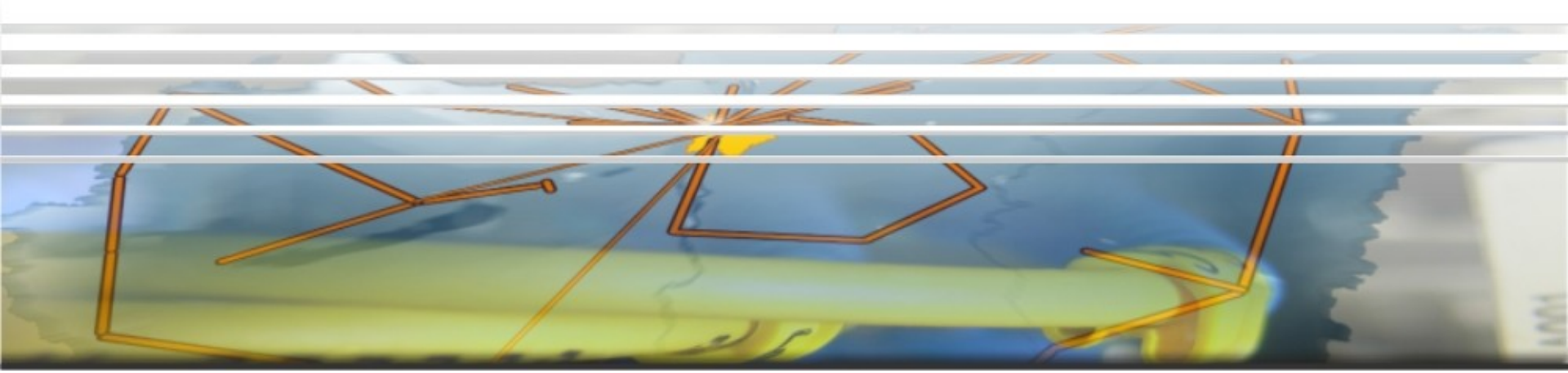
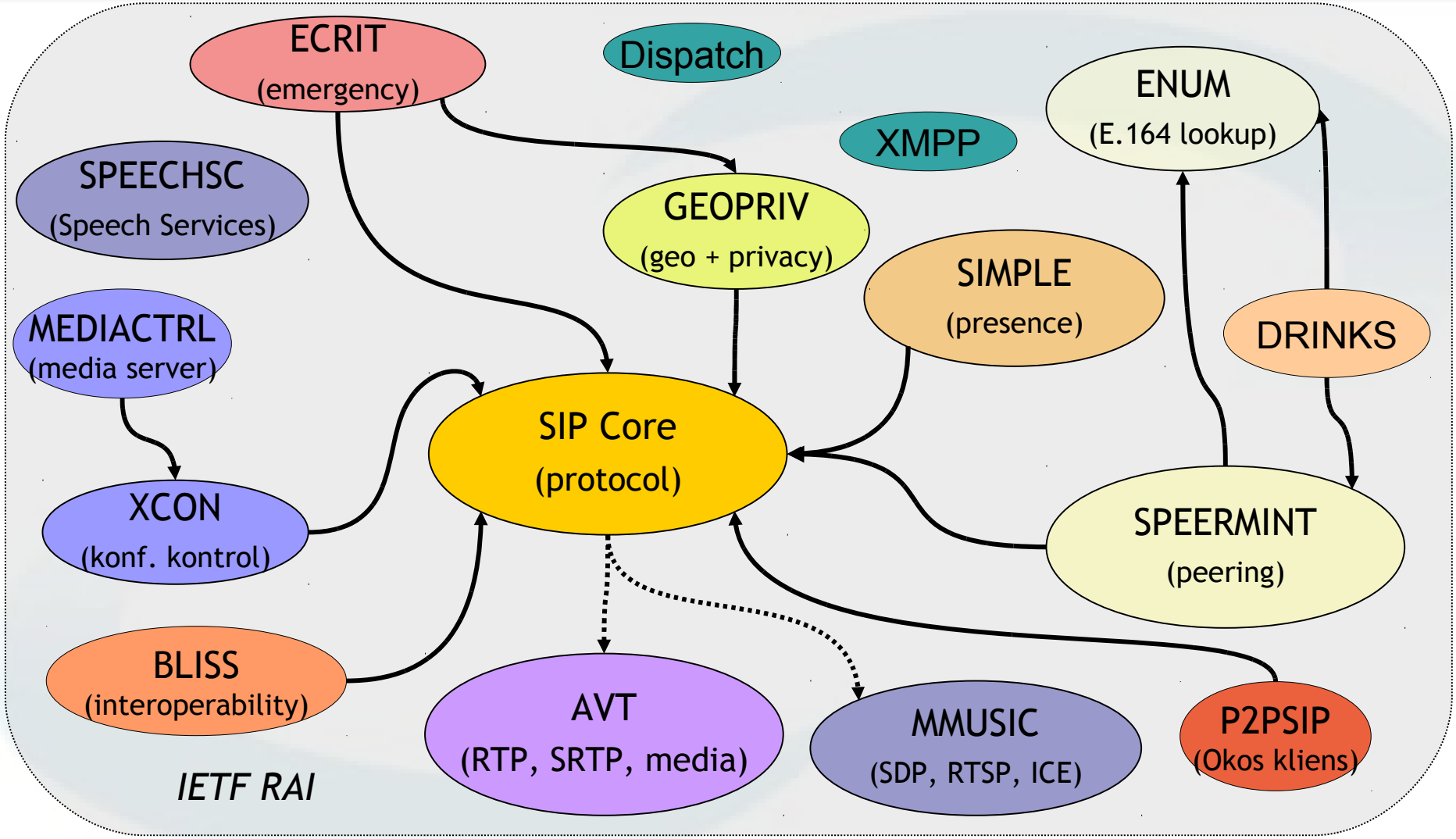
SIP Basics



