eecs 489 COMPUTER NETWORKS

Lecture 33:
Multimedia Applications

Network Service Model

What kinds of delivery service might an application want from the network?

Example services for individual packets:
- guaranteed delivery
- guaranteed delivery within 40 msec of transmission

Example services for a flow of packets:
- in-order delivery
- guaranteed minimum bandwidth (bytes/time unit)
- a bound on delay jitter (inter-packet delay spacing)

Network Delivery Variables

Loss
- some apps (e.g., audio, video) can tolerate some loss
- other apps (e.g., text transfer, remote access) require 100% reliable data transfer

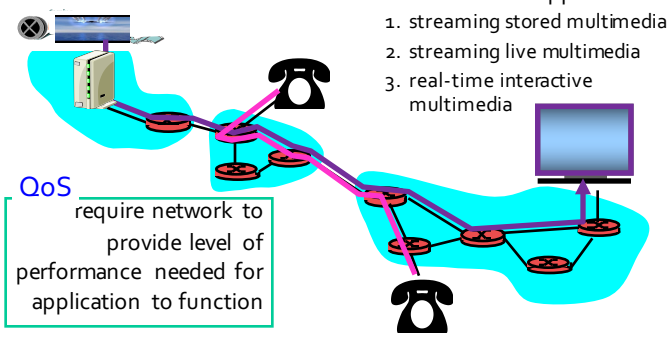
Timing
- some apps (e.g., Internet telephony, interactive games) require low delay to be "effective"

Bandwidth
- some apps (e.g., multimedia) require minimum amount of bandwidth to be "effective", and does not send more than a maximum rate
- other apps ("elastic apps", e.g., file transfer, email, web browsing) make use of whatever bandwidth they can get

Multimedia Apps Quality of Service

Multimedia applications: networked audio and video, distributed interactive worlds (multiplayer gaming)

Classes of MM applications:
1. streaming stored multimedia
2. streaming live multimedia
3. real-time interactive multimedia



QoS
require network to provide level of performance needed for application to function

QoS Requirements of Common Apps

Application	Data loss	Bandwidth	Time Sensitive
file transfer	no loss	elastic	same day
e-mail	no	elastic	same day
Web documents	no	elastic	interactive
instant messaging	no	elastic	interactive
live audio/video	loss-tolerant	audio: 5kbps-1Mbps video: 512kbps-6Mbps	yes, 100's msec
stored audio/video	loss-tolerant	10kbps-6Mbps	yes, few secs
interactive games	loss-tolerant	same as above	yes, 100's msec
		few kbps and up	

Multimedia Applications

Multimedia over Today's Internet

TCP/UDP/IP: "best-effort service"
no guarantees on delay, loss

But you said multimedia apps require QoS and level of performance to be effective!

Shouldn't multimedia traffic be given higher priority?
Why can't I make resource reservation and get quality of service (QoS) guarantees?

Whither Network Support for Multimedia Traffic?

Mechanisms needed to support QoS:
- Reservation protocol (RSVP)
- Admission control
- Packet classification
- Real-time scheduling
- Pricing scheme
- Accounting and billing

Proposed real-time services (defunct):
- Integrated Services: negotiated per flow
- Differentiated Services: negotiated per AS
- why not deployed?
  - over-provisioning of Internet backbone led to 2% utilization
  - no compelling case for deployment of real-time services
  - bottleneck is at the access networks, but no competitive choices

Multimedia Over Today's Internet

TCP/UDP/IP: "best-effort service"
no guarantees on delay, loss

But you said multimedia apps requires QoS and level of performance to be effective!

Today's Internet multimedia applications use application-level techniques to mitigate effects of delay, loss

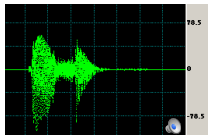
Multimedia Applications

Two broad categories:
- networked audio and video
- distributed interactive worlds (multiplayer gaming)

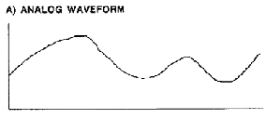
Classes of MM applications:
1. streaming stored multimedia
2. streaming live multimedia
3. real-time interactive multimedia

Signal Digitization: Overview

Analog audio signal:



Sampling: reading the signal at certain rate to collect samples



Signal can be reconstructed from the samples

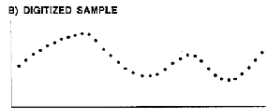
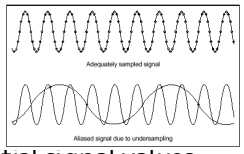


FIGURE 1: Analog waveform and digital equivalent

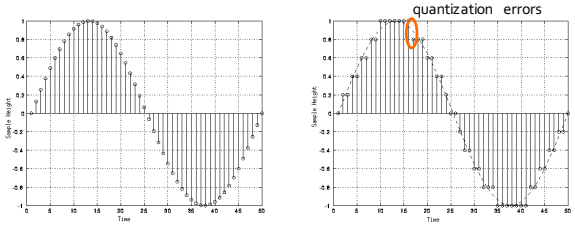
[http://www.naturenorth.com/spring/sound/shfrsnd.html]

Signal Digitization: Overview

Inadequate sampling can lead to aliasing:



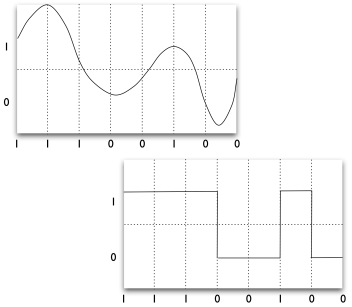
Quantization: partitioning potential signal values into levels and represent each level with a single number



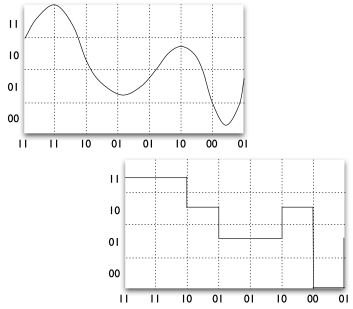
Bandwidth Requirement

Higher sampling rate and finer quantization levels give better signal reconstruction but require higher bandwidth

Example 1:
- sampling rate: 8 samples/sec
- quantization: 2 levels ⇒ 1 bit per sample
- bandwidth requirement: 8 bps



Example 2:
- sampling rate: 8 samples/sec
- quantization: 4 levels ⇒ 2 bit/sample
- bandwidth requirement: 16 bps



Audio Bandwidth Requirements

Several popular encoding standards:

- PCM (landline):
  - 8,000 samples/sec
  - 8 bits/sample (256 quantization levels)
  - ⇒ 64 Kbps data sent at the rate of 160 bytes/packet, once every 20 ms
- CD music:
  - 44,100 samples/sec
  - 16 bits/sample
  - ⇒ 705.6 Kbps mono, 1.411 Mbps stereo

Audio compression standards:

- GSM: 13 Kbps
- G.729: 8 Kbps
- G.723: 6.4 and 5.3 Kbps
- MPEG2 Layer 3 (MP3): 128 or 112 Kbps
- MPEG4/AAC: 96-128 Kbps, High-Efficiency AAC: 32-48 Kbps
- proprietary standards: Sorenson, Nellymoser

Video

Video is a sequence of images/frames displayed at a constant rate (moving pictures)

Digital image is an array of pixels, each pixel represented by bits

Examples:

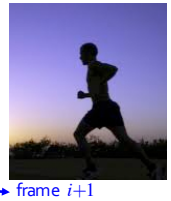
- single frame image encoding: 1024x1024 pixels, 24 bits/pixel ⇒ 3 MB/image
- movies: 24 frames/sec ⇒ 72 MB/sec
- TV: 30 frames/sec ⇒ 90 MB/sec
- Lower resolution requires less bandwidth, VHS quality 96 Mbps

Video Bandwidth Requirements

Sequence of images contains redundancy

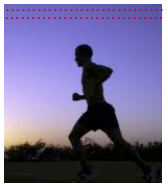
- spatial redundancy within an image
- temporal redundancy across images

Video compression works by removing spatial and temporal redundancies



frame i → frame $i+1$

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), a.k.a., run-length encoding



temporal coding example: instead of sending complete frame at $i+1$, send only differences from frame i

