



College of Information Studies

University of Maryland Hornbake Library Building College Park, MD 20742-4345

Streaming

Session 25

INST 346

Technologies, Infrastructure and Architecture

Goals for Today

- Video and audio data rates
- Streaming stored content
- Streaming conversational content

Video Streaming and CDNs: context

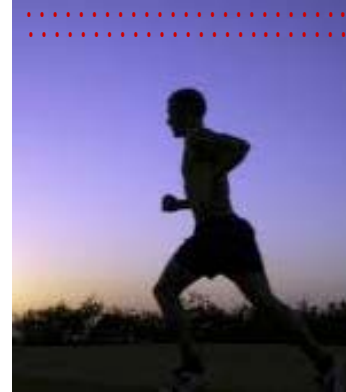
- video is the major consumer of Internet bandwidth
 - Netflix: 75 million users, 37% of residential traffic
 - YouTube: 1 billion users, 16% of residential traffic
- challenges: scale, bandwidth, heterogeneity
 - single mega-video server won't work
 - different users have different capabilities
 - wired vs. mobile
 - bandwidth rich vs. bandwidth poor
- *solution*: distributed, application-level infrastructure



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 I (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



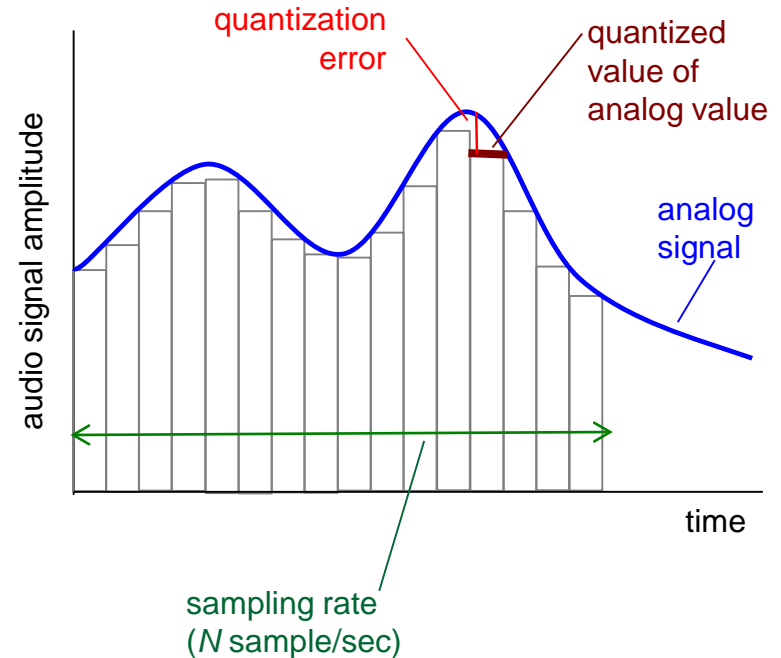
frame $i+1$

Multimedia: 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
- Internet telephony: 5.3 kbps and up

