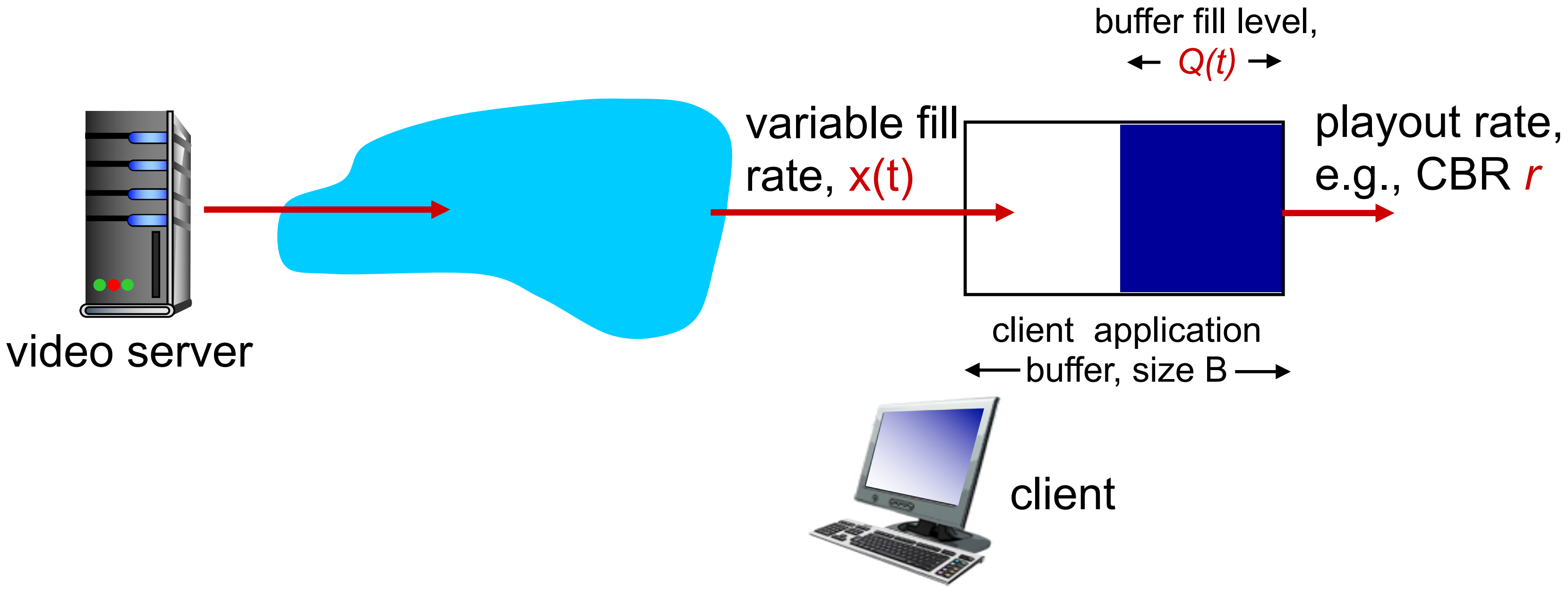
- continuous playout constraint: once client playout begins, playback must match original timing
  - … but network delays are variable (jitter), so will need client-side buffer to match playout requirements
- other challenges:
  - client interactivity: pause, fast-forward, rewind, jump through video
  - video packets may be lost, retransmitted

