The Session Initiation Protocol (SIP)

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

©1998-2000, Henning Schulzrinne
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 redirect server

Internet

RTP audio

CATV

Ethernet

user agent (UA)

128.59.19.141

128.119.40.186
SIP architecture: outbound proxy

wonderland.com

alice@ph7.wonderland.com

outbound proxy

Internet

macrosf.com

REGISTER sip:macrosoft.com SIP/2.0
To: sip:bob@macrosoft.com
From: sip:bob@macrosoft.com
Contact: sip:bob@p42.macrosoft.com

 INVITE sip:bob@macrosoft.com SIP/2.0

bob@p42.macrosoft.com

INVITE sip:bob@p42.macrosoft.com SIP/2.0
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

INVITE sip:alice@wonderland.com
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 (→ 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 → 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

1. INVITE henning@columbia.edu

2. 200 OK

3. ACK hgs@play

4. INVITE hgs@play

5. 200 OK

6. ACK hgs@play

7. 200 OK

8. ACK henning@columbia.edu

9. media stream
SIP operation in redirect mode

1. INVITE henning@ieee.org
2. 302 Moved temporarily
   Contact: hgs@columbia.edu
3. INVITE hgs@columbia.edu
4. 200 OK
5. ACK henning@ieee.org
6. ACK hgs@columbia.edu

(302: redirection for single call; 301 permanently)
Basic SIP call
SIP operation in redirect mode

```
INVITE alice@ieee.org
302 Moved temporarily
ACK
INVITE a12@columbia.edu
100 Trying
180 Ringing
200 OK
ACK
```
SIP – more detail

INVITE sip:bob@b.microsoft.com SIP/2.0
To: sip:bob@b.microsoft.com
From: sip:alice@wonderland.com
Call-ID: 1234@a.wonderland.com
Cseq: 1 INVITE
Contact: sip:alice@a.wonderland.com

c=IN IP4 128.59.19.38
m=audio 3456 RTP/AVP 0

INVITE sip:carol@c.microsoft.com SIP/2.0
To: <sip:bob@b.microsoft.com>;tag=17
From: sip:alice@wonderland.com
Call-ID: 1234@a.wonderland.com
Cseq: 1 INVITE
Contact: sip:alice@a.wonderland.com

SIP/2.0 180 Ringing
Contact: sip:carol@microsoft.com

SIP/2.0 302 Moved temporarily
To: <sip:bob@b.microsoft.com>;tag=42
Contact: sip:carol@macrosoft.com

ACK sip:bob@b.microsoft.com SIP/2.0
To: sip:bob@b.microsoft.com
From: sip:alice@wonderland.com
Call-ID: 1234@a.wonderland.com
Cseq: 1 INVITE
Contact: sip:alice@a.wonderland.com

SIP/2.0 100 Trying

INVITE sip:carol@c.microsoft.com SIP/2.0
To: sip:bob@b.microsoft.com
From: sip:alice@wonderland.com
Call-ID: 1234@a.wonderland.com
Cseq: 1 INVITE
Contact: sip:alice@a.wonderland.com

c=IN IP4 128.59.19.38
m=audio 3456 RTP/AVP 0

SIP/2.0 180 Ringing

SIP/2.0 200 OK
From: sip:alice@wonderland.com
To: sip:bob@b.microsoft.com
Call-ID: 1234@a.wonderland.com
Cseq: 1 INVITE
Contact: sip:alice@a.wonderland.com

c=IN IP4 208.211.10.148
m=audio 4500 RTP/AVP 0

SIP/2.0 200 OK
From: sip:alice@wonderland.com
To: <sip:bob@b.microsoft.com>;tag=42
Call-ID: 1234@a.wonderland.com
Cseq: 1 INVITE
Contact: sip:alice@a.wonderland.com

c=IN IP4 208.211.10.148
m=audio 4500 RTP/AVP 0

SIP/2.0 200 OK

ACK sip:carol@c.microsoft.com SIP/2.0
To: sip:carol@c.microsoft.com
From: sip:alice@wonderland.com
Call-ID: 1234@a.wonderland.com
Cseq: 1 INVITE
Contact: sip:alice@a.wonderland.com

c=IN IP4 208.211.10.148
m=audio 4500 RTP/AVP 0

BYE sip:alice@a.wonderland.com SIP/2.0
Cseq: 2 BYE

SIP/2.0 200 OK
## Invitation modes

<table>
<thead>
<tr>
<th>signaling</th>
<th>media</th>
</tr>
</thead>
<tbody>
<tr>
<td>unicast</td>
<td>multicast session</td>
</tr>
<tr>
<td>unicast</td>
<td>telephony</td>
</tr>
<tr>
<td>multicast</td>
<td>reach first</td>
</tr>
<tr>
<td></td>
<td>dept. conference</td>
</tr>
</tbody>
</table>

