Session Initiation Protocol

SIP protocol and its extensions

SIP Service Architecture

SIP in 3G

A lot of this material
is based on proposals =>
may change quickly

Sources

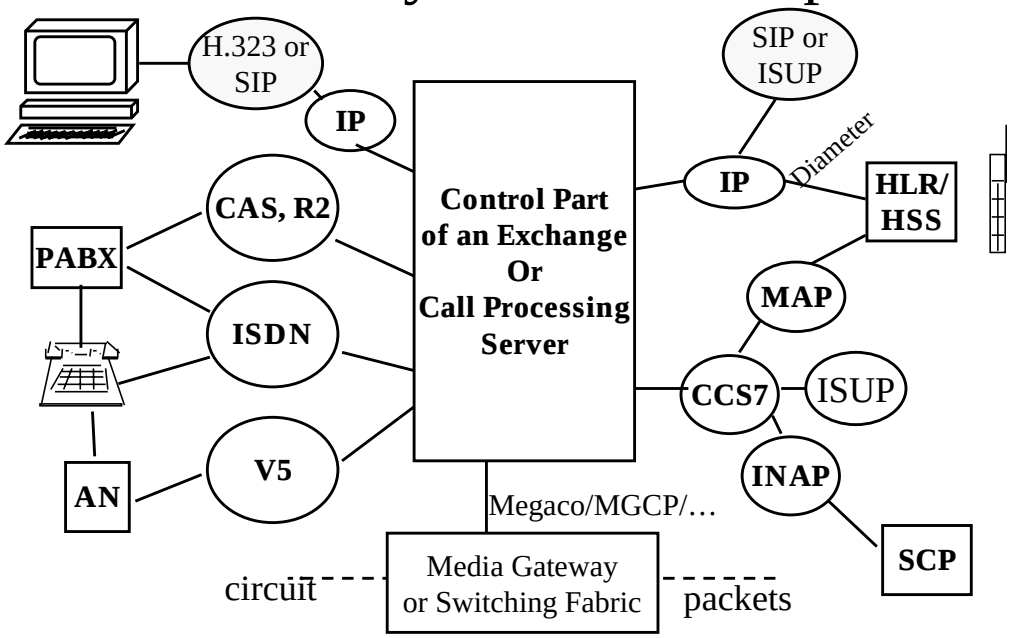
IETF:

RFC 3261: SIP: Session Initiation Protocol
RFC 3262: Reliability of Provisional Responses in SIP (PRACK)
RFC 3265: SIP Specific Event Notification
RFC 3311: SIP UPDATE method
RFC 3398: ISUP to SIP mapping
RFC 3428: SIP Extension for Instant Messaging
RFC 2327: SDP: Session Description Protocol
RFC 3264: An Offer/Answer Model with Session Description Protocol (SDP)

3G Release 5:

3GPP TS 24.228 v5.2.0 (2002-09) Signaling flows for the IP MM call control based on SIP and SDP; stage 3 (Release 5)
3GPP TS 24.229 v5.3.0 (2002-12) IP multimedia call control protocol based on SIP and SDP, Stage 3 (Release 5)
3GPP TS 29.228 v5.1.0 (2002-09) IMS Cx and Dx interfaces, Signaling flows and message contents; (Release 5)
Etc…

Summary of course scope



SIP Features

- Part of IETF toolkit
  - Reusing other protocols & mechanisms: HTTP, etc.
  - Flexible
  - Extensible
- Moves intelligence to End System entities
  - End-to-end protocol
- Interoperability
- Scalability (although some state in network)
- Service creation easy
- URLs and Addresses reusage
- Same routing as SMTP
- Reuses infrastructure (all applications will use SIP entities for different services)

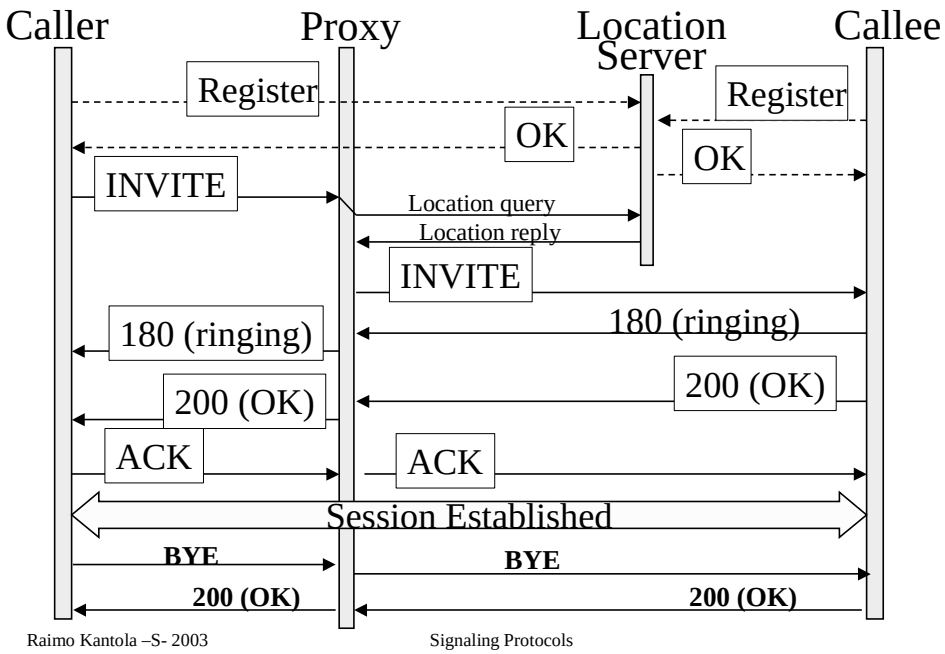
SIP overview

- Simplicity
  - Ascii based - simple tools for development
  - Lower call setup time than in H.323
  - basic protocol + extensions structure adopted
- Caller preferences, Ability to support many media types
- Runs over UDP or TCP (or SCTP)
- Used between both service and call control entities
- Has been adopted for 3G IP Multimedia signaling
- Originally subscriber signaling, proposed also as network to network signaling
- Quality of Specification is not (yet) very good! Leaves a lot of decisions to the implementor
- A lot of development during the last 3 years!

Sip Entities

- User Agents
  - Can act as client and as server
- Servers:
  - Redirect Servers
    - Send back alternative location of the user (similar as HTTP servers)
  - Proxy servers
    - Act on behalf of client (forwards requests)
    - Forking proxies
  - Registrars
    - Accepts registrations
  - Location Servers (not part of SIP architecture)
    - Gives back location of user (received from registrars)
    - E.g. HSS in 3GPP IMS architecture
    - Protocol between Location server and SIP server not defined by SIP specs (e.g. LDAP)

Basic SIP call setup and release



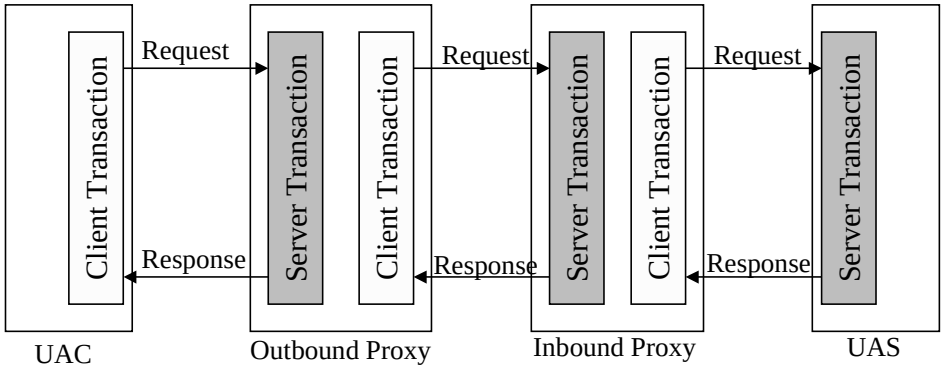
”Basic call” Example

- Caller sends INVITE
- Callee can accept, reject, forward the call
- If the callee accepts the call: responds with an optional provisional (1xx), and a final (≥200) response
- The caller confirms final response via ACK
- Conversation
- Caller or callee sends BYE
- BYE is acknowledged by 200 OK
- Low call setup times, post dial delay: 1.5 RTT !

SIP messages have headers and a body

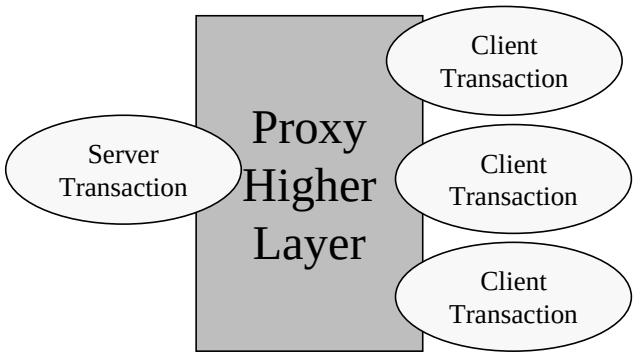
- Headers carry control information and are processed e.g. by Proxies
- Body can be e.g. SDP – session description protocol
  - end-to-end information (cmp H.245) describing session requirements e.g. coding methods, etc
- Message delivery is transaction oriented= have request + reply: e.g INVITE+200 OK

User Agent is split into User Agent Client (UAC) and User Agent Server(UAS)



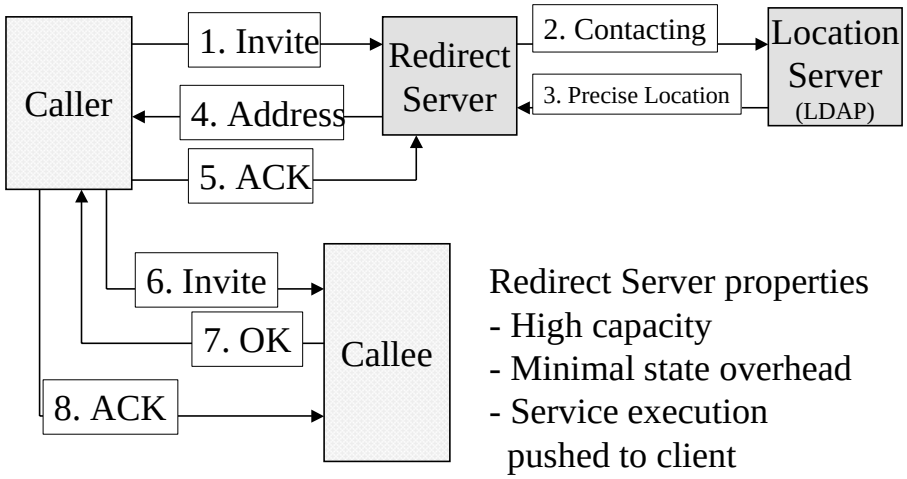
- All Communication follows this transaction model, except 2xx and ACK.
- Server transactions filter incoming requests, absorbing retransmissions
- Only UAS can generate 2xx and only UAC can generate ACK.

A Stateful Proxy can fork a transaction



Forking = multicast of INVITEs to N addresses

Redirect Server pushes processing to clients



Redirect Server properties
- High capacity
- Minimal state overhead
- Service execution pushed to client

Stateful Proxy vs Stateless Proxy

- Maintains call context
- Replicates UAS/UAC to process requests and responses
- Call state and transaction state can be maintained
- Forking proxies require state
- TCP proxies must be stateful for reliability
- Enhanced services require state
- Can collect charging info

- No call context
- Response is not based on UA replication
- Provides client anonymity
- Restricted gateway access
- High processing capacity
- Easier to replicate than the stateful proxy
- Also semi-stateful is possible

UA = User Agent, UAC = UA Client
UAS = UA Server

Some SIP issues

- Parties can release the "call session" but since they have obtained each others IP-addresses, they can continue sending media streams to each other!!
- How to push INVITE to B-party, if B-party does not have a permanent IP address which is most often the case!

} Integration of Proxy with Firewall and NAT

- Response messages (e.g. 180) are not reliably delivered. This may cause tear down of the call if it was initiated from ISDN

PRACK method

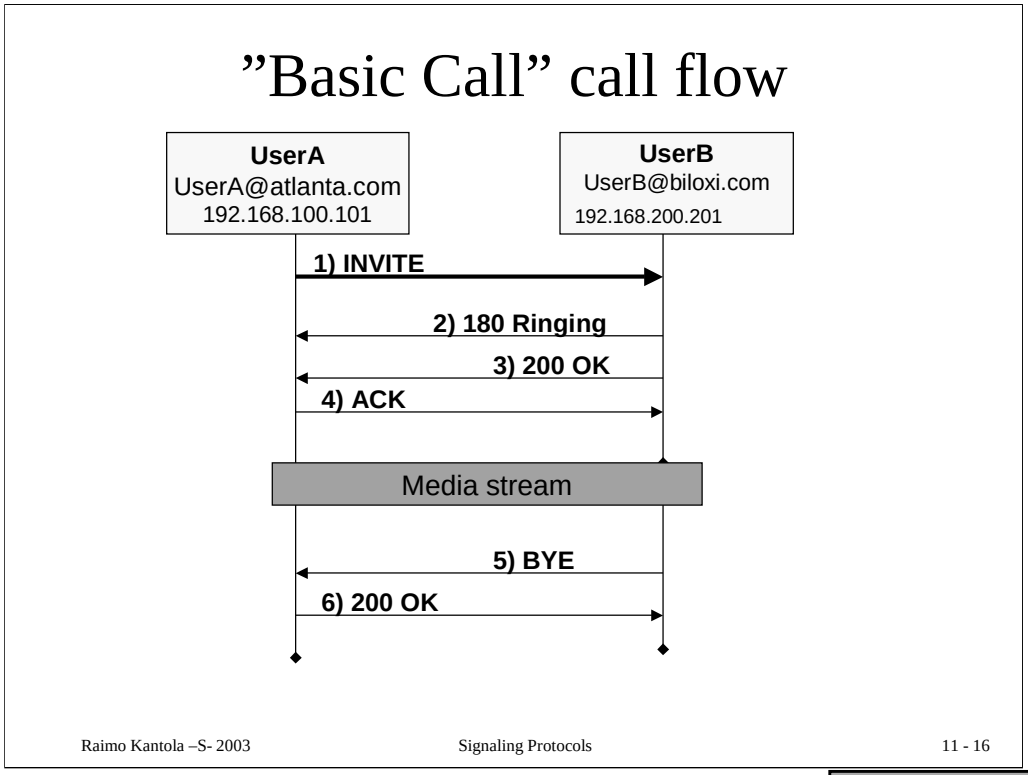
- If BYE is lost, Proxy does not know that call has ended

