Telematics
Chapter 10: Multimedia Networking

Prof. Dr. Mesut Güneş
Computer Systems and Telematics (CST)
Distributed, embedded Systems
Institute of Computer Science
Freie Universität Berlin
http://cst.mi.fu-berlin.de
Contents

- Design issues
- Multimedia networking applications
- Streaming stored audio and video
- Making the best out of best effort service
- Protocols for real-time interactive applications
- Providing multiple classes of service
- Providing QoS guarantees
Design Issues

- Two kinds of applications
  - Non-realtime applications
    - Email, FTP,
  - Realtime applications
    - Audio, video

- Principles
  - Classify multimedia applications
  - Identify network services applications need
  - Making the best of the best effort service

- Protocols and Architectures
  - Specific protocols for best-effort
  - Mechanisms for providing QoS
  - Architectures for QoS

OSI Reference Model

- Application Layer
- Presentation Layer
- Session Layer
- Transport Layer
- Network Layer
- Data Link Layer
- Physical Layer

Prof. Dr. Mesut Güneş • cst.mi.fu-berlin.de • Telematics • Chapter 10: Multimedia Networking
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.*
Quality of Service in the Internet

- Problem today:
  - IP is packet switched, therefore no guarantees on transmission is given
    - Throughput, transmission delay, ...
  - The Internet transmits data at **best effort**
  - But: many applications need a certain Quality of Service (QoS)

<table>
<thead>
<tr>
<th>Application</th>
<th>Reliability</th>
<th>Delay</th>
<th>Jitter</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>E-Mail</td>
<td>high</td>
<td>low</td>
<td>low</td>
<td>low</td>
</tr>
<tr>
<td>File Transfer</td>
<td>high</td>
<td>low</td>
<td>low</td>
<td>medium</td>
</tr>
<tr>
<td>Web Access</td>
<td>high</td>
<td>medium</td>
<td>low</td>
<td>medium</td>
</tr>
<tr>
<td>Remote Login</td>
<td>high</td>
<td>medium</td>
<td>medium</td>
<td>low</td>
</tr>
<tr>
<td>Audio on Demand</td>
<td>low</td>
<td>low</td>
<td>high</td>
<td>medium</td>
</tr>
<tr>
<td>Video on Demand</td>
<td>low</td>
<td>low</td>
<td>high</td>
<td>high</td>
</tr>
<tr>
<td>IP Telephony</td>
<td>low</td>
<td>high</td>
<td>high</td>
<td>low</td>
</tr>
<tr>
<td>Video Conference</td>
<td>low</td>
<td>high</td>
<td>high</td>
<td>high</td>
</tr>
</tbody>
</table>
QoS Parameters

- Throughput [bit/s]
  - Which minimum/maximum/average data rate is necessary?
- Transmission delay [s]
  - Which maximum delay is tolerable?
- Jitter [s]
  - Which fluctuations in the transmission delay are tolerable?
- Availability [%]
  - With which probability the communication service is available?
Multimedia networking applications
MM Networking Applications

- Classes of MM applications:
  - Stored streaming
  - Live streaming
  - 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

\[ d_i = t'_i - t_i \]
\[ j_i = d_{i+1} - d_i \]

**Jitter** \( j_i \) 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 play out begins before all data has arrived
    - Timing constraint for still-to-be transmitted data: in time for play out
Streaming Stored Multimedia: What is it?

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

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

Prof. Dr. Mesut Güneş • cst.mi.fu-berlin.de • Telematics • Chapter 10: Multimedia Networking
Streaming Stored Multimedia: Interactivity

- **VCR-like functionality:**
  - Client can
    - pause, rewind, FF, push slider bar
  - Delay restrictions
    - 10 sec initial delay OK
    - 1-2 sec until command effect OK

- **Timing constraint for still-to-be transmitted data:** in time for play out
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-to-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

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

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
  - Provide small number of service classes (maybe two)

What’s your opinion?
How should the Internet evolve to better support multimedia?

<table>
<thead>
<tr>
<th>Approach</th>
<th>Unit of allocation</th>
<th>Guarantee</th>
<th>Deployment to date</th>
<th>Complexity</th>
<th>Mechanisms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Making the best of best-effort service</td>
<td>None</td>
<td>None or soft</td>
<td>Everywhere</td>
<td>Minimal</td>
<td>Application-layer support, CDN, over-provisioning</td>
</tr>
<tr>
<td>Differential QoS</td>
<td>Classes of flows</td>
<td>None or soft</td>
<td>Some</td>
<td>Medium</td>
<td>Policing, scheduling</td>
</tr>
<tr>
<td>Guaranteed QoS</td>
<td>Individual flows</td>
<td>Soft or hard, once a flow is admitted</td>
<td>Little</td>
<td>High</td>
<td>Policing, scheduling, call admission and signaling</td>
</tr>
</tbody>
</table>
Multimedia networking applications

