Chapter 7
Multimedia Networking

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 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) in substantially unaltered form, that you mention their source (after all, we’d like people to use our book!)
- If you post any slides in substantially unaltered form 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

All material copyright 1996-2007
J.F Kurose and K.W. Ross, All Rights Reserved
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.
Chapter 7: goals

Principles
- classify multimedia applications
- identify network services applications need
- making the best of best effort service

Protocols and Architectures
- specific protocols for best-effort
- mechanisms for providing QoS
- architectures for QoS
Chapter 7 outline

7.1 multimedia networking applications
7.2 streaming stored audio and video
7.3 making the best out of best effort service
7.4 protocols for real-time interactive applications RTP, RTCP, SIP

7.5 providing multiple classes of service
7.6 providing QoS guarantees
MM Networking Applications

**Classes of MM applications:**

1) stored streaming
2) live streaming
3) interactive, real-time

**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.

**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.

But you said multimedia apps require QoS and level of performance to be effective!
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?
The path of audio
Sampling

- Take measurements at regular intervals
- Number of samples in a second is the *sampling rate*
  - The faster the rate, the more accurate the recording, up to Nyquist sampling rate

*Figure 11.4. Two sampling rates; the rate on the right is twice as fast as that on the left.*
Figure 11.6. (a) Three-bit precision for samples requires that the indicated reading is approximated as +10. (b) Adding another bit makes the sample twice as accurate.
Compression

- The principle of compression
- Two categories of compression:
  - Lossless compression
  - Lousy compression
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
Chapter 7 outline

7.1 multimedia networking applications
7.2 streaming stored audio and video
7.3 making the best out of best effort service
7.4 protocols for real-time interactive applications RTP, RTCP, SIP
7.5 providing multiple classes of service
7.6 providing QoS guarantees
Streaming Stored Multimedia

application-level streaming techniques for making the best out of best effort service:
- client-side buffering
- use of UDP and 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), more shortly
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 recover: 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
Streaming Multimedia: client rate(s)

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
User Control of Streaming Media: RTSP (Real-Time Streaming Protocol)

**HTTP**
- does not target multimedia content
- no commands for fast forward, etc.

**RTSP: RFC 2326**
- client-server application layer protocol
- user control: rewind, fast forward, pause, resume, repositioning, etc...

**What it doesn't do:**
- doesn't define how audio/video is encapsulated for streaming over network
- doesn't restrict how streamed media is transported (UDP or TCP possible)
- doesn't specify how media player buffers audio/video

Q: In band or out of band?
RTSP: out of band control

FTP uses an “out-of-band” control channel:
- file transferred over one TCP connection.
- control info (directory changes, file deletion, rename) sent over separate TCP connection
- “out-of-band”, “in-band” channels use different port numbers

RTSP messages also sent out-of-band:
- RTSP control messages use different port numbers than media stream: out-of-band.
  - port 554
- media stream is considered “in-band”.
RTSP Example

Scenario:
- metafile communicated to web browser
- browser launches player
- player sets up an RTSP control connection, data connection to streaming server
Example

<title>Twister</title>
<session>
    <group language=en lipsync>
        <switch>
            <track type=audio
                e="PCMU/8000/1"
                src = "rtsp://audio.example.com/twister/audio.en/lofi">
            <track type=audio
                e="DVI4/16000/2" pt="90 DVI4/8000/1"
                src="rtsp://audio.example.com/twister/audio.en/hifi">
        </switch>
        <track type="video/jpeg"
            src="rtsp://video.example.com/twister/video">
    </group>
</session>
RTSP Operation

1. The web browser requests a presentation description from the server using an HTTP GET request.
2. The server sends the presentation description back to the browser.
3. The browser requests a media stream by sending a SETUP request to the server.
4. The server responds with a media stream.
5. The browser can start playing the media stream by sending a PLAY request.
6. The browser can pause the media stream by sending a PAUSE request.
7. The browser can tear down the media stream by sending a TEARDOWN request.