Streaming stored video: revisited

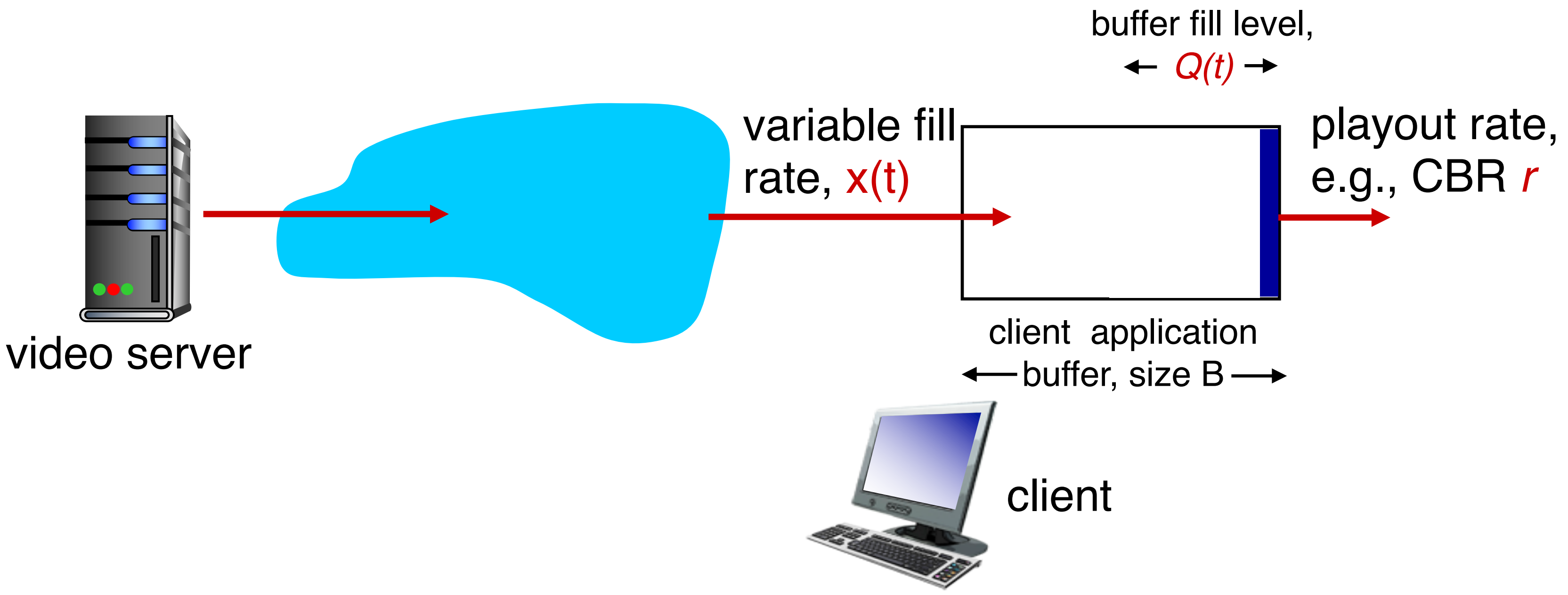


- *client-side buffering and playout delay:* compensate for network-added delay, delay jitter