Video Encoding Bit Rate

Constant bit rate (CBR) encoding: encoding rate fixed

- variable quality, lower quality, but easy to provision for

Variable bit rate (VBR) encoding: encoding rate changes as amount of potential spatial and temporal compression varies

- non-deterministic bandwidth requirement, but constant quality, better quality

Video Bandwidth Requirements

Video compression standards:

- MPEG1: 1 Mbps (CD-ROM, VCD: VHS quality)
- MPEG2: 3-9 (usually 6) Mbps (DVD quality)
- MPEG4/H.264: 512 Kbps to 6 Mbps (HDTV quality)
- MPEG7/H.265: 256 Kbps to 3 Mbps
- proprietary standards: On2, Windows Media
- Scalable H.264: video encoded into several layers, cumulatively each additional layer gives higher resolution

Characteristics of Multimedia Apps

Typically delay sensitive

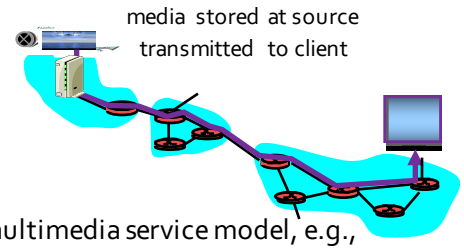
- live audio < 150 msec end-to-end delay is not perceptible
- 150-400 msec end-to-end delay is tolerable
- > 400 msec end-to-end delay makes audio unintelligible
- streaming audio: can wait 5 secs or more before starts of playback
- low variability of packet delays (jitter) within the same packet stream

But loss tolerant: infrequent losses cause minor glitches

- 1% to 10% or even 20% loss may be tolerable

Antithesis of data, which is loss intolerant but delay tolerant

Streaming Stored Multimedia

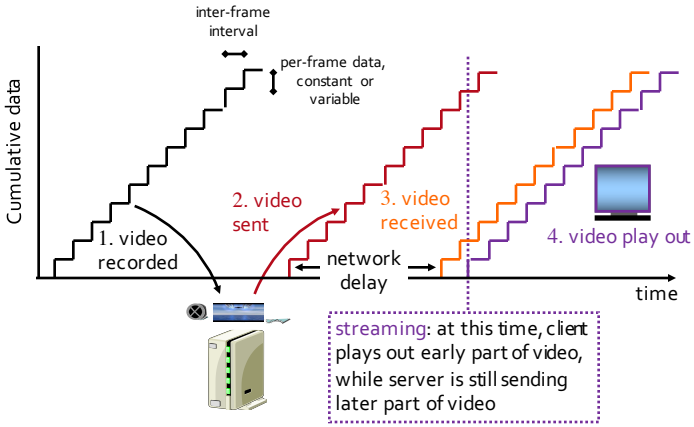


Most common multimedia service model, e.g., YouTube, Hulu, Video on Demand (VoD)

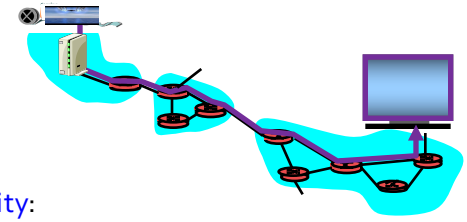
Streaming:

- client playout begins before all data has arrived
- subsequent data must arrive in time for playout

Streaming Stored Multimedia



Streaming Stored Multimedia: Interactivity



Player functionality:
client can pause, rewind, FF, push slider bar

User tolerance to interactive delay:
- 10 sec initial play out delay OK
- 1-2 sec until command effect OK

Streaming Live Multimedia

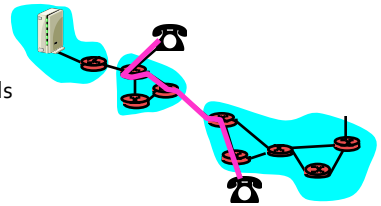
Uses:
- live sporting event
- breaking news, financial news

Streaming:
- by some regulations, playback cannot lag more than 30 seconds from live transmission

Interactivity:
- fast forward impossible
- rewind, pause technically possible (may not be licensed)

Interactive, Real-Time Multimedia

Sample applications:
- IP telephony
- video conferencing
- distributed interactive worlds (multiplayer gaming)



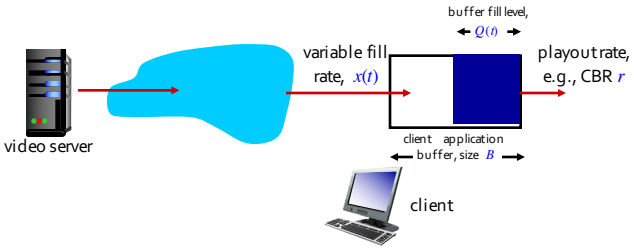
Session initialization
- how does callee advertise its IP address, port number, encoding algorithms?