03/16/10
NIIF Budapest

MÉSZÁROS Mihály

IETF Working Groups

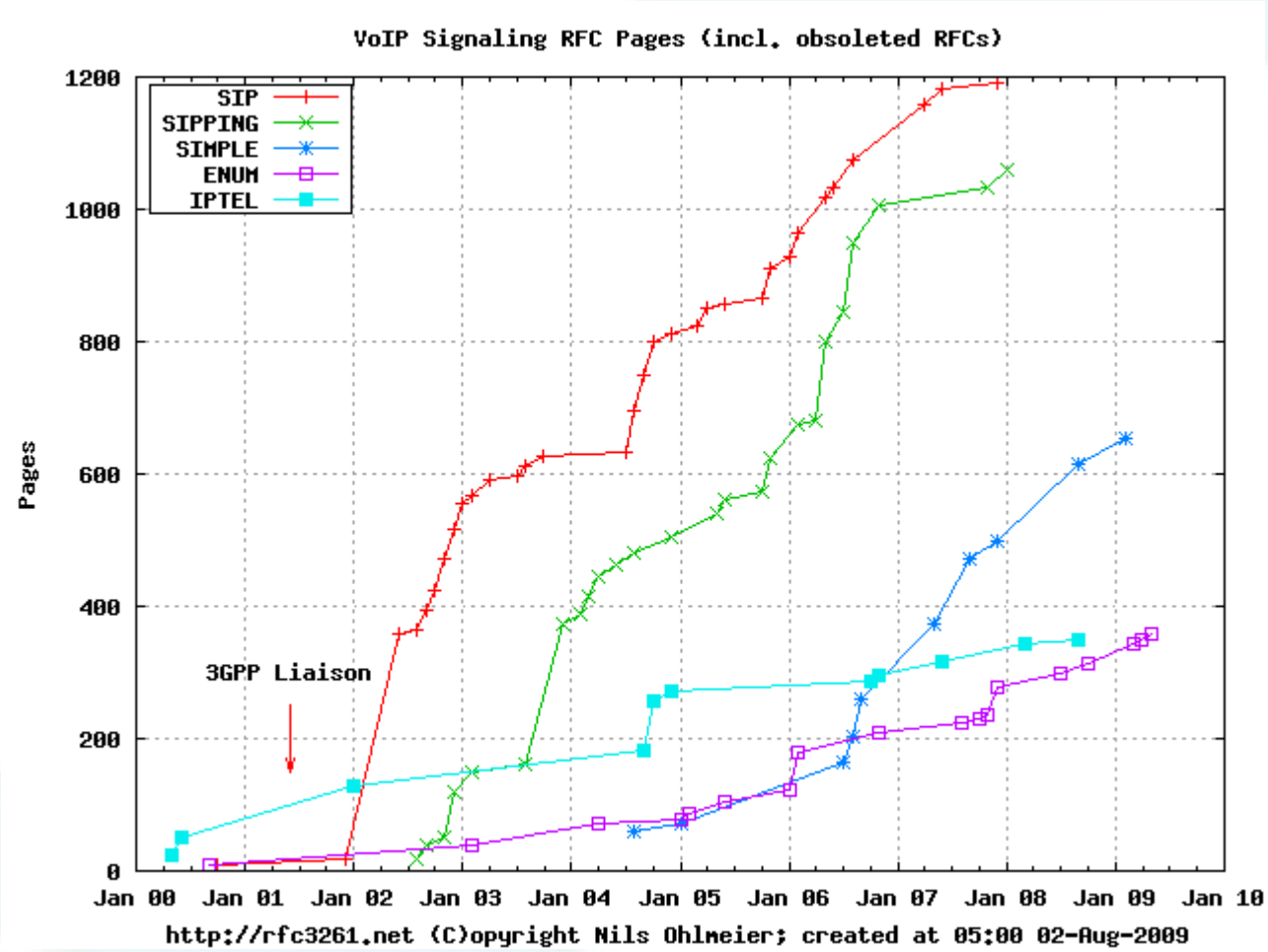


SIP Short History

- Simple Conference Invitation Protocol
  - Feb. 22, 1996 first drafts
- RFC 2543
  - March 17, 1999 First approved RFC
- RFC 3261
  - July 3, 2002 new RFC obsoleting 2543
- More details:
  - http://www.cs.columbia.edu/sip/history.html

Despite first draft name, currently the 'S' in SIP acronym is no more for Simple!

