Multimedia Networking

Prof. Anja Feldmann, Ph.D.

Balakrishnan Chandrasekaran, Ph.D.

(Based on slide deck of Computer Networking, 7th ed., Jim Kurose and Keith Ross.)

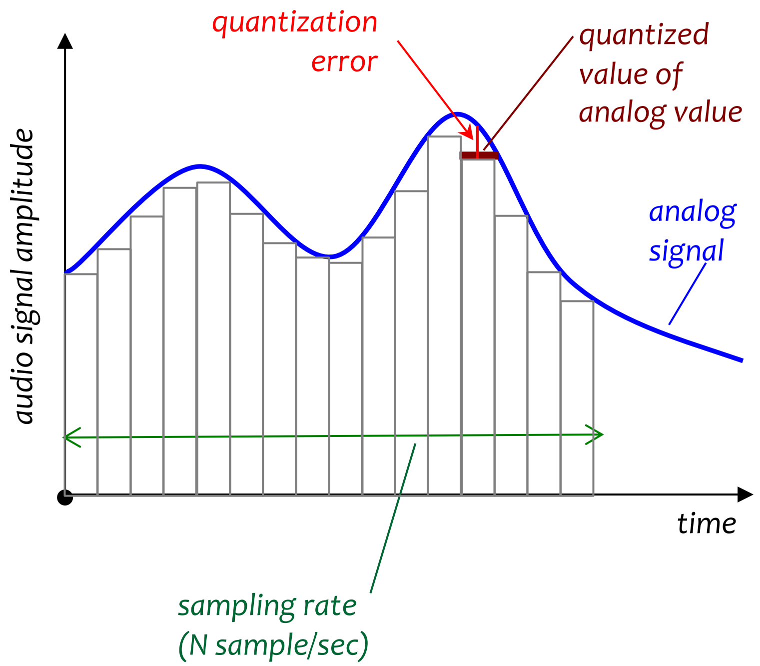
Agenda

- Multimedia Applications
- Streaming Stored Video
- Voice-over-IP

Multimedia Applications

Audio

- Analog audio signal sampled at *constant rate*
  - *Telephone: 8,000 samples/s*
  - *CD music: 44,100 samples/s*



- 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*

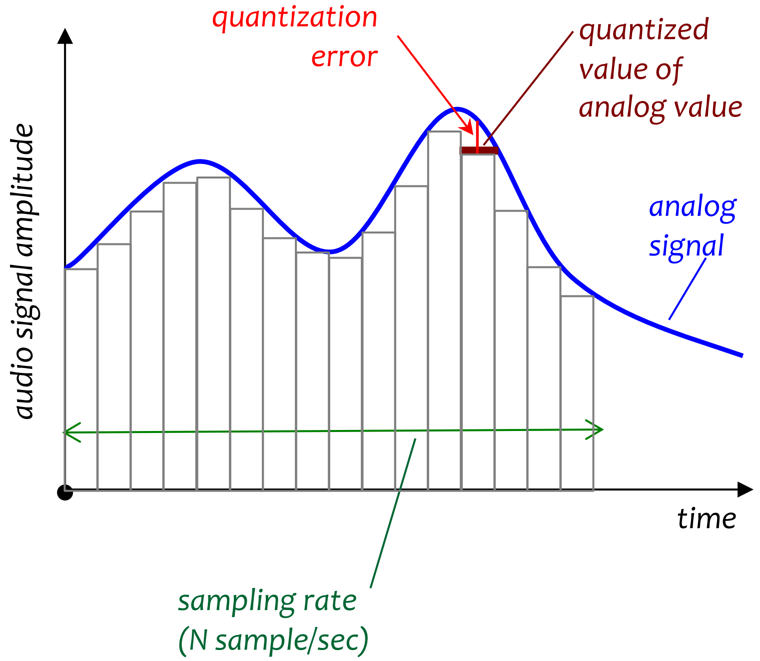
Audio

Example

- *8,000 samples/s ; 256 quantized values ⟹ 64,000 bps*

Receiver converts bits back to analog signal:

- Suffers *some* quality reduction



Example bitrates

- *CD: 1.411 Mbps*
- *MP3: 96, 128, 160 Kbps*
- *Internet telephony: 5.3 Kbps and upwards*