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 $\rightarrow$ always absolute (virtual hosts)
- no extension mechanism $\rightarrow$ Require, Supported
- 8859.1 coding $\rightarrow$ 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

<table>
<thead>
<tr>
<th></th>
<th>SIP</th>
<th>Corba</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>optional fields</td>
<td>versioning hard</td>
</tr>
<tr>
<td></td>
<td>two-level hierarchy</td>
<td>general, C-like</td>
</tr>
<tr>
<td>hiding</td>
<td>dynamic</td>
<td>directory-based</td>
</tr>
<tr>
<td>multiple</td>
<td>forking proxy</td>
<td>no</td>
</tr>
<tr>
<td>transport</td>
<td>UDP, TCP, …</td>
<td>TCP</td>
</tr>
<tr>
<td>strength</td>
<td>inter-domain</td>
<td>intra-domain</td>
</tr>
<tr>
<td>generality</td>
<td>session set-up</td>
<td>RPC, events, …</td>
</tr>
</tbody>
</table>
SIP Details
SIP syntax

**request**

```
method URL  SIP/2.0
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"
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
```
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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVITE</td>
<td>initiate call</td>
</tr>
<tr>
<td>ACK</td>
<td>confirm final response</td>
</tr>
<tr>
<td>BYE</td>
<td>terminate (and transfer) call</td>
</tr>
<tr>
<td>CANCEL</td>
<td>cancel searches and “ringing”</td>
</tr>
<tr>
<td>OPTIONS</td>
<td>features support by other side</td>
</tr>
<tr>
<td>REGISTER</td>
<td>register with location service</td>
</tr>
<tr>
<td>INFO</td>
<td>mid-call information (ISUP, DTMF)</td>
</tr>
<tr>
<td>COMET</td>
<td>precondition met</td>
</tr>
<tr>
<td>PRACK</td>
<td>provisional acknowledgement</td>
</tr>
<tr>
<td>SUBSCRIBE</td>
<td>subscribe to event</td>
</tr>
<tr>
<td>NOTIFY</td>
<td>notify subscribers</td>
</tr>
</tbody>
</table>
### SIP response codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1xx</td>
<td>provisional</td>
</tr>
<tr>
<td>100</td>
<td>continue</td>
</tr>
<tr>
<td>180</td>
<td>ringing</td>
</tr>
<tr>
<td>2xx</td>
<td>success</td>
</tr>
<tr>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>3xx</td>
<td>redirect</td>
</tr>
<tr>
<td>300</td>
<td>multiple choices</td>
</tr>
<tr>
<td>301</td>
<td>moved permanently</td>
</tr>
<tr>
<td>302</td>
<td>moved temporarily</td>
</tr>
</tbody>
</table>
### SIP response codes

<table>
<thead>
<tr>
<th>4xx</th>
<th>client error</th>
<th>480</th>
<th>temporarily unavailable</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>bad request</td>
<td>481</td>
<td>call leg doesn’t exist</td>
</tr>
<tr>
<td>401</td>
<td>unauthorized</td>
<td>482</td>
<td>loop detected</td>
</tr>
<tr>
<td>403</td>
<td>forbidden</td>
<td>483</td>
<td>too many hops</td>
</tr>
<tr>
<td>404</td>
<td>not found</td>
<td>484</td>
<td>address incomplete</td>
</tr>
<tr>
<td>407</td>
<td>proxy auth. required</td>
<td>485</td>
<td>ambiguous</td>
</tr>
<tr>
<td>408</td>
<td>request timeout</td>
<td>486</td>
<td>busy here</td>
</tr>
<tr>
<td>420</td>
<td>bad extension</td>
<td>487</td>
<td>request cancelled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>488</td>
<td>not acceptable</td>
</tr>
</tbody>
</table>
## SIP response codes

