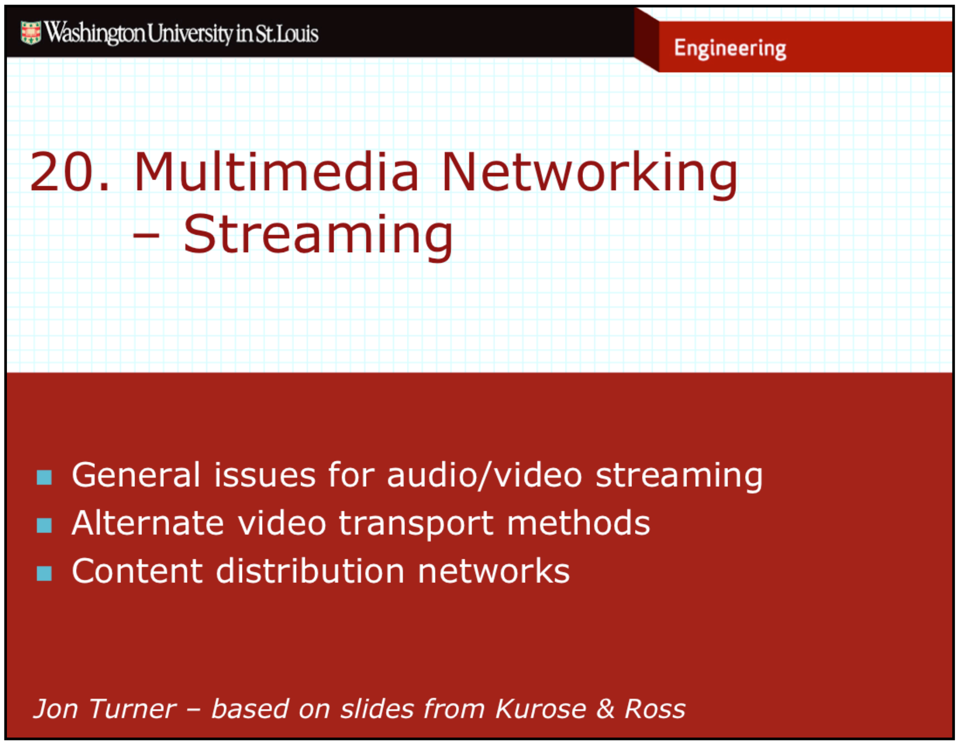


Washington University in St.Louis

Engineering

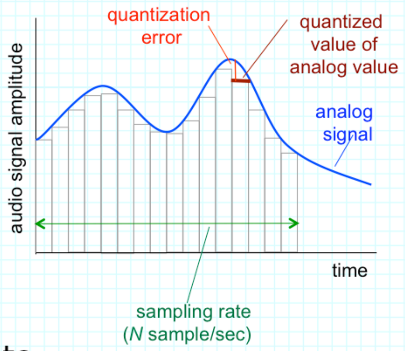
20. Multimedia Networking – Streaming

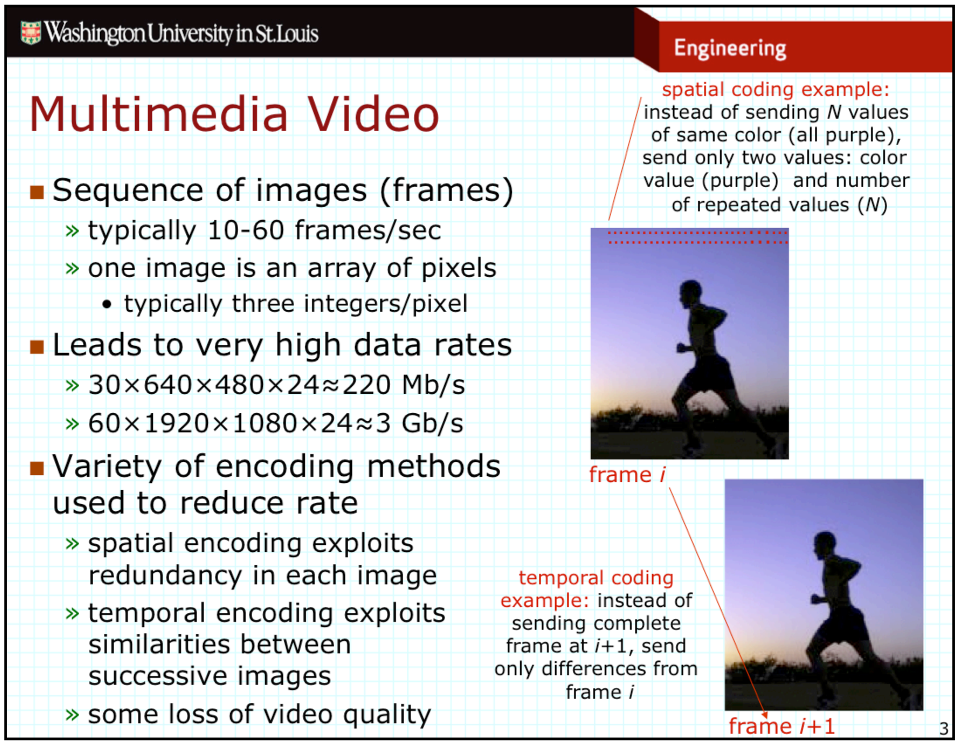
- General issues for audio/video streaming
- Alternate video transport methods
- Content distribution networks

Jon Turner – based on slides from Kurose & Ross

Multimedia Audio

- Analog audio sampled at constant rate
  - » telephony: 8,000 samples/sec
  - » CD audio: 44K samples/sec
- Each sample is represented by fixed precision integer
  - » integer values are sent in packets
- Receiver uses integer samples to reconstruct approximation of original audio signal
  - » integer representation leads to *quantization error*
- Variety of parameter choices
  - » traditional telephony: 8Kx8 leads to 64 Kb/s data rate
  - » CD: 44Kx16 yields 700 Kb/s (per channel)
  - » MP3 uses various rates: 96, 128, 160 Kb/s


quantization error
quantized value of analog value
analog signal
audio signal amplitude
time
sampling rate (*N* sample/sec)

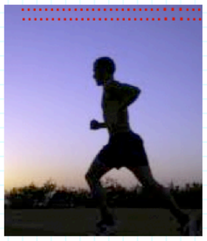


Washington University in St. Louis — Engineering

Multimedia Video

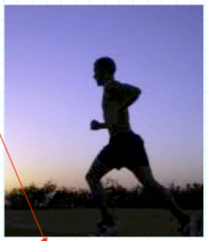
- Sequence of images (frames)
  - » typically 10-60 frames/sec
  - » one image is an array of pixels
    - typically three integers/pixel
- Leads to very high data rates
  - » 30×640×480×24≈220 Mb/s
  - » 60×1920×1080×24≈3 Gb/s
- Variety of encoding methods used to reduce rate
  - » spatial encoding exploits redundancy in each image
  - » temporal encoding exploits similarities between successive images
  - » some loss of video quality

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*)