KeepAlive = re-INVITE mechanism

- Ascii coding increases the signaling overhead in Radio access

Addressing

- **sip:user@host[parameters][headers]**
- SIP-addresses are like URLs, with prefix sip: which gives schema
  - sip:joe.smith@hut.fi
  - sip:joe.smith@hut.fi?subject=Protocol
  - sip:sales@hotel.xy;geo.position:=48.54_-123.84_120
- Address must include host, other parameters are optional (username, port, etc…)
- Email-addresses can be reused
- “Click-to-call” on web-pages, MM messages, etc… is easy implemented



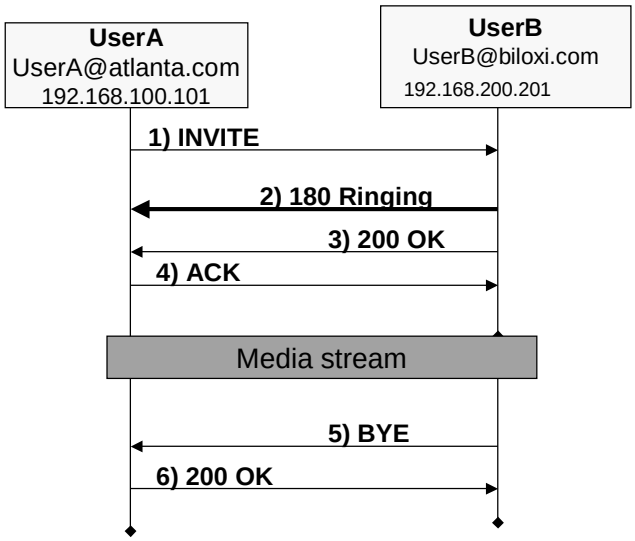

```
   INVITE sip:UserB@biloxi.com SIP/2.0
   Via: SIP/2.0/UDP
client.atlanta.com:5060;branch=z9hG4bK74bf9
   Max-Forwards: 70
   From: BigGuy <sip:UserA@atlanta.com>;tag=9fxced76sl
   To: LittleGuy <sip:UserB@biloxi.com>
   Call-ID: 3848276298220188511@atlanta.com
   CSeq: 1 INVITE
   Contact: <sip:UserA@192.168.100.101>
   Content-Type: application/sdp
   Content-Length: 147

   v=0
   o=UserA 2890844526 2890844526 IN IP4
client.atlanta.com
   s=-
   c=IN IP4 192.168.100.101
   t=0 0
   m=audio 49172 RTP/AVP 0
   a=rtpmap:0 PCMU/8000
```

In this example, UserA sends INVITE to UserB. In the initial INVITE UserA puts Max-Forwards: 70. In Contact: header UserA puts his address where he can be reached in the moment.

In SDP body UserA specifies what kind of session he wants to establish. In this case it would be audio session on port 49172 (RTP over UDP as transport) with PCM μ coding with 8000 samples per second.

Example from: http://www.ietf.org/internet-drafts/draft-ietf-sipping-call-flows-01.txt

”Basic Call” call flow



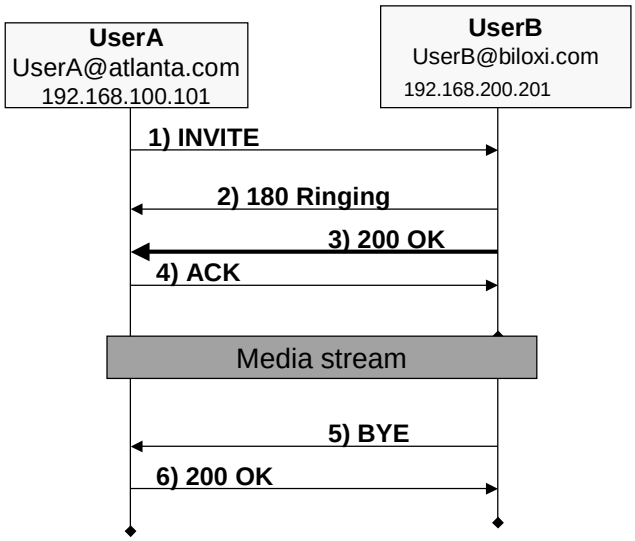
Raimo Kantola –S- 2003 Signaling Protocols 11 - 17

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP
client.atlanta.com:5060;branch=z9hG4bK74bf9
 ;received=192.168.100.101
From: BigGuy <sip:UserA@atlanta.com>;tag=9fxced76sl
To: LittleGuy <sip:UserB@biloxi.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.com
CSeq: 1 INVITE
Content-Length: 0
```

UserB answers with 180 Ringing provisional answer. UserB also adds tag to To: header, but maintains Call-ID and CSeq

Note that From: and To: fields are not changed. That is because UserB acts as Server (sending responses back), and call is started by UserA (in From: field). Order will be preserved untill transaction is complete

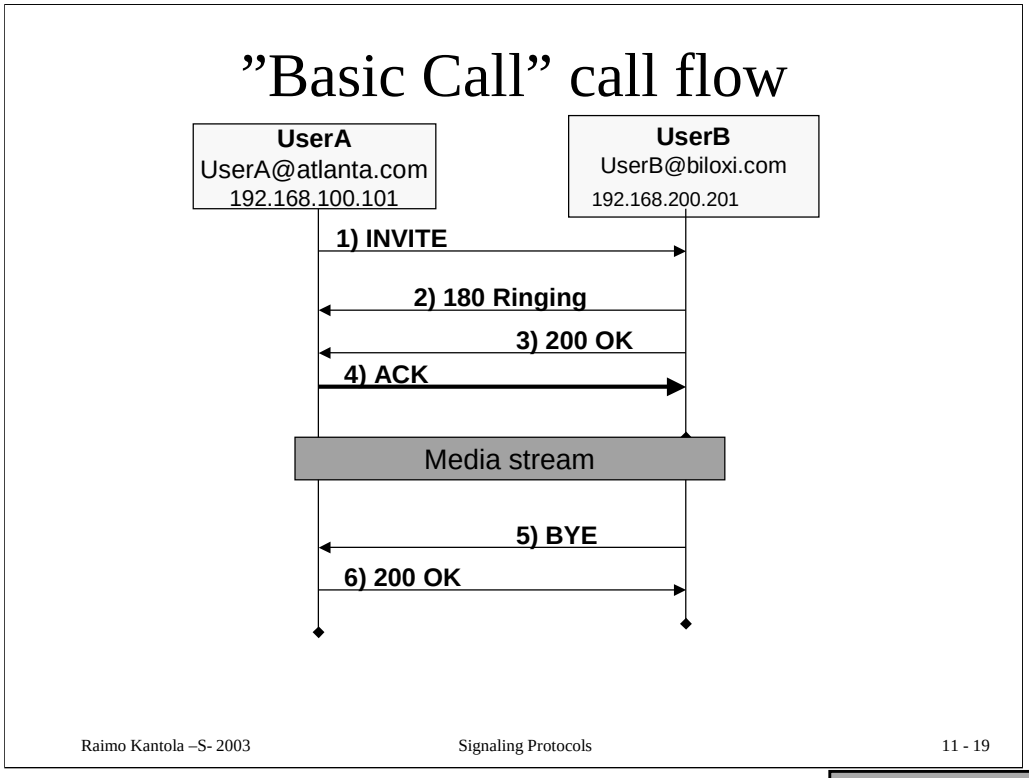
”Basic Call” call flow




```
   SIP/2.0 200 OK
   Via: SIP/2.0/UDP
client.atlanta.com:5060;branch=z9hG4bK74bf9
    ;received=192.168.100.101
   From: BigGuy <sip:UserA@atlanta.com>;tag=9fxced76sl
   To: LittleGuy <sip:UserB@biloxi.com>;tag=8321234356
   Call-ID: 3848276298220188511@atlanta.com
   CSeq: 1 INVITE
   Contact: <sip:UserB@192.168.200.201>
   Content-Type: application/sdp
   Content-Length: 145

   v=0
   o=UserB 2890844527 2890844527 IN IP4
client.biloxi.com
   s=-
   c=IN IP4 192.168.200.201
   t=0 0
   m=audio 3456 RTP/AVP 0
   a=rtpmap:0 PCMU/8000
```

UserB now sends 200 OK final response. He also adds Contact: field (where he could be reached).

In SDP body of the message UserB gives his preferences for session establishment (audio, RTP over UDP, port 3456, PCM μ modulation with 8000 samples per second), and also his IP address in c= field.

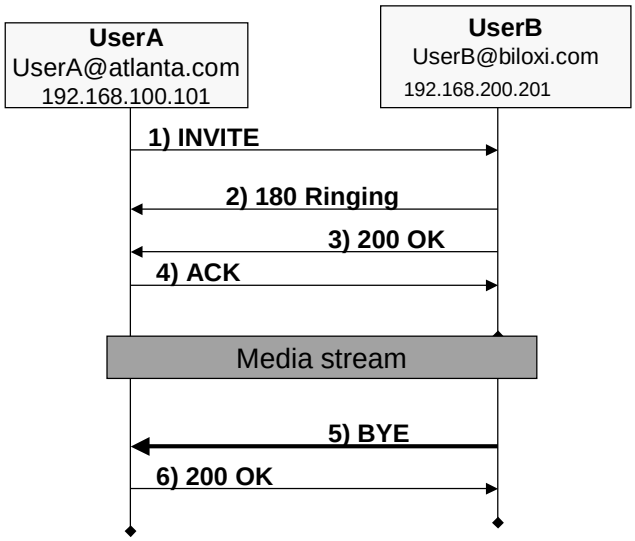



```
ACK sip:UserB@biloxi.com SIP/2.0
Via: SIP/2.0/UDP
client.atlanta.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: BigGuy <sip:UserA@atlanta.com>;tag=9fxced76sl
To: LittleGuy <sip:UserB@biloxi.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.com
CSeq: 1 ACK
Content-Length: 0
```

UserA acknowledges OK response. After ACK media session can be established, using parameters given in the handshake before (in INVITE and in 200 OK response)

CSeq header value is still 1 (because this is part of the same transaction), but method name has been changed to ACK

”Basic Call” call flow

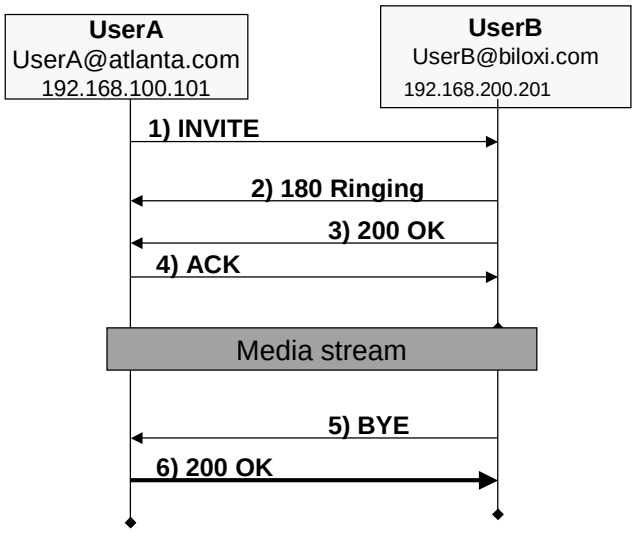



```
BYE sip:UserA@192.168.100.101 SIP/2.0
 Via: SIP/2.0/UDP
client.biloxi.com:5060;branch=z9hG4bKnashds7
 Max-Forwards: 70
 From: LittleGuy <sip:UserB@biloxi.com>;tag=8321234356
 To: BigGuy <sip:UserA@atlanta.com>;tag=9fxced76sl
 Call-ID: 3848276298220188511@atlanta.com
 CSeq: 121 BYE
 Content-Length: 0
```

After the conversation UserB wishes to end the call. It sends BYE request to UserA.

Notice that CSeq of UserB has totally different sequence. That is because UserB maintains its own CSeq numbering independently.

”Basic Call” call flow




```
  SIP/2.0 200 OK
   Via: SIP/2.0/UDP
client.biloxi.com:5060;branch=z9hG4bKnashds7
    ;received=192.168.200.201
   From: LittleGuy <sip:UserB@biloxi.com>;tag=8321234356
   To: BigGuy <sip:UserA@atlanta.com>;tag=9fxced76sl
   Call-ID: 3848276298220188511@atlanta.com
   CSeq: 121 BYE
   Content-Length: 0
```

UserA sends 200 OK and conversation is ended. Note that method name (BYE) in CSeq header helps UserB to know that UserA sent OK for BYE request.

Requests invoke SIP methods

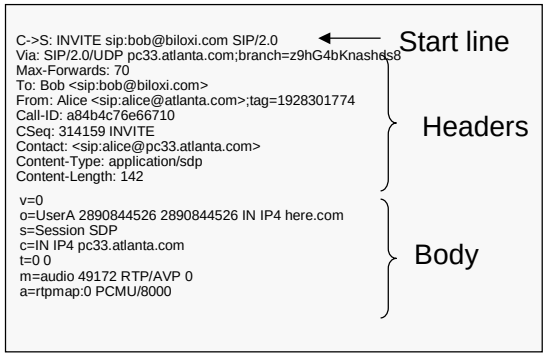
- SIP methods are invoked on servers when requests arrive:
  - A REGISTER request sends location information of users to Registrars, registers with the location service
  - An INVITE request invites a user to participate in a session or conference
    - The message body contains a description of the session (usually SDP)
  - ACK requests are used to confirm responses for INVITE, for reliable message exchanges
  - CANCEL requests cancel the pending request of the session
  - BYE requests are used to terminate active sessions
    - Any party of the session can send it
  - OPTIONS requests are used to query information about servers' capabilities
  - PRACK requests are used to confirm provisional responses

SIP responses are classified by first digit

- HTTP look-alike
- Hierarchically organized three digit codes: status code - text associated with the code
- Provisional and final responses:
  - 1xx responses are informational messages e.g., 180 Ringing
  - 2xx response shows a successful transaction e.g., 200 OK
  - 3xx responses are redirect messages e.g., 301 Moved Permanently
  - 4xx responses indicate errors in requests e.g., 400 Bad Request
  - 5xx responses indicate server errors e.g., 500 Version not supported
  - 6xx responses indicate global failures e.g., 600 Busy everywhere

SIP Message Format

- START-LINE
  - SIP version used
  - In requests: address and method used
  - In responses: status code
- HEADERS
  - Information about call
- BODY (payload)
  - Usually SDP message



In this example a typical INVITE message from Alice to Bob is shown. Alice is registered at domain atlanta.com, and has sip address sip:alice@atlanta.com.

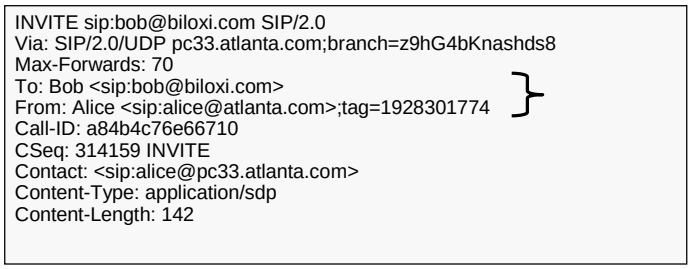
Alice tries to invite Bob at his sip address bob@biloxi.com

Headers will be explained later. It is important to know that there are several mandatory headers in every SIP message. Headers always end with Content-Lenght header, that shows how long is the body of the message (in bytes)

In this example, body of the message is SDP (Session Description Protocol). It gives the description of the session that Alice wants to establish (types of codecs, session parameters, etc).

To and From header fields

- **To: specifies the logical call destination**
- **From: specifies the logical call source**
- **Present in all SIP messages**



To: header specifies the logical call destination. In other words, this is the recipient's address. Optional parameter "display-name" (in this case "Bob") is just for human-user interface, while sip address is sip:bob@biloxi.com

From: header is the logical address of the originator of the request. "Logical" address means that it does not necessarily mean that it is the current address where Alice is reachable. In this case, Alice registered herself, and is contactable on sip:alice@pc33.atlanta.com. But Alice also has her permanent logical sip address sip:alice@atlanta.com. Proxy server will forward request to her logical addresses to her current address.

"tag" parameter is used in 'To' and 'From' header fields. Tag is used to identify a dialog* between parties. The dialog is identified by combination of tags from both parties (in 'To' and in 'From' heders), plus Call-ID parameter.

In this example, Alice's User Agent added 'tag=1928301774' (which is unique value, created by Alice's User Agent, and it is always different)

Note: In this example there is no tag in To: field. The reason for this is that this message is initial INVITE, and Bob's User Agent will create tag when receives message.

*Dialog – a peer-to-peer SIP relation between two Uas that persist from some time. It is identified by call identifier (Call-ID), local tag (in From: header)

Call-ID and CSeq header fields

- **Call-ID: It helps to uniquely identify a particular SIP dialog or registration**
  - It helps to match requests and responses
  - It helps to detect duplicated messages
- **CSeq: It is a number that uniquely identifies the transaction in a call**
- **Present in all SIP messages**

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142
```