Client-side buffering, playout

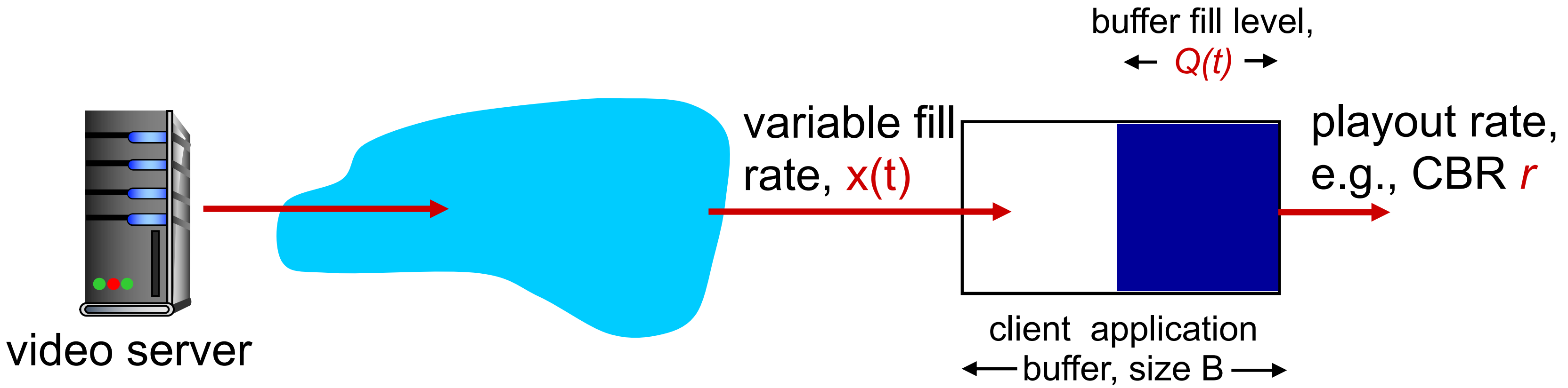


Client-side buffering, playout



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 and playout rate r is constant

Client-side buffering, playout



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

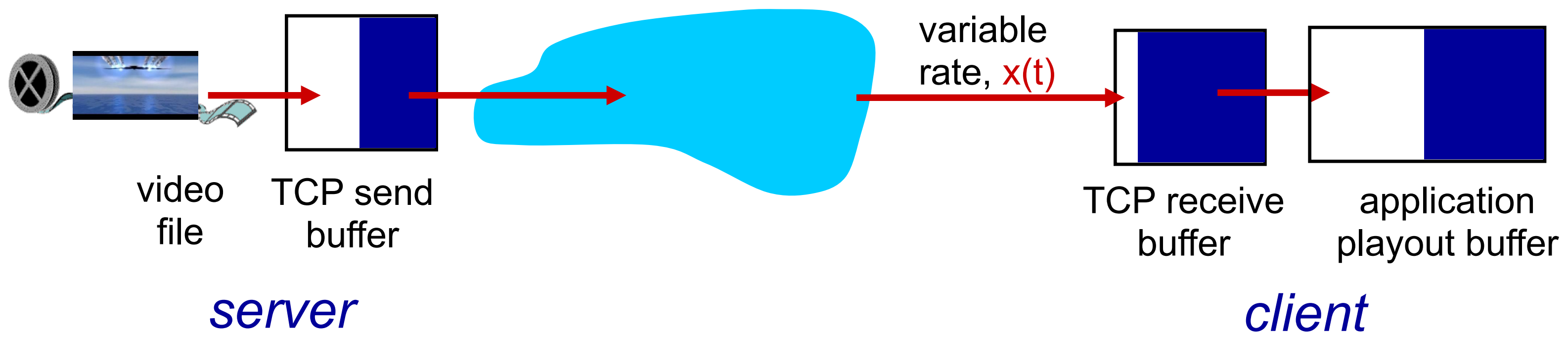
- $\bar{x} < r$: buffer eventually empties (causing freezing of video playout until buffer again fills)
- $\bar{x} > r$: buffer will not empty, provided initial playout delay is large enough to absorb variability in x(t)
  - *initial playout delay tradeoff:* buffer starvation less likely with larger delay, but larger delay until user begins watching

Streaming multimedia: UDP

- server sends at rate appropriate for client
  - often: send rate = encoding rate = constant rate
  - transmission rate can be oblivious to congestion levels
- short playout delay (2-5 seconds) to remove network jitter
- error recovery: application-level, time permitting
- RTP [RFC 2326]: multimedia payload types
- UDP traffic may *not* get through firewalls

Streaming multimedia: HTTP/TCP

- multimedia file retrieved via HTTP GET
- send at maximum possible rate under TCP



- fill rate fluctuates due to TCP congestion control, retransmissions (in-order delivery)
- larger playout delay: smooth TCP delivery rate
- HTTP/TCP passes more easily through firewalls