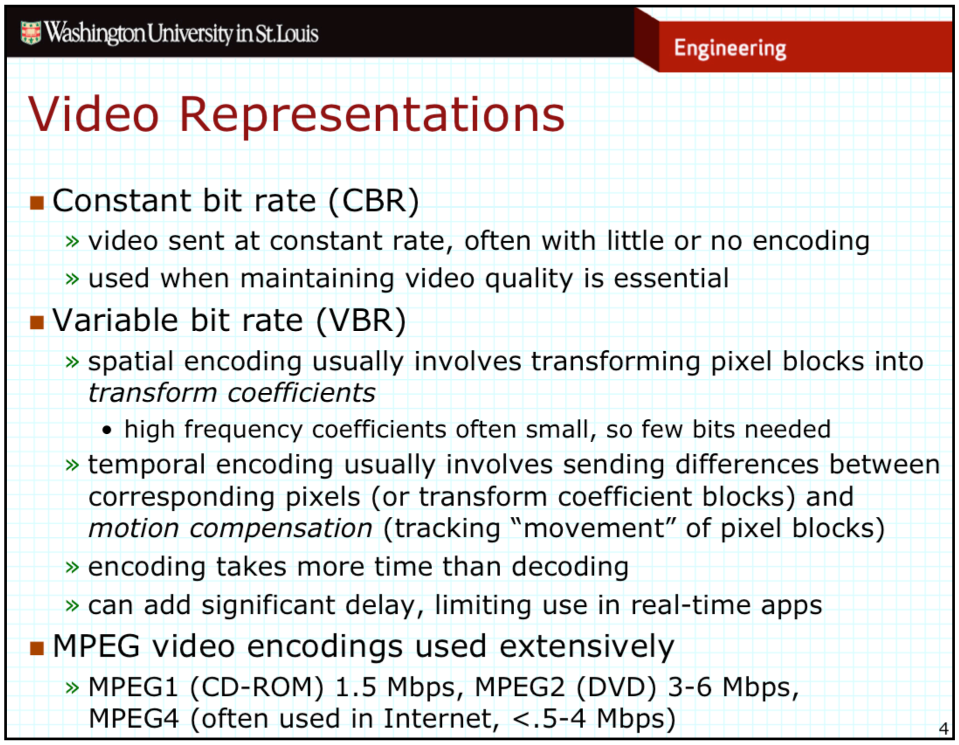
frame *i*

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



frame *i*+1

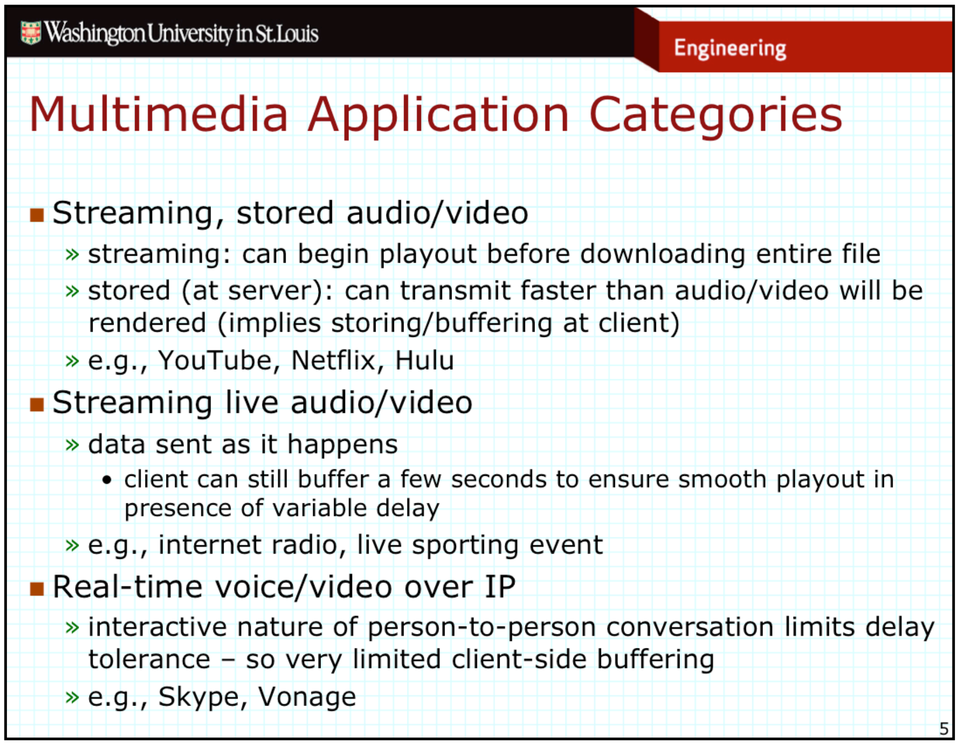
3



Video Representations

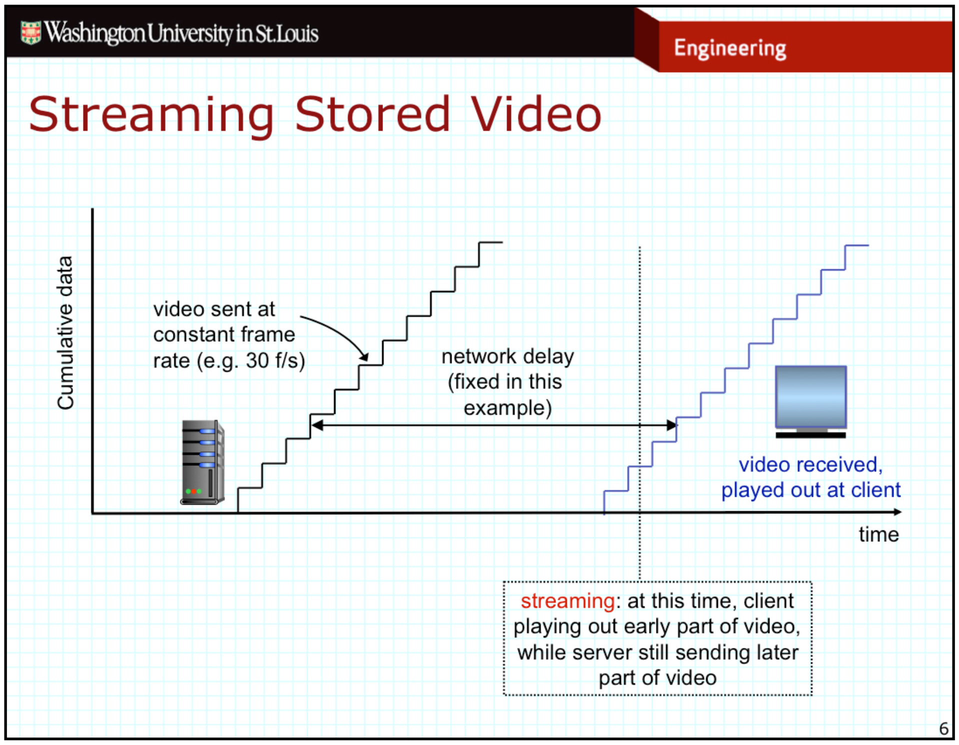
- Constant bit rate (CBR)
  - » video sent at constant rate, often with little or no encoding
  - » used when maintaining video quality is essential
- Variable bit rate (VBR)
  - » spatial encoding usually involves transforming pixel blocks into *transform coefficients*
    - high frequency coefficients often small, so few bits needed
  - » temporal encoding usually involves sending differences between corresponding pixels (or transform coefficient blocks) and *motion compensation* (tracking "movement" of pixel blocks)
  - » encoding takes more time than decoding
  - » can add significant delay, limiting use in real-time apps
- MPEG video encodings used extensively
  - » MPEG1 (CD-ROM) 1.5 Mbps, MPEG2 (DVD) 3-6 Mbps, MPEG4 (often used in Internet, <.5-4 Mbps)

