The Session Initiation Protocol (SIP)

Henning Schulzrinne
Columbia University, New York
schulzrinne@cs.columbia.edu
http://www.cs.columbia.edu/~hgs

Last modified September 28, 2000

Overview

- basic protocol operation
- design alternatives
- details: reliability, forking, ...
- services: mute, transfer, ...
- authentication and anonymity
- mobility
- comparison with H.323
- future directions

SIP Basics

SIP: Session Initiation Protocol

IETF-standardized *peer-to-peer* signaling protocol (RFC 2543):

- locate user given email-style address
- set up session
- (re)-negotiate session parameters
- manual and automatic forwarding ("name/number mapping")
- *personal mobility* ➠ different terminal, same identifier
- "forking" of calls
- terminate and transfer calls

SIP features

- provides call control (hold, forward, transfer, media changes, ...)
- leverages web infrastructure: security, “cgi-bin”, electronic payments, PICS, cookies, ...
- web-oriented: return HTML pages (“web IVR”)
- network-protocol independent: UDP, TCP, SCTP (or AAL5 or X.25)
- extends to presence information (“buddy lists”), instant messages and event notification

SIP servers and clients

UAC: user-agent client (caller application)

UAS: user-agent server ➠ accept, redirect, refuse call

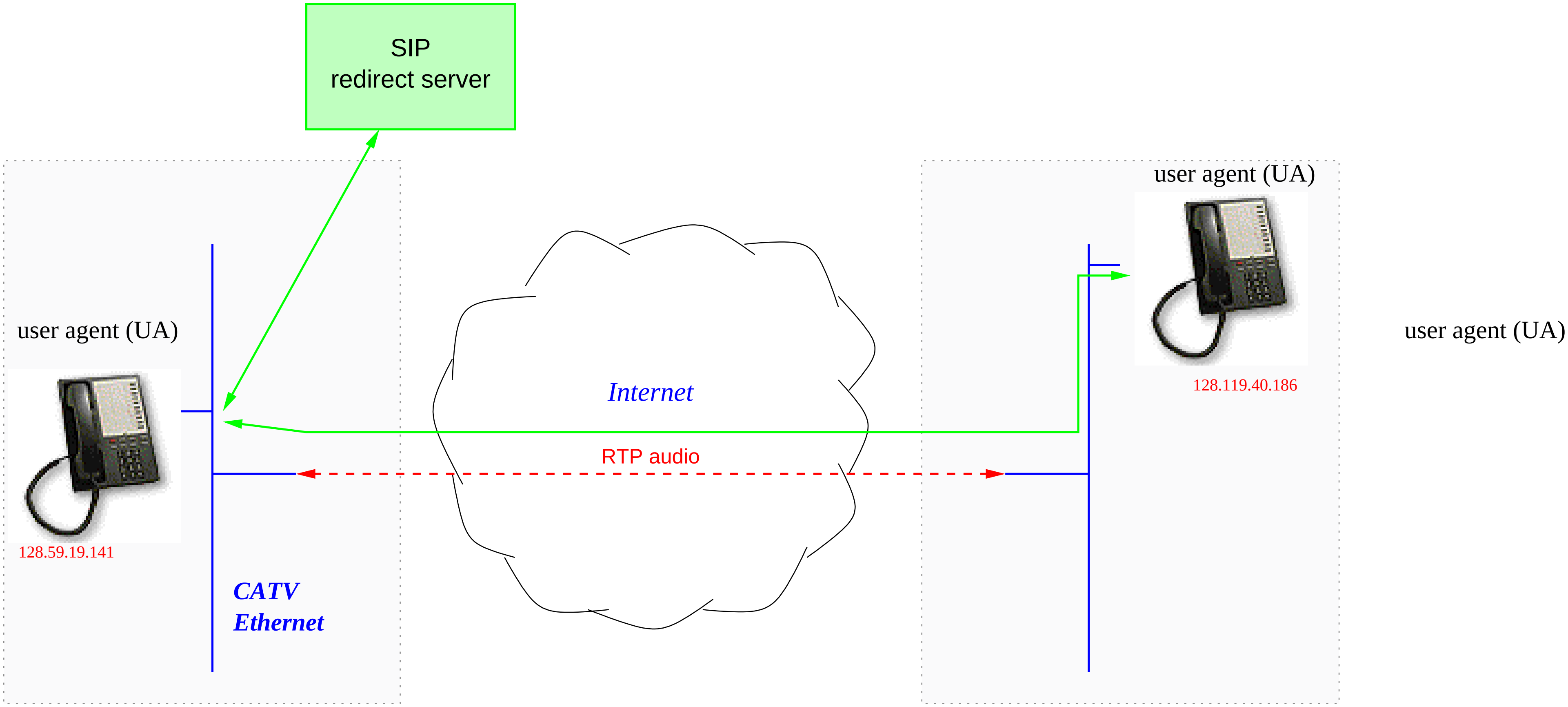
redirect server: redirect requests

proxy server: server + client

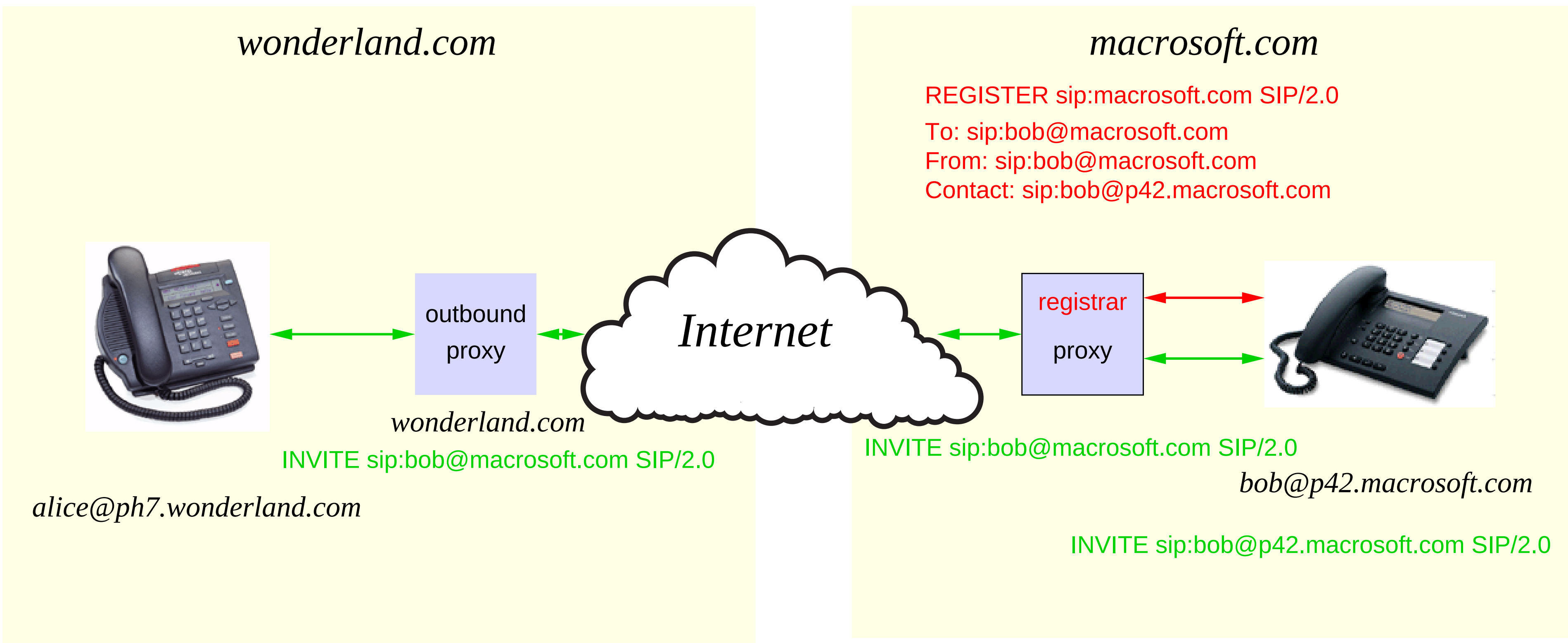
registrar: track user locations

- user agent = UAC + UAS
- often combine registrar + (proxy or redirect server)

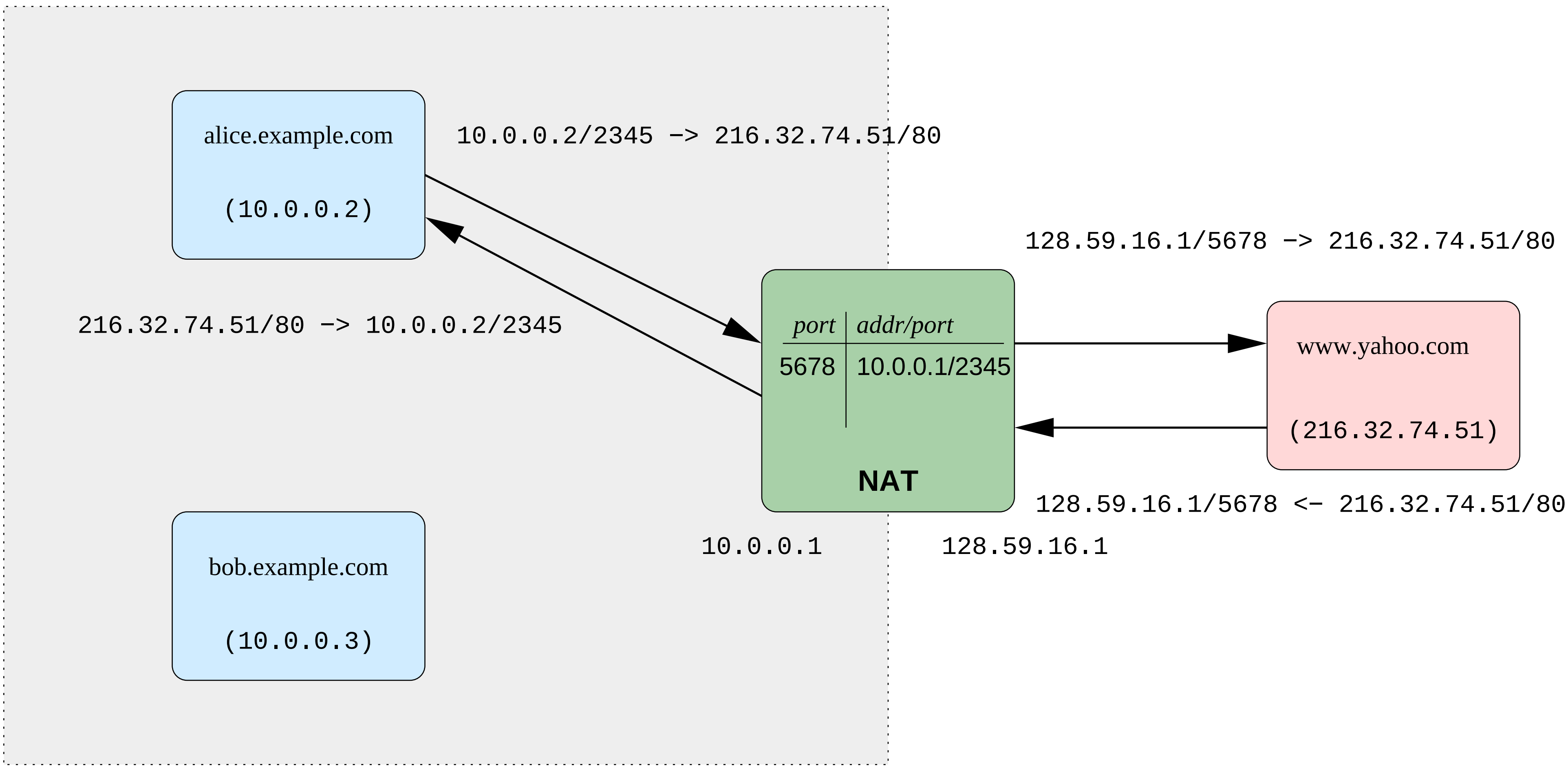
SIP architecture: peer-to-peer



SIP architecture: outbound proxy

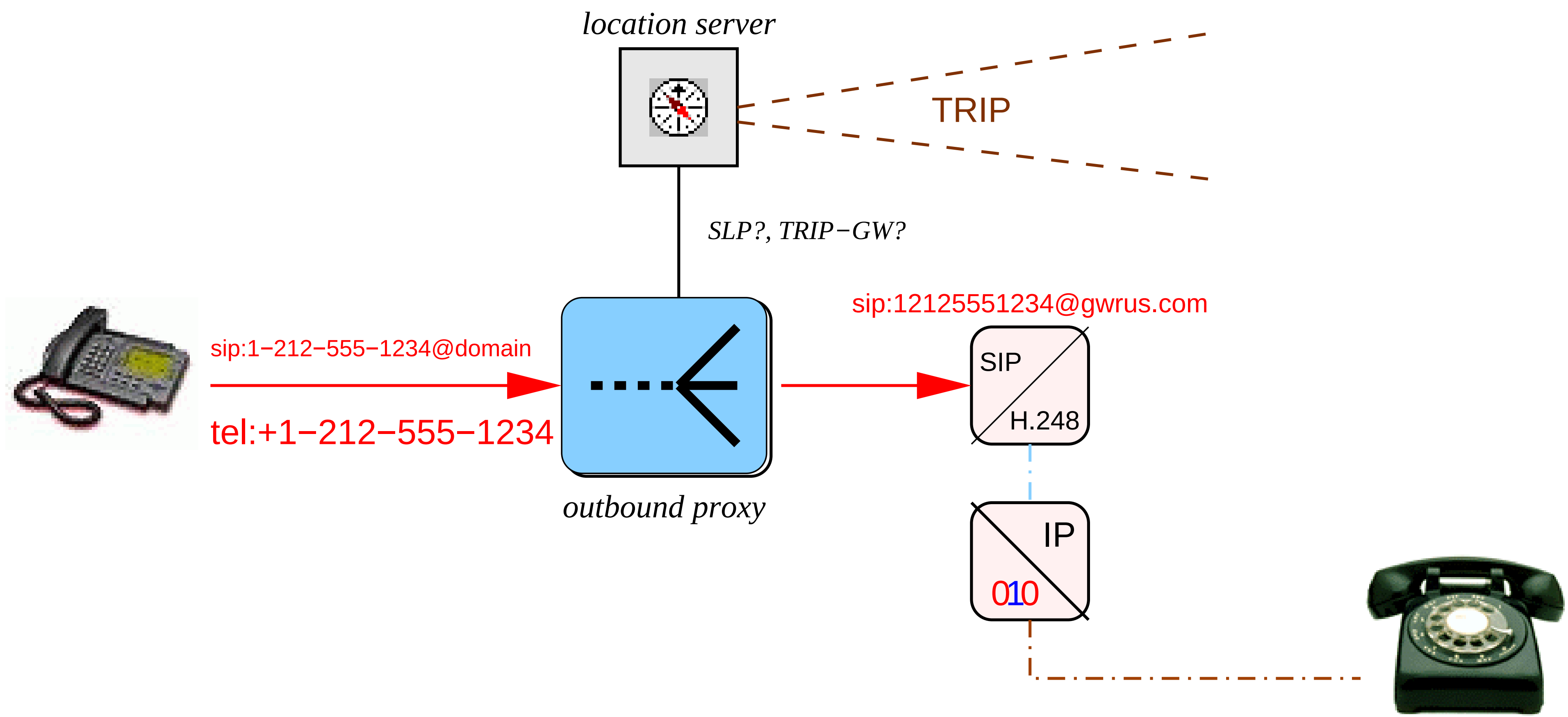


SIP architecture: carrier



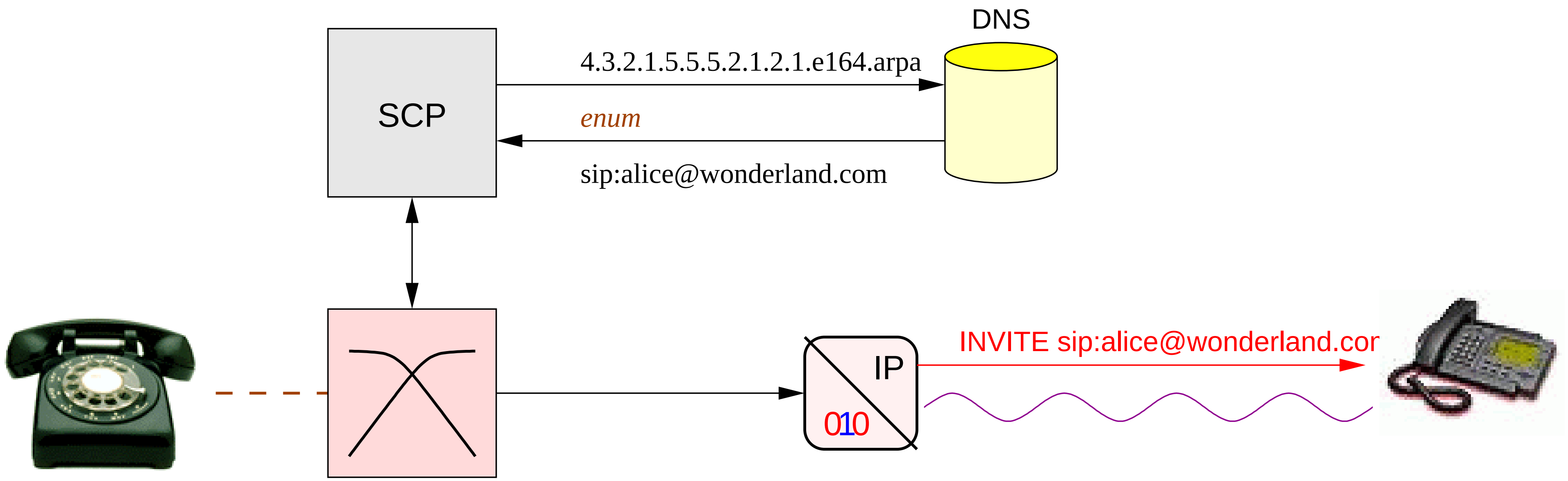
(Firewall is controlled by SIP proxy and enforces its policy.)