Call-ID header consists of locally unique number (generated by User Agent) and @domain parameter, making it globally unique. The algorithm for making locally unique number is implementation issue. There are specifications available how to make algorithms in a way that will give unique numbers at the end.

Call-ID, together with tags in From: and To: headers identifies particular dialog. (see previous slides's notes for details)

Cseq: header contains single decimal sequence number and the request method. This number helps to order thransactions within one dialog, and to help differentiate between retransmissions and new requests. For example, callee can receive one request with CSeq:31 INVITE. Callee may answer to this request (e.g. with "200 OK" response). If callee receives again INVITE with same CSeq: 31 INVITE, it will know that its response was lost. It is also sure that this is not reinvitation, because CSeq nuber is the same.

Content-Type and Content-Length header fields

- **Content-Type: It describes the media type of the message body**
- **Content-Length: The number of octets in the message <u>body</u>**
  - It is mandatory in all SIP messages.

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142
```

Raimo Kantola –S- 2003 Signaling Protocols 11 - 27

Content-Type header gives the media type of the body of the message. It must be present if there is some data in the body. Content-Type is header reused from HTTP and it has same values, e.g. text/html for html body or application/sdp for SDP.

Content-Length gives the size of the body. It does not include CLRF separating header fields and body (headers and body are always separated with one blank line).

If there is no body in the message, then Content-Length is set to 0.

Max-Forwards

- **Max-Forwards field must be used with any SIP method**
- **It limits the number for proxies or gateways on the way of SIP message to the destination.**

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142
```


This field is used to prevent looping of the message. Every proxy that accepts the message will decrease this field by 1. If it reaches 0, then message is discarded, and message ”483 Too Many Hops” is sent to originator.

Every User Agent Client must insert Max-Forwards header field into each request. Value to be set is recommended to be 70. This number is big enough to prevent message being discarded unnecessarily (if message cannot reach destination in 70 hops, it is assumed that something is not configured properly anyway).

VIA header indicates path taken by the request so far

- **Branch parameter is used to detect loops**
- **Contains transport protocol, client's host name and possibly port number, and can contain other parameters**

```
INVITE sip:bob@192.0.2.4 SIP/2.0
Via: SIP/2.0/UDP server10.biloxi.com;branch=z9hG4bK4b43c2ff8.1
Via: SIP/2.0/UDP bigbox3.site3.atlanta.com;branch=z9hG4bK77ef4c2312983.1
 ;received=192.0.2.2
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
 ;received=192.0.2.1
Max-Forwards: 68
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142
```


Via: header is added by every entity that the request passes. It also determines the path for the response.

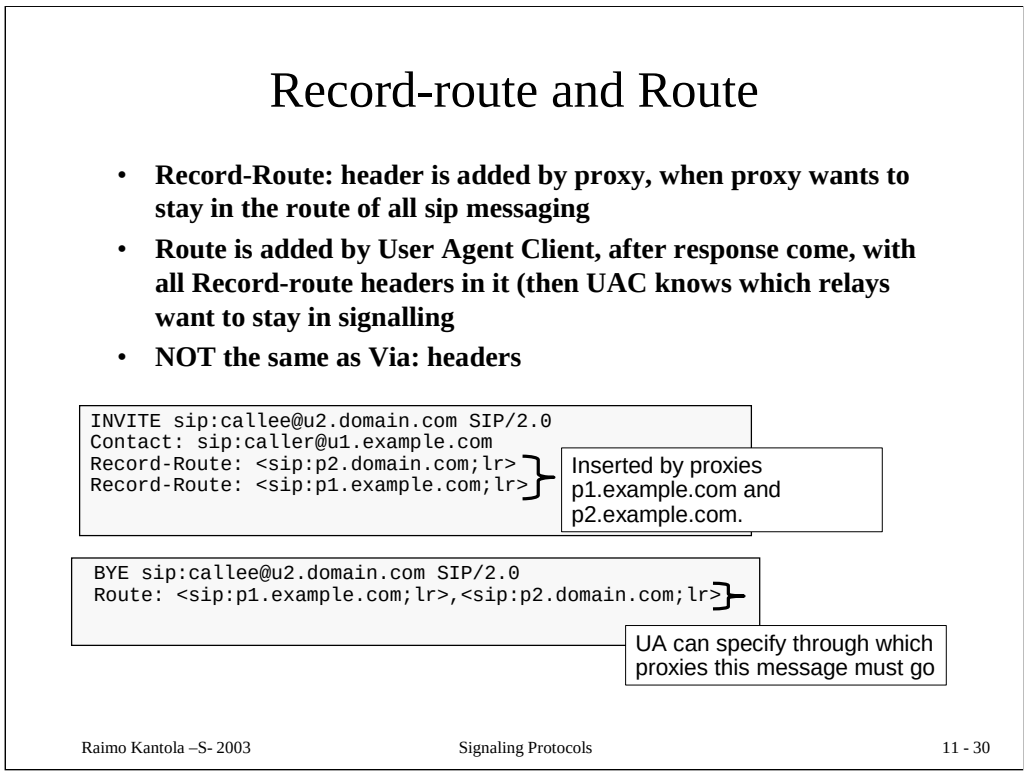
Every relay on the path of the request will put its address in the topmost Via: header.

In the response, when a relay receives the message, it will check if the topmost Via: header is its address. It can determine what is particular call by branch parameter.

It will then examine where to send response back (topmost Via: header value, after removing Via: with its own address)

Generally, for the requests every proxy on the way adds one Via: header with its address as value.

In sending response back, Via: header value is used for routing the response back.



(explanation from RFC 3261) U1 sends:

```
      INVITE sip:callee@domain.com SIP/2.0
      Contact: sip:caller@u1.example.com
```

to P1. P1 is an outbound proxy. P1 is not responsible for domain.com, so it looks it up in DNS and sends it there. It also adds a Record-Route header field value:

```
      INVITE sip:callee@domain.com SIP/2.0
      Contact: sip:caller@u1.example.com
      Record-Route: <sip:p1.example.com;lr>
```

P2 gets this. It is responsible for domain.com so it runs a location service and rewrites the Request-URI. It also adds a Record-Route header field value. There is no Route header field, so it resolves the new Request-URI to determine where to send the request:

```
      INVITE sip:callee@u2.domain.com SIP/2.0
      Contact: sip:caller@u1.example.com
      Record-Route: <sip:p2.domain.com;lr>
      Record-Route: <sip:p1.example.com;lr>
```

The callee at u2.domain.com gets this and responds with a 200 OK:

```
      SIP/2.0 200 OK
      Contact: sip:callee@u2.domain.com
      Record-Route: <sip:p2.domain.com;lr>
      Record-Route: <sip:p1.example.com;lr>
```

The callee at u2 also sets its dialog state's remote target URI to sip:caller@u1.example.com and its route set to:

```
      (<sip:p2.domain.com;lr>,<sip:p1.example.com;lr>)
```

This is forwarded by P2 to P1 to U1 as normal. Now, U1 sets its dialog state's remote target URI to sip:callee@u2.domain.com and its route set to:

```
      (<sip:p1.example.com;lr>,<sip:p2.domain.com;lr>)
```

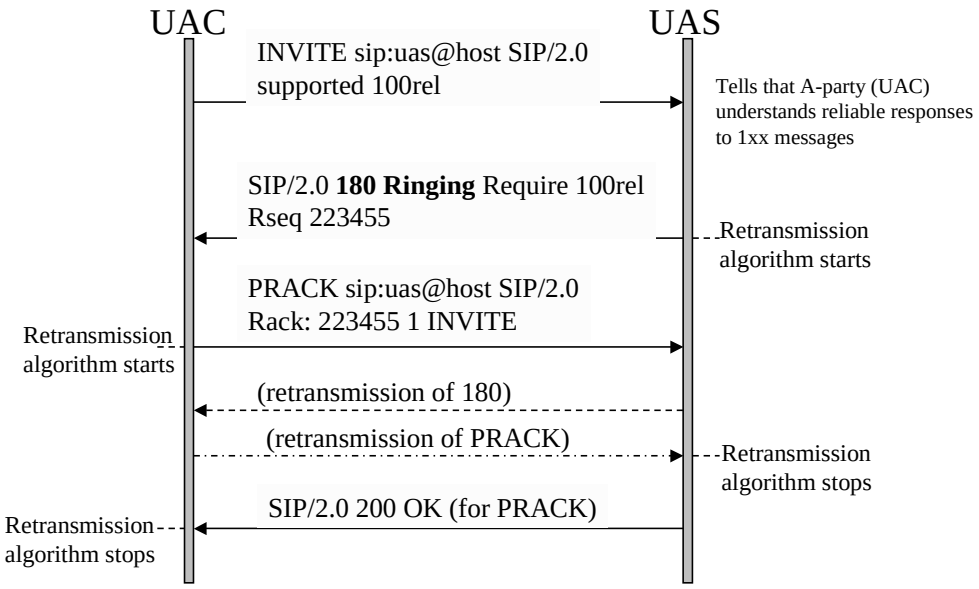
SIP Extensions

- Needed to satisfy additional requirements
- Must conform to design rules
- SIP is not intended to solve every problem (another protocol might be used instead)

Feature Negotiation (OPTIONS)

- *Supported* features can be specified in request and response
  - **Supported** UAC and UAS tell features they support
- *Required* features can be specified in request and response
  - **Require** UAC tells UAS about required options
  - **Proxy-Require** required options for proxy/redirect servers
  - Many extensions use Require and Proxy-Require to specify their support
- New methods can be added without changing the protocol
  - server can respond with **405 Not Supported**
  - returns list of supported methods in **Allow** header
  - client can ask which methods are supported using OPTIONS

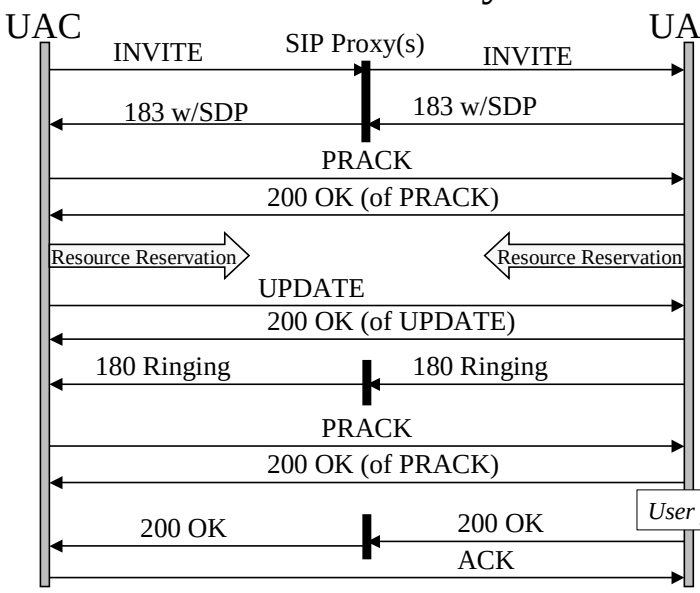
Reliable Provisional response in SIP



QoS support - UPDATE

- Usage rule for 183-Session-Progress
  - If “a=qos” appeared in SDP, UAS sends 183 with “Session: qos” and SDP
- Additional Method - UPDATE
  - If “a=qos” appeared in SDP with “confirm” attribute, UAS/UAC sends UPDATE with success/failure status of each precondition.
  - 200 OK must acknowledge the UPDATE message
- Additional Status Response - 580 Precondition Failure
  - If a mandatory precondition can’t be met, UAS terminates INVITE with this status response

Phone should not ring before QoS and Security are OK



W/SDP = "a-qos:" strength direction
"a-secure:" strength direction
strength = mandatory|optional |success|failure
direction = send|recv|sendrecv

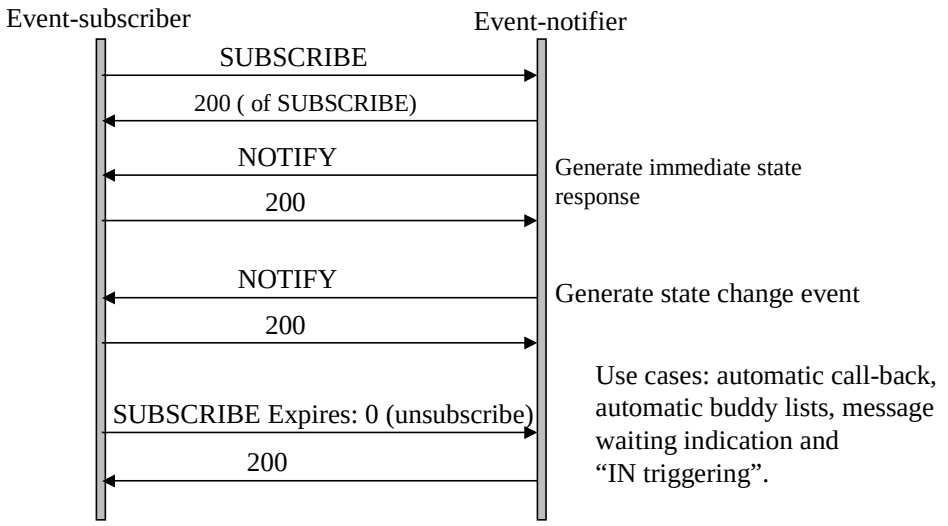
UPDATE confirms that precon-ditions are OK at the originator

PRACK method is used to ensure delivery of 183 and 180

Use case: 3G signaling!