Video

Sequence of images (typically) displayed at constant rate

- *e.g., 24 images/s*

Digital image: array of pixels

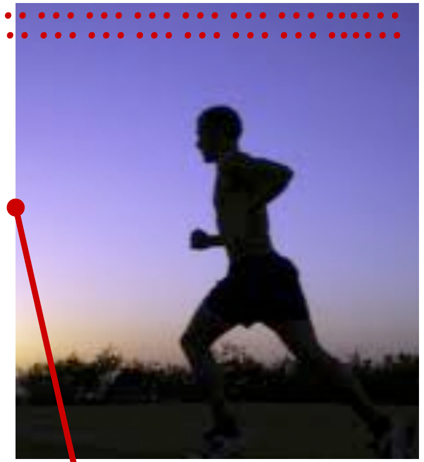
- Each pixel represented by bits

Coding: use redundancy *within* and *between* images to decrease the number of 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

Video

Constant bit rate (CBR)

- Video encoding rate is fixed

Variable bit rate (VBR)

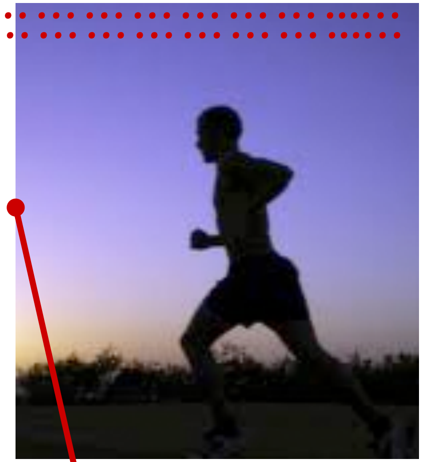
- Video encoding rate changes as the amount of spatial, temporal coding changes

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

Application types

Streaming stored audio, video

- ***Streaming***: Receiver can begin playout *before* downloading entire file
- ***Stored (at server)***: Can transmit faster than audio or video will be rendered (requires storing or *buffering* at client)
- *Examples: YouTube, Netflix, and Hulu*

Conversational voice or video over IP

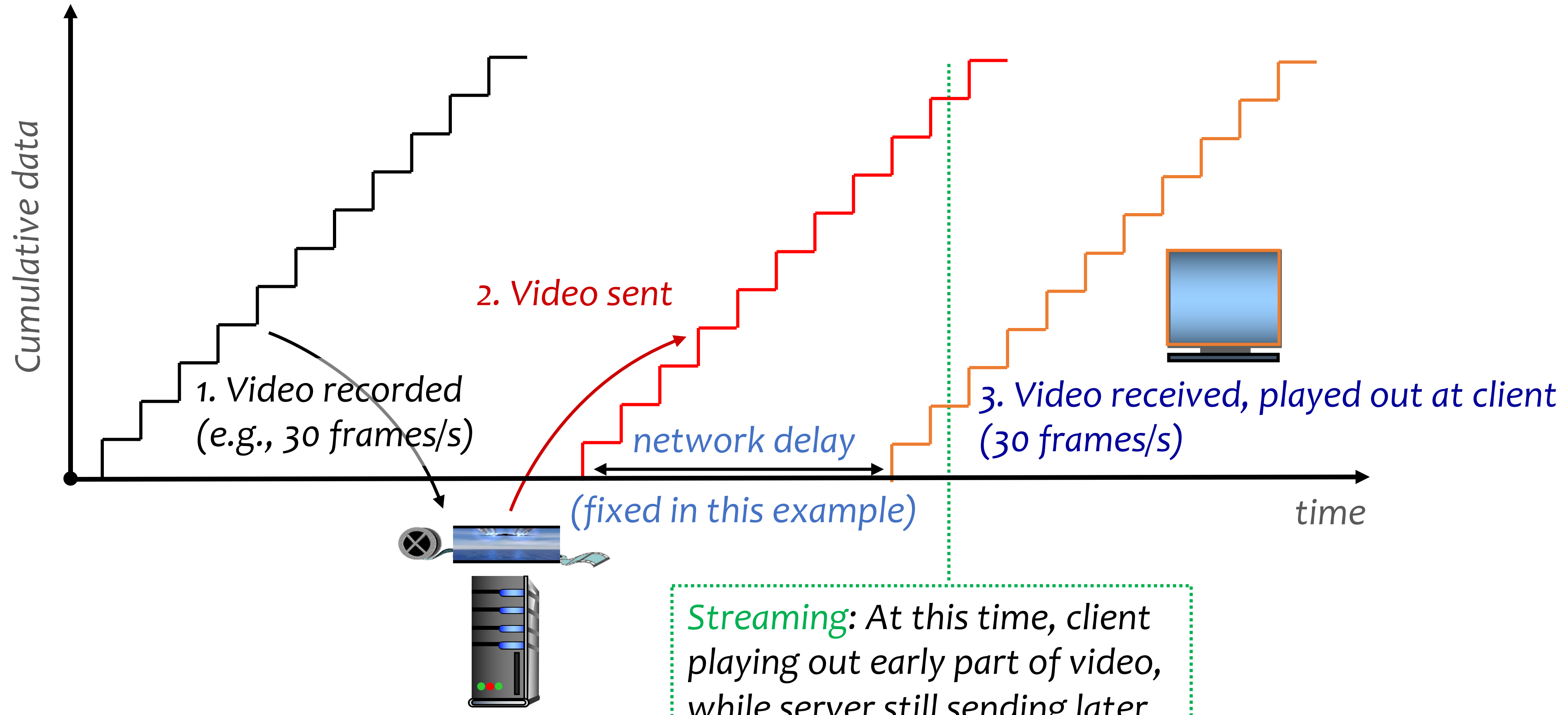
- Interactive nature of human-to-human conversation limits *delay tolerance*
- e.g., Skype

