Multimedia: Real-time Conversations

CS 352, Lecture 24, Spring 2020

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

Srinivas Narayana

RUTGERS
UNIVERSITY | NEW BRUNSWICK

Course announcements

- Project 3 due today

- No more quizzes!

- Final exam over a 5-day window (05/07—05/12)
  - If you need any special arrangements, you must contact me ASAP
  - Exam on Sakai; 2-hour block
  - Open book. Same conditions and honor code as second mid-term
  - No googling or collaboration allowed

Review of concepts

- DASH: media presentation description; dynamic, adaptive
- Voice over IP (VoIP):
  - Real-time guarantees: need delay < 400ms
  - Delay loss in addition to (congestive) loss
  - Adaptive playout: adjust playout delay for each talk spurt
  - Loss recovery using forward error correction & interleaving
- Case study: Skype: peers as relay nodes
  - Connectivity across NATs requires a special infrastructure
  - A common theme to provide media connectivity: SIP
  - This lecture: RTP and SIP

Protocols for real-time communication: RTP and SIP

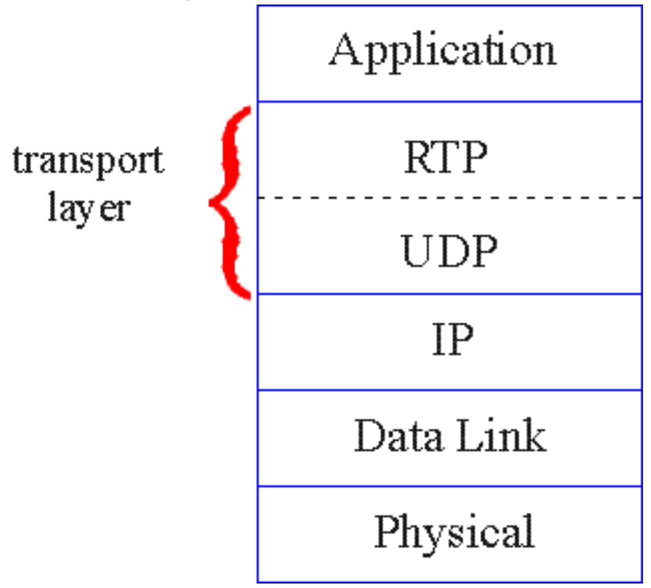
Real-Time Protocol (RTP)

- 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



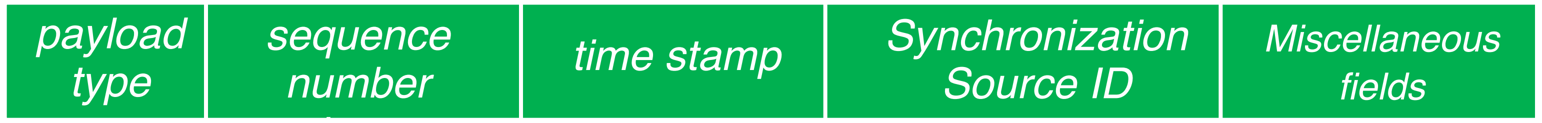
RTP example

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



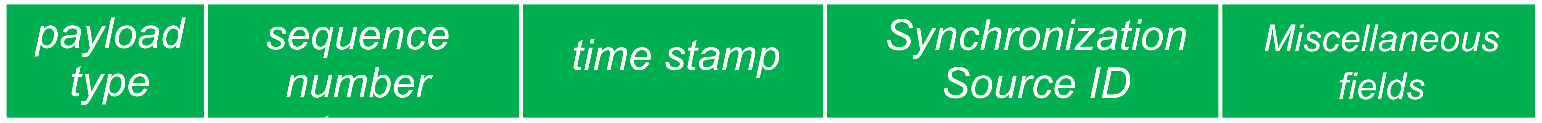
payload type (7 bits): indicates type of encoding currently being used. If sender changes encoding during call, sender informs receiver via payload type field

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]