4

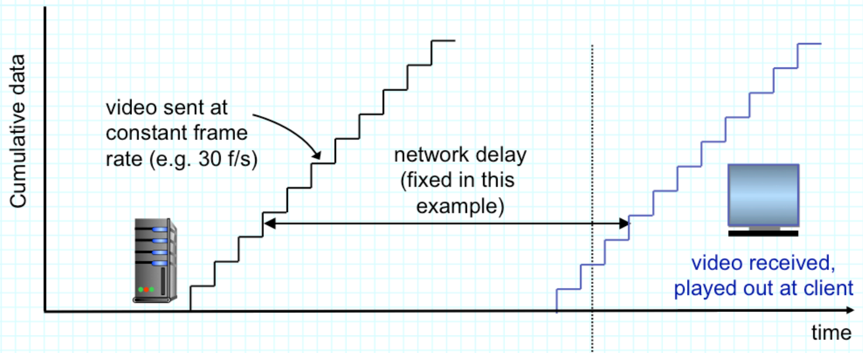


Multimedia Application Categories

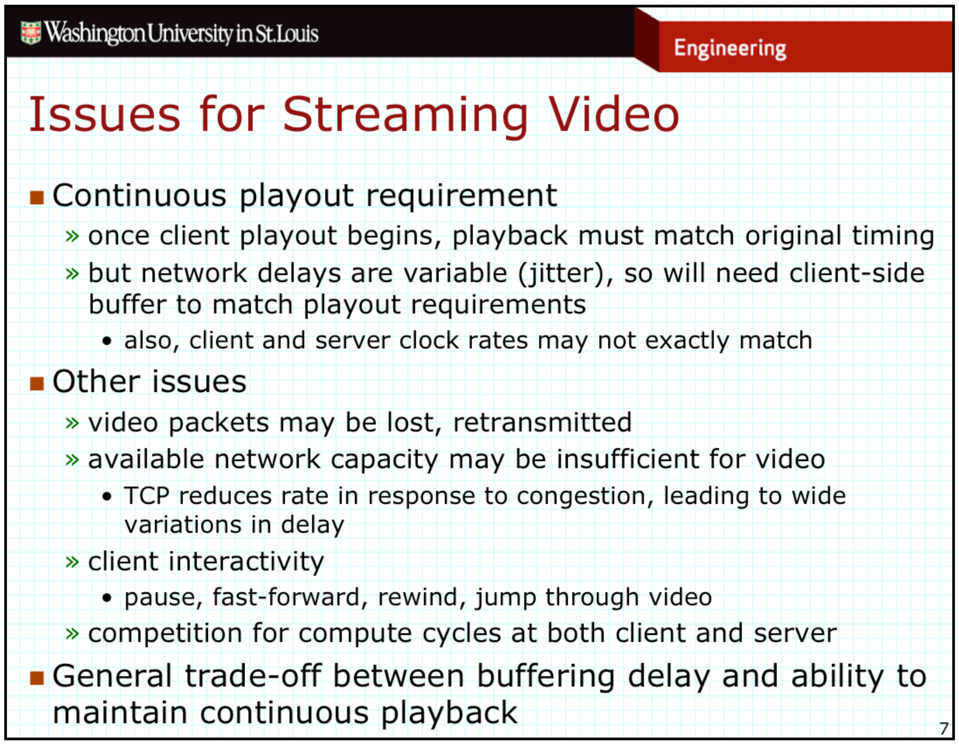
- Streaming, stored audio/video
  - » streaming: can begin playout before downloading entire file
  - » stored (at server): can transmit faster than audio/video will be rendered (implies storing/buffering at client)
  - » e.g., YouTube, Netflix, Hulu
- Streaming live audio/video
  - » data sent as it happens
    - client can still buffer a few seconds to ensure smooth playout in presence of variable delay
  - » e.g., internet radio, live sporting event
- Real-time voice/video over IP
  - » interactive nature of person-to-person conversation limits delay tolerance – so very limited client-side buffering
  - » e.g., Skype, Vonage



Streaming Stored Video


Cumulative data
video sent at constant frame rate (e.g. 30 f/s)
network delay (fixed in this example)
video received, played out at client
time

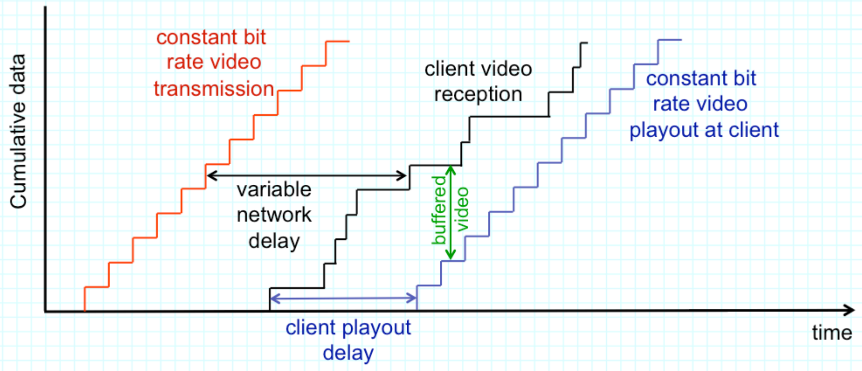
streaming: at this time, client playing out early part of video, while server still sending later part of video



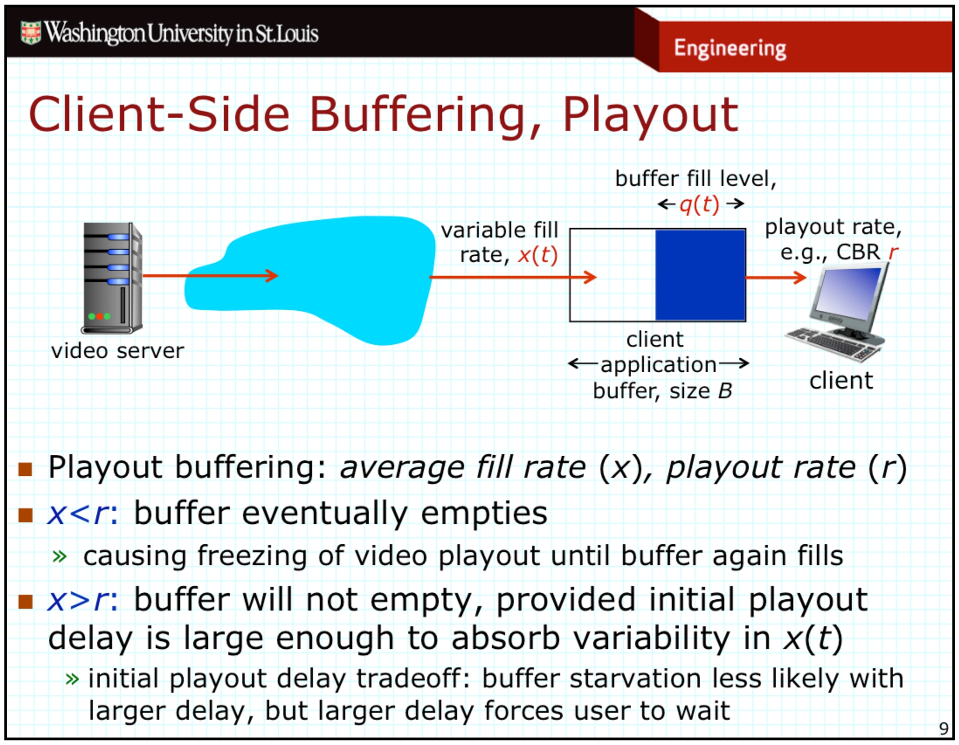
Issues for Streaming Video

- Continuous playout requirement
  - » 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
    - also, client and server clock rates may not exactly match
- Other issues
  - » video packets may be lost, retransmitted
  - » available network capacity may be insufficient for video
    - TCP reduces rate in response to congestion, leading to wide variations in delay
  - » client interactivity
    - pause, fast-forward, rewind, jump through video
  - » competition for compute cycles at both client and server
- General trade-off between buffering delay and ability to maintain continuous playback

