Multimedia: Real-time Conversations

CS 352, Lecture 23

http://www.cs.rutgers.edu/~sn624/352-S19

Srinivas Narayana

(heavily adapted from slides by Prof. Badri Nath and the textbook authors)

RUTGERS
UNIVERSITY | NEW BRUNSWICK

Review: 3 types of multimedia networking

- *streaming, stored* audio, video
  - *streaming:* can begin playout before downloading entire file
  - *stored (at server):* can transmit faster than audio/video will be rendered (implies storing/buffering at client)
  - e.g., YouTube, Netflix, Hulu
- *conversational* voice/video over IP
  - interactive nature of human-to-human conversation limits delay tolerance
  - e.g., Skype
- *streaming live* audio, video
  - e.g., live sporting event (futbol)

Voice over IP (VoIP)

Voice-over-IP (VoIP)

- *VoIP end-end-delay requirement*: needed to maintain "conversational" aspect
  - higher delays noticeable, impair interactivity
  - < 150 msec: good
  - > 400 msec: bad
  - includes application-level (packetization, playout), network delays
- *session initialization:* how does callee advertise IP address, port number, encoding algorithms?
- *value-added services:* call forwarding, screening, recording
- *emergency services:* 911

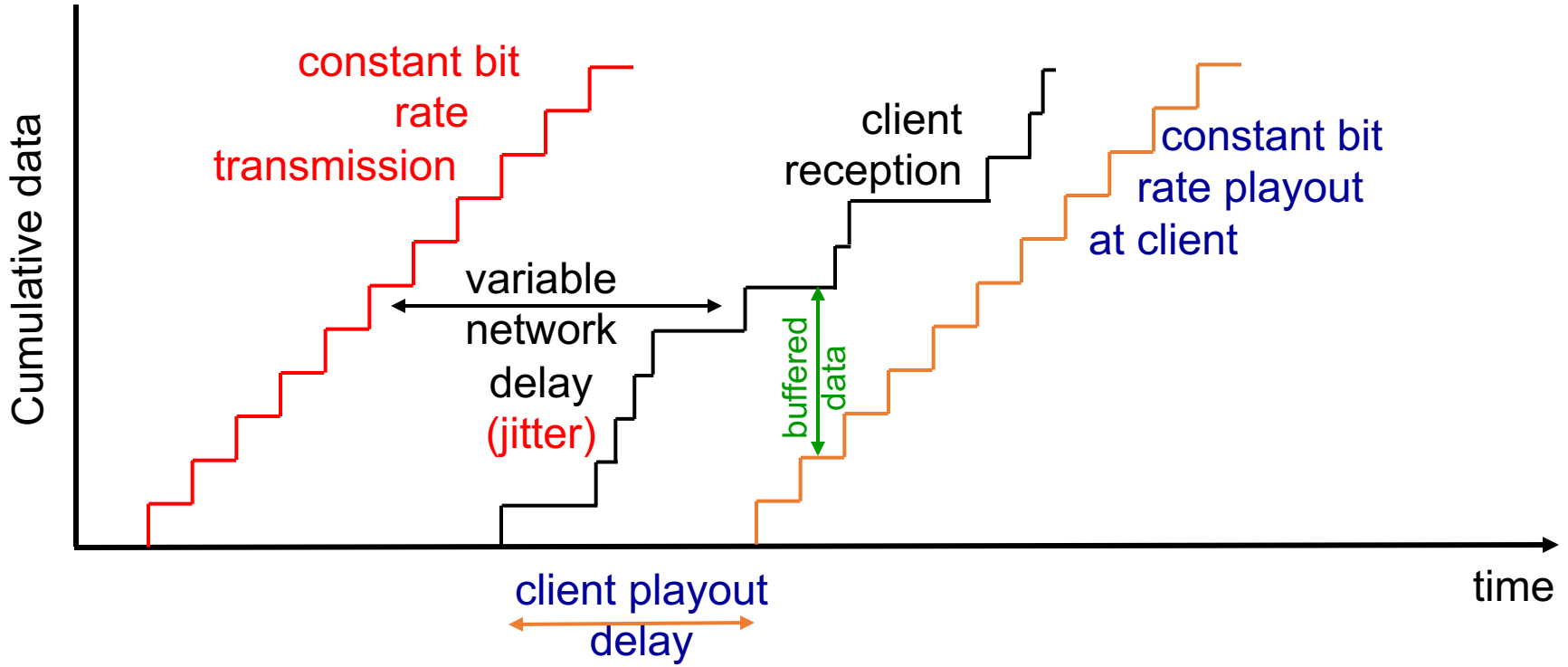
VoIP characteristics

- speaker's audio: alternating talk spurts, silent periods
  - 8K 1-Byte samples per second: 8 KBps during talk spurt
  - Packets generated only during talk spurts
  - 20 msec chunks at 8 KBytes/sec: 160 Bytes of data
- application-layer header added to each chunk
- Chunk + header encapsulated into UDP or TCP segment
- Application sends segment into socket every 20 msec during talk spurt