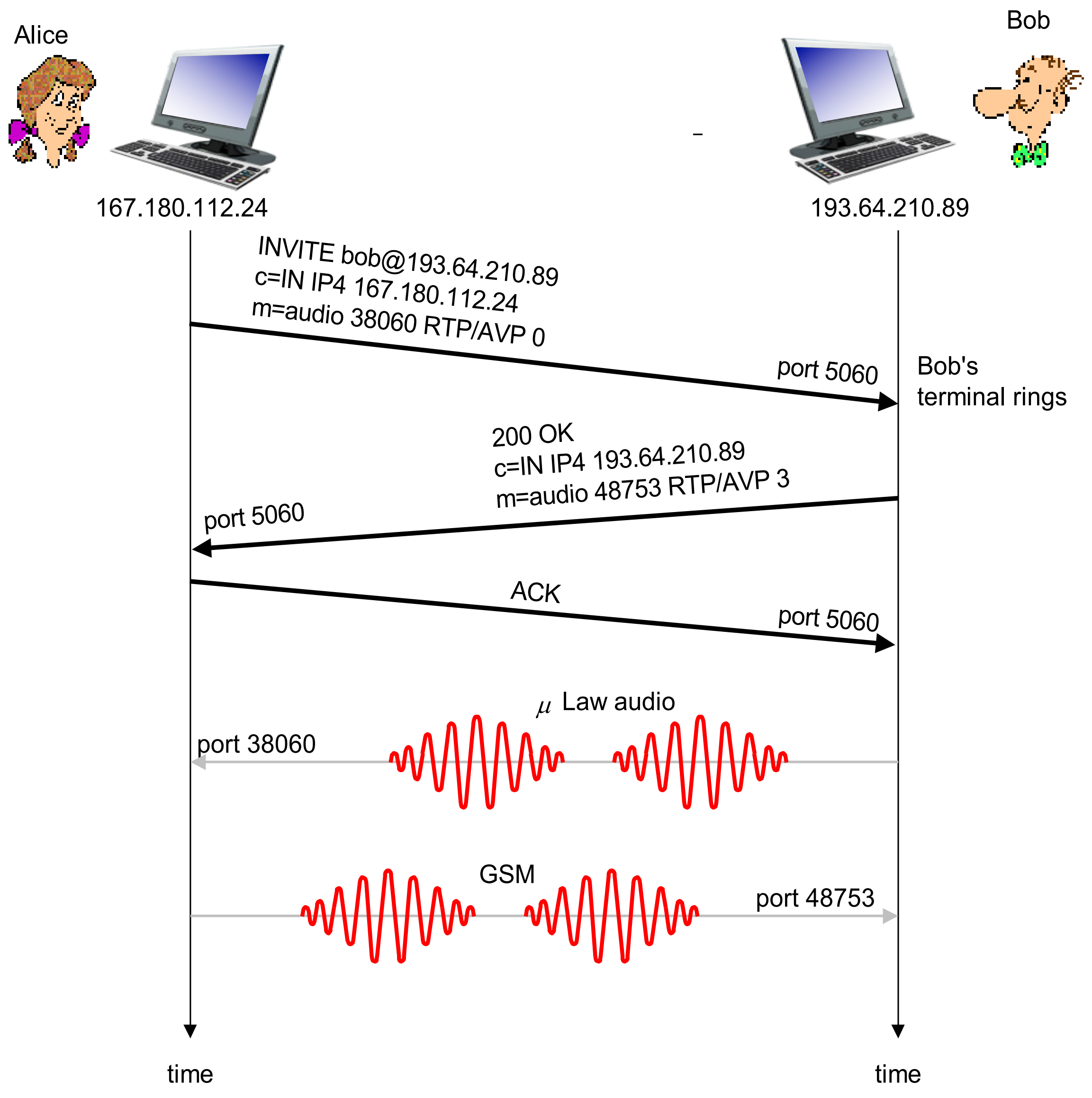
long-term vision:

- 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 5060
- 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)