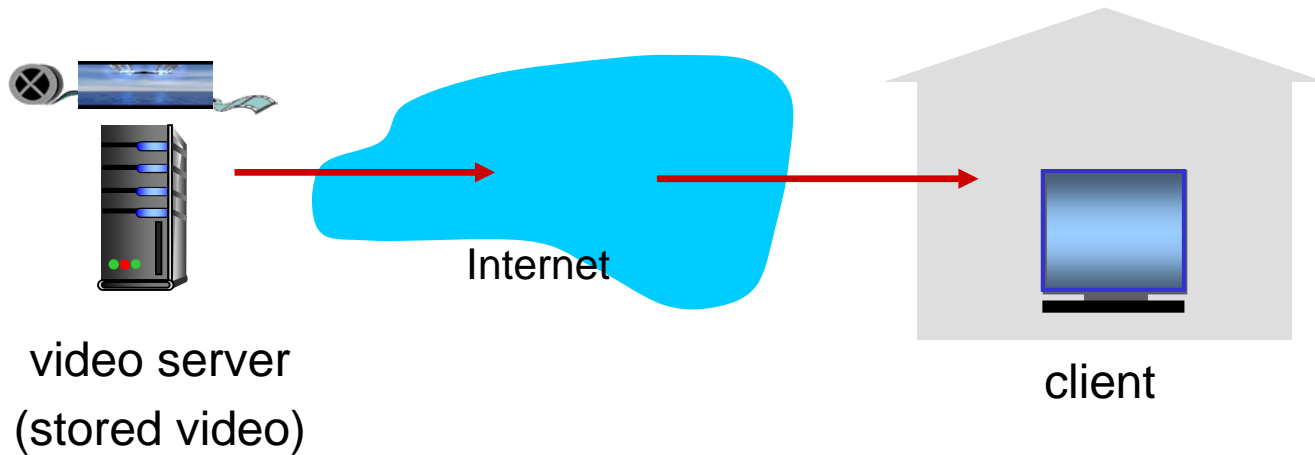


Multimedia networking: 3 application types

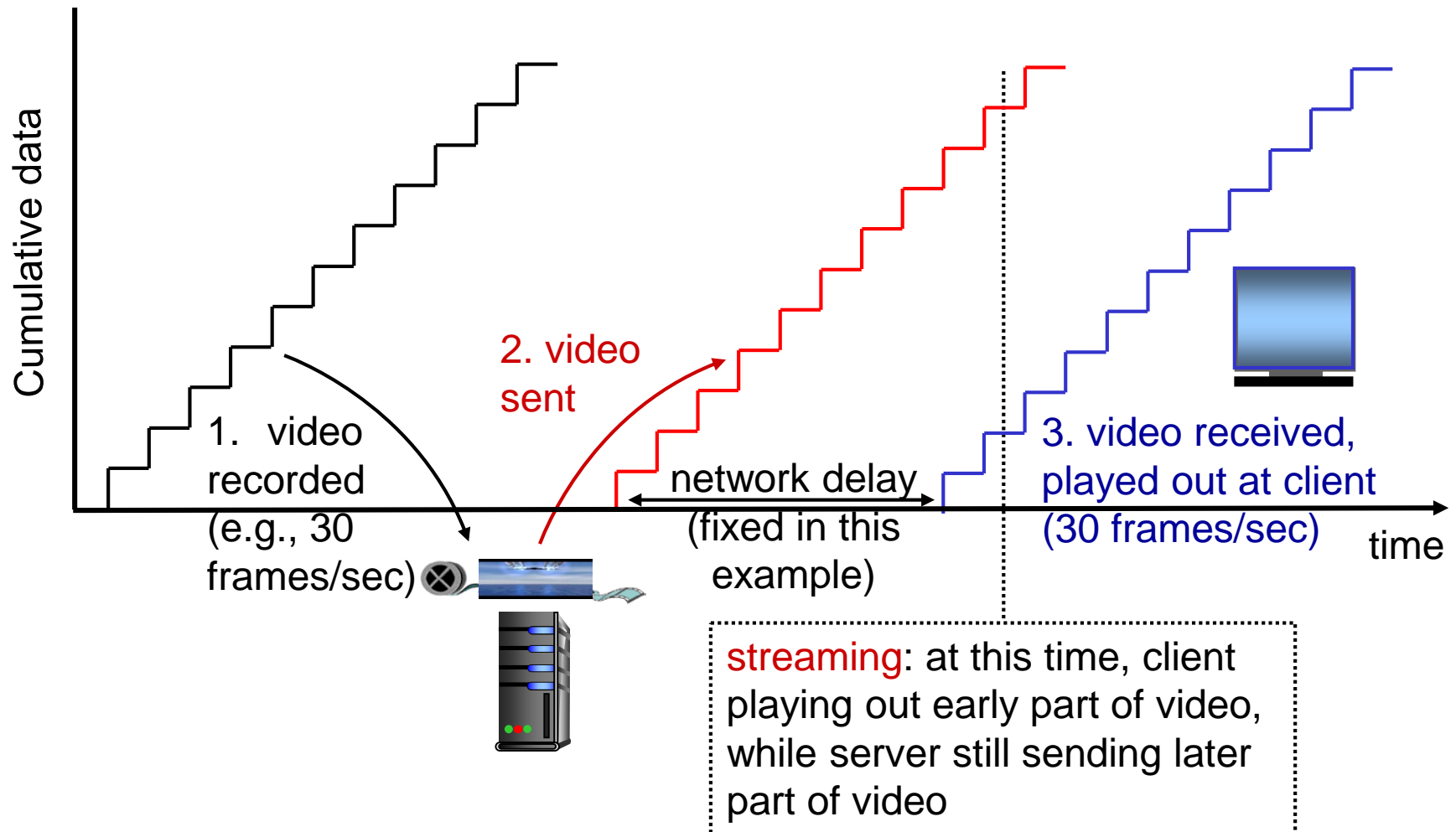
- *streaming, stored* audio, video
 - *streaming*: can begin playout before downloading entire file
 - *stored (at server)*: can transmit faster than audio/video will be rendered (implies storing/buffering at client)
 - e.g., YouTube, Netflix, Hulu
- *conversational* voice/video over IP
 - interactive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

Streaming stored video:

simple scenario:



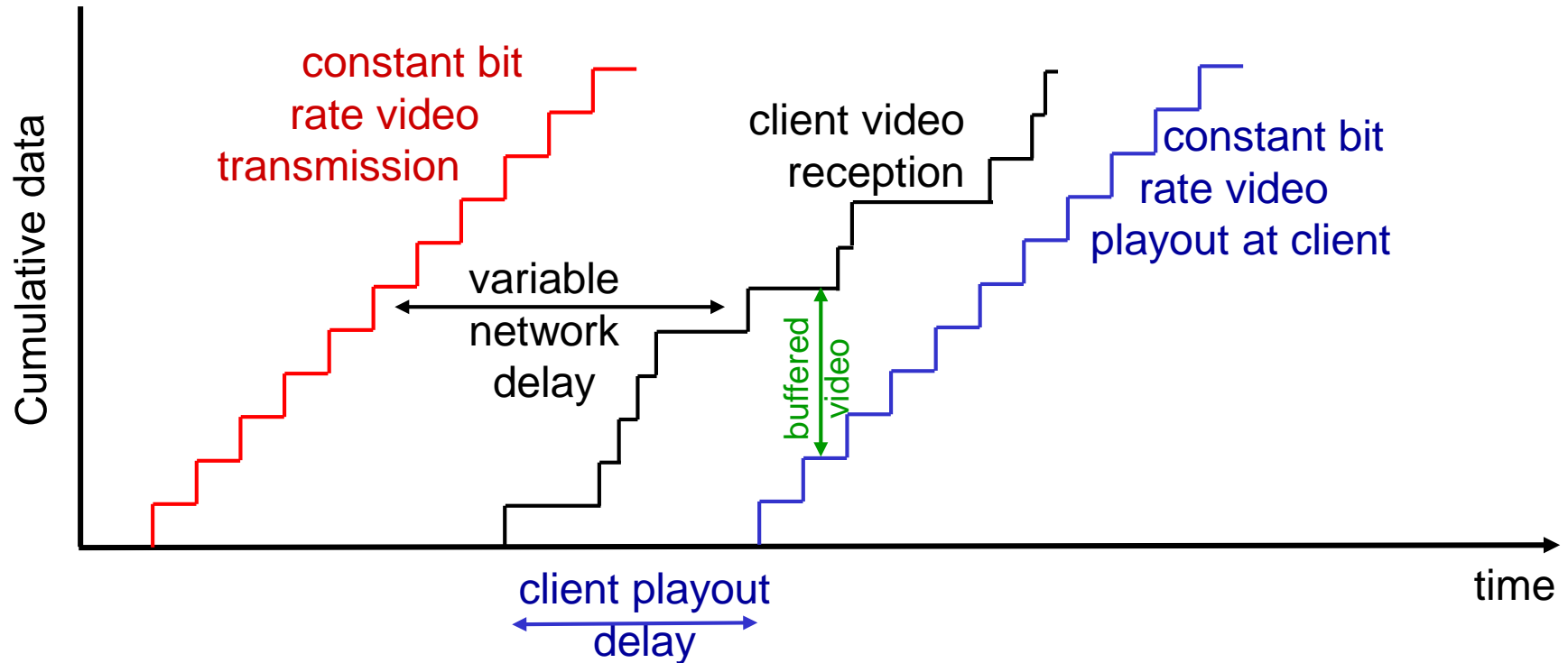
Streaming stored video:



Streaming stored video: challenges

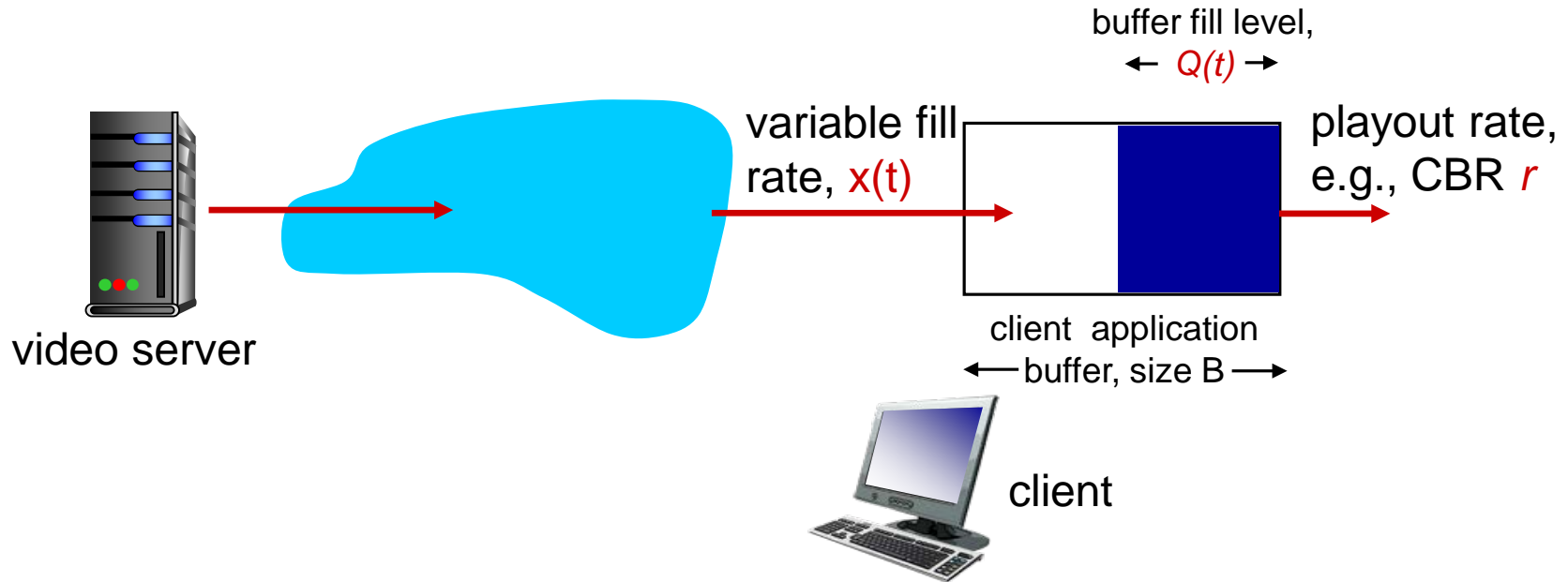
- **continuous playout constraint**: once client playout begins, playback must match original timing
 - ... but **network delays are variable** (jitter), so will need **client-side buffer** to match playout requirements
- other challenges:
 - client interactivity: pause, fast-forward, rewind, jump through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