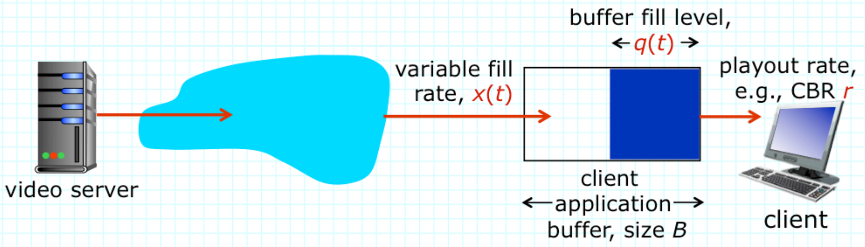
Dealing with Variable Network Delay



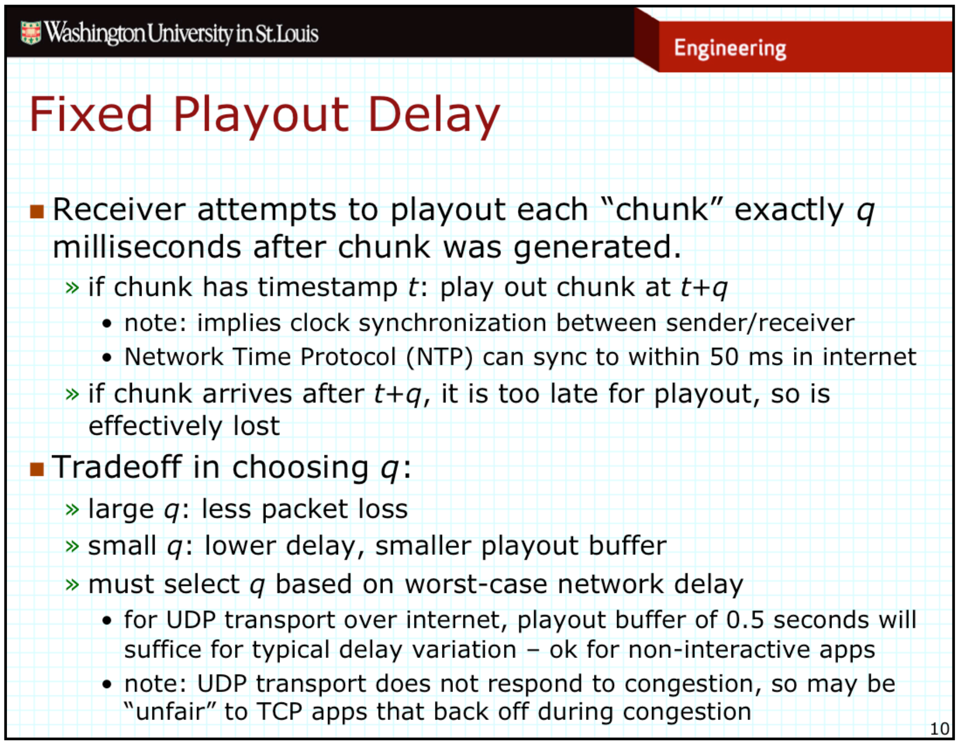
- Client-side buffering and playout delay
  - » compensate for network-added delay, delay jitter
  - » requires average available network capacity at least equal to video data rate



Client-Side Buffering, Playout

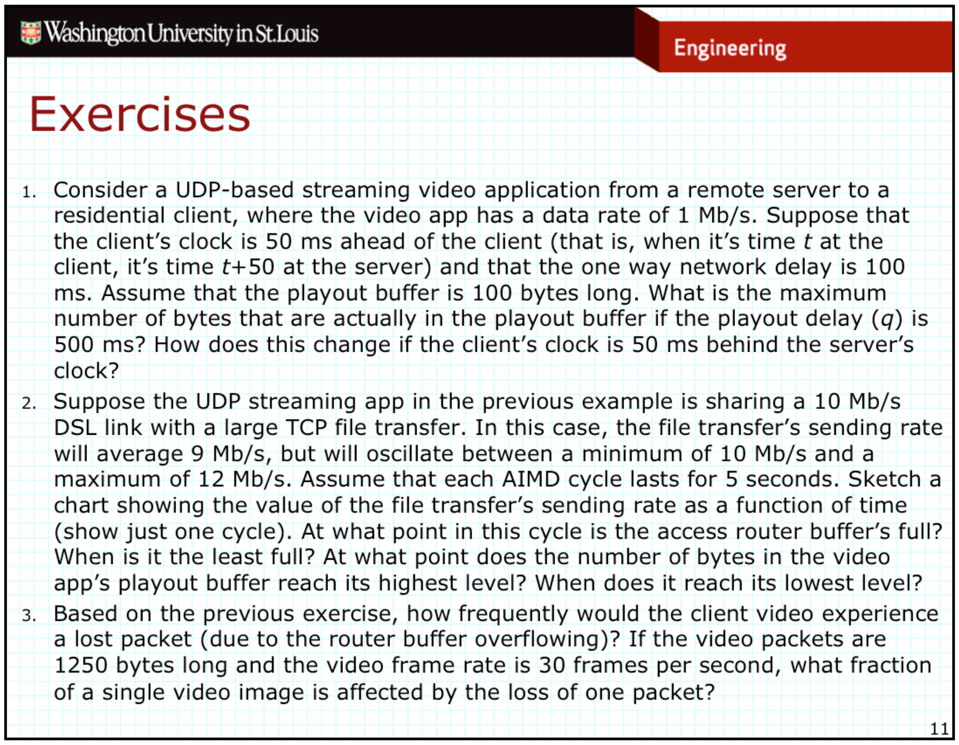

buffer fill level, $q(t)$
variable fill rate, $x(t)$
playout rate, e.g., CBR r
video server
client application buffer, size B
client

- Playout buffering: *average fill rate* (x), *playout rate* (r)
- $x<r$: buffer eventually empties
  - » causing freezing of video playout until buffer again fills
- $x>r$: 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 forces user to wait



