

The Session Initiation Protocol (SIP)

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

May 2001

Overview

- protocol architecture
- typical component architectures
- addressing and locating SIP entities
- protocol operation and extensions
- reliability
- services, features and caller preferences
- security and QoS
- programming SIP services

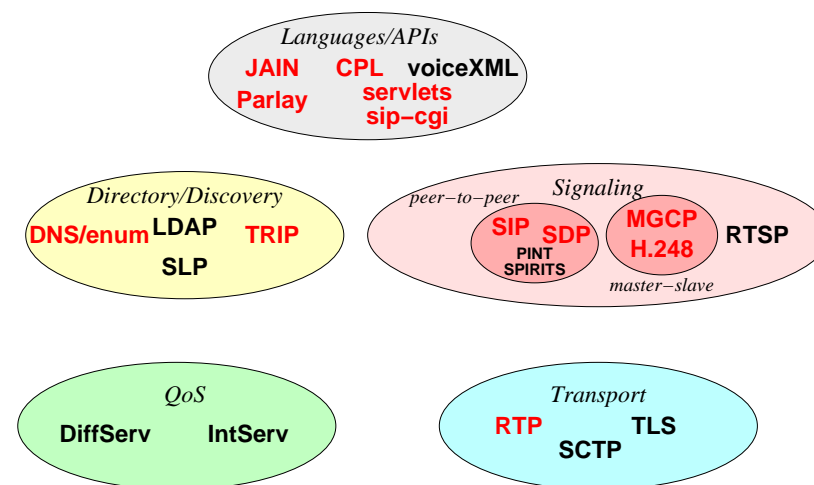
May 2001

Introduction

- SIP = core protocol for establishing *sessions* in the Internet
- transports session description information from initiator (caller) to callees
- allows to change parameters in mid-session
- terminate session

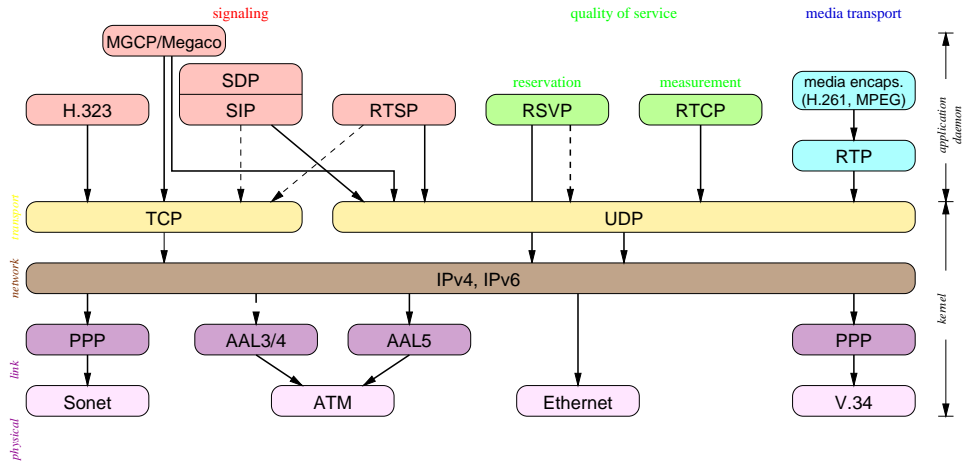
May 2001

VoIP protocol architecture



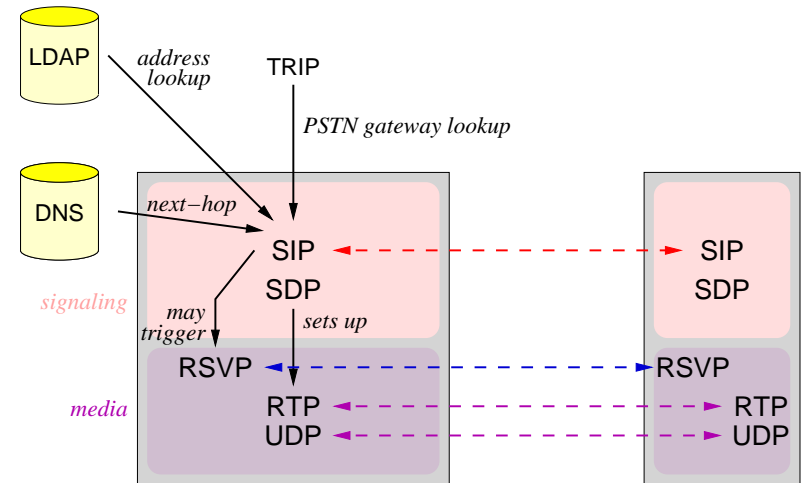
May 2001

Multimedia protocol stack



May 2001

SIP protocol use



May 2001

SIP applications

- setting up voice-over-IP calls
- setting up multimedia conferences
- event notification (subscribe/notify) → IM and presence
- text and general messaging
- signaling transport

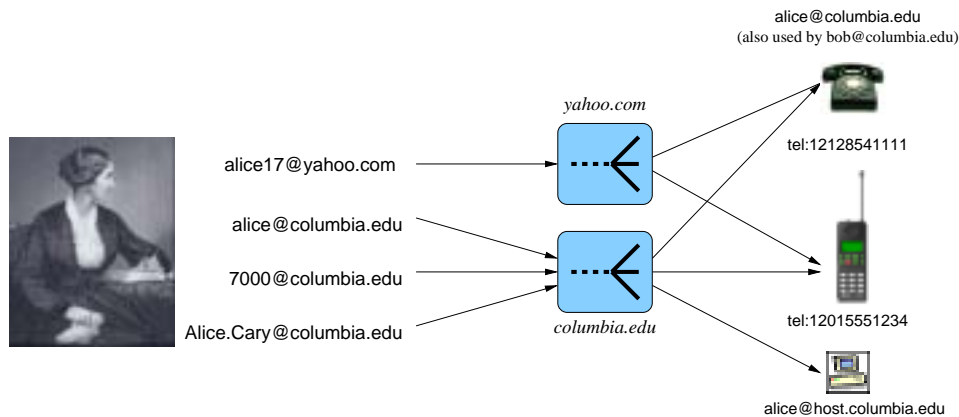
May 2001

SIP addressing

May 2001

Personal mobility

SIP uses email-style addresses to identify users



May 2001

SIP addressing

- typically, same as user's email address:
alice@example.com
12125551212@gateways-r-us.com
- written as URL, e.g., sip:alice@example.com
- can add parameters, such as type (user="phone") or transport protocol

May 2001

tel URLs (RFC 2806)