Audio and Video 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
  - $2^8 = 256$ possible quantized values
- Each quantized value represented by bits
  - 8 bits for 256 values

- Example:
  - 8,000 samples/sec, 256 quantized values $\Rightarrow 64,000$ bps
  - Receiver converts bits back to analog signal
    - Some quality reduction

- Example rates
  - CD: 1.411 Mbps (stereo)
  - MP3: 96, 128, 160 kbps
  - GSM: 13 kbps
  - Internet telephony
    - G.729: 8 kbps
    - G.723.3: 5.3 kbps and 6.4 kbps
A few words about video compression

- **Video**: sequence of images displayed at constant rate
  - Frame rate 24 images/sec
- **Digital image**: array of pixels
  - Each pixel represented by bits
- **Redundancy**
  - Spatial (within image)
  - Temporal (from one image to next)

- **Example**
  - Single image of 1024 pixels
  - Each pixel encoded into 24 bits
  - 3 Mbyte without compression
  - Compression ratio 10:1 results in 300 Kbyte

- **Examples**:
  - MPEG1 (CD-ROM) 1.5 Mbps
  - MPEG2 (DVD) 3-6 Mbps
  - MPEG4 <1 Mbps
    - Often used in Internet
  - Research: layered (scalable) video
    - Adapt layers to available bandwidth
Streaming stored audio and video
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 with 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 play out!
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 and media player
- UDP or TCP for step (3)
Streaming Multimedia: Client Buffering

- Client-side buffering, play out delay compensate for network-added delay, delay jitter
Streaming Multimedia: Client Buffering

- Client-side buffering, play out 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
    - Fill rate = constant rate - packet loss
  - Short play out 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 play out 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
Streaming stored audio and video

RTSP
User Control of Streaming Media: RTSP

- HTTP
  - Does not target multimedia content
  - No commands for fast forward, etc.

- RTSP: RFC2326
  - Client-server application layer protocol
  - User control
    - rewind
    - fast forward
    - pause, resume
    - repositioning, etc.