Fixed Playout Delay

- Receiver attempts to playout each "chunk" exactly *q* milliseconds after chunk was generated.
  - if chunk has timestamp *t*: play out chunk at *t+q*
    - note: implies clock synchronization between sender/receiver
    - Network Time Protocol (NTP) can sync to within 50 ms in internet
  - if chunk arrives after *t+q*, it is too late for playout, so is effectively lost
- Tradeoff in choosing *q*:
  - large *q*: less packet loss
  - small *q*: lower delay, smaller playout buffer
  - must select *q* based on worst-case network delay
    - for UDP transport over internet, playout buffer of 0.5 seconds will suffice for typical delay variation – ok for non-interactive apps
    - note: UDP transport does not respond to congestion, so may be "unfair" to TCP apps that back off during congestion



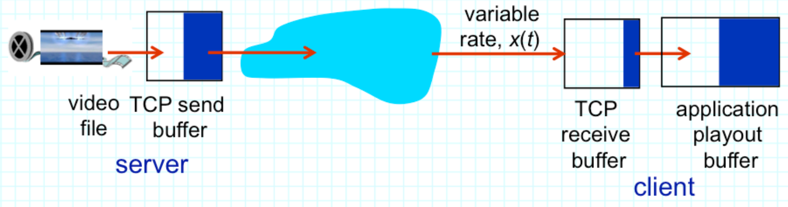
Exercises

1. Consider a UDP-based streaming video application from a remote server to a residential client, where the video app has a data rate of 1 Mb/s. Suppose that the client's clock is 50 ms ahead of the client (that is, when it's time *t* at the client, it's time *t*+50 at the server) and that the one way network delay is 100 ms. Assume that the playout buffer is 100 bytes long. What is the maximum number of bytes that are actually in the playout buffer if the playout delay (*q*) is 500 ms? How does this change if the client's clock is 50 ms behind the server's clock?
2. Suppose the UDP streaming app in the previous example is sharing a 10 Mb/s DSL link with a large TCP file transfer. In this case, the file transfer's sending rate will average 9 Mb/s, but will oscillate between a minimum of 10 Mb/s and a maximum of 12 Mb/s. Assume that each AIMD cycle lasts for 5 seconds. Sketch a chart showing the value of the file transfer's sending rate as a function of time (show just one cycle). At what point in this cycle is the access router buffer's full? When is it the least full? At what point does the number of bytes in the video app's playout buffer reach its highest level? When does it reach its lowest level?
3. Based on the previous exercise, how frequently would the client video experience a lost packet (due to the router buffer overflowing)? If the video packets are 1250 bytes long and the video frame rate is 30 frames per second, what fraction of a single video image is affected by the loss of one packet?

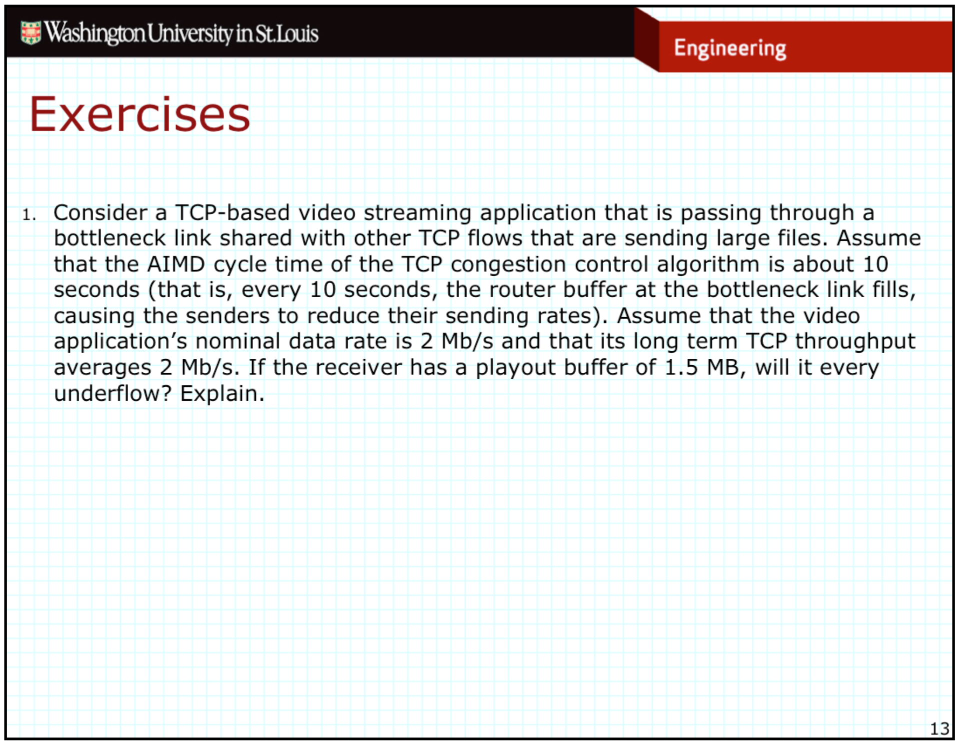
11

Streaming Multimedia: HTTP

- Multimedia file retrieved via HTTP GET
  - » natural integration of video apps with web/browser
- Send at maximum "fair" rate under TCP

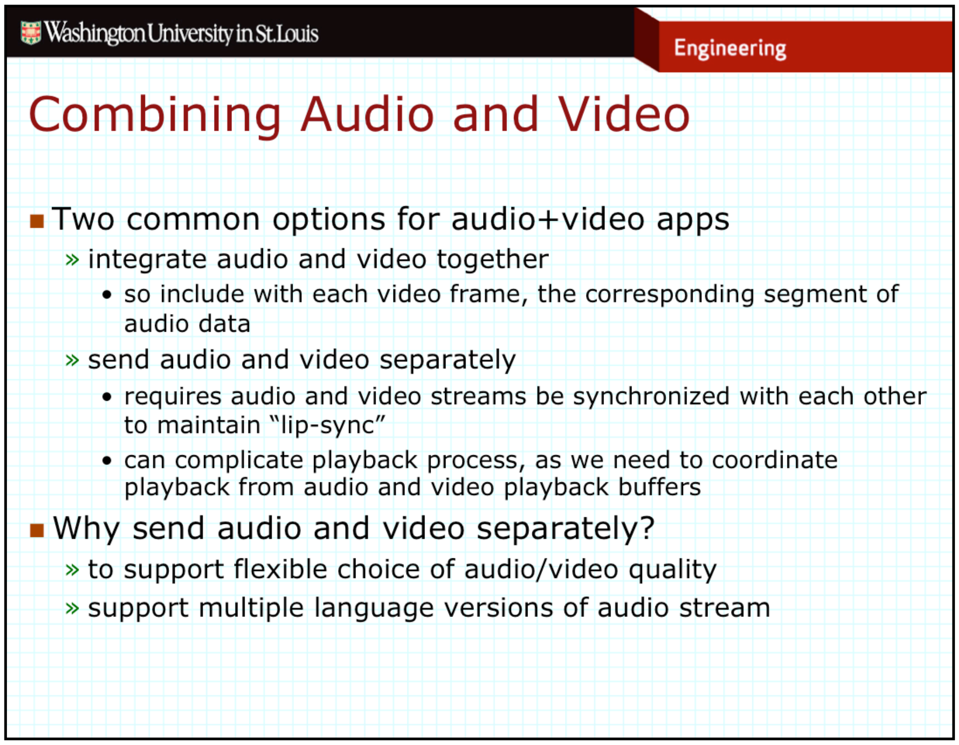


- Packet retransmissions can lead to increased delay
- Fill rate fluctuates due to TCP congestion control
  - » requires larger playout delay (5-30 sec)
    - allow time for TCP to pass through full AIMD cycle
  - » ok for stored video, but no good for conversational apps