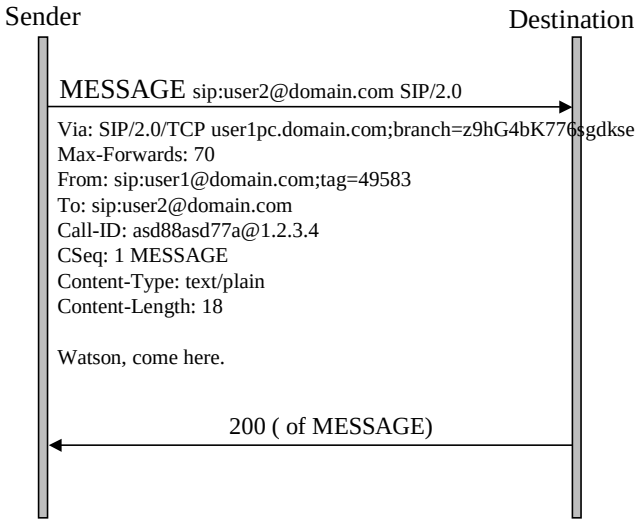
SDP = Session Description Protocol (carried in SIP message body)

SIP event notifications tell about remote significant events to the local party



Use cases: automatic call-back, automatic buddy lists, message waiting indication and "IN triggering".

SIP MESSAGE provides Instant Messaging capability in Pager mode



Note:
-body is MIME or S/MIME
- there is no dialog!

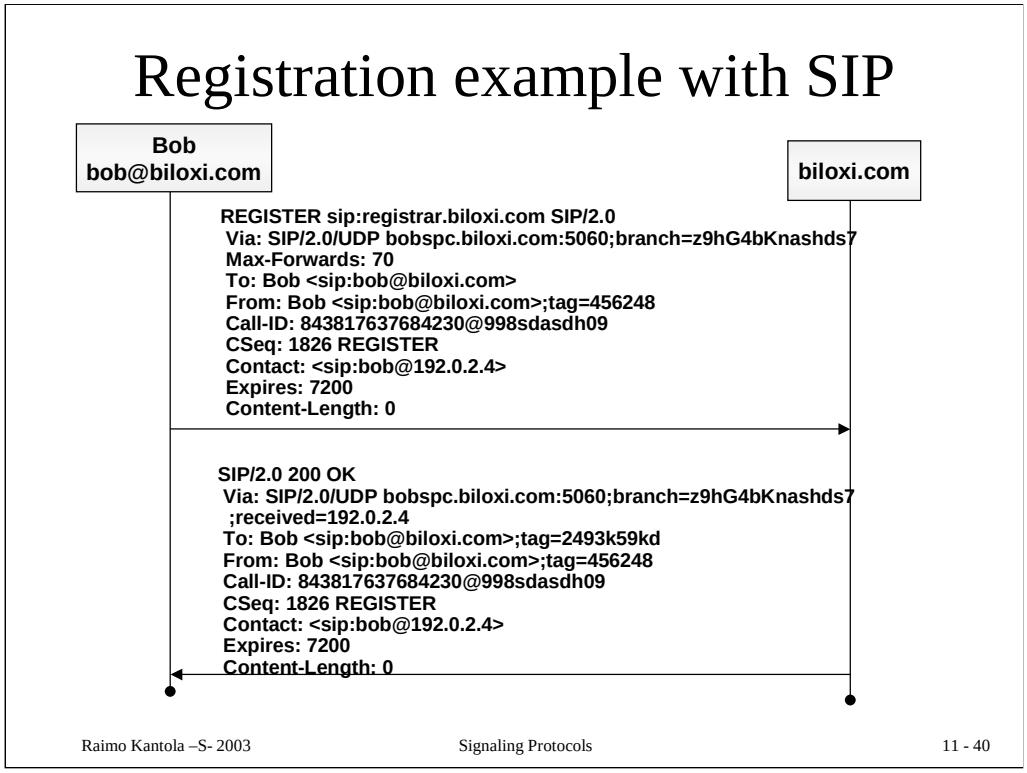
Use example:
-maps to SMS in GSM

Issue:
- authentication of sender and charging! Must use Proper security features

More SIP extensions

- MESSAGE
  - For instant messaging
- INFO
  - To transport mid-session information (very useful in SIP-PSTN gateways)
- Automatic configuration
  - DHCP or Service Location Protocol (SLP)
- Caller Preferences
  - New headers: Accept-Contact, Reject-Contact, Request-Disposition
- REFER
  - For session transfer (Refer-To: and Reffered-By:)
- …

Call Setup Examples based on Generic SIP

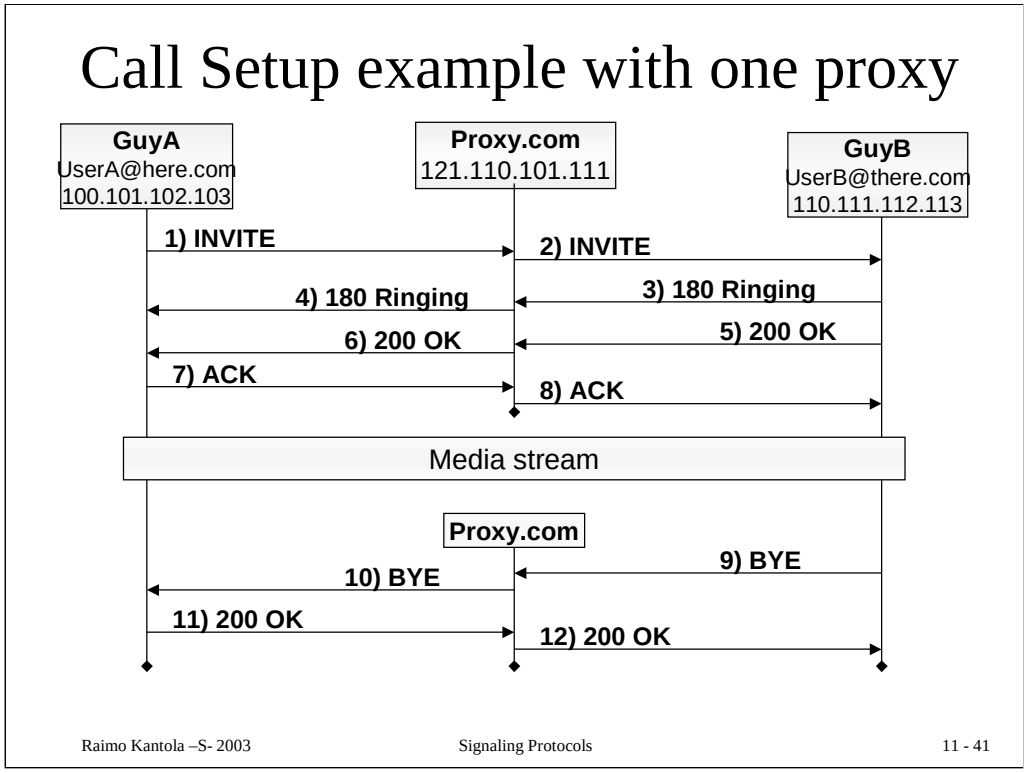


In this example Bob's User Agent performs a successful registration to a registrar whose domain name is biloxi.com.

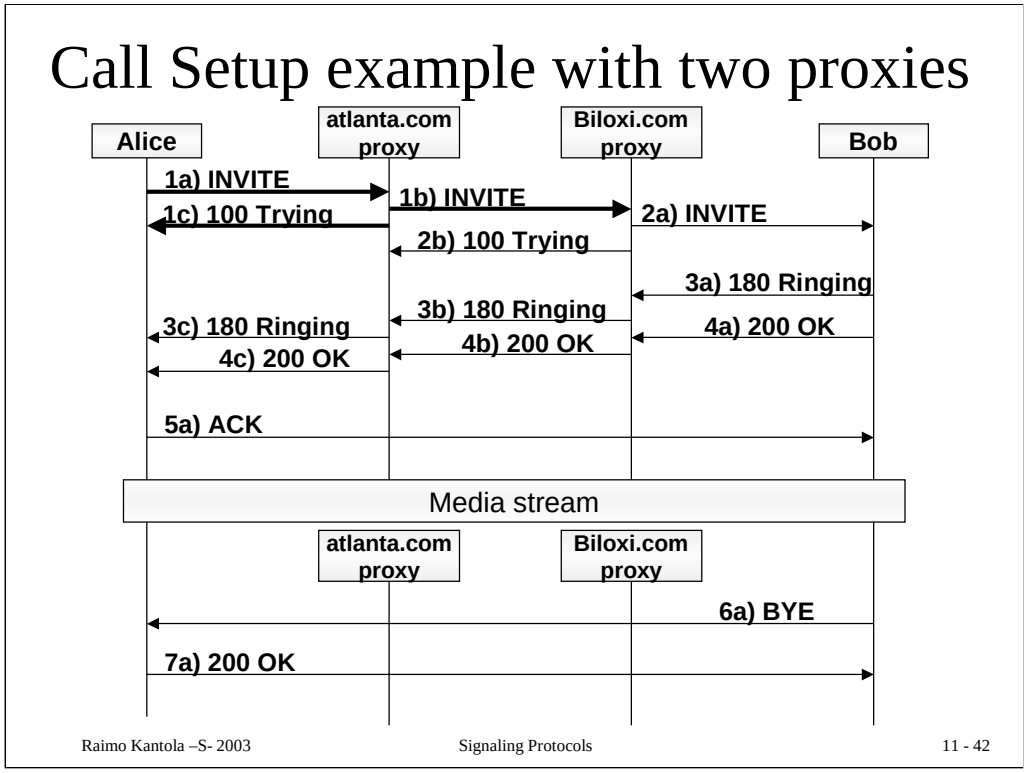
Registration is set to expire after two hours (7200 seconds). It may be seen in header Expires:

Registrar answers with 200 OK response, meaning that registration was succesfull.

(example from RFC3261)



In this example GuyA (UserA@here.com at 100.101.102.103) performs a successful call (session) initiation to GuyB (UserB@there.com at 100.101.102.103). For the sake of simplicity in the example both party is registered the same SIP proxy (proxy.com).



In this example Alice, who is registered to atlanta.com, performs a successful call (session) initiation to Bob, who is registered to biloxy.com.

1a) Alice -> atlanta.com proxy

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142
```

(Alice's SDP not shown)

Alice's UA sets Max-Forwards to 70. INVITE is addressed to logical SIP address, and Alice relies on proxy to find Bob. Alice is currently at pc33.atlanta.com and she puts current address in header Contact:

1b) INVITE atlanta.com proxy -> biloxi.com proxy

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP bigbox3.site3.atlanta.com
;branch=z9hG4bK77ef4c2312983.1
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
 ;received=192.0.2.1
Max-Forwards: 69
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142
```

(Alice's SDP not shown)

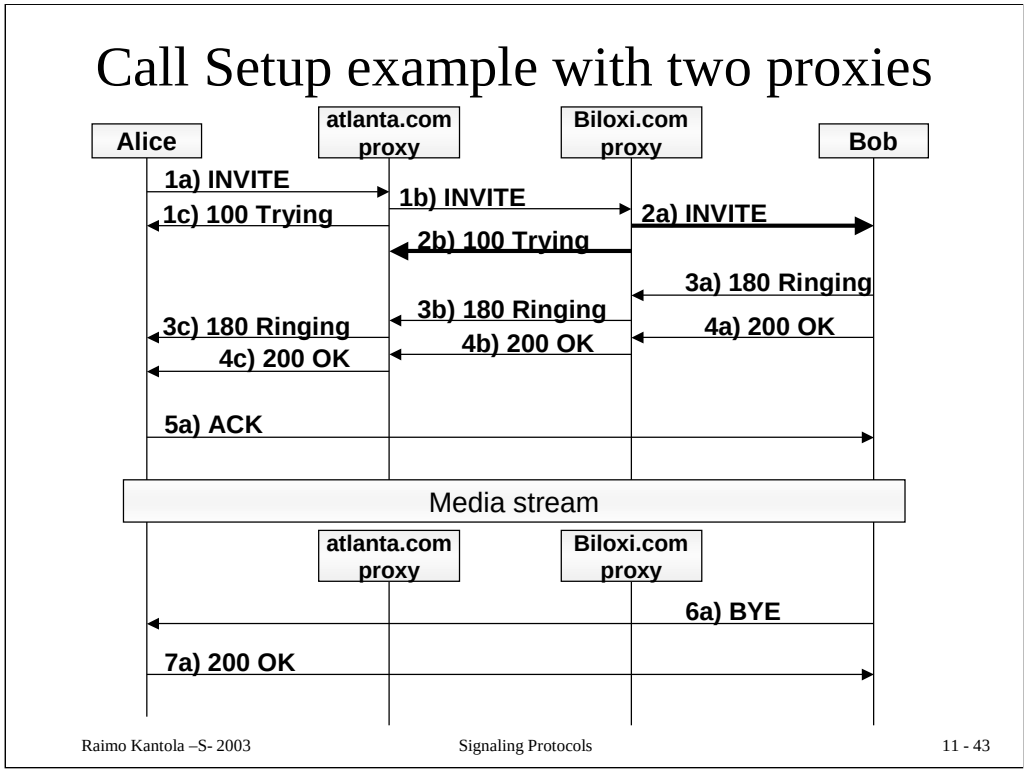
Atlanta.com proxy decreases Max-forwards by 1. Adds Via: header (puts its own address and branch there)

Atlanta.com leaves all other headers unchanged and resolves Bob's domain proxy. It then sends INVITE to biloxi.com proxy

1c) 100 Trying atlanta.com proxy -> Alice

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
```

After atlanta.com has forwarded INVITE to biloxi.com proxy, it sends 100 Trying to Alice. It will make her User Agent aware that call establishment is in the process (everything goes OK). It will also prevent Alice from retransmitting INVITE



In this example Alice, who is registered to atlanta.com, performs a successful call (session) initiation to Bob, who is registered to biloxy.com.

2a) biloxi.com proxy -> Bob