<table>
<thead>
<tr>
<th>5xx</th>
<th>server error</th>
<th>6xx</th>
<th>global failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>server internal error</td>
<td>600</td>
<td>busy</td>
</tr>
<tr>
<td>501</td>
<td>not implemented</td>
<td>601</td>
<td>decline</td>
</tr>
<tr>
<td>502</td>
<td>bad gateway</td>
<td>604</td>
<td>does not exist</td>
</tr>
<tr>
<td>503</td>
<td>service unavailable</td>
<td>606</td>
<td>not acceptable</td>
</tr>
<tr>
<td>504</td>
<td>gateway time-out</td>
<td></td>
<td></td>
</tr>
<tr>
<td>505</td>
<td>version not supported</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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

```plaintext
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

INVITE sip:kelly@p4711.sales.acme.com SIP/2.0  
To: sip:info@acme.com  
From: sip:alice@wonderland.com  
Via: sales.acme.com;branch=+h(info,alice,17,1,kelly@sales)  
Via: acme.com;branch=+h(info,alice,17,1,kelly@acme)  
Via: acme.com;branch=+h(info,alice,17,1,bob@sales)  
Via: acme.com;branch=+h(info,alice,17,1,info)  
Via: ph123.wonderland.com

INVITE sip:bob@sales.acme.com SIP/2.0  
To: sip:info@acme.com  
From: sip:alice@wonderland.com  
Via: acme.com;branch=+h(info,alice,17,1,info)  
Via: ph123.wonderland.com

INVITE sip:kelly@sales.acme.com SIP/2.0  
To: sip:info@acme.com  
From: sip:alice@wonderland.com  
Via: acme.com;branch=+h(info,alice,17,1,kelly@acme)  
Via: sales.acme.com;branch=+h(info,alice,17,1,kelly@sales)  
Via: acme.com;branch=+h(info,alice,17,1,info)  
Via: ph123.wonderland.com

INVITE sip:kelly@p4711.sales.acme.com SIP/2.0  
To: sip:info@acme.com  
From: sip:alice@wonderland.com  
Via: sales.acme.com;branch=+h(info,alice,17,1,kelly@sales)  
Via: acme.com;branch=+h(info,alice,17,1,bob@sales)  
Via: acme.com;branch=+h(info,alice,17,1,info)  
Via: ph123.wonderland.com

INVITE sip:alice@wonderland.com SIP/2.0  
To: sip:info@acme.com  
From: sip:alice@wonderland.com  
Call-ID: 17  
CSeq: 1 INVITE  
Via: ph123.wonderland.com

INVITE sip:bob@sales.acme.com SIP/2.0  
To: sip:info@acme.com  
From: sip:alice@wonderland.com  
Via: acme.com;branch=+h(info,alice,17,1,bob@sales)  
Via: sales.acme.com;branch=+h(info,alice,17,1,info)  
Via: ph123.wonderland.com

branch=+h(To,Call-ID,CSeq,URL)
Forcing request paths

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

A

INVITE PB
Contact: A

200 OK
Contact: B
Record-Route: PB, PA

ACK PA
Route: PB,B

BYE A
Route: A;maddr=PB

PA

INVITE PB
Contact: A
Record-Route: PA

200 OK
Contact: B
Record-Route: PB, PA

ACK PB
Route: B

BYE A
Route: A;maddr=PA

PB

INVITE B
Contact: A
Record-Route: PB, PA

200 OK
Contact: B
Record-Route: PB, PA

ACK B
Route: PB,B

BYE A
Route: A;maddr=PB

B
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

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject</td>
<td>topic of call, short message</td>
</tr>
<tr>
<td>Organization</td>
<td>caller and callee, possibly filled in by proxy</td>
</tr>
<tr>
<td>Date</td>
<td>date of call (replay prevention)</td>
</tr>
<tr>
<td>Server</td>
<td>make and model of server</td>
</tr>
<tr>
<td>User-Agent</td>
<td>make and model of client</td>
</tr>
<tr>
<td>Accept-Language</td>
<td>human languages preferred</td>
</tr>
<tr>
<td>Priority</td>
<td>call priority (normal, urgent, ...)</td>
</tr>
<tr>
<td>In-Reply-To</td>
<td>reference to earlier call-id</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>component</th>
<th>full</th>
<th>compact</th>
</tr>
</thead>
<tbody>
<tr>
<td>headers, CRLF</td>
<td>71</td>
<td>35</td>
</tr>
<tr>
<td>body (SDP)</td>
<td>120</td>
<td>120</td>
</tr>
<tr>
<td>addresses (4)</td>
<td>96</td>
<td>96</td>
</tr>
<tr>
<td>other</td>
<td>72</td>
<td>72</td>
</tr>
<tr>
<td><strong>sum</strong></td>
<td><strong>359</strong></td>
<td><strong>323</strong></td>
</tr>
</tbody>
</table>
SIP message size