VoIP: packet loss, 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 and loss concealment, packet loss rates between 1% and 10% can be tolerated

Delay jitter



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

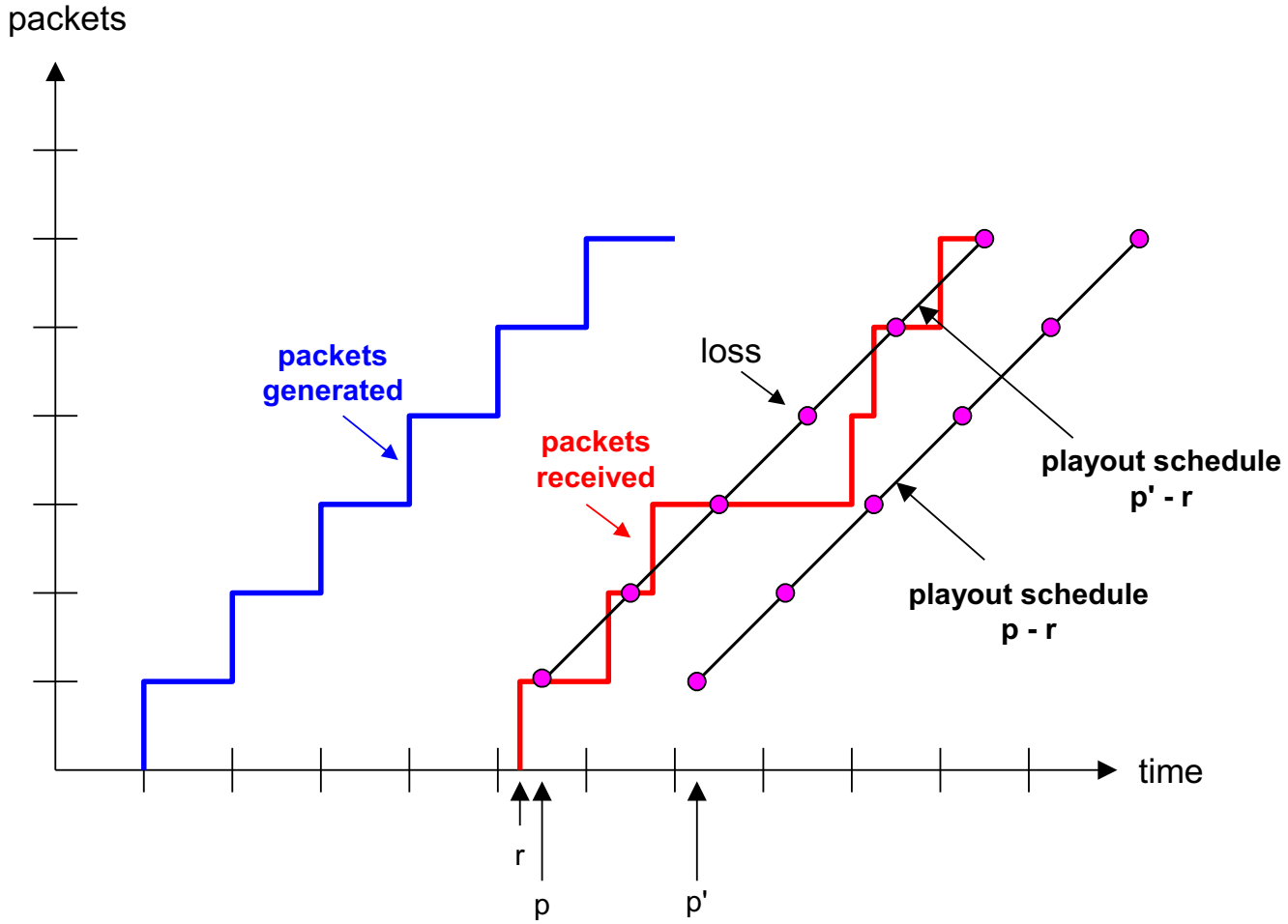
VoIP: fixed playout delay

- 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 but poor *interactivity*
  - *small q:* better interactive experience but more "loss"

VoIP: 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 at p’



Adaptive playout delay (1/3)

- *goal:* low playout delay, low late loss rate
- *approach:* adapt playout delay at the level of a talk spurt
  - estimate network delay, adjust playout delay @ beginning of talk spurt
  - silent periods will be compressed and elongated: not that noticeable
  - Goal: chunks still played out every 20 msec during talk spurt

- adaptively estimate packet delay:

$$d_i = (1-\alpha)d_{i-1} + \alpha\,(r_i - t_i)$$

- d_i: delay estimate after ith packet
- α: small constant, e.g. 0.1
- r_i: time received
- t_i: time sent (timestamp)
- $r_i - t_i$: measured delay of ith packet

Adaptive playout delay (2/3)

- also useful to estimate average deviation of delay, v_i :

$$v_i = (1-b)v_{i-1} + b * abs(r_i - t_i - d_i)$$

  - 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:

$$playout\text{-}time_i = t_i + d_i + Kv_i$$

  - remaining packets in talk spurt are played out periodically

Adaptive playout delay (3/3)

Q: How does receiver determine whether packet is first in a talk spurt?