- http://www.rfc3261.net/



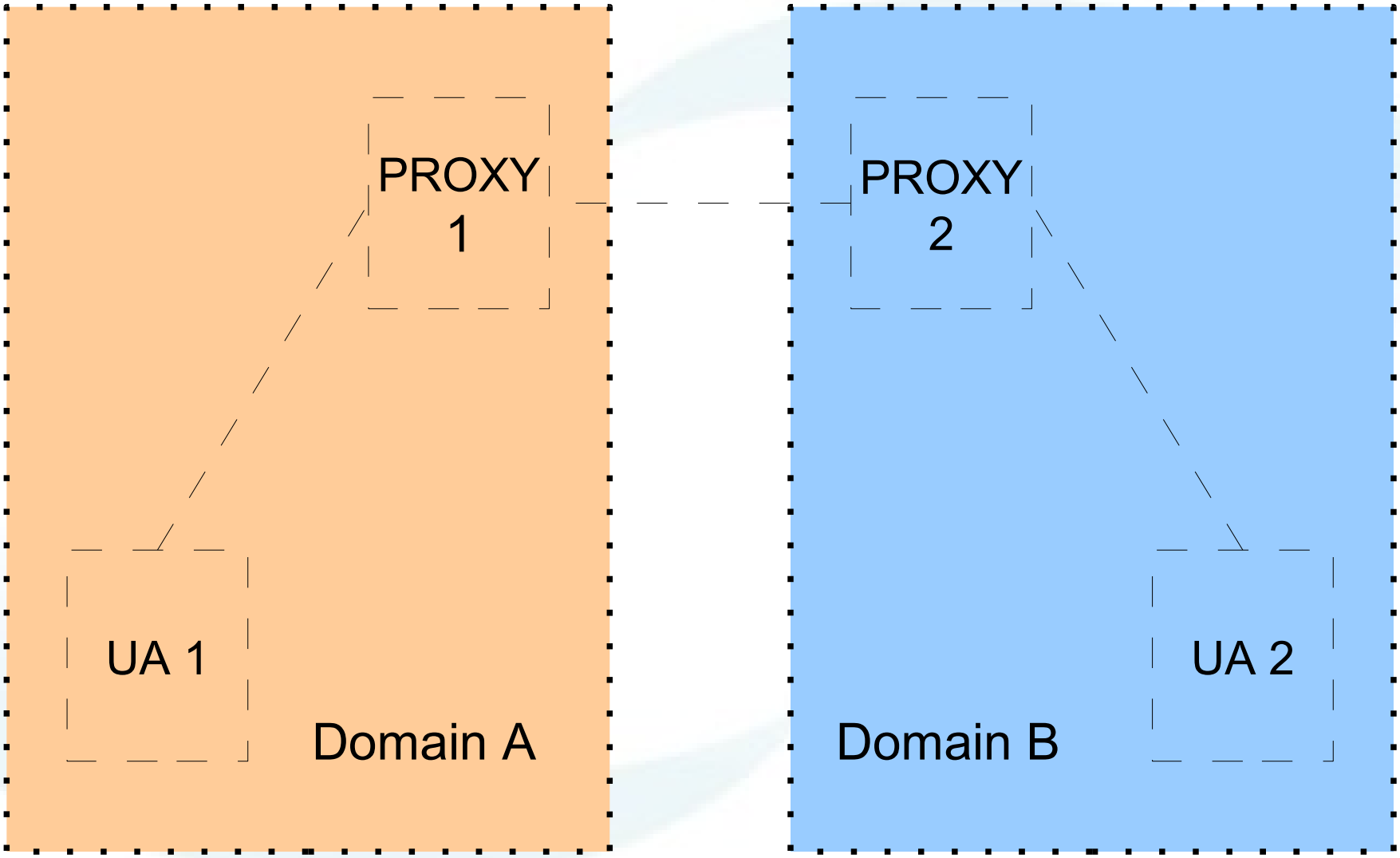
SIP Basics Concepts

- HTTP, SMTP like protocol over IPv4, IPv6 and IP Multicast
- SIP is a text-based protocol and uses the UTF-8 charset
- Distributed end2end design similar to Internet design
- Almost all intelligence in the endpoint
- Network use and know very basic information. Almost only transaction state and routing information.
- Outcome:
  - Flexibility
    - New service can introduced without changing the network
  - Scalability
    - Network knows about only basic state information => network are using less resource.
  - Recovery from failure
    - Basic small information can recovered more faster.
  - Relatively expensive endpoints

SIP functionality

- User Location
  - determination of the end system to be used for communication
  - Registrar
  - DNS (RFC3263)
- User availability
  - determination of the end system to be used for communication
  - Location service (some kind of basic presence)
  - OPTION method (ping)
- Session setup
  - INVITE
- Session management
  - ReINVITE
    - retraget
    - codec change (upgrade voice call to video + voice)
    - hold
    - change media ip address
  - Tear down (Bye, Cancel)

SIP Trapezoid



RFC3261 example

- http://www.tech-invite.com/Ti-sip-ex3261.html

Wireshark - Live sample capture, call flow