- \[ \Sigma \text{INVITE}, 100, 200, \text{ACK}, \text{BYE}, 200 \approx 1500 \text{ bytes} \]
- \[ \equiv 1.5 \text{ s of 8 kb/s voice} \]
- 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 \( 0.5, 1, 2, 4, 4, \ldots \) seconds
- … until provisional (1xx) response
- then with interval 4 seconds
SIP reliability: INVITE

- retransmit request after 0.5, 1, 2, 4, 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

- Initial
  - INVITE
  - $T1*2^n$ INVITE
  - Calling
    - 1xx
    - Status ACK
    - Call proceeding
      - 1xx
      - Status ACK
    - Completed
      - Status ACK

Event:
- Request sent
- 7 INVITE sent
Reliability for provisional responses
Interaction with resource reservation

avoid “fast busy” after ringing ➔ interleave
SIP Registration and User Location
• 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="df84f1ce4341ae6cbe5a359a9c8e88",
  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

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
<td>0</td>
</tr>
<tr>
<td>o</td>
<td>root 2890844527 2890844527</td>
</tr>
<tr>
<td>s</td>
<td>the subject of the call</td>
</tr>
<tr>
<td>c</td>
<td>IN IP4 128.59.16.1</td>
</tr>
<tr>
<td>t</td>
<td>0 0</td>
</tr>
<tr>
<td>m</td>
<td>audio 3456 RTP/AVP 0 97</td>
</tr>
<tr>
<td>a</td>
<td>rtpmap:0 PCMU/8000</td>
</tr>
<tr>
<td>a</td>
<td>rtpmap:97 G723/8000</td>
</tr>
<tr>
<td>m</td>
<td>video 4180 RTP/AVP 98</td>
</tr>
<tr>
<td>a</td>
<td>rtpmap:98 H263/90000</td>
</tr>
<tr>
<td>c</td>
<td>IN IP4 128.59.16.2</td>
</tr>
</tbody>
</table>

**Session ID**: v

**Version**: o

**Subject of the Call**: s

**Creator**: c

**Port**: t

**Audio**: m

**RTP Payload Type**: a

**RTP Format and Clock Rate**: a

**Destination Address**: c
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

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

depend-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="wf84f1ceczx41ae6cbe5aea9c8e88d359",
  opaque="42", stale="FALSE", algorithm="MD5"

- A tries again:

  INVITE sip:UserB@ssl.wcom.com SIP/2.0
  Authorization: Digest username="UserA", realm="GW service",
  nonce="wf84f1ceczx41ae6cbe5aea9c8e88d359",
  opaque="42", uri="sip:UserB@ssl.wcom.com",
  response="42ce3cefa4b22f50c6a6071bc8"
**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(A₁), nonce-value : H(A₂))
- for MD5: H(H(A₁) : nonce-value : H(A₂))
- where A₁ = username : realm : passwd
- A₂ = 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” → 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 \( \leq \) 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

<table>
<thead>
<tr>
<th></th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>q</td>
<td>location preference</td>
</tr>
<tr>
<td>class</td>
<td>business, residence</td>
</tr>
<tr>
<td>description</td>
<td>show to caller</td>
</tr>
<tr>
<td>duplex</td>
<td>full or half-duplex</td>
</tr>
<tr>
<td>feature</td>
<td>call handling features</td>
</tr>
<tr>
<td>language</td>
<td>languages spoken</td>
</tr>
<tr>
<td>media</td>
<td>audio, video, text/numeric, ...</td>
</tr>
<tr>
<td>mobility</td>
<td>fixed or mobile</td>
</tr>
<tr>
<td>priority</td>
<td>“only in case of emergency”</td>
</tr>
<tr>
<td>scheme</td>
<td>URL schemes (tel, http, ...)</td>
</tr>
<tr>
<td>service</td>
<td>IP, PSTN, ISDN, pager, ...</td>
</tr>
</tbody>
</table>
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
* Proposed Standard, Feb. 1999 – RFC2543

* bakeoffs every 4 months → cross-vendor interoperability tests

<table>
<thead>
<tr>
<th>host</th>
<th>when</th>
<th>companies</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Columbia University</td>
<td>April 1999</td>
<td>16</td>
</tr>
<tr>
<td>2 pulver.com</td>
<td>August 1999</td>
<td>15</td>
</tr>
<tr>
<td>3 Ericsson</td>
<td>December 1999</td>
<td>26</td>
</tr>
<tr>
<td>4 3Com</td>
<td>April 2000</td>
<td>36</td>
</tr>
<tr>
<td>5 pulver.com</td>
<td>August 2000</td>
<td></td>
</tr>
<tr>
