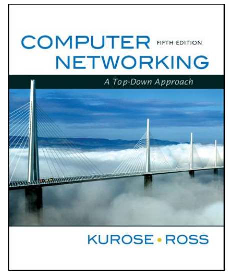
Lecture 9: Media over IP



Computer Networking: A Top Down Approach
5th edition.
Jim Kurose, Keith Ross
Addison-Wesley, April 2009.

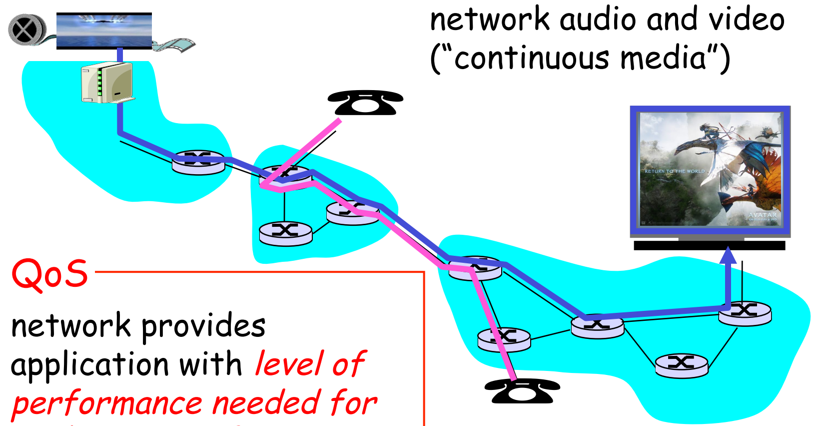
These slides are adapted from the slides provided by the authors of the book (to the right), available from the publisher's website.

Announcement

- No class this Wed.
  - Prepare for Mid-term
  - Work on Lab 2
- Lab 2 assigned
  - Due May 11 (part I)
  - Due May 18 (part II)
  - 2 students per group, find your partner now.

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.*

Roadmap

- **Multimedia networking applications**
- RTP & RTCP
- Streaming stored audio & video (RTSP)
- Internet video phone (SIP, H.323)
- Recovery from loss

MM Networking Applications

Classes of MM applications:

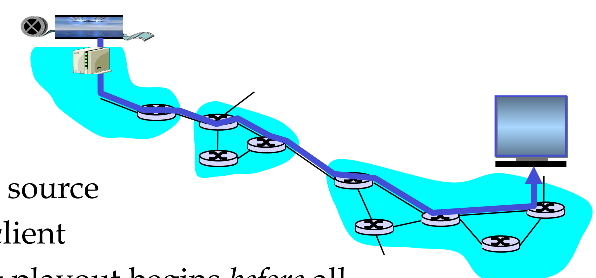
1) Streaming stored audio & video
2) Streaming live audio & video
3) Interactive, real-time audio & video

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

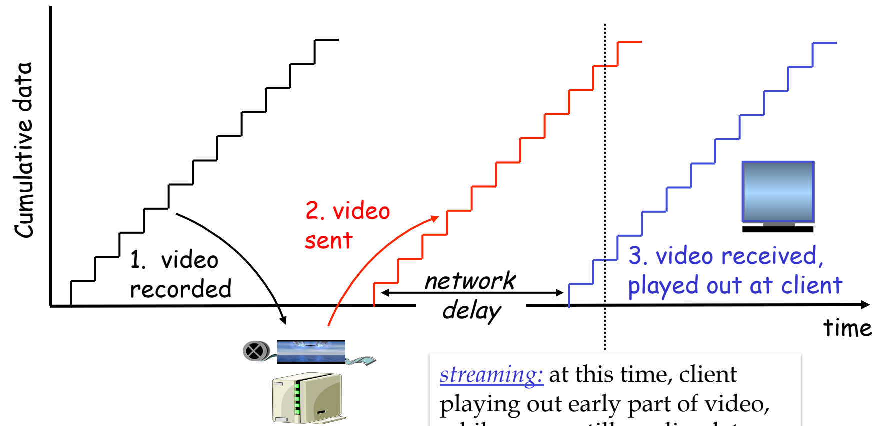
App 1: Streaming Stored Multimedia



Stored streaming:

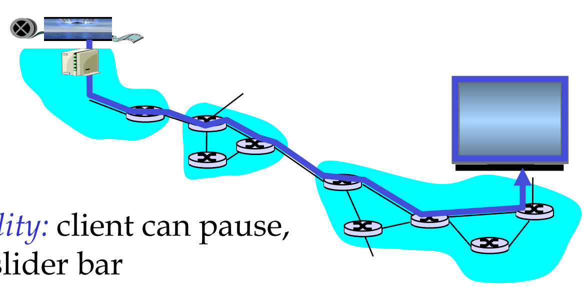
- media stored at source
- transmitted to client
- *streaming:* client playout begins *before* all data has arrived
- timing constraint for still-to-be transmitted data: in time for playout

Streaming Stored Multimedia: What is it?



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

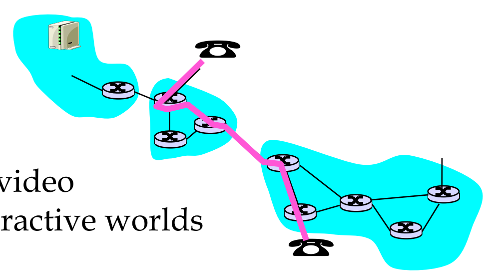
App 2: Streaming *Live* Multimedia

Examples:
- Internet radio talk show
- live sporting event

Streaming
- playback buffer
- playback can lag tens of seconds after transmission
- still have timing constraint

Interactivity
- fast forward **impossible**
- rewind, pause possible!

App 3: Real-Time Interactive Multimedia

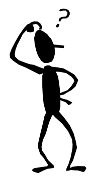


- applications: IP telephony, video conference, distributed interactive worlds
- end-end delay requirements:
  - audio: < 150 msec good, < 400 msec OK
    - includes application-level (packetization) and network delays
    - higher delays noticeable, impair interactivity
- session initialization
  - callee advertises 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 requires ?
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

Roadmap

- Multimedia networking applications
- **Transport level: RTP & RTCP**
- **App level**: Streaming stored audio & video (RTSP)
- **App level**: Internet video phone (SIP, H.323)
- **App level**: Recovery from loss

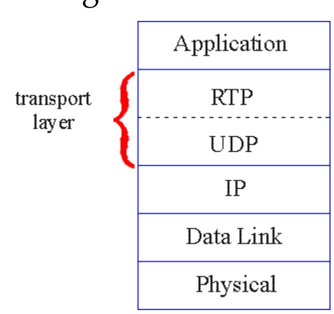
Real-Time Protocol (RTP)

- 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



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 Header

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.

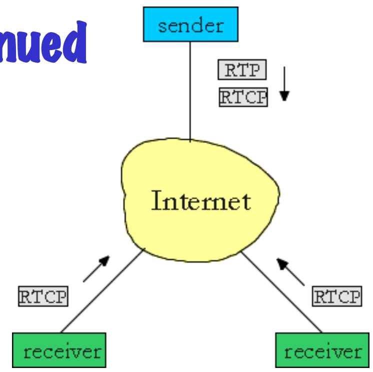
RTP Header (2)

- *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, 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: # 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

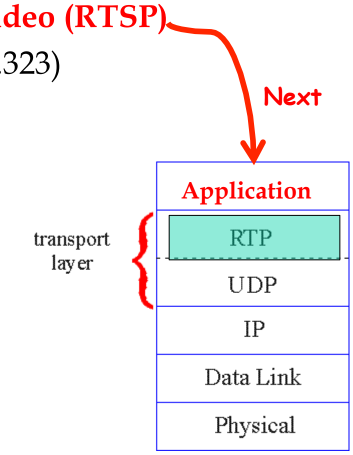
- RTCP can synchronize different media streams within a RTP session
- consider videoconferencing app for which each sender generates one RTP stream for video, one for audio.
- timestamps in RTP packets tied to the video, audio sampling clocks
  - *not* tied to wall-clock time
- each RTCP sender-report packet contains (for most recently generated packet in associated RTP stream):
  - timestamp of RTP packet
  - wall-clock time for when packet was created.
- receivers uses association to **synchronize playout of audio, video**