- also can use tel URLs for telephone numbers, e.g., tel:+12125551212 or fax:+358.555.1234567
- either global (tel:+1...) or local (tel:0w003585551234567;phone-context=+3585551234 numbers)
- allow post-dialing digits: ;postd=pp32
- also modem:+3585551234567;type=v32b?7e1;type=v110

May 2001

SIP building blocks



SIP user agent IP phone, PC, conference bridge



SIP redirect server returns new location for requests



SIP stateless proxy routes call requests



SIP (forking) proxy routes call requests

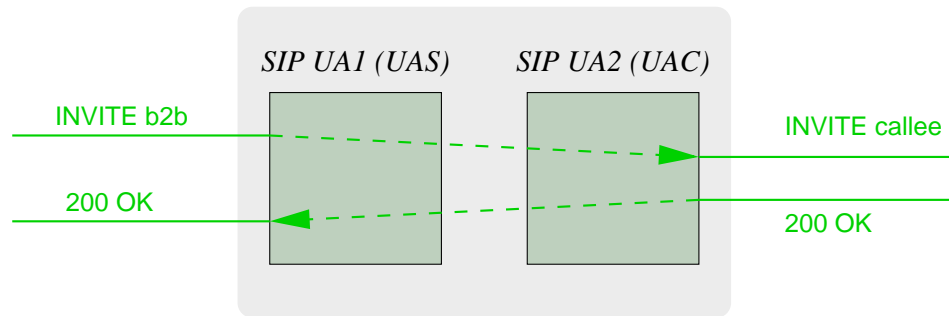


SIP registrar maintains mappings from names to addresses

May 2001

Back-to-back UA (B2BUA)

- two (or more) user agents, where incoming calls trigger outgoing calls to somebody else
- also, “third-party call control” (later)
- useful for services and anonymity



May 2001

Maintaining state in SIP entities

Stateless: each request and response handled independently

(Transaction) stateful: remember a whole request/response *transaction*

Call stateful: remember a call from beginning to end

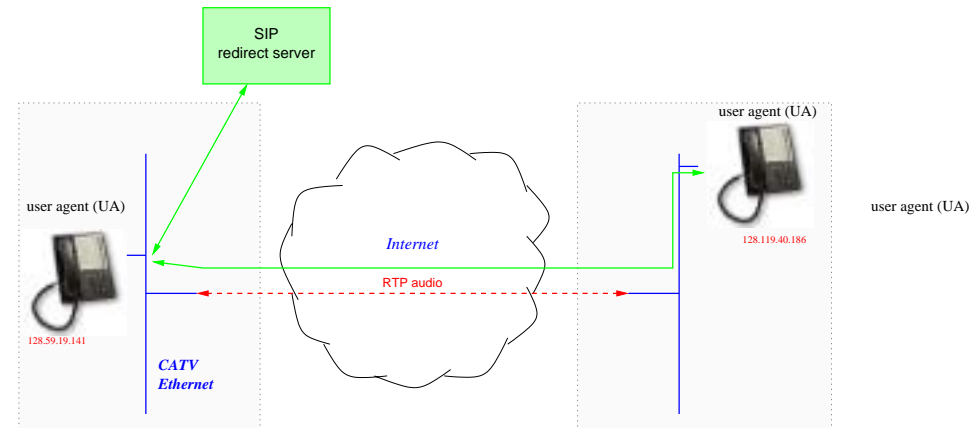
May 2001

SIP building block properties

	media	stateless	stateful	call state
UA (UAC, UAS)	yes	no	unlikely	common
proxy	no	yes	common	possible (firewall)
redirect registrar	no	no	yes	N/A

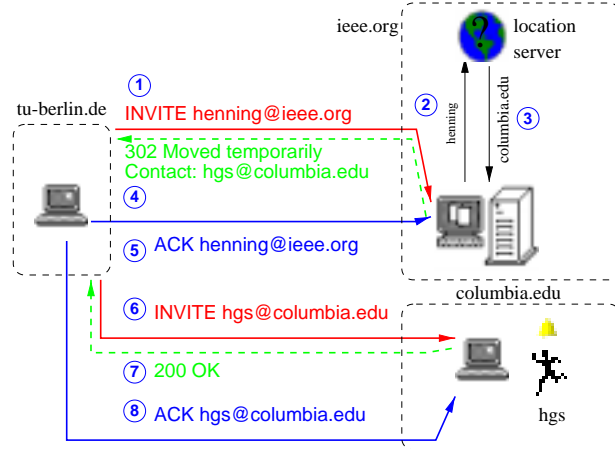
May 2001

SIP architecture: peer-to-peer



May 2001

SIP operation in redirect mode



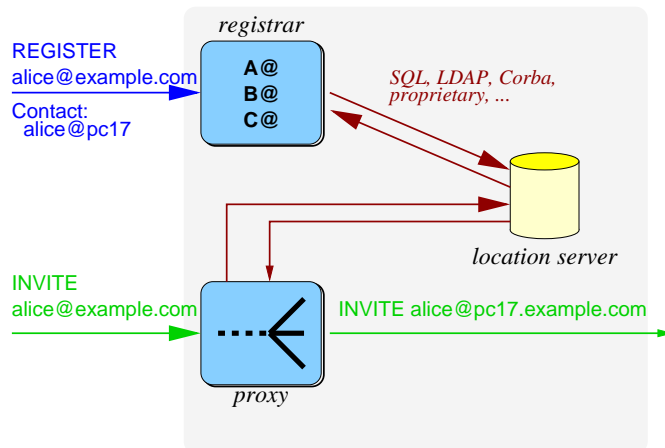
(302: redirection for single call; 301 permanently)

May 2001

Locating SIP users

May 2001

Locating users: registrars and location servers



May 2001

Basic user location mechanism

1. host(SIP URL) → host name of proxy
2. DNS: host name of proxy → SIP server(s)
3. if SIP UAS: alert user; done
4. if SIP proxy/redirect server: map $URL_n \rightarrow URL_{n+1}$, using any information in request
5. go to step 1

One minor exception...

May 2001

Basic SIP “routing” mechanisms

- will fill in details later
- route using request URIs
- all but first request in call typically bypass proxies and go direct UAC – UAS
- however, can use “record-routing” to force certain proxies to be visited all the time
- responses always traverse the same route as requests

May 2001

Outbound proxies

- normally, proxy serves one or more domains
- outbound proxies are used for *all* outbound requests from within a domain
- typically, for managing corporate firewalls and policy enforcement
- may also provide dial plans or route tel/fax URLs
- other uses: lawyer client billing, ...

