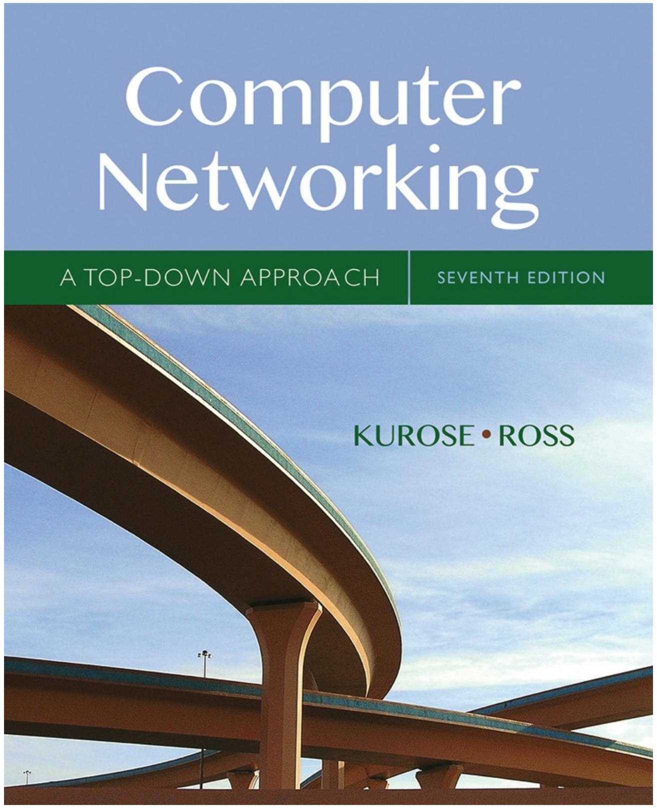
Chapter 7
Multimedia Networking



A note on the use of these ppt slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a lot of work on our part. In return for use, we only ask the following:

- *If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)*
- *If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.*

Thanks and enjoy! JFK/KWR

Computer Networking: A Top Down Approach

7th edition

Jim Kurose, Keith Ross

Pearson/Addison Wesley
April 2016

Multimedia networking: outline

7.1 multimedia networking applications

7.2 streaming stored video

7.3 voice-over-IP

Multimedia networking: outline

7.1 multimedia networking applications

7.2 streaming stored video

7.3 voice-over-IP

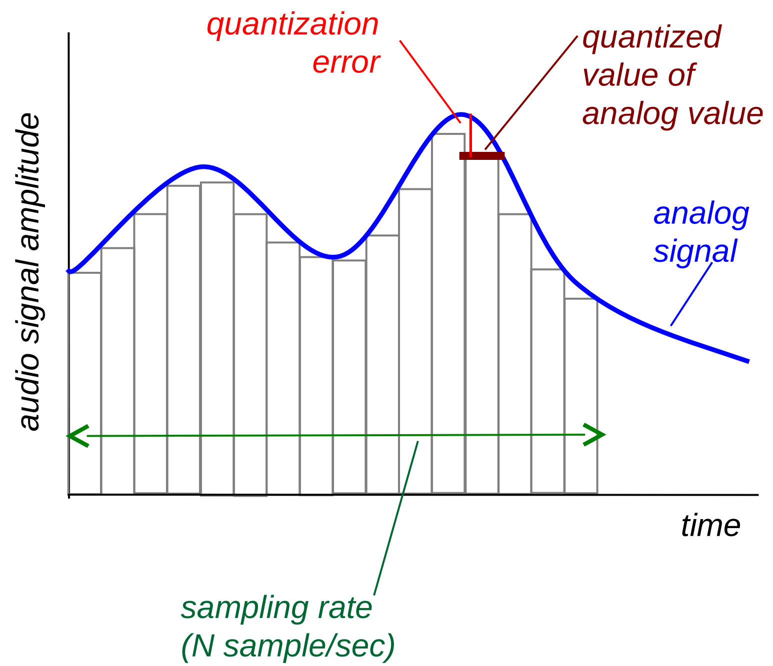
Multimedia: bit rate comparison

- *Frank is looking at Facebook pages*
  - *new photo every 10 seconds,*
  - *photos are on average 200 Kbytes in size*
- *Martha is listening to many MP3 songs*
  - *one after the other, each encoded at a rate of 128 kbps*
- *Victor is watching a video*

	Bit rate	Bytes transferred in 67 min
Facebook Frank	160 kbps	80 Mbytes
Martha Music	128 kbps	64 Mbytes
Victor Video	2 Mbps	1 Gbyte

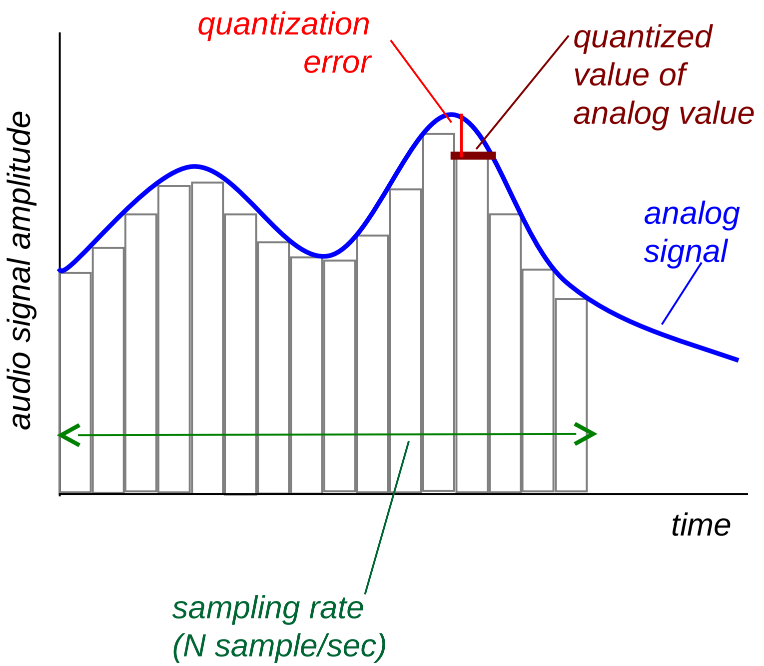
Multimedia: audio

- 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



Multimedia: audio

- example: 8,000 samples/sec, 256 quantized values: 64,000 bps
- receiver converts bits back to analog signal:
  - some quality reduction



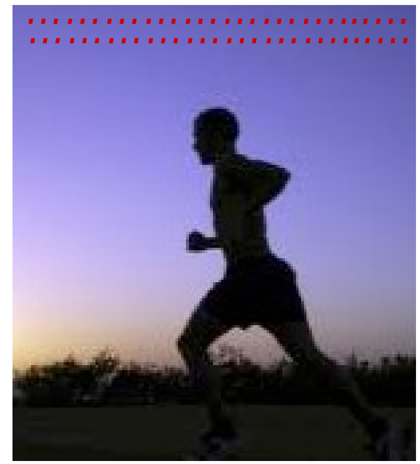
example rates

- CD: 1.411 Mbps
- MP3: 96, 128, 160 kbps

Multimedia: video

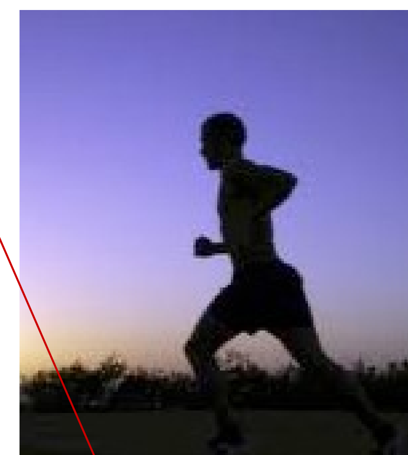
- *video: sequence of images displayed at constant rate*
  - *e.g. 24, 30 images/sec*
- *digital image: array of pixels*
  - *each pixel represented by bits*
- *coding: use redundancy within and between images to decrease # bits used to encode image*
  - *spatial (within image)*
  - *temporal (from one image to next)*