```
INVITE sip:bob@192.0.2.4 SIP/2.0
Via: SIP/2.0/UDP server10.biloxi.com;branch=z9hG4bK4b43c
Via: SIP/2.0/UDP bigbox3.site3.atlanta.com
;branch=z9hG4bK77ef4c2312983.1
 ;received=192.0.2.2
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
 ;received=192.0.2.1
Max-Forwards: 68
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142
```

(Alice's SDP not shown)

Biloxi.com proxy decreases max-forwards by 1. It also adds its address in Via: header, and assigns branch to it.

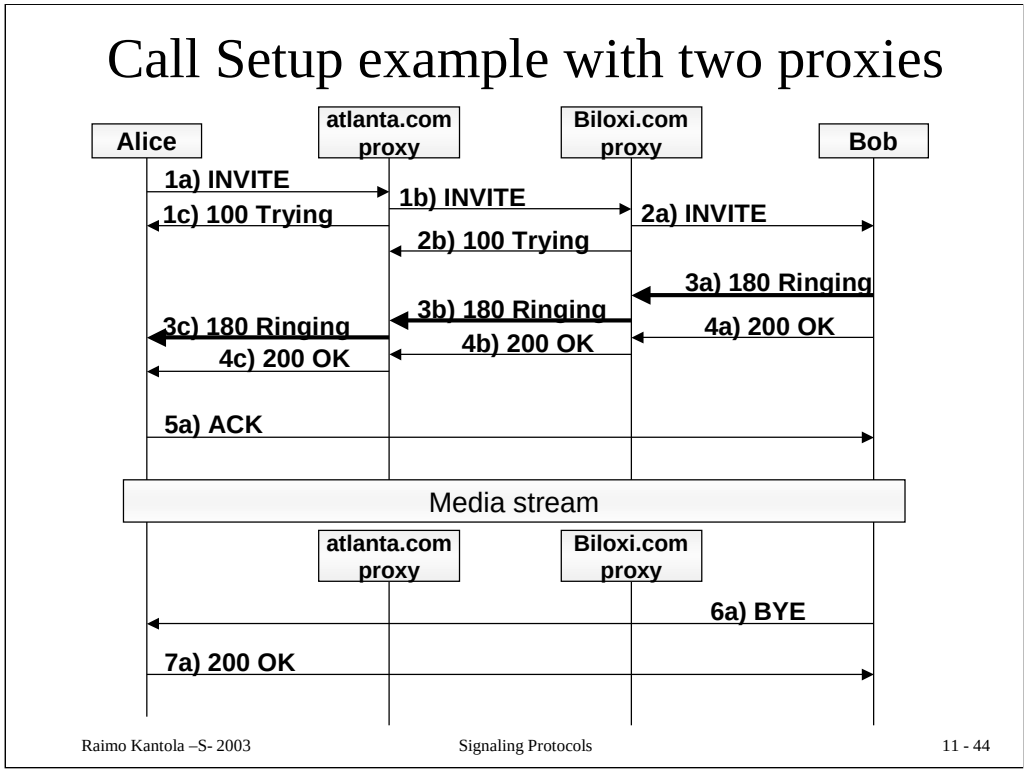
Note that in first line there is Bob's current address (sip:bob@192.0.2.4). It has been obtained from Location server.

2b) INVITE biloxi.com proxy -> atlanta.com proxy

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP bigbox3.site3.atlanta.com
;branch=z9hG4bK77ef4c2312983.1
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
 ;received=192.0.2.1
Max-Forwards: 69
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142
```

(Alice's SDP not shown)

After biloxi.com has forwarded INVITE to Bob, it sends 100 Trying to átlanta.com proxy.



In this example Alice, who is registered to atlanta.com, performs a successful call (session) initiation to Bob, who is registered to biloxy.com.

3a) Bob -> biloxi.com proxy

SIP/2.0 180 Ringing
Via: SIP/2.0/UDP server10.biloxi.com
;branch=z9hG4bK4b43c2ff8.1;received=192.0.2.3
Via: SIP/2.0/UDP bigbox3.site3.atlanta.com
;branch=z9hG4bK77ef4c2312983.1;received=192.0.2.2
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
;received=192.0.2.1
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
Contact: <sip:bob@192.0.2.4>
CSeq: 314159 INVITE
Content-Length: 0

Bob's UA sends 180 Ringing provisional response. It will send it back via the same route (usin Via: headers). Response will go to the topmost Via: address.

Also, Bob's UA adds tag to To: header. Now dialog is completely determined (with Call-ID, From: and To: tags)

3b) biloxi.com proxy -> atlanta.com proxy

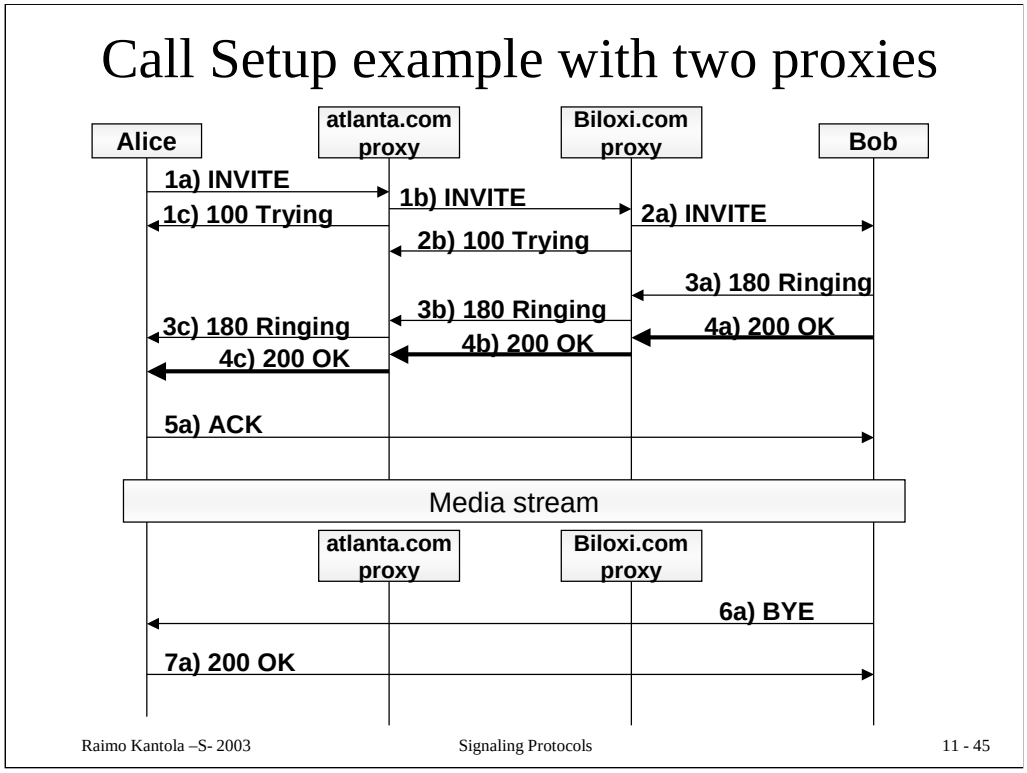
```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP bigbox3.site3.atlanta.com
;branch=z9hG4bK77ef4c2312983.1;received=192.0.2.2
Via: SIP/2.0/UDP pc33.atlanta.com
;branch=z9hG4bKnashds8 ;received=192.0.2.1
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
Contact: <sip:bob@192.0.2.4>
CSeq: 314159 INVITE
Content-Length: 0
```

From biloxi.com proxy to atlanta.com proxy.

3c) atlanta.com proxy -> Alice

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
 ;received=192.0.2.1
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
```

Alice receives 180 Ringing response. Now Alice's UA knows that Bob has been alerted.



4a) Bob -> biloxi.com proxy

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP server10.biloxi.com
;branch=z9hG4bK4b43c2ff8.1;received=192.0.2.3
Via: SIP/2.0/UDP bigbox3.site3.atlanta.com
;branch=z9hG4bK77ef4c2312983.1 ;received=192.0.2.2
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
 ;received=192.0.2.1
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:bob@192.0.2.4>
Content-Type: application/sdp
Content-Length: 131
```

(Bob's SDP not shown)

4b) biloxi.com proxy -> atlanta.com proxy

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP
bigbox3.site3.atlanta.com;branch=z9hG4bK77ef4c2312983.1
 ;received=192.0.2.2
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
 ;received=192.0.2.1
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:bob@192.0.2.4>
Content-Type: application/sdp
Content-Length: 131
```

(Bob's SDP not shown)

4c) 100 Trying atlanta.com proxy -> Alice

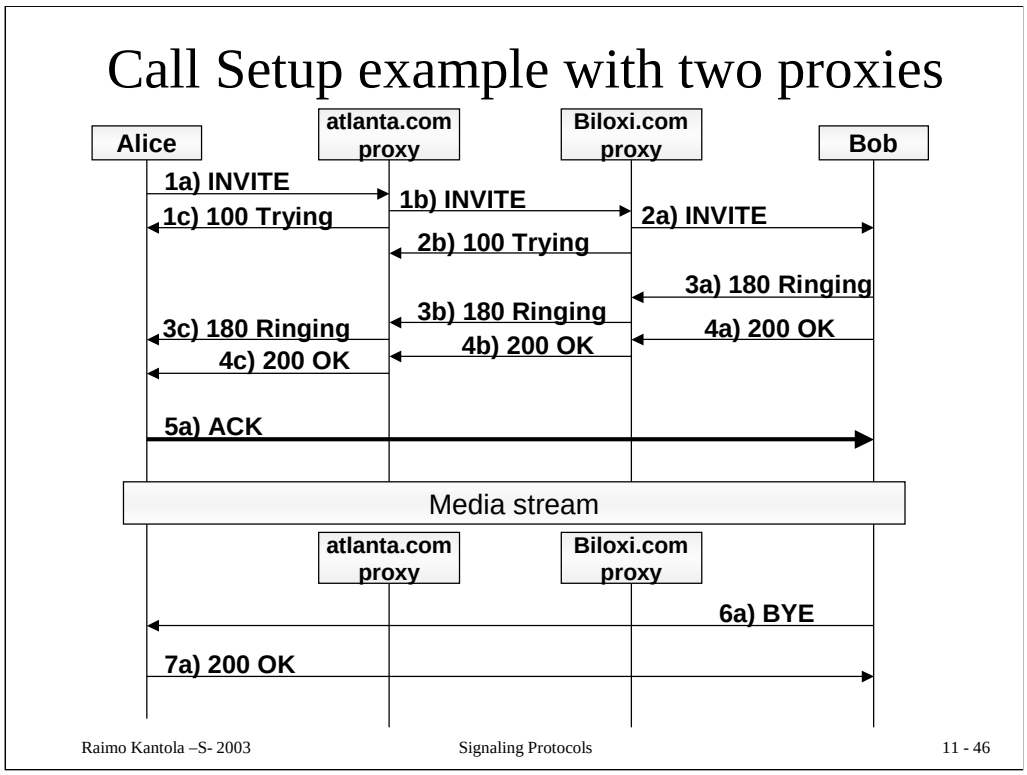
```
SIP/2.0 200 OK
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
 ;received=192.0.2.1
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
```

Bob accepts invitation and sends 200 OK final response. It will go through the same set of proxies (determined by Via: headers)

Bob puts his current addres in Contact: header. Alice is now able to contact him directly. Subsequent SIP messages may go directly to Bob, and not through proxies

Biloxi.com proxy forwards response to topmost via: header address. (it removed its own address in Via: header previously)

Alice receives 200 OK with Bob's session parameters in the message body (not shown here). If Alice can accept it, she will send ACK message back to Bob directly. Now Alice has Bob's address where he is contactable in Contact: header.

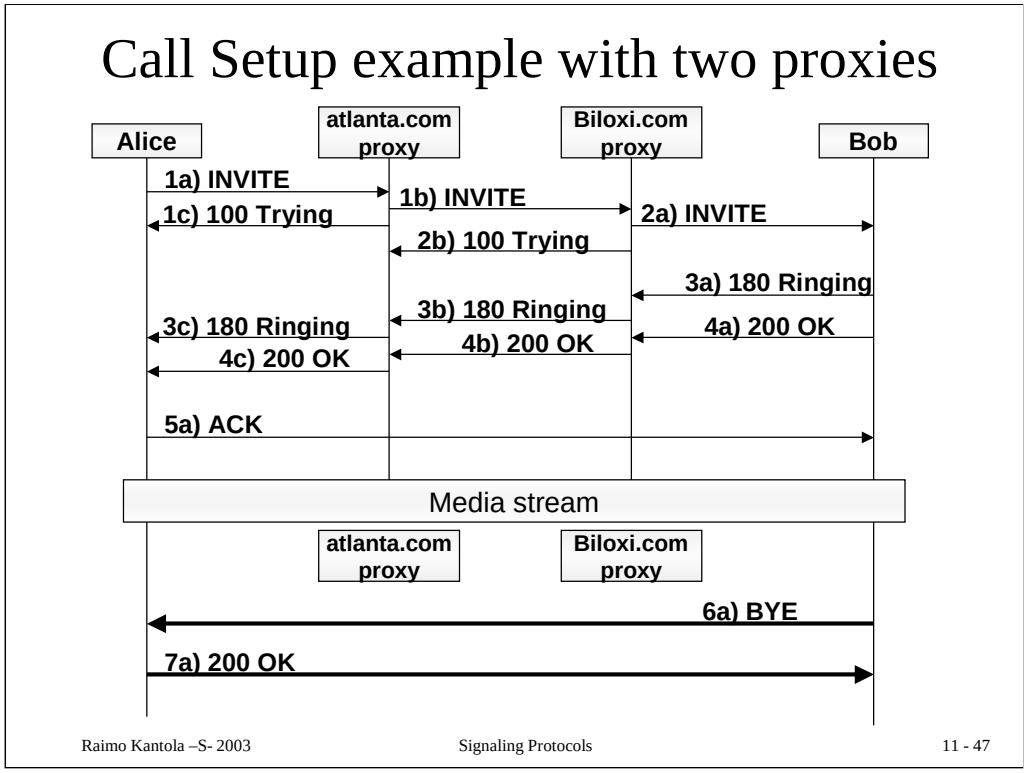


In this example Alice, who is registered to atlanta.com, performs a successful call (session) initiation to Bob, who is registered to biloxy.com.

5a) Alice -> Bob

```
ACK sip:bob@192.0.2.4 SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnas
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 ACK
Content-Length: 0
```

The media session between Alice and Bob is now established. They agreed on session parameters (described in SDP body)



6a) Bob -> Alice