Streaming stored video: revisited

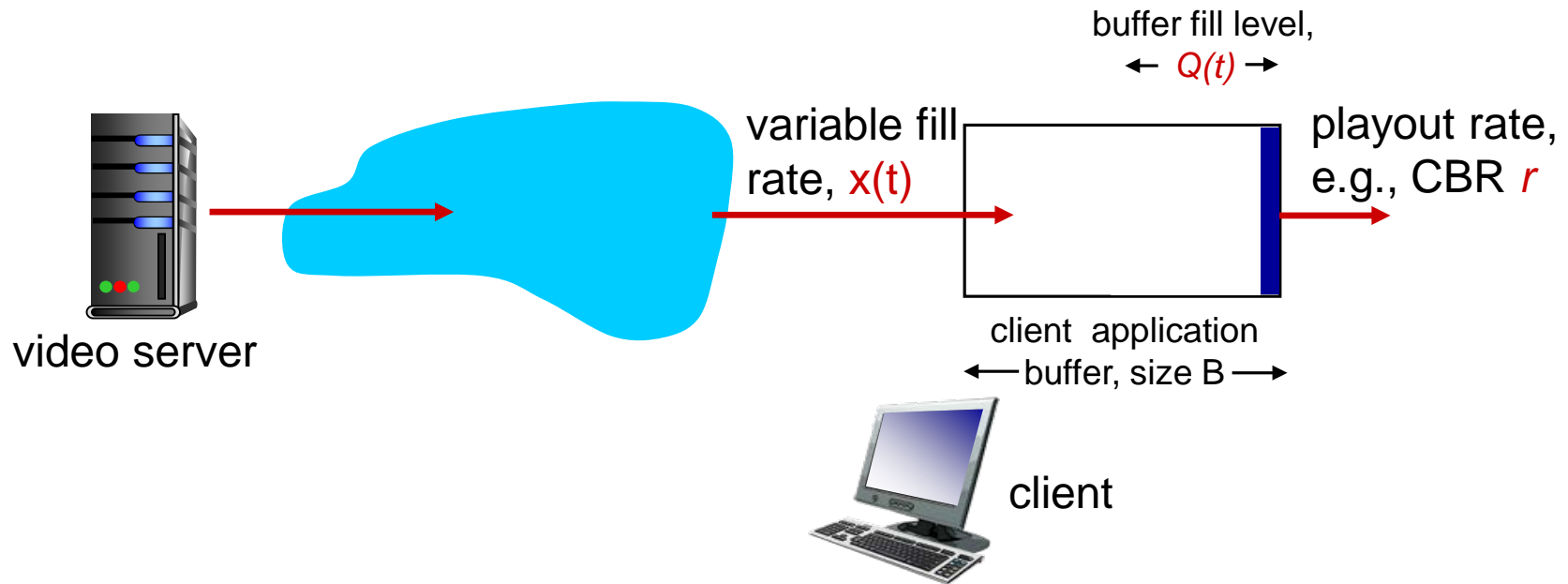


- *client-side buffering and playout delay*: compensate for network-added delay, delay jitter