May 2001

Locating users: DNS SRV

- email: DNS MX record allows mapping of domain to mail host, e.g.

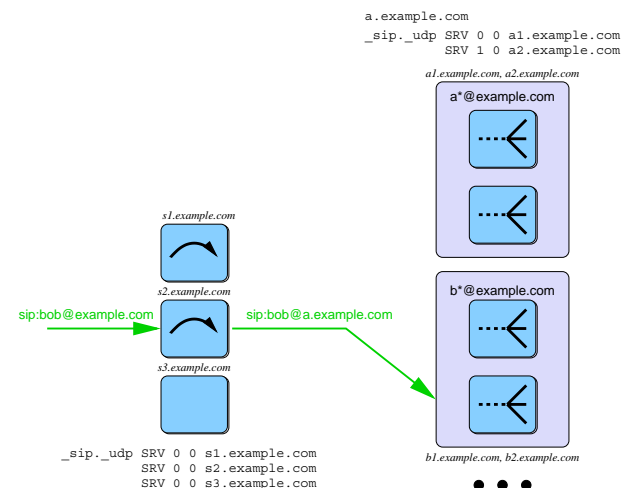
```
host -t mx yahoo.com
```

```
yahoo.com          MX      1  mx2.mail.yahoo.com
yahoo.com          MX      1  mx3.mail.yahoo.com
yahoo.com          MX      1  mx1.mail.yahoo.com
yahoo.com          MX      9  mta-v1.mail.yahoo.com
```

- SIP: use a newer record for general-purpose mapping, SRV (RFC 2782)
 - mapping from service and transport protocol to one or more servers, including protocols
- ```
_sip._tcp SRV 0 0 5060 sip-server.cs.columbia.edu.
 SRV 1 0 5060 backup.ip-provider.net.
_sip._udp SRV 0 0 5060 sip-server.cs.columbia.edu.
 SRV 1 0 5060 backup.ip-provider.net.
```
- allows priority (for back-up) and weight (for load balancing)

May 2001

## Using DNS SRV for scalable load-balancing



May 2001

## Aside: SIP scaling

---

- HTTP request director ↔ SIP client-based
- HTTP randomized DNS (short TTL!) ↔ SRV weights and priorities
- can't just distribute requests randomly, since backend (registration) synchronization is needed
- registration scaling: requests/second \* 3600; e.g., 100 requests/second → 360,000 users/server
- major bottlenecks are logging and database updates
- generally, higher registration than INVITE rates

May 2001

# SIP protocol operation

May 2001

## SIP requests and responses

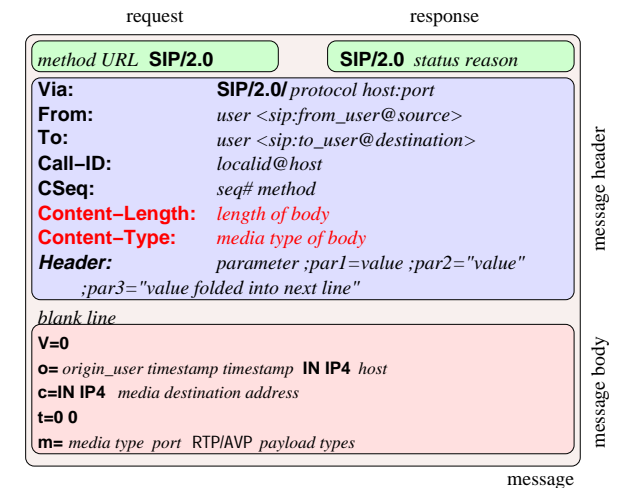
---

- text, not binary, format
- look very similar to HTTP/1.1
- requests and responses are similar except for first line
- requests and responses can contain *message bodies*: typically session descriptions, but also ASCII or HTML

May 2001

## SIP syntax

---



May 2001

## 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

May 2001

## SIP methods

|           |                                                    |
|-----------|----------------------------------------------------|
| INVITE    | initiate call                                      |
| ACK       | confirm final response                             |
| BYE       | terminate (and transfer) call                      |
| CANCEL    | cancel searches and "ringing"                      |
| OPTIONS   | features support by other side                     |
| REGISTER  | register with location service                     |
| INFO      | mid-call information (ISUP)                        |
| COMET     | precondition met                                   |
| PRACK     | provisional acknowledgement                        |
| SUBSCRIBE | subscribe to event                                 |
| NOTIFY    | notify subscribers                                 |
| REFER     | ask recipient to issue SIP request (call transfer) |

May 2001

## SIP invitation and media negotiation

*alice@wonderland.com**calls**bob@macrosoft.com***INVITE sip:bob@macrosoft.com SIP/2.0**

From: sip:alice@wonderland.com  
 To: sip:bob@macrosoft.com  
 Call-ID: 31415@wonderland.com  
 CSeq: 42 INVITE  
 Content-Type: application/sdp

v=0  
 o=user1 536 2337 IN IP4 h3.wonderland.com  
 c=IN IP4 h3.wonderland.com  
 m=audio 3456 RTP/AVP 0 1  
 m=video 4000 RTP/AVP 38 39

**SIP/2.0 200 OK**

From: sip:alice@wonderland.com  
 To: sip:bob@macrosoft.com  
 Call-ID: 31415@wonderland.com  
 CSeq: 42 INVITE  
 Content-Type: application/sdp

v=0  
 o=user1 535 687637 IN IP4 m.macrosoft.com  
 c=IN IP4 m.macrosoft.com  
 m=audio 1200 RTP/AVP 1  
 m=video 0 RTP/AVP

