Internet Telephony: More than just re-inventing the telephone

Henning Schulzrinne
Dept. of Computer Science
Columbia University
New York, New York
schulzrinne@cs.columbia.edu

University of Massachusetts at Amherst

March 29, 1999

(Joint work with Jonathan Lennox, Maria Papadopouli, Ping Pan, Jonathan Rosenberg, Elin Wedlund, and Jianqi Yin)

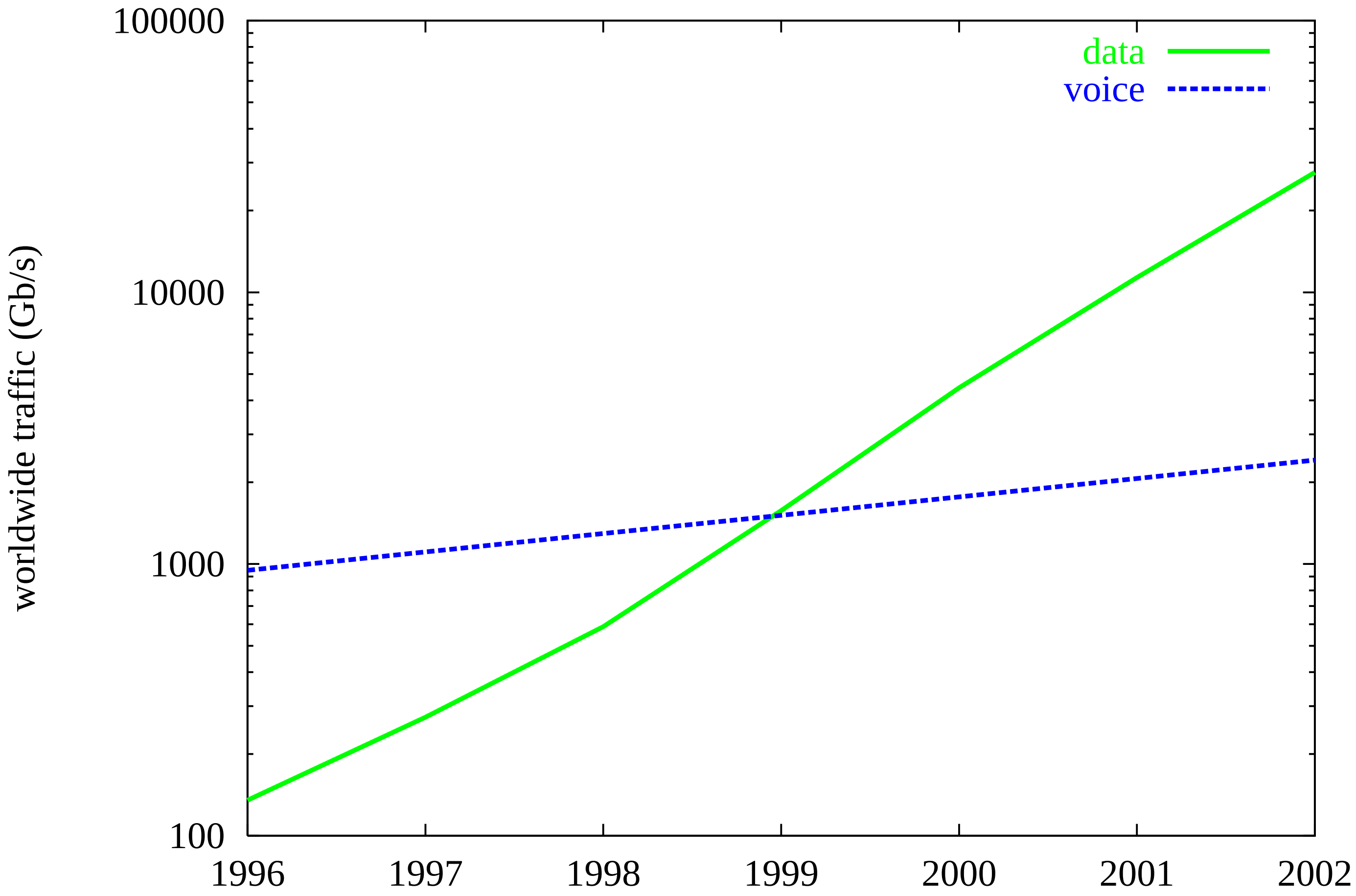
Overview

- Internet telephony: motivation and problems
- protocol architecture
- quality of service:
  - light-weight resource reservation
  - forward error control
- services ➠ signaling
- the “programmable” telephone
- Internet telephony “appliances”
- mobile services

Historical perspective

1876	invention of telephone
1915	first transcontinental telephone (NY–SF)
1920's	first automatic switches
1956	TAT-1 transatlantic cable (35 lines)
1962	digital transmission (T1)
1965	1ESS analog switch
1977	4ESS digital switch
1980s	Signaling System #7 (out-of-band)