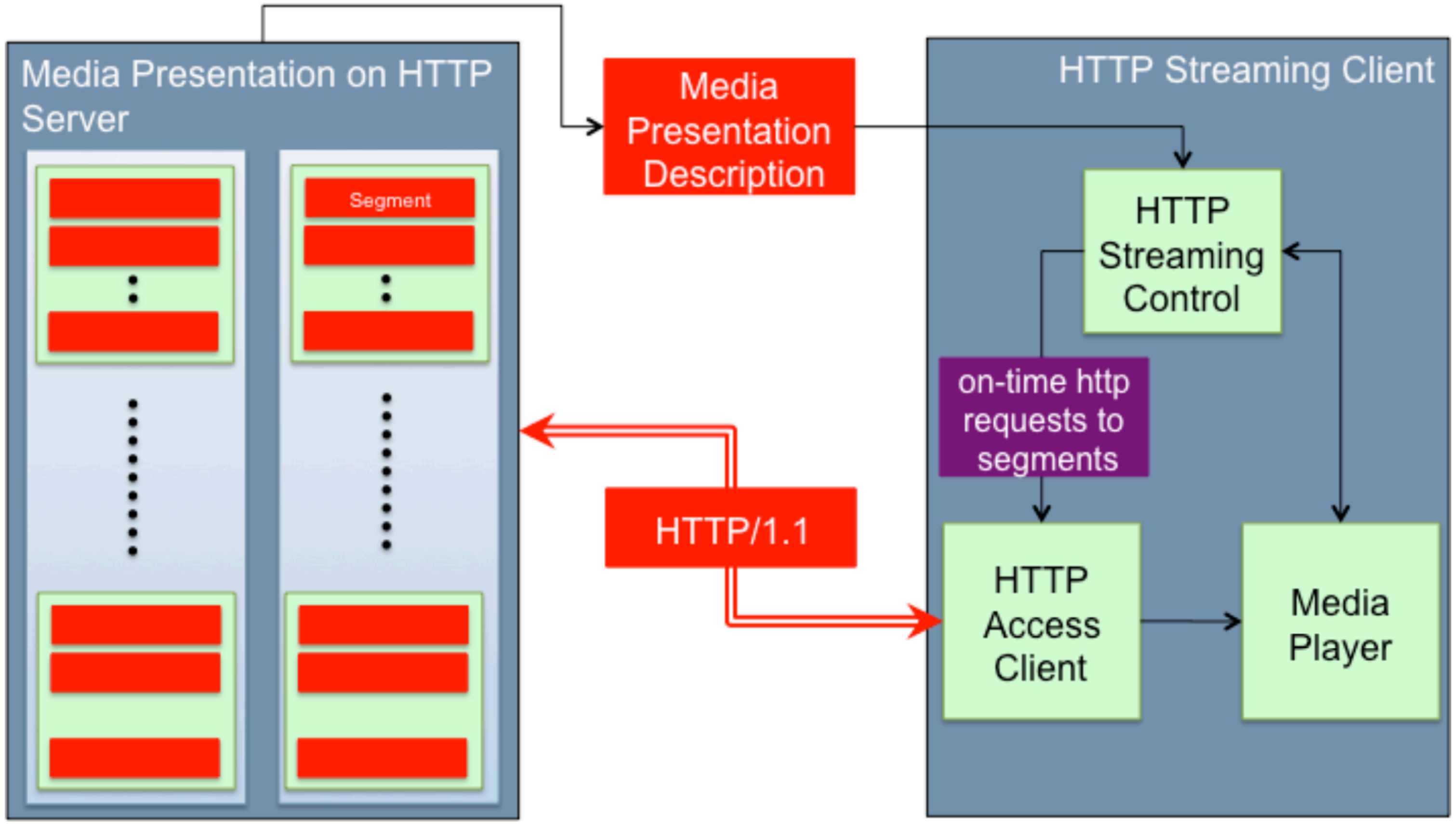
Streaming multimedia with DASH

- Dynamic Adaptive Streaming over HTTP
- Used by Netflix and other video streaming services
- Client-centric approach to video delivery
  - Adaptive: Client performs video bit rate adaptation
  - Dynamic: Can retrieve a single video from multiple sources
- Retain benefits of existing Internet and end host systems
- Server is standard HTTP server
  - Provides video/audio content in multiple formats and encodings
  - DASH allows the use of CDNs for data delivery