- if no loss, receiver looks at successive timestamps
  - difference of successive stamps > 20 msec ➔ talk spurt begins.

- with possible losses, receiver must look at both time stamps and sequence numbers
  - difference of successive stamps > 20 msec *and* sequence numbers without gaps --> talk spurt begins.

VoIP: recovery from packet loss (1/4)

Challenge: recover from packet loss given small tolerable delay between original transmission and playout

- Often, retransmission isn't effective:

- If playout time has expired, no point in retransmitting data!
- Even if playout time is large and hasn't expired during the loss, detecting the loss takes an ACK/NAK
  - By which time one RTT has elapsed!
  - Retransmitted data to get to receiver takes at least another ½ RTT
  - It may be too late!

VoIP: recovery from packet loss (2/4)

Forward Error Correction (FEC)

- Send enough bits to allow recovery *without retransmission*
- Recall *parity* from the network layer?

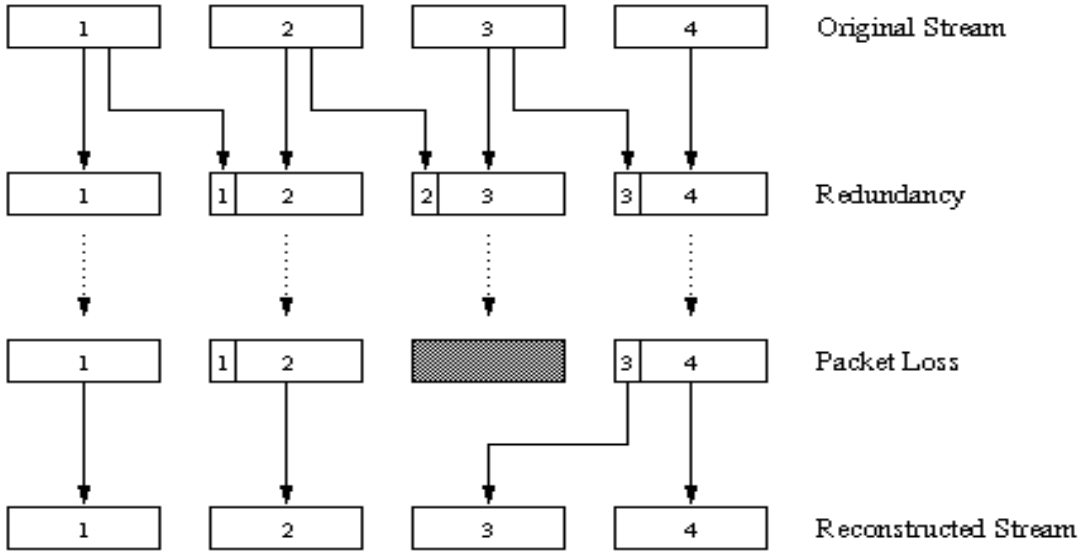
Simple FEC

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

VoIP: recovery from packet loss (3/4)