<td>6 Sylantro</td>
<td>December 2000</td>
<td></td>
</tr>
<tr>
<td>7 ETSI</td>
<td>April 2001</td>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Company</th>
</tr>
</thead>
<tbody>
<tr>
<td>3Com</td>
</tr>
<tr>
<td>AudioTalk Networks</td>
</tr>
<tr>
<td>Broadsoft</td>
</tr>
<tr>
<td>Catapult</td>
</tr>
<tr>
<td>Cisco</td>
</tr>
<tr>
<td>Carnegie-Mellon University</td>
</tr>
<tr>
<td>Columbia University</td>
</tr>
<tr>
<td>Delta Information Systems</td>
</tr>
<tr>
<td>dynamicsoft</td>
</tr>
<tr>
<td>Ellemtel</td>
</tr>
<tr>
<td>Ericsson</td>
</tr>
<tr>
<td>Hewlett-Packard</td>
</tr>
<tr>
<td>Hughes Software Systems</td>
</tr>
<tr>
<td>Indigo Software</td>
</tr>
<tr>
<td>Iwatsu Electric</td>
</tr>
<tr>
<td>Komodo</td>
</tr>
<tr>
<td>Lucent</td>
</tr>
<tr>
<td>MCI Worldcom</td>
</tr>
<tr>
<td>Mediatrix</td>
</tr>
<tr>
<td>Microappliances</td>
</tr>
<tr>
<td>Netergy</td>
</tr>
<tr>
<td>Netspeak</td>
</tr>
<tr>
<td>Nokia</td>
</tr>
<tr>
<td>ObjectSoftware</td>
</tr>
<tr>
<td>Nortel</td>
</tr>
<tr>
<td>Nuera</td>
</tr>
<tr>
<td>Pingtel</td>
</tr>
<tr>
<td>RaveTel</td>
</tr>
<tr>
<td>Siemens</td>
</tr>
<tr>
<td>Telogy</td>
</tr>
<tr>
<td>Ubiquity</td>
</tr>
<tr>
<td>Ubiquity</td>
</tr>
<tr>
<td>Vegastream</td>
</tr>
<tr>
<td>Vovida</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>functionality</th>
<th>C lines (≈)</th>
</tr>
</thead>
<tbody>
<tr>
<td>generic RFC822-style parser</td>
<td>500</td>
</tr>
<tr>
<td>HTTP generic headers</td>
<td>330</td>
</tr>
<tr>
<td>SIP, RTSP</td>
<td>300</td>
</tr>
</tbody>
</table>
“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
failure

busy

location
url: sip:jones@voicemail.
exmaple.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, microprocessor (ARM) for signaling + DSP for speech coding, echo cancellation
Columbia e*phone

- DSP for voice coding and signaling \(\Rightarrow\) 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, \ldots
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 requests
- 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:

request

response (Location)
Parallel search with CANCEL
Sequential search
IN call forwarding features

SIP can implement intelligent network features:

<table>
<thead>
<tr>
<th>name</th>
<th>feature</th>
<th>SIP note</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCF</td>
<td>selective call forwarding</td>
<td>302, Contact</td>
</tr>
<tr>
<td>SCR</td>
<td>selective call reject</td>
<td>302, Contact</td>
</tr>
<tr>
<td>CFU</td>
<td>call-forwarding unconditional</td>
<td>302, Contact</td>
</tr>
<tr>
<td>CFB</td>
<td>call-forwarding busy</td>
<td>302, Contact</td>
</tr>
<tr>
<td>CFNR, CFDA</td>
<td>call forwarding, no response</td>
<td>302, Contact</td>
</tr>
<tr>
<td>DND</td>
<td>call forwarding to voice mail</td>
<td>302, Contact</td>
</tr>
</tbody>
</table>

⚠️ differences as server program or in end system
## IN call handling features

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

1. A sends INVITE to B
2. B sends INVITE to C
3. D sends INVITE to C

Also: A, B

INVITE
BYE
session
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: \textit{Alexander.G.Bell} $\rightarrow$ \textit{agb}
- list aliases
- active badges
- \texttt{REGISTER} announces location, with time limit
- \texttt{REGISTER} + \texttt{Contact} sets new location
Interaction with directory services

- LDAP (with dynamic extensions)
- rwhois
- whois++ (RFC 1913)
- possibly implement SIP interface \(\rightarrow\) 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

telemarketer

INVITE (Also:)

BYE (Also:)

2(C)

4

3

1

5

auto-dialer

customer
SIP and H.323
## SIP – H.323 Comparison

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

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 → 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) → 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

- Caller
- Gatekeeper