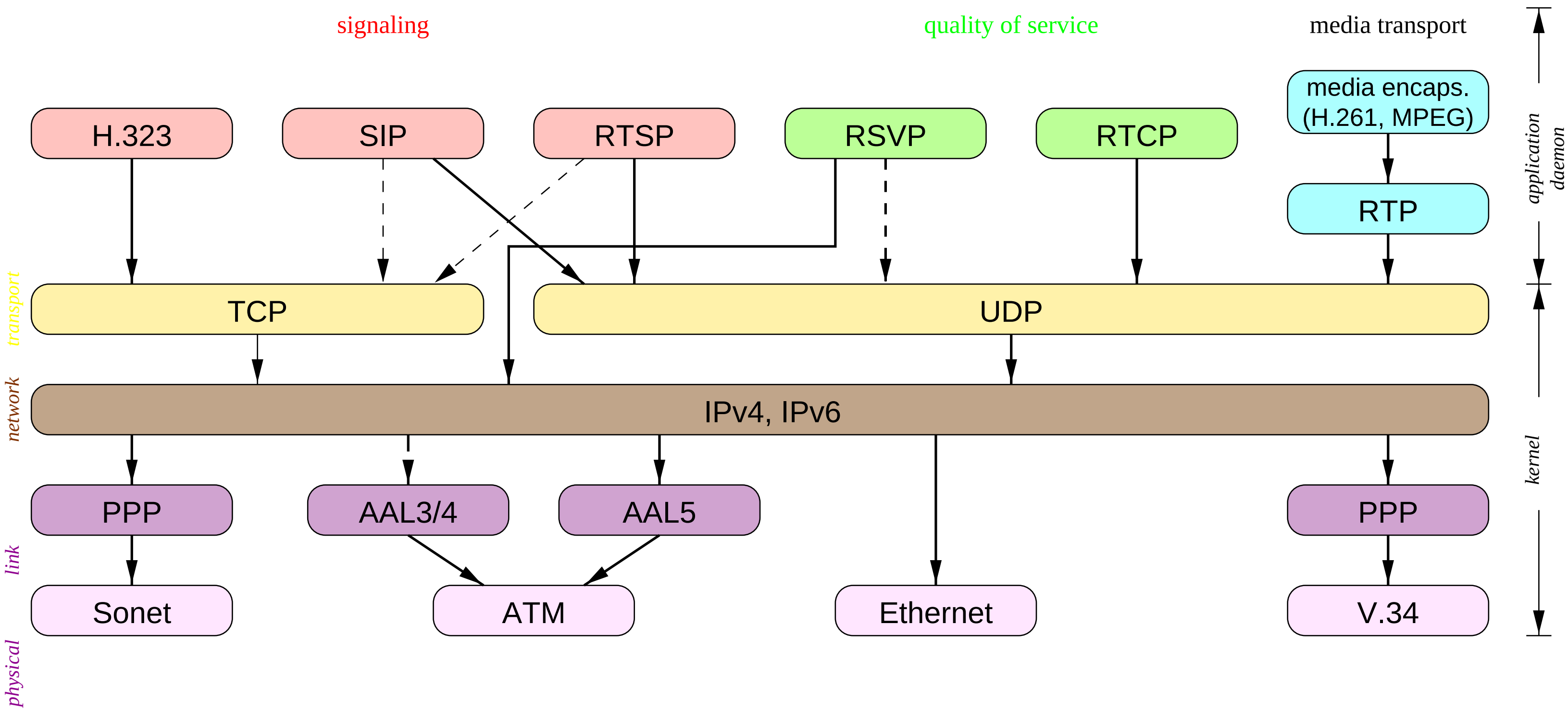
Data vs. Voice Traffic



The phone works — why bother with VoIP?

user perspective	carrier perspective
• variable compression: tin can to broadcast quality	• silence suppression ➠ traffic ↓
• security through encryption	• shared facilities ➠ management, redundancy
• caller, talker identification	• advanced services (email/web integration)
• better user interface (browser)	• cheaper switching ($0.005 vs. $5/kb/s)
• internat. calls: TAT transatlantic cable = $0.03/hr	• 9.6 kb/s fax as data
• no local access fees (3.4c)	
• easy: video, whiteboard, . . .	

Internet multimedia protocol stack

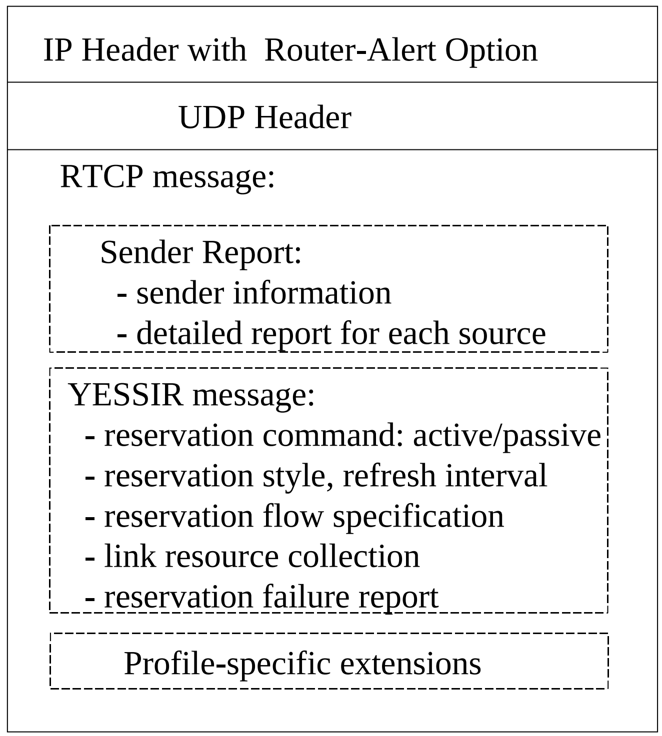


YESSIR: Yet another Sender Session Internet Reservation

- RSVP: separate daemon, API
- ➠ integrate into application that needs it (embedded systems!)
- in-band ➠ easier firewall
- RTP: common data transport protocol for audio/video
- router alert option in RTCP packets
- resource demands: payload type, measurement, flow specs, ...
- soft-state + RTCP BYE
- partial reservations: add links as session ages $\leftrightarrow$ fragmentation