Client-side buffering, playout

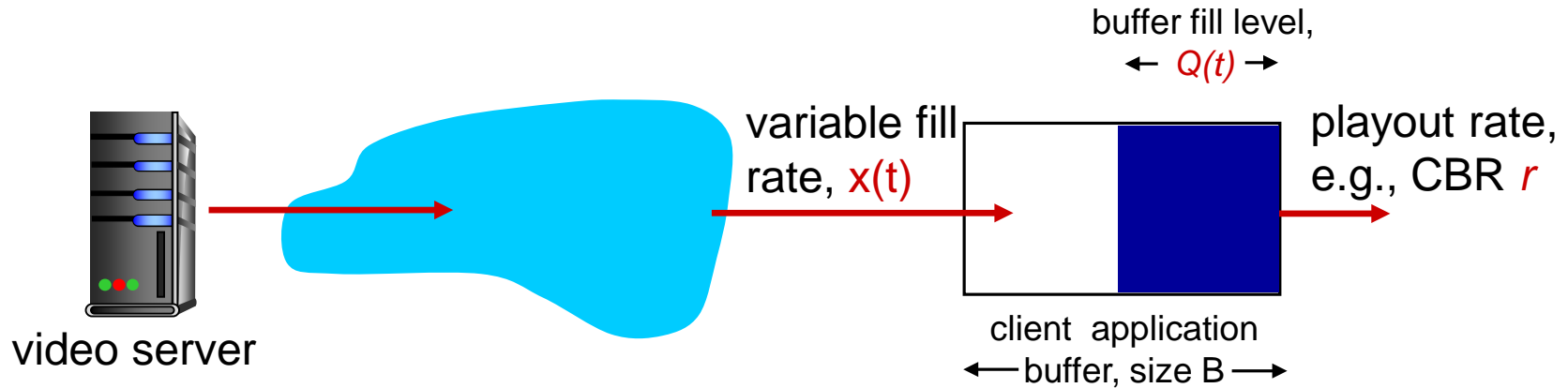


Client-side buffering, playout



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: DASH

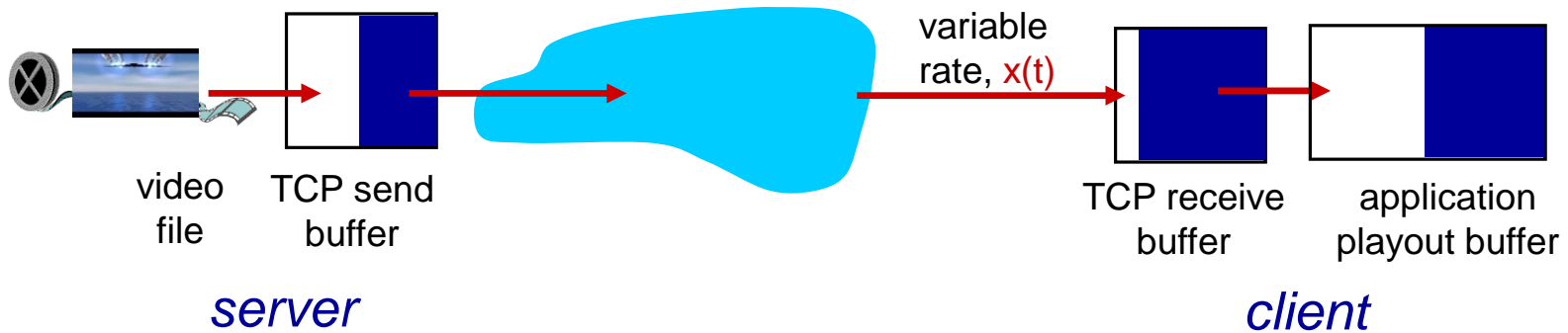
- *DASH*: *D*ynamic, *A*daptive *S*treaming over *H*TTP
- *server*:
 - divides video file into multiple chunks
 - each chunk stored, encoded at different rates
 - *manifest file*: provides URLs for different chunks
- *client*:
 - periodically measures server-to-client bandwidth
 - consulting manifest,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

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
- HTTP/TCP passes more easily through firewalls

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?