Dynamic Adaptive Streaming over HTTP (DASH)

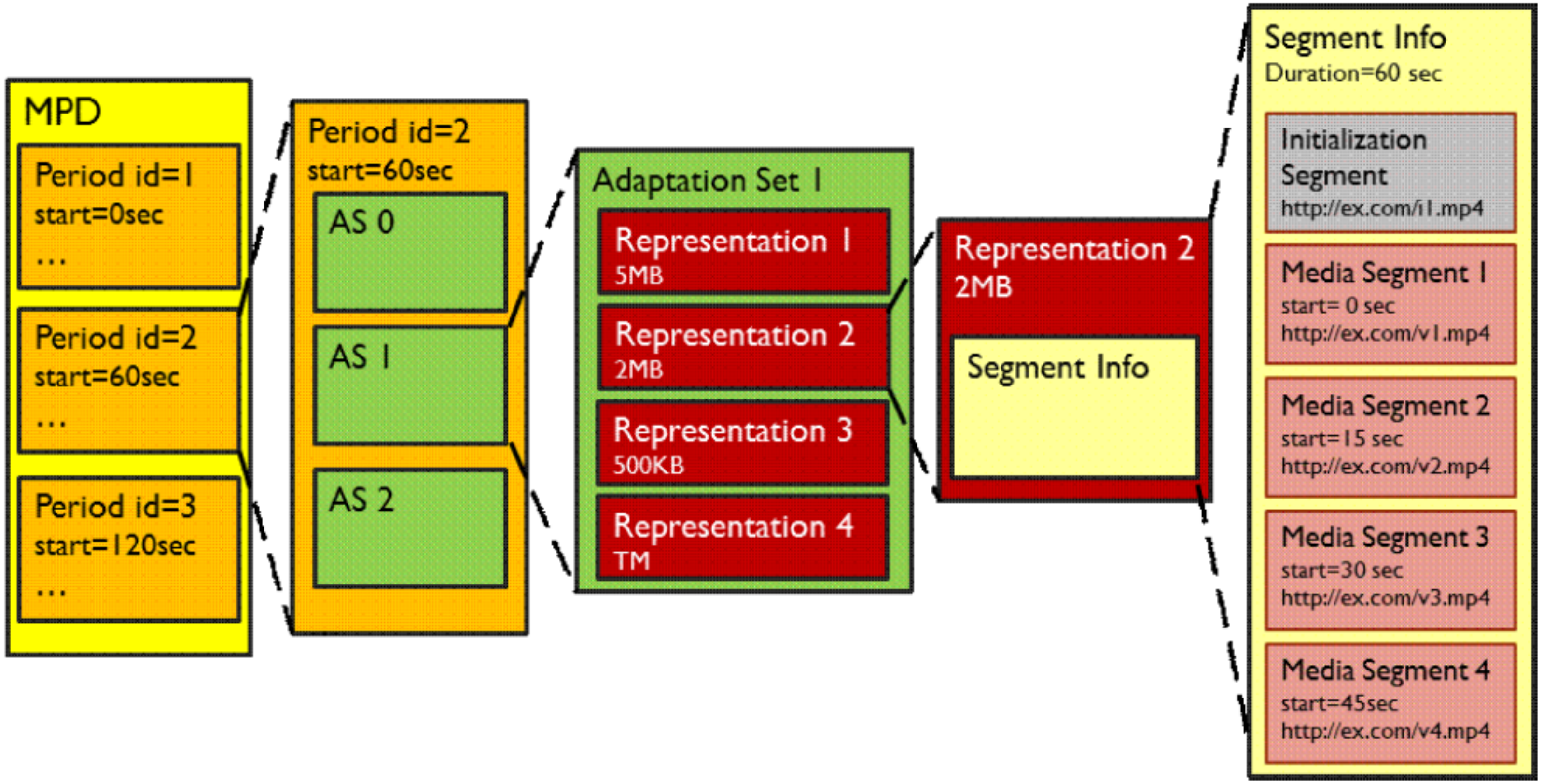
DASH: Key ideas

- Content chunks
- Each chunk can be independently retrieved
- Client-side algorithms to determine and request a varying bit rate for each chunk
  - Goal: ensure good quality of service