*accept audio, decline video*

May 2001

## 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

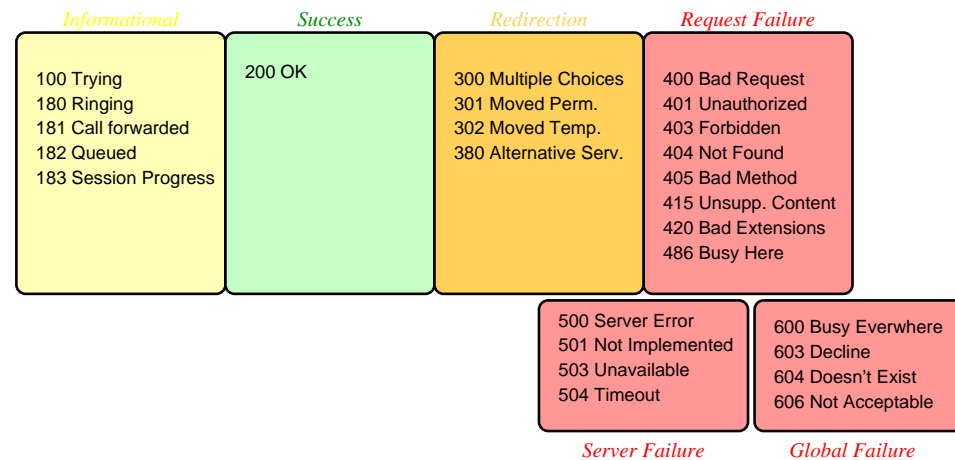
May 2001

## SIP call legs

- *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

May 2001

## SIP responses



May 2001

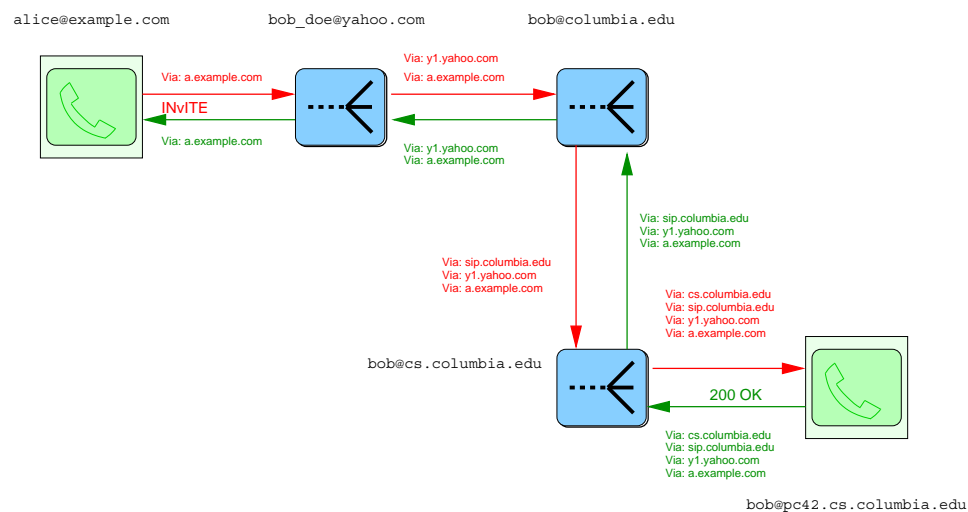
## SIP response routing

- requests are routed via URL
- 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
```

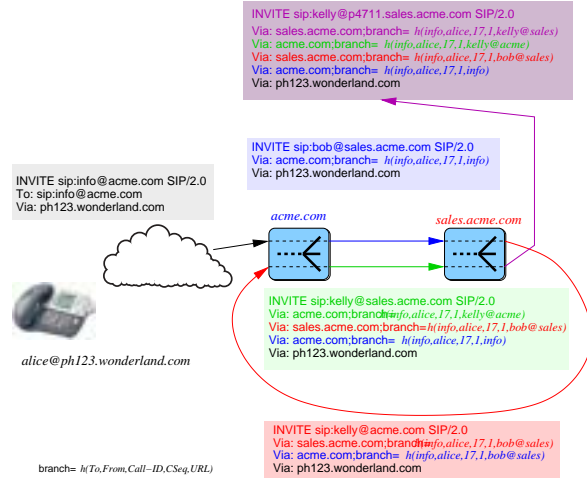
May 2001

## SIP response routing



May 2001

## SIP spirals



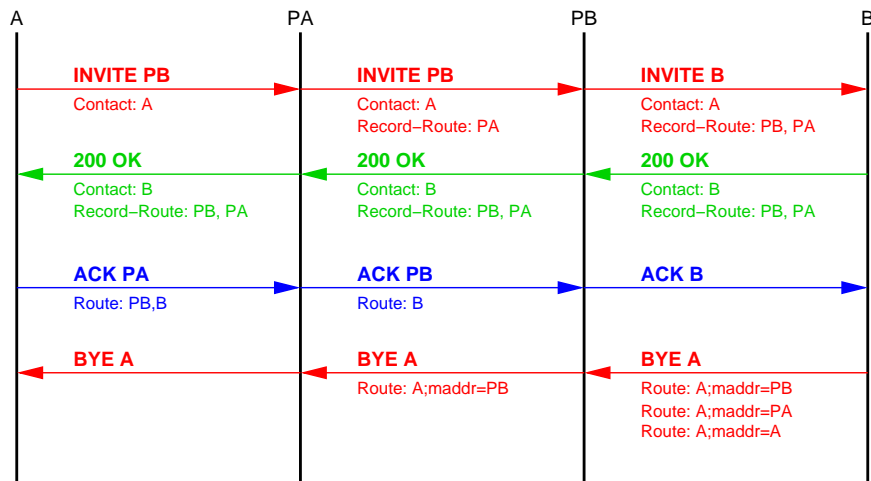
May 2001

## 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

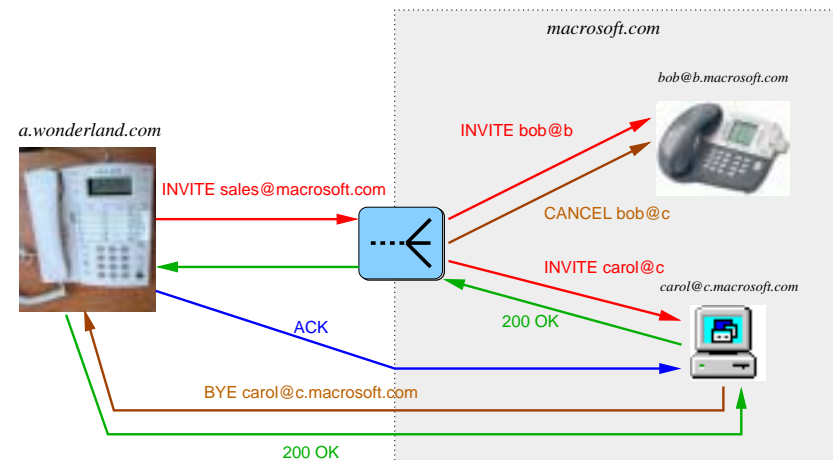
May 2001

## Request routing



May 2001

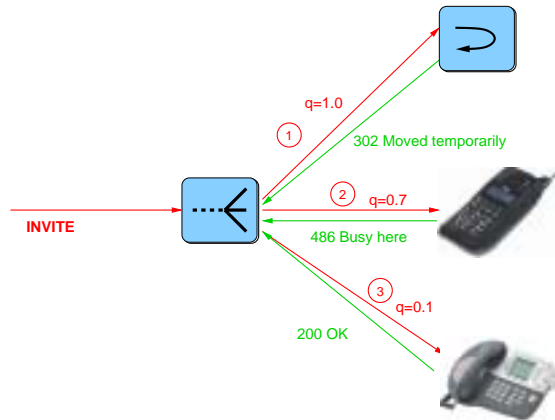
## SIP request forking



May 2001

## SIP sequential request forking

Use q values to govern order of sequential search:



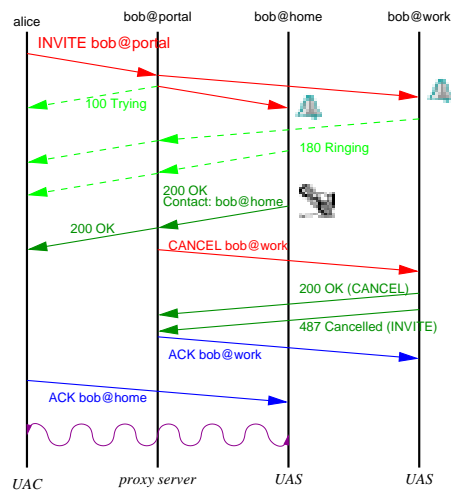
May 2001

## SIP request forking

- branches tried in sequence or parallel (or some combination)
- recursion: may try new branches if branch returns 3xx
- return best final answer = lowest status code
- forward provisional responses

May 2001

## Parallel forking call flow



May 2001

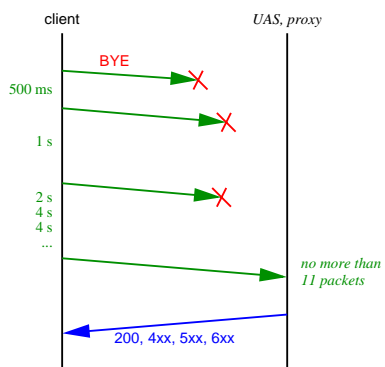
## SIP transport issues

- SIP operates over any packet network, reliable or unreliable
- choices:
  - UDP:** most common
    - low state overhead
    - small max. packet size
  - TCP:** can combine multiple signaling flows over one link
    - use with SSL
    - connection setup overhead
    - HOL blocking for trunks
  - SCTP:** new protocol
    - no HOL blocking
    - fallback address (but SRV provides this already)
    - connection setup overhead

May 2001

## Transport reliability for all but INVITE

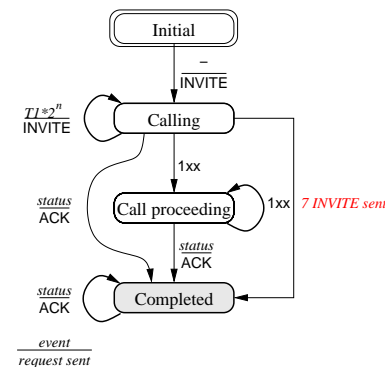
- used for BYE, OPTIONS, SUBSCRIBE, NOTIFY, ...
- 1xx sent by UAS or proxy only if no final answer expected within 200 ms
- if provisional response, re-transmit with  $T^2$  (4) seconds



May 2001

## INVITE reliability

- INVITE is special – long time between request and final response
- 100 (by proxy) indicates request has been received
- proxy usually forwards 1xx from all branches
- only retransmit until 100
- ACK confirms receipt of final response



May 2001

## Other signaling approaches

May 2001

## Differences to classical signaling

| name        | examples    | network        | “channel” |