SIP architecture: VoIP to PSTN



SIP architecture: PSTN to VoIP

enum database



SIP: basic operation

1. use directory service (e.g., LDAP) to map name to *user@domain*
2. locate SIP servers using DNS SRV, CNAME or A RR
3. called server may map name to *user@host* using aliases, LDAP, canonicalization program, . . .
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

Also, tel:, h323: URLs $\longrightarrow$ outbound proxy maps to gateway

SIP–DNS interaction

extended email-like domain resolution ➠ try until success:

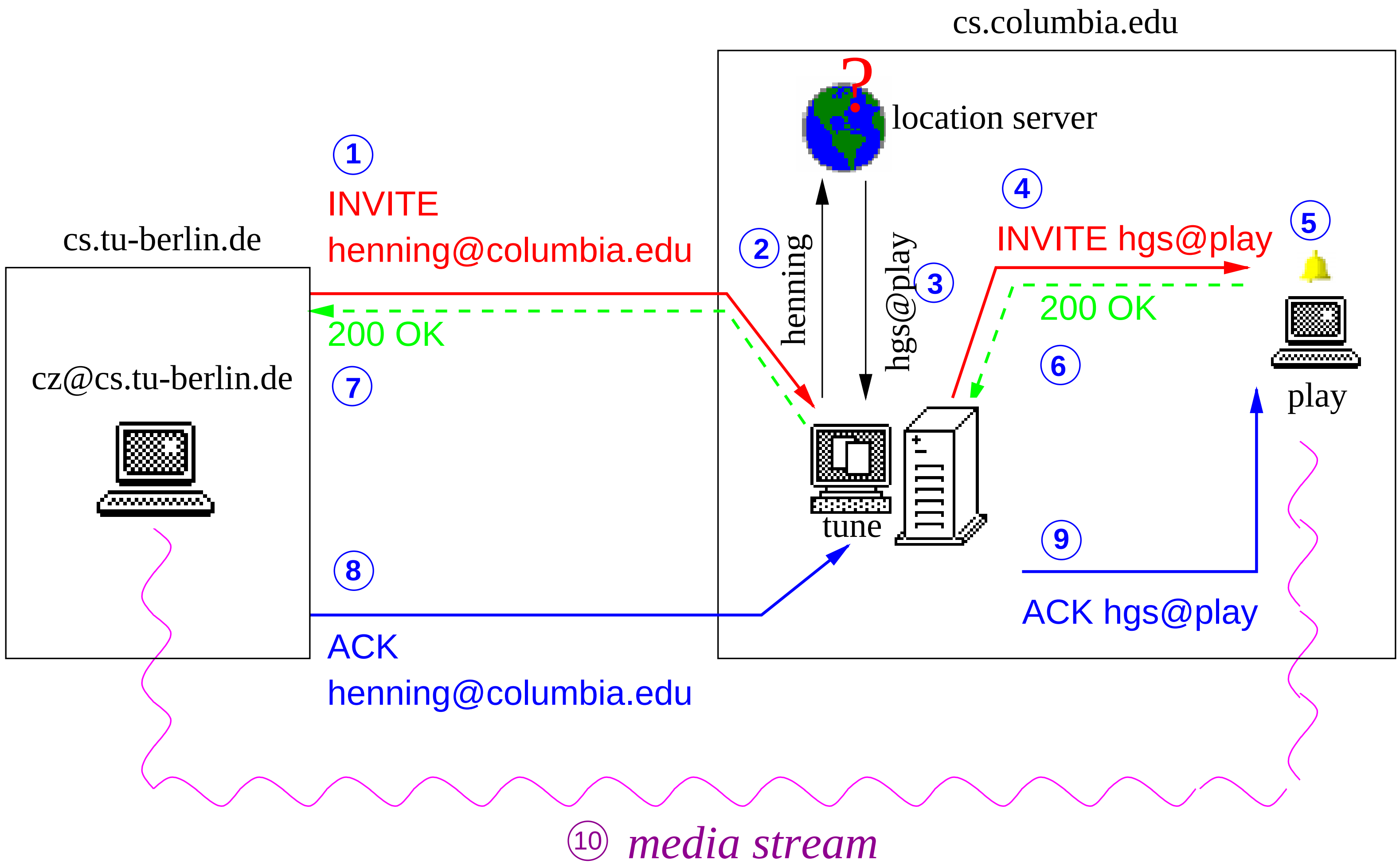
1. try SRV DNS record for "_sip._udp" and "_sip._tcp" in domain, with priority and weights for randomized load balancing

2. DNS CNAME or A record

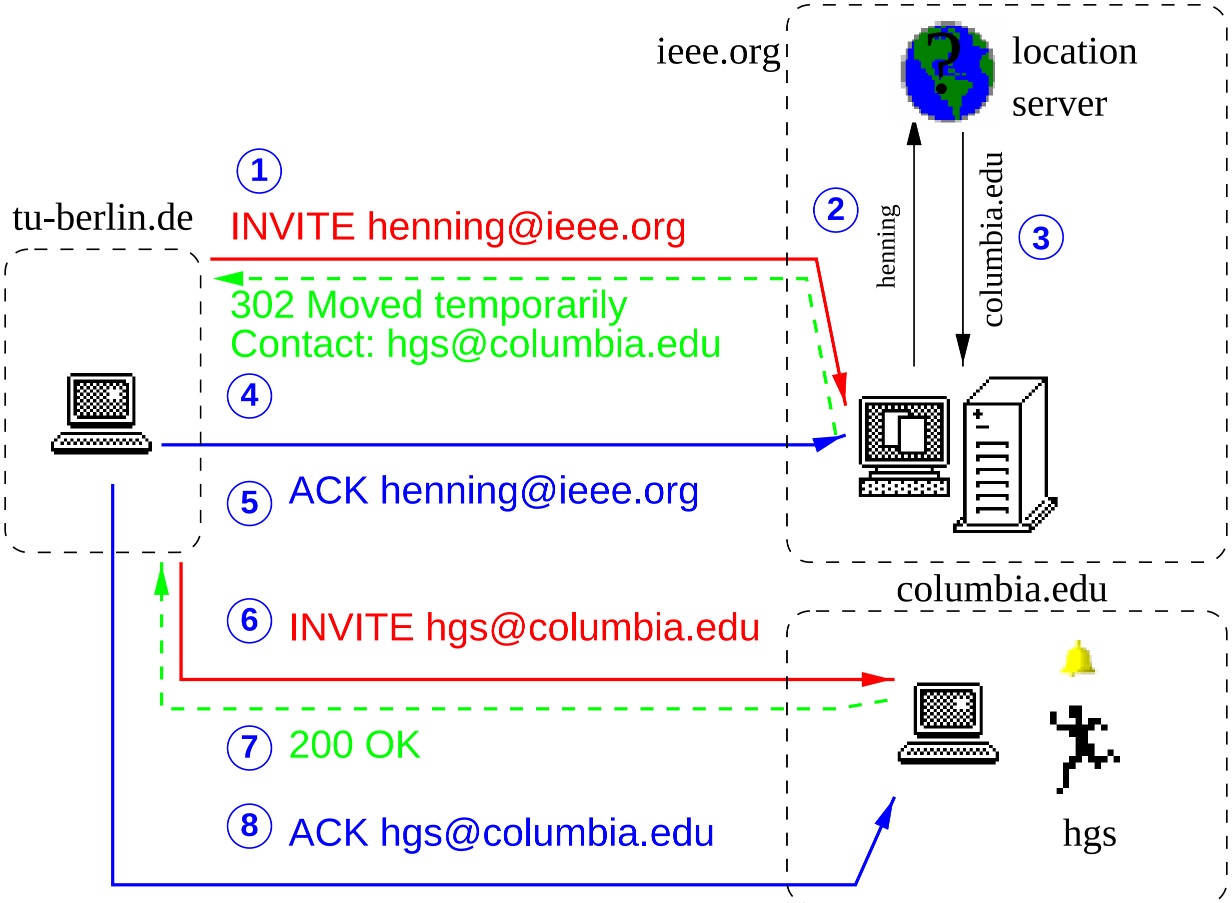
3. may try SMTP EXPN command to get new address; goto (1)

4. if all else fails, send SIP request via MIME

SIP operation in proxy mode

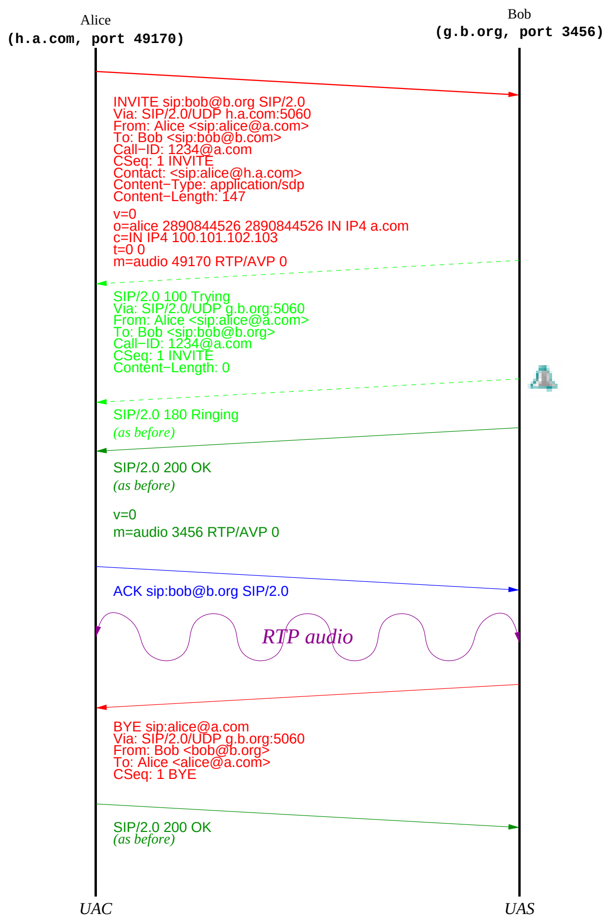


SIP operation in redirect mode

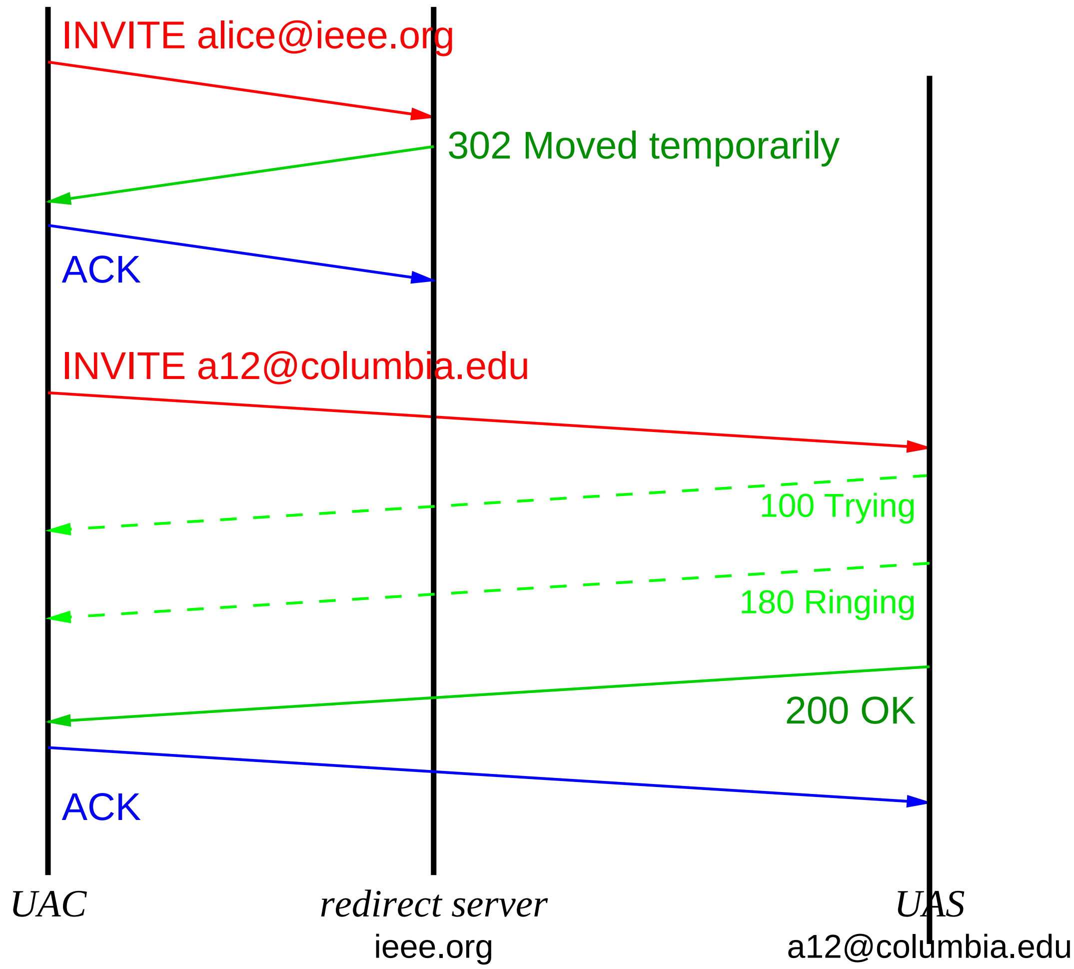


(302: redirection for single call; 301 permanently)

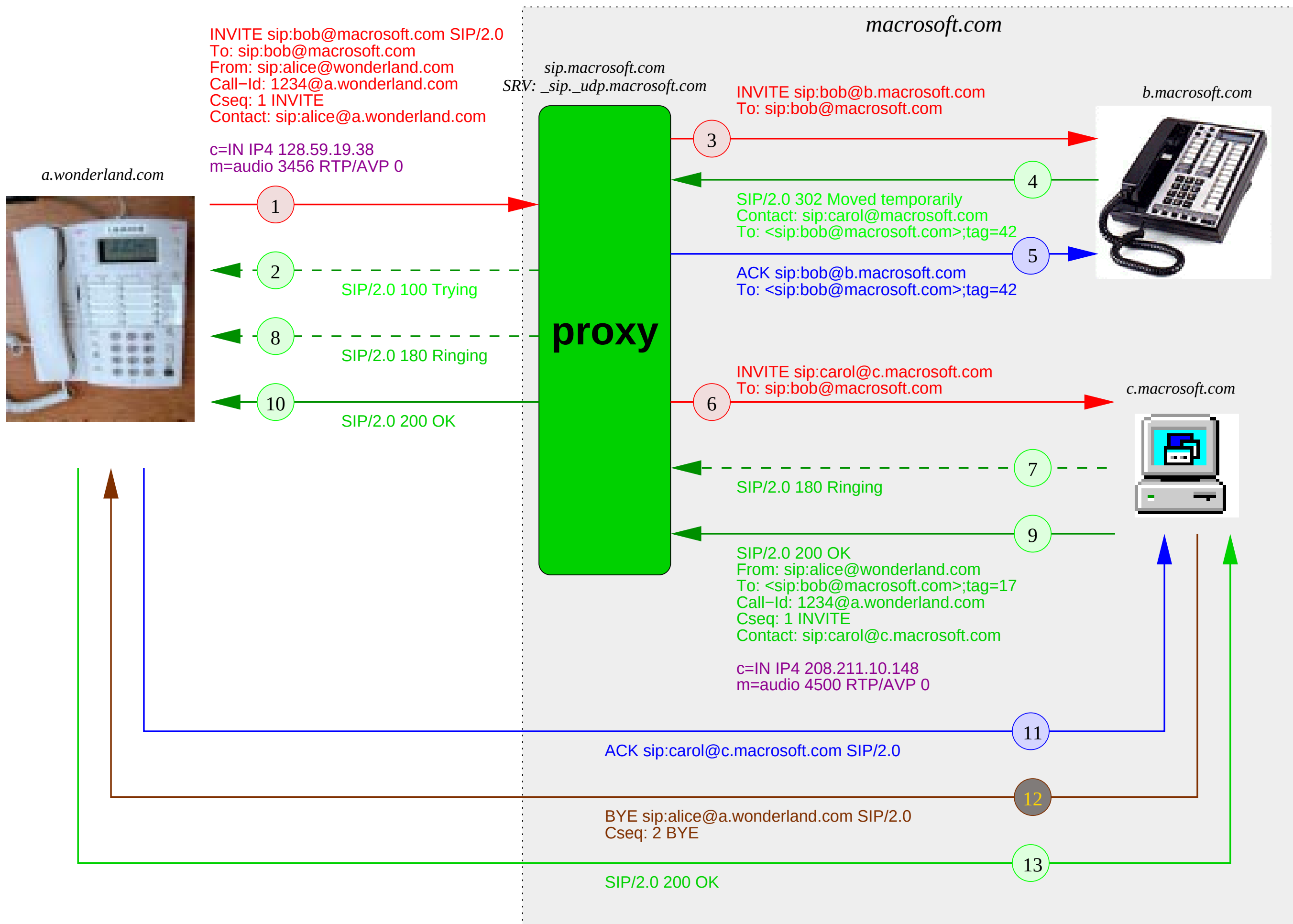
Basic SIP call



SIP operation in redirect mode



SIP – more detail



Invitation modes

signaling	media	
	unicast	multicast
unicast	telephony	multicast session
multicast	reach first	dept. conference

➠ SIP for all modes, SAP also for multicast/multicast

Proxy and redirect servers

proxy: may *fork* requests ➠ parallel or sequential search

- stateless: forward request or response
- transaction stateful: remember full request/response ⟶ needed for forking
- call stateful
- *outbound (near-end) proxy:* outgoing calls ➠ address lookup, policy, firewalls
- *(far-end) proxy:* closer to callee ➠ callee firewall, call path hiding

redirect server: lower state overhead, more messages

SIP requests and responses

- HTTP look-alike
- provisional and final responses:
  - 1xx = searching, ringing, queueing, . . .
  - 2xx = success
  - 3xx = forwarding
  - 4xx = client mistakes
  - 5xx = server failures
  - 6xx = busy, refuse, not available anywhere

SIP protocol request

```
INVITE sip:schulzrinne@cs.columbia.edu SIP/2.0
From: Christian Zahl <sip:cz@cs.tu-berlin.de>
To: Henning Schulzrinne <sip:schulzrinne@cs.columbia.edu>
Via: SIP/2.0/UDP 131.215.131.131, SIP/2.0 foo.com
Call-ID: 3678134014@cloud9.cs.tu-berlin.de
Content-Type: application/sdp
Content-Length: 187
CSeq: 8348 INVITE
Subject: New error codes

v=0
c=IN IP4 128.59.16.191
m=audio 1848 RTP/AVP 0
```

SIP requests

- *call leg:* From, To, Call-ID
- requests from callee to caller reverse To and From
- caller and callee keep their own CSeq space
- either side can send more INVITEs or BYE

SIP URLs

sip:[*user:pw@*]*host*:[*port*]
;transport=UDP;maddr=224.2.0.1

- used in Request-URI, Contact headers (redirect, registration), web pages
- transport and maddr specify transport
- can specify methods, header and body in web pages, email
- example: `sip:a.g.bell@belltel.com`

SIP Protocol Design

SIP protocol design

SIP and RTSP are not HTTP ➠

support UDP: no data stream, low latency desired

multicast: group signaling, user location

avoid HTTP mistakes: e.g.,

- relative request paths ⟶ always absolute (virtual hosts)
- no extension mechanism ⟶ Require, Supported
- 8859.1 coding ⟶ Unicode (ISO 10646)

SIP protocol design: robustness

SIP is designed to be robust against server failures:

- no state in proxy servers during call (cf. H.323 GK)
- 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

SIP and RTSP protocol design: encoding

- “Internet binary”
- ASN.1
- textual
- Jini/RMI, Corba, DCOM

Protocol design: internet binary

IP, TCP, RTP, RSVP, Q.931, . . . ➠

- fixed fields and/or type-length-value (TLV)
- efficient if aligned
- fewer ambiguities
- nesting, options tedious
- simple applications are hard
- not self-describing

Protocol design: ASN.1

SNMP (BER), H.323/H.245 (PER) ➠

- not self-describing ➠ need external description
- BER: inefficient, lots of options
- PER: external description needed even for data types
- internationalization not clear

Protocol design: textual

SMTP (RFC 822), HTTP, SIP, RTSP:

- random textual: ftp, POP, IMAP, gopher, ... ➠ new parser for each protocol
- SMTP, HTTP, SIP, RTSP
  - $C \rightarrow S$: method, object, *attribute: value;parameter*, [body]
  - $S \rightarrow C$: status code, message, [body]
    - binary data not important
    - extensions: PEP, JEPI, PICS, ...
    - easy to parse & generate for Tcl, Perl, Python, ...
    - overhead (space, time)? unidirectional?
    - but $\neq$ HTTP: not object retrieval, state (RTSP), ...