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

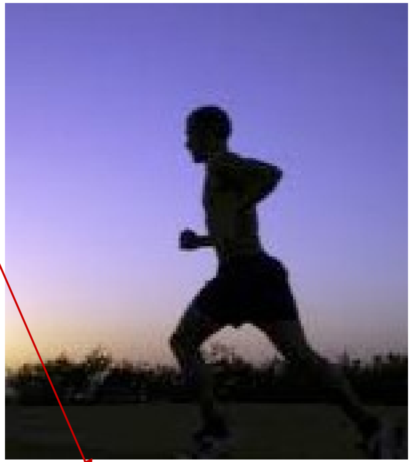
Multimedia: video

- *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, < 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

Youtube bit rate

SDR: Standard Dynamic Range
HDR: High Dynamic Range

Velocità in bit video consigliate per i caricamenti SDR

Tipo	Velocità in bit video, frequenza fotogrammi standard (24, 25, 30)	Velocità in bit video, frequenza fotogrammi elevata (48, 50, 60)
2160p (4K)	35-45 Mbps	53-68 Mbps
1440p (2K)	16 Mbps	24 Mbps
1080p	8 Mbps	12 Mbps
720p	5 Mbps	7,5 Mbps
480p	2,5 Mbps	4 Mbps
360p	1 Mbps	1,5 Mbps

Velocità in bit video consigliate per i caricamenti HDR

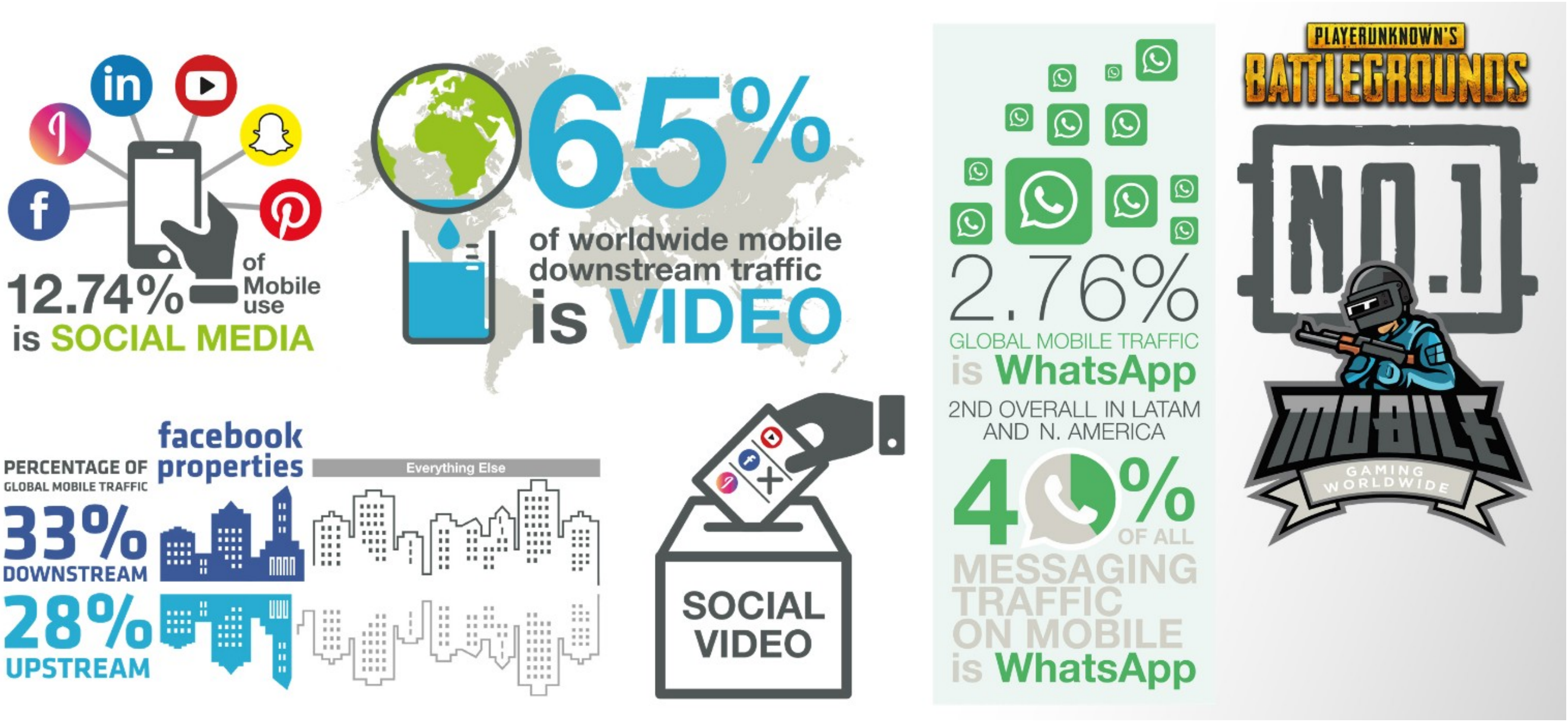
Tipo	Velocità in bit video, frequenza fotogrammi standard (24, 25, 30)	Velocità in bit video, frequenza fotogrammi elevata (48, 50, 60)
2160p (4K)	44-56 Mbps	66-85 Mbps
1440p (2K)	20 Mbps	30 Mbps
1080p	10 Mbps	15 Mbps
720p	6,5 Mbps	9,5 Mbps
480p	Non supportato	Non supportato
360p	Non supportato	Non supportato

Velocità in bit audio consigliate per i caricamenti

Tipo	Velocità in bit audio
Mono	128 kbps
Stereo	384 kbps
5.1	512 kbps

Source: https://support.google.com/youtube/answer/1722171?hl=it#zippy=%2Ccontenitore-mp%2Ccodec-audio-aac-lc%2Cfrequenza-fotogrammi%2Cvelocit%C3%A0-in-bit

Current situation of mobile traffic



Source: https://www.sandvine.com/phenomena

Multimedia networking: 3 application types

- *streaming, stored audio, video*
  - *streaming: can begin play out 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*
- *conversational voice/video over IP*
  - *interactive nature of human-to-human conversation limits delay tolerance*
    - *<150, [150:400], >400 ms*
  - *e.g., Skype*
- *streaming live audio, video*
  - *e.g., live sporting event (football)*