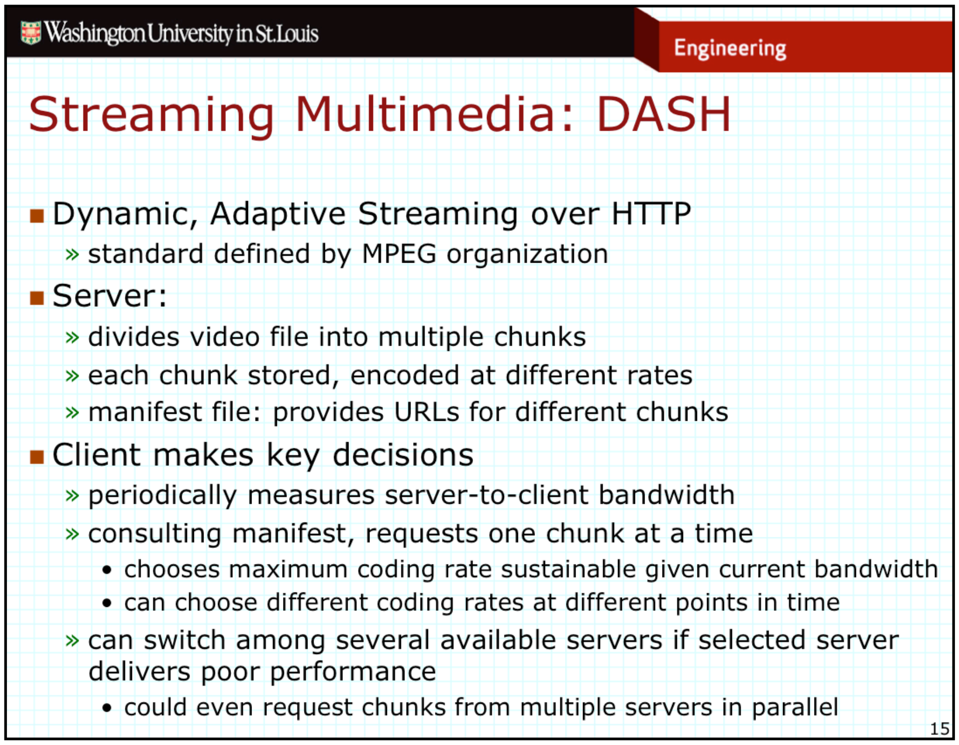
Exercises

1. Consider a TCP-based video streaming application that is passing through a bottleneck link shared with other TCP flows that are sending large files. Assume that the AIMD cycle time of the TCP congestion control algorithm is about 10 seconds (that is, every 10 seconds, the router buffer at the bottleneck link fills, causing the senders to reduce their sending rates). Assume that the video application's nominal data rate is 2 Mb/s and that its long term TCP throughput averages 2 Mb/s. If the receiver has a playout buffer of 1.5 MB, will it every underflow? Explain.



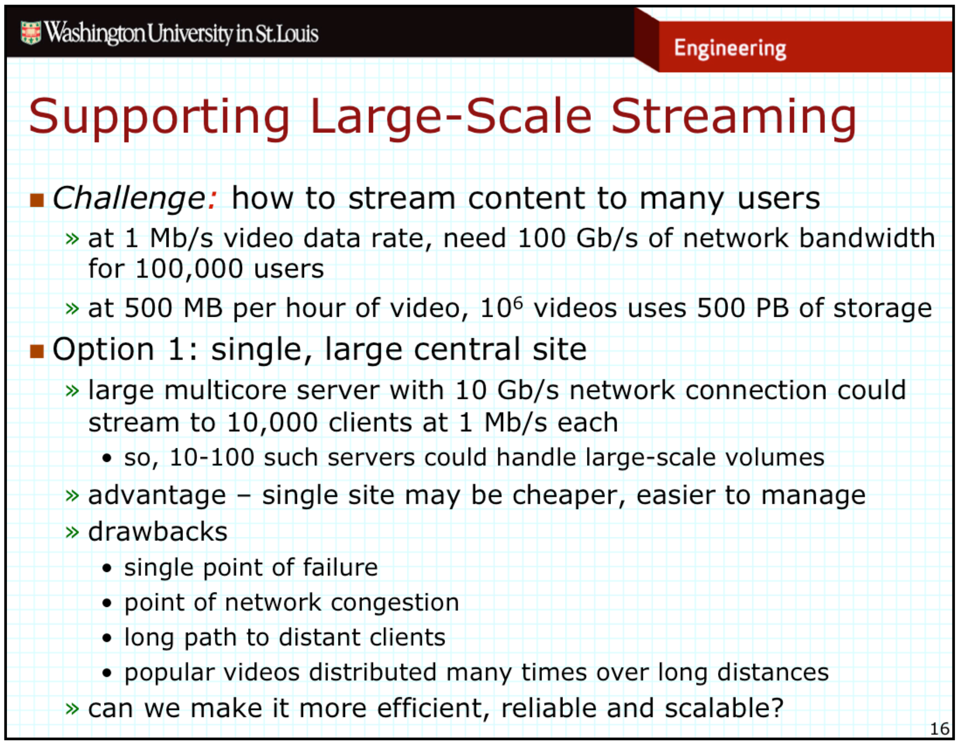
Combining Audio and Video

- Two common options for audio+video apps
  - » integrate audio and video together
    - so include with each video frame, the corresponding segment of audio data
  - » send audio and video separately
    - requires audio and video streams be synchronized with each other to maintain "lip-sync"
    - can complicate playback process, as we need to coordinate playback from audio and video playback buffers
- Why send audio and video separately?
  - » to support flexible choice of audio/video quality
  - » support multiple language versions of audio stream



Streaming Multimedia: DASH

- Dynamic, Adaptive Streaming over HTTP
  - » standard defined by MPEG organization
- Server:
  - » divides video file into multiple chunks
  - » each chunk stored, encoded at different rates
  - » manifest file: provides URLs for different chunks
- Client makes key decisions
  - » periodically measures server-to-client bandwidth
  - » consulting manifest, requests one chunk at a time
    - chooses maximum coding rate sustainable given current bandwidth
    - can choose different coding rates at different points in time
  - » can switch among several available servers if selected server delivers poor performance
    - could even request chunks from multiple servers in parallel