RPC: RMI, Corba, DCOM

RMI, Corba, DCOM: ➠ *potentially* replace *all* upper-layer Internet protocols

- cost of entry?
- maturity (security, extensions, multicast, . . .)
- performance?
- tools (binary)?
- scalable to global name/object space?

Summary: SIP and Corba

	SIP	Corba
data	optional fields	versioning hard
	two-level hierarchy	general, C-like
hiding	dynamic	directory-based
multiple	forking proxy	no
transport	UDP, TCP, …	TCP
strength	inter-domain	intra-domain
generality	session set-up	RPC, events, …

SIP Details

SIP syntax

request | response

method URL **SIP/2.0** | **SIP/2.0** *status reason*

Via: **SIP/2.0/** *protocol host:port*
From: *user <sip:from_user@source>*
To: *user <sip:to_user@destination>*
Call−ID: *localid@host*
CSeq: *seq# method*
Content−Length: *length of body*
Content−Type: *media type of body*
Header: *parameter ;par1=value ;par2="value"*
;par3="value folded into next line"

message header

blank line

V=0
o= *origin_user timestamp timestamp* **IN IP4** *host*
c=IN IP4 *media destination address*
t=0 0
m= *media type port* RTP/AVP *payload types*

message body

message

SIP syntax

- field names and some tokens (e.g., media type) are case-insensitive
- everything else is case-sensitive
- white space doesn't matter except in first line
- lines can be folded
- multi-valued header fields can be combined as a comma-list

SIP methods

INVITE	initiate call
ACK	confirm final response
BYE	terminate (and transfer) call
CANCEL	cancel searches and “ringing”
OPTIONS	features support by other side
REGISTER	register with location service
INFO	mid-call information (ISUP, DTMF)
COMET	precondition met
PRACK	provisional acknowledgement
SUBSCRIBE	subscribe to event
NOTIFY	notify subscribers

SIP response codes

1xx		*provisional*
	100	continue
	180	ringing
2xx		*success*
	200	OK
3xx		*redirect*
	300	multiple choices
	301	moved permanently
	302	moved temporarily

SIP response codes

4xx	*client error*	
	400	bad request
	401	unauthorized
	403	forbidden
	404	not found
	407	proxy auth. required
	408	request timeout
	420	bad extension

480	temporarily unavailable
481	call leg doesn’t exist
482	loop detected
483	too many hops
484	address incomplete
485	ambiguous
486	busy here
487	request cancelled
488	not acceptable

SIP response codes

5xx		*server error*			
	500	server internal error	6xx		*global failure*
	501	not implemented		600	busy
	502	bad gateway		601	decline
	503	service unavailable		604	does not exist
	504	gateway time-out		606	not acceptable
	505	version not supported			

Headers: call and request identification

Call-ID: globally (time, space) unique call identifier

To: *logical* call destination

From: call source

CSeq: request within call leg

call leg = Call-ID + To + From

Tagging To

- after forking and merging, hard to tell who responded
- UAS responds with random tag added to disambiguate

```
To: "A. G. Bell" <sip:agb@bell-telephone.com>
  ;tag=a48s
```

- future requests are ignored if they contain the wrong tag

SIP request routing

- send requests to local proxy or host in Request-URI
- each proxy checks for loop, prepends a Via header with own address

```
Via: SIP/2.0/UDP erlang.bell-telephone.com:5060
```

- UAS copies Via headers to response
- on receipt, make sure it's own address
- branch indicates proxy fork
- received set by receiver ➠ NATs
- maddr if received via multicast

SIP response routing

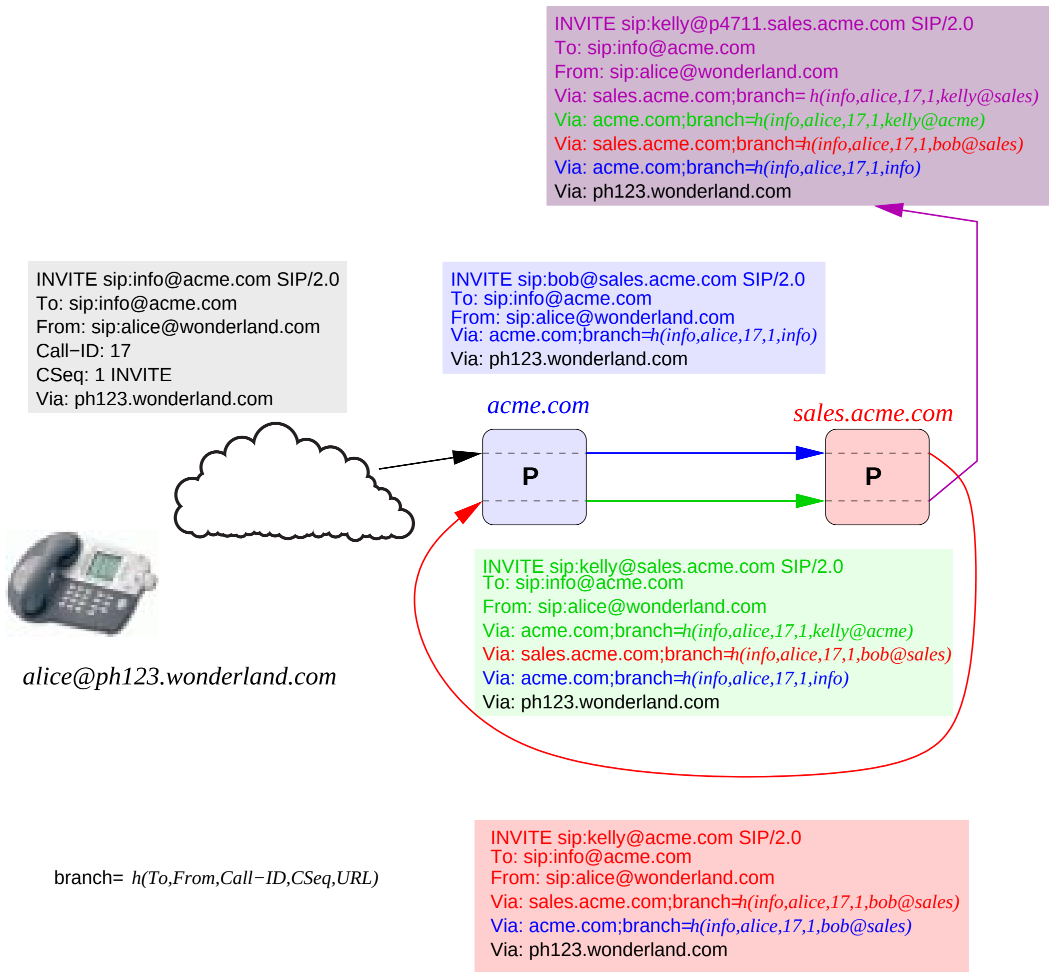
- response traces back request route *without proxy server state*
- forward to host, port in next Via
- TCP: re-use connection if possible, create new one if needed
- UDP: may send responses to same port as requests

```
Via: SIP/2.0/UDP server.domain.org:5060
  ;received=128.1.2.3
```

Loop and misdirection prevention

- Via header before forwarding
- “spirals”: revisit same server, with different request URI
- Max-Forwards limits number of hops
- Expires limits search time

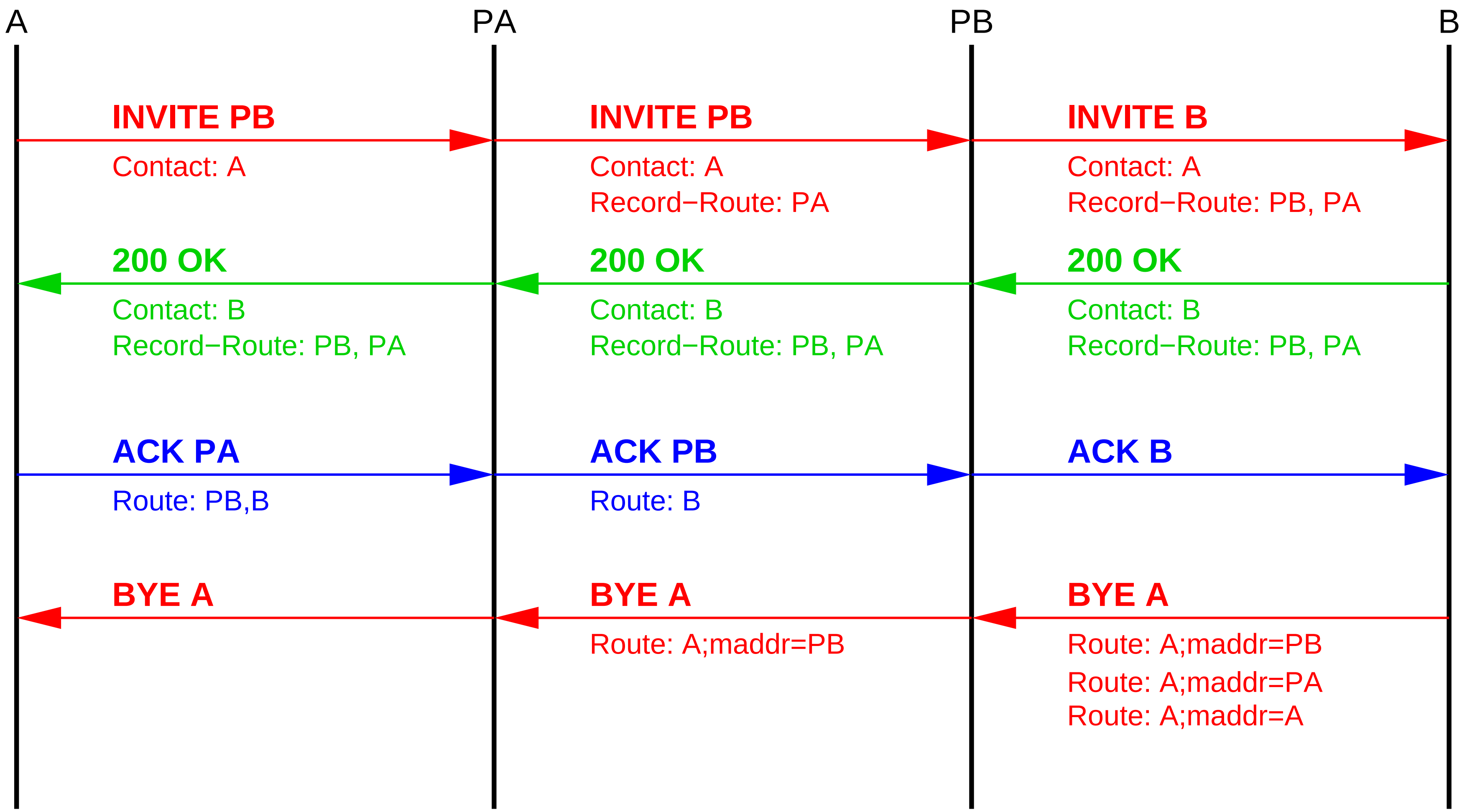
Spirals: revisiting proxy servers



Forcing request paths

- usually, bypass proxies on subsequent requests
- some proxies want to stay in the path $\rightarrow$ call-stateful:
  - firewalls
  - anonymizer proxies
  - proxies controlling PSTN gateways
- use Record-Route and Route

Record-Route and Route



Forcing request paths

- proxies that want to be in path add themselves as first Record-Route

```
Record-Route:
 Alice <sip:alice@wonderland.com;maddr=216.112.6.38>,
 <sip:alice@gw.wonderland.com;maddr=216.112.6.39>
```

- maddr identifies exact host (SRV!)
- UAS copies Record-Route into final response
- UAC copies Record-Route into Route, reversing order
- UAC adds Contact as last item
- each sender removes topmost and places it in request URL

Forcing request paths – reverse direction

- request from called party also traverse same proxies
- but can't just use Record-Route values
- use From in Route header and copy maddr from Record-Route

Call and caller identification

Subject	topic of call, short message
Organization	caller and callee, possibly filled in by proxy
Date	date of call (replay prevention)
Server	make and model of server
User-Agent	make and model of client
Accept-Language	human languages preferred
Priority	call priority (normal, urgent, . . .)
In-Reply-To	reference to earlier call-id

Content description

Describes message body:

Content-Disposition	display? session? script?
Content-Encoding	compression (gzip)
Content-Language	English, German, . . . – alternatives!
Content-Length	bytes in body
Content-Type	MIME type (e.g., application/sdp)

negotiated by Accept-*.

SIP message size

- standard headers have one-letter compact forms
- minimal request/response, with email address, host ≈ 20 bytes:

component	full	compact
headers, CRLF	71	35
body (SDP)	120	120
addresses (4)	96	96
other	72	72
sum	359	323

SIP message size

- $\sum$ INVITE, 100, 200, ACK, BYE, 200 $\approx$ 1500 bytes
- $\equiv$ 1.5 s of 8 kb/s voice
- gzip compression improves by about 25%

SIP extensions: new methods

- methods can be added at any time without changing the protocol
- server complains with 405 if not implemented, returns list of methods in Allow header

SIP headers

- receiver ignores headers, parameters it doesn't understand
- headers are not negotiated, but *features* are
- features: behavior, *maybe* headers, parameters, . . .

SIP extensions and feature negotiation

- if crucial, mark with "Require: *feature*"
- IANA-registered features are simple names, private features use reverse domain names
- indicate features supported in Supported:

```
C->S:   INVITE sip:watson@bell-telephone.com SIP/2.0
        Require: com.example.billing
        Supported: 100rel
        Payment: sheep_skins, conch_shells

S->C:   SIP/2.0 420 Bad Extension
        Unsupported: com.example.billing

S->C:   SIP/2.0 421 Extension Required
        Require: 183
```

Inquiring about capabilities

OPTIONS request returns:

Allow	methods
Accept	media types
Accept-Encoding	compression methods
Accept-Language	human languages
Supported	supported features

SIP reliability: all but INVITE

- SIP: UDP and TCP, same messages, same behavior
- requests contain

  Call-ID: globally unique in time and space

  CSeq: command sequence number ➠ duplicate detection

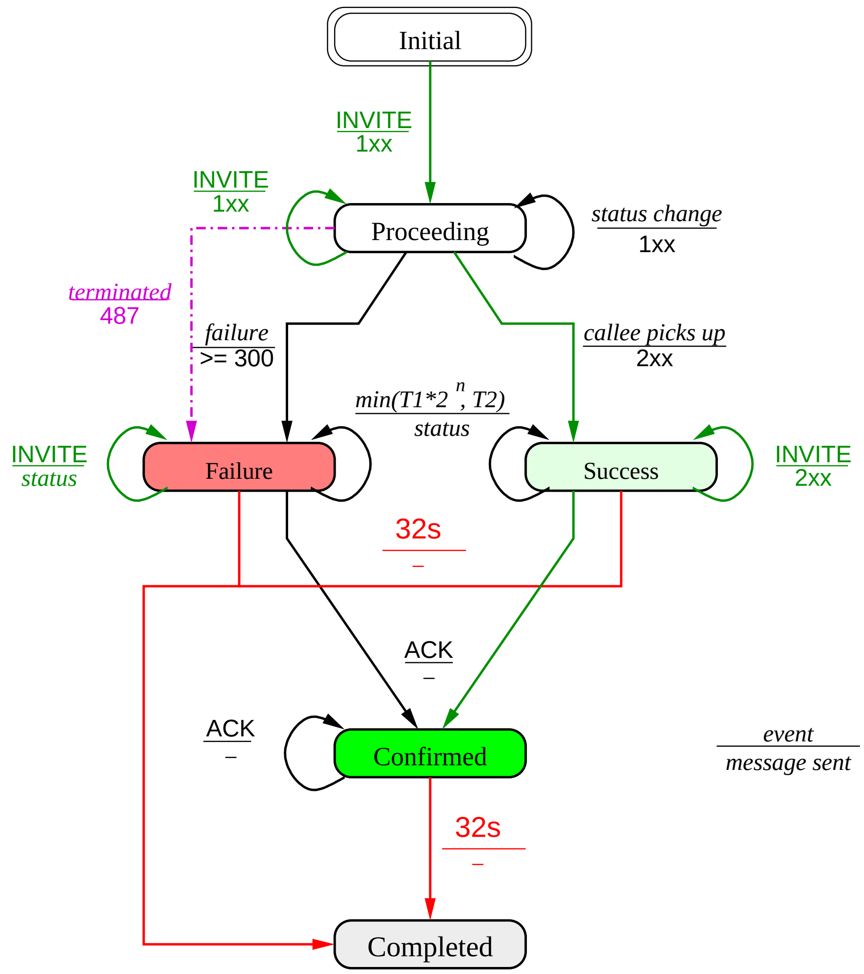
  Timestamp: timestamp at origin ➠ RTT estimation

- retransmit $\leq$ 11 times at 0.5, 1, 2, 4, 4, . . . seconds
- . . . until provisional (1xx) response
- then with interval 4 seconds

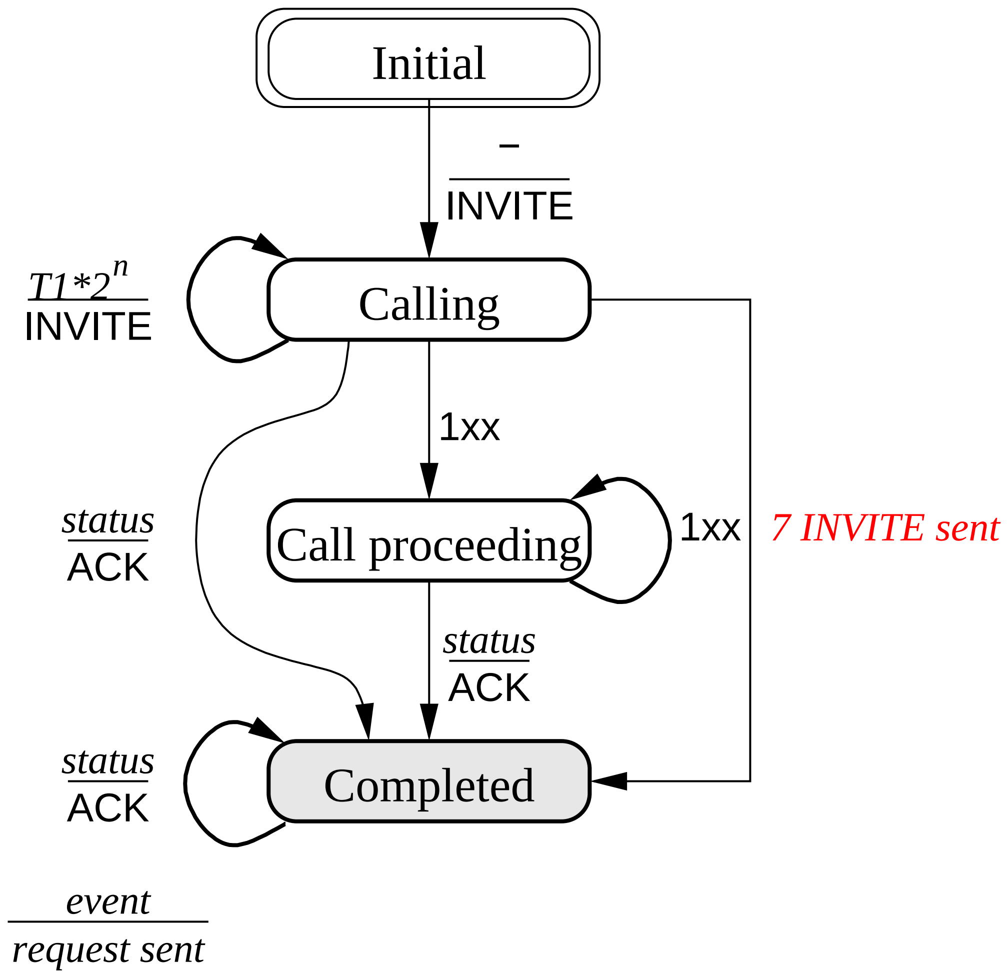
SIP reliability: INVITE

- retransmit request after 0.5, 1, 2, 4, 4, 4, 4 seconds
- until provisional or final response
- client confirms final response via ACK ➠
  1. $C \rightarrow S$: INVITE
  2. $S \rightarrow C$: 100, *user location*, *ringing*, ...
  3. $S \rightarrow C$: 200
  4. $C \rightarrow S$: ACK
- server repeats final response (as above) if no ACK