YESSIR

plain RTCP SRs or additional information:



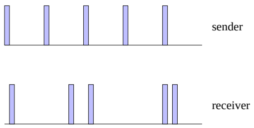
end-to-end refresh (vs. hop-by-hop)

RSVP and YESSIR performance

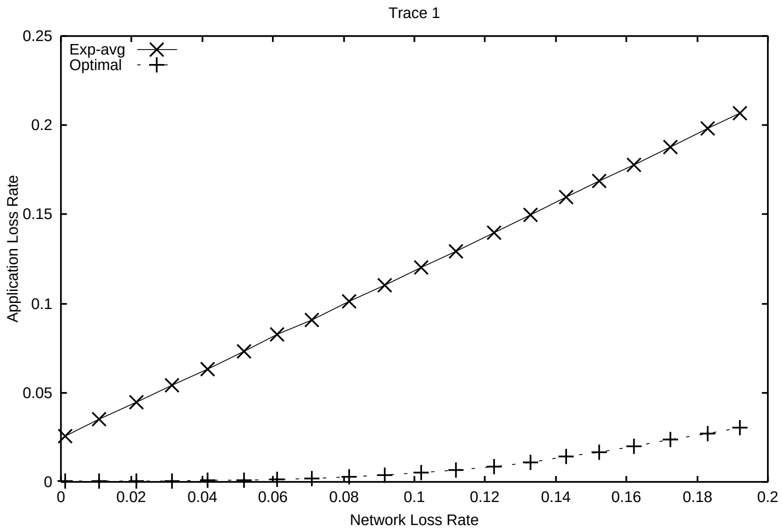
	setup	refresh
	μs	
RSVP	1,105	624
YESSIR	356	344

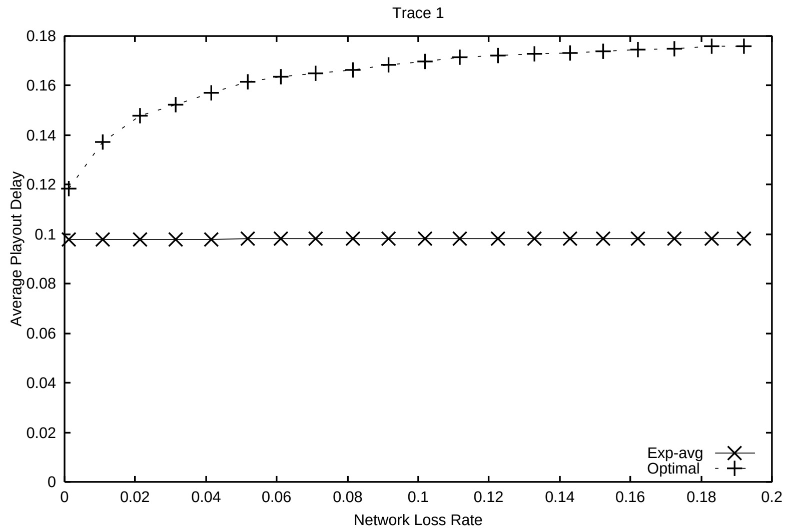
Integrating packet FEC into adaptive voice playout buffers

- playout buffer: trade loss (2. . . 20%) for delay (50. . . 500 ms)



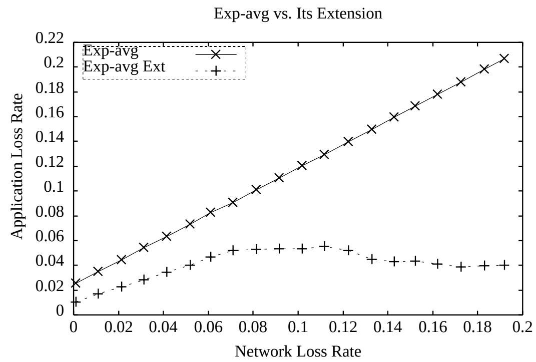
- (n, k) FEC: add $n - k$ additional packets for total of n