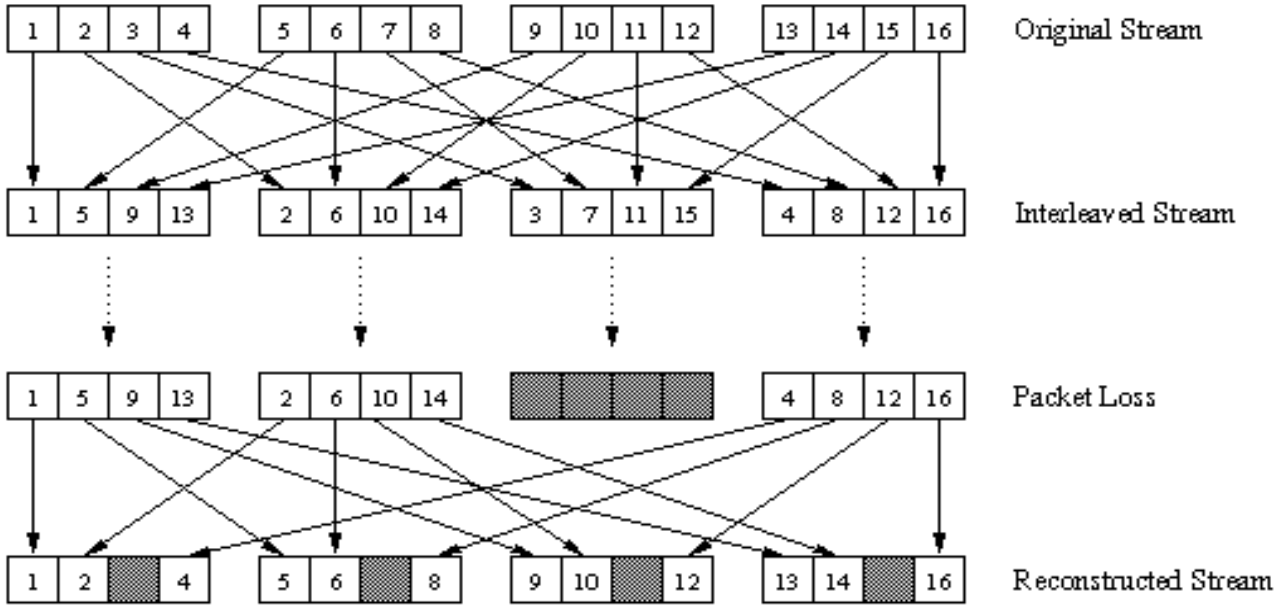
Piggybacking FEC scheme

- send lower resolution audio stream as redundant information along with (the usual) higher quality stream
- e.g., nominal stream PCM at 64 kbps and redundant stream GSM at 13 kbps



- non-consecutive loss: receiver can conceal loss
- generalization: can also append (n-1)st and (n-2)nd low-bit rate chunk

VoIP: recovery from packet loss (4/4)



interleaving to conceal loss:

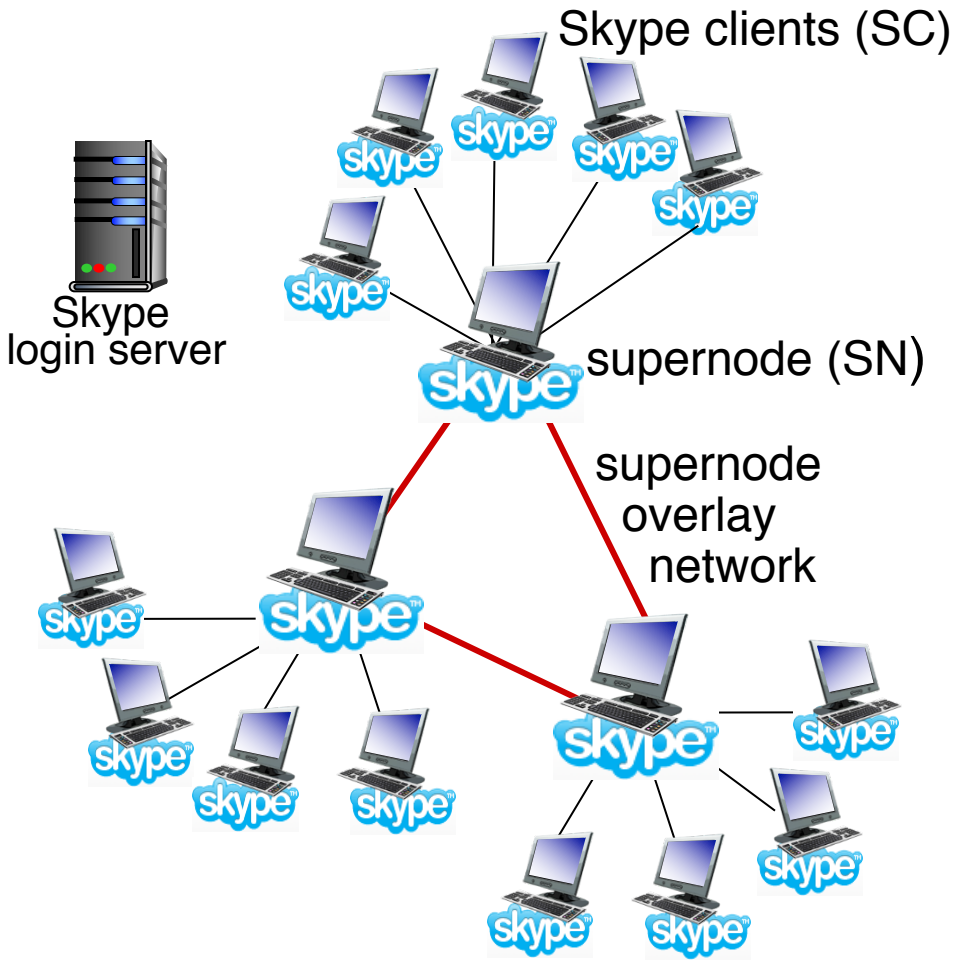
- audio chunks divided into smaller units, e.g. four 5 msec units per 20 msec audio chunk
- packet contains small units from different chunks