Diagram:
- Web browser
- Web server
- Media player
- Media server
- Client
- Server
RTSP Exchange Example

C: SETUP rtsp://audio.example.com/twister/audio RTSP/1.0
   Cseq: 1
   Transport: rtp/udp; compression; port=3056; mode=PLAY

S: RTSP/1.0 200 1 OK
   Cseq: 1
   Session 4231

C: PLAY rtsp://audio.example.com/twister/audio.en/lofi RTSP/1.0
   Cseq: 2
   Session: 4231
   Range: npt=0-

C: PAUSE rtsp://audio.example.com/twister/audio.en/lofi RTSP/1.0
   Cseq: 3
   Session: 4231
   Range: npt=37

C: TEARDOWN rtsp://audio.example.com/twister/audio.en/lofi RTSP/1.0
   Cseq: 4
   Session: 4231

S: 200 3 OK
Chapter 7 outline

7.1 multimedia networking applications
7.2 streaming stored audio and video
7.3 making the best out of best effort service
7.4 protocols for real-time interactive applications RTP, RTCP, SIP
7.5 providing multiple classes of service
7.6 providing QoS guarantees
Real-time interactive applications

- PC-2-PC phone
  - Skype
- PC-2-phone
  - Dialpad
  - Net2phone
  - Skype
- videoconference with webcams
  - Skype
  - Polycom

Going to now look at a PC-2-PC Internet phone example in detail

7: Multimedia Networking    7-35
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.
Delay Jitter

- 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 \( p' \)
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.

\[
\begin{align*}
  t_i &= \text{timestamp of the } i\text{th packet} \\
  r_i &= \text{the time packet } i \text{ is received by receiver} \\
  p_i &= \text{the time packet } i \text{ is played at receiver} \\
  r_i - t_i &= \text{network delay for } i\text{th packet} \\
  d_i &= \text{estimate of average network delay after receiving } i\text{th packet}
\end{align*}
\]

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:
  $$p_i = t_i + d_i + Kv_i$$
  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
- Suitable for stored media
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

- **CDN**: bring content closer to clients
Chapter 7 outline

7.1 multimedia networking applications
7.2 streaming stored audio and video
7.3 making the best out of best effort service
7.4 protocols for real-time interactive applications RTP, RTCP, SIP
7.5 providing multiple classes of service
7.6 providing QoS guarantees
Real-Time Transport Protocol (RTP)

- RTP specifies packet structure for packets carrying audio, video data
- RFC 3550
- RTP packet provides
  - payload type identification
  - packet sequence numbering
  - time stamping
- RTP runs in end systems
- RTP packets encapsulated in UDP segments
- interoperability: if two Internet phone applications run RTP, then they may be able to work together
RTP runs on top of UDP

RTP libraries provide transport-layer interface that extends UDP:
- port numbers, IP addresses
- payload type identification
- packet sequence numbering
- time-stamping

7: Multimedia Networking
**RTP Example**

- Consider sending 64 kbps PCM-encoded voice over RTP.
- Application collects encoded data in chunks, e.g., every 20 msec = 160 bytes in a chunk.
- Audio chunk + RTP header form RTP packet, which is encapsulated in UDP segment.

- RTP header indicates type of audio encoding in each packet
  - Sender can change encoding during conference.
- RTP header also contains sequence numbers, timestamps.

7: Multimedia Networking  7-51
RTP and QoS

- RTP does not provide any mechanism to ensure timely data delivery or other QoS guarantees.
- RTP encapsulation is only seen at end systems (not) by intermediate routers.
  - Routers providing best-effort service, making no special effort to ensure that RTP packets arrive at destination in timely matter.
## RTP Header

### Payload Type (7 bits)
Indicates type of encoding currently being used. If sender changes encoding in middle of conference, sender informs receiver via payload type field.

- Payload type 0: PCM mu-law, 64 kbps
- Payload type 3, GSM, 13 kbps
- Payload type 7, LPC, 2.4 kbps
- Payload type 26, Motion JPEG
- Payload type 31, H.261
- Payload type 33, MPEG2 video

### Sequence Number (16 bits)
Increments by one for each RTP packet sent, and may be used to detect packet loss and to restore packet sequence.
RTP Header (2)

- **Timestamp field (32 bytes long):** sampling instant of first byte in this RTP data packet
  - for audio, timestamp clock typically increments by one for each sampling period (for example, each 125 usecs for 8 KHz sampling clock)
  - if application generates chunks of 160 encoded samples (20ms), then timestamp increases by 160 for each RTP packet when source is active. Timestamp clock continues to increase at constant rate when source is inactive.

- **SSRC field (32 bits long):** identifies source of the RTP stream. Each stream in RTP session should have distinct SSRC (Synchronization source identifier).
Real-Time Control Protocol (RTCP)

- works in conjunction with RTP.
- each participant in RTP session periodically transmits RTCP control packets to all other participants.
- each RTCP packet contains sender and/or receiver reports
  - report statistics useful to application: # packets sent, # packets lost, interarrival jitter, etc.
- feedback can be used to control performance
  - sender may modify its transmissions based on feedback
RTCP - Continued

- each RTP session: typically a single multicast address; all RTP /RTCP packets belonging to session use multicast address.