|-------------|-------------|----------------|-----------|
| in-band     | E&M, DTMF   | same           | same      |
| out-of-band | ISUP, Q.931 | different      | different |
| IP          | SIP         | typically same | different |

IP signaling meets media only at end systems, while PSTN out-of-band intersects at every switch

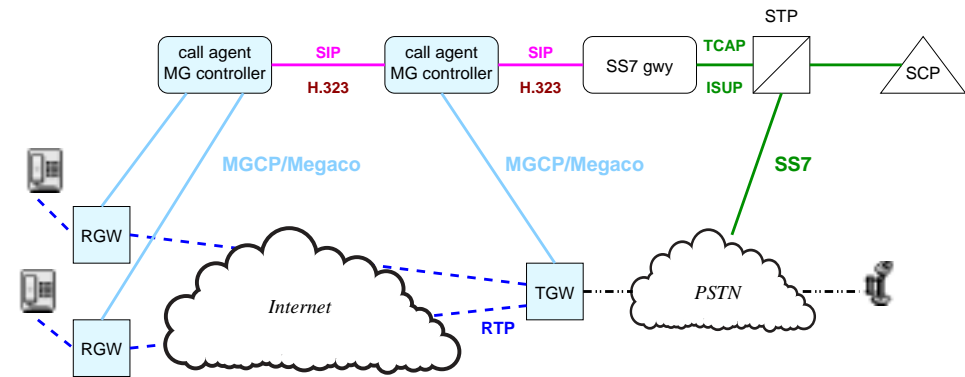
May 2001

## Aside: Alternative architecture: master-slave

- master-slave: MGC (media gateway controller) controls one or more gateways
- allows splitting of signaling and media functionality
- “please send audio from circuit 42 to 10.1.2.3”
- uses MGCP (implemented) or Megaco/H.248 (standardized, but just beginning to be implemented)
- gateway can be residential
- basis of PacketCable NCS (network control system) architecture
- service creation similar to digital PBX or switch
- end system has no semantic knowledge of what’s happening
- → can charge for caller id, call waiting

May 2001

## MGCP/SIP architecture



May 2001

## Extending SIP

| extension       | behavior    | determine? |
|-----------------|-------------|------------|
| new headers     | ignored     | –          |
| new headers     | mandatory   | Supported  |
| new method      |             | OPTIONS    |
| new body type   |             | Accept     |
| new status code | class-based |            |
| new URL type    |             | ?          |

May 2001

## 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
```

May 2001

# User identification

## Standard call/caller identification

---

**Request-URI:** next hop

**To:** logical call destination

**From:** logical call origin

**Organization:** organization of caller/callee

**Subject:** subject of call

**Call-Info:** additional information about caller or callee

```
Call-Info:
 <http://www.example.com/alice/photo.jpg> ;purpose=icon,
 <http://www.example.com/alice/> ;purpose=info
```

**User-Agent:** make and model of user agent

## Additional call information

---

**Priority:** call priority: emergency, urgent, normal, non-urgent

**Alert-Info:** render instead of ring tone

```
Alert-Info: <http://www.example.com/sounds/moo.wav>
```

**In-Reply-To:** call-id being returned

## draft-ietf-sip-privacy

---

- To/headerFrom are chosen by end system → may lie

- need privacy indications similar to caller id

```
Remote-Party-ID: "John Doe"
 <sip:jdoo@foo.com>;party=calling;
 id-type=subscriber;privacy=full
```

