

Ch.7 Multimedia networking

7.1 multimedia networking applications

7.2 streaming *stored* video

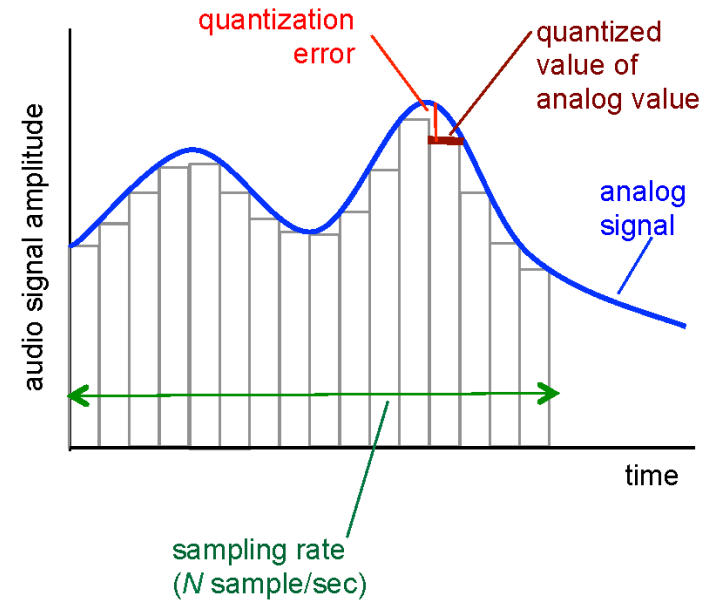
7.3 voice-over-IP

7.4 protocols for *real-time* conversational applications

7.5 network support for multimedia

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

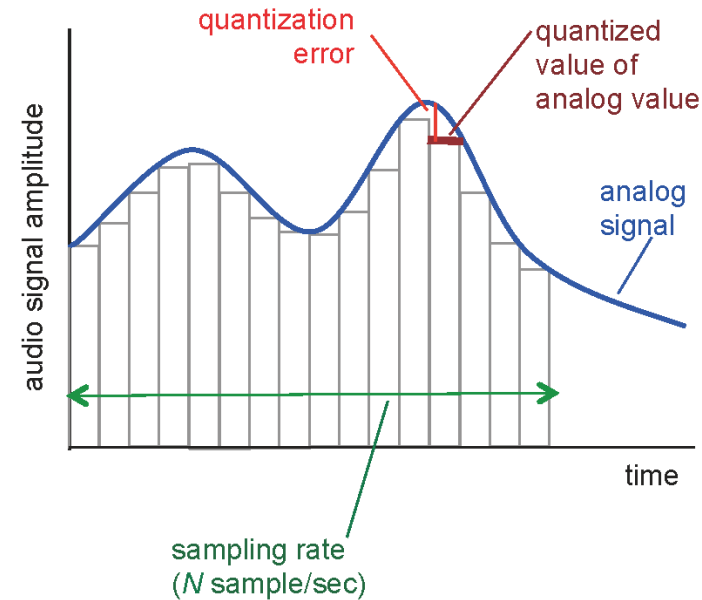


Multimedia: audio

- ❖ 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
- ❖ Internet telephony: 5.3 kbps and up



Multimedia: video

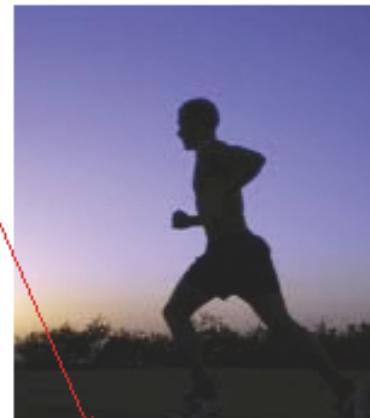
- ❖ video: sequence of images displayed at constant rate
 - e.g. 24 images/sec
- ❖ digital image: array of pixels
 - each pixel represented by bits
- ❖ coding: use redundancy *within* and *between* images to decrease # bits used to encode image
 - spatial (within image)
 - temporal (from one image to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