sipcall2.dump - Wireshark

File Edit View Go Capture Analyze Statistics Telephony Tools Help

Filter: Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Info
1	0.000000	195.111.159.65	195.111.96.36	SIP/SDP	Request: INVITE sip:4503099@deverto1.vh.hbone.hu, with session descri
2	0.002083	195.111.96.36	195.111.159.65	SIP	Status: 100 Trying
3	0.061169	195.111.96.36	195.111.159.65	SIP/SDP	Status: 200 OK, with session description
4	0.061336	195.111.159.65	195.111.96.36	SIP	Request: ACK sip:4503099@195.111.96.36:5060
5	6.451119	195.111.159.65	195.111.96.36	SIP	Request: BYE sip:4503099@195.111.96.36:5060
6	6.453449	195.111.96.36	195.111.159.65	SIP	Status: 200 OK

sipcall2.dump - VoIP Calls

Detected 1 VoIP Call. Selected 1 Call.

Start Time	Stop Time	Initial Speaker	From	To	Protocol	Packets	State	Comments
0.000	6.453	195.111.159	sip:4503084@195.1	sip:4503099@devert	SIP	6	COMPLET	

Total: Calls: 1 Start packets: 0 Completed calls: 1 Rejected calls: 0

Prepare Filter | Graph | Player | Select All | Bezárás

sipcall2.dump - Graph Analysis

Time	195.111.159.65 → 195.111.96.36	Comment
0,000	INVITE SDP (telephone-event)	SIP From: sip:4503084@195.111.159.65 To:si
0,002	100 Trying	SIP Status
0,061	200 OK SDP (telephone-event)	SIP Status
0,061	ACK	SIP Request
6,451	BYE	SIP Request
6,453	200 OK	SIP Status

