Multimedia and Quality of Service: What is it?

multimedia applications: network audio and video ("continuous media")

QoS

network provides application with level of performance needed for application to function.
MM Networking Applications

Fundamental characteristics:
- typically delay sensitive
  - end-to-end delay
  - delay jitter
- loss tolerant: infrequent losses cause minor glitches
- antithesis of data, which are loss intolerant but delay tolerant.

Classes of MM applications:
1) stored streaming
2) live streaming
3) interactive, real-time

Jitter is the variability of packet delays within the same packet stream.
Streaming Stored Multimedia

Stored streaming:
- media stored at source
- transmitted to client
- *streaming*: client playout begins before all data has arrived
- timing constraint for still-to-be transmitted data: in time for playout
Streaming Stored Multimedia: What is it?

1. video recorded
2. video sent
3. video received, played out at client

Cumulative data

network delay

streaming: at this time, client playing out early part of video, while server still sending later part of video
Streaming *Stored* Multimedia: Interactivity

- **VCR-like functionality**: client can pause, rewind, FF, push slider bar
  - 10 sec initial delay OK
  - 1-2 sec until command effect OK
  - timing constraint for still-to-be transmitted data: in time for playout
Streaming **Live** Multimedia

**Examples:**
- Internet radio talk show
- Live sporting event

**Streaming** (as with streaming *stored* multimedia)
- Playback buffer
- Playback can lag tens of seconds after transmission
- Still have timing constraint

**Interactivity**
- Fast forward impossible
- Rewind, pause possible!
Real-Time Interactive Multimedia

• **applications:** IP telephony, video conference, distributed interactive worlds

• **end-end delay requirements:**
  – audio: < 150 msec good, < 400 msec OK
    • includes application-level (packetization) and network delays
    • higher delays noticeable, impair interactivity

• **session initialization**
  – how does callee advertise its IP address, port number, encoding algorithms?
Multimedia Over Today’s Internet

TCP/UDP/IP: “best-effort service”
• no guarantees on delay, loss

Today’s Internet multimedia applications use application-level techniques to mitigate (as best possible) effects of delay, loss
How should the Internet evolve to better support multimedia?

**Integrated services philosophy:**
- fundamental changes in Internet so that apps can reserve end-to-end bandwidth
- requires new, complex software in hosts & routers

**Laissez-faire**
- no major changes
- more bandwidth when needed
- content distribution, application-layer multicast
  - application layer

**Differentiated services philosophy:**
- fewer changes to Internet infrastructure, yet provide 1st and 2nd class service

What’s your opinion?
A few words about audio compression

• analog 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
  – 8 bits for 256 values

• 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
A few words about video compression

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

**Examples:**
- MPEG 1 (CD-ROM) 1.5 Mbps
- MPEG2 (DVD) 3-6 Mbps
- MPEG4 (often used in Internet, < 1 Mbps)

**Research:**
- layered (scalable) video
  - adapt layers to available bandwidth
Streaming Stored Multimedia

application-level streaming techniques for making the best out of best effort service:

- client-side buffering
- use of UDP versus TCP
- multiple encodings of multimedia

Media Player

- jitter removal
- decompression
- error concealment
- graphical user interface w/ controls for interactivity
Internet multimedia: simplest approach

- audio or video stored in file
- files transferred as HTTP object
  - received in entirety at client
  - then passed to player

audio, video not streamed:
- no, “pipelining,” long delays until playout!
Internet multimedia: streaming approach

- browser GETs **metafile**
- browser launches player, passing metafile
- player contacts server
- server **streams** audio/video to player
Streaming from a streaming server

- allows for non-HTTP protocol between server, media player
- UDP or TCP for step (3)
Streaming Multimedia: Client Buffering

- client-side buffering, playout delay compensate for network-added delay, delay jitter
Streaming Multimedia: Client Buffering

- client-side buffering, playout delay compensate for network-added delay, delay jitter
Streaming Multimedia: UDP or TCP?