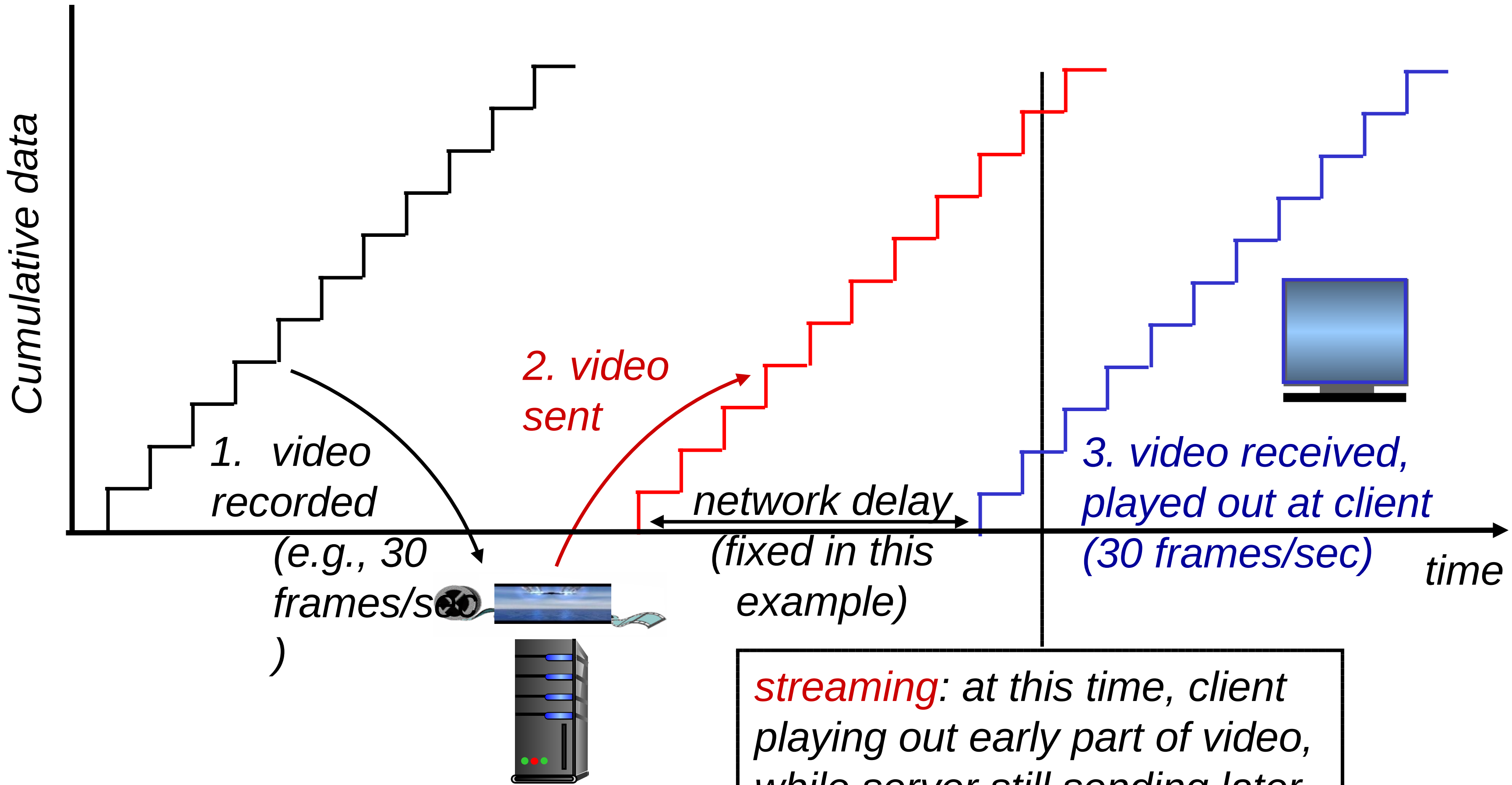
Multimedia networking: outline

7.1 multimedia networking applications

7.2 streaming stored video

7.3 voice-over-IP

Streaming stored video:

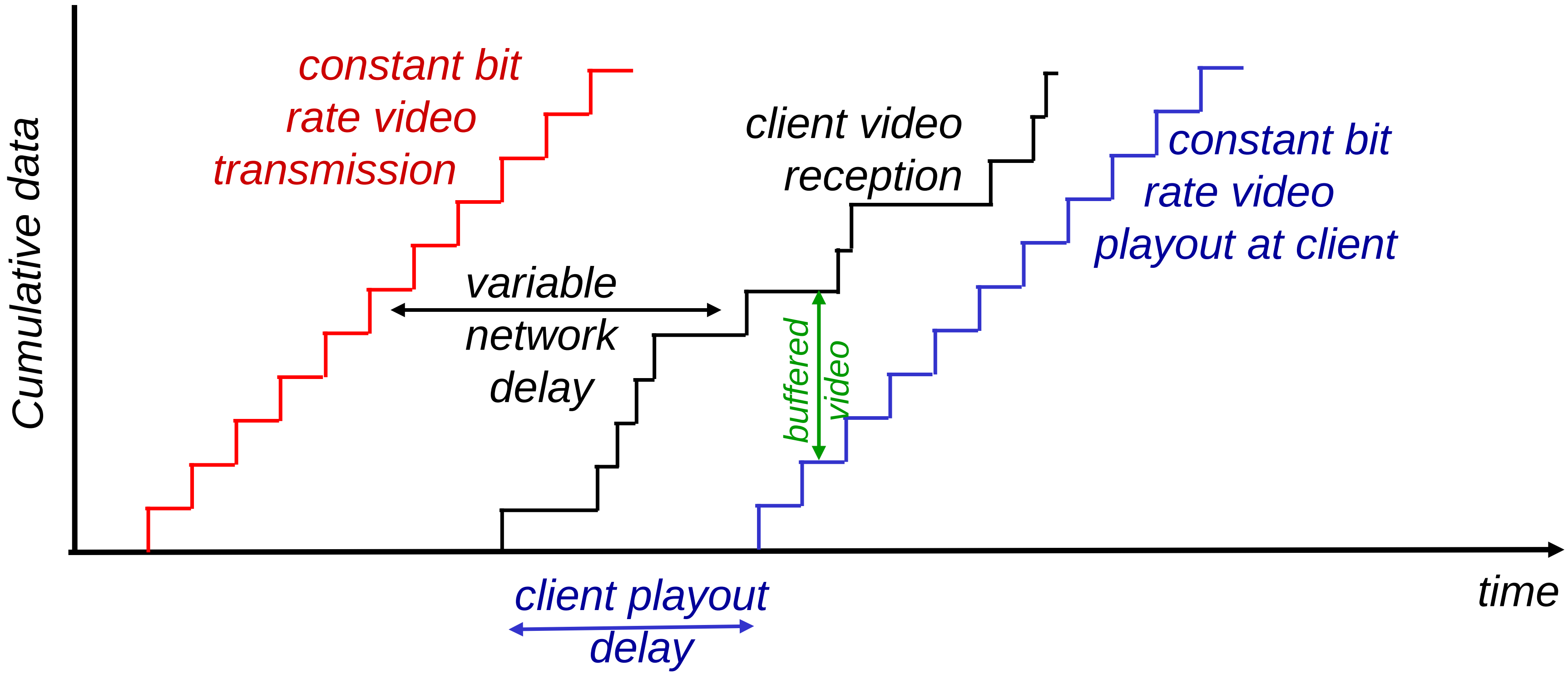


streaming: at this time, client playing out early part of video, while server still sending later part of video

Streaming stored video: challenges

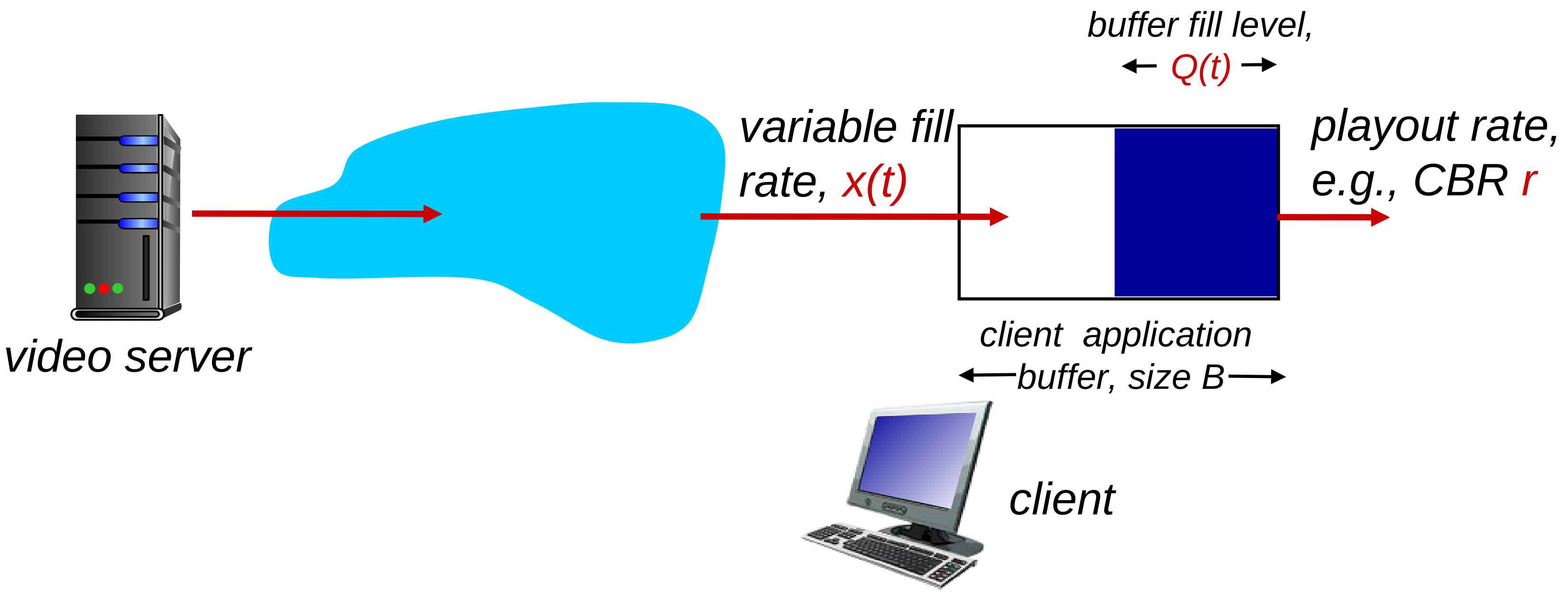
- *continuous playout constraint*: once client playout begins, playback must match original timing
  - … but *network delays are variable* (jitter), so will need *client-side buffer* to match playout requirements
- *other challenges:*
  - *client interactivity: pause, fast-forward, rewind, jump through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

