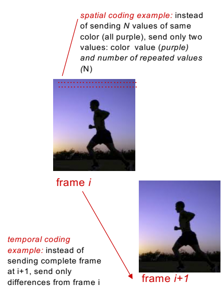
Video Streaming and CDNs: context

- stream video traffic: major consumer of Internet capacity
  - Netflix, YouTube, Amazon Prime: 80% of residential ISP traffic (2020)
- challenge: scale - how to reach ~1B users?
  - single mega-video server won't work (why?)
- challenge: heterogeneity
  - different users have different capabilities (e.g., wired versus mobile; network capacity rich versus capacity poor)
- *solution: distributed, application-level infrastructure*

27

Multimedia: video

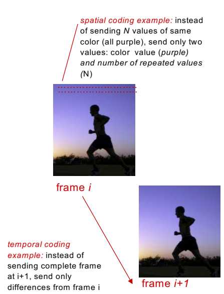
- video: sequence of images displayed at constant rate
  - e.g., 24 images/sec
- digital image: array of pixels
  - each pixel represented by bits
- coding: use redundancy *within* and *between* images to decrease # bits used to encode image
  - spatial (within image)
  - temporal (from one image to next)



28

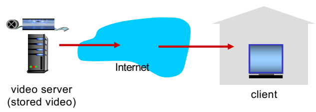
Multimedia: video

- 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, 64Kbps – 12 Mbps)
- We can create multiple versions of the same video at different quality



29

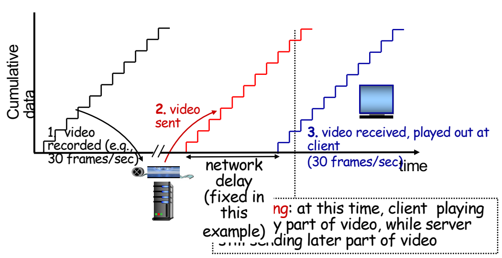
Streaming stored Video



Main challenges:

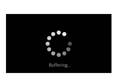
- server-to-client goodput will *vary* over time, with changing network congestion levels (in house, in access network, in network core, at video server)
- packet loss and delay due to congestion will delay playout, or result in poor video quality

Streaming stored video



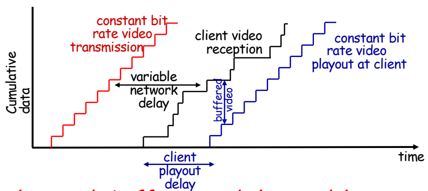
Streaming stored video: challenges

- 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

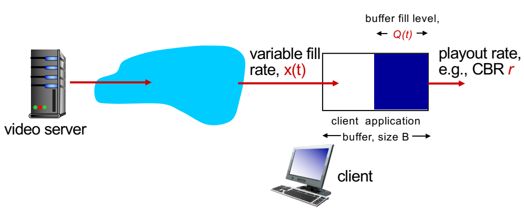
Streaming stored video: playout buffering



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

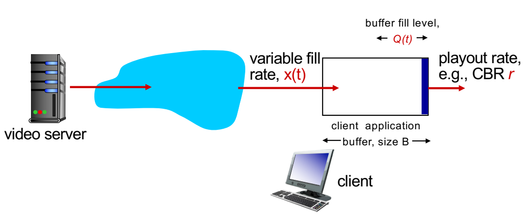
33

Client-side Buffering, Playout



34

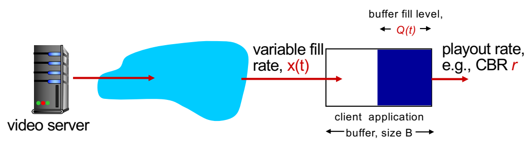
Client-side Buffering, Playout



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

35

Client-side Buffering, Playout



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

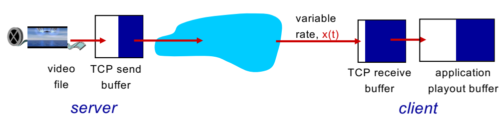
- $\bar{x} < r$: Buffer eventually empties (causing freezing of video playout until buffer again fills)
- $\bar{x} > r$: Buffer will not empty, provided initial playout delay is large enough to absorb variability in x(t)
  - *Initial playout delay tradeoff:* Buffer starvation less likely with larger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

Closing Remarks 36

36

Streaming Multimedia: HTTP

- Multimedia file retrieved via HTTP GET
- Send at maximum possible rate under TCP