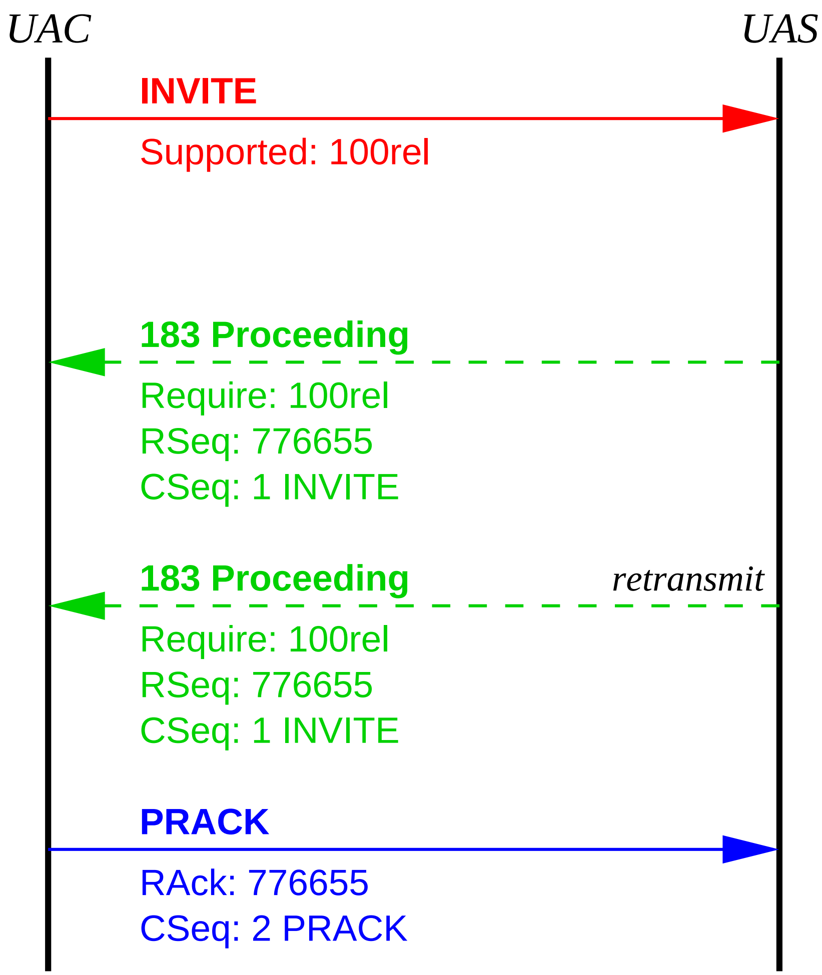
SIP state transition – server



SIP state transition – client

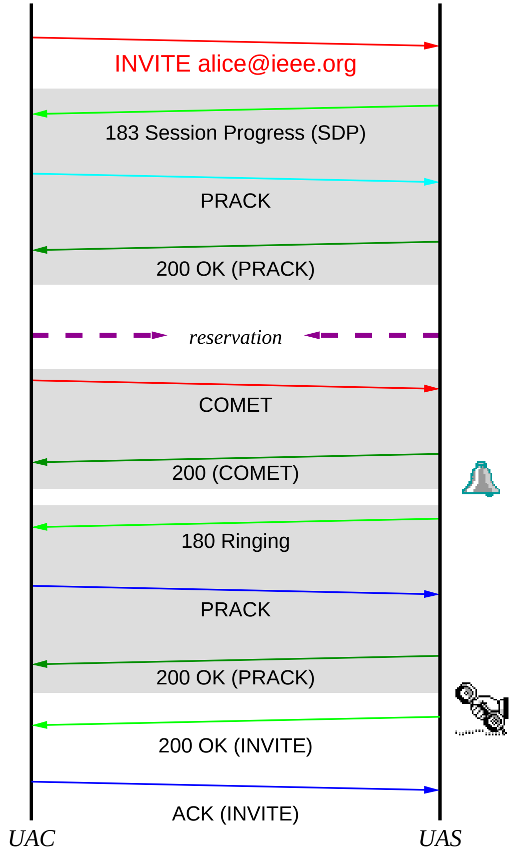


Reliability for provisional responses



Interaction with resource reservation

avoid "fast busy" after ringing ➠ interleave



SIP Registration and User Location

REGISTER

- registration one (common) way of letting local proxy know where you are
- on startup, send REGISTER to `sip.mcast.net` via multicast
- or pre-configured address
- registrations expire – determined by server
- cancel *all* registrations with `Expires: 0` or individual registrations in Contact header
- returns list of current registrations
- registrations should be authenticated
- registrations may be proxied ➠ mobility

REGISTER example

Send this registration to `sip.mcast.net`, forwarded to `home.edu`:

```
REGISTER sip:registrar.home.edu SIP/2.0
Contact: sip:room234@nyc.hilton.com
  ;q=0.9;expires=3600
Contact: sip:me@home.edu ;q=0.5
  ;expires=86400
Contact: mailto:me@home.edu
  ;q=0.3;expires="Su, Dec 31 2000"
```

User requests contact list

```
REGISTER sip:ss2.wcom.com SIP/2.0
Via: SIP/2.0/UDP there.com:5060
From: LittleGuy <sip:UserB@there.com>
To: LittleGuy <sip:UserB@there.com>
Call-ID: 123456792@here.com
CSeq: 1 REGISTER
Authorization:Digest username="UserB",
  realm="MCI WorldCom SIP",
  nonce="df84f1cec4341ae6cbe5ap359a9c8e88",
  uri="sip:ss2.wcom.com",
  response="aa7ab4678258377c6f7d4be6087e2f60"
Content-Length: 0
```

User requests contact list, cont’d.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP there.com:5060
From: LittleGuy <sip:UserB@there.com>
To: LittleGuy <sip:UserB@there.com>
Call-ID: 1234567892@here.com
CSeq: 1 REGISTER
Contact: LittleGuy <sip:UserB@there.com>
Contact: sip:+1-972-555-2222@gw1.wcom.com;user=phone
Contact: tel:+1-972-555-2222
Contact: mailto:UserB@there.com
Content-Length: 0
```

SIP Session (Media) Description

SIP message body

- requests and response can contain any (binary/text) object
- typically:
  - requests ➠ session (media) description
  - response ➠ session description on success, HTML or plain text on failure

SIP message body

described by:

Accept	media type
Accept-Language	language of response
Content-Type	type of media (text/html, application/sdp, . . .)
Content-Length	length of message body

MIME: `multipart/mixed`

Session description: SDP

- application-specific: media vs. events
- caller and callee indicate receive capabilities and receive address/port
- media address may not be same as signaling address ➠ PINT with PSTN addresses

Session Description Protocol (SDP)

- originally for Mbone session advertisements
- used for Mbone tools (sdr), RTSP, H.332
- *parameter=value*, no continuation lines
- global and per-media objects
- others (SMIL) in progress ➠ nesting (and/or)

SDP example for Internet telephony

session id version session creator

global

v=0
o= *root 2890844527 2890844527* IN IP4 *gw1.example.com*
s=*the subject of the call*
c=IN IP4 *128.59.16.1*
t= *0 0*

destination address

start/end time

port

RTP payload type

audio

m=audio *3456 RTP/AVP 0 97*
a=rtpmap:0 PCMU/8000
a=rtpmap:97 G723/8000

video

m=video *4180 RTP/AVP 98*
a=rtpmap:98 H263/90000
c=IN IP4 *128.59.16.2*

RTP format and clock rate

SIP Security, Authentication and Privacy

Security

hop-by-hop encryption & authentication: IPsec, SSL

proxy authentication: Proxy-Authenticate, for firewalls and PSTN gateways

URL-based authentication: plain-text URL password

end-to-end HTTP authentication: basic (password) and digest (challenge-response)

end-to-end cryptographic: PGP – as filter

also: anonymous calls

SIP authentication

Basic: include plain-text password in request, immediately or after 401 (Unauthorized) or 407 (Proxy Authorization) response

Digest: challenge-response with shared secret

Certificate: sign non-Via parts of request headers, body with PGP, PKCS #7

SSL, SSH: but only for TCP

- but: need more elaborate cryptographic capability indication in SDP

Basic authentication

- Challenge by UAS:

```
SIP/2.0 401 Unauthorized
WWW-Authenticate: Basic realm="business"
```

- client responds with

```
INVITE sip:alice@wonderland.com SIP/2.0
CSeq: 2 INVITE
Authorization: QWxhZGRpbjpvcGVuIHNlc2FtZQ==
```

  where authorization is base64(*userid*:*password*)

- usually caller → callee, but challenge can be in request

Digest authentication

- A calls B and fails:

```
SIP/2.0 401 Unauthorized
Authenticate: Digest realm="GW service",
   domain="wcom.com", nonce="wf84f1ceczx41ae6cbe5aea9
   opaque="42", stale="FALSE", algorithm="MD5"
```

- A tries again:

```
INVITE sip:UserB@ss1.wcom.com SIP/2.0
Authorization:Digest username="UserA", realm="GW serv
   nonce="wf84f1ceczx41ae6cbe5aea9c8e88d359",
   opaque="42", uri="sip:UserB@ss1.wcom.com",
   response="42ce3cef44b22f50c6a6071bc8"
```

Digest authentication

username: user authenticating herself

realm: several per user, used also for display

nonce: copied into Authorization

opaque: copied into Authorization

uri: original request URL

response: 32 hex digits:

KD (H(A_1), nonce-value : H(A_2))

for MD5: H(H(A_1) : nonce-value : H(A_2)))

where A_1 = *username* : *realm* : *passwd*

A_2 = method : *uri*

PGP authentication

- Request authorization – not necessary:

```
SIP/2.0 401 Unauthorized
WWW-Authenticate: pgp version="5.0"
  realm="Your Startrek identity, please",
  algorithm=md5, nonce="913082051"
```

- retry request:

```
Authorization: pgp version="5.0" ,
  realm="Your Startrek identity, please",
  nonce="913082051", signature="iQB1..."
```

PGP authentication

- computed across nonce, realm, method, header fields following Authorization, body
- may also be signed by third party (e.g., outbound proxy)

PGP encryption

- encrypt part of SIP message

```
INVITE sip:watson@boston.bell-telephone.com SIP/2.0
...
Encryption: PGP version=2.6.2,encoding=ascii

hQEMAxkp5GPd+j5xAQf/ZDIfGD/...
```

- here, encrypt

```
Subject: Mr. Watson, come here.
Content-Type: application/sdp

v=0
...
```

Anonymous calls

- near-end proxy that scrambles identifying information ("anonymous remailer") ➠ no call-state needed
- far-end proxy hides exact callee location
- Via hiding
- source and media IP addresses valuable ➠ NAPT
- can have third-parties vouch for calls ("caller-id") ➠ proxy signs request with (phone) company id

Anonymous calls

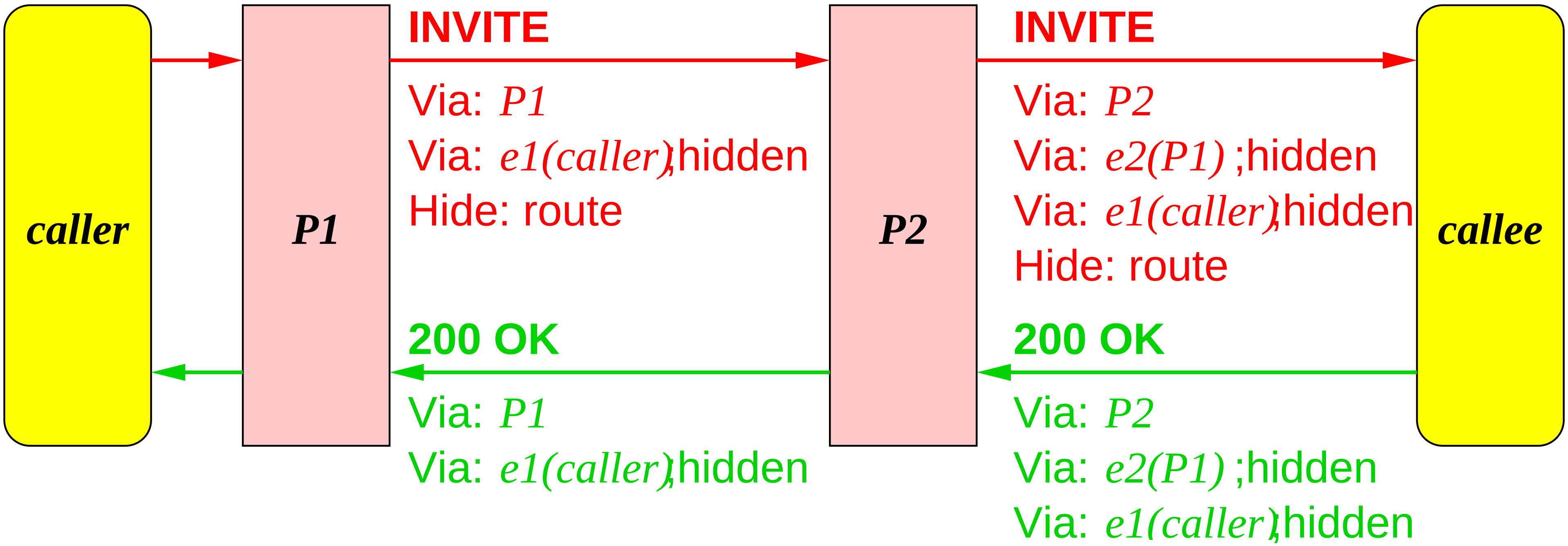
traceable: encrypt salted version

recognizable: “payphone” $\longrightarrow$ same caller, same identification ➠ non-salted encryption

confirmable: hash without key

non-returnable: (teachers) ➠ encrypt only URL, not name

Hiding signaling paths: Via hiding



encrypt with “salt”

Getting SIP through firewalls and NATs

- SIP proxy as firewall controller or NAT ALG
- much easier than H.323:
  - single protocol vs. H.225.0 + H.245
  - SDP $\ll$ H.245.0
  - single-stage negotiation
  - no need to maintain TCP connections during call
- need to understand INVITE, ACK and BYE
- if final SDP in success ACK: ACK only

SIP billing/charging

What for?

- transport ➠ resource reservation protocol
- SIP services (call processing) ➠ authentication
- PSTN gateway services
- media server services (translation, storage)

How?

- resource reservation protocols
- SIP-in-DIAMETER approach
- server log files

SIP Caller Preferences

Preferences

callee: scripts, CPL, REGISTER advice in Contact, . . .

caller: help guide routing (“no home number”) and order of attempts when forking (“try videophone first, then phone, then answering service”)

“caller proposes, callee disposes”

Extended SIP Contact header

q	location preference
class	business, residence
description	show to caller
duplex	full or half-duplex
feature	call handling features
language	languages spoken
media	audio, video, text/numeric, . . .
mobility	fixed or mobile
priority	“only in case of emergency”
scheme	URL schemes (tel, http, . . .)
service	IP, PSTN, ISDN, pager, . . .

Contact example

q=*quality* gives preference.

```
SIP/2.0 302 Moved temporarily
Contact: sip:hgs@erlang.cs.columbia.edu
 ;action=redirect ;service=IP,voice-mail
 ;media=audio ;duplex=full ;q=0.7;
Contact: tel:+1-415-555-1212 ; service=ISDN
 ;mobility=fixed ;language=en,es,iw ;q=0.5
Contact: tel:+1-800-555-1212 ; service=pager
 ;mobility=mobile
 ;duplex=send-only;media=text; q=0.1; priority=urgent;
 ;description="For emergencies only"
Contact: mailto:hgs@cs.columbia.edu
```

Accept-Contact and Reject-Contact

- determine order of contacting users:

```
Accept-Contact: sip:sales@acme.com ;q=0,
;media="!video" ;q=0.1,
;mobility="fixed"  ;q=0.6,
;mobility="!fixed" ;q=0.4
```

➠ “avoid connecting me to sales; I prefer a landline phone; try

- Reject-Contact: rule out destinations

```
Reject-Contact: ;class=personal
```

Request-Disposition

- proxy or redirect
- cancel ringing second phone after first picked up?
- allow forking?
- search recursively?
- search sequentially or in parallel?
- queue the call?