- **UDP**
  - server sends at rate appropriate for client (oblivious to network congestion !)
    - often send rate = encoding rate = constant rate
    - then, fill rate = constant rate - packet loss
  - short playout delay (2-5 seconds) to remove network jitter
  - error recovery: time permitting

- **TCP**
  - send at maximum possible rate under TCP
  - fill rate fluctuates due to TCP congestion control
  - larger playout delay: smooth TCP delivery rate
  - HTTP/TCP passes more easily through firewalls
Q: how to handle different client receive rate capabilities?
- 28.8 Kbps dialup
- 100 Mbps Ethernet

A: server stores, transmits multiple copies of video, encoded at different rates
Real-time interactive applications

• PC-2-PC phone
  – Skype
• PC-2-phone
  – Dialpad
  – Net2phone
  – Skype
• videoconference with webcams
  – Skype
  – Polycom
Interactive Multimedia: Internet Phone

Introduce Internet Phone by way of an example

• speaker’s audio: alternating talk spurts, silent periods.
  – 64 kbps during talk spurt
  – pkts generated only during talk spurts
  – 20 msec chunks at 8 Kbytes/sec: 160 bytes data

• application-layer header added to each chunk.

• chunk+header encapsulated into UDP segment.

• application sends UDP segment into socket every 20 msec during talkspurt
Internet Phone: Packet Loss and 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, losses concealed, packet loss rates between 1% and 10% can be tolerated.
• consider end-to-end delays of two consecutive packets: difference can be more or less than 20 msec (transmission time difference)
Internet Phone: 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
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 $p'$

![Diagram showing packet generation and playout schedules]
Adaptive Playout Delay (1)

- **Goal:** minimize playout delay, keeping late loss rate low
- **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.

\[
t_i = \text{timestamp of the } i\text{th packet}
\]

\[
r_i = \text{the time packet } i \text{ is received by receiver}
\]

\[
p_i = \text{the time packet } i \text{ is played at receiver}
\]

\[
r_i - t_i = \text{network delay for } i\text{th packet}
\]

\[
d_i = \text{estimate of average network delay after receiving } i\text{th packet}
\]

dynamic estimate of average delay at receiver:

\[
d_i = (1 - u)d_{i-1} + u(r_i - t_i)
\]

where \( u \) is a fixed constant (e.g., \( u = .01 \)).
Adaptive playout delay (2)

• also useful to estimate average deviation of delay, \( v_i \):
  \[
v_i = (1-u)v_{i-1} + u |r_i - t_i - d_i|
  \]

• estimates \( d_i, v_i \) calculated for every received packet
• (but used only at start of talk spurt)

• for first packet in talk spurt, playout time is:
  \[
p_i = t_i + d_i + K v_i
  \]
  • where K is positive constant

• remaining packets in talkspurt are played out periodically
Adaptive Playout (3)

**Q:** 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 → talk spurt begins.

- with loss possible, receiver must look at both time stamps and sequence numbers.
  - difference of successive stamps > 20 msec and sequence numbers without gaps → talk spurt begins.
Recovery from packet loss (1)

**Forward Error Correction (FEC): simple scheme**

- for every group of $n$ chunks create redundant chunk by exclusive OR-ing $n$ original chunks
- send out $n+1$ chunks, increasing bandwidth by factor $1/n$.
- can reconstruct original $n$ chunks if at most one lost chunk from $n+1$ chunks

- playout delay: enough time to receive all $n+1$ packets
- tradeoff:
  - increase $n$, less bandwidth waste
  - increase $n$, longer playout delay
  - increase $n$, higher probability that 2 or more chunks will be lost
Recovery from packet loss (2)

2nd 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.