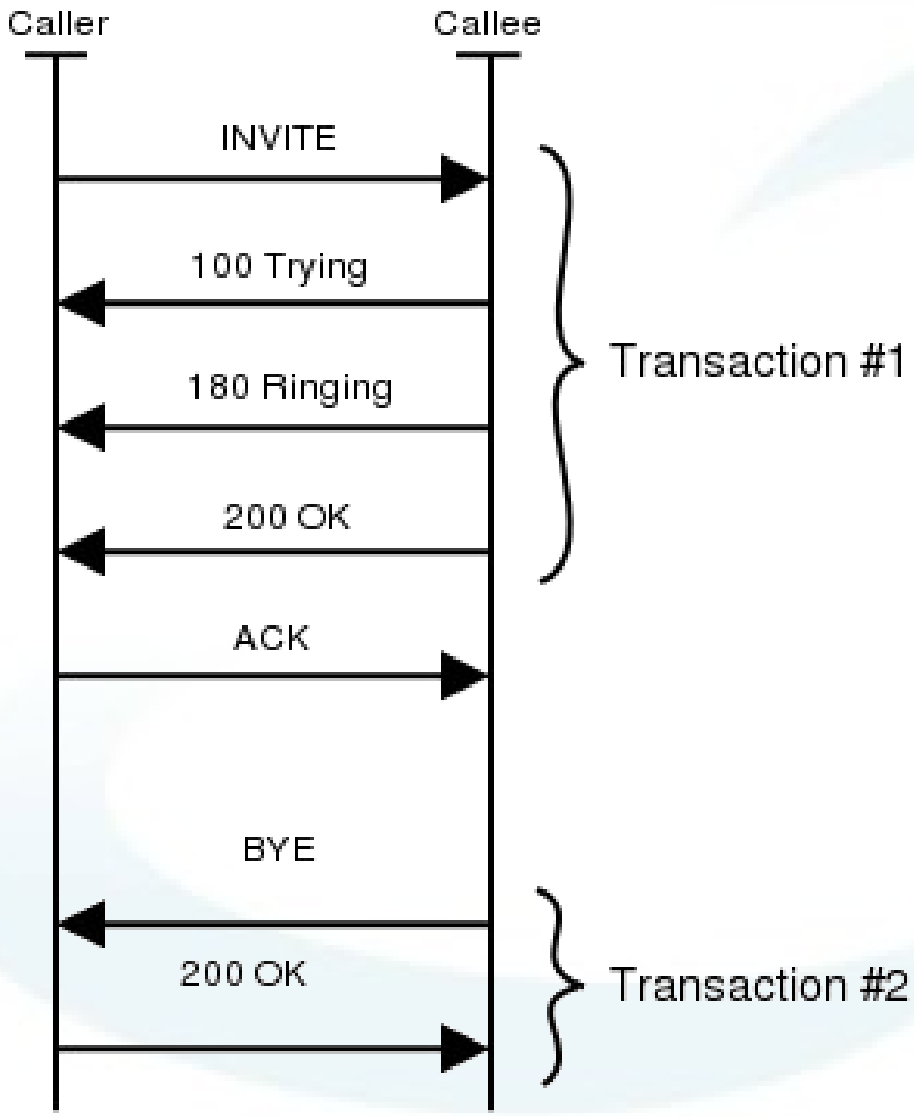
Mentés másként | Bezárás

```
From: "Misi" <sip:4503084@195.111.159.65>;tag=as46f36973\r\n
To: <sip:4503099@deverto1.vh.hbone.hu>\r\n
Contact: <sip:4503084@195.111.159.65>\r\n
Call-ID: 251c8169294d51bd1c46f2e637bba8dc@195.111.159.65\r\n
CSeq: 102 INVITE\r\n
User-Agent: NIIF Asterisk PBX\r\n
Max-Forwards: 70\r\n
Remote-Party-ID: "Misi" <sip:4503084@195.111.159.65>;privacy=off;screen=no\r\n
Date: Mon, 22 Feb 2010 11:49:32 GMT\r\n
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY\r\n
```

```
0000  00 11 5d 5b b8 c0 00 09  6b 52 b3 d8 08 00 45 00   ..][.... kR....E.
0010  03 a2 6f 72 00 00 40 11  81 94 c3 6f 9f 41 c3 6f   ..or..@. ...o.A.o
0020  60 24 13 c4 13 c4 03 8e  1b 22 49 4e 56 49 54 45   `$...... ."INVITE
0030  20 73 69 70 3a 34 35 30  33 30 39 39 40 64 65 76    sip:450 3099@dev
0040  65 72 74 6f 31 2e 76 68  2e 68 62 6f 6e 65 2e 68   erto1.vh .hbone.h
0050  75 20 53 49 50 2f 32 2e  30 0d 0a 56 69 61 3a 20   u SIP/2. 0..Via:
```