Streaming Multimedia Techniques

Application-level streaming techniques to make the best out of best-effort service:
- receiver-side buffering
- adaptive playback
- FEC
- use of UDP versus TCP
- multiple encodings of multimedia

Media Player
- jitter removal
- decompression
- adaptive playback
- error concealment
- graphical user interface with controls for interactivity

Receiver-side Buffering



Why not buffer/download the whole file before playing it back?

- takes too long
- client may not have enough buffer space
- owner doesn't allow storage

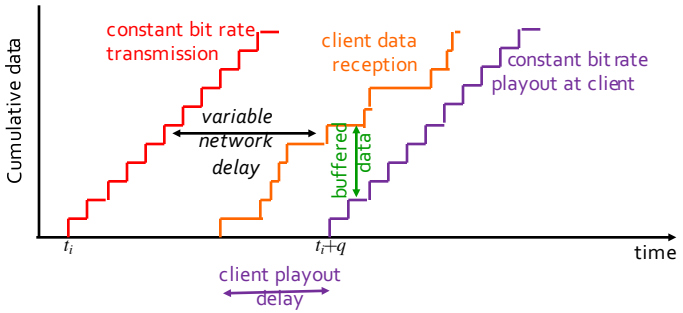
Playback Application

Multimedia applications on the Internet are playback application, i.e. samples generated at time t_i are played back at time t_i+q

Continuous playout challenge: once client playout begins, playback must match original timing

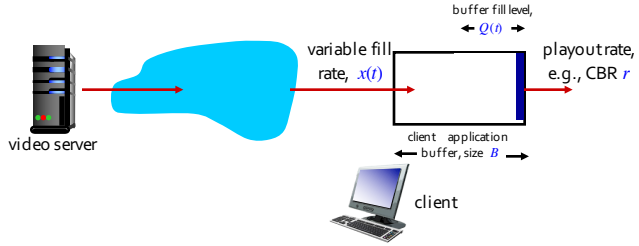
- but network delays are variable (jitter), so will need client-side buffer to compensate for delay introduced by the network and delay jitter (variance in delay)

Delay Jitter



Jitter removal: receiver-side buffering and delayed playback

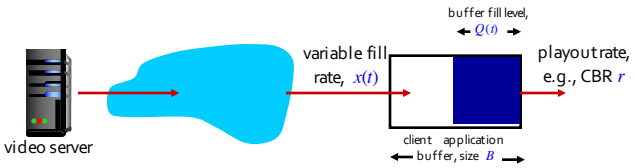
Receiver-side Buffering



Playback point (p_i): transmitted time (t_i) + e2e delay (τ_i) + buffer time (b_i)

1. fill out buffer initially until playout begins at time p_i
2. playout begins at time p_i
3. buffer fill level varies over time as fill rate $x(t)$ varies and playout rate r is constant

Receiver-side Buffering



Playout buffering: average fill rate (x), playout rate (r):

- $x < r$: buffer eventually empties (causing freezing of video playout until buffer fills again)
- $x > r$: buffer will not empty, provided initial playout delay (q) is large enough to absorb variability in $x(t)$

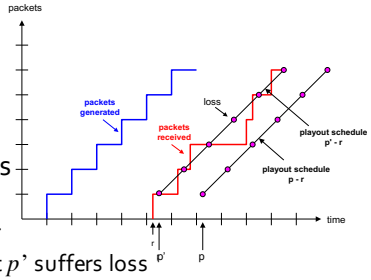
Fixed Playout Delay

Receiver plays out each sample exactly q msecs after it was generated

- sample i has time stamp t: play out sample at t_i+q
- if sample i arrives after t_i+q: too late for play out, data "lost," playback stalled (Rebuffering . . . .)