- Callee

**RAS**

- SETUP
- CALL PROCEEDING
- ALERTING
- CONNECT

Audio:

- Port 1720, 1300
- 2 dynamic ports
## Services

<table>
<thead>
<tr>
<th>Service</th>
<th>H.323</th>
<th>SIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call transfer</td>
<td>H.450.2</td>
<td>“30x”</td>
</tr>
<tr>
<td>Call diversion</td>
<td>H.450.3</td>
<td>“30x”</td>
</tr>
<tr>
<td>Call hold</td>
<td>H.450.4</td>
<td>SDP-based</td>
</tr>
<tr>
<td>Call park</td>
<td>H.450.5</td>
<td>REGISTER</td>
</tr>
<tr>
<td>Call waiting</td>
<td>H.450.6</td>
<td>INVITE</td>
</tr>
<tr>
<td>Message waiting</td>
<td>H.450.7</td>
<td>email, NOTIFY</td>
</tr>
<tr>
<td>Call forward busy</td>
<td>H.450.9</td>
<td>“30x”</td>
</tr>
</tbody>
</table>
**H.323 vs. SIP: Basic Call Control**

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

<table>
<thead>
<tr>
<th>Service</th>
<th>H.323v1</th>
<th>v2</th>
<th>v3</th>
<th>SIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call holding</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Call transfer</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Call forwarding</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Call waiting</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>
## H.323 vs. SIP: Advanced Features

<table>
<thead>
<tr>
<th>Service</th>
<th>H.323v1</th>
<th>v2</th>
<th>v3</th>
<th>SIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Third party control</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Conference</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Click-to-dial</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>PINT</td>
</tr>
<tr>
<td>Capability exchange</td>
<td>better</td>
<td>better</td>
<td>better</td>
<td>yes</td>
</tr>
<tr>
<td>HTML transport</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Call forking</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>
### H.323 vs. SIP: Quality of Service

<table>
<thead>
<tr>
<th></th>
<th>H.323v1</th>
<th>v2</th>
<th>v3</th>
<th>SIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call setup delay</td>
<td>6-7 RTT</td>
<td>3-4</td>
<td>1.5-2.5</td>
<td>1.5</td>
</tr>
<tr>
<td>Loss recovery</td>
<td>TCP</td>
<td>TCP</td>
<td>better</td>
<td>better</td>
</tr>
<tr>
<td>Fault detection</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Mid-call failure</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td>live</td>
</tr>
<tr>
<td>Registrar failure</td>
<td>fail</td>
<td>fail</td>
<td>backup</td>
<td>multicast</td>
</tr>
<tr>
<td>GK/Proxy redundancy</td>
<td>no</td>
<td>no</td>
<td>backup</td>
<td>SLP, DNS, DHCP</td>
</tr>
<tr>
<td>Loop detection</td>
<td>no</td>
<td>no</td>
<td>PathValue</td>
<td>Via, hops, time</td>
</tr>
</tbody>
</table>
### H.323 vs. SIP: Manageability

<table>
<thead>
<tr>
<th></th>
<th>H.323v1</th>
<th>v2</th>
<th>v3</th>
<th>SIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admission control</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no (RSVP)</td>
</tr>
<tr>
<td>Policy control</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>ob proxy</td>
</tr>
<tr>
<td>Resource reservation</td>
<td>local</td>
<td>local</td>
<td>local</td>
<td>no (RSVP)</td>
</tr>
</tbody>
</table>
## H.323 vs. SIP: Scalability

<table>
<thead>
<tr>
<th></th>
<th>H.323v1</th>
<th>v2</th>
<th>v3</th>
<th>SIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complexity</td>
<td>more</td>
<td>more</td>
<td>more+</td>
<td>less</td>
</tr>
<tr>
<td>Server processing</td>
<td>SF</td>
<td>SF</td>
<td>SF/SL, TSF</td>
<td>SL, TSF/TSL</td>
</tr>
<tr>
<td>Inter-server</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

TS: transaction state; SF: call statefull; SL: call stateless
### H.323 vs. SIP: Flexibility

<table>
<thead>
<tr>
<th></th>
<th>H.323v1</th>
<th>v2</th>
<th>v3</th>
<th>SIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transport protocols</td>
<td>TCP</td>
