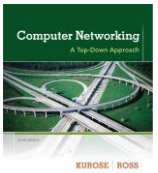
Chapter 7
Multimedia Networking



Computer Networking: A Top Down Approach
6th edition
Jim Kurose, Keith Ross
Addison-Wesley
March 2012

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

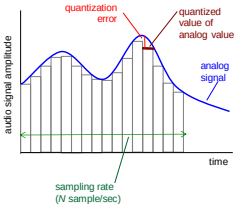
The course notes are adapted for CSCI 363 at Bucknell Spring 2016, Xiannong Meng

Multimedia networking: outline

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
  - music CD : 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

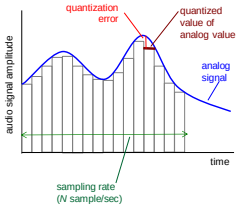


Multimedia: audio

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

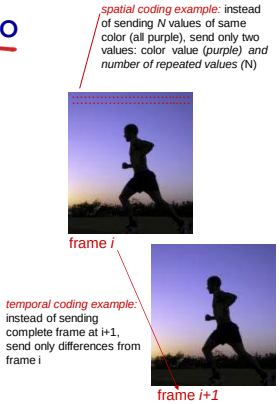
example rates

- CD: 1.411 Mbps (44,100 samples/s, 16 bit/s or 705.6 kbps for mono, 32bit/s or 1.411mbps for stereo)
- MP3: 96 kbps, 128 kbps, 160 kbps
- Internet telephony: 5.3 kbps and up



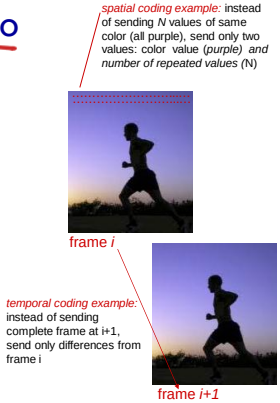
Multimedia: video

- 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)



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)

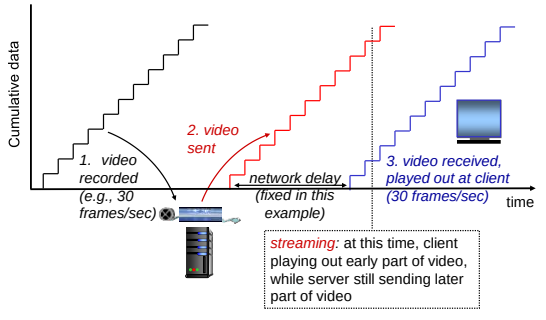


Multimedia networking: 3 application types

- *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 (interactive)* 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)

Multimedia networking: outline

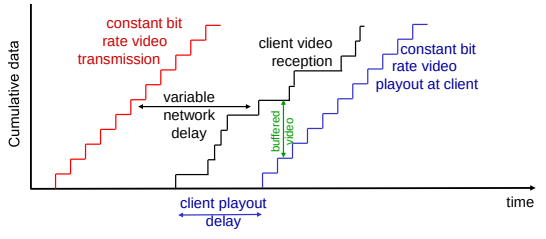
Streaming stored video:



Streaming stored video: challenges

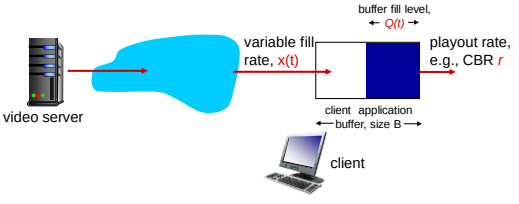
- *continuous playout constraint*: once client playout begins, playback must match original timing
  - … but *network delays are variable* (jitter), so will need *client-side buffer* to match playout requirements
- other challenges:
  - client interactivity: pause, fast-forward, rewind, jump through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

Streaming stored video: revisted

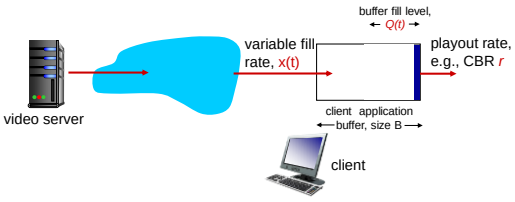


- *client-side buffering and playout delay:* compensate for network-added delay, delay jitter

Client-side buffering, playout

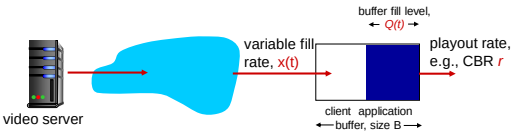


Client-side buffering, playout



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 (~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 until user begins watching

Streaming multimedia: UDP

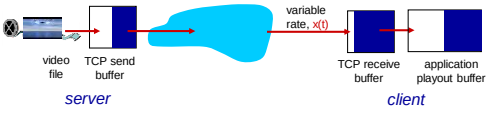
- server sends at rate appropriate for client
  - often: send rate = encoding rate = constant rate
  - transmission rate can be oblivious to congestion levels
- short playout delay (2-5 seconds) to remove network jitter
- error recovery: application-level, timeipermitting
- RTSP (Real Time Streaming Protocol) [RFC 2326]: multimedia payload types
- UDP may *not* go through firewalls

UDP stream example and issues

- Video consumption rate: 2 Mbps (given)
- Then the server would transmit one UDP packet full of data every (8000 bits)/(2 Mbps) = 4 ms, assuming each packet contains 8000 bits data
- Some issues:
  - UDP doesn't handle variable network bandwidth well
  - UDP streaming requires a media control server (e.g., RTSP server) to process client-server interaction, and to track client state
  - Many firewalls are designed to block UDP traffic

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

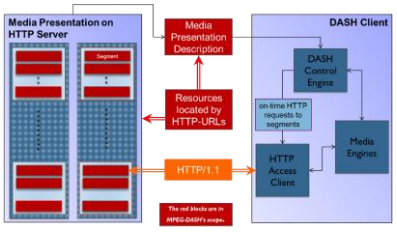
Streaming multimedia: DASH

- *DASH: Dynamic, Adaptive Streaming over HTTP* (a.k.a. MPEG-DASH)
- *"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: DASH

- *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)

MPEG-DASH structure



http://dashif.org/mpeg-dash/