An infrastructure of SIP servers to support mobility and callee resolution

SIP registrar

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

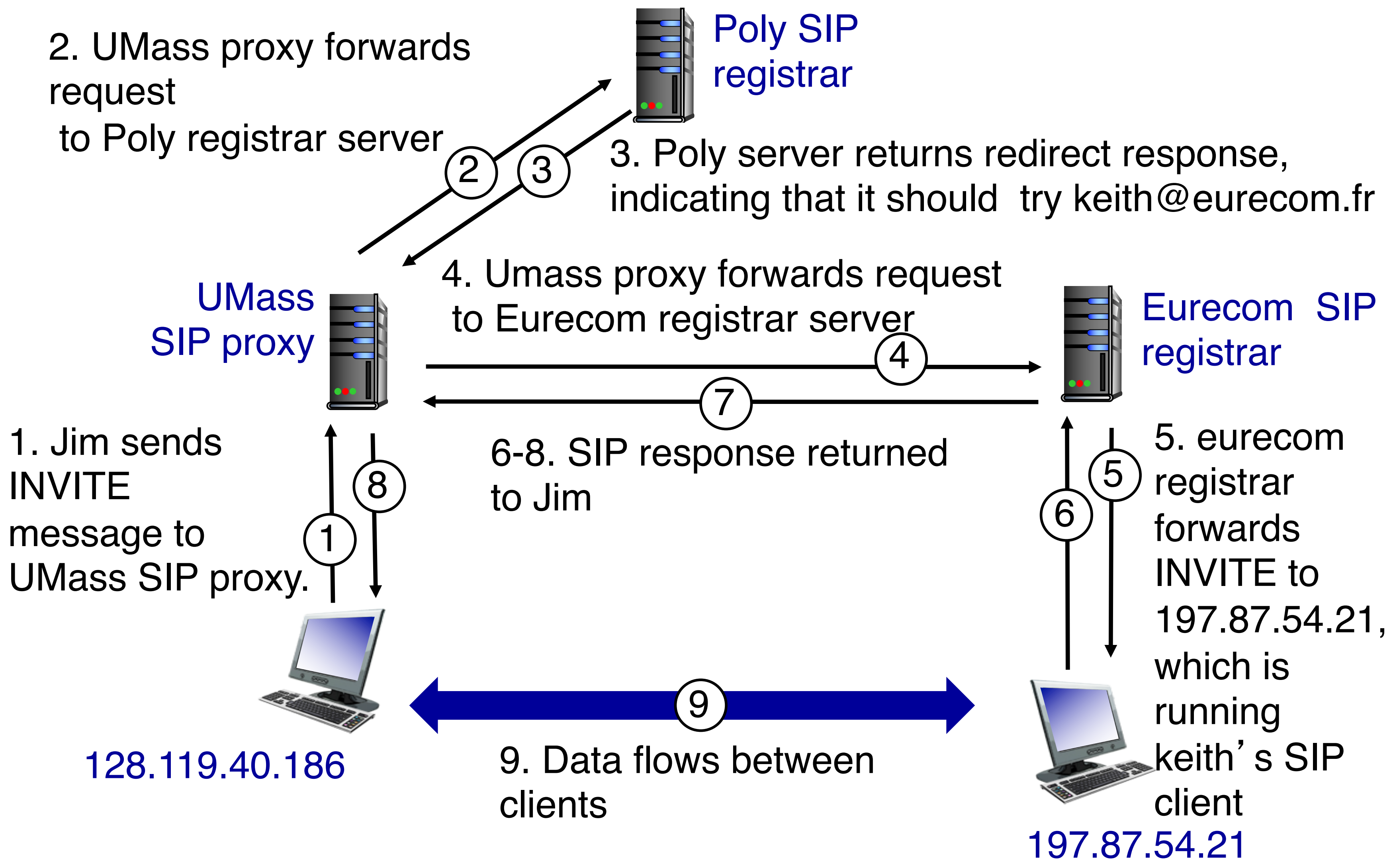
register message: to support mobility at Bob's end

```
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 the same set of SIP servers
- Alice's 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
- Alice and Bob's media traffic sent directly between each other

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


Poly SIP registrar
2. UMass proxy forwards request to Poly registrar server
3. Poly server returns redirect response, indicating that it should try keith@eurecom.fr
UMass SIP proxy
4. Umass proxy forwards request to Eurecom registrar server
Eurecom SIP registrar
1. Jim sends INVITE message to UMass SIP proxy.
6-8. SIP response returned to Jim
5. eurecom registrar forwards INVITE to 197.87.54.21, which is running keith's SIP client
1
2
3
4
5
6
7
8
9
128.119.40.186
9. Data flows between clients
197.87.54.21

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

Outro

The big picture

- Computer networks are a stack of layers
    - Built that way for modularity
    - Each layer does one set of functions very well
    - Each layer depends on the layers beneath it
    - But modularity can sometimes result in inefficiency

- Many general and useful principles
    - Borrowed from real life (ex: listen before you speak)
    - Borrowed from systems in general (ex: use indirection for flexibility)
    - Applicability goes the other way as well (ex: how to meter freeway ramps)

CS 352 for the impatient

- A set of slides summarizing the most important things to take away from this course

- “Too long; didn’t read” versions of 352 at
- https://www.cs.rutgers.edu/~sn624/352-S20/lectures/tldr.pptx
- https://www.cs.rutgers.edu/~sn624/352-S20/lectures/tldr.pdf

You’ve gone through 24 lectures of 352…
now what?

A few possibilities

- The course is over. Go about life as usual
- Apply your new-found skills to solve a problem you care about
- Develop a deeper understanding of these technologies
- Consider improving the state of the art

(1) Go about life as usual

- This material will still be useful for a good “CS life”
  - Deeper understanding of the abstractions you use (ex: sockets. How big should socket buffers be?)
  - Why we need certain technologies (ex: video bit-rate adaptation)