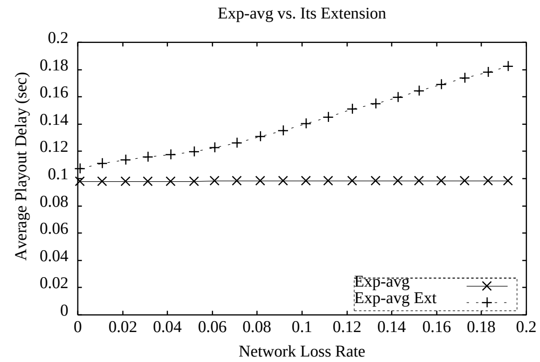


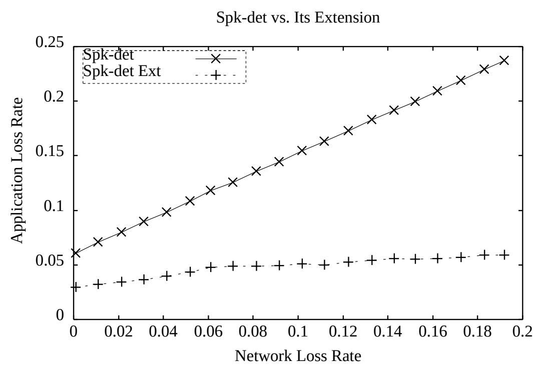


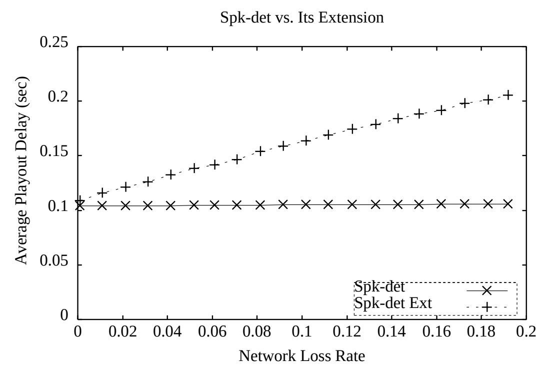
FEC: virtual delay

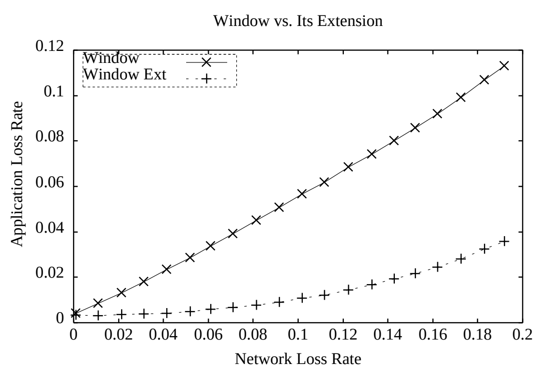
- virtual delay = min(arrival, recovery) $-$ departure
- playout delay $\approx \alpha \cdot \sigma$
- if loss $<$ target loss, $\alpha \leftarrow \alpha + \delta$
- recover lost and **late** packets
- 20% loss: application loss/5, delay * 2

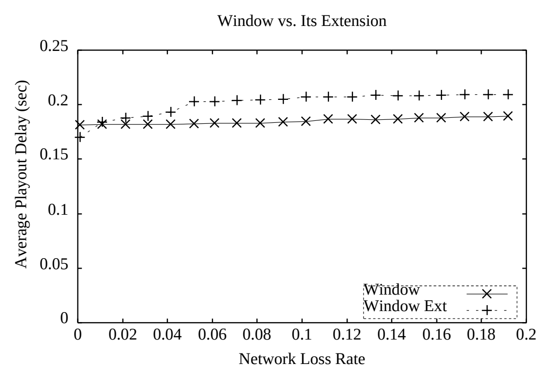

Exp-avg vs. Its Extension
Application Loss Rate
Network Loss Rate
Exp-avg
Exp-avg Ext


Exp-avg vs. Its Extension
Average Playout Delay (sec)
Network Loss Rate
Exp-avg
Exp-avg Ext


Spk-det vs. Its Extension
Application Loss Rate
Network Loss Rate
Spk-det
Spk-det Ext


Spk-det vs. Its Extension
Average Playout Delay (sec)
Network Loss Rate
Spk-det
Spk-det Ext


Window vs. Its Extension
Application Loss Rate
Network Loss Rate
Window
Window Ext


Window vs. Its Extension
Average Playout Delay (sec)
Network Loss Rate
Window
Window Ext

New playout algorithm: delayed optimal

- after talkspurt, one knows optimal delay
- use optimal combination for next talkspurt
- $D_i = \alpha D_{i-1} + (1 - \alpha) D_{\text{opt}}$
- user perception function: minimal delay that achieves loss target

New playout algorithm: Analytical

- assume independent loss p, delayed randomly d
- compute playout probability
- tabulate delay distribution histogram
- at end of talkspurt, find matching playout delay for target loss rate

SIP: Session Initiation Protocol

- call user
- re-negotiate call parameters
- manual and automatic forwarding
- call center: reach first (load distribution) or reach all (department conference)
- *personal mobility* (complements data link/IP mobility) ➠ change of terminal (PC, digital cordless, palmtop), location
- “forking” of calls
- terminate and transfer calls
- web security, cookies