```
BYE sip:alice@pc33.atlanta.com SIP/2.0
Via: SIP/2.0/UDP 192.0.2.4;branch=z9hG4bKnashds10
Max-Forwards: 70
From: Bob <sip:bob@biloxi.com>;tag=a6c85cf
To: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 231 BYE
Content-Length: 0
```

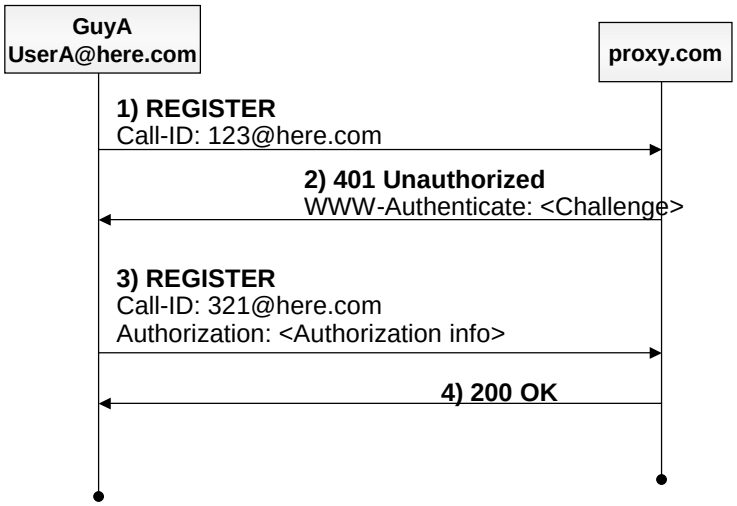
Bob after a while decides to disconnect. Bob's UA has its own CSeq sequencing (note) and From: and To: fields are swapped, because Bob is originating a request. But Bob still refers to the same dialog (can be seen by Call-ID and tags)

7a) Bob -> Alice

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.0.2.4;branch=z9hG4bKnashds10
From: Bob <sip:bob@biloxi.com>;tag=a6c85cf
To: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 231 BYE
Content-Length: 0
```

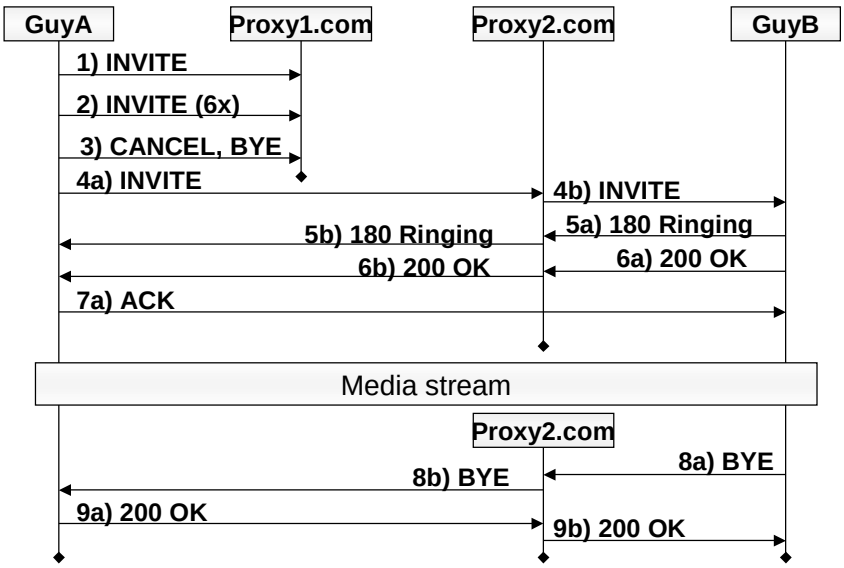
Alice acknowledges BYE, and call is over. This 200 OK will refer to the BYE request, and that can be seen from CSeq field, carrying BYE method name

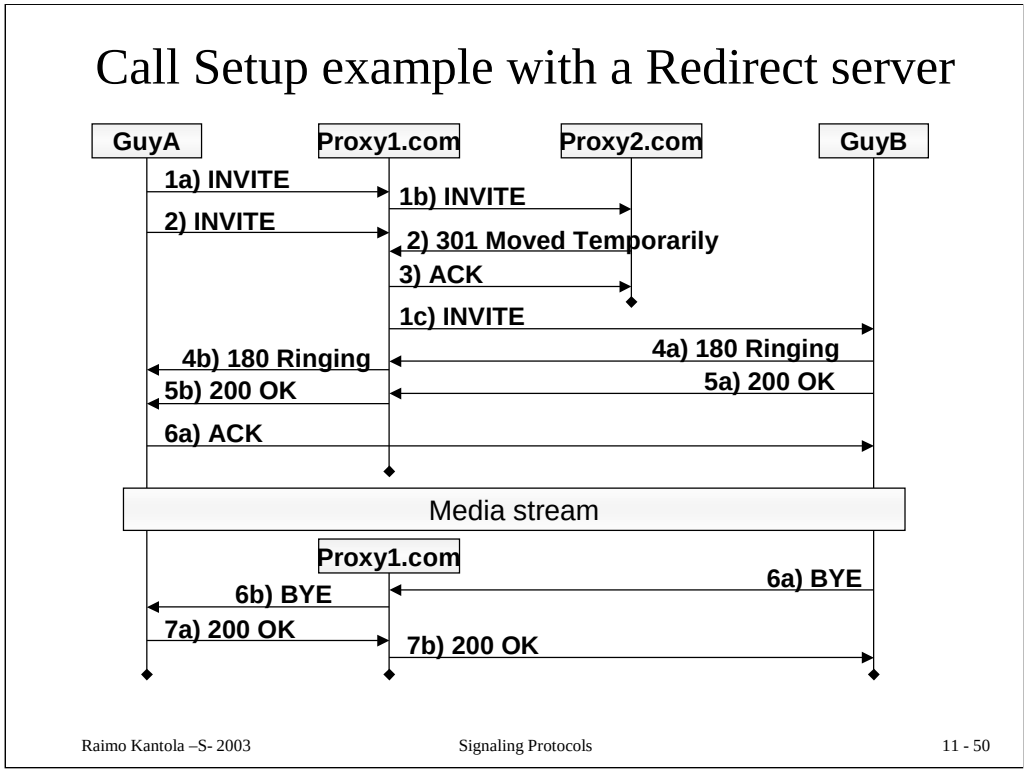
Registration example with SIP authentication



In this example GuyA (UserA@here.com at 100.101.102.103) performs a successful registration to a proxy whose domain name is proxy.com.

Call Setup example with a non-working proxy



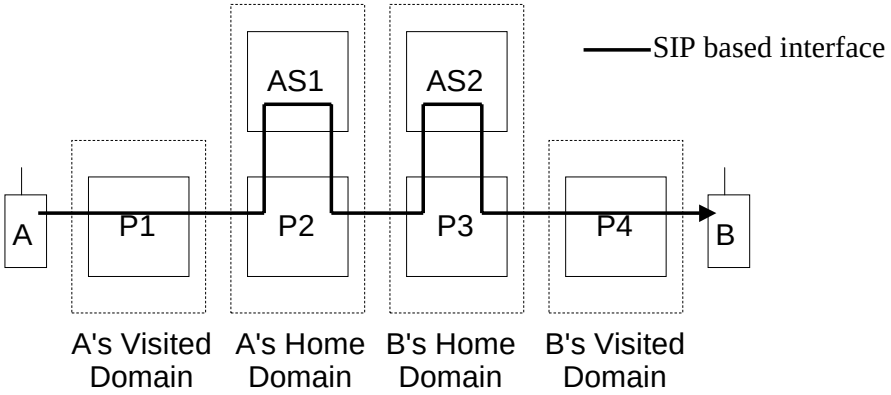


In this example GuyA, who is registered to proxy1.com, performs a successful call (session) initiation to GuyB, who is registered to proxy2.com.

Services use many protocols

- New services and more flexible service creation should differentiate IP Communications Network from PSTN
- Services should combine different forms of communication, thus multiple protocols are needed:
  - SIP for media sessions and session related services, subscriptions and notifications?, messaging?
  - HTTP for web and transactions
  - SMTP for e-mail
  - RTSP for media streaming
- The use of these protocols is orchestrated by the service logic: context is set up using SIP.

Routing and Service Model in 3G



P1, P4: Outbound Proxies

P2, P3: Registrar Proxies

AS1, AS2: Application Servers

NB: Also AS based on direct processing of call state: There is no Basic call state model like in IN

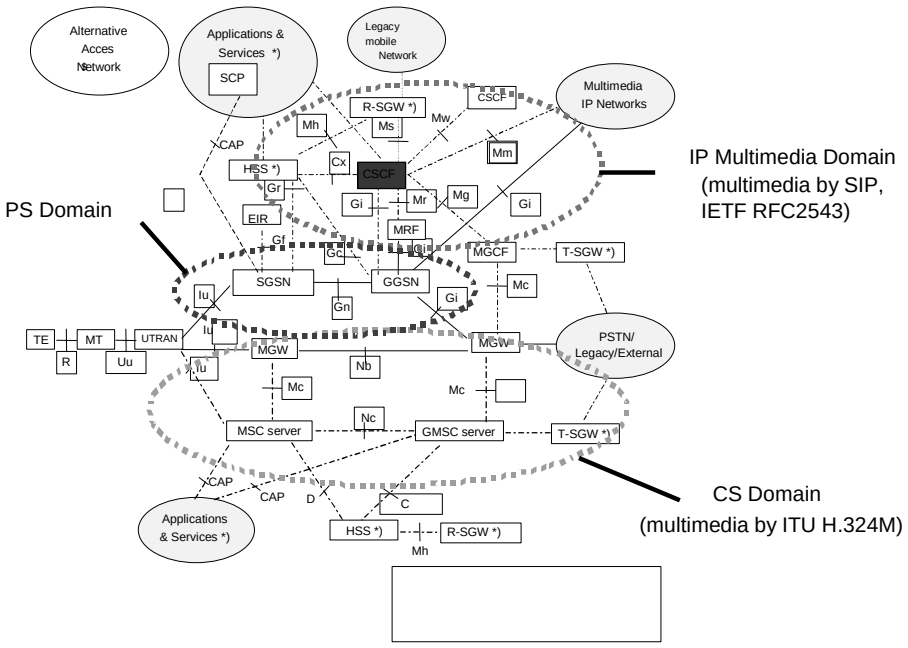
SIP Entities & Service Capabilities

- User Agent (= UAC + UAS)
  - Can run services, such as forwarding, filtering etc.
  - Not always connected (out of coverage/battery etc.)
- Redirect Server
  - Can do services that require only Request-URI change, e.g. translation, parameter addition etc.
- Proxy Server
  - Can change certain headers and stay in the signaling path
  - Forking, actions based on responses
- Back-to-Back User Agent (=both ways User Agent)
  - Can e.g. issue requests to a call leg or modify SDP
  - In many cases necessary

Application Server in 3G?

- Fuzzy Definition but has SIP+ interface!
- Can be a Redirect or Proxy Server or Back-to-Back UA
- The key is that it should be *programmable*
  - Routing based on service logic: what to do when user not registered or busy
  - URI translation: Reachability chains
  - Interfaces to other protocols: HTTP, SMTP, RTSP etc.
- Can be single purpose boxes, or multi-purpose boxes, or controllers who orchestrate things

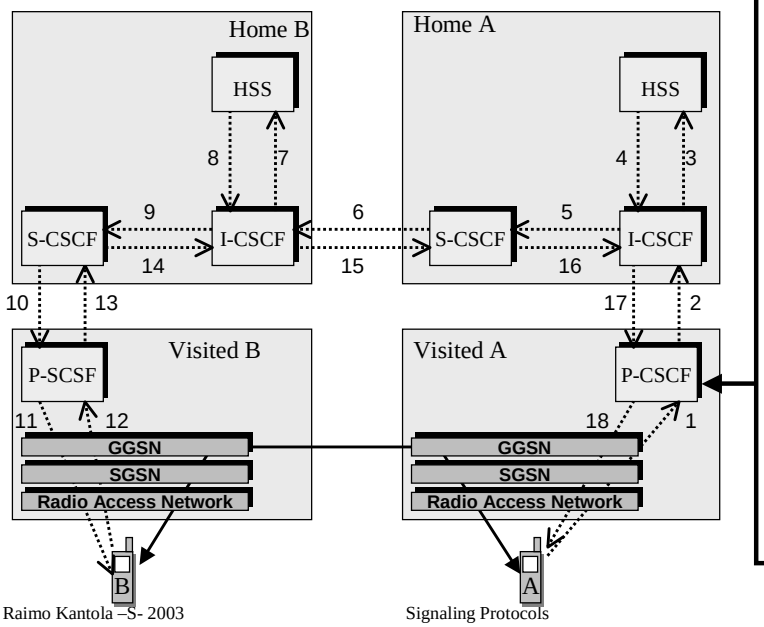
3GPP Network Model (preliminary: …



CSCF (Call/Session Control Function) is the primary SIP node in the network.

(from www.sipforum.org)

Different Kinds of CSCFs



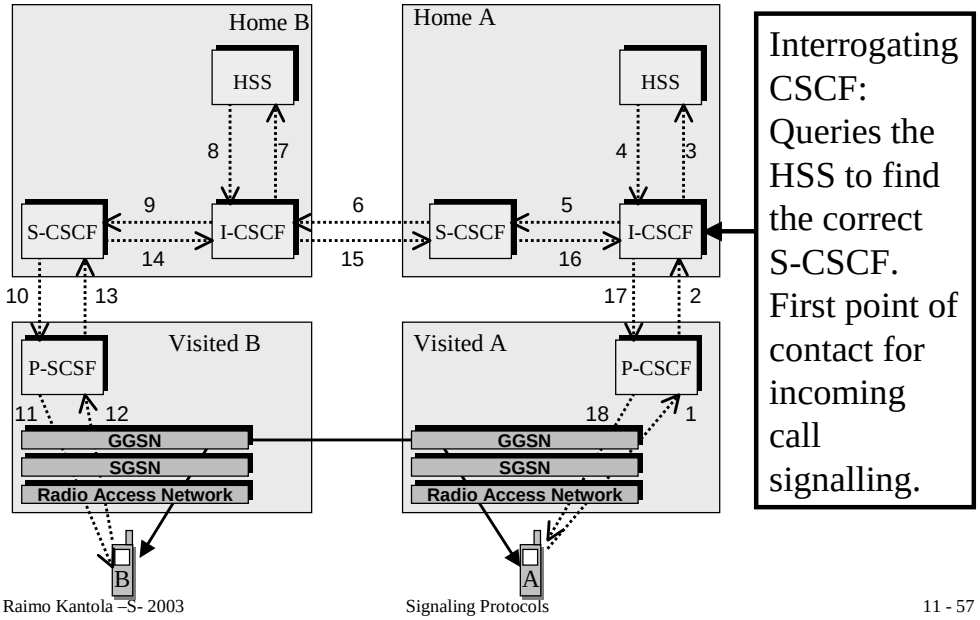
Currently, 3GPP has defined three different functional behaviors which the CSCF will exhibit.