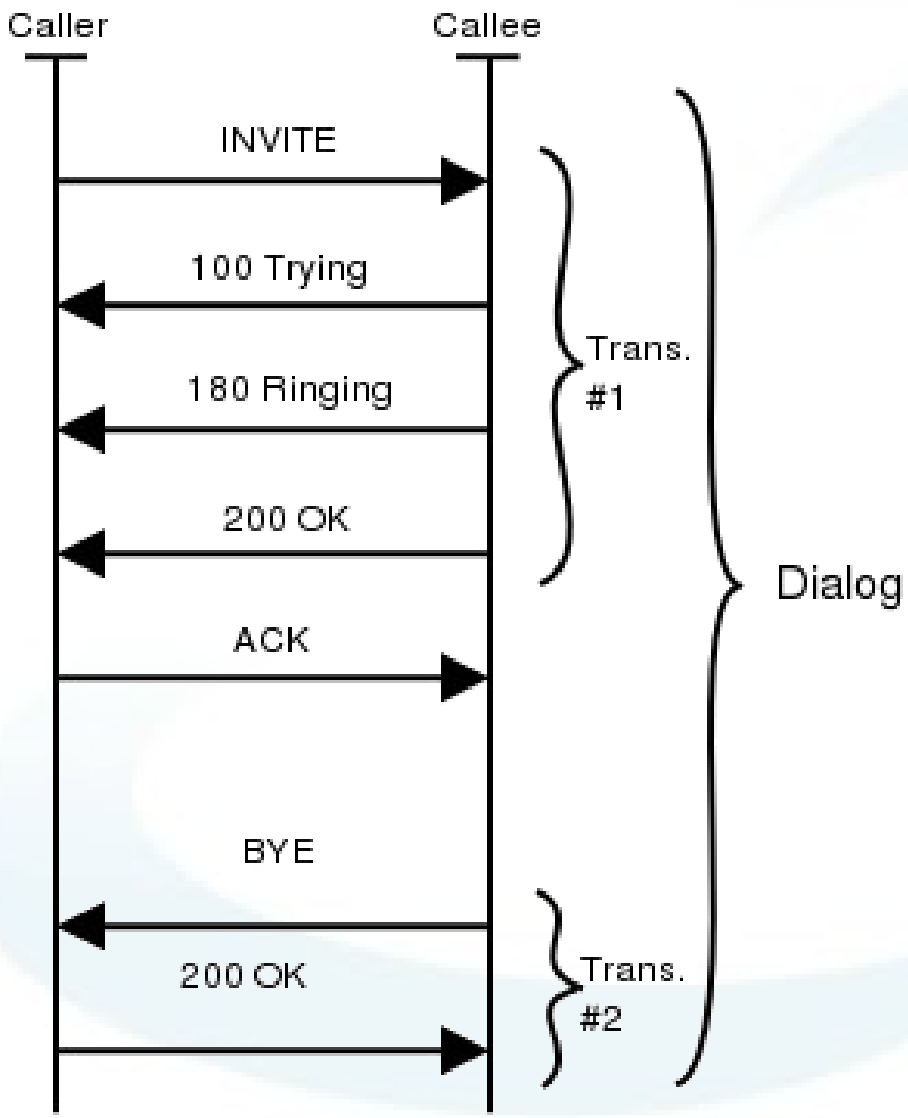
File: "/home/misi/work/sipcall2.dump" 3686 Bytes 00:00:06 | Packets: 6 Displayed: 6 Marked: 0 | Profile: Default

Transaction



- request
- zero or more provisional (1XX) response
- one final
  - 2XX
  - 3XX
  - 4XX
  - 5XX
  - 6XX
- If method is INVITE then ACK
  - part of transaction if status code is "negative" final (>=3XX)
    - generated by transaction
  - not part of transaction if status code is "positive" final (2XX)
    - generated by UA
- Transaction ID
  - Via header, branch ID parameter