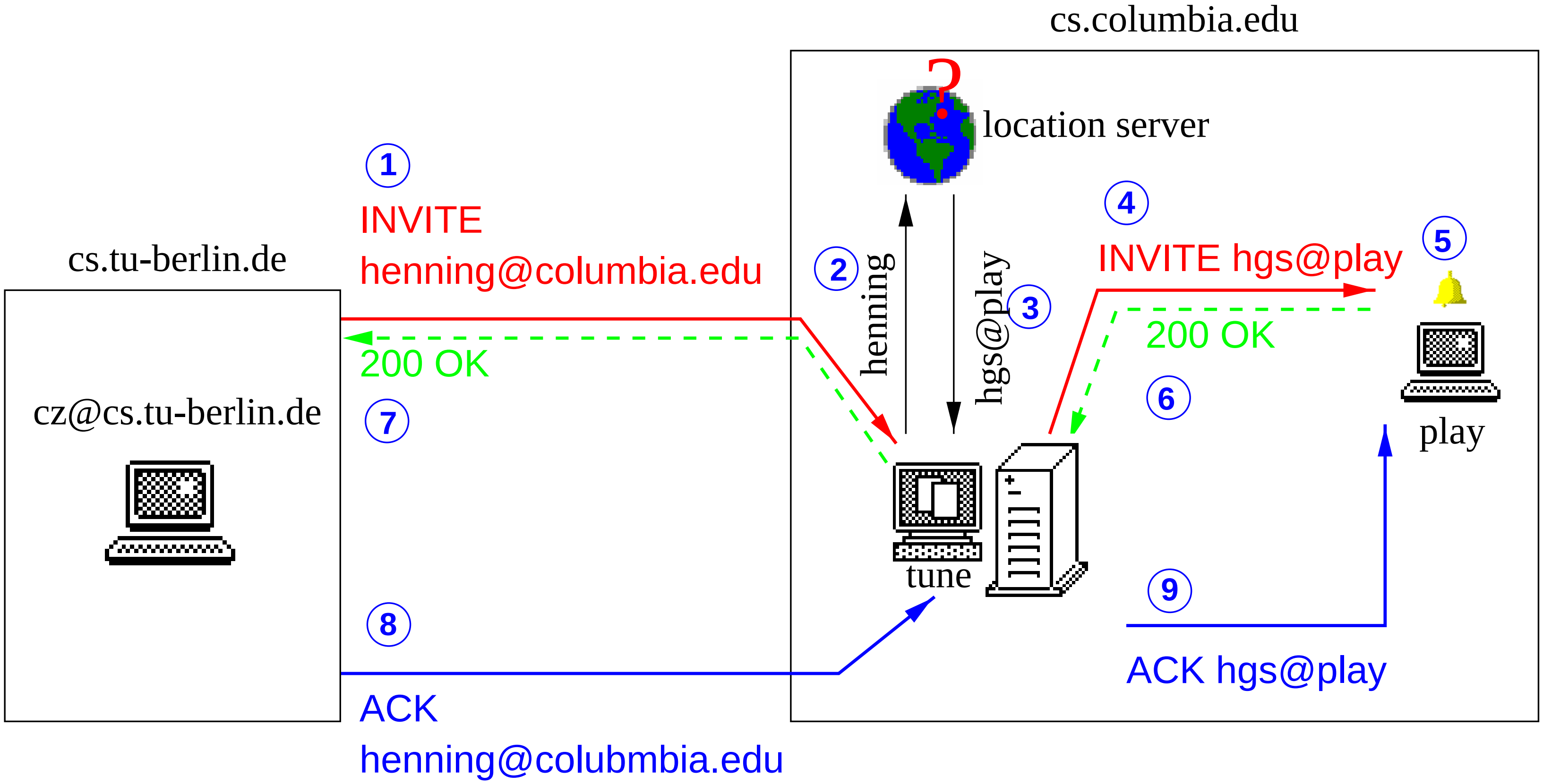
SIP addresses food chain

yellow pages	"president of the United States"	
	↓ *WWW search engines*	
common names	"Bill Clinton, Whitehouse"	
	↓ *directory services*	
host-independent	president@whitehouse.gov	
	↙ *SIP*	↘ *SIP*
host-specific	sip:bubba@oval.eop.gov	sip:+1-202-456-1111@net2ph.com
	↓ *DNS*	
IP address	198.137.241.30	