```
Request-Disposition: proxy, recurse, parallel
```

SIP Protocol Status and Implementations

Status

- Proposed Standard, Feb. 1999 – RFC2543
- bakeoffs every 4 months $\longrightarrow$ cross-vendor interoperability tests

	host	when	companies
1	Columbia University	April 1999	16
2	pulver.com	August 1999	15
3	Ericsson	December 1999	26
4	3Com	April 2000	36
5	pulver.com	August 2000	
6	Sylantro	December 2000	
7	ETSI	April 2001	

SIP implementations

Roughly in order of maturity:

- proxies and redirect servers for service creation
- PC-based user agents – Windows and other OS
- Ethernet phones
- softswitches (Megaco/MGCP/. . .) “crossbar”
- firewall and NAT enhancements
- SIP-H.323 translators
- unified messaging

On-going SIP Implementations

3Com
AudioTalk Networks
Broadsoft
Catapult
Cisco
Carnegie-Mellon University
Columbia University
Delta Information Systems
dynamicsoft
Ellemtel
Ericsson
Hewlett-Packard

Hughes Software Systems
Indigo Software
Iwatsu Electric
Komodo
Lucent
MCI Worldcom
Mediatrix
Microappliances
Netergy
Netspeak
Nokia

ObjectSoftware
Nortel
Nuera
Pingtel
RaveTel
Siemens
Telogy
Ubiquity
Vegastream
Vovida

Columbia University SIP implementations

- *sipd* proxy/redirect server, registrar
- *sipc* user agent
- SIP C++ library
- SIP-H.323 gateway
- SIP multiparty conference server (“bridge”)
- PSTN gateway
- SIP/RTSP unified messaging server

sipd = SIP registration + redirect server

- registration via unicast and multicast
- location server functionality:
  1. lists (`ug-students@cs`), ambiguous names (`lee@cs`)
  2. if no match, map (`b.clinton@whitehouse`) to user name
  3. if no registration, look up in LDAP
- Apache (httpd)-style configuration and logging
- basic, digest and PGP authentication
- sip-cgi and CPL

SIP server implementation

HTTP, SIP, RTSP (+ email) share common format ➠

functionality	C lines (≈)
generic RFC822-style parser	500
HTTP generic headers	330
SIP, RTSP	300

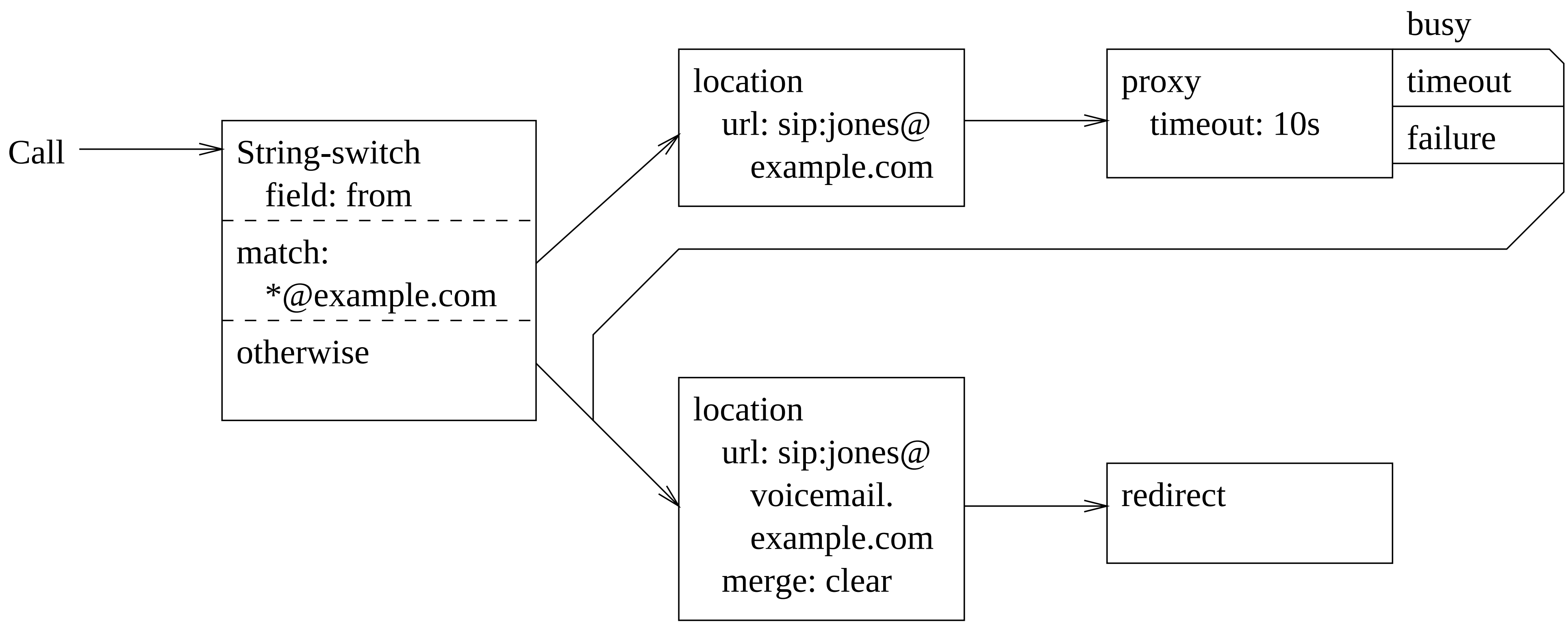
“Active Phone Networks”

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

Internet phone “appliance”

- phone = \$49.95; PC $>$ \$600 (GPF included)
- *Ethernet phone* ➠ no PBX for switching
- examples (not all SIP yet): 3Com/S4, Columbia University, e-tel, Mitel, Nortel, Pingtel, Siemens, Symbol Technologies, …
- typically, mircoprocessor (ARM) for signaling + DSP for speech coding, echo cancellation

Columbia e*phone

- DSP for voice coding *and* signaling ➠ limited memory (e*phone: 512 kB SRAM)
- only need minimal IP stack (IP/UDP/RTP, DHCP, SIP, tftp, DNS), not TCP
- also, MP3 radio
- sensor interfaces to the world: chair, IR, temperature, . . .

Columbia e*phone

SIP Services

SIP services

- buddy lists and notifications
- proxy and fanout
- IN services
- MCUs and “multi-unicast”

Signaling ⟵ event notification

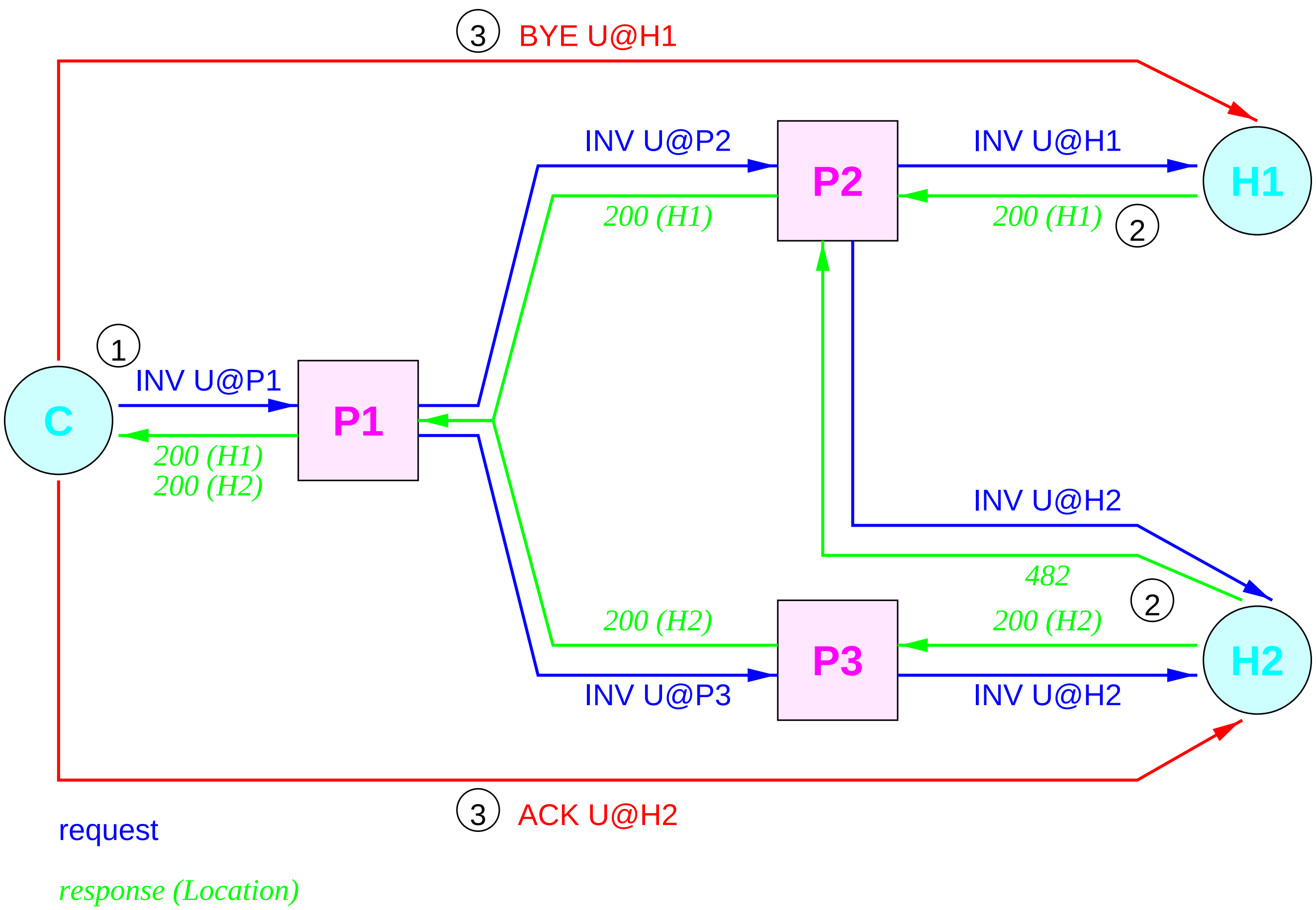
- call queueing . . . buddy lists . . . event notification
- also: message waiting, pickup group, ACD
- SUBSCRIBE to events (e.g., message waiting, pending call, presence)
- server NOTIFY
- can use forking
- handle subscriptions using CPL
- transition to multicast if large group of subscribers

SIP “fan-out”

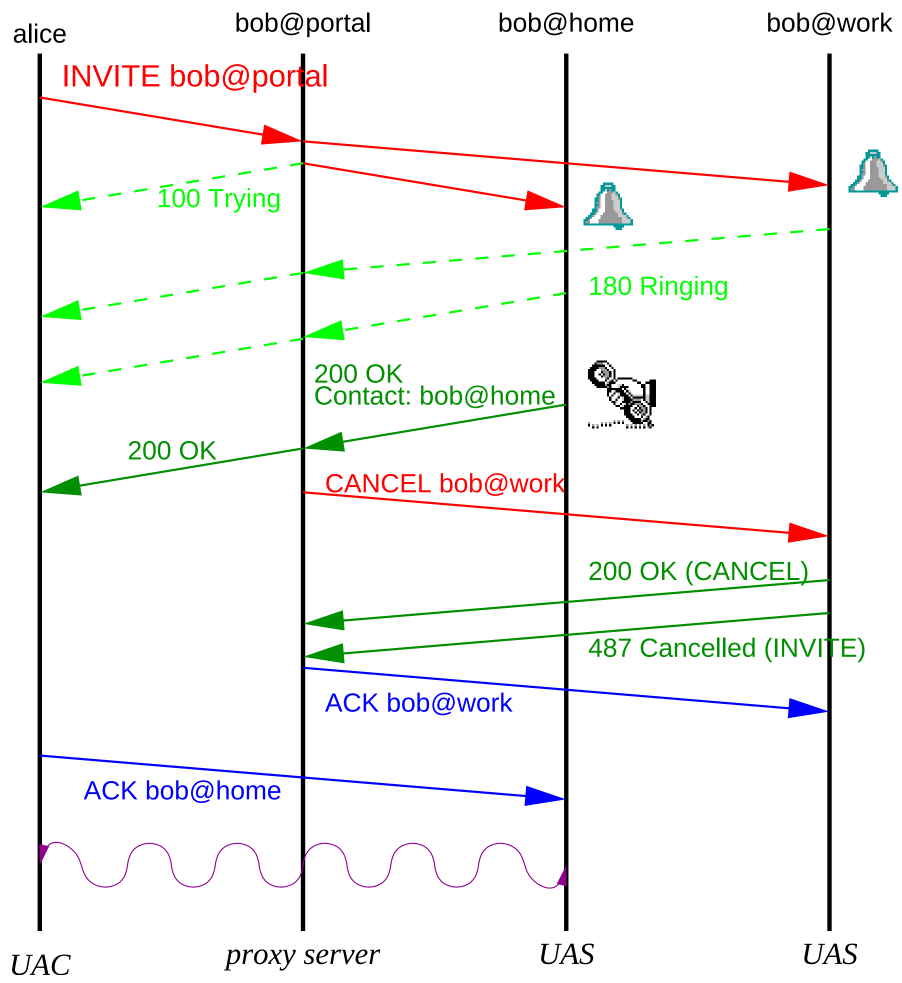
- proxy server may issue several request
- e.g., all known login locations
- waits for definitive response (≥ 200)
- 3xx (redirect) code: possibly recurse
- returns “best” (lowest-class) definitive response
- 200 (OK) and 6xx (Busy, . . .) terminate search
- CANCEL: terminate other search branches

Branching requests

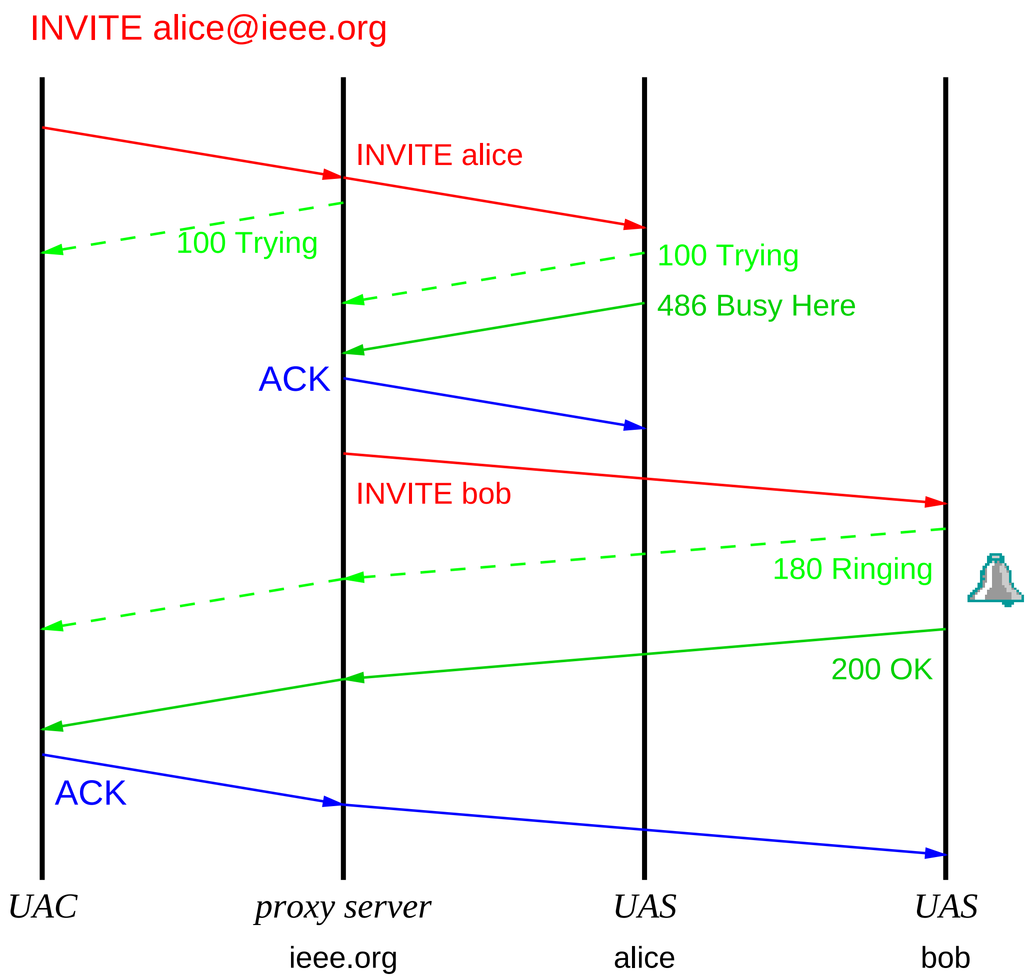
Search for callee in several places:



Parallel search with CANCEL



Sequential search



IN call forwarding features

SIP can implement intelligent network features:

name	feature	SIP note
SCF	selective call forwarding	302, Contact
SCR	selective call reject	302, Contact
CFU	call-forwarding unconditional	302, Contact
CFB	call-forwarding busy	302, Contact
CFNR, CFDA	call forwarding, no response	302, Contact
DND	call forwarding to voice mail	302, Contact

➠ differences as server program or in end system

IN call handling features

name	description	SIP notes
CW	call waiting	not: $>$ 1 call pres.
(A)CB	call back	email, log file
ICS	incoming call screening	end system, proxy
OCS	outgoing call screening	firewall + outbound proxy
CID, CND	calling # delivery	From
CLIR, CIDR, CNDB	calling # delivery blocking	leave out, anonymizer
TWC	three-way calling	Also

SIP advanced services

- Also for third-party control: A asks B to send request to C
- alternative: TRANSFER request (in progress)
- generic establishment of call legs
- Request-Disposition for enumerated features
- Contact headers for feature description

Building advanced services

Construct from element *behavior*, not feature descriptions:

request URL: next resolution stage

From: logical call source

To: logical call destination

SDP "c=": address media is to be sent to – Internet or PSTN!

Also: indication of additional requests to send

Contact: indication of alternate participants or future direct destination

Building advanced services: rules

- SIP responses go to requestor
- INVITE establishes single data association
- don't ring for new additional participant in existing call ➠ call transfer
- BYE terminates From leg only
- OPTIONS may use Also
- call ends when last party leaves
- alternative: TRANSFER asks to send INVITE

Multipoint Control Units (MCUs)

URL = *conference-id@mcu-host*

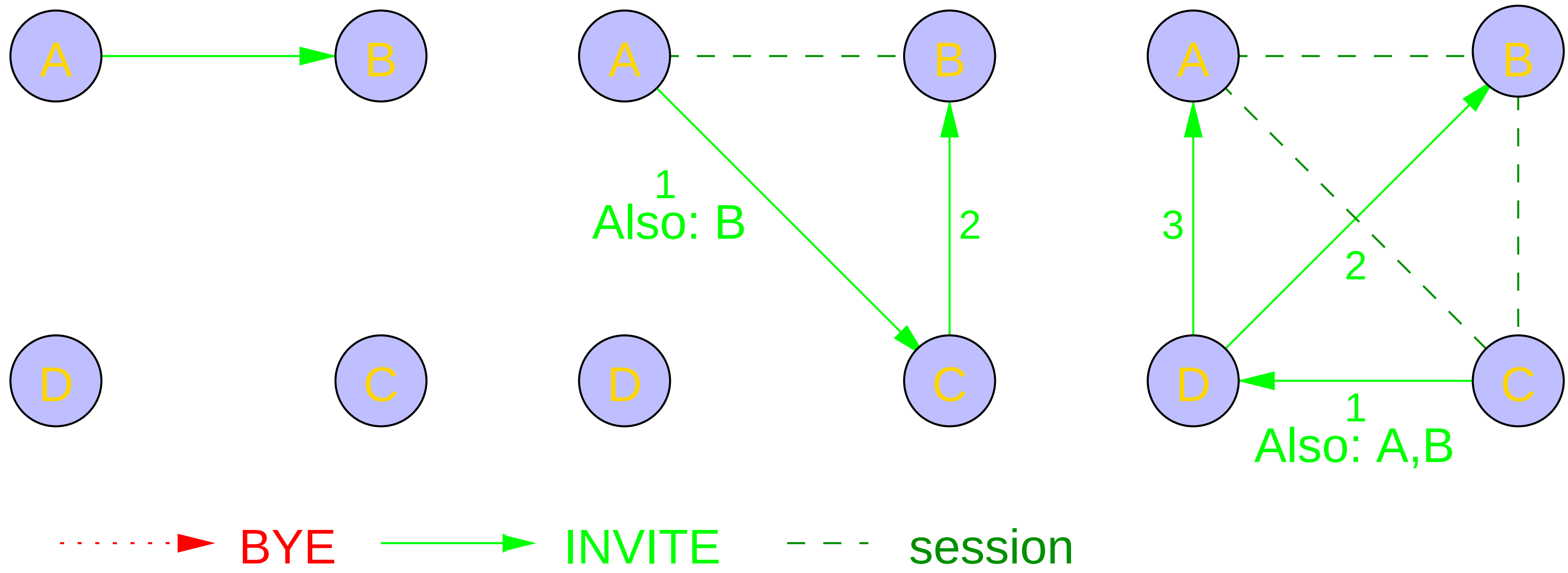
call in: new participant invites MCU

call out: MCU invites participants

Mesh

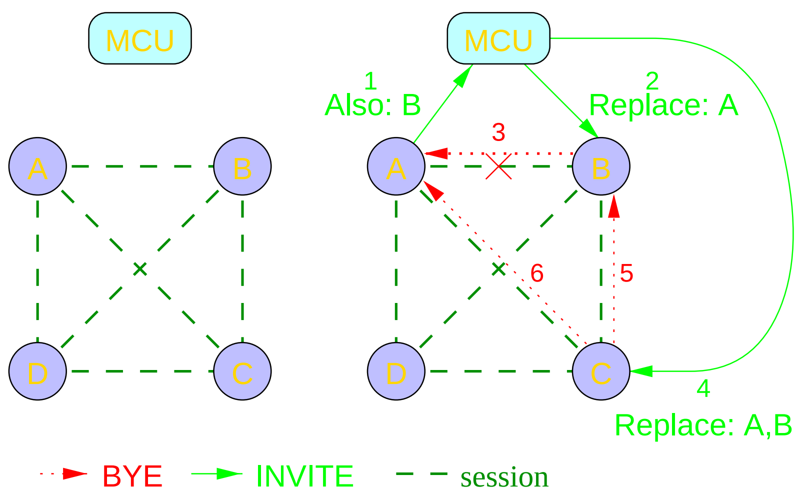
- multicast not always available
- easier for adding third party to call
- full mesh of all participants
- if x wants to add party y, invite y with list of other participants in Also:
- any member of call can invite
- difficulty: synchronization

Mesh



MCUs: transition from mesh to MCU

- transition from mesh to MCU
- Replaces = "inverse" Also
- ask recipient to delete calls with named parties
- recipient sends BYE



SIP user location

- local multicast of invitation
- login-based via NFS
- recursive “finger”-traversal
- name translation: *Alexander.G.Bell* ➠ *agb*
- list aliases
- active badges

- REGISTER announces location, with time limit
- REGISTER + Contact sets new location

Interaction with directory services

- LDAP (with dynamic extensions)
- rwhois
- whois++ (RFC 1913)
- possibly implement SIP interface ➠ simpler clients

Automatic call distribution (ACD)

- caller connects to server for company, indicates language, subject, organization, urgency, . . .
- alternatives:
  - proxy server maintains queue state, forwards
  - (local) multicast signaling ➠ first suitable agent answers
    proxy suppresses multiple responses
    avoids centralized state maintenance

Hold

➠ temporarily disable media delivery

- multicast: use RTCP “interest indication”
- thus, unicast only
- send INVITE with SDP port number = 0 for media

music-on-hold ➠

- ask RTSP server to stream to callee address
- send INVITE with SDP address of music server (multicast!)

Camp-on service

Choices:

1. callee indicates time to call back
2. "polling": caller issues repeated INVITE
3. caller indicates desire to wait:

```
C->S: INVITE sip:watson@example.com SIP/2.0
      Call-Disposition: queue

S->C: 181 Queued: 2 pending
      181 Queued: 1 pending
      200 OK
```

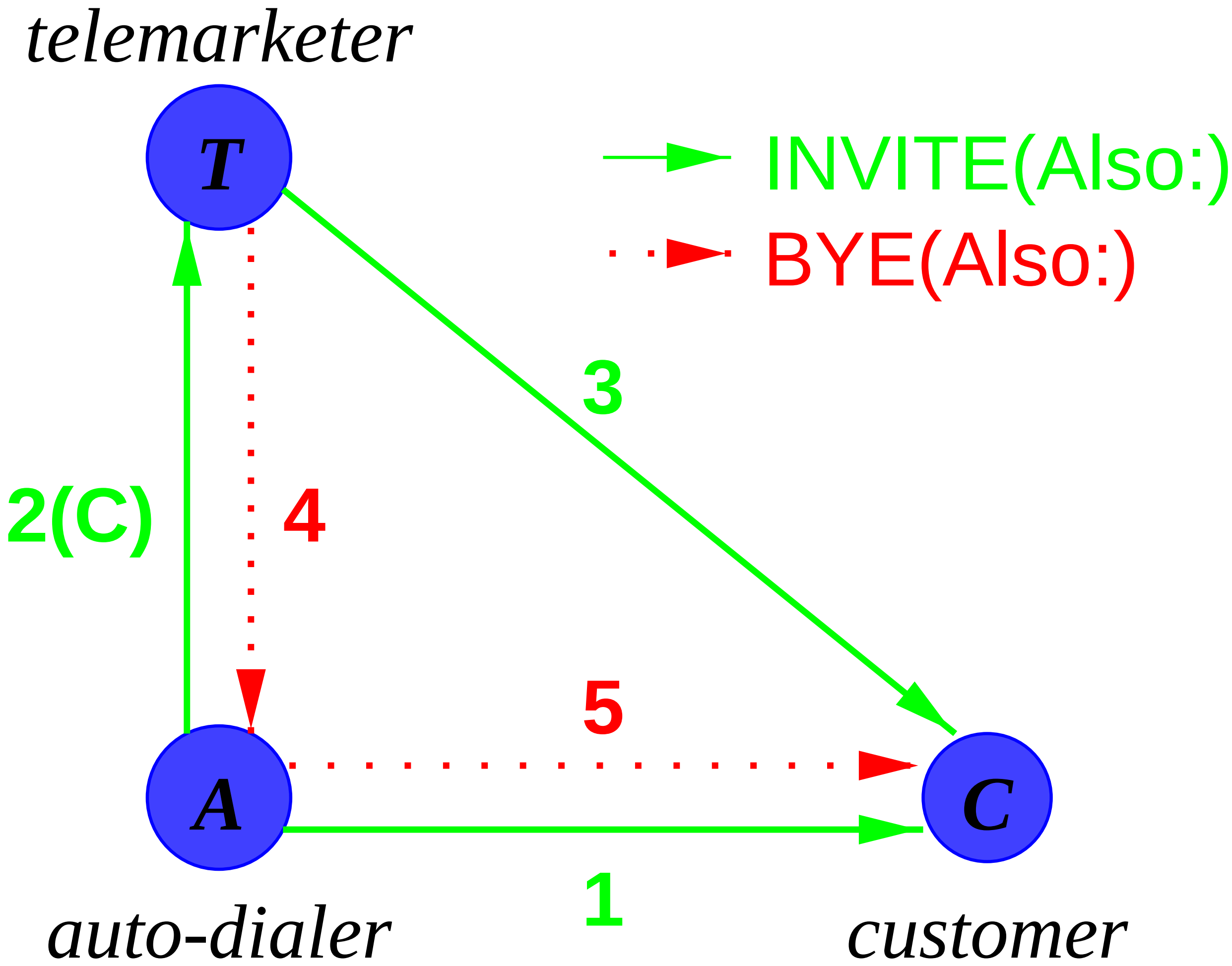
Outgoing call handling

Three-party setups:

- secretary dials for boss
- auto-dialer hands call to telemarketer
- attended call transfer
- operator services

➠ treat as three-party calls

Outgoing call handling: telemarketing



SIP and H.323

SIP – H.323 Comparison

	H.323	SIP
Architecture	stack	element
Origin	ITU	IETF
Conference control	yes	no
Protocol	mostly TCP	mostly UDP
Encoding	ASN.1, Q.931	HTTPish
Emphasis	telephony	multimedia, multicast, events
Address	flat alias, E.164, email	SIP, E.164 URLs

Both SIP and H.323 are evolving: SIP additions, H.323v2 implemented, v3 to be decided.

SIP and H.323 elements

H.323	SIP + SDP
H.225.0 + RAS	SIP
H.245	SDP, SMIL, . . .
gatekeeper	proxy

H.323 Resource Reservation

- *local* admission decision
- prior to call setup $\longrightarrow$ no information about bandwidth available
- works only for “yellow cable Ethernet”
- other applications have to notify GK
- SIP: RSVP, YESSIR, DiffServ + call preconditions

SIP vs. H.323: Call Setup

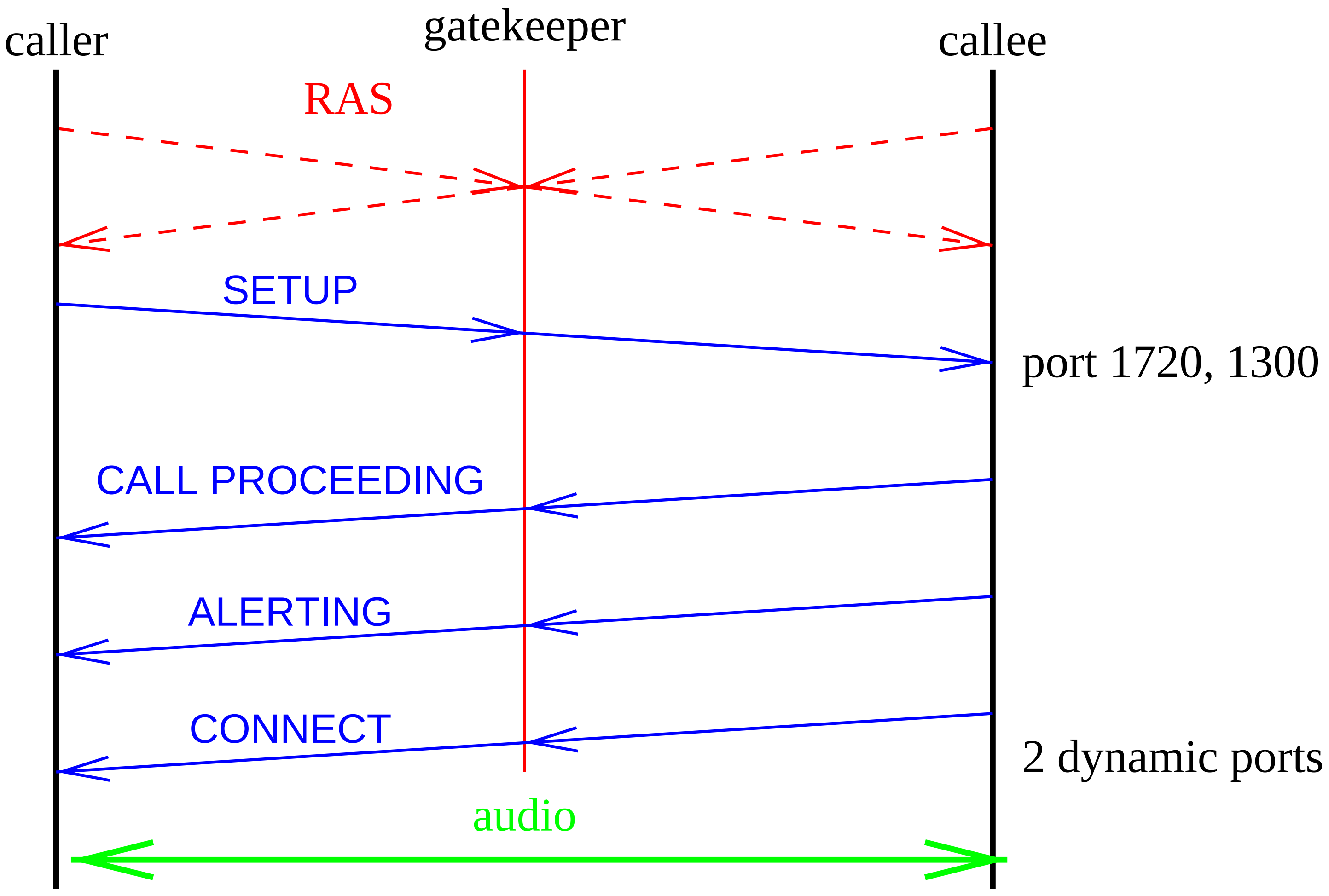
H.323v1: several TCP connections (H.245, Q.931) $\longrightarrow$ very long latency (6.5-8 RTTs), particularly with packet loss; currently in *NetMeeting*

H.323v2: merge H.245 and Q.931 ("FastConnect")

H.323v3: allow UDP

End systems need to support all versions.

H.323v3 call setup



Services

Service	H.323	SIP
Call transfer	H.450.2	“30x”
Call diversion	H.450.3	“30x”
Call hold	H.450.4	SDP-based
Call park	H.450.5	REGISTER
Call waiting	H.450.6	INVITE
Message waiting	H.450.7	email, NOTIFY
Call forward busy	H.450.9	“30x”

H.323 vs. SIP: Basic Call Control

(modified from Dalgic and Fang, *Comparison of H.323 and SIP*)

Service	H.323v1	v2	v3	SIP
Call holding	no	yes	yes	yes
Call transfer	no	yes	yes	yes
Call forwarding	no	yes	yes	yes
Call waiting	no	yes	yes	yes

H.323 vs. SIP: Advanced Features

Service	H.323v1	v2	v3	SIP
Third party control	no	no	no	yes
Conference	yes	yes	yes	yes
Click-to-dial	?	?	?	PINT
Capability exchange	better	better	better	yes
HTML transport	no	no	no	yes
Call forking	no	no	no	yes

H.323 vs. SIP: Quality of Service

	H.323v1	v2	v3	SIP
Call setup delay	6-7 RTT	3-4	1.5-2.5	1.5
Loss recovery	TCP	TCP	better	better
Fault detection	yes	yes	yes	yes
Mid-call failure	fail	fail	fail	live
Registrar failure	fail	fail	backup	multicast
GK/Proxy redundancy	no	no	backup	SLP, DNS, DHCP
Loop detection	no	no	PathValue	Via, hops, time

H.323 vs. SIP: Manageability

	H.323v1	v2	v3	SIP
Admission control	yes	yes	yes	no (RSVP)
Policy control	yes	yes	yes	ob proxy
Resource reservation	local	local	local	no (RSVP)

H.323 vs. SIP: Scalability

	H.323v1	v2	v3	SIP
Complexity	more	more	more+	less
Server processing	SF	SF	SF/SL, TSF	SL, TSF/TSL
Inter-server	no	no	yes	yes

TS: transaction state; SF: call statefull; SL: call stateless

H.323 vs. SIP: Flexibility

	H.323v1	v2	v3	SIP
Transport protocols	TCP	TCP	TCP/UDP	any
Extensibility	unlabeled vendor extensions			IANA, labeled
Customization		harder		easier
Version compatibility	N/A	yes	yes	N/A
SCN interoperability	good	good	good	TBD
protocol encoding	binary (ASN.1, Q.931)			text

Mobility Support Using SIP

Elin Wedlund and Henning Schulzrinne, “Mobility Support Using SIP”, *WoWMoM*, *Seattle*, August 1999.

Overview

pure-IP mobility $\leftrightarrow$ IP over GSM, 3G, ...

- SIP
- mobile applications
- mobile IP issues for Internet telephony
- mobility support using SIP
- performance
- future work

Mobility in an IP environment

Terminal mobility: terminal moves between subnets

Personal mobility: different terminals, same address

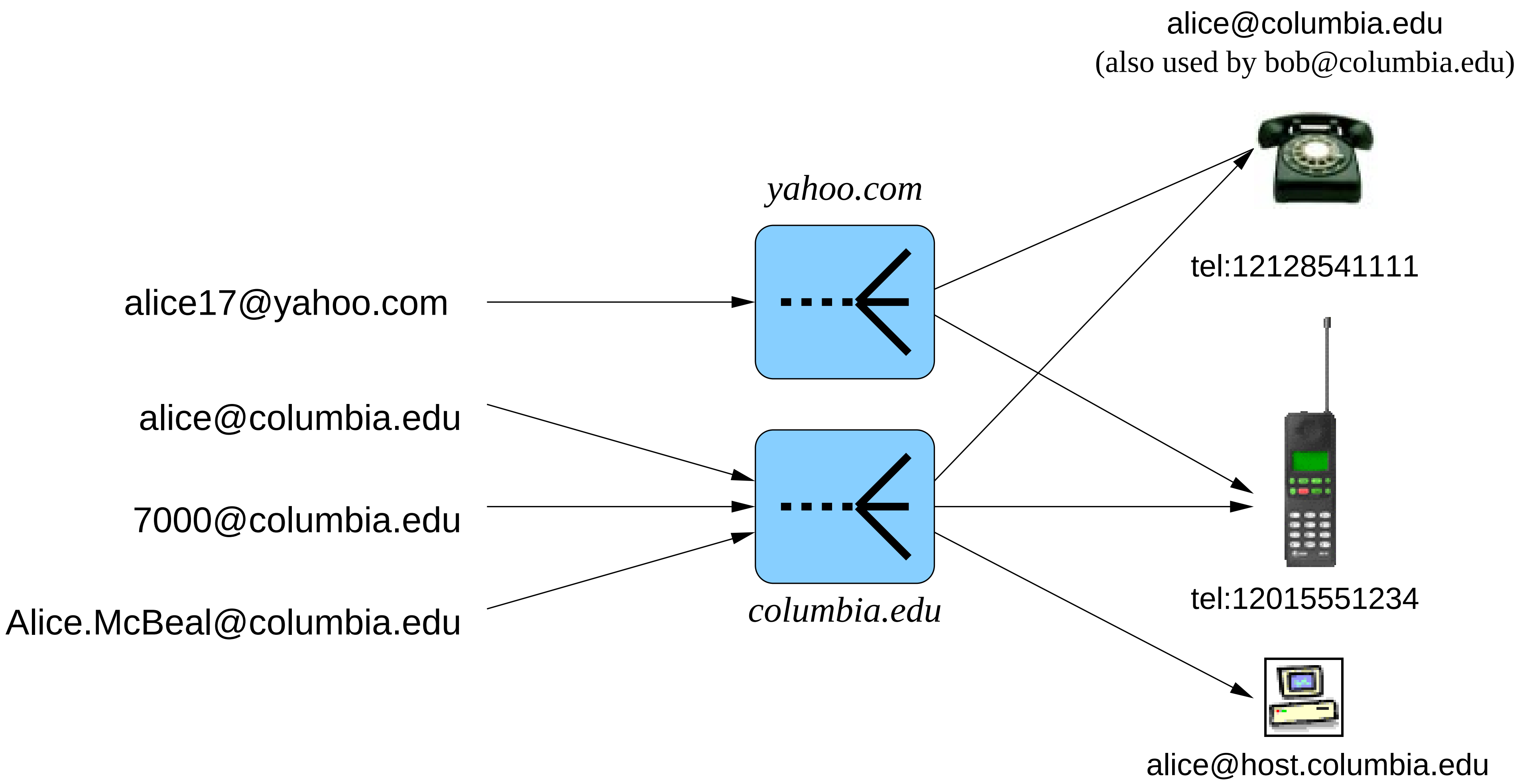
Service mobility: keep same services while mobile

Session mobility: move active session across terminals

Terminal mobility

- domain of IEEE 802.11, 3GPP, mobile IP, . . .
- main problems in some versions:
  - handover performance
  - handover failure due to lack of resources in new network
  - authentication of redirection

Personal mobility



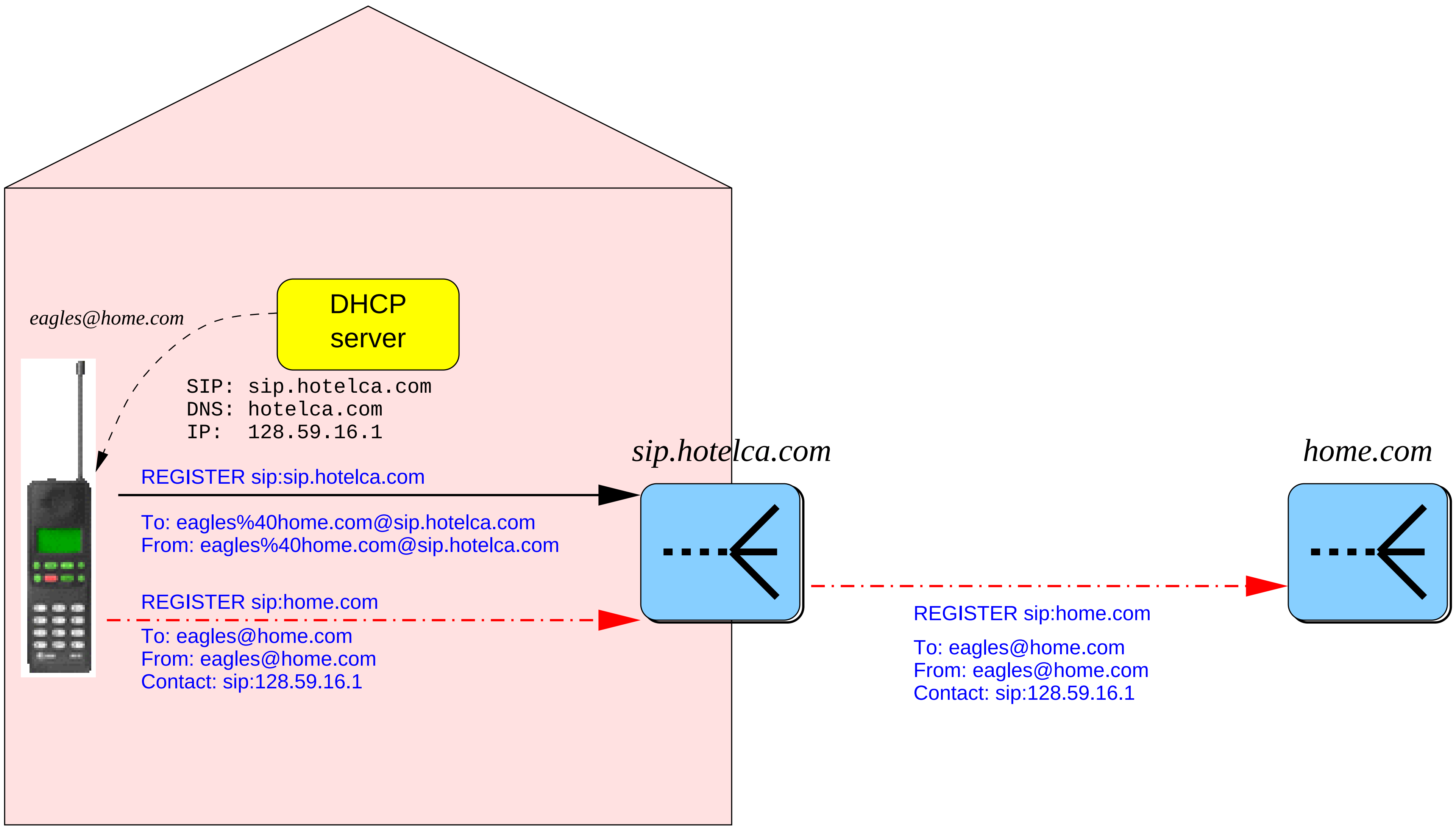
Personal mobility

- switch between PDA, cell phone, PC, Ethernet phone, Internet appliance, ...
- several "generic" addresses, one person/function, many terminals
- e.g., `tel:2129397042`, `hgs@cs.columbia.edu`, `schulzrinne@yahoo.com` or `support@acme.com`
- SIP is designed for that – proxying and redirection does translation
- but: need mapping mechanisms to recognize registrations as belonging to the same person
- some possible solutions:
  - dip into LDAP personnel database or `/etc/passwd` to match phone number and variations of name (*J.Doe*, *John.Doe*, *Doe*)

– need dialing plan to recognize `7042@cs.columbia.edu` and `tel:2129397042` as same

Visiting a remote network

- register locally (multicast, DHCP) **and**
- register at home


eagles@home.com
DHCP server
SIP: sip.hotelca.com
DNS: hotelca.com
IP: 128.59.16.1
sip.hotelca.com
home.com
REGISTER sip:sip.hotelca.com
To: eagles%40home.com@sip.hotelca.com
From: eagles%40home.com@sip.hotelca.com
REGISTER sip:home.com
To: eagles@home.com
From: eagles@home.com
Contact: sip:128.59.16.1
REGISTER sip:home.com
To: eagles@home.com
From: eagles@home.com
Contact: sip:128.59.16.1

Getting home services

- may *want* to use home services – e.g., lawyer per-client billing, third-party authentication