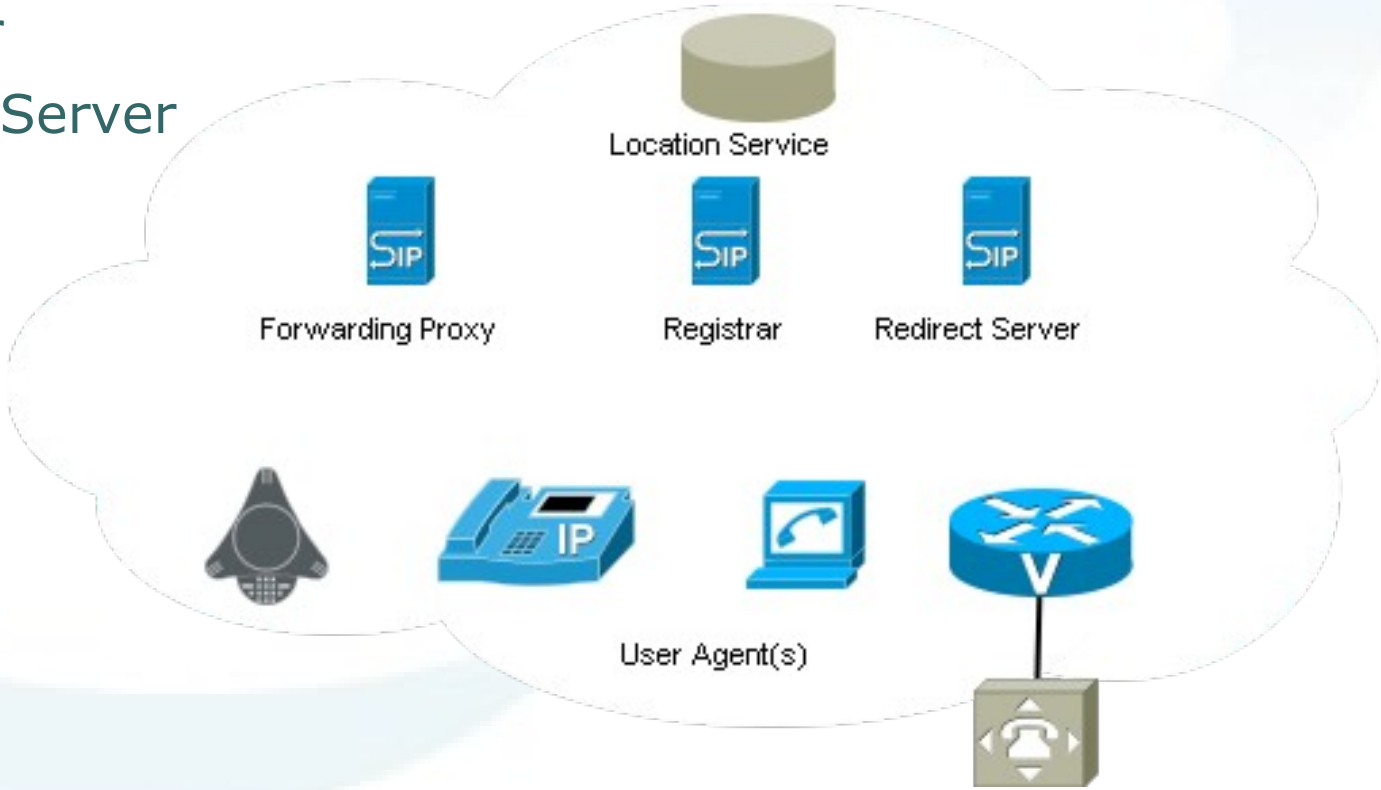
Dialog (sequence of transactions)



- Dialog initiator methods
  - INVITE,SUBSCRIBE,REFER
- peer-to peer relationship between two UA
- sequencing (CSeq)
- dialog state
  - dialog ID
    - Call-ID
    - local tag (From)
    - remote tag (To)
  - local sequence
  - remote sequence
  - local URI
  - remote URI
  - remote target(contact)
  - secure flag (Boolean)
  - Route-set
    - ordered URI list

Terminology (RFC3261)

- User Agent
- Servers
  - Forwarding Proxy
  - Registrar
  - Redirect Server



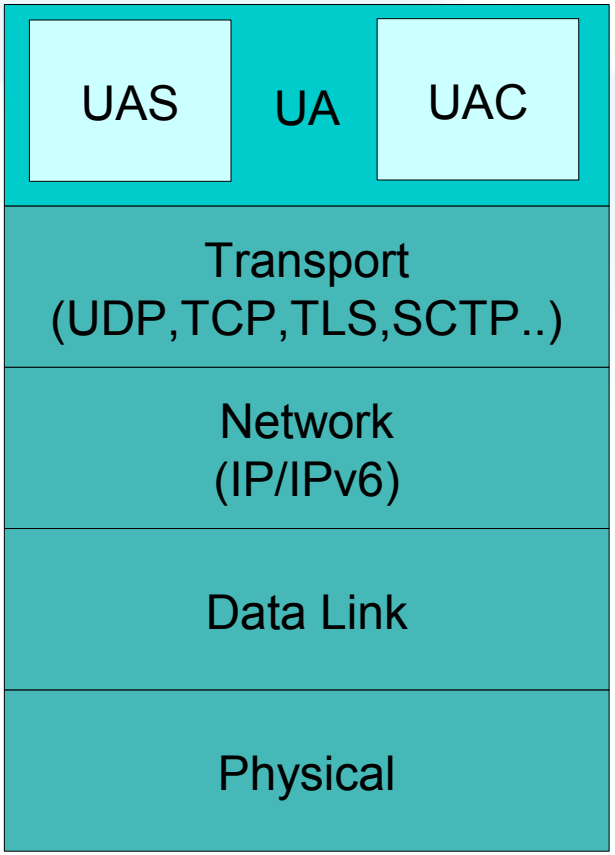
UA classification

- Equipment
  - Soft Phone
  - Hard Phone
- Media capability
  - VoIP
    - Audio only
  - Videoconference, Telepresence
    - audio and video
- Supported features
  - Presence
    - body list
  - Presentation,
    - second video stream
    - screen sharing
  - FECC
  - Text messaging, chat
  - File Transfer