- screen=yes: was verified by proxy
- type can be subscriber, user, alias, return (calls), term (terminal)
- may add geographic user location

# SIP services

May 2001

## Invitation modes

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

► SIP for all modes, SAP/SDP also for multicast/multicast

May 2001

## SIP-based services

**Call forwarding:** basic INVITE behavior (proxy/redirect)

**Call transfer:** REFER method (see later)

**Call hold:** set media address to 0.0.0.0 – can be done individually per media

**Caller id:** From, plus extensions

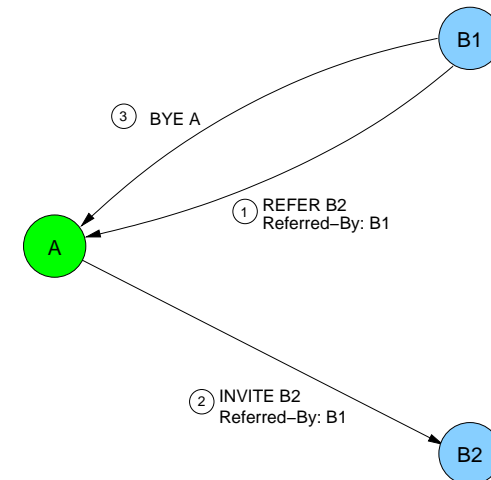
**DTMF carriage:** carry as RTP payload (RFC 2833)

**Calling card:** B2BUA + voice server

**Voice mail:** UA with special URL(s) + possibly RTSP

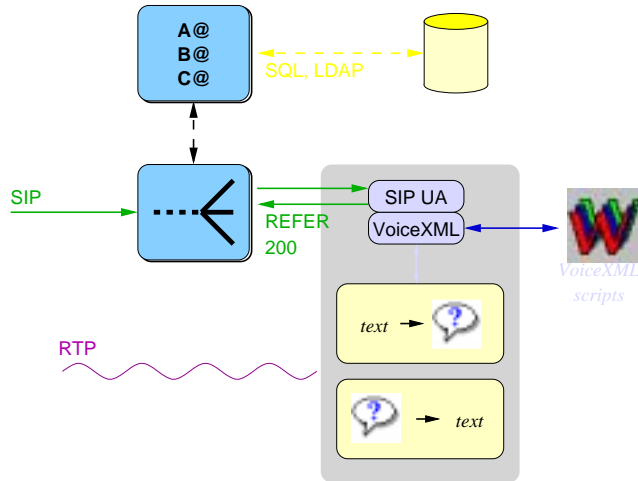
May 2001

## Call transfer



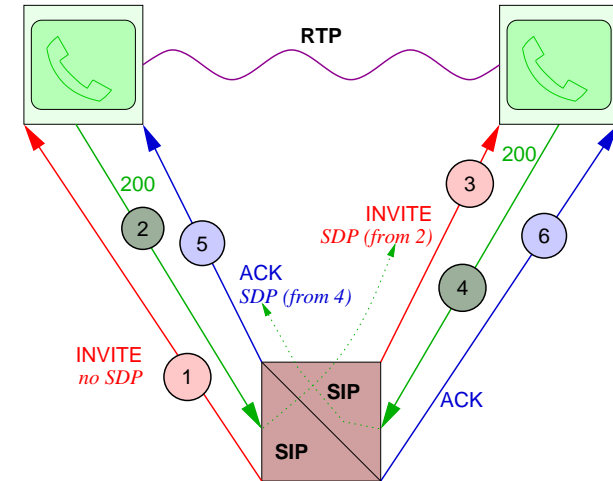
May 2001

## IVR and VoiceXML



May 2001

## Third-party call control



May 2001

## SIP billing/charging

### What for?

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

### How?

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

May 2001

## Security issues

May 2001

## Threats

---

- spoofing From in REGISTER: call redirection
- spoofing From in INVITE: bypass call filtering
- snooping media packets
- billing confusion (identifier munging)
- denial-of-service attacks

## SIP security

---

| layer/mechanism | approach     | characteristics                                   |
|-----------------|--------------|---------------------------------------------------|
| network layer   | IPsec        | adjacent nodes, all or nothing, hard to configure |
| transport layer | TLS          | adjacent nodes, all or nothing                    |
| SIP INVITE      | basic/digest | shared secrets with random parties                |
| SIP REGISTER    | basic/digest | securing headers?                                 |
| SIP general     | S/MIME       | in progress                                       |

Basic (plaintext password) and digest (challenge-response) are very similar to HTTP security mechanisms.

## 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: QWxhZGRpbjpvcmVudHNIc2FtZQ==
```

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="42ce3cef44b22f50c6a6071bc8"
```

May 2001

## Digest authentication

---

**username:** user authenticating herself

**realm:** several per user, used also for display

**nonce:** copied into Authorization

**opaque:** copied into Authorization

**uri:** original request URL

**response:** 32 hex digits:

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

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

where  $A_1 = \text{username} : \text{realm} : \text{passwd}$

$A_2 = \text{method} : \text{uri}$

May 2001

# Quality of Service

May 2001

## Quality of service

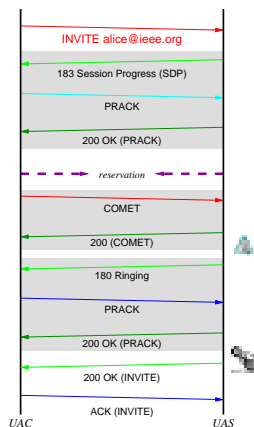
---

- SIP and data paths disjoint  $\Rightarrow$  SIP can't reserve resources
- but: SDP may provide information to end systems on desired QoS
- SDP contains range of codecs to allow mid-call adaptation

May 2001

## Interaction with resource reservation

avoid “fast busy” after ringing → interleave



May 2001

## SIP Caller Preferences

May 2001

## 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”

May 2001

## Extended SIP Contact header

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

May 2001

## 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
```

May 2001

## 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
```