Streaming stored video: revisted

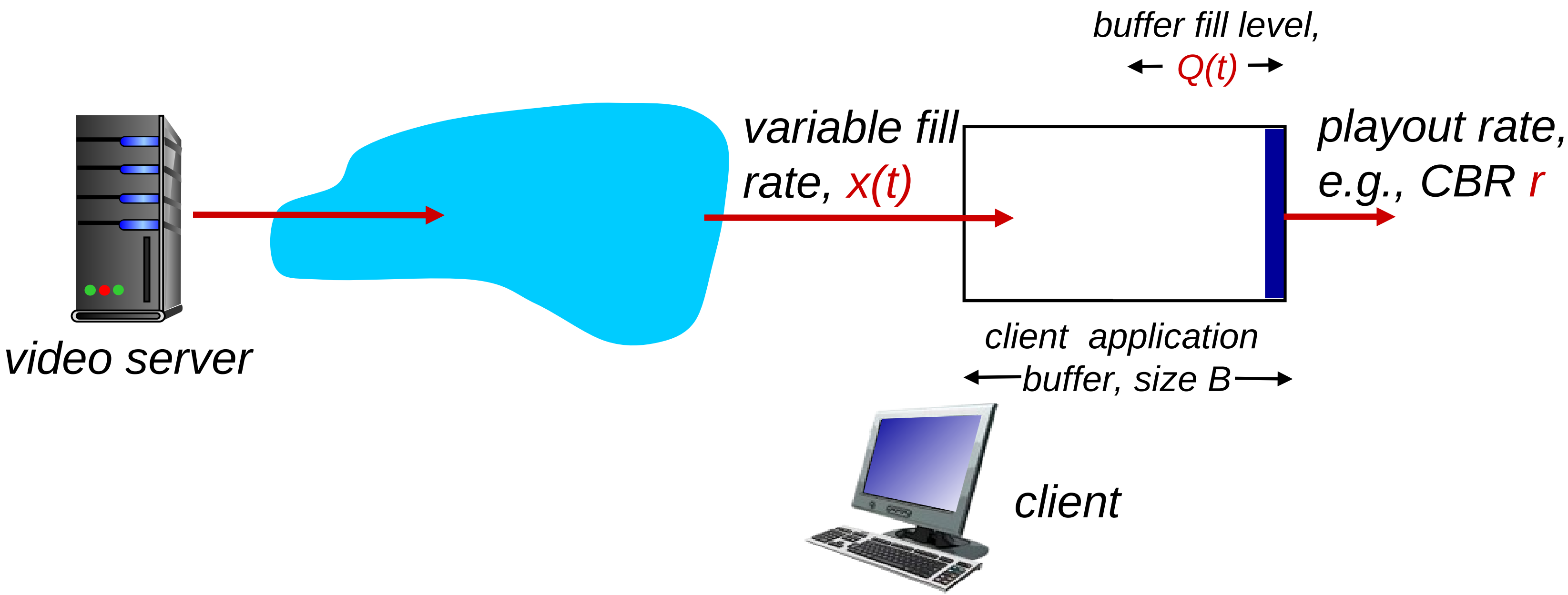


- *client-side buffering and playout delay:* compensate for network-added delay, delay jitter

Client-side buffering, playout



Client-side buffering, playout

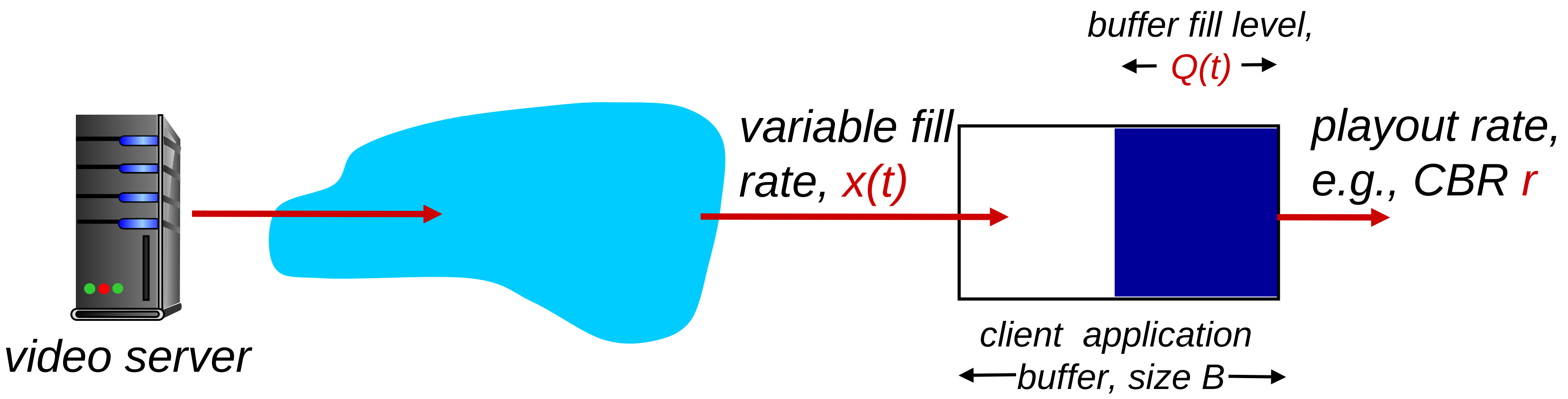


1. Initial fill of buffer until playout begins at t_p

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):