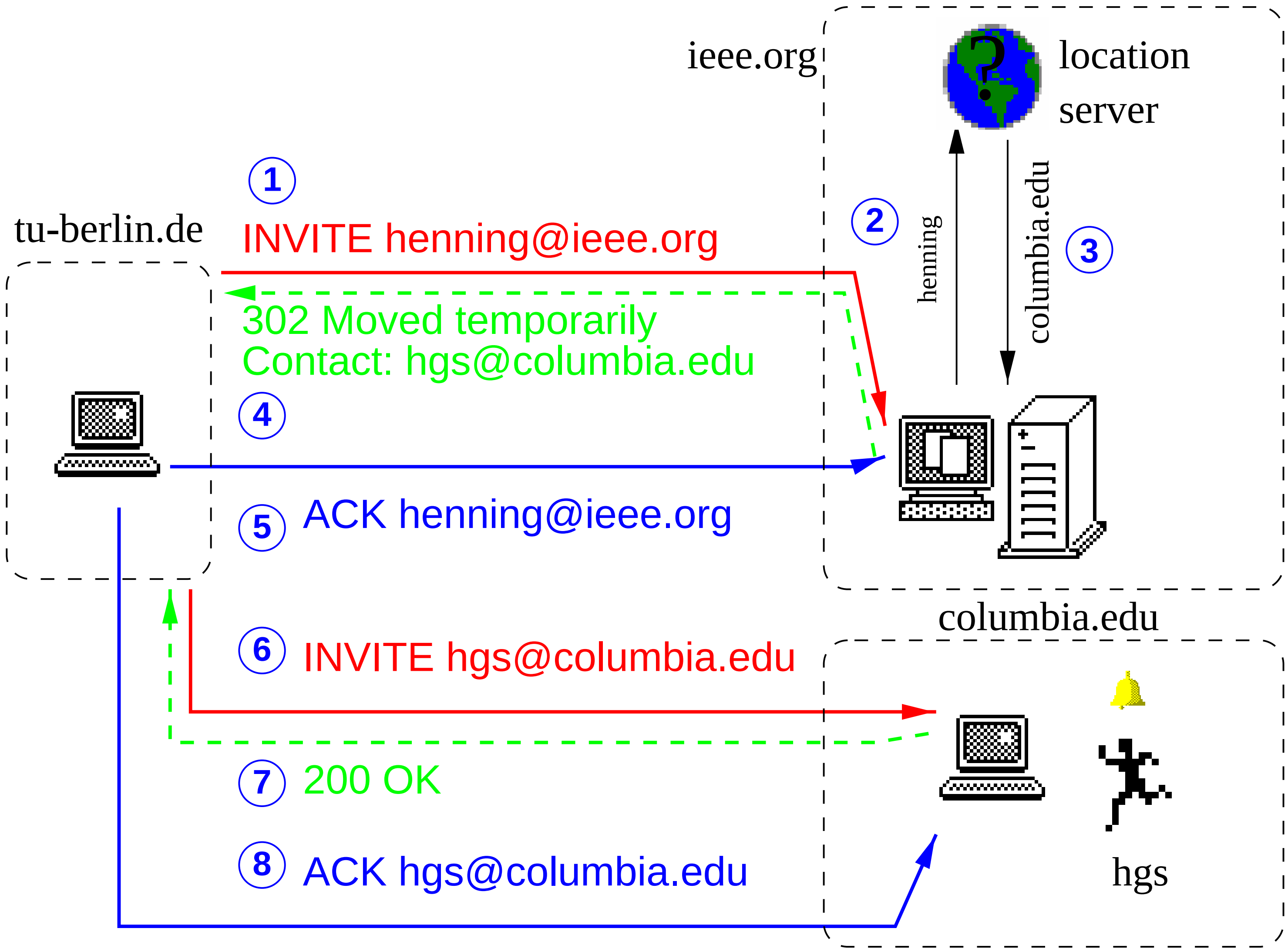
SIP: basic operation

1. use directory service (e.g., LDAP) to map name to *user@domain*
2. locate SIP servers using DNS SRV, CNAME, A
3. called server may map name to *user@host*
4. callee accepts, rejects, forward ($\rightarrow$ new address)
5. if new address, go to step 2
6. if accept, caller confirms
7. …conversation …
8. caller or callee sends BYE

SIP operation in proxy mode



SIP operation in redirect mode



SIP protocol design: robustness

SIP is designed to be robust against server failures:

- no state in proxy servers during call
- responses are “self-routing”
- subsequent requests and retransmissions can take different path (backup server)
- proxy servers can “lose memory” any time ➠ still function
- UDP ➠ less state than TCP, no time-wait

Invitation modes

invitation	conference	
	unicast	multicast
unicast	telephony	Internet TV session
multicast	reach first	dept. conference

SIP user location

- SIP is independent of mechanism to locate user
- examples:
  - local multicast of invitation
  - login-based via NFS
  - recursive “finger”-traversal
  - name translation: *Alexander.G.Bell* ➠ *agb*
  - active badges
  - SIP:
    - REGISTER announces location, with time limit
    - REGISTER + Location sets new location
    - forwarding within host ($\neq$ standard port)

Implementations

- 18 implementations at Columbia SIP “bake-off”
- Columbia sipd:
  - registration via unicast and multicast
  - handles mailing lists (ug-students@cs), ambiguous names (lee@cs)
  - maps names (b.clinton@whitehouse)
  - Apache (httpd)-style configuration and logging
  - “basic” authentication
  - how many servers for 2300 requests/second?

Signaling $\longleftarrow$ event notification

- call queueing ... buddy lists ... event notification
- SUBSCRIBE to events
- server NOTIFY
- can use forking
- handle subscriptions using CPL
- transition to multicast if large group of subscribers

Programmable phone service

- “caller proposes, callee disposes, administrator decides”
- web = static pages $\longrightarrow$ cgi-bin $\longrightarrow$ Java
- “if somebody is trying to call for the 3rd time, allow mobile”
- “try office and lab in parallel, if that fails, try home”
- “allow call to mobile if I’ve talked to person before”
- “if on telemarketing list, forward to dial-a-joke”
- phone: CTI = complex, not generally for end users
- “cgi-bin” for Internet telephones: generate requests, proxy, responses ➠ sip-cgi, complete control