- whenever there is non-consecutive loss, receiver can conceal the loss.
- can also append (n-1)st and (n-2)nd low-bit rate chunk
Recovery from packet loss (3)

Interleaving
- chunks divided into smaller units
- for example, four 5 msec units per chunk
- packet contains small units from different chunks

- if packet lost, still have most of every chunk
- no redundancy overhead, but increases playout delay
Summary: Internet Multimedia: bag of tricks

- use UDP to avoid TCP congestion control (delays) for time-sensitive traffic
- client-side adaptive playout delay: to compensate for delay
- server side matches stream bandwidth to available client-to-server path bandwidth
  - chose among pre-encoded stream rates
  - dynamic server encoding rate
- error recovery (on top of UDP)
  - FEC, interleaving, error concealment
  - retransmissions, time permitting
Providing Multiple Classes of Service

• thus far: making the best of best effort service
  – one-size fits all service model

• alternative: multiple classes of service
  – partition traffic into classes
  – network treats different classes of traffic differently (analogy: VIP vs. regular service)

• granularity: differential service among multiple classes, not among individual connections

• history: ToS bits
Multiple classes of service: scenario
Scenario 1: mixed FTP and audio

- Example: 1Mbps IP phone, FTP share 1.5 Mbps link.
  - bursts of FTP can congest router, cause audio loss
  - want to give priority to audio over FTP

Principle 1
packet marking needed for router to distinguish between different classes; and new router policy to treat packets accordingly
Principles for QOS Guarantees (I)

• what if applications misbehave (audio sends higher than declared rate)
  – policing: force source adherence to bandwidth allocations
• marking and policing at network edge:

Principle 2

provide protection (isolation) for one class from others
Principles for QOS Guarantees (II)

• Allocating fixed (non-sharable) bandwidth to flow: inefficient use of bandwidth if flows doesn’t use its allocation

Principle 3
While providing isolation, it is desirable to use resources as efficiently as possible
Scheduling And Policing Mechanisms

- **scheduling**: choose next packet to send on link
- **FIFO (first in first out) scheduling**: send in order of arrival to queue
  - real-world example?
  - **discard policy**: if packet arrives to full queue: who to discard?
    - Tail drop: drop arriving packet
    - priority: drop/remove on priority basis
    - random: drop/remove randomly
Scheduling Policies: more

Priority scheduling: transmit highest priority queued packet

• multiple classes, with different priorities
  – class may depend on marking or other header info, e.g. IP source/dest, port numbers, etc.
Scheduling Policies: still more

round robin scheduling:

• multiple classes
• cyclically scan class queues, serving one from each class (if available)
Scheduling Policies: still more

Weighted Fair Queuing:
- generalized Round Robin
- each class gets weighted amount of service in each cycle
Policing Mechanisms

Goal: limit traffic to not exceed declared parameters

Three common-used criteria:

• (Long term) Average Rate: how many pkts can be sent per unit time (in the long run)
• Peak Rate: e.g., 6000 pkts per min. (ppm) avg.; 1500 ppm peak rate
• (Max.) Burst Size: max. number of pkts sent consecutively (with no intervening idle)
Policing Mechanisms

**Token Bucket:** limit input to specified Burst Size and Average Rate.

- bucket can hold \( b \) tokens
- tokens generated at rate \( r \) token/sec unless bucket full
- over interval of length \( t \): number of packets admitted less than or equal to \( r t + b \).
Policing Mechanisms (more)

- token bucket, WFQ combine to provide guaranteed upper bound on delay, i.e., QoS guarantee!

\[ D_{\text{max}} = \frac{b}{R} \]

Arriving traffic

Token rate, \( r \)

Bucket size, \( b \)

Per-flow rate, \( R \)

WFQ

\[ D_{\text{max}} = \frac{b}{R} \]
Principles for QOS Guarantees (III)

- *Basic fact of life*: can not support traffic demands beyond link capacity

Principle 4

Call Admission: flow declares its needs, network may block call (e.g., busy signal) if it cannot meet needs
QoS guarantee scenario

- Resource reservation
  - call setup, signaling (RSVP)
  - traffic, QoS declaration
  - per-element admission control

- QoS-sensitive scheduling (e.g., WFQ)
IETF Integrated Services

• architecture for providing QOS guarantees in IP networks for individual application sessions
• resource reservation: routers maintain state info (a la VC) of allocated resources, QoS req’s
• admit/deny new call setup requests:

**Question:** can newly arriving flow be admitted with performance guarantees while not violated QoS guarantees made to already admitted flows?
Call Admission