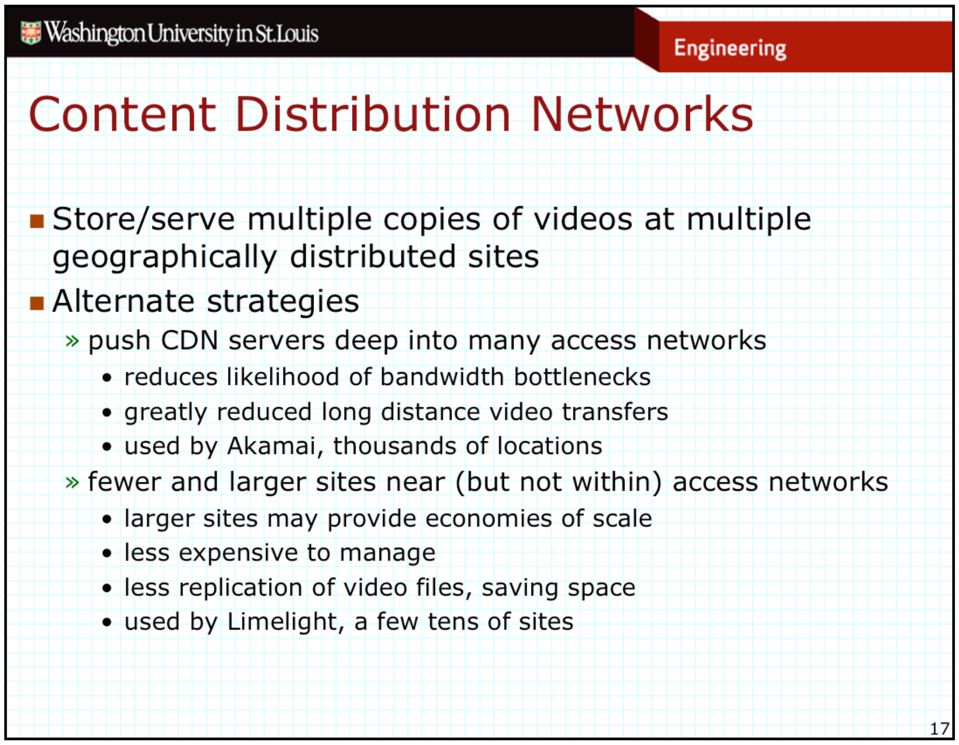


Washington University in St. Louis — Engineering

Supporting Large-Scale Streaming

- *Challenge:* how to stream content to many users
  - » at 1 Mb/s video data rate, need 100 Gb/s of network bandwidth for 100,000 users
  - » at 500 MB per hour of video, 10^6 videos uses 500 PB of storage
- Option 1: single, large central site
  - » large multicore server with 10 Gb/s network connection could stream to 10,000 clients at 1 Mb/s each
    - so, 10-100 such servers could handle large-scale volumes
  - » advantage – single site may be cheaper, easier to manage
  - » drawbacks
    - single point of failure
    - point of network congestion
    - long path to distant clients
    - popular videos distributed many times over long distances
  - » can we make it more efficient, reliable and scalable?

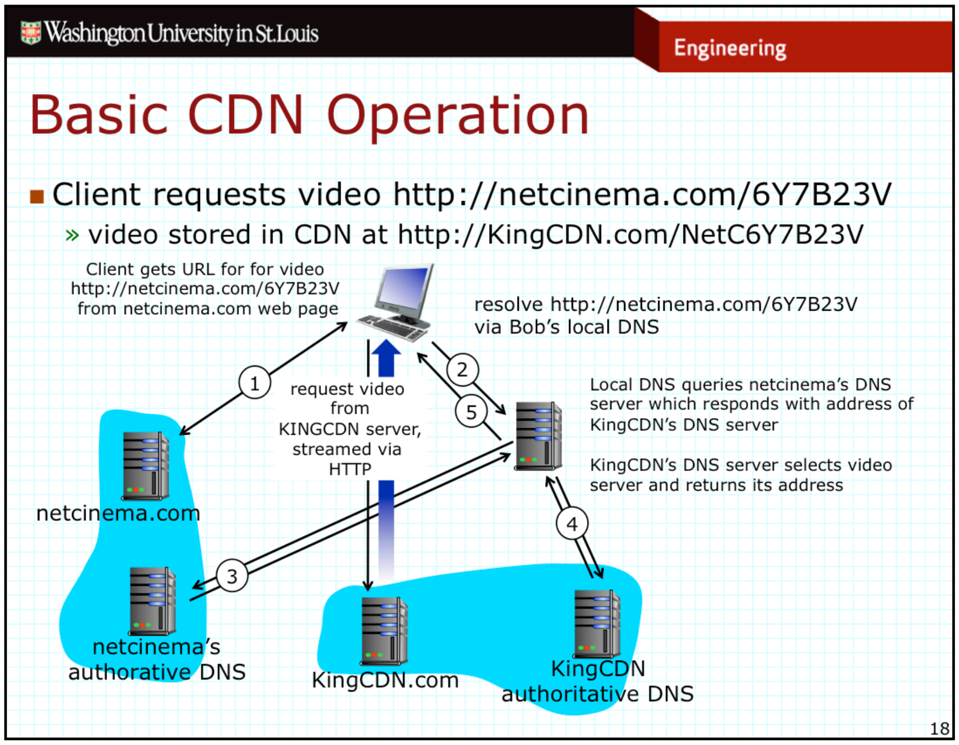
16



Content Distribution Networks

- Store/serve multiple copies of videos at multiple geographically distributed sites
- Alternate strategies
  - » push CDN servers deep into many access networks
    - reduces likelihood of bandwidth bottlenecks
    - greatly reduced long distance video transfers
    - used by Akamai, thousands of locations
  - » fewer and larger sites near (but not within) access networks
    - larger sites may provide economies of scale
    - less expensive to manage
    - less replication of video files, saving space
    - used by Limelight, a few tens of sites