May 2001

## 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?

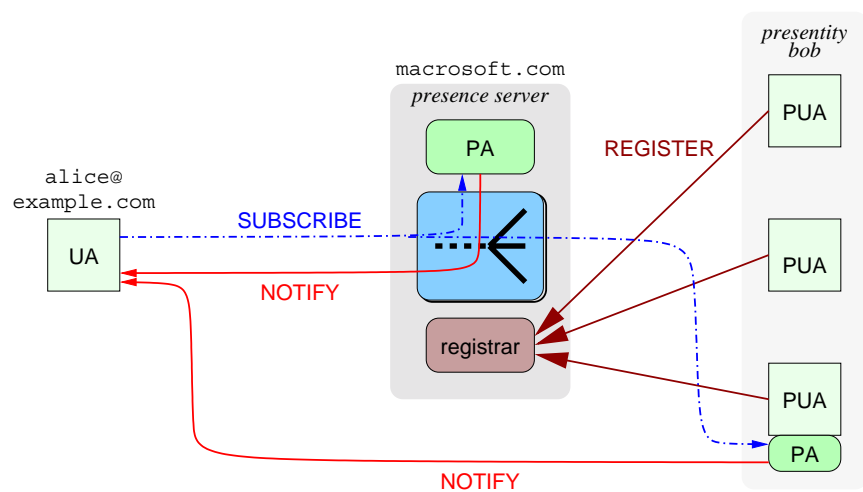
```
Request-Disposition: proxy, recurse, parallel
```

May 2001

# SIP presence, events and instant messaging

May 2001

## SIP presence architecture



May 2001

## SIP presence components

**Presence entity:** logical entity being subscribe to, e.g., alice@wonderland.com, with several agents

**Registrar:** receives REGISTER requests

**Presence user agent (PUA):** generates REGISTER, but no SUBSCRIBE or NOTIFY → any non-presence-aware SIP software

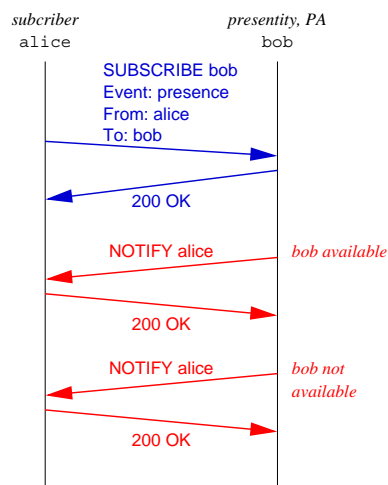
**Presence agent:** receive SUBSCRIBE, generate NOTIFY

**Presence server:** SIP proxy + PA

**Presence client:** SIP UA + PA

May 2001

## SIP presence protocol



May 2001

## SIP SUBSCRIBE example

```
SUBSCRIBE sip:bob@microsoft.com SIP/2.0
Event: presence
To: sip:bob@microsoft.com
From: sip:user@example.com
Contact: sip:user@userpc.example.com
Call-ID: knsd08alas9dy@3.4.5.6
CSeq: 1 SUBSCRIBE
Expires: 3600
Content-Length: 0
```

- Forked to all PUAs that have REGISTERed with method SUBSCRIBE.
- 200 (OK) response contains current state.

May 2001

## SIP NOTIFY example

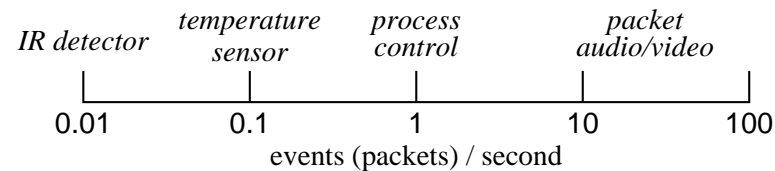
```
NOTIFY sip:user@userpc.example.com
To: sip:user@example.com
From: sip:alice@wonderland.com
Call-ID: knsd08alas9dy@3.4.5.6
CSeq: 1 NOTIFY
Content-Type: application/xpidf+xml
```

```
<?xml version="1.0"?>
<!DOCTYPE presence
PUBLIC "-//IETF//DTD RFCxxxx XPIDF 1.0//EN" "xpidf.dtd">
<presence>
 <presentity uri="sip:alice@wonderland.com;method="SUBSCRIBE">
 <atom id="779js0a98">
 <address uri="sip:alice@wonderland.com;method=INVITE">
 <status status="closed"/>
 </address>
 </atom>
 </presentity>
</presence>
```

May 2001

## SIP events

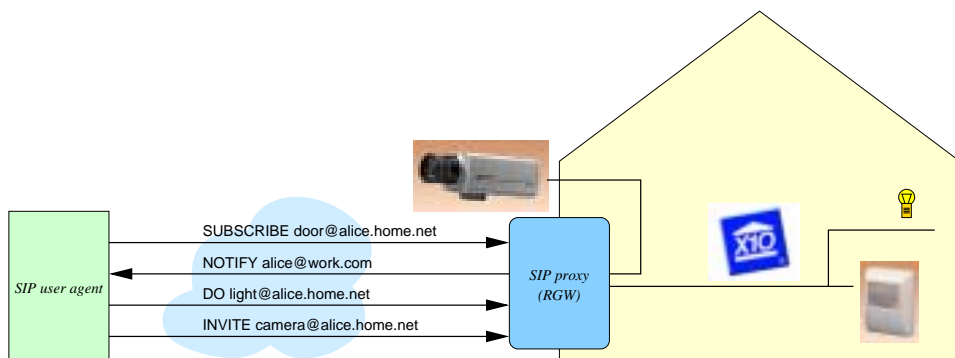
- single-valued (light-switch) to complex (CD changer) to multi-valued (temperature samples)
- both built-in and mediated (X10)
- often combined with audio/video in same system: security, industrial control, home entertainment
- notification rates vary  $\Rightarrow$  gradual transition to continuous media



- Event describes event type

May 2001

## Example home architecture



(Work with Telcordia)

May 2001

## SIP IM

- send text or any other MIME type
- either as SDP-initiated session or as individual messages
- use MESSAGE

May 2001

# Programming SIP Services

## Programming SIP services

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

## Programming services

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

May 2001

## 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

May 2001

## 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

May 2001

## 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");
}
```

May 2001

```

$destination = $addresses{$to};