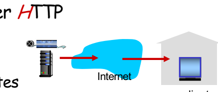
- Fill rate fluctuates due to TCP congestion control, retransmissions (in-order delivery)
- Larger playout delay: Smooth TCP delivery rate

Closing Remarks 37

37

Streaming multimedia: DASH

- *DASH: D*ynamic, *A*daptive *S*treaming over *H*TTP
- *server:*
  - divides video file into multiple chunks
  - each chunk stored, encoded at different rates
  - *manifest file:* provides URLs for different chunks



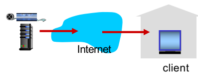
- *client:*
  - periodically measures server-to-client goodput
  - consulting manifest, requests one chunk at a time
    - chooses maximum coding rate sustainable given current goodput
    - can choose different coding rates at different points in time (depending on measured goodput at the time)

Closing Remarks 38

38

Streaming multimedia: DASH

- *"intelligence"* at client: client determines
  - *when* to request chunk (so that buffer starvation, or overflow does not occur)
  - *what encoding rate* to request (higher quality when more goodput can be achieved)
  - *where* to request chunk (can request from URL server that is "close" to client or has high achievable goodput)



Streaming video = encoding + DASH + playout buffering

Content distribution networks (CDNs)

- *challenge:* how to stream content (selected from millions of videos) to hundreds of thousands of *simultaneous* users?
- option 1: single, large "mega-server"
  - single point of failure
  - point of network congestion
  - long path to distant clients
  - multiple copies of video sent over outgoing link

....quite simply: this solution *doesn't scale*

Content distribution networks (CDNs)

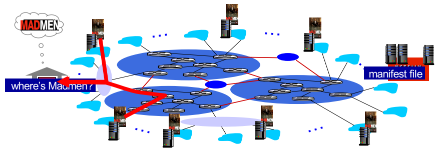
- *challenge:* how to stream content (selected from millions of videos) to hundreds of thousands of *simultaneous* users?
- option 2: store/serve multiple copies of videos at multiple geographically distributed sites *(CDN)*
  - *enter deep:* push CDN servers deep into many access networks
    - close to users
    - Akamai: 240,000 servers deployed in more than 120 countries (2015)
  - *bring home:* smaller number (10's) of larger clusters in POPs near (but not within) access networks
    - used by Limelight

Akamai

Limelight NETWORKS

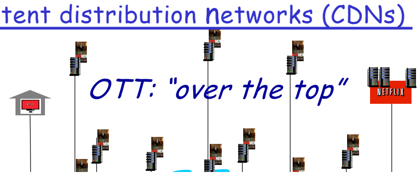
Content distribution networks (CDNs)

- CDN: stores copies of content at CDN nodes
  - e.g. Netflix stores copies of MadMen
- subscriber requests content from CDN
  - directed to nearby copy, retrieves content
  - may choose different copy if network path congested



42

Content distribution networks (CDNs)



OTT challenges: coping with a congested Internet

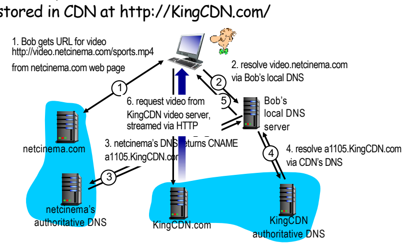
- from which CDN node to retrieve content?
- viewer behavior in presence of congestion?
- what content to place in which CDN node?

43

CDN content access: a closer look

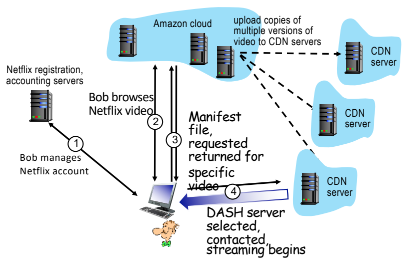
Bob (client) requests video http://netcinema.com/sports.mp4

- video stored in CDN at http://KingCDN.com/



44

Case study: Netflix



45

46

Sample Q1

AIMD

Given a TCP Reno connection's window oscillates between 8 and 16 segments at steady state (no flow control limitation):

- What's the time between two consecutive loss events, given RTT=100ms?

  9 RTTs * 100 ms = 900 ms

- What's the average sending rate for this connection?

  $$\frac{\frac{8+16}{2} = 12 \text{ segments}}{1 \text{ RTT} = 100\text{ms}} = 120 \text{ segments/sec.}$$

- What if another connection shares the link?

  $$\frac{120}{2} = 60 \text{ segments/sec.}$$

47