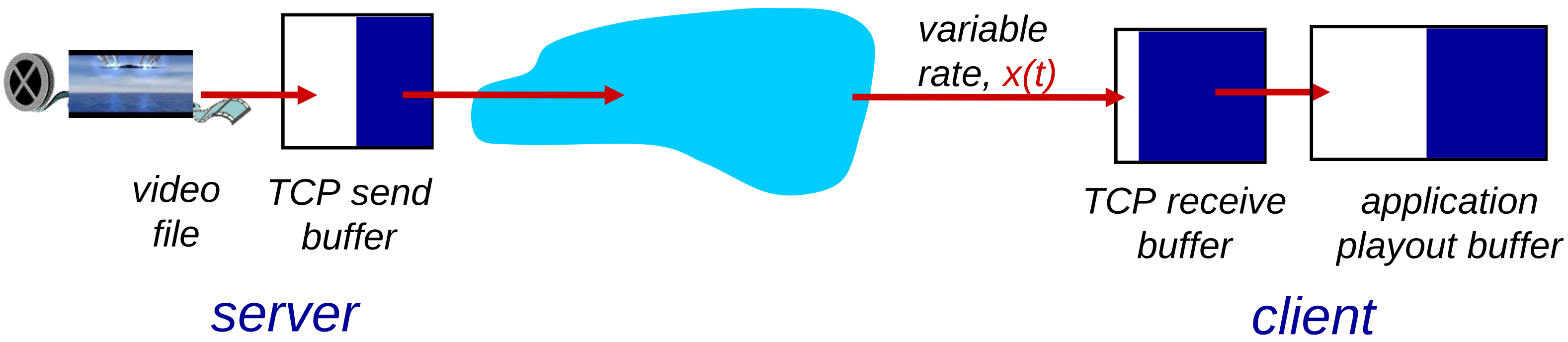
- $\bar{x} < r$: buffer eventually empties (causing freezing of video playout until buffer again fills)
- $\bar{x} > r$: buffer will not empty, provided initial playout delay is large enough to absorb variability in x(t)
  - *initial playout delay tradeoff:* buffer starvation less likely with larger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

Streaming multimedia: HTTP

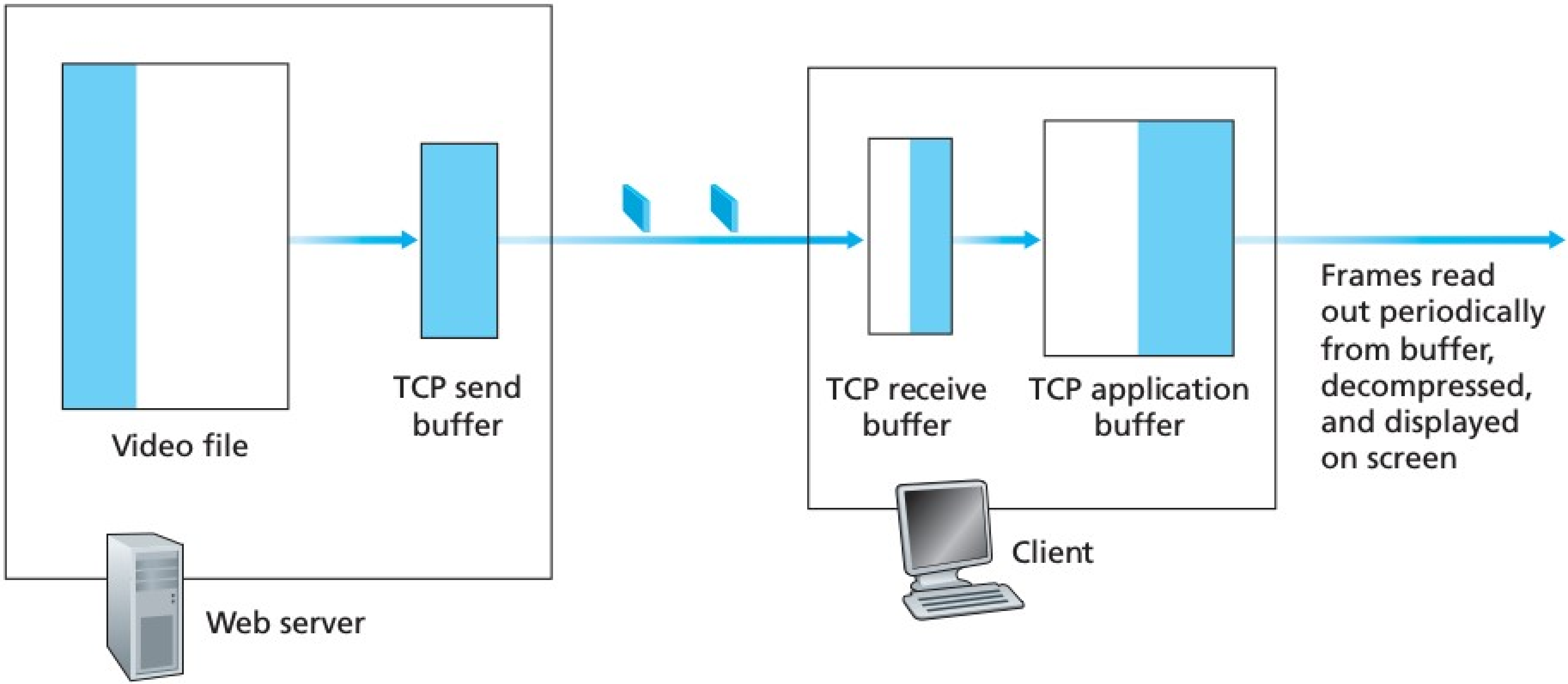
- multimedia file retrieved via HTTP GET
- send at maximum possible rate under TCP (if > consumption rate we have prefetching)



- 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

Client Application an TCP Buffers

- *full client application buffer indirectly imposes limit on the transmission rate from server to client streaming over*



Repositioning the Video

- What happen when the user wants to jump to a future point in time in the video?
- Exploitation of the HTTP byte-range header
  - specifies the specific range of bytes to retrieve
- User jump to a new position
  - client sends a new HTTP request with the byte-range header from which byte in the file should the server send data
  - the server receiving the new HTTP request can forget about any earlier request
- Jump to future point or earlier termination…

  …waste of network bandwidth and server resources !

Streaming multimedia: DASH

- *DASH: Dynamic, Adaptive Streaming over HTTP*
- *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 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 (depending on available bandwidth at time)*

Streaming multimedia: DASH

- *DASH: Dynamic, Adaptive Streaming over HTTP*
- *"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 bandwidth available)*
  - *where to request chunk (can request from URL server that is "close" to client or has high available bandwidth)*

Content distribution networks

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

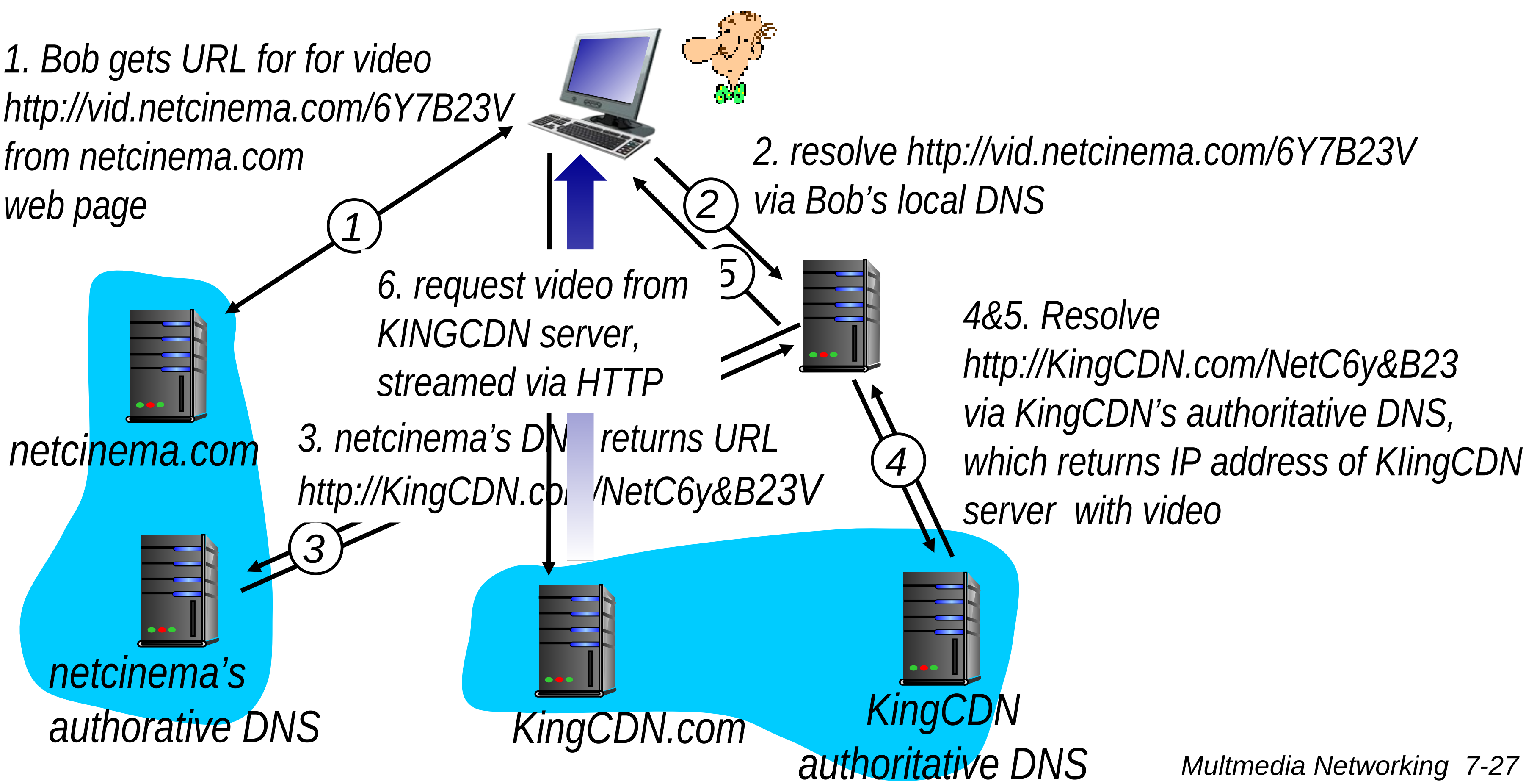
- *challenge: how to stream content (selected from millions of videos) to hundreds of thousands of simultaneous users?*