Streaming live audio, video

- e.g., Live sporting events

Streaming Stored Video

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

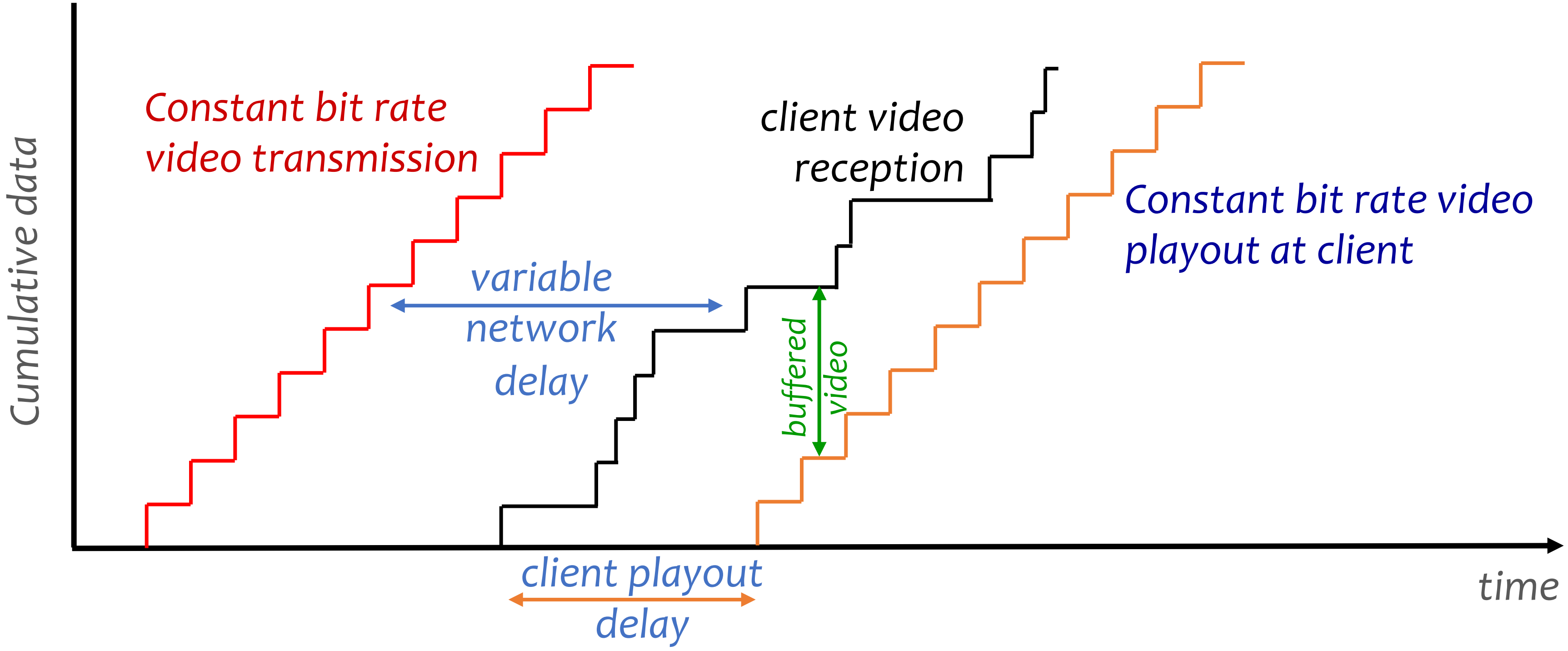
Continuous playout constraint

- *Once client playout begins, playback* ***must*** *match original timing*
- *… but* network delays are ***variable (jitter)***; need *client-side buffer* to match playout requirements

Other challenges

- Client interactivity: Pause, fast-forward, rewind, and jump through video
- Video packets may be lost or retransmitted