- UA can add `Route` header to force outbound proxy to route request through home proxy
- *cannot* be used to enforce network policy

Service mobility

Examples:

- speed dial & address book
- media preferences
- special feature buttons (voice mail, do-not-disturb)
- incoming call handling instructions
- buddy lists

$\longrightarrow$ independent of terminal (including pay phone!), across providers

Service mobility

- REGISTER can retrieve configuration information (e.g., speed dial settings, distinctive ringing or voice mail settings)
- but needs to be device-independent
- most such services (e.g., voicemail forwarding, call filtering) should remain on server(s)

Separate issue: how does the payphone (or colleague's phone) recognize you?

- PDA (IR)
- i-button
- fingerprint

- speech recognition, . . .

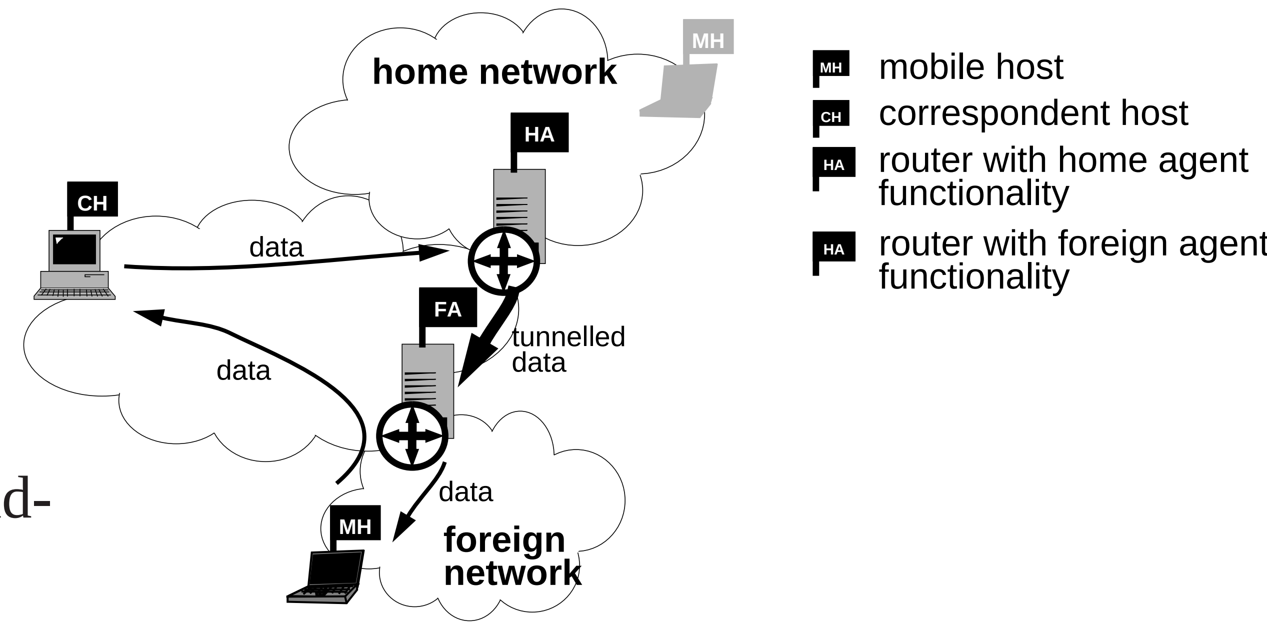
One device, but changing set of owners!

Service mobility – call handling

- need uniform basic service description model $\longrightarrow$ Call Processing Language (CPL)
- CPL = XML-based flow graph for inbound & outbound calls
- CPL for local call handling
- update CPL from terminal: add telemarketer to block list
- harder: synchronize CPL changes across multiple providers
- one possibility: REGISTER updates information, but device needs to know that it has multiple identities
- merging of call logs

Terminal mobility – details

- move to new network ➠ IP address changes (DHCP)
- mobile IP hides address changes
- but: little deployment
- encapsulation overhead
- dog-legged routing
- may not work with IP address filtering



Aside: Where is Mobile IP Needed?

Not needed if short-lived, restartable client-server connections:

http	short, stateless
smtp	short, restartable
pop, imap	short, restartable
telnet	yes, but rarely used by mobiles (?)
ftp	restartable, rare
chat, irc	yes, but fixable (proxy, protocol)

Requirements for VoIP Mobility

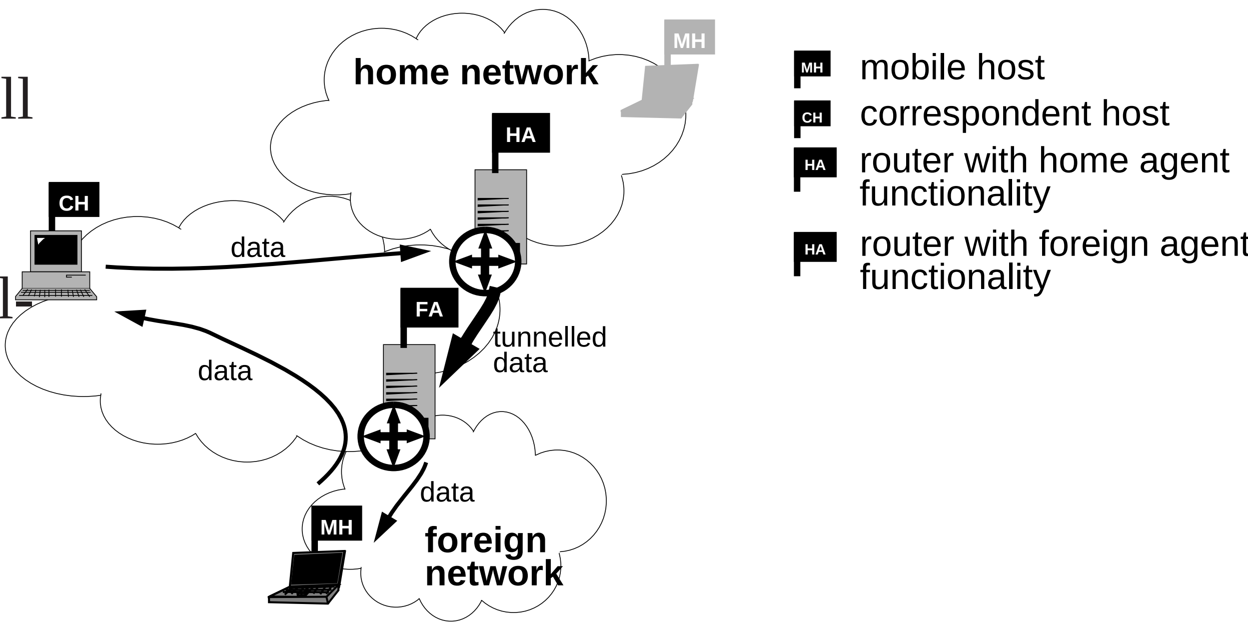
- fast hand-off, preferably without network support:
  - voice packet every 20–50 ms
  - FEC can recover 2–3 packets
- low packetization overhead:

headers	IP+UDP+RTP	40 bytes
G.729 payload	8 kb/s, 10 ms	$n \cdot 10$ bytes

- simple end systems

Mobile IP Issues

- encapsulation
- dog-legged routing
- binding updates still through HA
- may fail with IP address filters
- stack/infrastructure changes

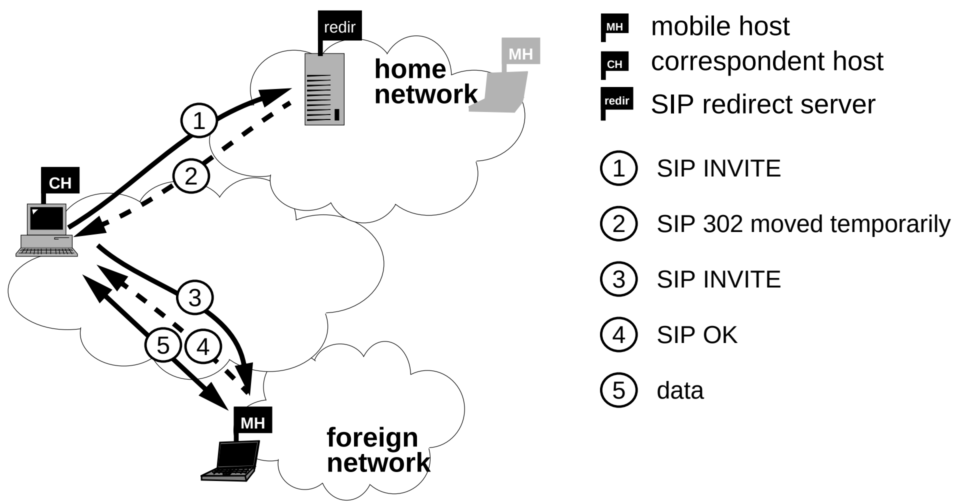


SIP Mobility Overview

- designed for *personal mobility*, but boundary to terminal mobility fluid
- pre-call mobility ➠ SIP proxy, redirect
- mid-call mobility ➠ SIP re-INVITE, RTP
- recovery from disconnection

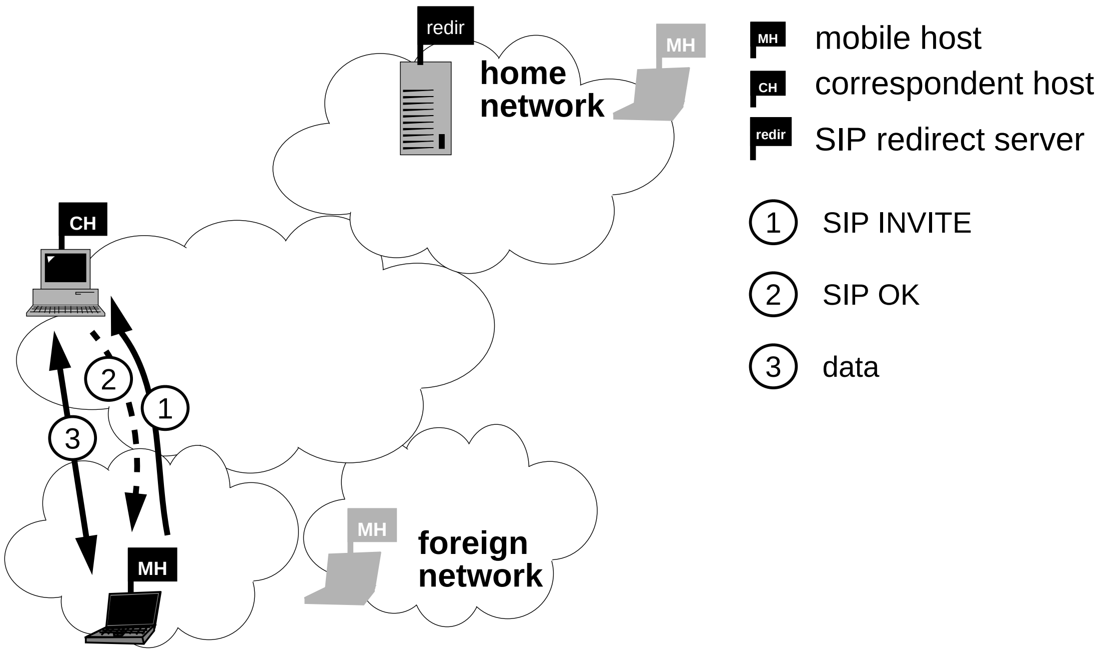
SIP mobility: pre-call

- MH acquires IP address via DHCP
- optional: MH finds SIP server via multicast REGISTER
- MH updates home SIP server
- optimization: hierarchical LR (later)



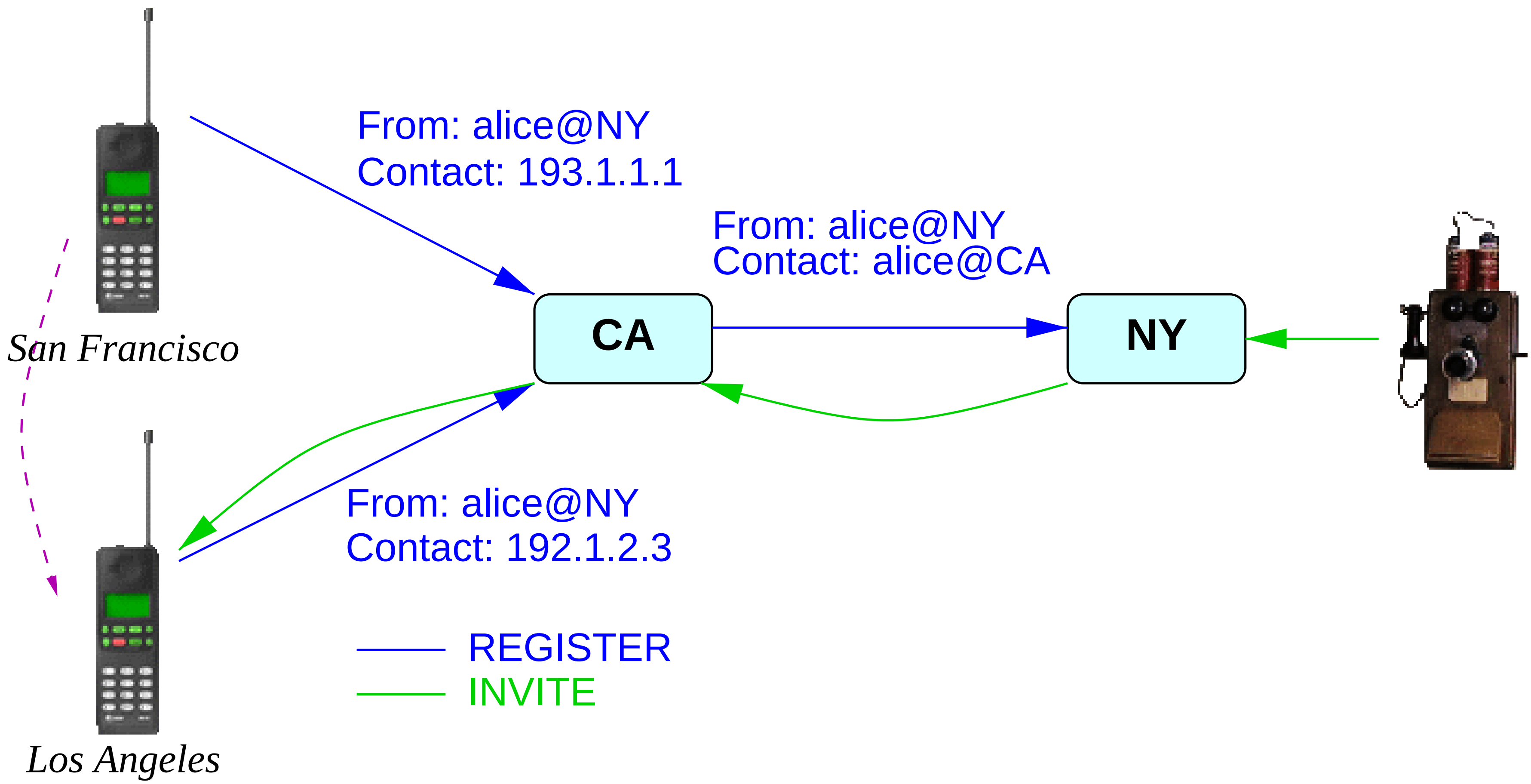
SIP Mobility: Mid-call

MH→CH: new INVITE, with Contact and updated SDP



SIP Mobility: Multi-stage Registration

Don't want to bother home registrar with each move



802.11 Movement Detection: Ad-Hoc Mode

no “access point” ➠ regular station as BS

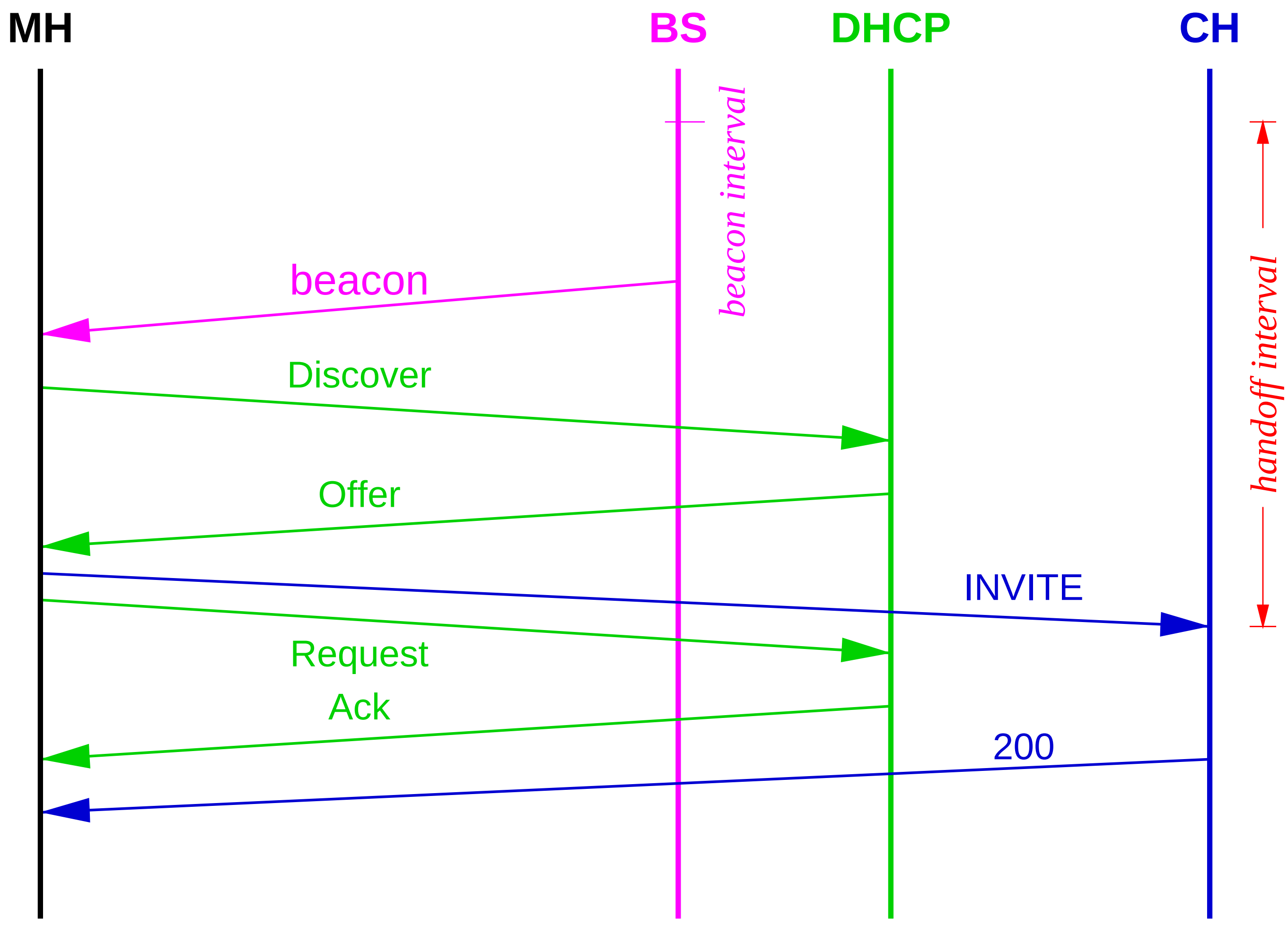
- BS serves as default router
- periodic multicast beacon
- pick best: driver provides SNR, strength
- could use regular multicast packets for quick BS discovery

802.11 Movement Detection: Infrastructure Mode

access point (AP) for BSS

- attachment handled by MAC layer, invisible to application
- BSSID is contained in 802.11 packet, but
  - BSSID not visible to application
  - driver doesn't get notified if MH attaches to new AP
- modified driver that polls hardware?

Handoff Performance



Open Issues

- handoff performance in a loaded network
- soft hand-off: IP-level vs. application proxies
- soft hand-off for 802.11 infrastructure mode possible?
- RTP issues: collision detection

Conclusion

- mobile telephony = most common mobile application
- all-IP network: can't punt hand-off
- terminal mobility as special case of personal mobility
- SIP-based mobility ➠ immediate deployment

Programming SIP Services

Programming SIP services

	safety	language?	party?
SIP-cgi	same as scripting	any	callee
servlets	same as Java	Java	callee
CPL	very	XML	both
applets	same as Java	Java	caller

Programming services

- “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 SIP Servers

- extend SIP user/proxy/redirect server functionality without changing server software
- server manages retransmission, loop detection, authentication, . . .
- Perl, Tcl, VB scripts

Examples

- Call forward on busy/no answer
- Administrative screening (firewall)
- Central phone server
- Intelligent user location
- Third-party registration control
- Calendarbook access
- Client billing allocation (lawyer's office)
- End system busy
- Phone bank (call distribution/queueing)

cgi Script Functionality

called for any method except ACK or CANCEL

- proxying of requests
- returning responses
- generate new requests

once for each request or response or timeout

cgi Script Mechanism

environment variables: headers, methods, authenticated user, . . .

stdin: body of request

stdout: new request, meta-requests:

- CGI- requests for proxying, response, default action
- script cookie for state across messages
- reexecute on all, final response, never

Cgi Example: Call Forwarding