- *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*
    - *used by Akamai, 1700 locations*
  - *bring home: smaller number (10's) of larger clusters in key points near (but not within) access networks*
    - *used by Limelight*

CDN: "simple" content access scenario

Bob (client) requests video http://vid.netcinema.com/6Y7B23V

- video stored in CDN at http://KingCDN.com/NetC6y&B23V



CDN cluster selection strategy

- *challenge:* how does CDN DNS select "good" CDN node to stream to client
  - pick CDN node geographically closest to client
  - pick CDN node with shortest delay (or min # hops) to client (CDN nodes periodically ping access ISPs, reporting results to CDN DNS)
  - IP anycast
- *alternative:* let *client* decide - give client a list of several CDN servers
  - client pings servers, picks "best"
  - Netflix approach

Multimedia networking: outline

7.1 multimedia networking applications

7.2 streaming stored video

7.3 voice-over-IP

Voice-over-IP (VoIP)

- *VoIP end-end-delay requirement: needed to maintain "conversational" aspect*
  - *higher delays noticeable, impair interactivity*
  - *< 150 msec: good*
  - *> 400 msec bad*
  - *includes application-level (packetization,playout), network delays*
- *session initialization: how does callee advertise IP address, port number, encoding algorithms?*
- *value-added services: call forwarding, screening, recording*

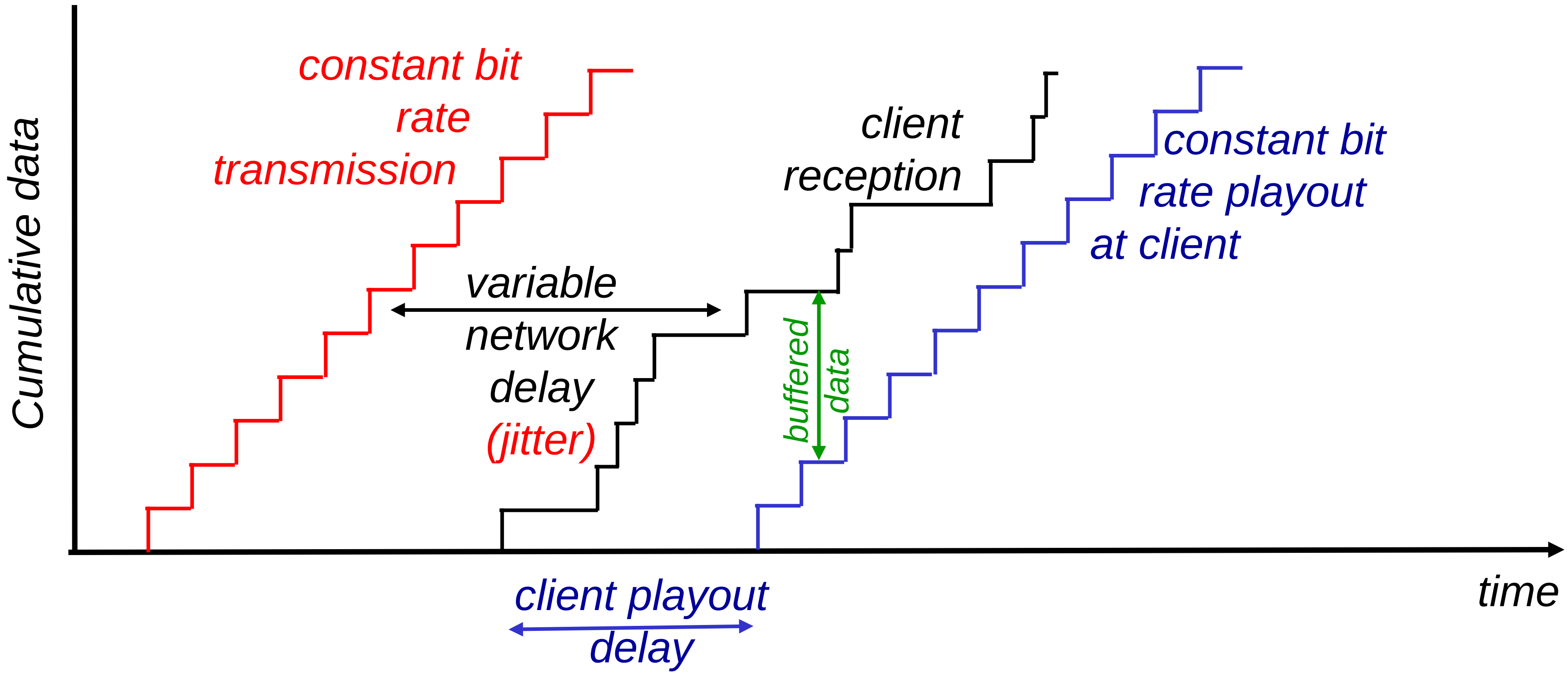
VoIP characteristics

- speaker's audio: alternating talk spurts, silent periods.
  - pkts generated only during talk spurts
  - Sender generates at 8 Kbytes/sec
  - With 20 msec chunks we have 160 bytes of data
- application-layer header added to each chunk
- chunk+header encapsulated into UDP or TCP segment
- application sends segment into socket every 20 msec during talkspurt

VoIP: packet loss, delay

- *network loss:* IP datagram lost due to network congestion (router buffer overflow)
- *delay loss:* IP datagram arrives too late for playout at receiver
  - delays: processing, queueing in network; end-system (sender, receiver) delays
  - typical maximum tolerable delay: 400 ms
- *loss tolerance:* depending on voice encoding, loss concealment, packet loss rates between 1% and 10% can be tolerated