Arriving session must:

• declare its QoS requirement
  – *R-spec*: defines the QoS being requested

• characterize traffic it will send into network
  – *T-spec*: defines traffic characteristics

• signaling protocol: needed to carry R-spec and T-spec to routers (where reservation is required)
  – *RSVP*
Intserv QoS: Service models [rfc2211, rfc 2212]

Guaranteed service:
• worst case traffic arrival: leaky-bucket-policed source
• simple (mathematically provable) bound on delay [Parekh 1992, Cruz 1988]

Controlled load service:
• "a quality of service closely approximating the QoS that same flow would receive from an unloaded network element."

\[
D_{\text{max}} = \frac{b}{R}
\]
Signaling in the Internet

- **New requirement**: reserve resources along end-to-end path (end system, routers) for QoS for multimedia applications
- **RSVP**: Resource Reservation Protocol [RFC 2205]
  - “... allow users to communicate requirements to network in robust and efficient way.” i.e., signaling!
- earlier Internet Signaling protocol: ST-II [RFC 1819]
RSVP Design Goals

1. accommodate heterogeneous receivers (different bandwidth along paths)
2. accommodate different applications with different resource requirements
3. make multicast a first class service, with adaptation to multicast group membership
4. leverage existing multicast/unicast routing, with adaptation to changes in underlying unicast, multicast routes
5. control protocol overhead to grow (at worst) linear in # receivers
6. modular design for heterogeneous underlying technologies
RSVP: does not...

- specify how resources are to be reserved
  - rather: a mechanism for communicating needs
- determine routes packets will take
  - that’s the job of routing protocols
  - signaling decoupled from routing
- interact with forwarding of packets
  - separation of control (signaling) and data (forwarding) planes
RSVP: overview of operation

- senders, receiver join a multicast group
  - done outside of RSVP
  - senders need not join group
- sender-to-network signaling
  - *path message*: make sender presence known to routers
  - path teardown: delete sender’s path state from routers
- receiver-to-network signaling
  - *reservation message*: reserve resources from sender(s) to receiver
  - reservation teardown: remove receiver reservations
- network-to-end-system signaling
  - path error
  - reservation error
IETF Differentiated Services

• want “qualitative” service classes
  – “behaves like a wire”
  – relative service distinction: Platinum, Gold, Silver

• *scalability*: simple functions in network core, relatively complex functions at edge routers (or hosts)
  – signaling, maintaining per-flow router state difficult with large number of flows

• don’t define service classes, provide functional components to build service classes
Diffserv Architecture

**Edge router:**
- per-flow traffic management
- marks packets as in-profile and out-profile

**Core router:**
- per class traffic management
- buffering and scheduling based on marking at edge
- preference given to in-profile packets
Edge-Router Packet Marking

- **profile:** pre-negotiated rate A, bucket size B
- **packet marking at edge based on per-flow profile**

Possible usage of marking:

- **class-based marking:** packets of different classes marked differently
- **intra-class marking:** conforming portion of flow marked differently than non-conforming one
Classification and Conditioning

- Packet is marked in the Type of Service (TOS) in IPv4, and Traffic Class in IPv6
- 6 bits used for Differentiated Service Code Point (DSCP) and determine PHB that the packet will receive
- 2 bits are currently unused
Classification and Conditioning

may be desirable to limit traffic injection rate of some class:

• user declares traffic profile (e.g., rate, burst size)
• traffic metered, shaped if non-conforming
Forwarding (PHB)

• PHB result in a different observable (measurable) forwarding performance behavior
• PHB does not specify what mechanisms to use to ensure required PHB performance behavior
• Examples:
  – Class A gets x% of outgoing link bandwidth over time intervals of a specified length
  – Class A packets leave first before packets from class B
Forwarding (PHB)

PHBs being developed:

- **Expedited Forwarding:** pkt departure rate of a class equals or exceeds specified rate
  - logical link with a minimum guaranteed rate

- **Assured Forwarding:** 4 classes of traffic
  - each guaranteed minimum amount of bandwidth
  - each with three drop preference partitions