Trade-off for q:

- large q: less packet loss
- small q: better interactive experience



Example: sender generates packets every 20 msec:

- first packet received at time r
- playout scheduled to begin at p' suffers loss
- playout scheduled to begin at p suffers no loss

Adaptive Playout Delay

Goal: minimize playout delay with minimal missed-playout rate

Observations (for audio):

- human speech can be deconstructed into talk spurts with intervening silent periods

Adaptive playout delay adjustment:

- estimate network delay, adjust playout delay only at the beginning of each talk spurt or buffer playback
- samples played out without jitter during playback
- only silent periods or rebuffering times are compressed and/or elongated

Adaptive Playout Delay

Let:

t_i : timestamp of packet i

r_i : the time packet i is received by receiver

p_i : the time packet i is played back at receiver

$\tau_i = r_i - t_i$: network delay for packet i

d_i : estimate of average network delay after receiving packet i

Dynamic estimate of average delay at receiver:

$$d_i = (1-u)d_{i-1} + u\tau_i$$

where u is a fixed constant (e.g., $u = .01$)

Adaptive Playout Delay

Can also estimate the average deviation of the delay, v_i:

$$v_i = (1-u)v_{i-1} + u\,|\,\tau_i - d_i\,|$$

The estimates d_i and v_i are calculated for every received packet, although they are only used to compute the playback time of the first packet in the playout buffer:

$$p_i = t_i + d_i + kv_i$$

where k is a positive constant, e.g., 4

Remaining packets in playout buffer are played out periodically, e.g., if the sample generation rate is Δt and p_0 is the first packet playout of the current buffer:

$$p_j = p_0 + j * \Delta t$$

Example: Internet Phone

Speaker's audio: alternating talk spurts and silent periods
- 64 kbps during talk spurt

Packets generated only during talk spurts
- 20 msec worth of samples at 8 Kbytes/sec: 160 bytes data

Application-layer header added to each packet

Samples and header encapsulated into UDP segment

Application sends a UDP segment once every 20 msec during talkspurt

Start of Talkspurt

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 $\Rightarrow$ talk spurt begins

With loss possible, receiver must look at both time stamps and sequence numbers
- difference of successive stamps > 20 msec and there's no gap in sequence numbers $\Rightarrow$ talk spurt begins

Dealing with Loss

Two types of loss:
1. network loss: network dropped packet
2. playback loss: packet arrives too late for playout at receiver
   - latency: delays due to processing, queueing in network; end-system (sender, receiver) delays
   - typical maximum tolerable one-way latency: 50-400 ms

Loss tolerance:
- depending on data encoding and application, losses may be concealed from user, e.g., by simply replaying the last sample
- packet loss rates between 1% and 10% may be tolerated

Loss Recovery

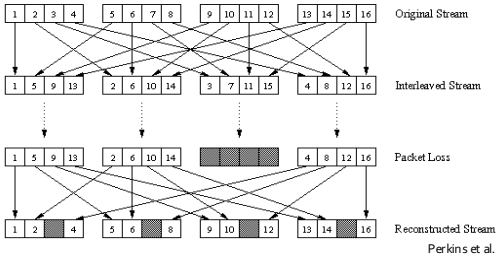
Due to the delay intolerance of multimedia application, ARQ may not be a feasible option

Requesting retransmission from the source is also not feasible for multicast and broadcast scenario

Error correcting code or forward error correction (FEC) allows for the detection and correction of errors by sending additional information

Do "Nothing": Interleaving

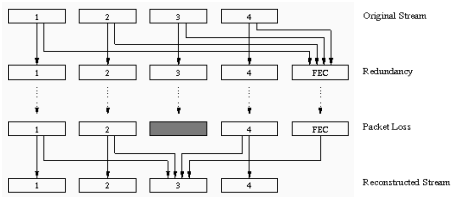
Idea: disperse the effects of packet loss
- samples are broken up into smaller units
- for example, 40 ms of sound is about 1 phoneme
- packet contains small units from different samples
- if packet is lost, still have most of every samples
- has no redundancy overhead, but adds to playout delay



Perkins et al.