Delay jitter



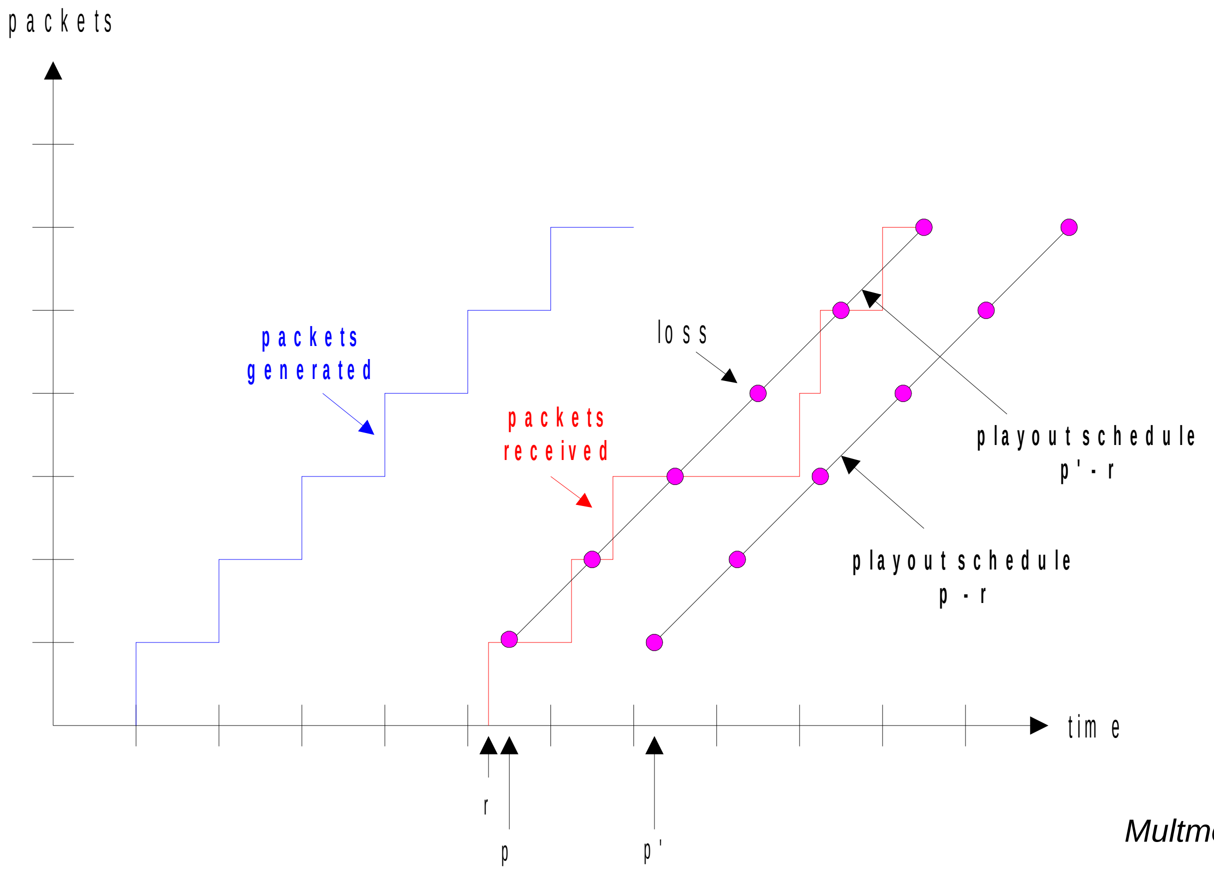
- end-to-end delays of two consecutive packets: difference can be more or less than 20 msec (transmission time difference)

VoIP: fixed playout delay

- *receiver attempts to playout each chunk exactly q msecs after chunk was generated.*
  - *chunk has time stamp t: play out chunk at t+q*
  - *chunk arrives after t+q: data arrives too late for playout: data “lost”*
- *tradeoff in choosing q:*
  - *large q: less packet loss*
  - *small q: better interactive experience*

VoIP: fixed playout delay

- sender generates packets every 20 msec during talk spurt.
- first packet received at time r
- first playout schedule: begins at p
- second playout schedule: begins at p'



Adaptive playout delay (1)

- goal: low playout delay, low late loss rate
- approach: adaptive playout delay adjustment:
  - estimate network delay, adjust playout delay at beginning of each talk spurt
  - silent periods compressed and elongated
  - chunks still played out every 20 msec during talk spurt
- adaptively estimate packet delay: (EWMA - exponentially weighted moving average, recall TCP RTT estimate):

$$d_i = (1-\alpha)d_{i-1} + \alpha\,\underbrace{(r_i - t_i)}_{\text{measured delay of ith packet}}$$

delay estimate after ith packet

small constant, e.g. 0.1

time received - time sent (timestamp)

measured delay of ith packet

Adaptive playout delay (2)

- *also useful to estimate average deviation of delay, v_i:*

$$v_i = (1-\beta)v_{i-1} + \beta\ |r_i - t_i - d_i|$$

- *estimates d_i, v_i calculated for every received packet, but used only at start of talk spurt*

- *for first packet in talk spurt, playout time is:*

$$\text{playout-time}_i = t_i + d_i + Kv_i$$

remaining packets in talkspurt are played out periodically

Adaptive playout delay (3)

Q: How does receiver determine whether packet is first in a talkspurt?

- if no loss, receiver looks at successive timestamps
  - difference of successive stamps > 20 msec -->talk spurt begins.
- with loss possible, receiver must look at both time stamps and sequence numbers
  - difference of successive stamps > 20 msec and sequence numbers without gaps --> talk spurt begins.

VoiP: recovery from packet loss (1)

Challenge: recover from packet loss given small tolerable delay between original transmission and playout

- *each ACK/NAK takes ~ one RTT*
- *alternative: Forward Error Correction (FEC)*
  - *send enough bits to allow recovery without retransmission (recall two-dimensional parity in Ch. 5)*

simple FEC

- *for every group of n chunks, create redundant chunk by exclusive OR-ing n original chunks*
- *send n+1 chunks, increasing bandwidth by factor 1/n*
- *can reconstruct original n chunks if at most one lost chunk from n+1 chunks, with playout delay*

VoiP: recovery from packet loss *(2)*

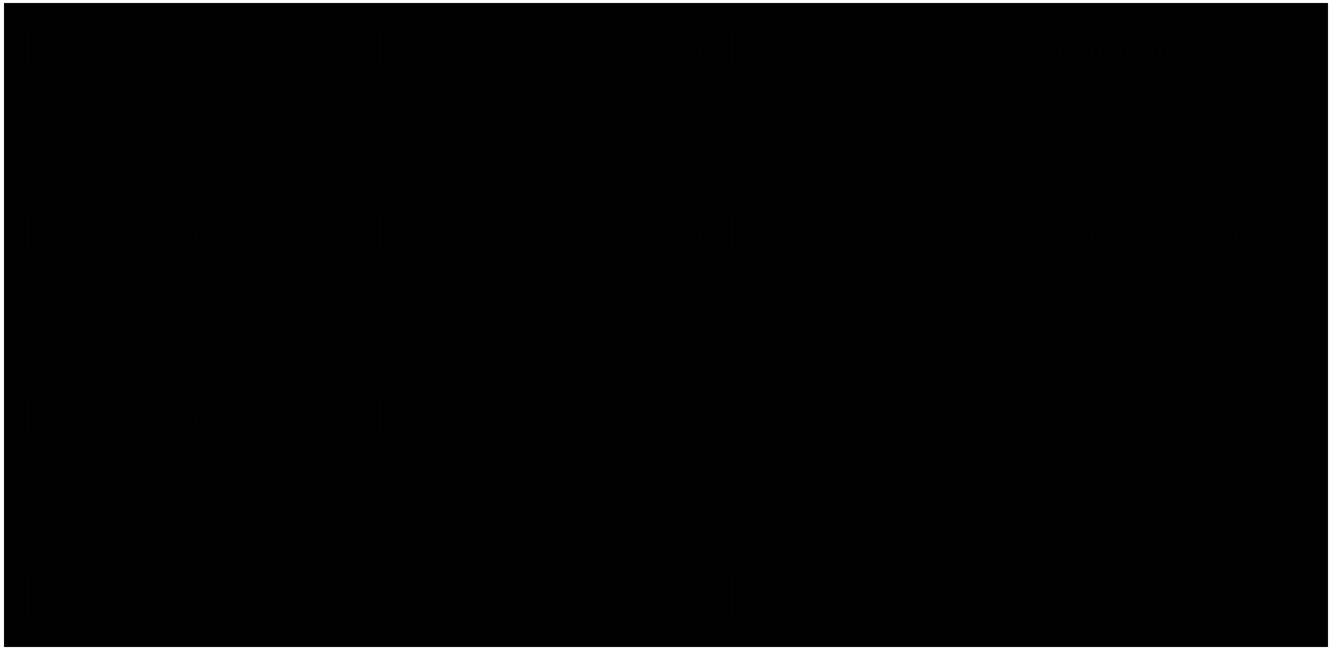
another FEC scheme:

- *"piggyback lower quality stream"*
- *send lower resolution audio stream as redundant information*
- *e.g., nominal stream PCM at 64 kbps and redundant stream GSM at 13 kbps*

[redacted]

- *non-consecutive loss: receiver can conceal loss*
- *generalization: can also append (n-1)st and (n-2)nd low-bit rate chunk*

VoiP: recovery from packet loss (3)

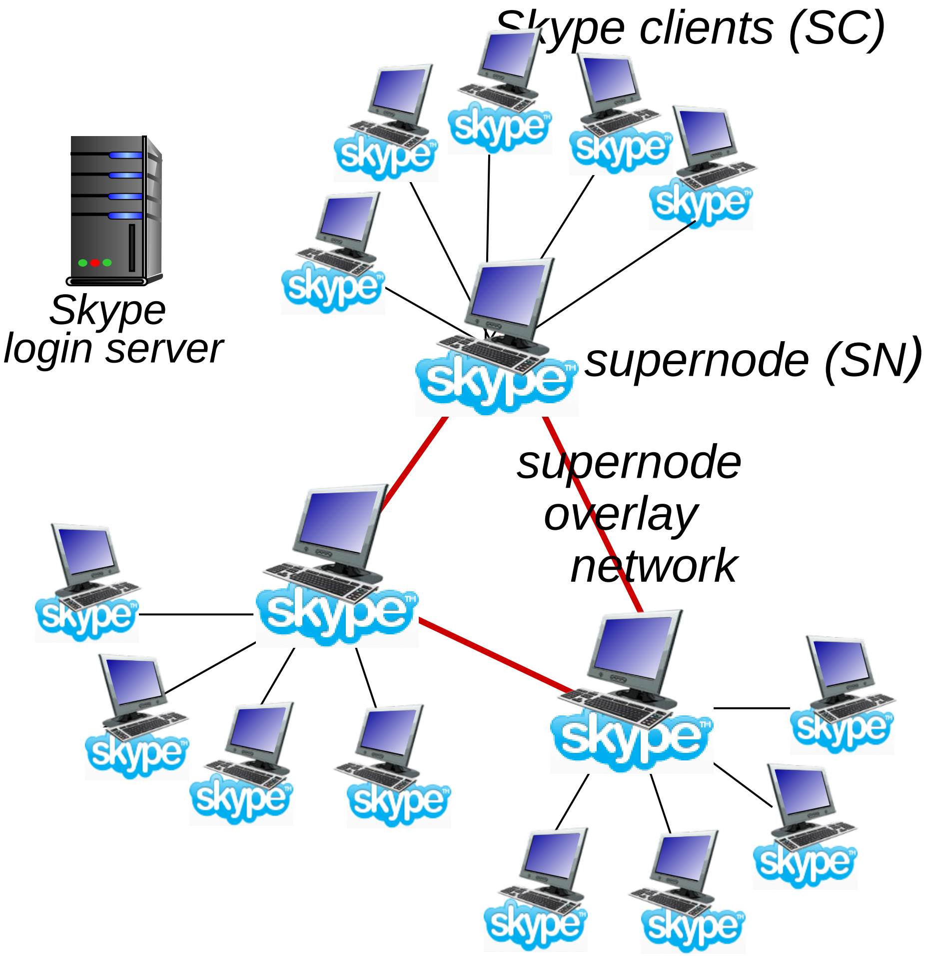


interleaving to conceal loss:

- *audio chunks divided into smaller units, e.g. four 5 msec units per 20 msec audio chunk*
- *packet contains small units from different chunks*
- *if packet lost, still have most of every original chunk*
- *no redundancy overhead, but increases playout delay*

Voice-over-IP: Skype

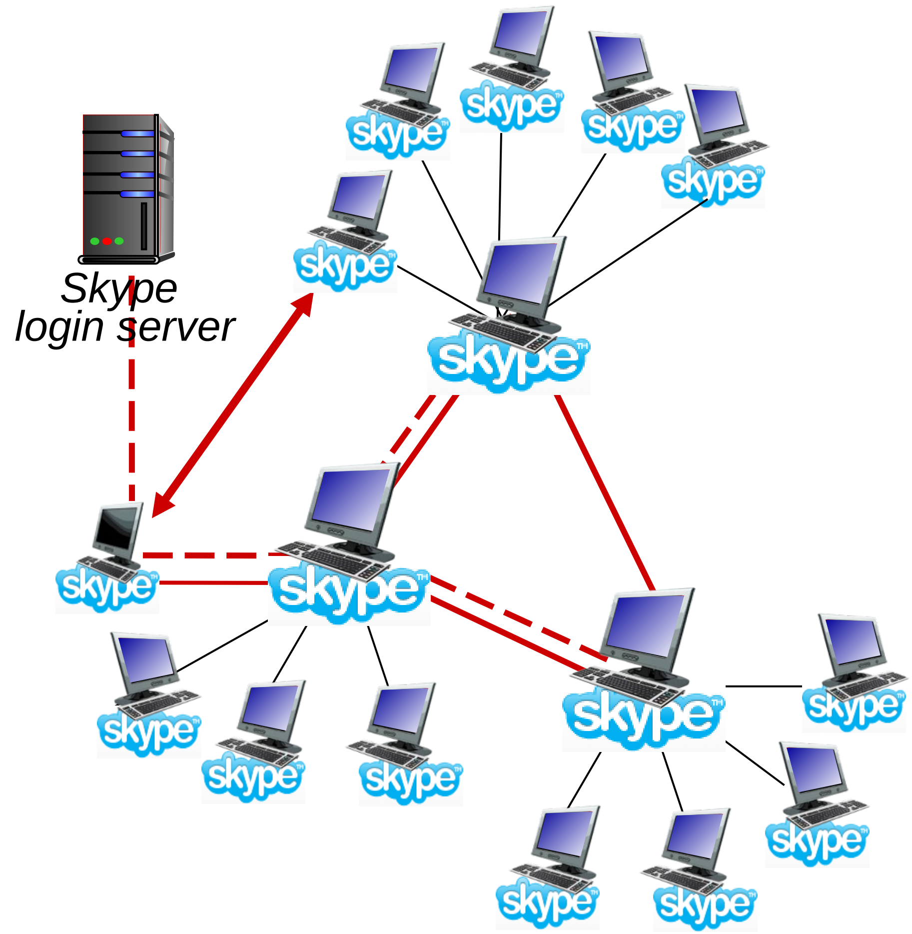
- proprietary application-layer protocol (inferred via reverse engineering)
  - encrypted msgs
- P2P components:
  - clients: skype peers connect directly to each other for VoIP call
  - super nodes (SN): skype peers with special functions
  - overlay network: among SNs to locate SCs
  - login server



P2P voice-over-IP: skype

skype client operation:

1. joins skype network by contacting SN (IP address cached) using TCP
2. logs-in (usename, password) to centralized skype login server
3. obtains IP address for callee from SN, SN overlay
   - or client buddy list
4. initiate call directly to callee



Skype: peers as relays

- *problem:* both Alice, Bob are behind "NATs"
  - *NAT prevents outside peer from initiating connection to insider peer*
  - *inside peer can initiate connection to outside*
- *relay solution:* Alice, Bob maintain open connection to their SNs
  - *Alice signals her SN to connect to Bob*
  - *Alice's SN connects to Bob's SN*
  - *Bob's SN connects to Bob over open connection Bob initially initiated to his SN*