- if packet lost, still have *most* of every original chunk
- no redundancy overhead, but increases playout delay

Case study: Skype

Voice-over-IP: Skype

- proprietary application-layer protocol (inferred via reverse engineering)
  - encrypted msgs
- P2P components:
  - clients: Skype peers connect directly to each other for VoIP call
  - super nodes (SN): Skype peers with special functions
  - overlay network: among SNs to locate SCs
  - login server



P2P voice-over-IP: Skype

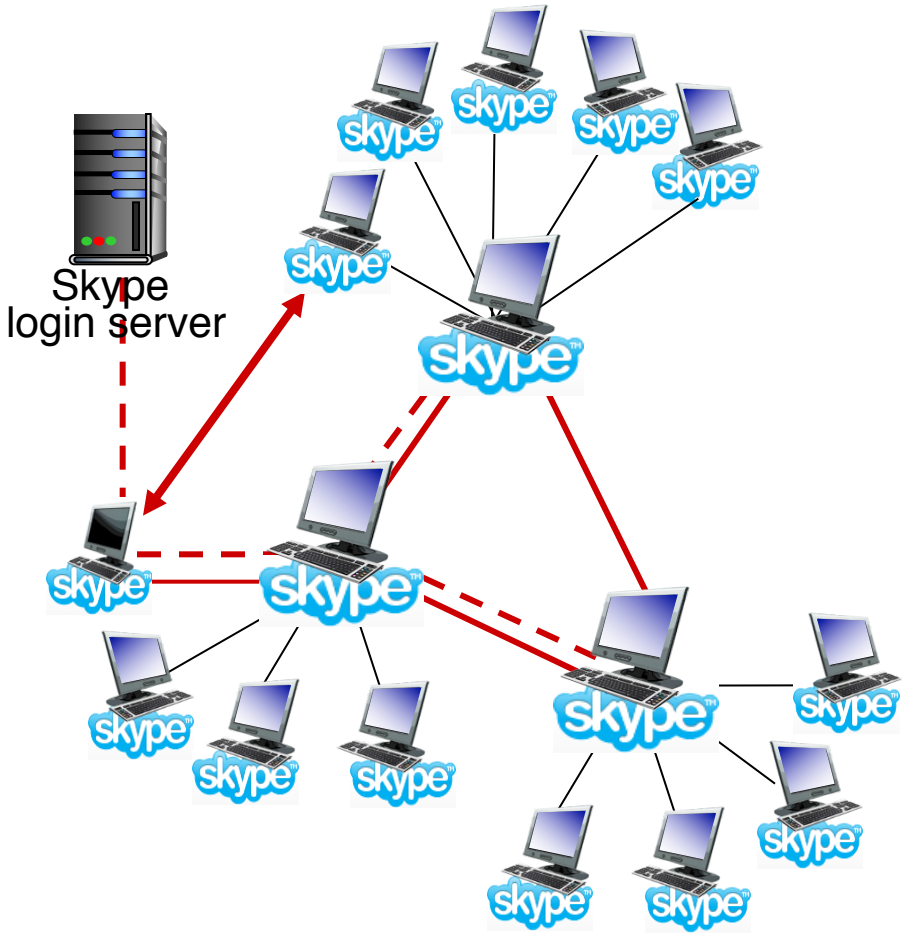
Skype client operation:

1. joins Skype network by contacting SN (IP address cached) using TCP

2. logs-in (username, password) to centralized Skype login server

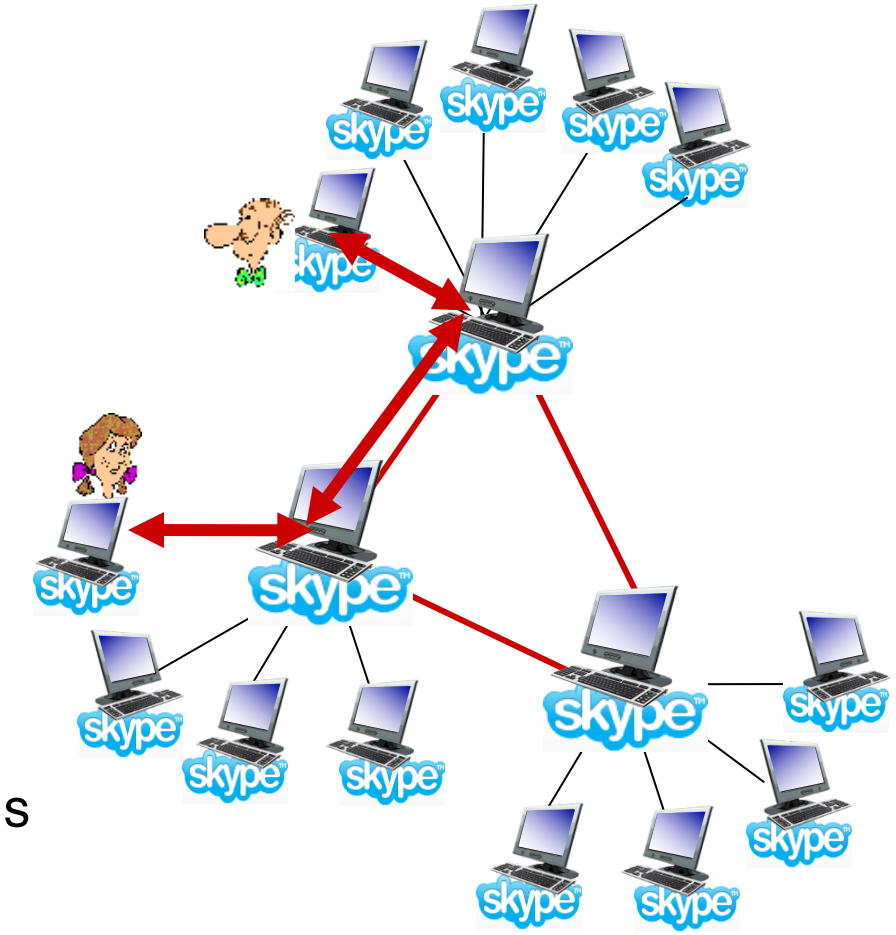
3. obtains IP address for callee from SN, SN overlay
   - or client buddy list

4. initiate call directly to callee



Skype: peers as relays

- *problem:* both Alice, Bob are behind "NATs"
  - NAT prevents outside peer from initiating connection to insider peer
  - inside peer *can* initiate connection to outside
- relay solution: Alice, Bob maintain open connection to their SNs
  - Alice signals her SN to connect to Bob
  - Alice's SN connects to Bob's SN
  - Bob's SN connects to Bob over open connection Bob initially initiated to his SN



Skype: peers as relays

- Pro: A solution to circumvent NATs on both sides of a connection

- Con: requires infrastructure outside of NAT (supernode peers)
- Con: data must be routed through SNs, which may become bottlenecks

- Note: Information up to date as of 2012
  - We need fresh reverse engineering studies to verify if conclusions are applicable now ☺

Protocols for real-time communication: RTP and SIP

Real-Time Protocol (RTP)

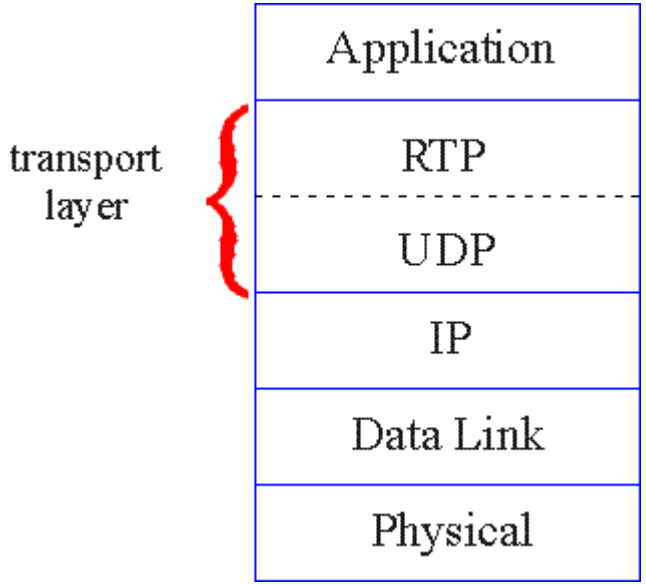
- RTP specifies packet structure for packets carrying audio, video data
- RFC 3550
- RTP packet provides
  - payload type identification
  - packet sequence numbering
  - time stamping

- RTP runs in end systems
- RTP packets encapsulated in UDP segments
- interoperability: if two VoIP applications run RTP, 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

example: 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

payload type (7 bits): indicates type of encoding currently being used. If sender changes encoding during call, 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 # (16 bits): increment by one for each RTP packet sent

- detect packet loss, restore packet sequence

RTP header

- *timestamp field (32 bits long):* sampling instant of first byte in this RTP data packet
  - for audio, timestamp clock increments by one for each sampling period (e.g., each 125 usecs for 8 KHz sampling clock)
  - if application generates chunks of 160 encoded samples, 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 RTP stream. Each stream in RTP session has distinct SSRC

SIP: Session Initiation Protocol [RFC 3261]