The Proxy CSCF (P-CSCF) provides a first point of contact for the handset. All signaling to and from the handset goes through the P-CSCF. In terms of SIP, it behaves as an outbound proxy.

The main purpose for this node is to provide emergency service breakout and to do some basic message manipulation to enable the visited domain operator to provide locally sensitive services (e.g. traffic reports, directory services, etc). It also does simple number internationalization (which allows the support of local dialin plans).

It will probably also play a role in quality of service reservations.

Different Kinds of CSCFs

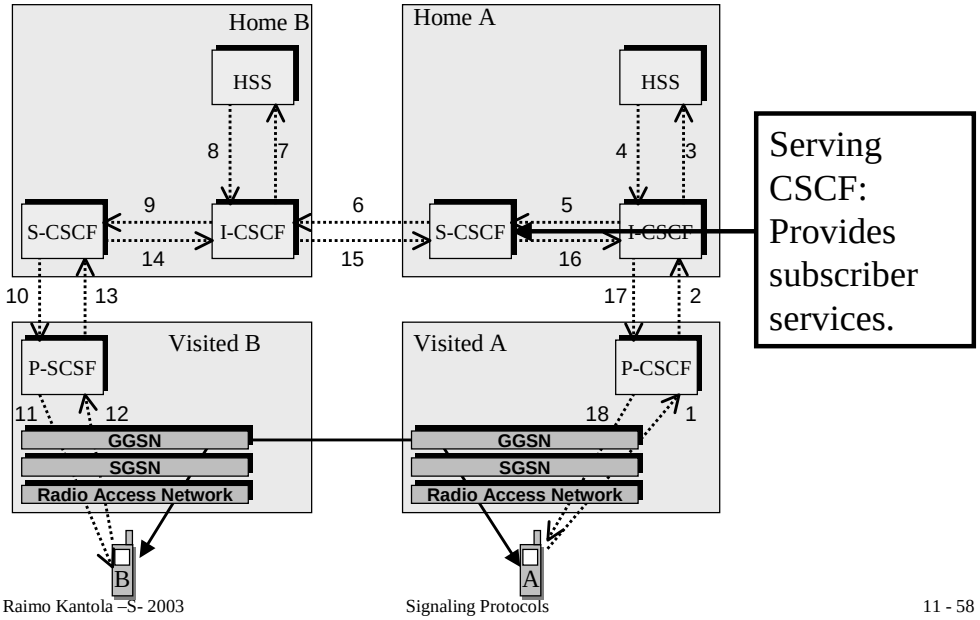


The Interrogating CSCF (I-CSCF) is mostly a load distribution node. Since DNS allows us simple statistical distribution among identical nodes, distributing load among the I-CSCFs is quite simple. But if all we relied on was statistical distribution, we wouldn't be able to allocate subscriptions on appropriate serving nodes according to their capabilities, nor would be be assured of the ability to keep call state information between transactions.

So, the I-CSCF, in conjunction with the HSS, allocates subscription information onto appropriate Serving CSCFs. The HSS keeps track of this information so that all transactions and all calls for the same user go through the same service node.

The HSS stores user profile information; it's somewhat similar to the HLR found in today's cellular networks.

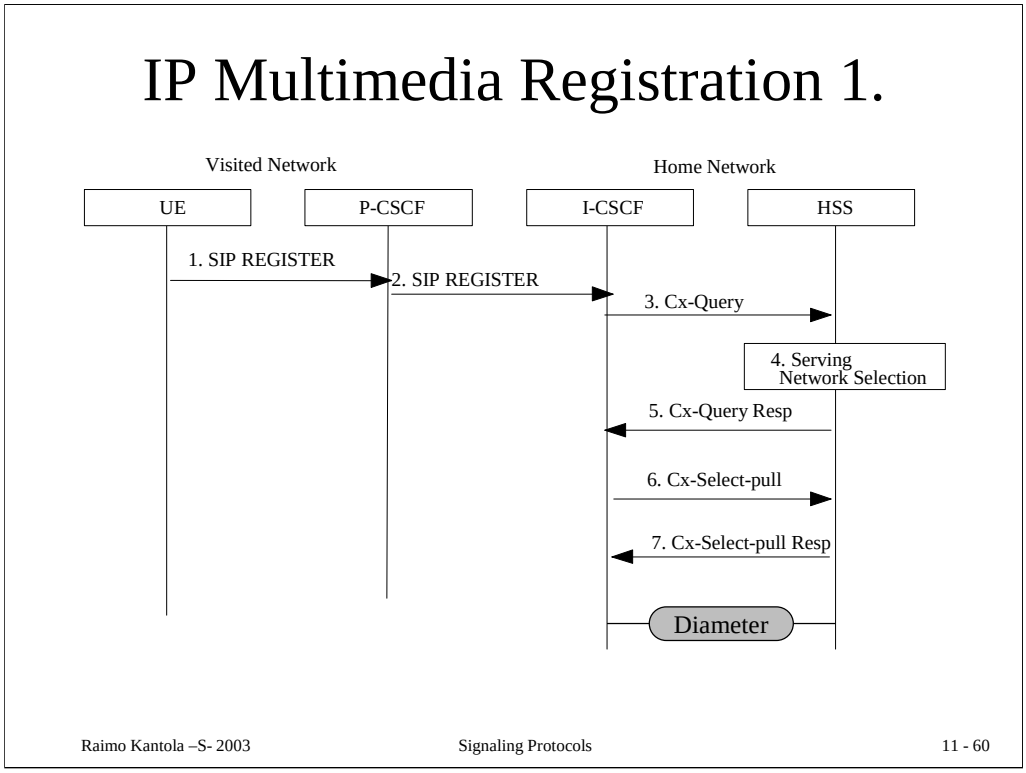
Different Kinds of CSCFs



The Serving CSCF (S-CSCF), quite simply, provides users services.

Of course, SIP allows the terminal to provide many services itself. The S-CSCF will be useful in providing, for example: call forwarding when the terminal is not available, call barring, centralized speed dial lists, VPN services, etc.

Appendix B – 3GPP IMS call flows



1. The UE has obtained a signalling channel through the access network, UE sends the Register information flow to the proxy (subscriber identity, home networks domain name).

2. The P-CSCF examines the “home domain name” to discover the entry point to the home network (I-CSCF).

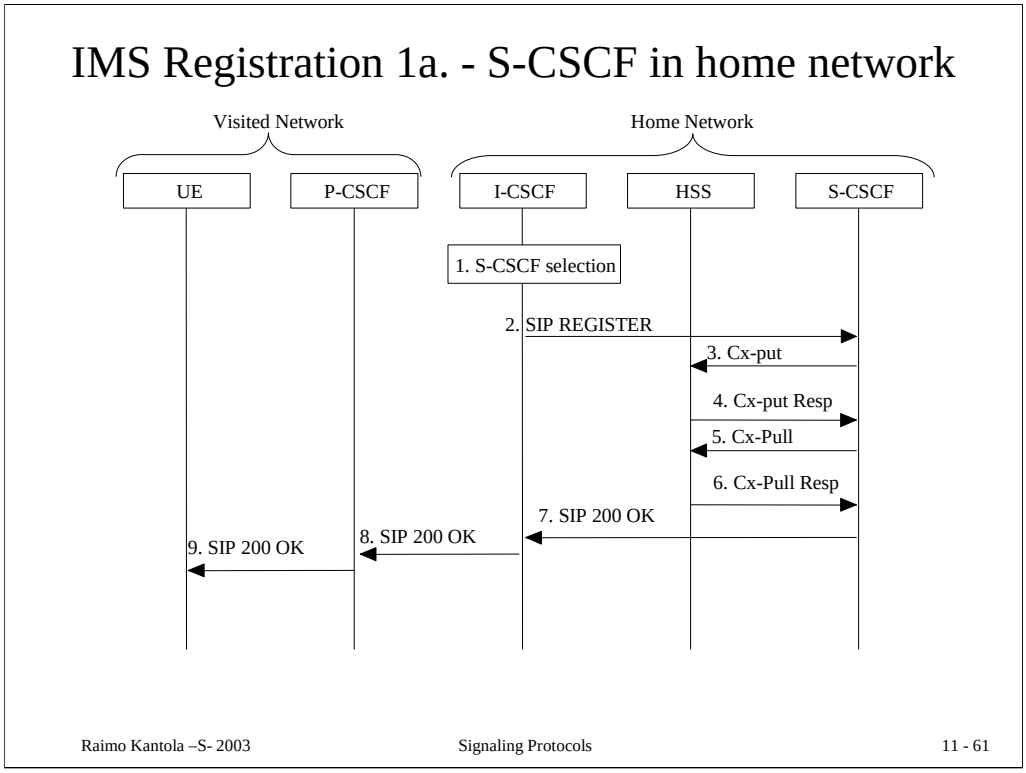
3. The I-CSCF shall send the Cx-Query information flow to the HSS (P-CSCF name, subscriber identity, home domain name, visited network capabilities, visited network contact name).

4. HSS selects whether the serving network is in the home network or the visited network

5. Indication of serving network selection is sent from the HSS to the I-CSCF

6. The I-CSCF sends Cx-Select-Pull (serving network indication, subscriber identity) to the HSS to request the information related to the required S-CSCF capabilities

7. The HSS sends Cx-Select-Pull Resp (required S-CSCF capbilities) to the I-CSCF.



1. The I-CSCF determines the name of an appropriate S-CSCF.

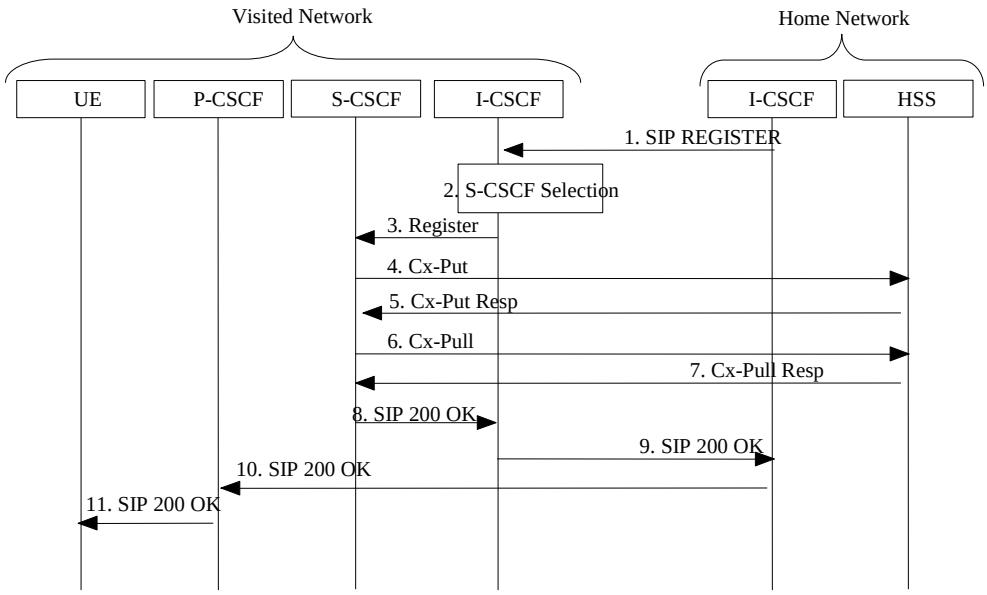
2. The I-CSCF sends the REGISTER (P-CSCFs "name" in the contact header, visited network capabilities, subscriber identity, visited network contact name) to the selected S-CSCF.

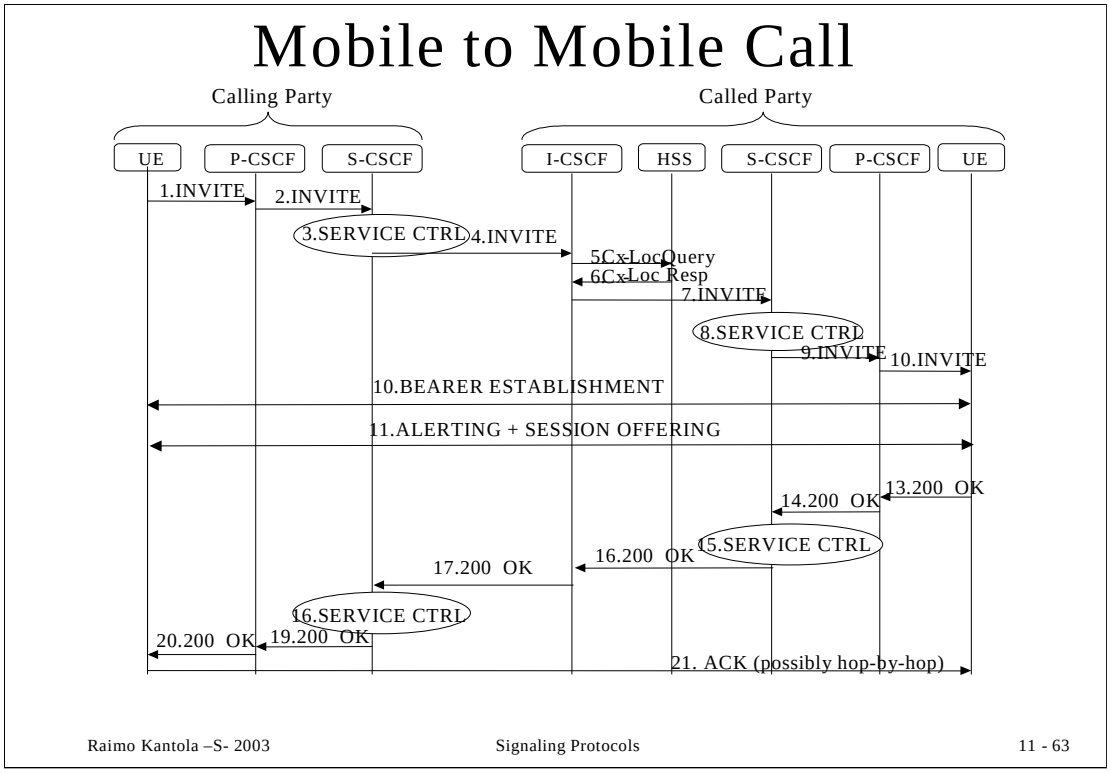
3,4. The S-CSCF sends Cx-Put (subscriber identity, S-CSCF name) to the HSS. The HSS stores the S-CSCF name for that subscriber. The HSS acknowledges Cx-Put.