- What it does not do:
  - Does not define compression schemes
  - Does not define how audio/video is encapsulated for streaming over network
  - Does not restrict how streamed media is transported (UDP or TCP possible)
  - Does not specify how media player buffers audio/video
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
● Metafile example

```xml
<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

Web browser

HTTP Get
Presentation description file

Web server

Media Player

Setup
Play
Media stream
Pause
Teardown

Streaming server

Prof. Dr. Mesut Güneş • cst.mi.fu-berlin.de • Telematics • Chapter 10: Multimedia Networking
RTSP Exchange Example

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

S: RTSP/1.0 200 1 OK
   Session 4231

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

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

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

S: 200 3 OK
Making the best out of best-effort service
Interactive Multimedia: Internet Phone

- Look at a PC-2-PC Internet phone example in detail
  - Speaker’s audio: alternating talk spurts and silent periods
    - 64 kbps during talk spurt
    - Packets generated only during talk spurts
    - 20 msec chunks at 8000 bytes/sec → 1 chunk = 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 talk spurt
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 play out at receiver
  - Delays:
    - Processing
    - Queuing 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 play out 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 play out, 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

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

● 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., \( u = 0.01 \)).
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
Ada

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 \( \Rightarrow \) 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 \( \Rightarrow \) talk spurt begins.
Making the best out of best effort service
Recovery from packet loss
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
Making the best out of best-effort-service
Content distribution networks (CDN)
Content distribution networks (CDNs)

- Content replication
  - Challenging to stream large files (e.g., Video) from single origin server in real time
  - Solution: replicate content at hundreds of servers throughout internet
    - Content downloaded to CDN servers ahead of time
    - Placing content “close” to user avoids impairments (loss, delay) of sending content over long paths
    - CDN server typically in edge/access network
Content distribution networks (CDNs)

- Content replication
  - CDN (e.g., Akamai) customer is the content provider (e.g., CNN)
  - CDN replicates customers’ content in CDN servers.
  - When provider updates content, CDN updates servers.
CDN example

- **Origin server (www.foo.com)**
  - Distributes HTML

- **CDN company (cdn.com)**
  - Distributes gif files
  - Uses its authoritative DNS server to route redirect requests

![Diagram of CDN example]

1. HTTP request for www.foo.com/sports/sports.html
2. DNS query for www.cdn.com

CDN server near client
More about CDNs

- Routing requests
  - CDN creates a “map”, indicating distances from leaf ISPs and CDN nodes
  - When query arrives at authoritative DNS server:
    - Server determines ISP from which query originates
    - Uses “map” to determine best CDN server
  - CDN nodes create application-layer overlay network
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
Protocols for real-time interactive applications
RTP, RTCP, SIP
Real-Time Protocol (RTP)

- RTP specifies packet structure for packets carrying audio and 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
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.
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 μ-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 bits 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, 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.
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
  - Number of packets sent
  - Number of 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 and RTCP packets are 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 uses 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.
Protocols for real-time interactive applications
Session Initiation Protocol (SIP)
● 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 and 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 μlaw)
- 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 μlaw 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:alice@hereway.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 506
- 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
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.
Protocols for real-time interactive applications
H.323
Alternative to SIP: H.323
Alternative to SIP: H.323

H.323

Audio Codec
G.711
G.723
G.729

Video Codec
H.261
H.263

Data Interface
T.120

Call Setup
Q.931

RAS
Q.931 Interface

Control
H.245

System Control

Transport
RTP
Control
RTP

TCP/IP

Network
Comparison: SIP vs. H.323

- H.323 is another signaling protocol for real-time, interactive
- 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.
Providing multiple classes of service
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

1.5 Mbps link
R1 output interface queue
Scenario 1: mixed FTP and audio

- Example: 1Mbps IP phone and 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 do not use its allocation

Principle 3

While providing isolation, it is desirable to use resources as efficiently as possible
Providing multiple classes of service
Scheduling and Policing Mechanisms
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

![Diagram of queue and link](image)
Scheduling Policies: Priority based

**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: Round Robin

- 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:
  ● Average Rate: how many packets can be sent per unit time (in the long run)
    ● Long term criteria
  ● Crucial question: what is the interval length:
    100 packets per sec or 6000 packets per min have same average!
  ● Peak Rate: e.g., 6000 packets per min. (ppm) avg.; 1500 ppm peak rate
  ● Burst Size: max. number of packets 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., QoS guarantee!

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

WFQ

Arriving traffic

Token rate, \( r \)

Bucket size, \( b \)

Per-flow rate, \( R \)

\[ R \leq \frac{b}{b_1} \]

\[ R \leq \frac{b}{b_2} \]

\[ R \leq \frac{b}{b_n} \]

\[ w_1 \]

\[ w_n \]
Providing multiple classes of service
Differentiated Services
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
- Do not 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
QoS in the Internet: Differentiated Services

- Class-based approach (Differentiated Services, DiffServ): do without guarantees, only manage aggregated data streams
- Thus, the complexity in routers is shifted “at the edge” of the network, internal network routers can be kept simpler.
- Divide the network into domains. A domain is a part of the entire network, which supports DiffServ. It consists of access routers for the domain (Ingress routers, yellow) and internal routers (core routers, blue).
- The domain defines service classes, each data flow is assigned to such a class. With coming into the domain (at a Ingress Router), each packet is assigned a class, and in the network the forwarding bases on the class parameters (per-hop behavior).
  Looking only at the aggregated data flows, a better scalability as for IntServ is achieved.
Application of DiffServ with IP

- Use of the Type of Service field in IPv4 for the classification (DSCP – Differentiated Service Code Point). The DSCP value defines the per-hop behavior of the packet from one router to the next one.

The code point defines the transmission class, which informs a router, how it has to treat the packet in forwarding.
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
Service Classes

- For DiffServ, amongst others the following classes (for specifying transmission behavior) are defined at the moment:
  - Default Forwarding realizes the usual Best Effort transmission
  - Expedited Forwarding tries to emulate a rented line
  - Assured Forwarding uses priorities and discarding probabilities used when an overload of a router is given

![DSCP free]
Expediting Forwarding

- Idea with Expedited Forwarding:
  - there are “regular” and “expedited” packets. Expedited packets are passed on in such a way, as if there would be no or only little further traffic. A minimum data transmission rate is guaranteed.
  - Routers can manage two separated queues for these packet types (Weighted Fair Queuing).
  - Possible: low loss rate/delay/jitter as well as a guaranteed data transmission rate.

![Diagram showing Expedited and Regular packets being processed differently.]
Assured Forwarding

- Improves differentiation: Definition of four priority classes with own resources (at the moment; there also is room for more classes)
- For each class, three probabilities for discarding a packet are defined: low, medium, high
- Thus: altogether 12 different service classes
- Principle:
  - The priority class determines the portion of the transmission capacity of the routers
  - During high load packets of lower priority would be discarded completely
  - Fairness: packets of each priority class should have chances to survive
  - Therefore definition of the probabilities for each class: by suitable selection of the probabilities, a small part of the lowest priority level still is still forwarded, while packets of higher priority classes are already discarded.
Assured Forwarding

1. Classification of the packets in service classes

2. Appropriate choice of DSCP tags

3. Bring the data streams in a form according to their flow specification. Exceeding the specifications leads to discarding data. If thereafter still too much data are present, discarding of packets in accordance with their probability

4. Weighted Fair Queuing in accordance to priority classes
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)

- Per hop behavior (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
Providing QoS guarantees
Principles for QOS Guarantees (more)

- Basic fact of life: cannot support traffic demands beyond link capacity

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

connectionless (stateless) forwarding by IP routers + best effort service = no network signaling protocols in initial IP design

- 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
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