```
use DB_File;
sub fail {
    my($status, $reason) = @_;
    print "SIP/2.0 $status $reason\n\n";
    exit 0;
}

tie %addresses, 'DB_File', 'addresses.db'
    or fail("500", "Address database failure");
$to = $ENV{'HTTP_TO'};
if (! defined( $to )) {
    fail("400", "Missing Recipient");
}
```

```
$destination = $addresses{$to};

if (! defined( $destination )) {
    fail("404", "No such user");
}

print "CGI-PROXY-REQUEST-TO $destination SIP/2.0\n";
print "CGI-Reexecute-On: never\n\n";
untie %addresses; # Close db file
```

The Call Processing Language

Jonathan Lennox
Columbia University
lennox@cs.columbia.edu

May 5, 2000

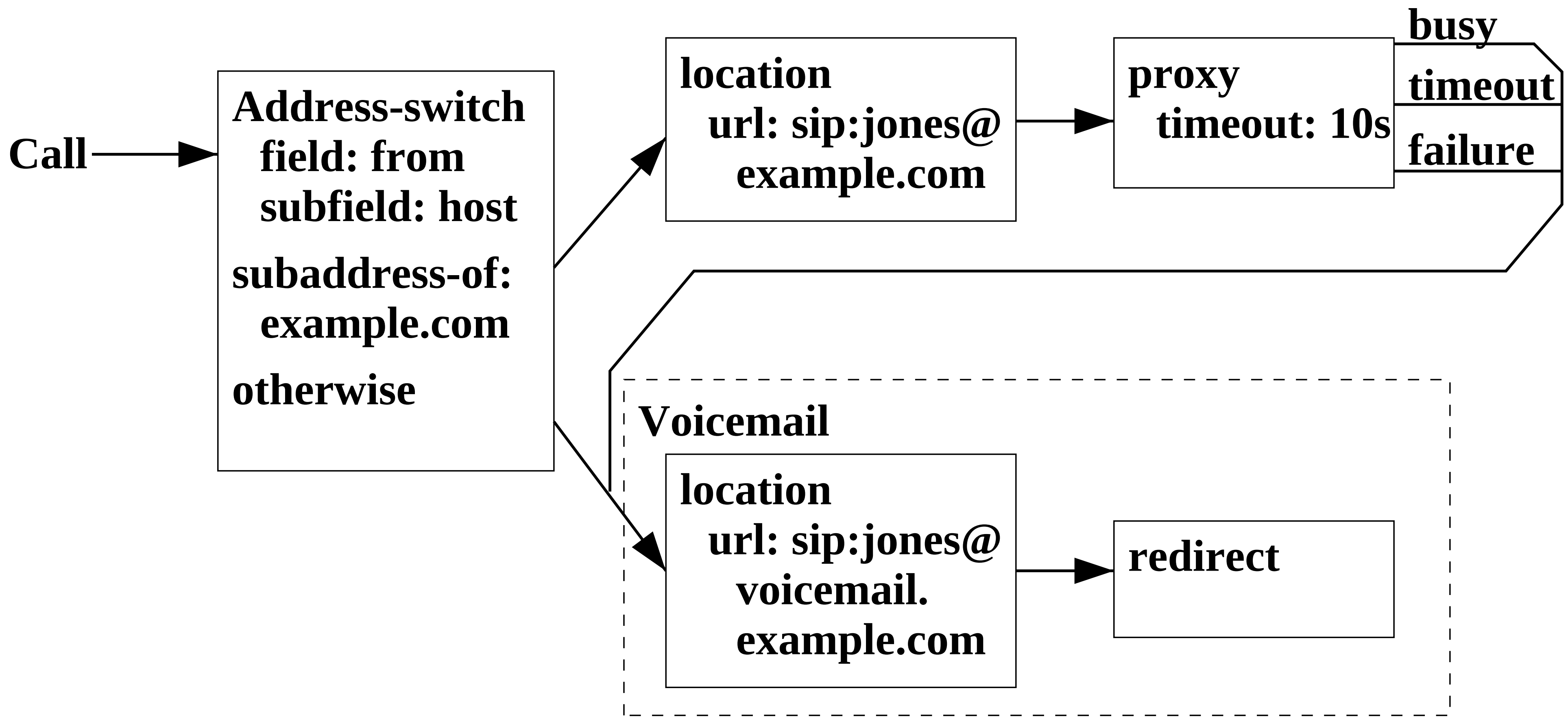
Purpose

Allow users to create simple Internet telephony services

Features:

- Creatable and editable by simple graphical tools
- Independent of signalling protocol
- Safe to run in servers

Abstract structure



Abstract structure (cont)

- Nodes and outputs — “boxes” and “arrows”
- Nodes have parameters
- Start from single root “call” node
- Progress down tree of control
- May invoke sub-actions
- Follow one output of each node, based on outcome
- Continue until we get to a node with no outputs

Textual representation

```
<cpl>
  <subaction id="voicemail">
    <location url="sip:jones@voicemail.example.com">
      <redirect />
    </location>
  </subaction>
```

Textual representation

```
  <incoming>
    <address-switch field="origin" subfield="host">
      <address subdomain-of="example.com">
        <location url="sip:jones@example.com">
          <proxy>
            <busy> <sub ref="voicemail" /> </busy>
            <noanswer> <sub ref="voicemail" /> </noanswer>
            <failure> <sub ref="voicemail" /> </failure>
          </proxy>
        </location>
      </address>
      <otherwise>
        <sub ref="voicemail" />
      </otherwise>
    </address-switch>
  </incoming>
</cpl>
```

Textual representation

- Represent scripts as XML documents
- Incoming, outgoing scripts are separate top-level tags
- Nodes and outputs are both tags
- Parameters are tag attributes
- Multiple outputs to one input represented by subactions

Switch nodes

Switch nodes make decisions.

Structure:

```
<type-switch field=var>
    <type condition1="value1">
        action1
    </type>
    <type condition2="value2">
        action2
    </type>
    <not-present>
        action3
    <otherwise>
        action4
    </otherwise>
</type-switch>
```

Address Switches: `address`

Switch based on textual strings:

is: (exact string match)

contains: substring match: only for “display”

subdomain-of: domain match: only for “host”, “tel”

Fields are “origin,” “destination,” “original-destination”, with subfields “address-type,” “user,” “host,” “port,” “tel,” “display”

String Switches: `string`

Switch based on textual strings, with conditions:

is: exact string match

contain: substring match

Fields: `subject`, `organization`, `user-agent`

Time switches: `time`

Switch based on the current time at the server.

timezone: which timezone the matching should apply in

Conditions:

- year, month, date, day, timeofday
- each condition is a list of ranges: $a_1 - b_1, a_2 - b_2, \ldots$
- must fall within a range of *all* specified conditions

Time switches: examples

```
<time month="12" date="25" year="1999">
```

December 25th, 1999, all day

```
<time month="5" date="4">
```

May 4th, every year, all day

```
<time day="1-5" timeofday="0900-1700">
```

9 AM – 5 PM, Monday through Friday, every week

Time switches: examples

```
<time timeofday="1310-1425,1440-1555,1610-1725"
   day="2,4">
```

1:10 – 2:25 PM, 2:40 – 3:55 PM, and 4:10 – 5:25 PM, Tuesdays and Thursdays, every week

```
<time date="1-7" day="1">
```

The first Monday of every month, all day

Location nodes

- A number of CPL actions (proxy, redirect) take locations
- *Location nodes* let you specify them
- These are full-featured nodes because we might want to make decisions based on outcomes of location lookups, or cascade locations
- A CPL script has an implicit global list of locations
- Location nodes can add to this list, or clear the list

Simple location nodes: `location`

Specify a location explicitly.

url: explicitly specified location

clear: clear earlier location values

Only one output; cannot fail. Don’t use an explicit output node in the URL.

Location lookup nodes: `lookup`

Specify a location abstractly, by where it should be looked up.

Parameters:

source: URL (ldap, http (CGI), etc) or non-URL source (“registration”) to search for locations

timeout: time to wait

use/ignore:
- use: caller-preferences parameters to use
- ignore: caller-preferences parameters to disregard

merge:

Outputs: `success`, `notfound`, `failure`

Location removal nodes: `remove-location`

Remove locations from the location set, based on caller preferences/callee capabilities. Has the same effect as a “Reject-Contact” header.

param: caller preference parameters to apply

value: values of parameters specified in “param”

location: caller preference location to apply

Signalling Actions: `proxy`

Proxy the call to the currently-specified set of locations, and automatically select one "best" final response.

timeout: time before giving up on the proxy attempt

recurse: recurse on redirect responses to the proxy attempt?

ordering: try location in parallel, sequential, first-only

- Outputs: `busy`, `noanswer`, `failure`
- If the proxy attempt was successful, script terminates

Signalling Actions: `redirect`

Redirect the call to the currently-specified set of locations. This has no specific parameters, and causes the script to terminate.

Signalling Actions: `reject`

Reject the call attempt. This causes the script to terminate.

status: “busy,” “notfound,” “reject,” or “error”, or a 4xx, 5xx, or 6xx code (for SIP).

reason: string explaining the failure.

Non-signalling action: `mail`

Notify a user of something through e-mail.

url: the address to contact, including any header parameters.

Non-signalling action: `log`

Store a record of the current call in a log.

name: the name of the log this should be stored

omment: a string explaining the log entry

Outputs: `success`, `failure`

Subactions

- XML syntax defines a tree; we want CPLs to be represented as directed acyclic graphs.
- *Subactions* are defined at the top level of the script, outside other actions.
- for acyclicity, top-level actions and subactions may only call subactions which were defined earlier in the script.
- Anywhere a node is expected, you can instead have a `sub` tag, with a `ref` parameter which refers to a subaction's id.

Example: Call Redirect Unconditional

```
<cpl>
  <incoming>
    <location url="sip:smith@phone.example.com">
     <redirect />
    </location>
  </incoming>
</cpl>
```

Example: Call Forward Busy/No Answer

```
<cpl>
  <subaction id="voicemail">
    <location url="sip:jones@voicemail.example.com" >
      <proxy />
    </location>
  </subaction>

  <incoming>
    <location url="sip:jones@jonespc.example.com">
       <proxy timeout="8s">
         <busy>
         </busy>
         <noanswer>
           <sub ref="voicemail" />
         </noanswer>
       </proxy>
    </location>
  </incoming>
</cpl>
```

Example: Call Screening

```
<cpl>
  <incoming>
    <address-switch field="origin" subfield="user">
      <address is="anonymous">
         <reject status="reject"
                 reason="I don’t accept anonymous calls" />
      </address>
    </address-switch>
  </incoming>
</cpl>
```

Example: Time-of-day Routing

```
<?xml version="1.0" ?>
<!DOCTYPE call SYSTEM "cpl.dtd">

<cpl>
  <incoming>
    <time-switch timezone="US/Eastern">
      <time day="1-5" timeofday="0900-1700">
        <lookup source="registration">
          <success>
            <proxy />
          </success>
        </lookup>
      </time>
      <otherwise>
        <location url="sip:jones@voicemail.example.com">
          <proxy />
        </location>
      </otherwise>
    </time-switch>
  </incoming>
</cpl>
```

Example: Non-call Actions

```
<?xml version="1.0" ?>
<!DOCTYPE call SYSTEM "cpl.dtd">

<cpl>
  <incoming>
    <lookup source="http://www.example.com/cgi-bin/locate.cgi?user=jones"
            timeout="8">
      <success>
        <proxy />
      </success>
      <failure>
        <mail url="mailto:jones@example.com&Subject=lookup%20failed" />
      </failure>
    </lookup>
  </incoming>
</cpl>
```

SIP Future

What is SIP good at?

- session setup = “out of band”
- resource location via location-independent identifier (“user@domain”, tel)
- particularly if location varies rapidly or filtering is needed (i.e., is inappropriate for DNS and LDAP)
- real-time: faster than email
- reach multiple end point simultaneously or in sequence = *forking*
- possibly hide end-point location
- delayed final answer (“ringing”) $\longleftrightarrow$ RTSP

What is SIP not meant for?

- bulk transport: media streams, files, pictures, ...
- asynchronous messaging (“email”)
- resource reservation
- high-efficiency general-purpose RPC

SIP and Corba

	SIP	Corba
data	optional fields	versioning hard
	two-level hierarchy	general, C-like
hiding	dynamic	directory-based
multiple	forking proxy	no
transport	UDP, TCP, ...	TCP
strength	inter-domain	inter-domain
generality	session set-up	RPC, events, ...

SIP servers can benefit from Corba *locally* for user location and service creation

Current SIP efforts

- SIP to Draft Standard
- QoS and security preconditions
- inter-domain AAA and billing
- session timer (liveness)
- early media (announcements)
- SIP for presence / IM
- SIP-H.323 interworking
- reliable provisional responses
- DHCP for SIP servers
- SIP for firewalls and NATs
- caller preferences
- services (transfer, multiparty calls, home)
- ISUP carriage

Other SIP Uses

- MGC $\longleftrightarrow$ MGC: SIP BCP
- PINT: establishing “legacy” phone calls
- Internet call waiting
- instant messaging and event notification

Internet telephony signaling: some open issues

- 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
- Internet emergency services

Summary and Conclusion

- SIP as flexible, extensible signaling protocol
- basic functionality + proxying: done
- extension to call control
- extension to event notification
- create *TP as basis for HTTP, SIP, RTSP, . . .

See http://www.cs.columbia.edu/sip