The Session Initiation Protocol (SIP)

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

©1998-1999, Henning Schulzrinne
Overview

- proxy and redirect modes
- registration
- reliability
- forking and loop detection
- forced routing
- hold and mute
- authentication and anonymity
- third-party call control
SIP Basics
SIP: Session Initiation Protocol

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

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

January 18, 2000
• terminate and transfer calls

• web security, cookies
SIP Features, cont’d.

- 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 or TCP (or AAL5 or X.25)
- easily extends to presence information (“buddy lists”) and event notification
SIP for H.323 experts

H.323        SIP + SDP
H.225.0 + RAS SIP
H.245        SDP, SMIL, ...
gatekeeper   proxy
SIP architecture: peer-to-peer

SIP redirect server

Internet

RTP audio

CATV Ethernet

128.59.19.141

128.119.40.186

January 18, 2000
SIP architecture: carrier

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

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

1. use directory service (e.g., LDAP) to map name to user@domain
2. locate SIP servers using DNS SRV, CNAME
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
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

January 18, 2000
SIP operation in proxy mode
SIP

January 18, 2000

media stream
SIP operation in redirect mode

1. INVITE henning@ieee.org
2. 302 Moved temporarily
   Contact: hgs@columbia.edu
3. ACK henning@ieee.org
4. Invited to columbia.edu
5. INVITE hgs@columbia.edu
6. 200 OK
7. ACK hgs@columbia.edu
8. hgs
(302: redirection for single call; 301 permanently)
SIP operation in redirect mode

INVITE alice@ieee.org

302 Moved temporarily

ACK

INVITE a12@columbia.edu

100 Trying

180 Ringing

200 OK

UAC

redirect server

iae.org

UAS

a12@columbia.edu
Interaction with resource reservation

avoid “fast busy” after ringing ➔ interleave

INVITE

YESSIR?

RR

SR

reserve
(no traffic - no charge)

200

ACK or BYE
## Invitation modes

<table>
<thead>
<tr>
<th>invitation</th>
<th>conference</th>
</tr>
</thead>
<tbody>
<tr>
<td>unicast</td>
<td>multicast</td>
</tr>
<tr>
<td>unicast</td>
<td>telephony</td>
</tr>
<tr>
<td>multicast</td>
<td>reach first</td>
</tr>
</tbody>
</table>

SIP for all modes, SAP also for multicast/multicast
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

often combine registrar + proxy or redirect server
Proxy and redirect servers

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

- *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

session description
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**:
- 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, 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: \(\rightarrow\) *potentially* replace *all* upper-layer Internet protocols

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

January 18, 2000
SIP Details
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
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

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>bad request</td>
</tr>
<tr>
<td>401</td>
<td>unauthorized</td>
</tr>
<tr>
<td>403</td>
<td>forbidden</td>
</tr>
<tr>
<td>408</td>
<td>request timeout</td>
</tr>
<tr>
<td>480</td>
<td>unavailable</td>
</tr>
<tr>
<td>481</td>
<td>invalid Call-ID</td>
</tr>
<tr>
<td>482</td>
<td>loop detected</td>
</tr>
</tbody>
</table>
SIP response codes
<table>
<thead>
<tr>
<th>5xx</th>
<th>server error</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>server internal error</td>
</tr>
<tr>
<td>501</td>
<td>not implemented</td>
</tr>
<tr>
<td>502</td>
<td>bad gateway</td>
</tr>
<tr>
<td>503</td>
<td>service unavailable</td>
</tr>
<tr>
<td>504</td>
<td>gateway time-out</td>
</tr>
<tr>
<td>505</td>
<td>version not supported</td>
</tr>
<tr>
<td>6xx</td>
<td>global failure</td>
</tr>
<tr>
<td>600</td>
<td>busy</td>
</tr>
<tr>
<td>601</td>
<td>decline</td>
</tr>
<tr>
<td>604</td>
<td>does not exist</td>
</tr>
<tr>
<td>606</td>
<td>not acceptable</td>
</tr>
</tbody>
</table>
REGISTER