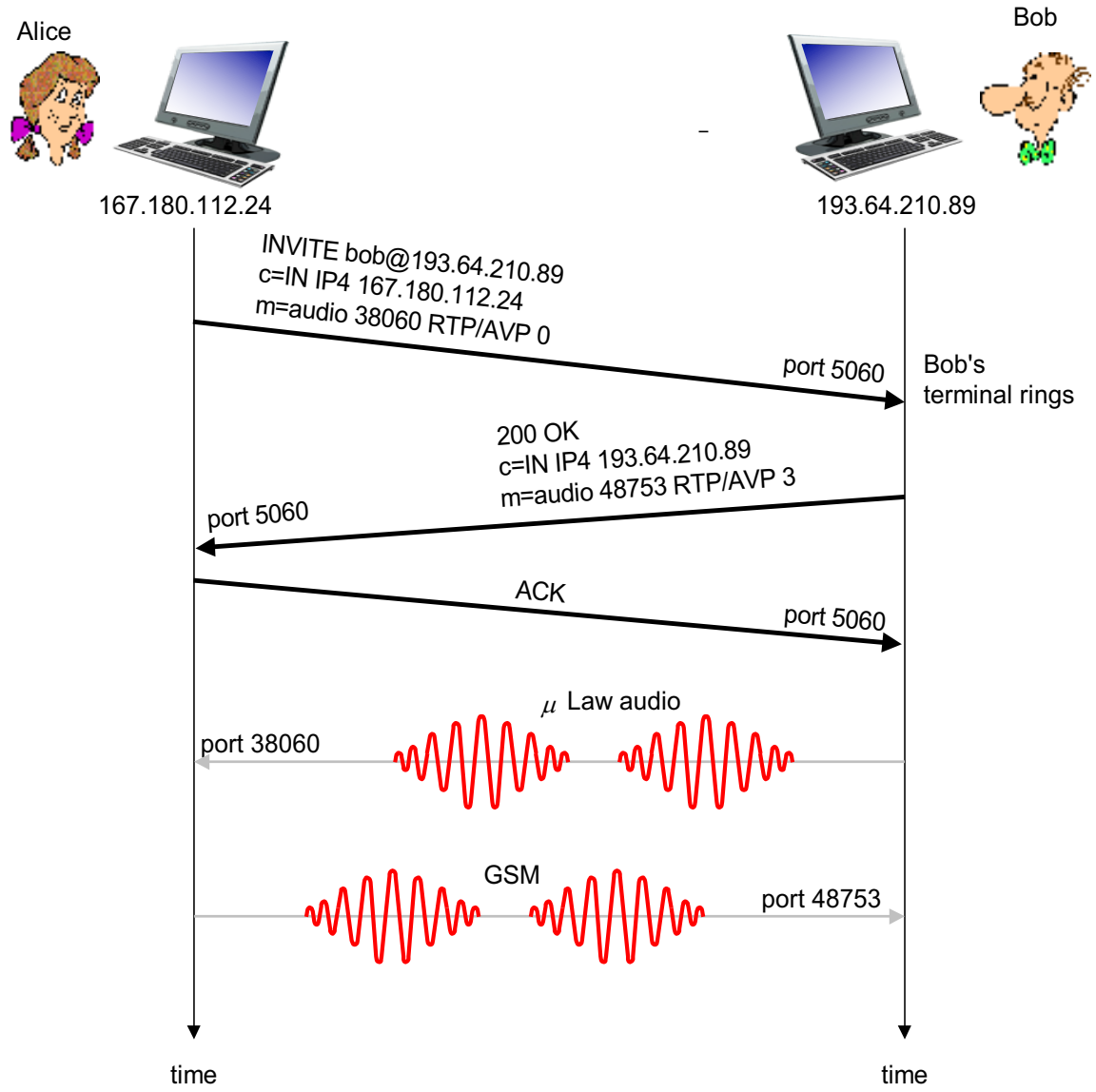
long-term vision:

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

SIP services

- SIP provides mechanisms for call setup:
  - for caller to let callee know she wants to establish a call
  - caller, callee can agree on media type, encoding
  - Possible to end call with specific reasons
- 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

Example: setting up call to known IP address



- Alice's SIP invite message indicates her port number, IP address, encoding she prefers to receive (PCM mu-law)
- Bob's 200 OK message indicates his port number, IP address, preferred encoding (GSM)
- SIP messages can be sent over TCP or 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: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 506
- Alice specifies in header that SIP client sends, receives SIP messages over UDP

Name translation, 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, smartphone, 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)

SIP registrar

- one function of SIP server: registrar
- when Bob starts SIP client, client sends SIP REGISTER message to Bob's registrar server