Source: Stockhammer MMSys 2011

DASH Data model



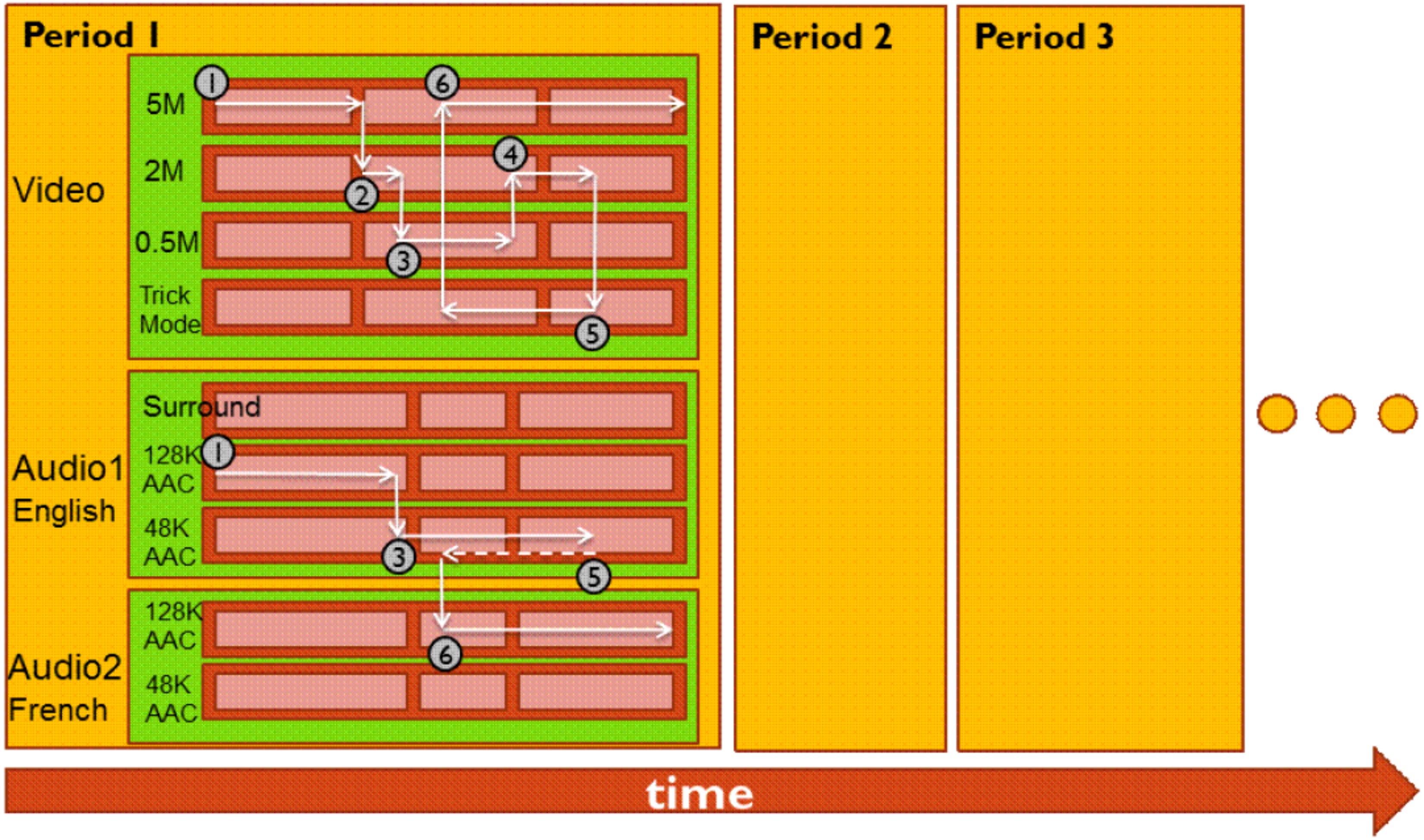
Media has several periods

Each period has several Adaptation Sets: Audio, video, close caption