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

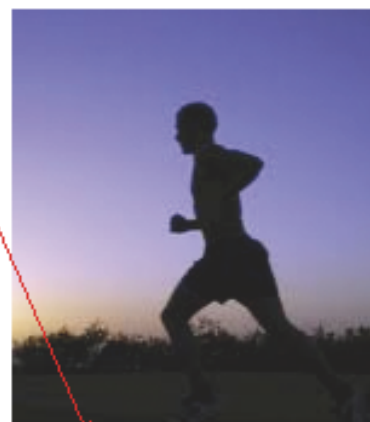
- ❖ *CBR: (constant bit rate):* video encoding rate fixed
- ❖ *VBR: (variable bit rate):* video encoding rate changes as amount of spatial, temporal coding changes
- ❖ *examples:*
 - MPEG 1 (CD-ROM) 1.5 Mbps
 - MPEG2 (DVD) 3-6 Mbps
 - MPEG4 (often used in Internet, < 1 Mbps)

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

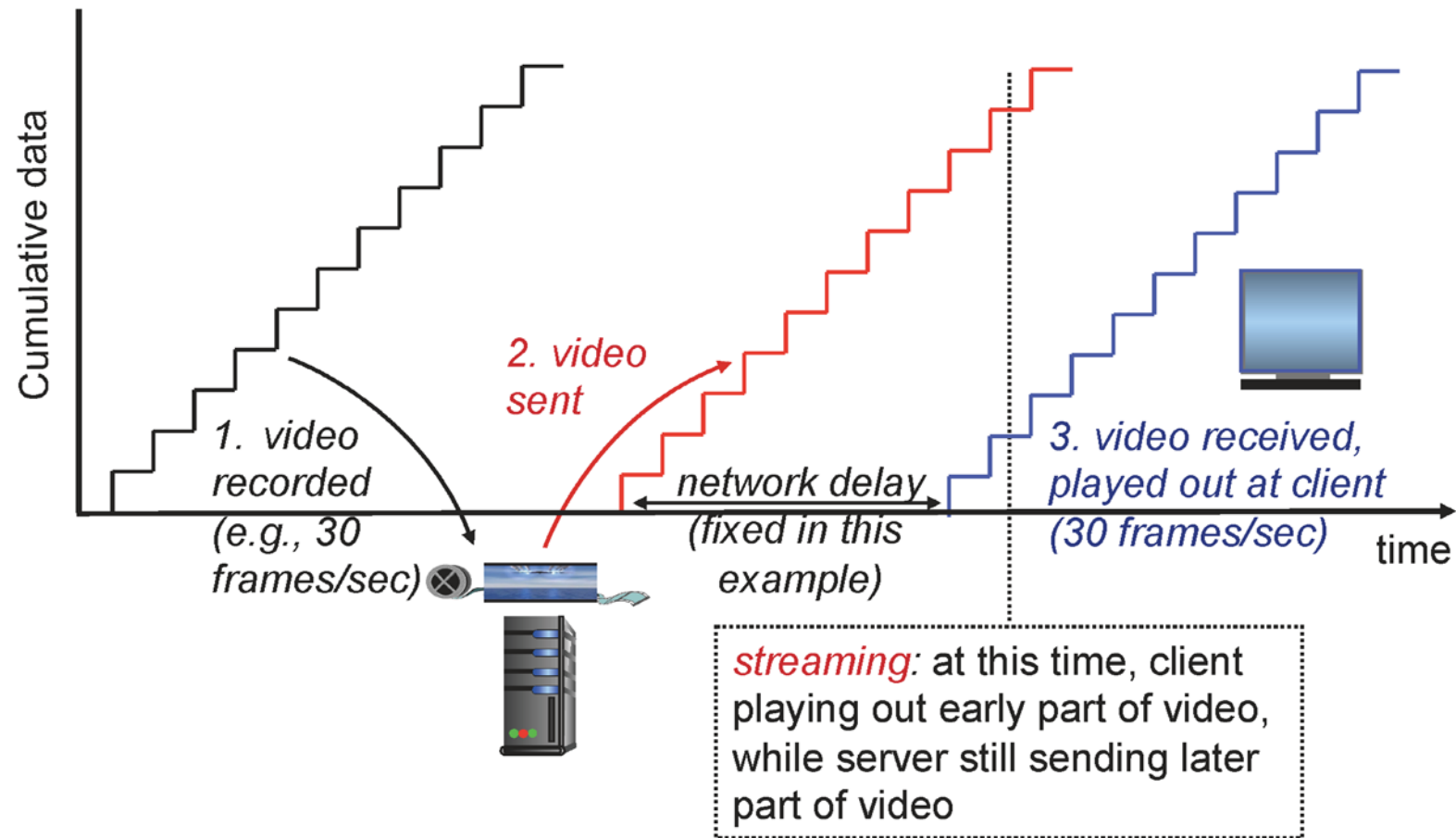


frame $i+1$

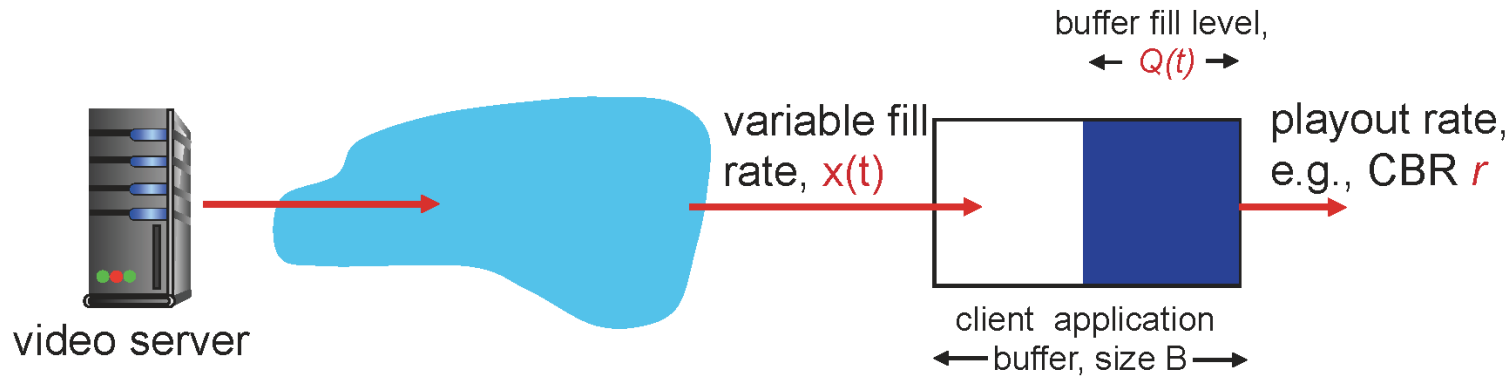
Multimedia networking: 3 application types

- ❖ *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:



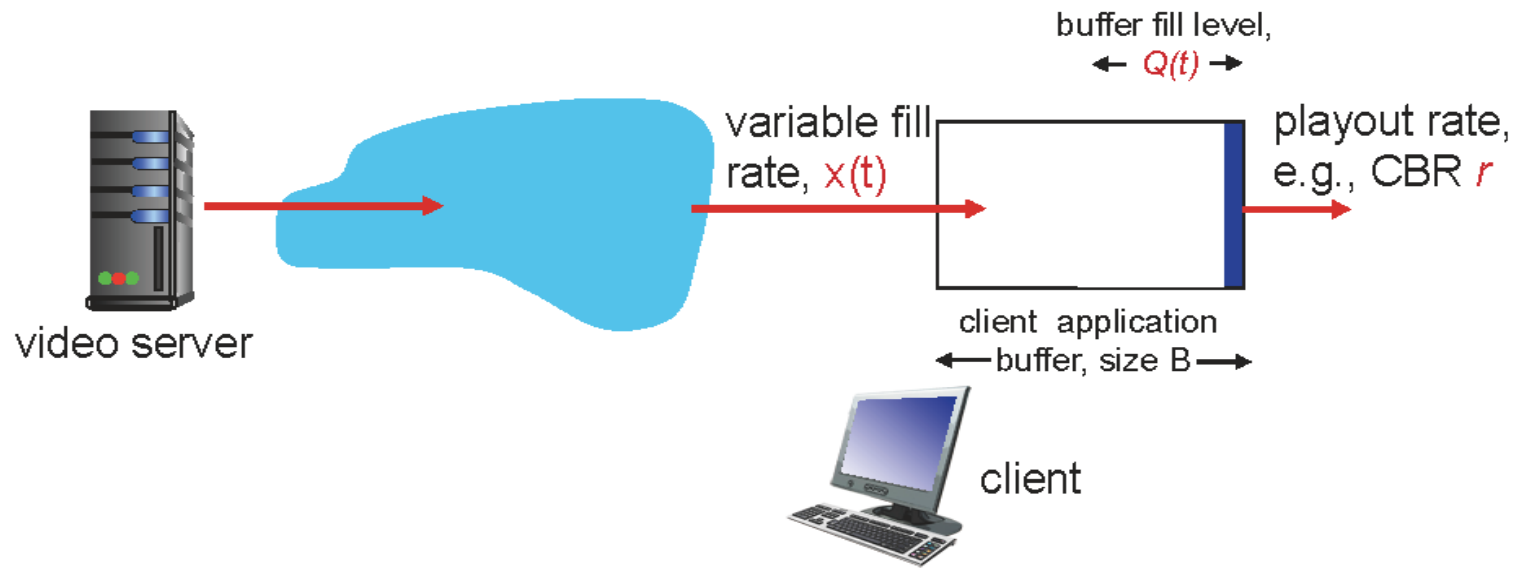
Client-side buffering, playout



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

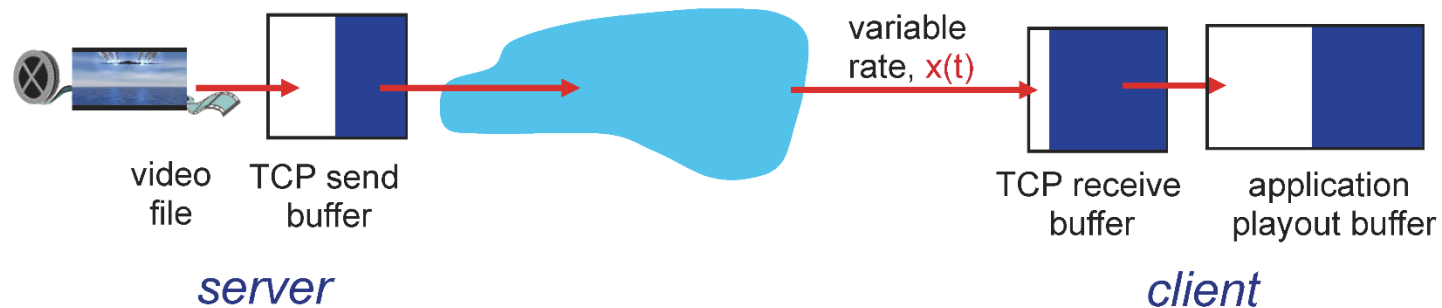
Client-side buffering, playout



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

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

Streaming multimedia: DASH

❖ *DASH: D*ynamic, *A*daptive *S*teaming 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)

Case study: Netflix