RTP and QoS

- RTP does **not** provide any mechanism to ensure timely data delivery or other QoS guarantees.
- But together with RTCP, it allows **monitoring** of QoS so that sender/receiver can adjust their operations accordingly
- 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.

Roadmap

- Multimedia networking applications
- RTP & RTCP
- **Streaming stored audio & video (RTSP)**
- Internet video phone (SIP, H.323)
- Recovery from loss



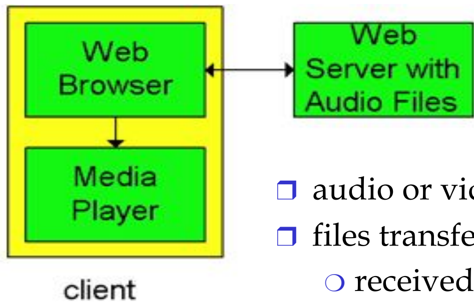
Streaming Stored Multimedia

- establish and control media sessions between end points
- 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

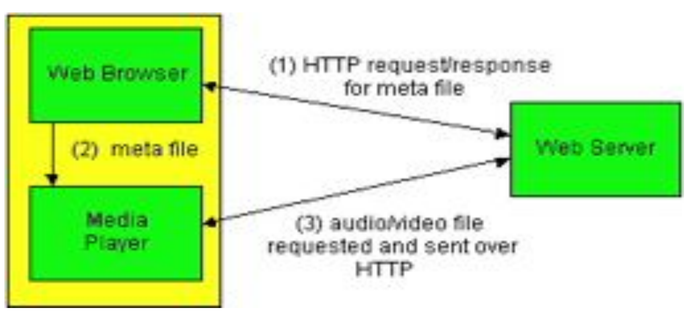


- audio or video stored in file
- files transferred as HTTP object
  - received in **entirety** at client
  - then passed to player

audio, video not streamed:

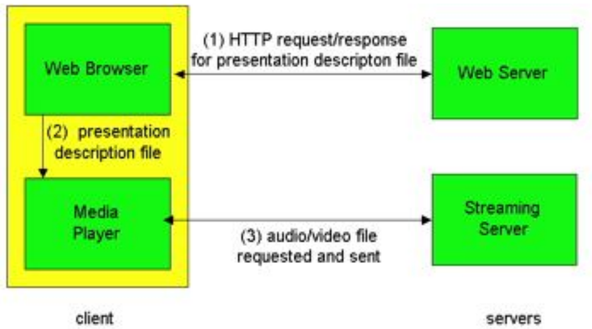
- no "pipelining," long delays until playout!

Internet multimedia: streaming approach



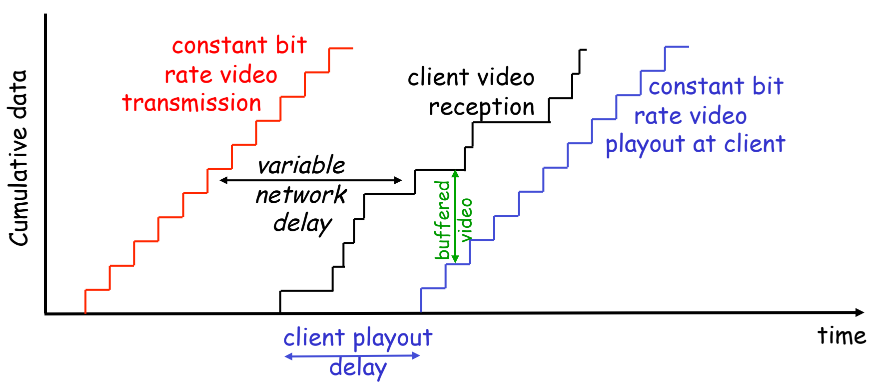
- browser GETs **metafile**
- browser launches player, passing metafile
- player contacts server
- server **streams** audio/video to player