5,6. The S-CSCF sends the Cx-Pull (subscriber identity) to the HSS in order to be able to download the relevant information from the subscriber profile to the S-CSCF. The S-CSCF stores the P-CSCFs name, as supplied by the visited network. The HSS returns the user information to the S-CSCF.

7,8,9. The S-CSCF returns 200 OK (serving network contact name, S-CSCF name) to the I-CSCF. The I-CSCF sends 200 OK (serving network contact name) to the P-CSCF. The P-CSCF stores the serving network contact name, and sends 200 OK to the UE.

IMS Registration 1b. - S-CSCF in visited network





The UE sends a INVITE (session destination) to the P-CSCF.

The P-CSCF forwards the INVITE (session destination) to the next hop name/address. In this case the next hop address is the S-CSCF.

The S-CSCF can read the information on whom originated the INV, and forwarded this based on the session destination. The S-CSCF forwards the invite message to the I-CSCF.

The I-CSCF sends ‘Cx-Location Query’ to the HSS to obtain the identity of the of the next hop which in this case is the S-CSCF.

The HSS sends the ‘Cx-Location Query Response’ to the I-CSCF.

The I-CSCF forwards the INVITE message to the S-CSCF for the called party.

The S-CSCF carries out Service Control for Called Party.

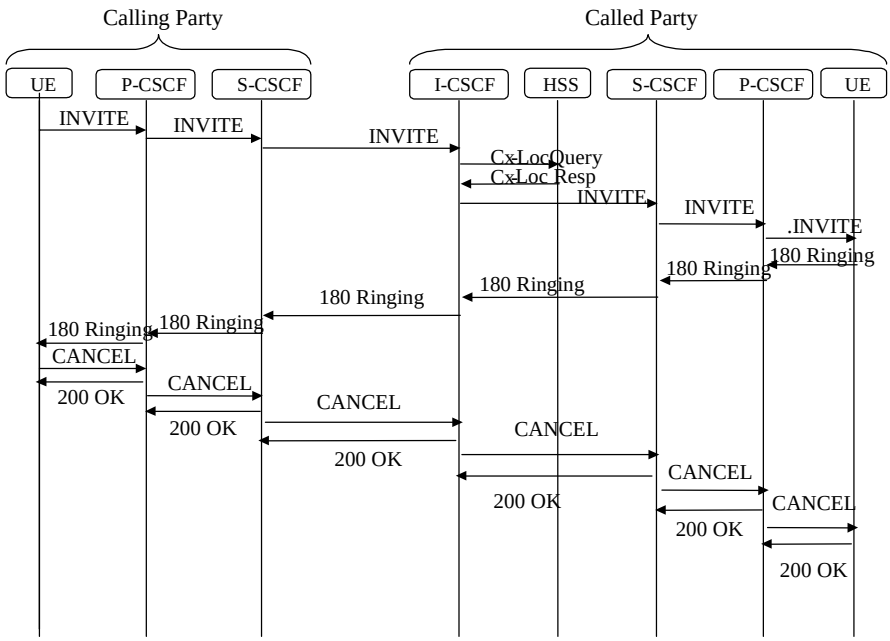
The S-CSCF forwards the INVITE message to the P-CSCF (Called Party)

The P-CSCF forwards the INVITE message to the UE (Called Party)

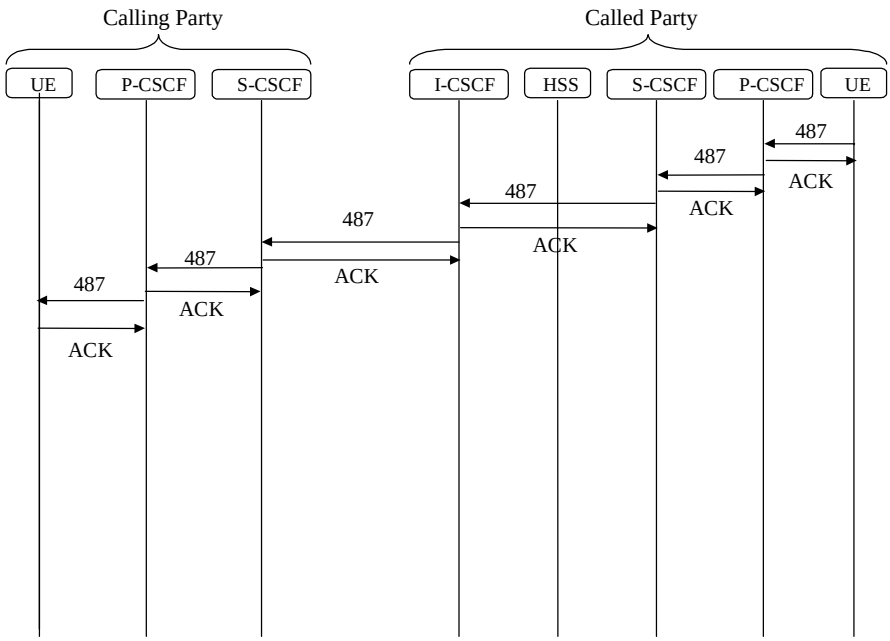
Bearer Establishment Process: Bearer reservation (temp establishment) which will require a 183 to carry the bearer information of the called party back to the calling party

Alerting/Session Offering Process

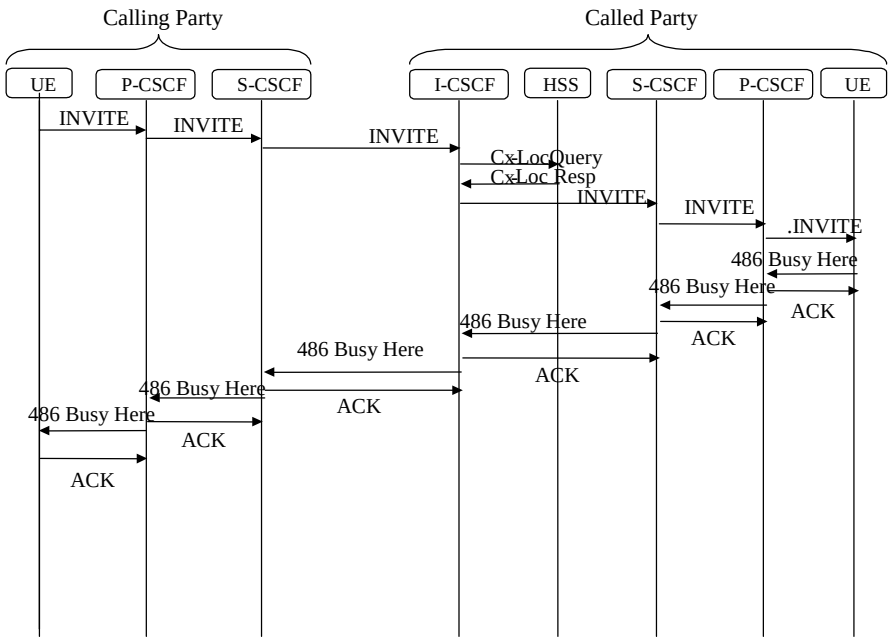
Call flow examples 1. - no answer



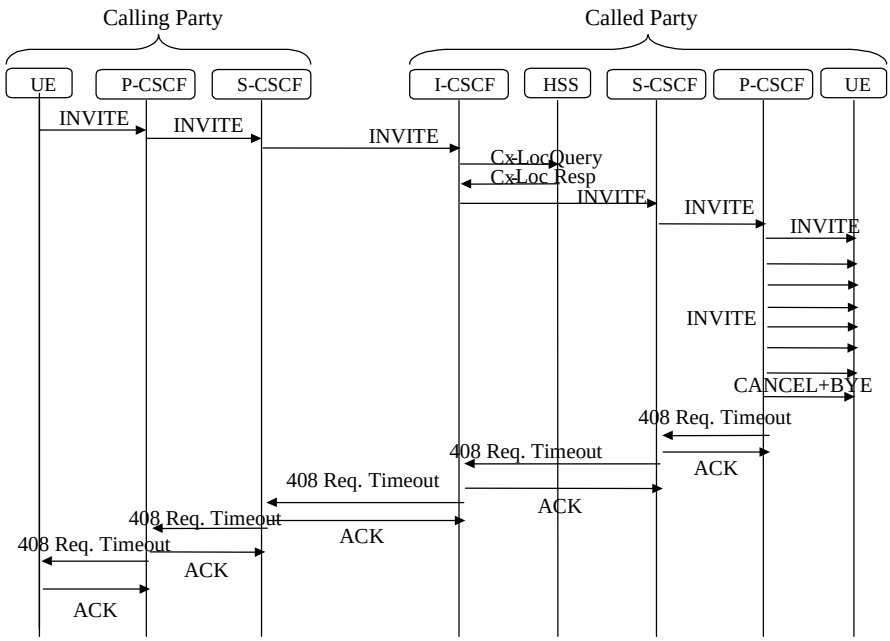
Call flow examples 1. - no answer 2.



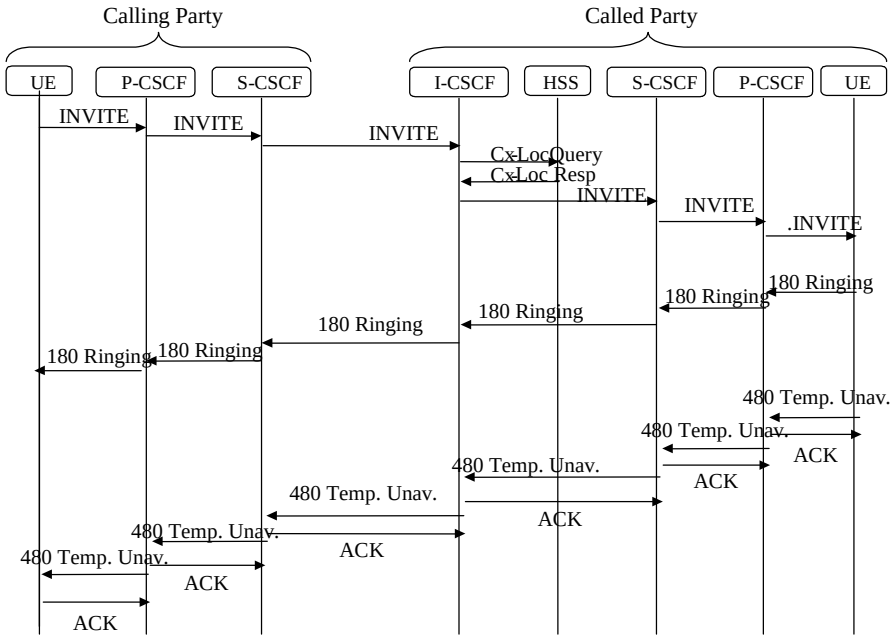
Call flow examples 2. - busy



Call flow examples 3. - no response



Call flow examples 4. - temporarily unavailable



How to Program Services

- Call Processing Language
- SIP CGI
- SIP Servlets
- SIP JAIN
- Soft SSF and INAP/CAP
- Parlay
- OSA

==>
There will be many competing ways to implement services!

=> Whatever… Different abstraction levels

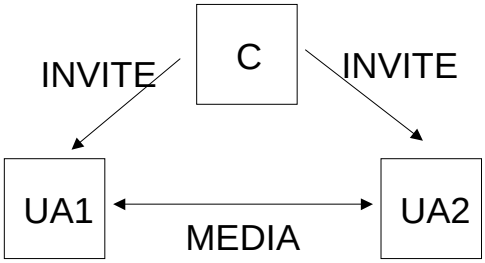
The claim is that it should be as open as flexible as creating services in the web these days

Server types for different services

- Media Server (SIP, RTSP, HTTP)
  - Announcements, IVR, Voicemail, Media on demand
- Conferencing Server (SIP)
  - Media mixer
- Presence Server (SIP)
  - Users status info, capabilities, willingness to communicate
- Web Server (HTTP), E-mail Server (SMTP), Messaging Server (SIP?), Text-to-Speech Server etc.
- Controller Server
  - Co-ordinates the overall service

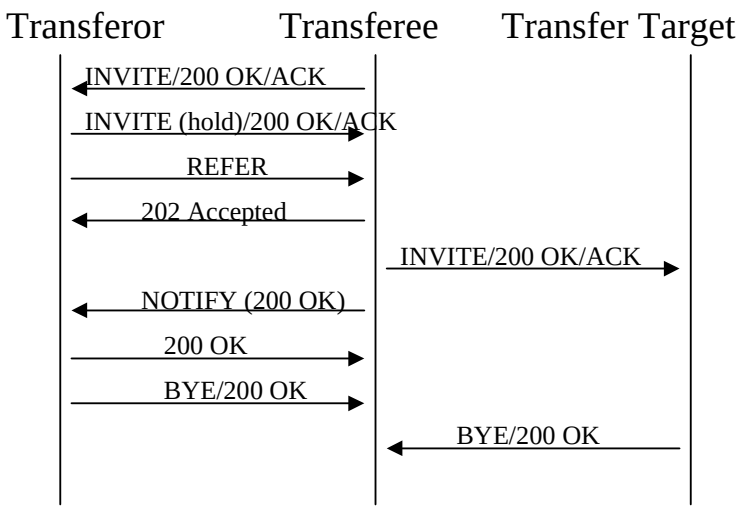
=> Server resources can be addressed by URLs, no need for tight coupling a la MGCP/Megaco

Third Party Call Control is based on SIP

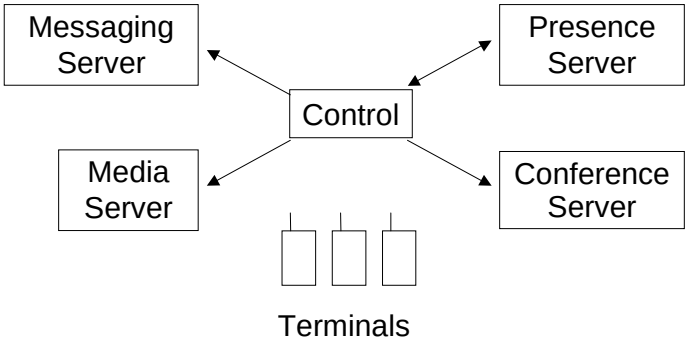


- Details are still to be solved in the IETF
- Powerful tool e.g. for inviting users to centralized conferences or sessions with a Media Server

REFER and Call Transfer



Auto-conferencing Service Example



1. One user orders the conference by filling a web form
2. Controller subscribes to each participants presence
3. When all available, send message or start IVR session to each participant to confirm willingness
4. Connect each participant to conference server. Play announcements to conference from media server when new parties join

Problems

- How to make "service routing"?
- How to make service components really independent?
- If there is dependency, how to move parameters between the components?
- How to secure call release?