  - A more nuanced understanding of real issues (ex: how are ISPs impacting net neutrality using QoS mechanisms?)
  - Enhanced abilities to troubleshoot your own tech problems (ex: why is website X not loading? Is it my Internet connection or the other end?)

(2) Solve a problem you care about

- Most concepts we discussed are supported by real, open-source, freely-available software
    - Many technology and protocol specifications are freely available (RFCs)

    - Linux kernel source code
    - Open source software routers
    - Open source browsers (Mozilla), mail clients (mutt), video clients
    - Most protocols are “open source”
    - Free or cheap infrastructure: EC2 servers, domain names, certificates

(2) Solve a problem you care about

- Improve video chat performance?
- Improve the usability of secure email?
- Improve web transfer performance?
- Make it easier to diagnose home wifi issues?
- <your idea here?>

(3) Deepen your understanding

- Fall 2020: CS 552 “Computer Networks”
  - A deeper take on the fundamentals of Internet design
  - https://www.cs.rutgers.edu/~sn624/552-F19/index.html

- Some questions we’ll talk about:
  - How does Google serve your web traffic so quickly?
  - How do large networks verify that their networks are functioning well?
  - How are high-speed routers built?
  - What transpires inside large data centers run by Amazon & Facebook?
  - How should you optimize your web-app to load faster on browsers?
  - … and more

(3) Deepen your understanding

- 552 requires
  - Paper readings
  - Deep understanding
  - Engaging in lively class discussions

- You will be assessed mainly through a software project
  - On a topic of your choice that *you* are excited about
  - The only requirement is that it must be connected to class material

- Every assessment is “take home”; there will be no exams.

(4) Push the state of the art

- Many of you will embark on CS-related careers
  - Can use your 352 know-how to do cutting-edge work in your org
- Some of you may consider graduate school
  - Networking is a great area to work in
  - Some of the most cited papers in CS, at least 2 Turing awards
  - 552 is a good place to lay a foundation for this path
- If you’re interested in working on small research projects, contact me.

Now it’s time for…

- Any questions, general or specific, about the course
- Any topics you’d like me to go over again
- Any feedback you’d like to voice about this course