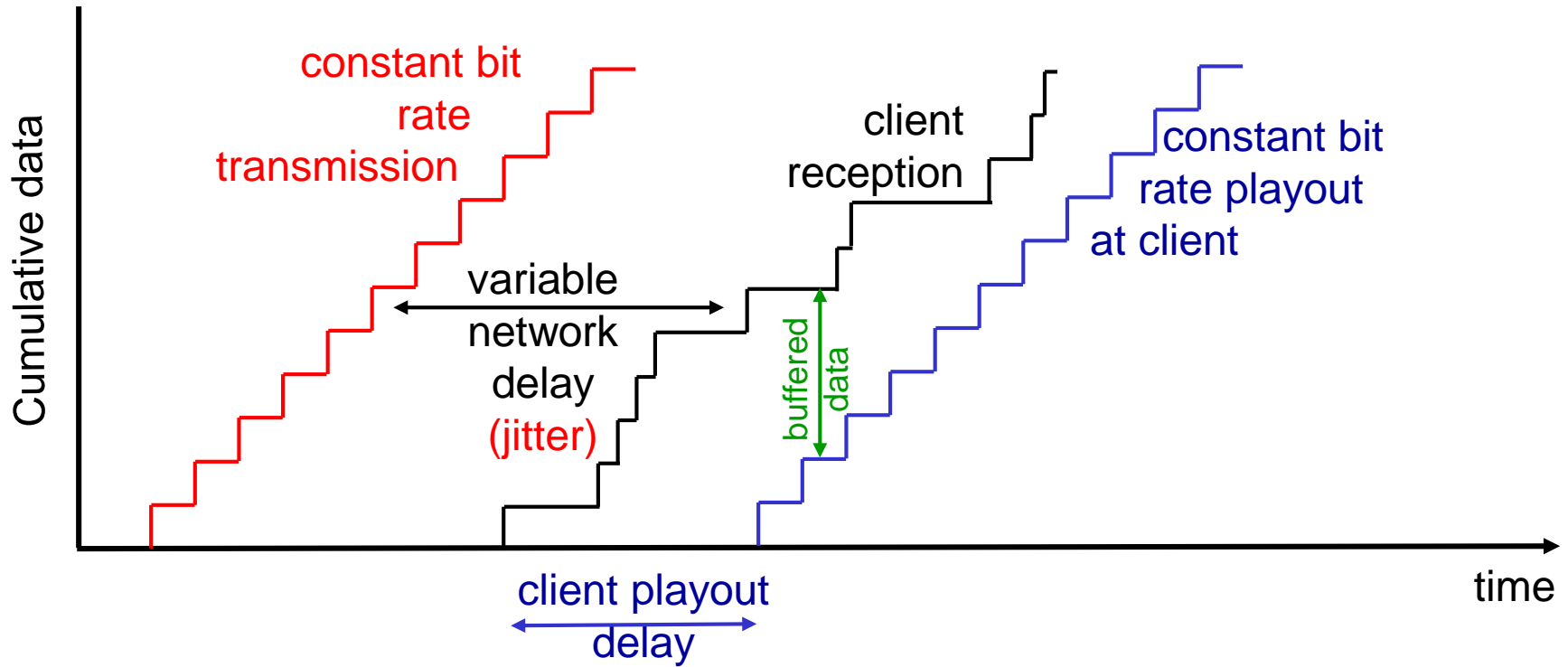
VoIP characteristics

- speaker's audio: alternating talk spurts, silent periods.
 - 64 kbps during talk spurt
 - packets generated only during talk spurts
 - 20 msec chunks at 8 Kbytes/sec: 160 bytes of data
- application-layer header added to each chunk
- chunk+header encapsulated into UDP (or TCP)
- 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 and loss concealment, packet loss rates between 1% and 10% can be tolerated

Delay jitter



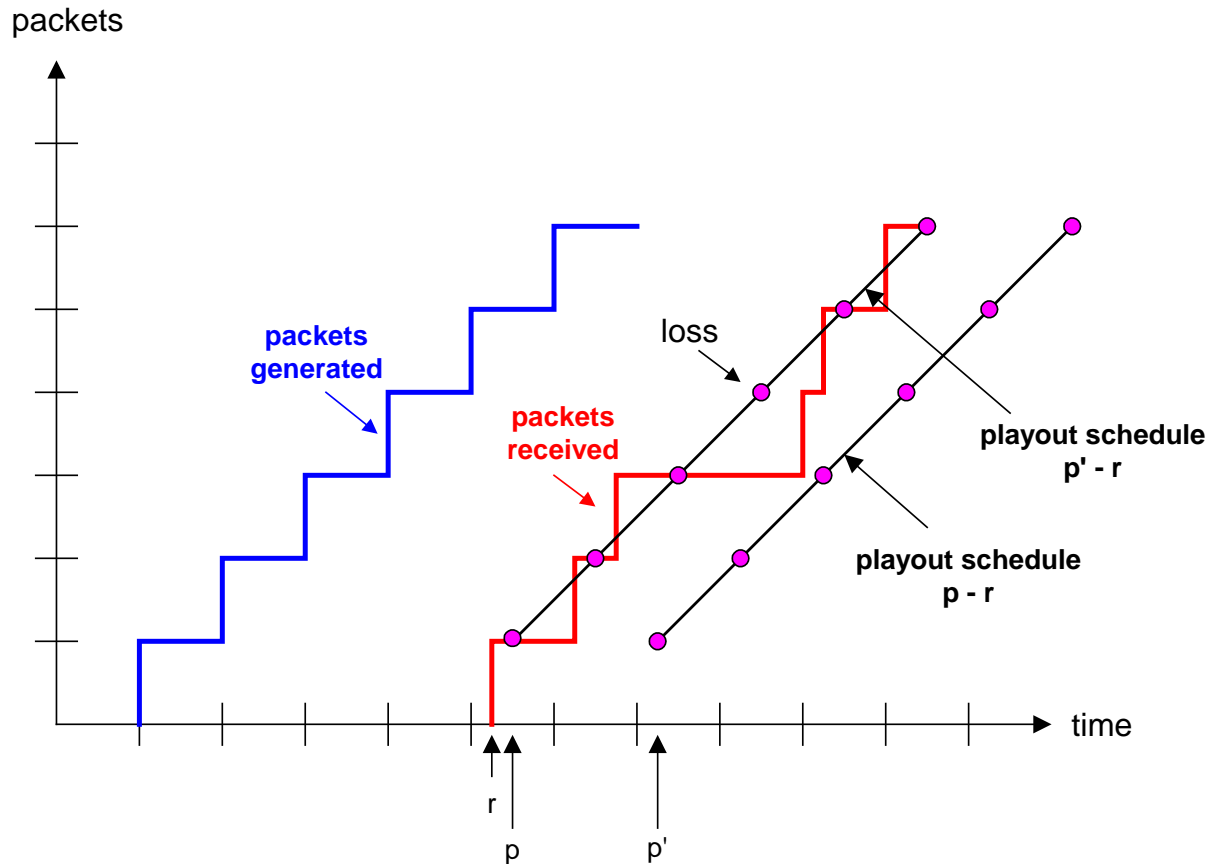
- end-to-end delays of two consecutive packets: difference can be more or less than 20 msec (transmission time difference)