Streaming from a streaming server



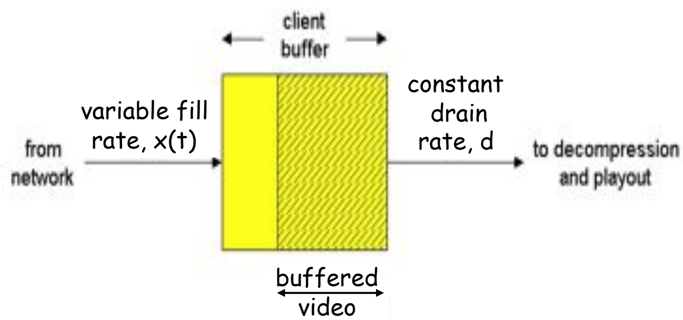
- allows for non-HTTP protocol between server, media player
- UDP or TCP for step (3), more shortly

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?

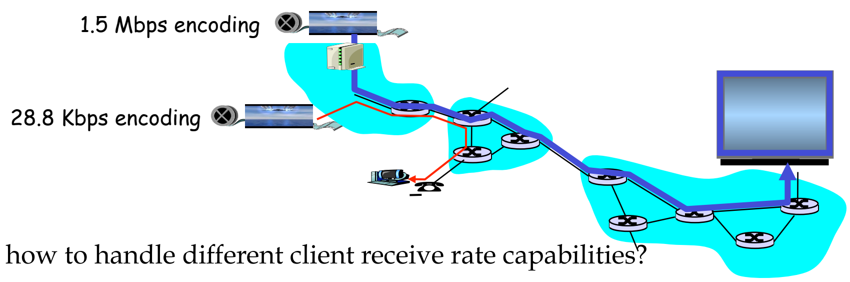
UDP

- 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

Streaming Multimedia: client rate(s)



Q: how to handle different client receive rate capabilities?

- 28.8 Kbps dialup
- 100 Mbps Ethernet

A1: server stores, transmits multiple copies of video, encoded at different rates

A2: server encodes video in scalable mode that can be retrieved at different rates

User Control of Streaming Media: RTSP

HTTP

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

RTSP: RFC 2326

- client-server application layer protocol
- user control (**VCR-like commands)**: 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

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

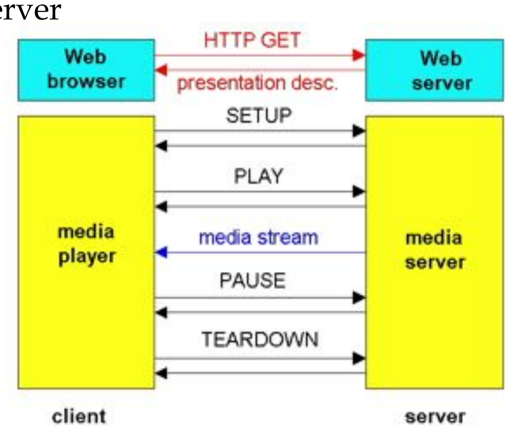
RTSP messages also sent out-of-band:

- 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



RTSP Implementation

Server

- **QuickTime Streaming Server**: Apple's closed-source streaming server that ships with Mac OS X Server
- **Darwin Streaming Server**: Open-sourced version of QuickTime Streaming Server maintained by Apple
- **pvServer**: Formerly called PacketVideo Streaming Server, this is Alcatel-Lucent's streaming server product.
- **Helix DNA Server**: RealNetworks' streaming server. Comes in both open-source and proprietary flavors.
- **Live555**: Open source C++ server and client libraries used in well known clients like VLC and mplayer.
- **VideoLAN**: Open source media player and streaming server
- **Windows Media Services**: Microsoft's streaming server included with Windows Server.