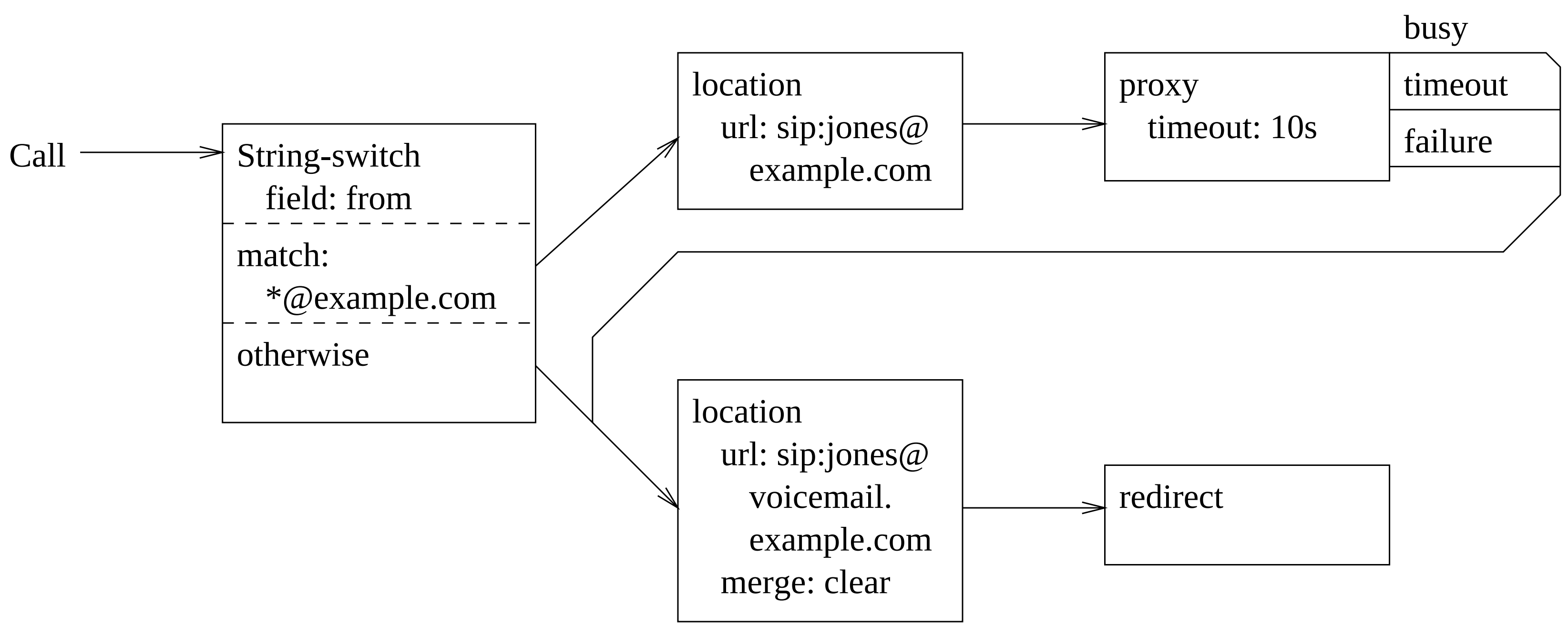
"Active Phone Networks"

language:

- don't want Turing-complete language
- fail safe: make phone calls even if crashes
- predictable resource consumption
- hide parallelism (searches)
- hide timers
- execute in callee's proxy server or end system (or phone button)

➠ CPL, an XML-based language

CPL example


Call
String-switch
field: from
match:
*@example.com
otherwise
location
url: sip:jones@
example.com
proxy
timeout: 10s
busy
timeout
failure
location
url: sip:jones@
voicemail.
example.com
merge: clear
redirect

CPL example

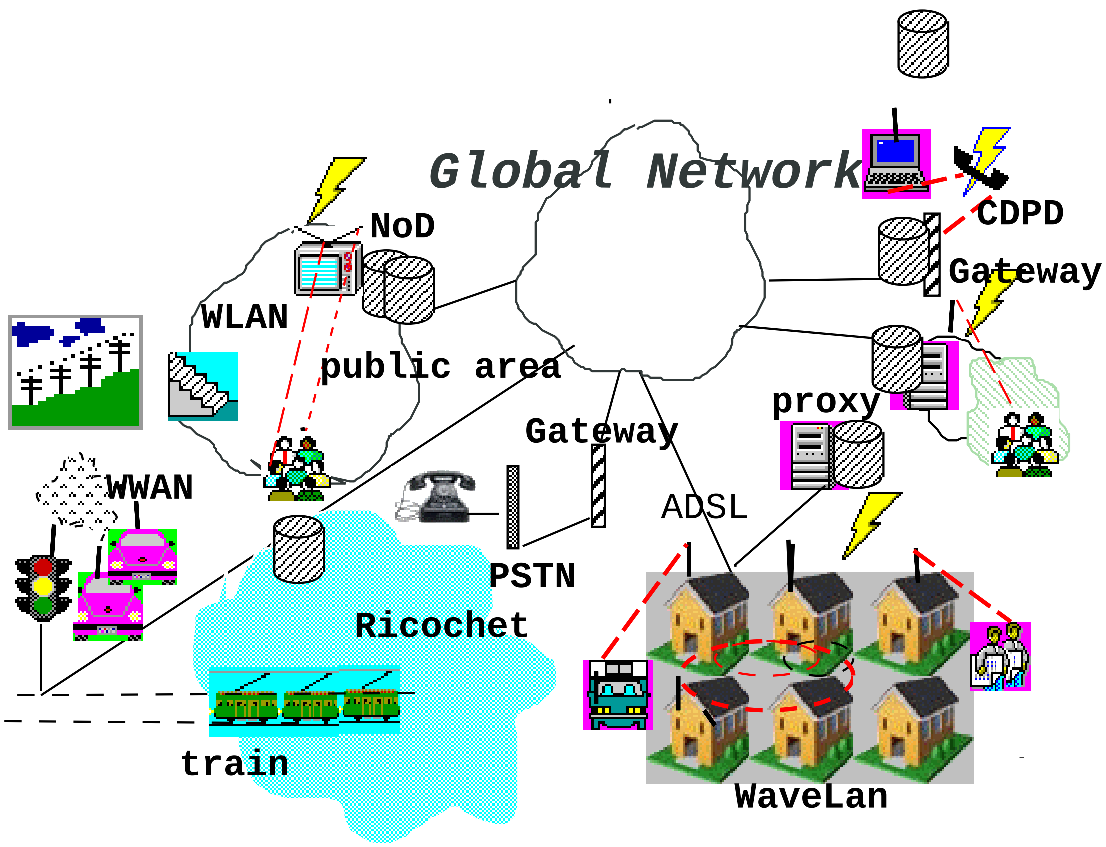
```
<call>
  <location url="sip:jones@jonespc.example.com">
     <proxy timeout="8s">
       <busy>
         <location url="sip:jones@voicemail.example.com" merge=
                   id="voicemail" >
            <proxy />
         </location>
       </busy>
       <noanswer>
         <link ref="voicemail" />
       </noanswer>
     </proxy>
  </location>
</call>
```