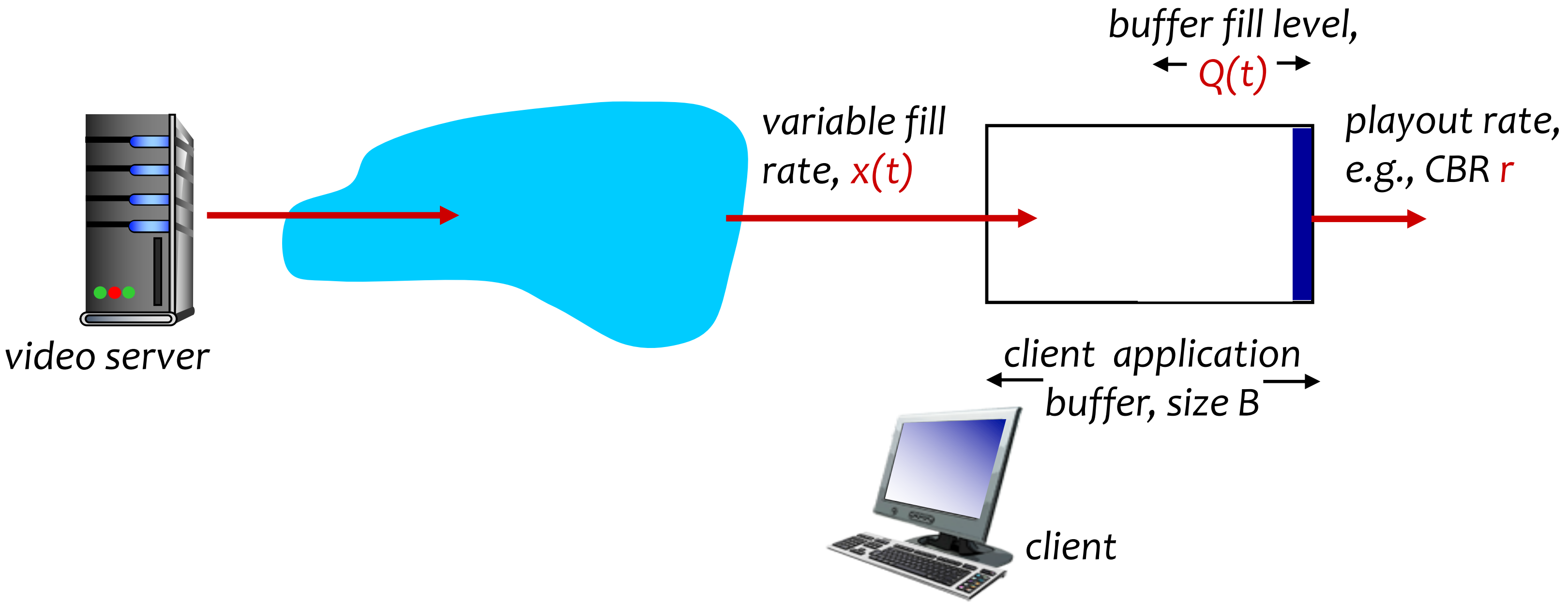
Streaming stored video: Revisited



Client-side buffering and playout delay