RTSP Implementation

Client

- QuickTime
- RealPlayer
- VLC media player
- Windows Media Player

Roadmap

- Multimedia networking applications
- RTP & RTCP
- Streaming stored audio & video (RTSP)
- **Internet video phone (SIP, H.323)**
- Recovery from loss

Real-time interactive applications

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

Now look at a PC-2-PC Internet phone example in detail

Interactive Multimedia: Internet Phone

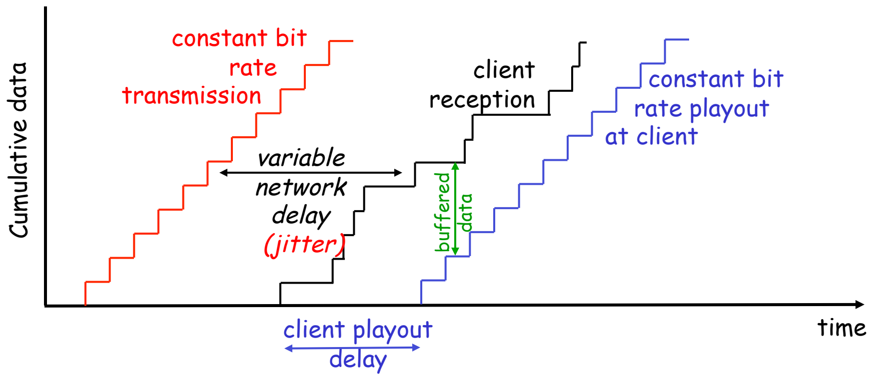
Introduce Internet Phone by way of an example

- 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



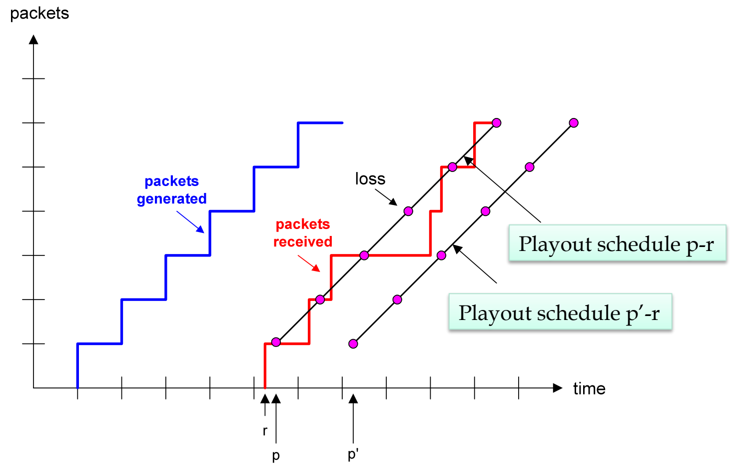
- consider end-to-end delays of two consecutive packets: difference can be more or less than 20 msec (transmission time difference)

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′



SIP: Session Initiation Protocol [RFC 3261]

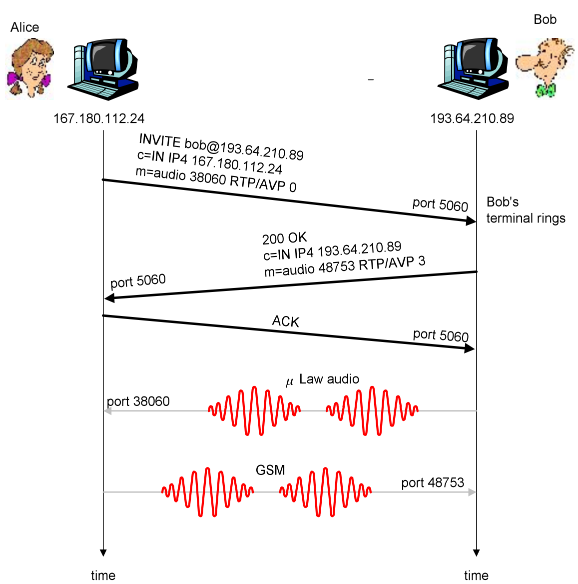
SIP long-term vision:

- 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

Name translation and user locataion

- 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
```

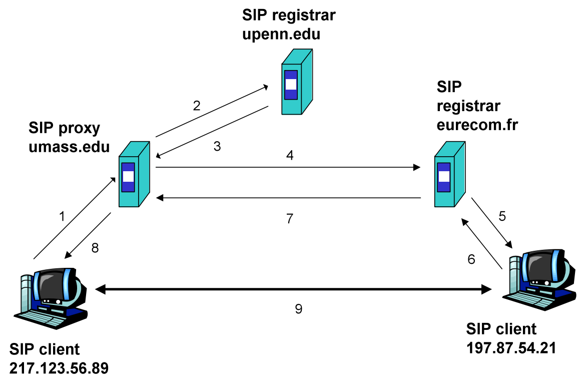
SIP Proxy

- 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.

H.323

- A protocol suite that defines multimedia conferencing over internet
  - Control & data channels sent using TCP
  - Audio and video streams sent via RTP/UDP
  - Specifies the operation of registra, gateway ….

Comparing SIP with 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.

Media over IP (Internet): Making it Work

- **Use UDP** to avoid TCP congestion control and the delay associated with it; required for time-sensitive media traffic
- **Use RTP/UDP** to enable QoS monitoring, sender and receiver can record the # of packets sent/received and adjust their operations accordingly
- Client-side uses **adaptive playout delay** to compensate for the delay (and the jitter)
- Server side matches stream bandwidth to available client-to-server path bandwidth
  - Chose among pre-encoded stream rates
  - Dynamic encoding rate

Roadmap

- Multimedia networking applications
- RTP & RTCP
- Streaming stored audio & video (RTSP)
- Internet video phone (SIP, H.323)
- Recovery from loss

Recovery from packet loss (1)

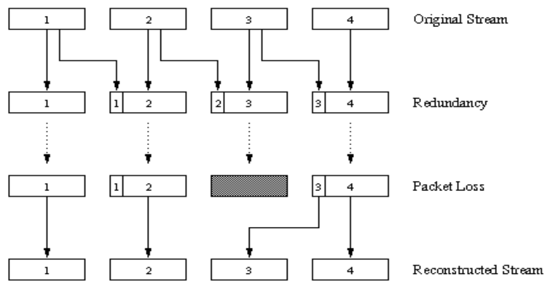
Forward Error Correction (FEC): simple scheme

- 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

Unequal FEC

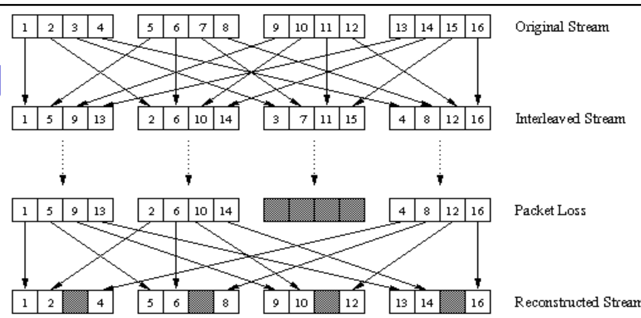
2nd FEC scheme

- "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

Interleaving



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

Media over IP (Internet): Making it Work

- Use UDP to avoid TCP congestion control and the delay associated with it; required for time-sensitive media traffic
- Use RTP/UDP to enable QoS monitoring, sender and receiver can record the # of packets sent/received and adjust their operations accordingly
- Client-side uses adaptive playout delay to compensate for the delay (and the jitter)
- Server side matches stream bandwidth to available client-to-server path bandwidth
  - Chose among pre-encoded stream rates
  - Dynamic encoding rate
- Error recovery (on top of UDP)
  - FEC and/or interleaving
  - Retransmissions (time permitting)
  - Unequal error protection (duplicate important parts)
  - Conceal errors (interpolate from nearby data)