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

- *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):

$$d_i = (1-\alpha)d_{i-1} + \alpha (r_i - t_i)$$

Diagram illustrating the components of the EWMA formula:

- d_i : delay estimate after i th packet
- $(1-\alpha)$: small constant, e.g. 0.1
- $r_i - t_i$: time received - time sent (timestamp)
measured delay of i th packet

VoiP: recovery from packet loss (I)

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

- each ACK/NAK takes \sim one RTT
- alternative: *Forward Error Correction (FEC)*
 - send enough bits to allow recovery without retransmission

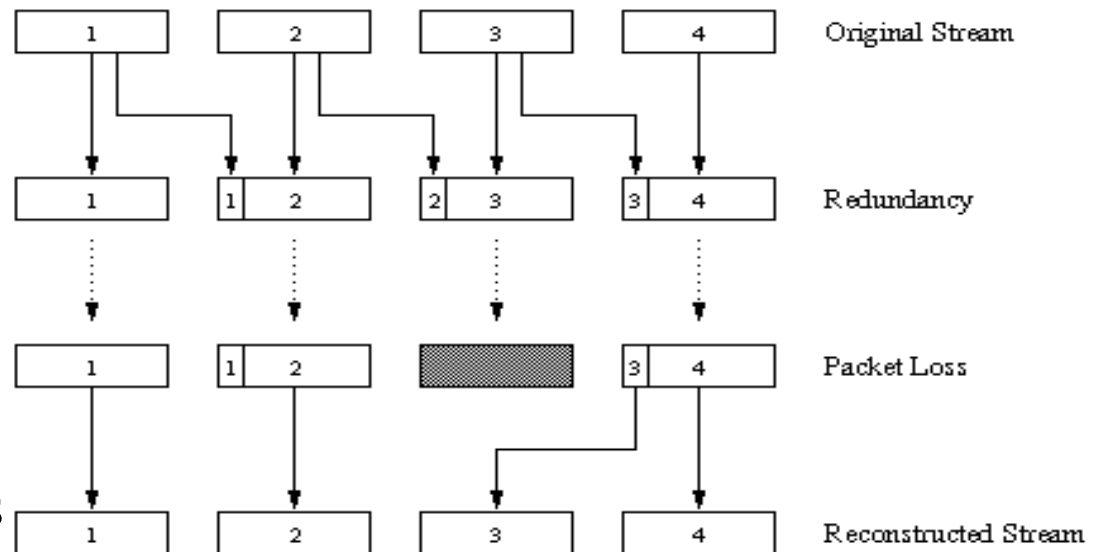
simple FEC

- 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