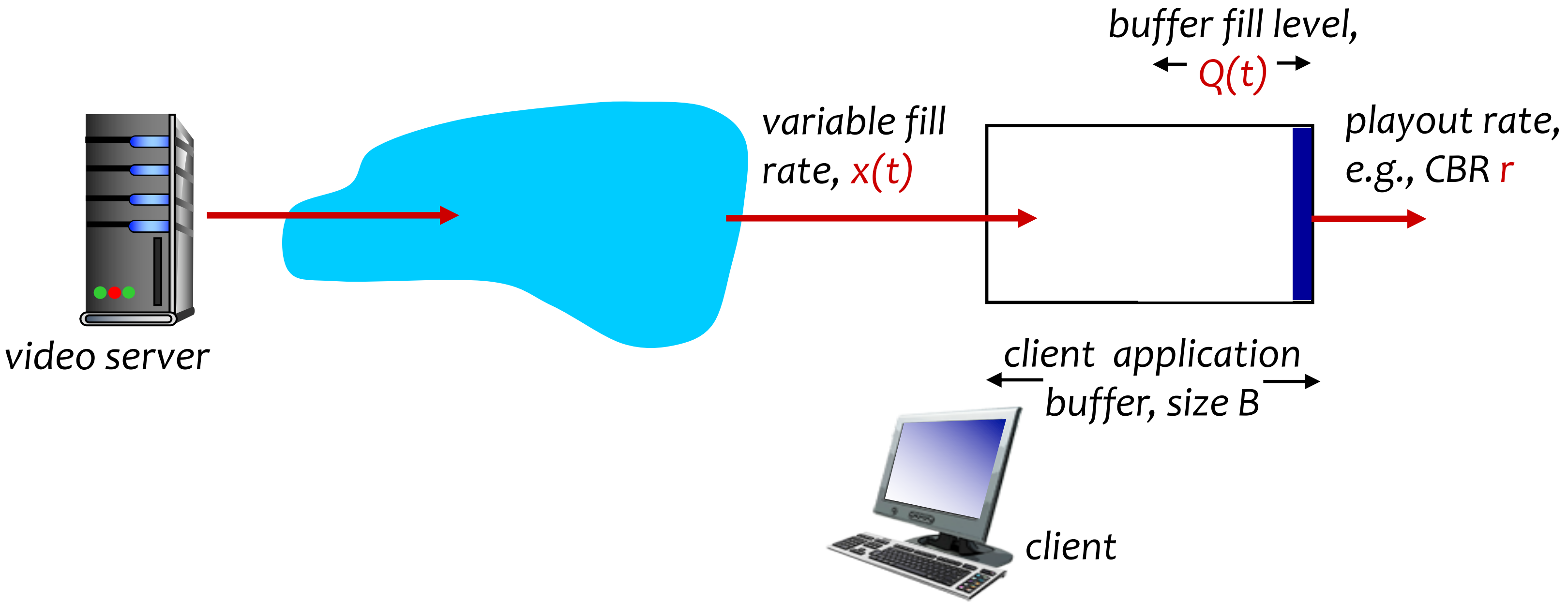
- Compensate for network-added delay and jitter