- RTP, RTCP packets distinguished from each other via distinct port numbers.

- to limit traffic, each participant reduces RTCP traffic as number of conference participants increases
RTCP Packets

**Receiver report packets:**
- fraction of packets lost, last sequence number, average interarrival jitter

**Sender report packets:**
- SSRC of RTP stream, current time, number of packets sent, number of bytes sent

**Source description packets:**
- e-mail address of sender, sender's name, SSRC of associated RTP stream
- provide mapping between the SSRC and the user/host name
Synchronization of Streams

- RTCP can synchronize different media streams within a RTP session.
- Consider videoconferencing app for which each sender generates one RTP stream for video, one for audio.
- Timestamps in RTP packets tied to the video, audio sampling clocks:
  - *not* tied to wall-clock time.
- Each RTCP sender-report packet contains (for most recently generated packet in associated RTP stream):
  - Timestamp of RTP packet
  - Wall-clock time for when packet was created.
- Receivers use association to synchronize playout of audio, video.
RTCP Bandwidth Scaling

- RTCP attempts to limit its traffic to 5% of session bandwidth.

**Example**

- Suppose one sender, sending video at 2 Mbps. Then RTCP attempts to limit its traffic to 100 Kbps.
- RTCP gives 75% of rate to receivers; remaining 25% to sender
- 75 kbps is equally shared among receivers:
  - with R receivers, each receiver gets to send RTCP traffic at 75/R kbps.
- sender gets to send RTCP traffic at 25 kbps.
- participant determines RTCP packet transmission period by calculating avg RTCP packet size (across entire session) and dividing by allocated rate
SIP: Session Initiation Protocol [RFC 3261]

SIP long-term vision:

- all telephone calls, video conference calls take place over Internet
- people are identified by names or e-mail addresses, rather than by phone numbers
- you can reach callee, no matter where callee roams, no matter what IP device callee is currently using


**SIP Services**

- Setting up a call, SIP provides mechanisms ..
  - for caller to let callee know she wants to establish a call
  - so caller, callee can agree on media type, encoding
  - to end call

- determine current IP address of callee:
  - maps mnemonic identifier to current IP address

- call management:
  - add new media streams during call
  - change encoding during call
  - invite others
  - transfer, hold calls
Setting up a call to known IP address

- Alice’s SIP invite message indicates her port number, IP address, encoding she prefers to receive (PCM ulaw).

- Bob’s 200 OK message indicates his port number, IP address, preferred encoding (GSM).

- SIP messages can be sent over TCP or UDP; here sent over RTP/UDP.

- Default SIP port number is 5060.
Setting up a call (more)

- codec negotiation:
  - suppose Bob doesn’t have PCM ulaw encoder.
  - Bob will instead reply with 606 Not Acceptable Reply, listing his encoders Alice can then send new INVITE message, advertising different encoder

- rejecting a call
  - Bob can reject with replies “busy,” “gone,” “payment required,” “forbidden”

- media can be sent over RTP or some other protocol
Example of SIP message

INVITE sip:bob@domain.com SIP/2.0
Via: SIP/2.0/UDP 167.180.112.24
From: sip:alice@hereway.com
To: sip:bob@domain.com
Call-ID: a2e3a@pigeon.hereway.com
Content-Type: application/sdp
Content-Length: 885

c=IN IP4 167.180.112.24
m=audio 38060 RTP/AVP 0

Notes:
- HTTP message syntax
- sdp = session description protocol
- Call-ID is unique for every call.

- Here we don’t know Bob’s IP address. Intermediate SIP servers needed.
- Alice sends, receives SIP messages using SIP default port 5060
- Alice specifies in Via: header that SIP client sends, receives SIP messages over UDP
Name translation and user location

- caller wants to call callee, but only has callee’s name or e-mail address.
- need to get IP address of callee’s current host:
  - user moves around
  - DHCP protocol
  - user has different IP devices (PC, PDA, car device)

- result can be based on:
  - time of day (work, home)
  - caller (don’t want boss to call you at home)
  - status of callee (calls sent to voicemail when callee is already talking to someone)

Service provided by SIP servers:
- SIP registrar server
- SIP proxy server
**SIP Registrar**

- When Bob starts SIP client, client sends SIP REGISTER message to Bob’s registrar server (similar function needed by Instant Messaging)

**Register Message:**

```
REGISTER sip:domain.com SIP/2.0
Via: SIP/2.0/UDP 193.64.210.89
From: sip:bob@domain.com
To: sip:bob@domain.com
Expires: 3600
```
**SIP Proxy**