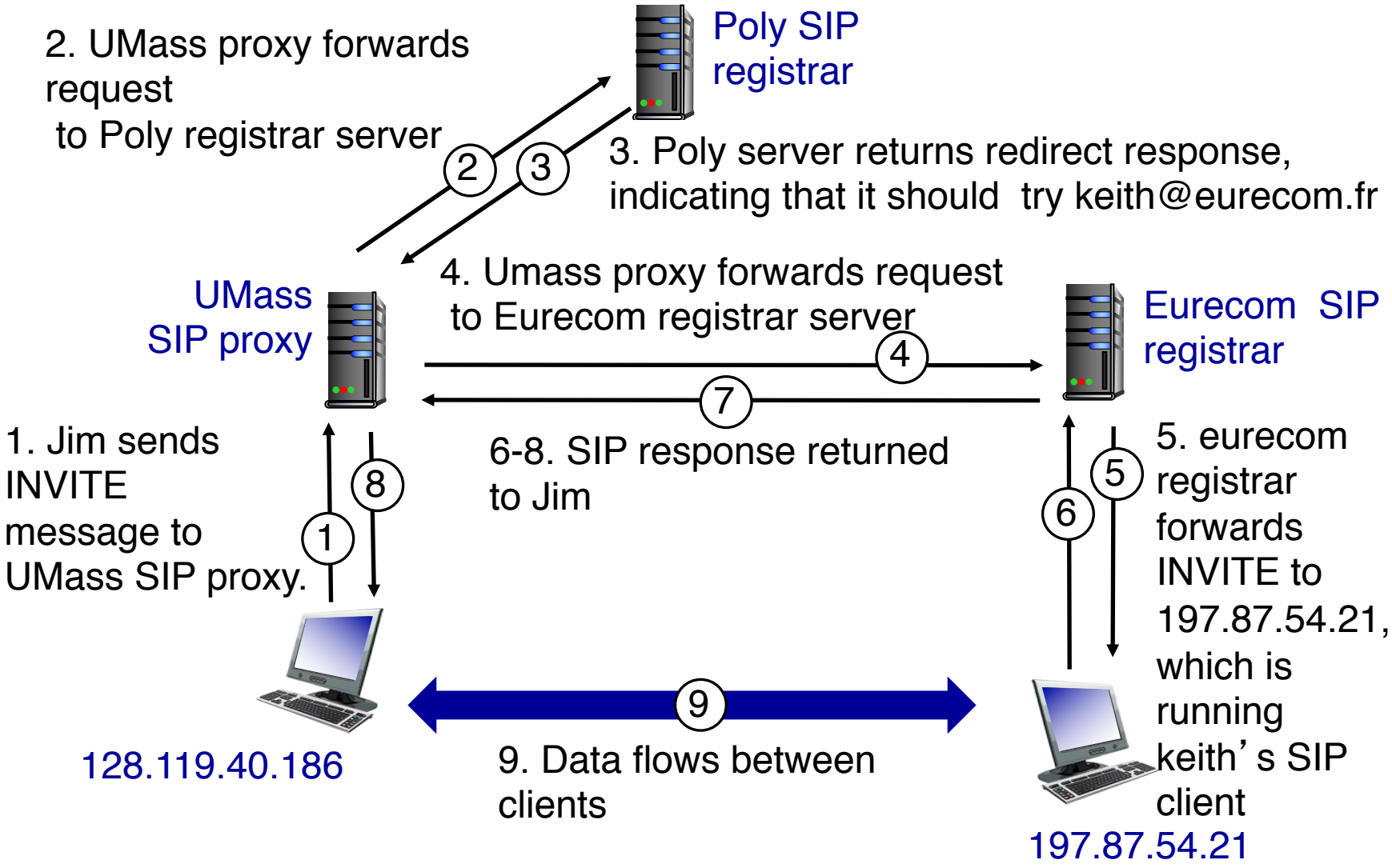
register message: in principle similar to IP mobility home agent

```
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

- another function of SIP server: *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
- Bob sends response back through same set of SIP proxies
  - Similar to mobile IP indirect routing
- proxy returns Bob's SIP response message to Alice
  - contains Bob's IP address
- SIP proxy analogous to local DNS server plus indirect routing relay

SIP example: jim@umass.edu calls keith@poly.edu



Internetworking PSTN and SIP?

- A gateway is required to convert SIP messages to circuit switched signaling in the public switched telephone network (PSTN) and vice-versa

- SIP proxy server will contact PSTN gateway
- A PSTN gateway initiates call to the PSTN callee
- Two-way audio conversation occurs through the PSTN gateway

Summary of real-time multimedia

- Important to bound playout delays, adapt to loss
- Fixed and adaptive playout delays at the granularity of “talk spurts”
- Forward error correction mechanisms to avoid retransmissions and conceal packet loss
- Relay-based call routing: used by Skype and SIP services
  - Useful to overcome NATs
  - Locate users through generic names, amidst mobility
  - Need extra infrastructure to make all this real