Several Representations (ex: codecs, bit rates) per Adaptation set

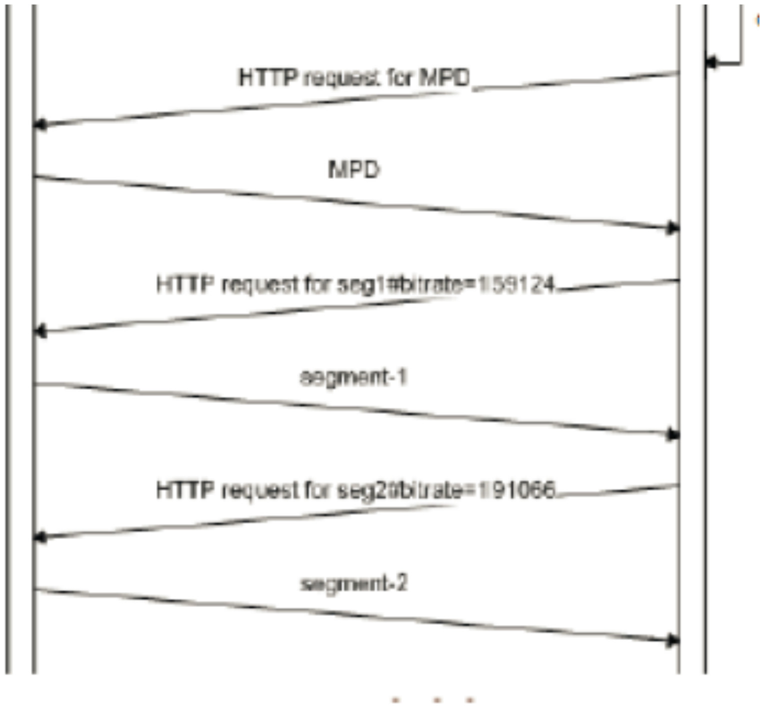
Several Chunks/Segments per Representation