<td>TCP</td>
<td>TCP/UDP</td>
<td>any</td>
</tr>
<tr>
<td>Extensibility</td>
<td>unlabeled vendor extensions</td>
<td>IANA, labeled</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customization</td>
<td>harder</td>
<td></td>
<td></td>
<td>easier</td>
</tr>
<tr>
<td>Version compatibility</td>
<td>N/A</td>
<td>yes</td>
<td>yes</td>
<td>N/A</td>
</tr>
<tr>
<td>SCN interoperability</td>
<td>good</td>
<td>good</td>
<td>good</td>
<td>TBD</td>
</tr>
<tr>
<td>Protocol encoding</td>
<td>binary (ASN.1, Q.931)</td>
<td>text</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Mobility Support Using SIP

Overview

pure-IP mobility ↔ 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

alice17@yahoo.com
alice@columbia.edu
7000@columbia.edu
Alice.McBeal@columbia.edu
yahoo.com
columbia.edu

tel:12128541111
tel:12015551234

alice@columbia.edu
(also used by bob@columbia.edu)
alice@host.columbia.edu
**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
DHCP server

sip.hotelca.com

home.com

eagles@home.com

SIP: sip.hotelca.com
DNS: hotelca.com
IP: 128.59.16.1

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

→ 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 → 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 \( n \cdot 10 \text{ bytes} \)

- 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

1. SIP INVITE
2. SIP OK
3. data
SIP Mobility: Multi-stage Registration

Don’t want to bother home registrar with each move

San Francisco

Los Angeles

REGISTER

INVITE
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

Handoff interval

MH | BS | DHCP | CH

beacon

Discover

Offer

Request

Ack

beacon interval

INVITE

200

Handoff interval
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

<table>
<thead>
<tr>
<th></th>
<th>safety</th>
<th>language?</th>
<th>party?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIP-cgi</td>
<td>same as scripting</td>
<td>any</td>
<td>callee</td>
</tr>
<tr>
<td>servlets</td>
<td>same as Java</td>
<td>Java</td>
<td>callee</td>
</tr>
<tr>
<td>CPL</td>
<td>very</td>
<td>XML</td>
<td>both</td>
</tr>
<tr>
<td>applets</td>
<td>same as Java</td>
<td>Java</td>
<td>caller</td>
</tr>
</tbody>
</table>
Programming services

- “caller proposes, callee disposes, administrator decides”
- web = static pages → cgi-bin → 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

Call

Address-switch
field: from
subfield: host
subaddress-of:
example.com
otherwise

location
url: sip:jones@
example.com

proxy
timeout: 10s

busy
timeout
failure

Voicemail

location
url: sip:jones@
voicemail.
example.com

redirect
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>
</cpl>
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>
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:

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

<time month="5" date="4">
  May 4th, every year, all day
</time>

<time day="1-5" timeofday="0900-1700">
  9 AM – 5 PM, Monday through Friday, every week
</time>
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>

<time date="1-7" day="1">
  The first Monday of every month, all day
</time>
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

**comment:** 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 acycliccility, 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

```xml
<cp1>
  <incoming>
    <location url="sip:smith@phone.example.com">
      <redirect />
    </location>
  </incoming>
</cp1>
```
Example: Call Forward Busy/No Answer

```xml
<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
<?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
<?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") \(\rightarrow\) 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

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

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 ⟷ 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