Internet phone “appliance”

- phone = \$49.99; PC > \$600 (GPF included)
- *Ethernet phone* ➠ no PBX for switching
- DSP for voice coding ➠ limited memory (128 kB SRAM!)
- implemented minimal IP stack (IP/UDP/RTP, DHCP, SIP)
- no TCP, no DNS
- MP3 radio
- interface to the world


128.59.19.38

Internet mobile services



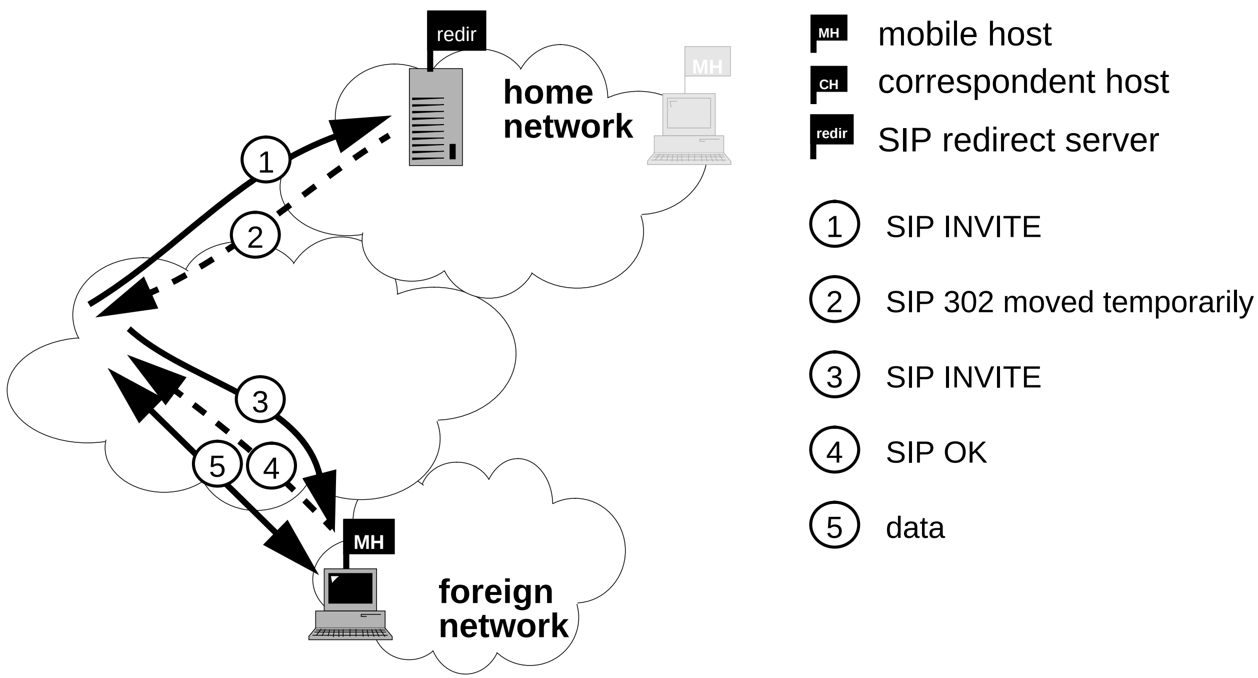
Internet mobile services: “social” ad-hoc networks

- connection sharing
  - multiple wireless networks: 2–10 Mb/s (WL Ethernet, IR) to 28 kb/s (Ricochet, CDPD)
  - share wide-area connections with neighbors
  - load sharing with mobile or in-home gateways
- social caching
  - subway model: in-car high-speed receiver updated in stations
  - socially optimal retrieval
  - anticipatory caching of streaming media
  - “leave the newspaper behind”

Internet cellular phone

- mobile IP: mask mobility to TCP connections
- imposes overhead:
  - all registrations to home agent
  - triangle routing (mostly)
  - encapsulation, address filtering problems
- ➠ use SIP and RTP for mobility management

SIP mobility



Other work: QOS

- RSVP refresh mechanisms
- aggregation of reservations (BGRP)
- quality-of-service characterization for IP telephony
- TCP-friendly adaptive services

Other work: signaling

- touch-tone transmission
- interoperation of SIP with SS7, ISDN and POTS
- large-scale IPtel gateways
- locating IPtel gateways (and other wide-area resources)
- charging for (adaptive) services and resources
- Internet voice mail

Other work: Internet radio & TV

- control of streaming services (➠ RTSP)
- “trunking” of RTP streams
- reliability and recovery of multicast routing protocols
- program insertion, “networks” for Internet radio and TV
- large-group scaling of RTCP feedback

Conclusion

- Internet telephony = first new service since web
- touches QOS, signaling, programming
- deployment inside out or outside in?
- operational issues: billing, 911, CALEA, ...

More information

Internet and telecom statistics:
http://www.cs.columbia.edu/˜hgs/internet

Papers: http://www.cs.columbia.edu/˜hgs/research/irt

RTP: http://www.cs.columbia.edu/˜hgs/rtp

RTSP: http://www.cs.columbia.edu/˜hgs/rtsp

SIP: http://www.cs.columbia.edu/˜hgs/sip