Dynamic bit rate changing of streams

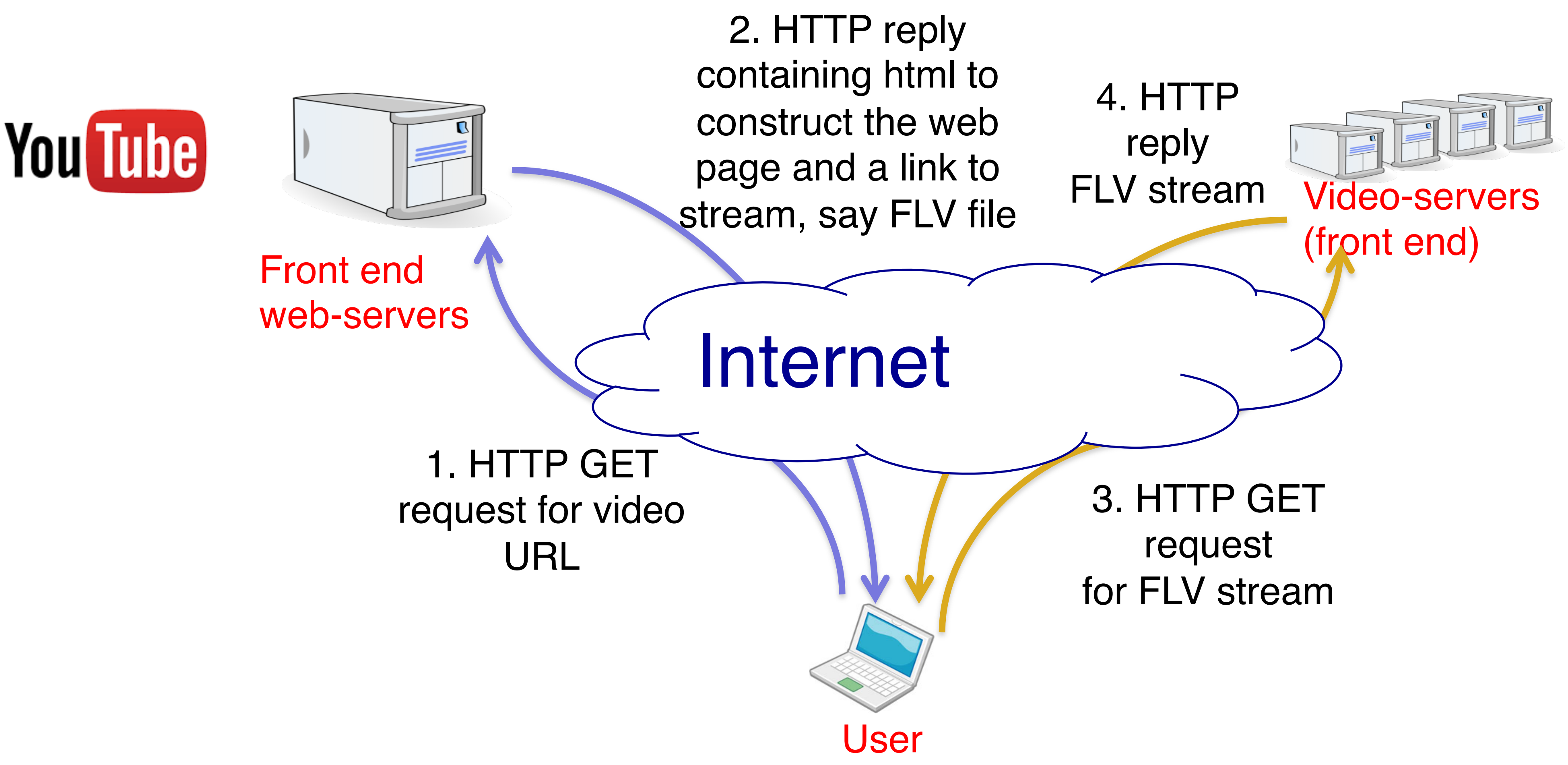


Media Presentation Descriptor (MPD)

- MPD requested over http
  - Also called “manifest”
- Describes all segments
- Timing information and byte ranges of chunks
- Client uses HTTP GET RANGE from a given AS + representation to ask a given bit rate
- Client could use a different URL for each AS + representation



Video Delivery using CDN



Server Selection

- File → server mapping done in at least three ways
- Dynamic DNS resolution
  - DNS returns different IP addresses for a given DNS name
- HTTP redirect
  - Use HTTP status code 3xx [with new URL]
  - Web browser does a GET from the new site
- IP anycast
  - Use BGP to announce the same IP address from different locations
  - Client reaches “nearest” location according to inter-domain routing

DASH Summary

- Widely used in video streaming services
- Allows independent requests per segment
  - Hence, independent segment quality and data sizes
  - Encoded through separate HTTP objects and corresponding HTTP byte ranges
  - Combined or separate audio & video streams
- Works well with CDNs
  - Independent representations or chunks can be queried from different locations if needed