- Alice sends invite message to her proxy server
  - contains address sip:bob@domain.com
- Proxy responsible for routing SIP messages to callee
  - possibly through multiple proxies.
- Callee sends response back through the same set of proxies.
- Proxy returns SIP response message to Alice
  - contains Bob’s IP address
- Proxy analogous to local DNS server
Example

Caller jim@umass.edu with places a call to keith@upenn.edu

(1) Jim sends INVITE message to umass SIP proxy. (2) Proxy forwards request to upenn registrar server. (3) upenn server returns redirect response, indicating that it should try keith@eurecom.fr

(4) umass proxy sends INVITE to eurecom registrar. (5) eurecom registrar forwards INVITE to 197.87.54.21, which is running keith's SIP client. (6-8) SIP response sent back (9) media sent directly between clients.

Note: also a SIP ack message, which is not shown.
Comparison with H.323

- H.323 is another signaling protocol for real-time, interactive applications
- H.323 is a complete, vertically integrated suite of protocols for multimedia conferencing: signaling, registration, admission control, transport, codecs
- SIP is a single component. Works with RTP, but does not mandate it. Can be combined with other protocols, services
- H.323 comes from the ITU (telephony).
- SIP comes from IETF: Borrows much of its concepts from HTTP
  - SIP has Web flavor, whereas H.323 has telephony flavor.
- SIP uses the KISS principle: Keep it simple stupid.
Chapter 7 outline

7.1 multimedia networking applications
7.2 streaming stored audio and video
7.3 making the best out of best effort service
7.4 protocols for real-time interactive applications RTP, RTCP, SIP
7.5 providing multiple classes of service
7.6 providing QoS guarantees
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 service vs regular service)
- granularity: differential service among multiple classes, not among individual connections
- history: ToS bits
Multiple classes of service: scenario

R1 output interface queue

1.5 Mbps link

H1

R1

R2

H3

H2

H4
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 (more)

- what if applications misbehave (audio sends higher than declared rate)
  - policing: force source adherence to bandwidth allocations
- marking and policing at network edge:
  - similar to ATM UNI (User Network Interface)

Principle 2
provide protection (isolation) for one class from others
Principles for QOS Guarantees (more)

- 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
**Principles for QOS Guarantees (more)**

- **Basic fact of life**: cannot 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
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..
  - Real world example?
Scheduling Policies: still more

round robin scheduling:

- multiple classes
- cyclically scan class queues, serving one from each class (if available)
- real world example?
Scheduling Policies: still more

**Weighted Fair Queuing:**
- generalized Round Robin
- each class gets weighted amount of service in each cycle
- real-world example?
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)
  - crucial question: what is the interval length: 100 packets per sec or 6000 packets per min have same average!

- **Peak Rate:** e.g., 6000 pkts per min. (ppm) avg.; 1500 pps 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 $r$ tokens/sec unless bucket full
- over interval of length $t$: number of packets admitted less than or equal to $(r \cdot t + b)$. 
Policing Mechanisms (more)

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

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

WFQ

WFQ

\[ \frac{r_1}{b_1} \]

arriving traffic

\[ \text{token rate, } r \]

\[ \text{bucket size, } b \]

\[ \text{per-flow rate, } R \]
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
Chapter 7 outline

7.1 multimedia networking applications
7.2 streaming stored audio and video
7.3 making the best out of best effort service
7.4 protocols for real-time interactive applications
    RTP, RTCP, SIP
7.5 providing multiple classes of service
7.6 providing QoS guarantees
Chapter 7 outline

- 7.1 Multimedia Networking Applications
- 7.2 Streaming stored audio and video
- 7.3 Real-time Multimedia: Internet Phone study
- 7.4 Protocols for Real-Time Interactive Applications
  - RTP, RTCP, SIP
- 7.5 Distributing Multimedia: content distribution networks
- 7.6 Beyond Best Effort
- 7.7 Scheduling and Policing Mechanisms
- 7.8 Integrated Services and Differentiated Services
- 7.9 RSVP
Principles for QOS Guarantees (more)

- **Basic fact of life:** cannot 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-policing 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."

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

- WFQ
  - token rate, \( r \)
  - bucket size, \( b \)
  - per-flow rate, \( R \)
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]
Chapter 7: Summary

**Principles**
- classify multimedia applications
- identify network services applications need
- making the best of best effort service

**Protocols and Architectures**
- specific protocols for best-effort
- mechanisms for providing QoS
- architectures for QoS
  - multiple classes of service
  - QoS guarantees, admission control