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

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

```

## The Call Processing Language

---

Jonathan Lennox  
Columbia University  
lennox@cs.columbia.edu

May 5, 2000

## Purpose

---

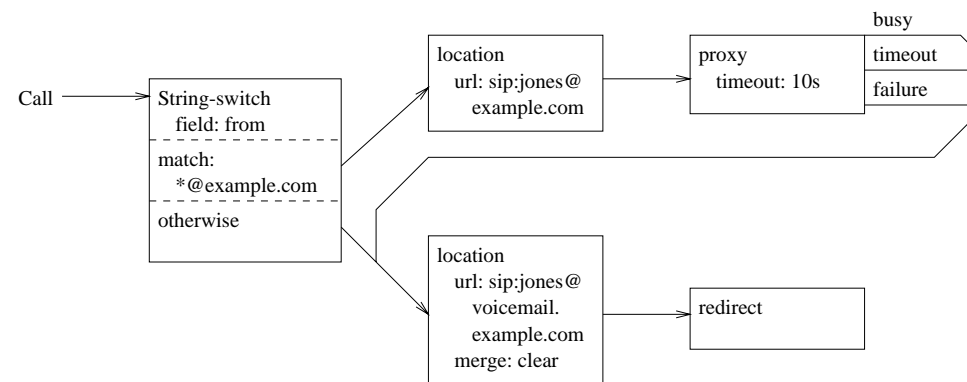
Allow users to create simple Internet telephony services

Features:

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

## Abstract structure

---



## Abstract structure (cont)

---

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

May 2001

## Textual representation

---

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

May 2001

## Textual representation

---

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

May 2001

## 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

May 2001

## Switch nodes

---

Switch nodes make decisions.

Structure:

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

May 2001

## 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”

May 2001

## String Switches: string

---

Switch based on textual strings, with conditions:

**is:** exact string match

**contain:** substring match

Fields: subject, organization, user-agent

May 2001

## 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:  $a_1 - b_1, a_2 - b_2, \dots$
- must fall within a range of *all* specified conditions

May 2001

## Time switches: examples

---

```
<time month="12" date="25" year="1999">
 December 25th, 1999, all day
```

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

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

## Time switches: examples

---

```
<time timeofday="1310-1425,1440-1555,1610-1725"
 day="2,4">
 1:10 – 2:25 PM, 2:40 – 3:55 PM, and 4:10 – 5:25 PM, Tuesdays and Thursdays,
 every week
```

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

## Location nodes

---

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

## Simple location nodes: location

---

Specify a location explicitly.

**url:** explicitly specified location

**clear:** clear earlier location values

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

## Location lookup nodes: `lookup`

---

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

Parameters:

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

**timeout:** time to wait

**use/ignore:**

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

**merge:**

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

May 2001

## 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

May 2001

## 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

May 2001

## Signalling Actions: `redirect`

---

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

May 2001

## Signalling Actions: reject

---

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

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

**reason:** string explaining the failure.

## Non-signalling action: mail

---

Notify a user of something through e-mail.

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

## Non-signalling action: log

---

Store a record of the current call in a log.

**name:** the name of the log this should be stored

**omment:** a string explaining the log entry

Outputs: success, failure

## Subactions

---

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

## Example: Call Redirect Unconditional

---

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

May 2001

## Example: Call Forward Busy/No Answer

---

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

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

May 2001

## 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>
```

May 2001

## Example: Time-of-day Routing

---

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

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

May 2001

## Example: Non-call Actions

---

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

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

May 2001

# SIP for Third-Generation Wireless Networks

May 2001

## 3G networks

---

- successor to 2G mobile networks: GSM (TDMA) and IS-95 (CDMA) in 900/1800 MHz range
- 2.5G: GSM → GPRS → EDGE
- use different air interfaces in 2 GHz range: W-CDMA, CDMA 2000, TD-CDMA
- 3GPP standardizes for W-CDMA (GSM follow-on), while 3GPP2 does CDMA 2000
- identified by releases (1999, R4, R5)

May 2001

## 3G and VoIP

---

- GPRS not suitable for VoIP: low bandwidth, high delay (500-600 ms RTT)
- initially (R4), CS voice to base station, then ATM/IP packets
- later (R5), in *Internet multimedia* (IM) subsystem → IP to UE (user equipment)
- uses AMR audio codec, with variable rate of 4.75 to 12.2 kb/s, or GSM HR or EFR
- UTRAN delays: see TR 25.932

May 2001

## Signaling in 3GPP IM subsystem

Uses SIP for session setup and defines new entities:

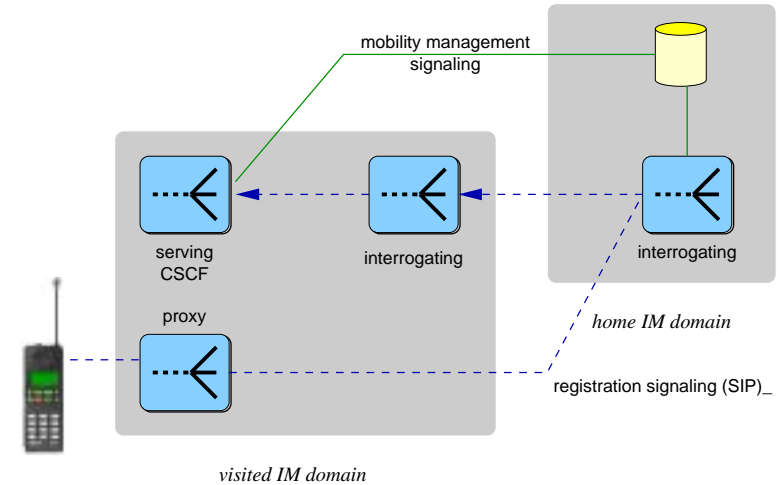
**Proxy CSCF:** first point of contact in visited network; finds the user's home network and provide some translation, security and authorization functions

**Serving CSCF:** controls sessions, acts as registrar and triggers and executes services. Accesses the user's profile; can be located in the home or visited network.

**Interrogating CSCF:** first point of contact in home network. It assigns the serving CSCF, contacts the HSS and forwards SIP request.

May 2001

## 3G SIP registration



May 2001

## Differences to “standard” SIP

- requires REGISTER before making call
- INVITE uses authentication information provided by REGISTER → Path header
- always visit I/P/S for “home” services
- compression on link from UE to P-CSCF

May 2001

## RFCs

draft-ietf-sip-rfc2543bis-03	base protocol spec
RFC 3087	<i>Control of Service Context using SIP Request-URI</i>
RFC 3050	<i>Common Gateway Interface for SIP</i>
RFC 2916	<i>E.164 number and DNS</i>
RFC 2833	<i>RTP Payload for DTMF Digits, Telephony Tones and Telephony Signals</i>
RFC 2806	<i>URLs for Telephone Calls</i>
RFC 2543	<i>SIP: Session Initiation Protocol</i>

May 2001

## For more information...

---

**SIP:** <http://www.cs.columbia.edu/sip>

**SDP:** <http://www.cs.columbia.edu/~hgs/internet/sdp.html>

**RTP:** <http://www.cs.columbia.edu/~hgs/rtp>

**Papers:** <http://www.cs.columbia.edu/IRT>