- on startup, send registration to sip.mcast.net via multicast
- registrations expire – determined by server
- returns list of current registrations
- registrations may be proxied ➤ mobility

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

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="Th, Aug 13 1998 10:15:65 GMT"
Contact: mailto:me@home.edu
  ;q=0.3;expires="Mo, Jan 1 2000"
Extended Contact header

for REGISTER:

- q: location preference
- expires: expiration time/date
- 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”
- service: IP, PSTN, ISDN, pager, …

January 18, 2000
**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
Headers: call and request identification

To, From: logical call destination, source ("call leg")

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

CSeq: request within call leg
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
- UAS copies Via headers to response
- on receipt, make sure it’s own address
- 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
- **Via** list of locations
- **Max-Forwards** limits number of hops
- **Expires** limits search time
<table>
<thead>
<tr>
<th><strong>Call and caller identification</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Subject</strong></td>
</tr>
<tr>
<td><strong>Organization</strong></td>
</tr>
<tr>
<td><strong>Date</strong></td>
</tr>
<tr>
<td><strong>Server</strong></td>
</tr>
<tr>
<td><strong>User-Agent</strong></td>
</tr>
<tr>
<td><strong>Accept-Language</strong></td>
</tr>
<tr>
<td><strong>Priority</strong></td>
</tr>
</tbody>
</table>
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
- client can ask via OPTIONS header ➔ Allow list
SIP extensions and feature negotiation

- receiver ignores headers, parameters it doesn’t understand
- if crucial, mark with \texttt{``Require: feature``}

C$\rightarrow$S: \texttt{INVITE sip:watson@bell-telephone.com SIP/2.0}
\texttt{Require: com.example.billing}
\texttt{Payment: sheep_skins, conch_shells}

S$\rightarrow$C: \texttt{SIP/2.0 420 Bad Extension}
\texttt{Unsupported: com.example.billing}

- methods: on failure (506), indicate via \texttt{Allow}
- inquire about capabilities: \texttt{OPTIONS \rightarrow Allow}, possibly supported media types
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 ≤ 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 → S: INVITE
2. S → C: 100, user location, ringing, …
3. S → C: 200
4. C → S: ACK

- server repeats final response (as above) if no ACK
SIP state transition – server

- Initial
- INVITE 1xx
- CANCEL 200
- INVITE 1xx
- status change 1xx
- CANCEL 200
- INVITE status
- Failure
- INVITE status
- Success
- max(T1*2^n, T2)
- Status
- 32s
- ACK
- Confirmed
- message sent
- event
- ACK
- CANCEL 200

January 18, 2000
SIP state transition – client

- Initial
  - INVITE
  - 7 INVITE sent
  - T1*2^n
    - INVITE
  - 1xx
  - Call proceeding
  - 1xx
  - Completed
  - ACK
  - status
  - ACK
  - status
  - ACK
  - give up
  - BYE
  - give up
  - BYE

32 s (for proxy)

January 18, 2000
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

- 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

January 18, 2000
• future: MIME, e.g., 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

v=0
o=g.bell 877283459 877283519 IN IP4 132.151.1.19
c=IN IP4 132.151.1.19
b=CT:64
t=3086272736 0
m=audio 3456 RTP/AVP 96
a=rtpmap:96 VDVI/8000/1
m=video 3458 RTP/AVP 31
m=application 32416 udp wb
a=orient:portrait
SIP Security, Authentication and Privacy
Security

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

proxy authentication: Proxy-Authentication

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

end-to-end cryptographic: PGP, S/MIME – as filter

also: anonymous calls
SIP authentication

**Basic:** include plain-text password in request, immediately or after 401 (Unauthorized) 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
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
SIP firewalls

• act as standard SIP proxy

• much easier than H.323:
  – single protocol vs. H.225.0 + H.245
  – SDP ≪ 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

January 18, 2000
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
- **SUBSCRIBE** to events
- 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:

1. INV U@P1
   - 200 (H1)
   - 200 (H2)

2. INV U@P2
   - 200 (H1)

3. INV U@H1
   - 200 (H1)

4. INV U@P3
   - 200 (H2)

5. INV U@H2
   - 200 (H2)

6. INV U@H2
   - 482

request

response (Location)
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</td>
</tr>
<tr>
<td>OCS</td>
<td>outgoing call screening</td>
<td>firewall?</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>

January 18, 2000
SIP advanced services

- Also for third-party control: A asks B to send request to C
- generic establishment of call legs
- Call-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
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
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
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

January 18, 2000
SIP user location

- SIP is independent of mechanism to locate user

- examples:
  - local multicast of invitation
  - login-based via NFS
  - recursive “finger”-traversal
  - name translation: Alexander.G.Bell ➞ agb
  - active badges
  - SIP:
    * REGISTER announces location, with time limit
    * REGISTER + Contact sets new location
    * forwarding within host (≠ standard port)
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, Call-Disposition: First ➤ 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

A 2(C) 4 T 3

auto-dialer

C 1

customer

INVITE(Also:)

BYE(Also:)

January 18, 2000
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
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

- 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 INVITE
- MH updates home SIP server
- optimization: hierarchical LR (later)
SIP Mobility: Mid-call

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

January 18, 2000
SIP Mobility: Multi-stage Registration

Don’t want to bother home registrar with each move

San Francisco

Los Angeles

Contact: alice@CA
From: alice@NY
Contact: 193.1.1.1

REGISTER
INVITE

CA

NY

From: alice@NY
Contact: alice@CA
From: alice@NY
Contact: 192.1.2.3

January 18, 2000
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

MH

beacon

Discover

Offer

Request

Ack

BS

beacon interval

DHCP

INVITE

200

CH

handoff interval

January 18, 2000
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
Internet phone “appliance”

- phone = $49.99; PC > $600 (GPF included)

- Ethernet phone ➤ no PBX for switching

- examples: 3Com NBX, Columbia University, e-tel, Nortel, Pingtel, Symbol Technologies (wireless), …

- Columbia e*phone:
  - DSP for voice coding ➤ limited memory (256 kB SRAM!)
  - implemented minimal IP stack (IP/UDP/RTP, DHCP, SIP, tftp, DNS)
  - no TCP
  - MP3 radio

January 18, 2000
– sensor interfaces to the world: chair, IR, temperature, ...
SIP Scripting
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
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”
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?
The Call Processing Language

Jonathan Lennox
Columbia University
lennox@cs.columbia.edu

IRT Group talk
March 3, 1999
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

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

January 18, 2000
Abstract Structure (cont)

- Nodes and outputs — “boxes” and “arrows”
- Nodes have parameters
- Start from single root “call” node
- Progress down acyclic graph
- Follow one output of each node, based on outcome
- Continue until we get to a node with no outputs
<?xml version="1.0" ?>
<!DOCTYPE call SYSTEM "cpl.dtd">

<call>
  <string-switch field="from">
    <string matches="*@example.com">
      <location url="sip:jones@example.com">
        <proxy>
          <busy> <link ref="voicemail" /> </busy>
          <noanswer> <link ref="voicemail" /> </noanswer>
          <failure> <link ref="voicemail" /> </failure>
        </proxy>
      </location>
    </string>
  </otherwise>
  <location url="sip:jones@voicemail.example.com" merge="clear" id="voicemail">
    <redirect />
  </location>
</string-switch>
</call>
Textual Representation (cont)

- Represent scripts as XML documents
- Nodes and outputs are both tags
- Parameters are tag attributes
- Multiple outputs to one input represented by link tags
Switch nodes

Switch nodes make decisions.

Structure:

```xml
<type-switch field=var>
  <type condition1="value1">
    -action1-
  </type>
  <type condition2="value2">
    -action2-
  </type>
  <otherwise>
    -action3-
  </otherwise>
</type-switch>
```
String Switches

Switch based on textual strings.