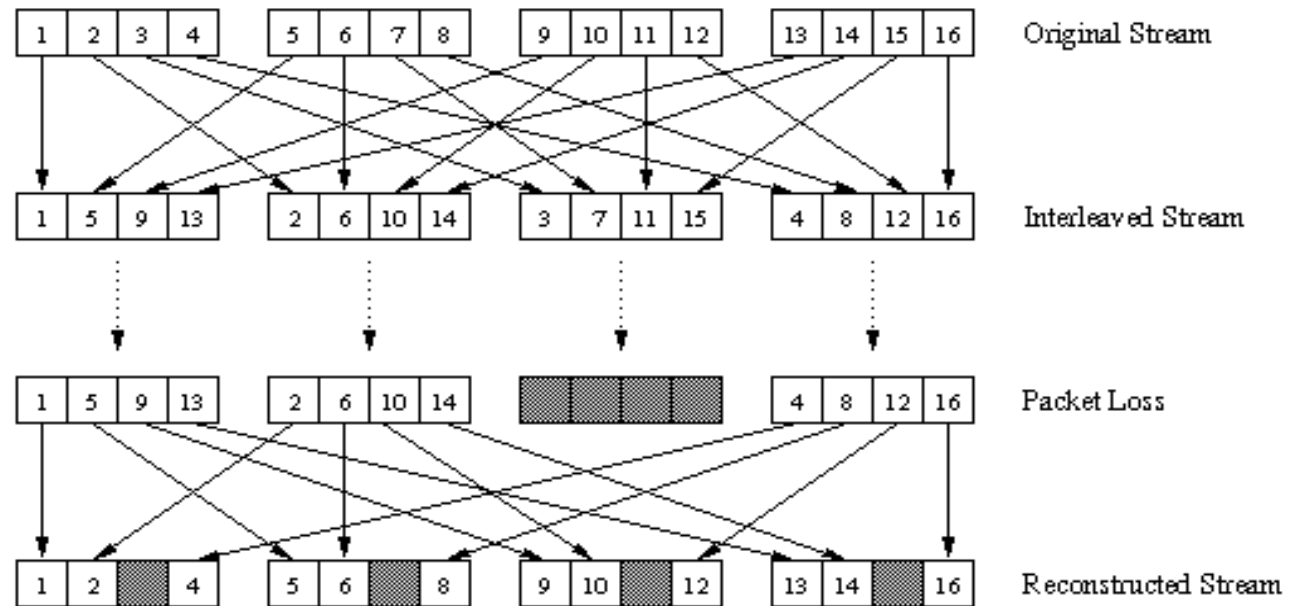
VoiP: recovery from packet loss (2)

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
- 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)

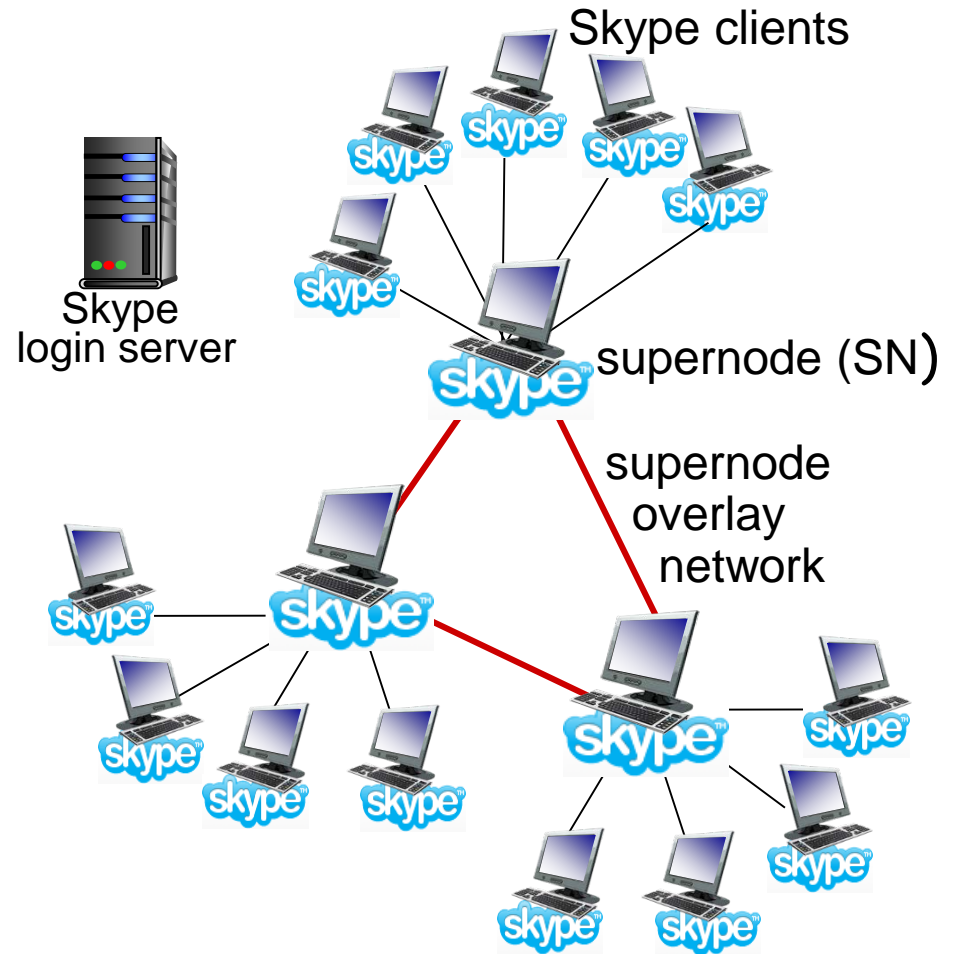


interleaving to conceal loss:

- 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
 - encrypted msgs
- P2P components:
 - **clients**: Skype peers connect directly to each other for VoIP call
 - **supernodes (SN)**: Skype peers with special functions
 - **overlay network**: among SNs to locate clients
 - **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 (username, password) to centralized Skype login server
3. obtains IP address for callee from SN overlay network
4. initiate call directly to callee