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

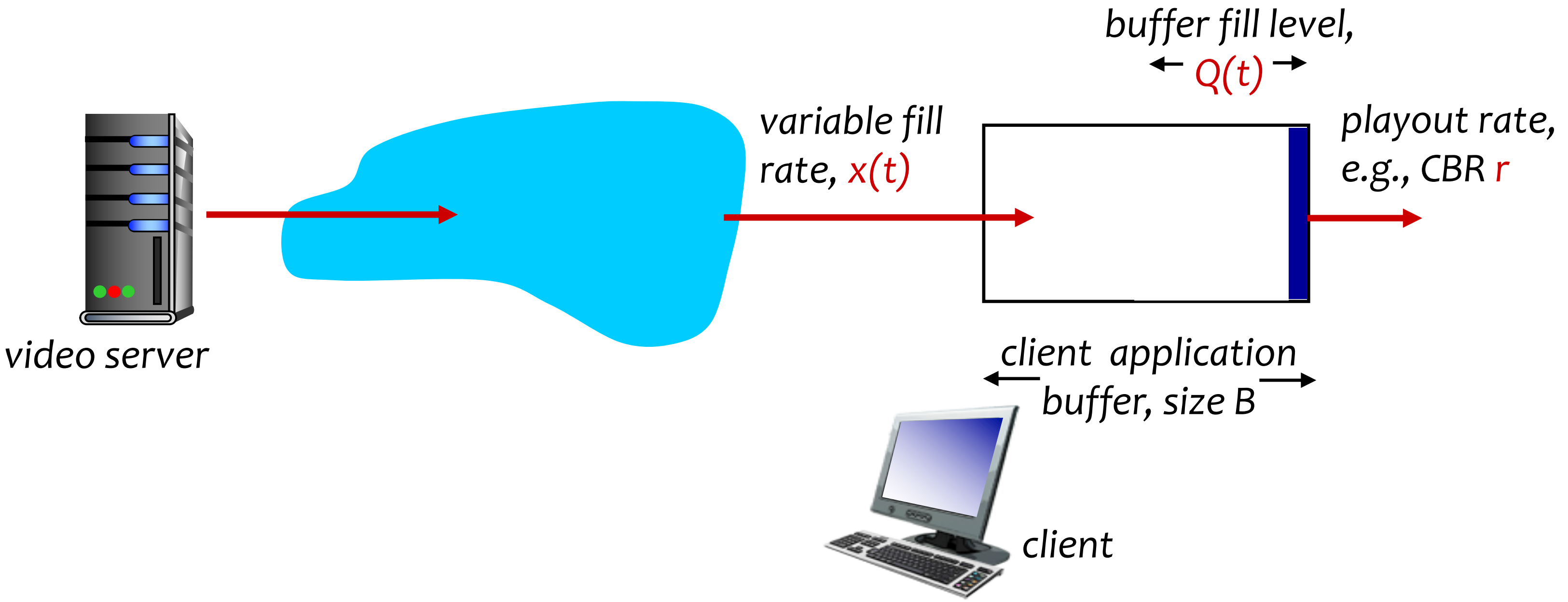
Client-side buffering, playout



Playout buffering: average fill rate (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)*

Client-side buffering, playout



Initial playout delay tradeoff

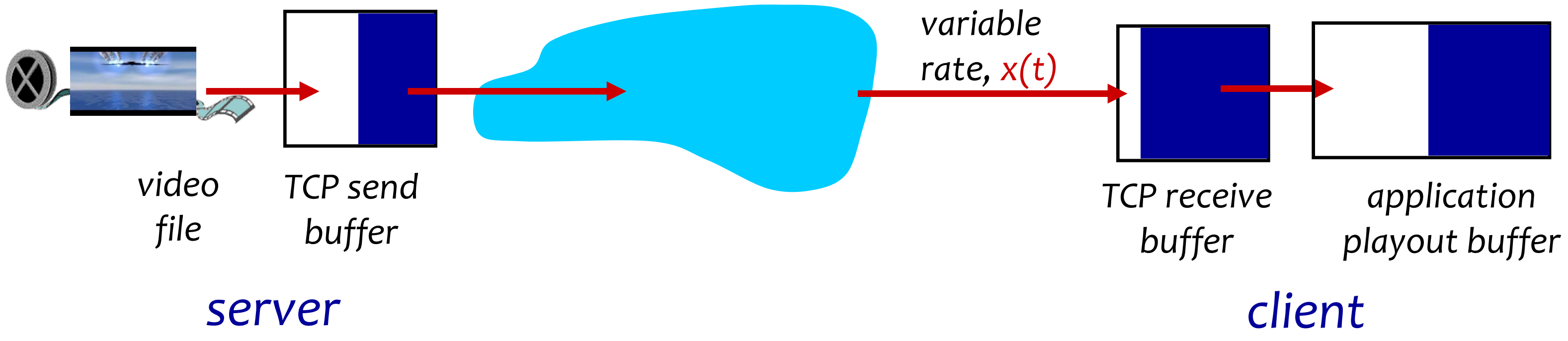
- Buffer starvation less likely with larger delay, but larger delay until user begins watching

Streaming: UDP

- Server sends at a 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: HTTP/TCP

- 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: Is TCP ill-suited?