FEC: Simple XOR Parity Packet

XOR operation across n packets
Transmit 1 parity packet for every n data packets
If 1 in n packet is lost, can fully recover

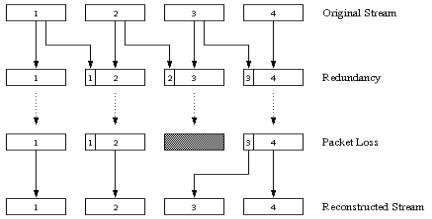


Playout delay must accommodate receipt of all $n+1$ packets
Tradeoff: larger n
- less bandwidth "wastage," but
- longer playout delay and
- higher probability that 2 or more packets can be lost

Perkins et al.

FEC: Piggy-back Lower-quality Stream

Send lower resolution stream as the redundant information
- for example, nominal stream PCM at 64 kbps and redundant stream GSM at 13 kbps



- whenever there is non-consecutive loss, the receiver can conceal the loss
- can also append both (k-1) and (k-2) low-bit rate samples

Perkins et al.

UDP or TCP?

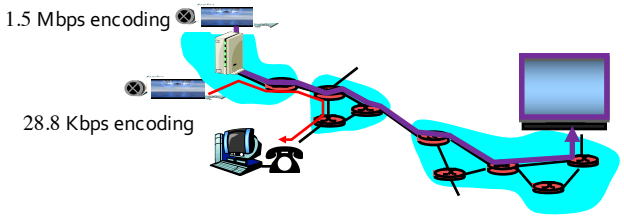
UDP
- server sends at rate appropriate for client (oblivious to network congestion!)
  - often send rate = encoding rate = constant rate
  - then, buffer fill rate = constant rate -packet loss
- short playout delay (2-5 seconds) to remove delay jitter
- retransmission of lost packet: time permitting
- NAT and firewall traversal problem

TCP
- send at maximum possible rate under TCP
- buffer fill rate fluctuates due to TCP congestion control
- use larger playout delay to smooth TCP delivery rate
- HTTP/TCP passes more easily through firewalls

Multiple Encodings

How to handle different client receive rate capabilities, e.g., 236 Kbps EDGE, 100 Mbps LAN?

Server stores and transmits multiple copies of video, encoded at different rates (or use scalable H.264!)



Multiple Encoding: DASH

DASH: Dynamic, Adaptive Streaming over HTTP

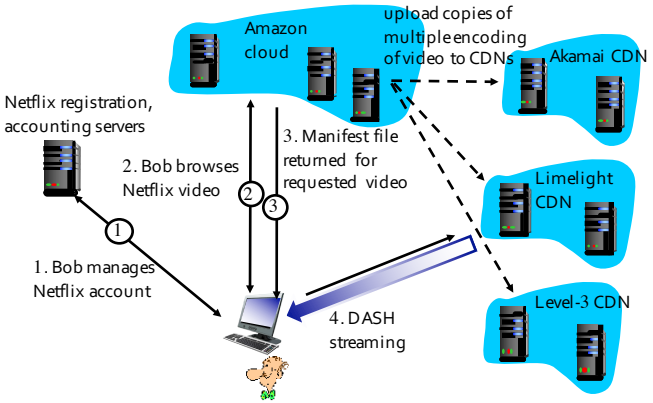
server:
- divides video file into multiple chunks
- each chunk stored, encoded at multiple rates
- manifest file: provides URLs for different chunks

client:
- periodically measures server-to-client bandwidth
- consults manifest, requests one chunk at a time
  - chooses maximum encoding rate sustainable given current bandwidth
  - can choose different encoding rates at different points in time (depending on available bandwidth at the time)

Multiple Encoding: 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 is available)
- where to request chunk (can request from URL server that is "close" to client or has high available bandwidth)

Case Study: Netflix



Internet Multimedia Bag of Tricks

Use UDP to avoid TCP congestion control (delays) for time-sensitive traffic

Receiver-side adaptive playout delay: to compensate for jitter

Sender-side matches streaming bandwidth to available client-to-server path bandwidth
- choose among pre-encoded stream rates
- dynamic server encoding rate
- scalable/layered coding

Error recovery (on top of UDP)
- retransmissions, time permitting
- conceal errors: repeat or interpolate nearby data or interleaving
- error correction: FEC