- type: “string”

- conditions: one of
  - “is” (exact string match)
  - “contains” (substring match)
  - “matches” (glob match)

- Possible fields:
  - “to,” “from” — all servers
  - “request-uri,” “subject,” “organization,” “priority,” “display-from,”, “display-to” — SIP servers

January 18, 2000
– ??? — H.323 servers
Time switches

Switch based on the current time at the server.

- type: “time”

- 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

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 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

Specify a location explicitly.

- Name: “location”
- Parameter: “url”: explicitly specified location
- Parameter: “merge”
- Only one output; cannot fail. Don’t use an explicit output node in the URL.
Location lookup nodes

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

- Name: “lookup”

- Parameter: one of
  - “url”: Gives URL of where to search for locations — ldap, http (CGI), etc.
  - “source”: Non-URL source of locations — currently only “registration”

- Parameter: timeout
  Gives time to wait for locations to be found

- Parameter: “merge”
- Outputs: “success,” “notfound,” “failure”.

January 18, 2000
Signalling Actions: Proxy

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

- Name: “proxy”
- Parameter: timeout
  Gives time to wait before giving up on the proxy attempt.
- Outputs: “busy,” “noanswer,” “failure”
- If the proxy attempt was successful, the 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.

- Name: “redirect”
Signalling Actions: Response

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

- Name: “response”

- Parameter: “status”: value is one of “busy,” “notfound,” “reject,” or “error”, or optionally a 4xx, 5xx, or 6xx numeric code (if the server is a SIP server).

- Parameter: “reason”: A string explaining the failure in more detail.
Non-signalling action: Notify

Notify a user of something through some non-telephony means (e.g., e-mail).

- Name: “notify”
- Parameter: “url”: the address to contact (mailto, impp, etc)
- Parameter: “comment”: a string explaining the notification
- Outputs: “success,” “failure”
Non-signalling action: Log

Store a record of the current call in a log.

- Name: “log”.
- Parameter: “name”: the name of the log where this should be stored
- Parameter: “comment”: a string explaining the log entry
- Outputs: “success,” “failure”
Links

- XML syntax defines a tree; we want CPLs to be represented as directed acyclic graphs.

- To do this, we need to have links in the tree

- Every node can take an additional parameter, “id,” which takes an arbitrary token.

- Anywhere a node is expected, you can instead have a “link” tag, with a “ref” parameter which refers to some node’s id.

- Server must verify that the resulting graph is acyclic.
Example: Call Redirect Unconditional

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

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

<?xml version="1.0" ?>
<!DOCTYPE call SYSTEM "cpl.dtd">
<call>
  <location url="sip:jones@jonespc.example.com">
    <proxy timeout="8s">
      <busy>
        <location url="sip:jones@voicemail.example.com" merge="clear" id="voicemail" />
        <proxy />
      </busy>
    </proxy>
  </location>
  <noanswer>
    <link ref="voicemail" />
  </noanswer>
</call>
Example: Call Screening

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

call
  string-switch field="from">
    string matches="anonymous@*">
      response status="reject"
        reason="I don’t accept anonymous calls" />
  </string>
</string-switch>
</call>
```
Example: Time-of-day Routing

<?xml version="1.0" ?>
<!DOCTYPE call SYSTEM "cpl.dtd">
<call>
  <time-switch>
    <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>
</call>
Example: Non-call Actions

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

<call>
  <lookup url="http://www.example.com/cgi-bin/locate.cgi?user=jones"
           timeout="8s">
    <success>
      <proxy />
    </success>
    <failure>
      <notify url="mailto:jones@example.com"
                comment="The lookup server failed">
        <success>
          <response status="error" />
        </success>
      </notify>
    </failure>
  </lookup>
</call>
Open issues

In no particular order...

- How do we make decisions based on media types? Media types, source address, bandwidth?
- The user should be able to specify time zones for time-switch — how?
- H.323 attributes?
- What additional Proxy results are needed?
- What additional Rejection reasons are needed?
- Is “notify” too general? Should we just have “mailto” instead?
- Other issues?