Recall TCP's objective

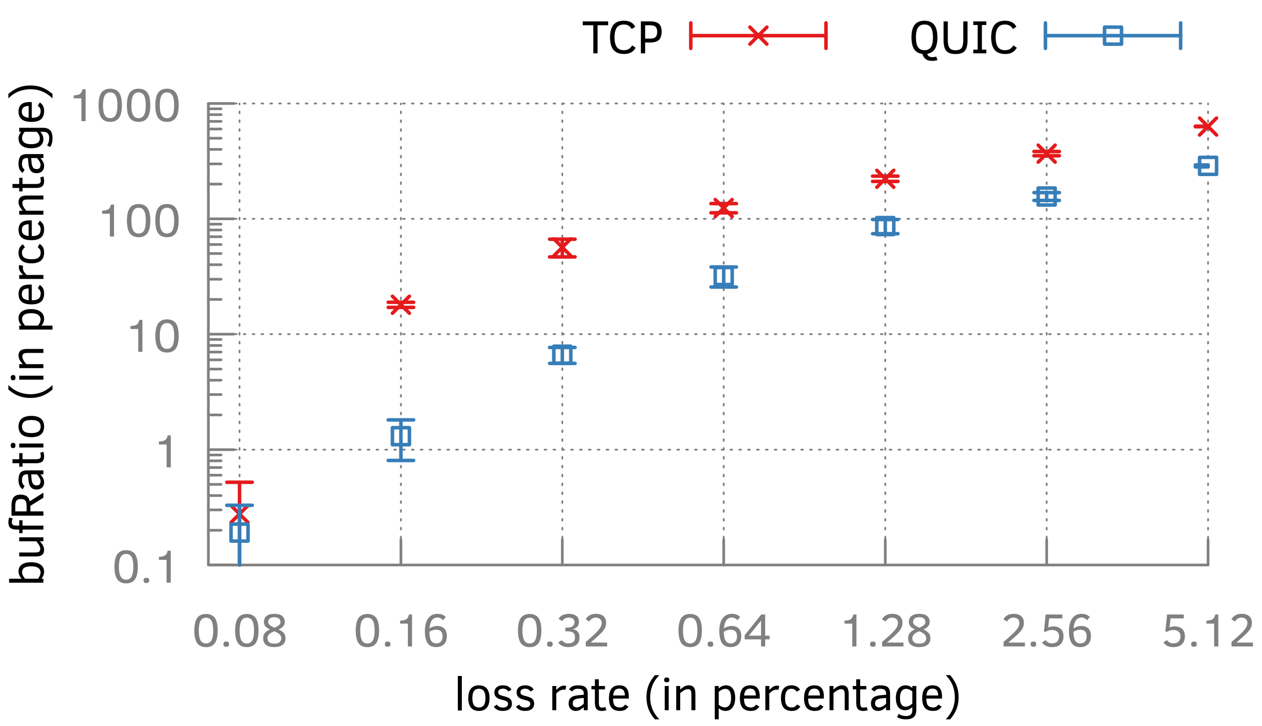
- *Reliably* deliver every packet (or video frame)

Wait, what are "frames"?

- ***Frame***: One of the many still images which compose the video

- So when streaming video using TCP, we assume …
  - Each frame ***must*** be reliably delivered by the server to the client
  - … even when network conditions are not ideal!

Streaming: TCP is ill-suited …



Streaming: A peek at an ideal solution

A closer look at ***frame types***

- ***I-Frames***: independent; do not require other frames for decoding
- ***P-Frames***: use data from prior frames for compression; need previous frames for decoding
- ***B-Frames***: use both previous and forward frames

In case of packet losses …

- Use I-Frames to completely recover from loss (think of it loss reset)
- Loss in P- and B-Frames do not affect end user's video watching experience (think of QoE)

Ideal solution: Selectively or partially reliably transport

- *Transmit I-Frames reliably, but P- and B-Frames unreliably!*
- What about losses? Forward error correction or selective retransmissions.

Streaming: A peek at an ideal solution

Contact Mirko Palmer (*mpalmer @ mpi-inf.mpg.de*) for more details!

Summary

- Multimedia Applications
  - Rich and complex use cases, many challenges
- Streaming Stored Video
  - Buffering, playback, TCP and UDP