17

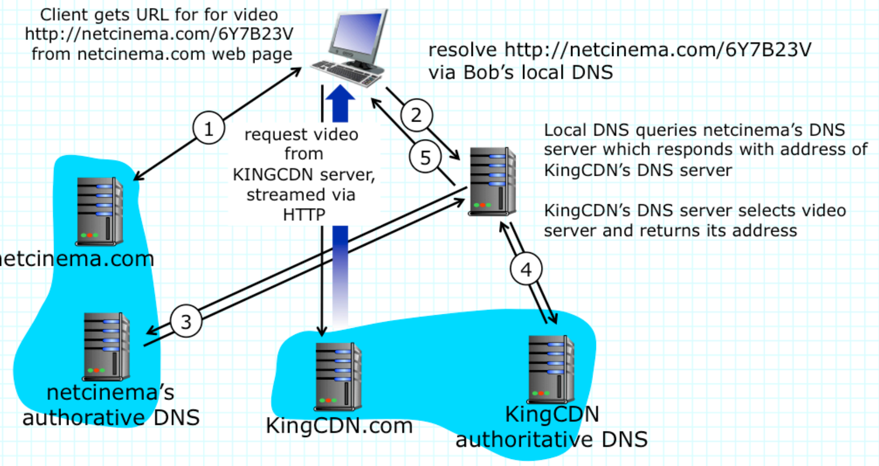


Washington University in St.Louis

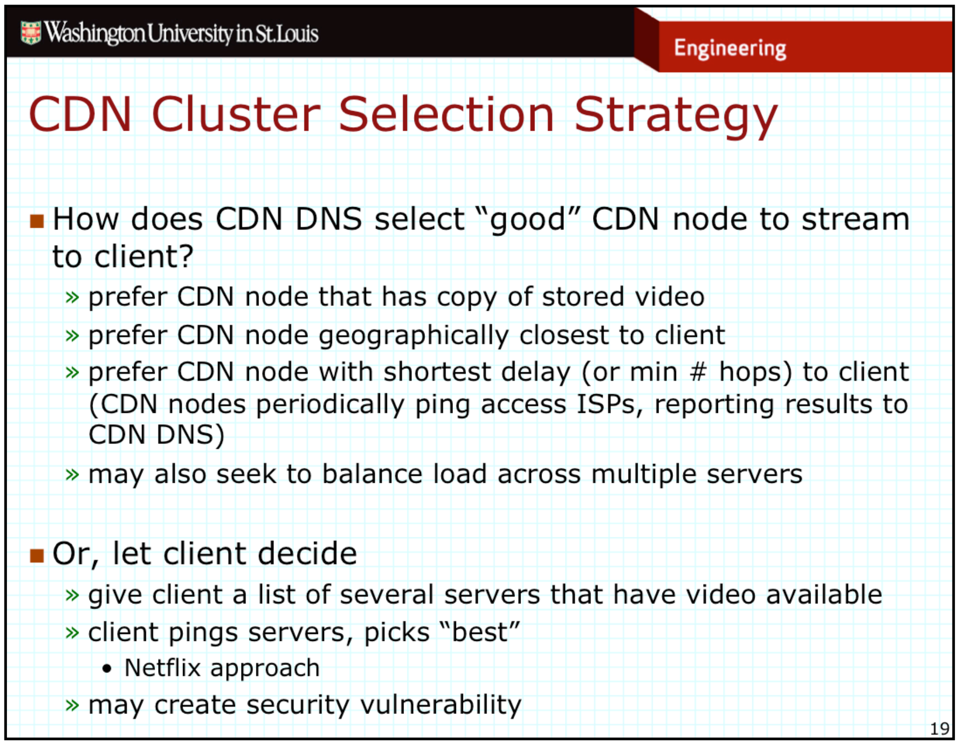
Engineering

Basic CDN Operation

- Client requests video http://netcinema.com/6Y7B23V
  - » video stored in CDN at http://KingCDN.com/NetC6Y7B23V


Client gets URL for for video
http://netcinema.com/6Y7B23V
from netcinema.com web page
resolve http://netcinema.com/6Y7B23V
via Bob's local DNS
1
2
5
request video
from
KINGCDN server,
streamed via
HTTP
Local DNS queries netcinema's DNS
server which responds with address of
KingCDN's DNS server
KingCDN's DNS server selects video
server and returns its address
netcinema.com
4
3
netcinema's
authorative DNS
KingCDN.com
KingCDN
authoritative DNS

18



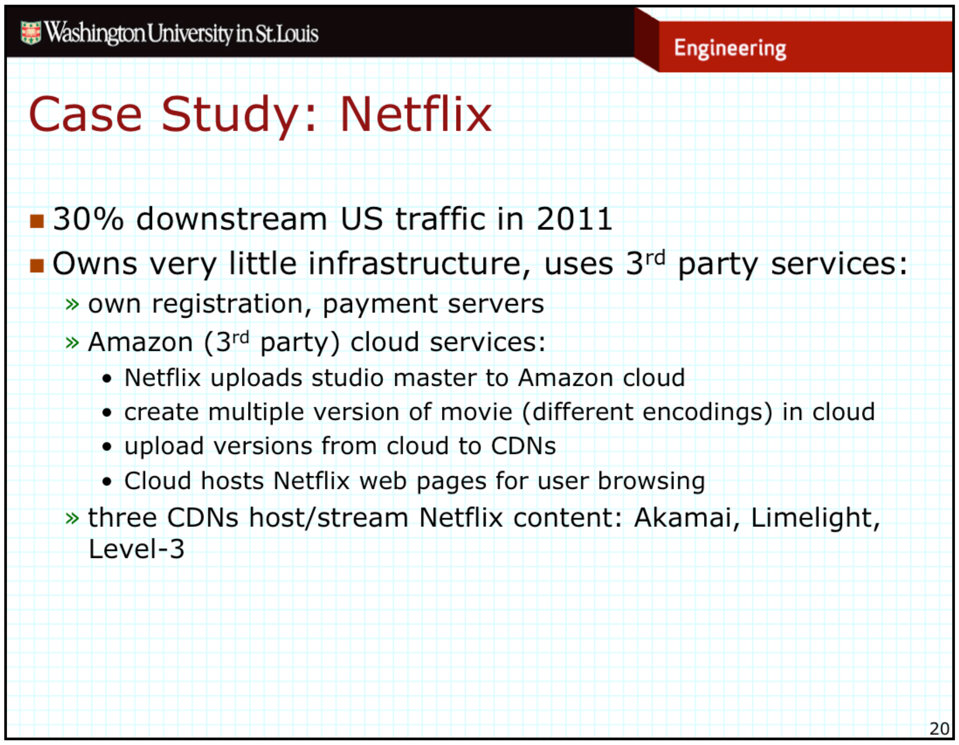
Washington University in St.Louis

Engineering

CDN Cluster Selection Strategy

- How does CDN DNS select "good" CDN node to stream to client?
  - » prefer CDN node that has copy of stored video
  - » prefer CDN node geographically closest to client
  - » prefer CDN node with shortest delay (or min # hops) to client (CDN nodes periodically ping access ISPs, reporting results to CDN DNS)
  - » may also seek to balance load across multiple servers
- Or, let client decide
  - » give client a list of several servers that have video available
  - » client pings servers, picks "best"
    - Netflix approach
  - » may create security vulnerability

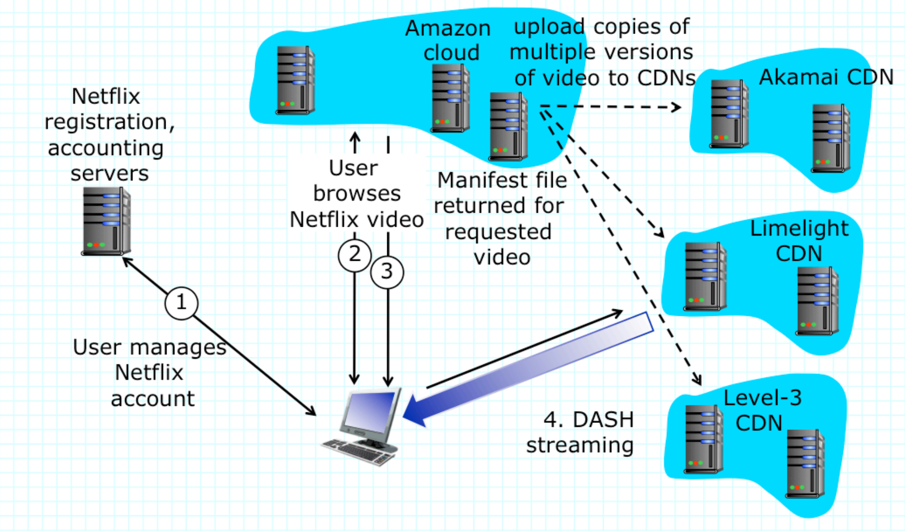
19



Case Study: Netflix

- 30% downstream US traffic in 2011
- Owns very little infrastructure, uses 3rd party services:
  - » own registration, payment servers
  - » Amazon (3rd party) cloud services:
    - Netflix uploads studio master to Amazon cloud
    - create multiple version of movie (different encodings) in cloud
    - upload versions from cloud to CDNs
    - Cloud hosts Netflix web pages for user browsing
  - » three CDNs host/stream Netflix content: Akamai, Limelight, Level-3

Case Study: Netflix


Amazon cloud
upload copies of multiple versions of video to CDNs
Akamai CDN
Netflix registration, accounting servers
User browses Netflix video
Manifest file returned for requested video
Limelight CDN
1
2
3
User manages Netflix account
Level-3 CDN
4. DASH streaming