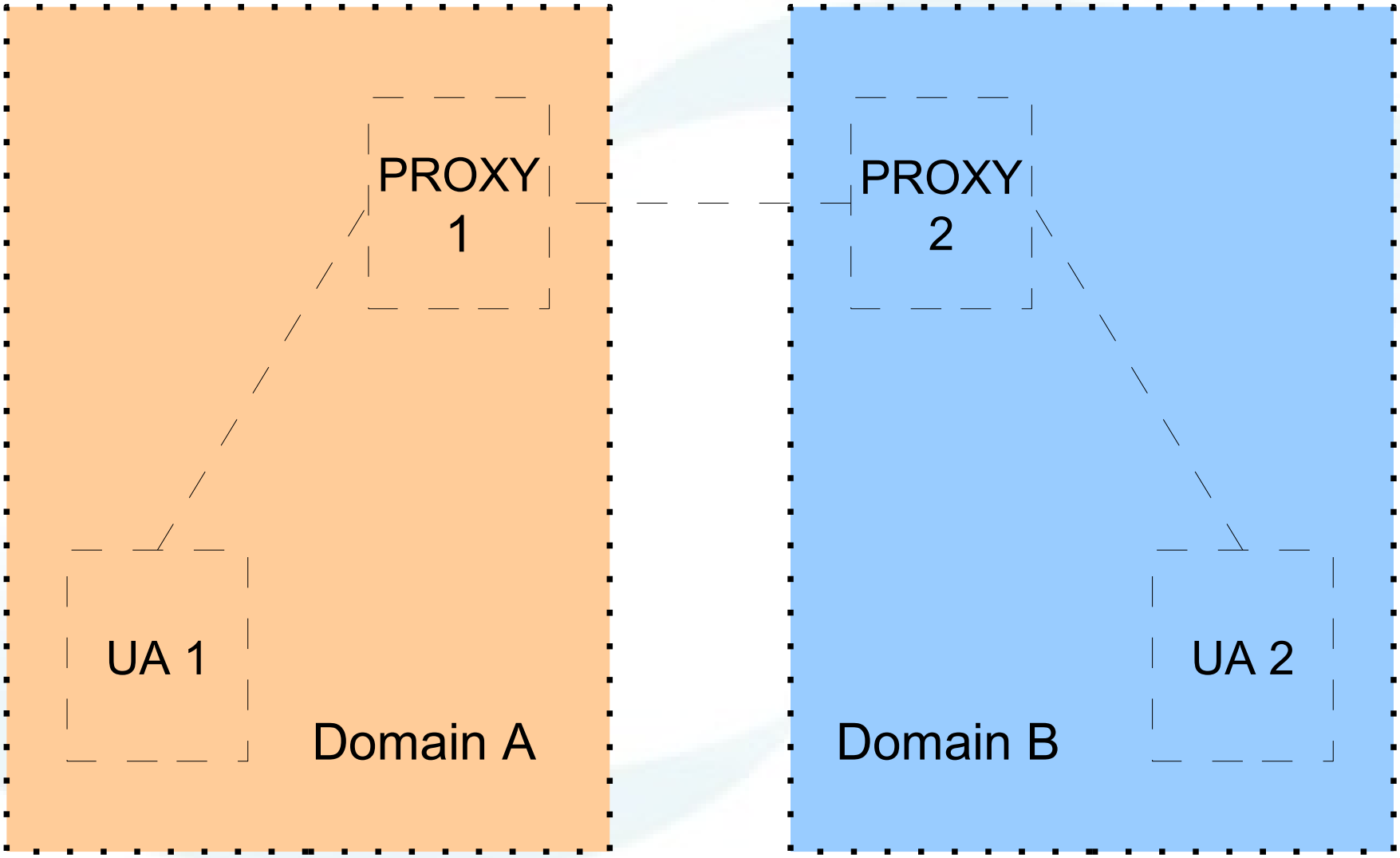

source: http://www.denphone.com/files/cisco-telepresence-1000.jpg

User Agent

- **User Agent (UA)**
  - **UA Client (UAC):** is a logical entity that creates a new request, and then uses the client transaction state machinery to send it.
  - **UA Server (UAS):** is a logical entity that generates a response to a SIP request.



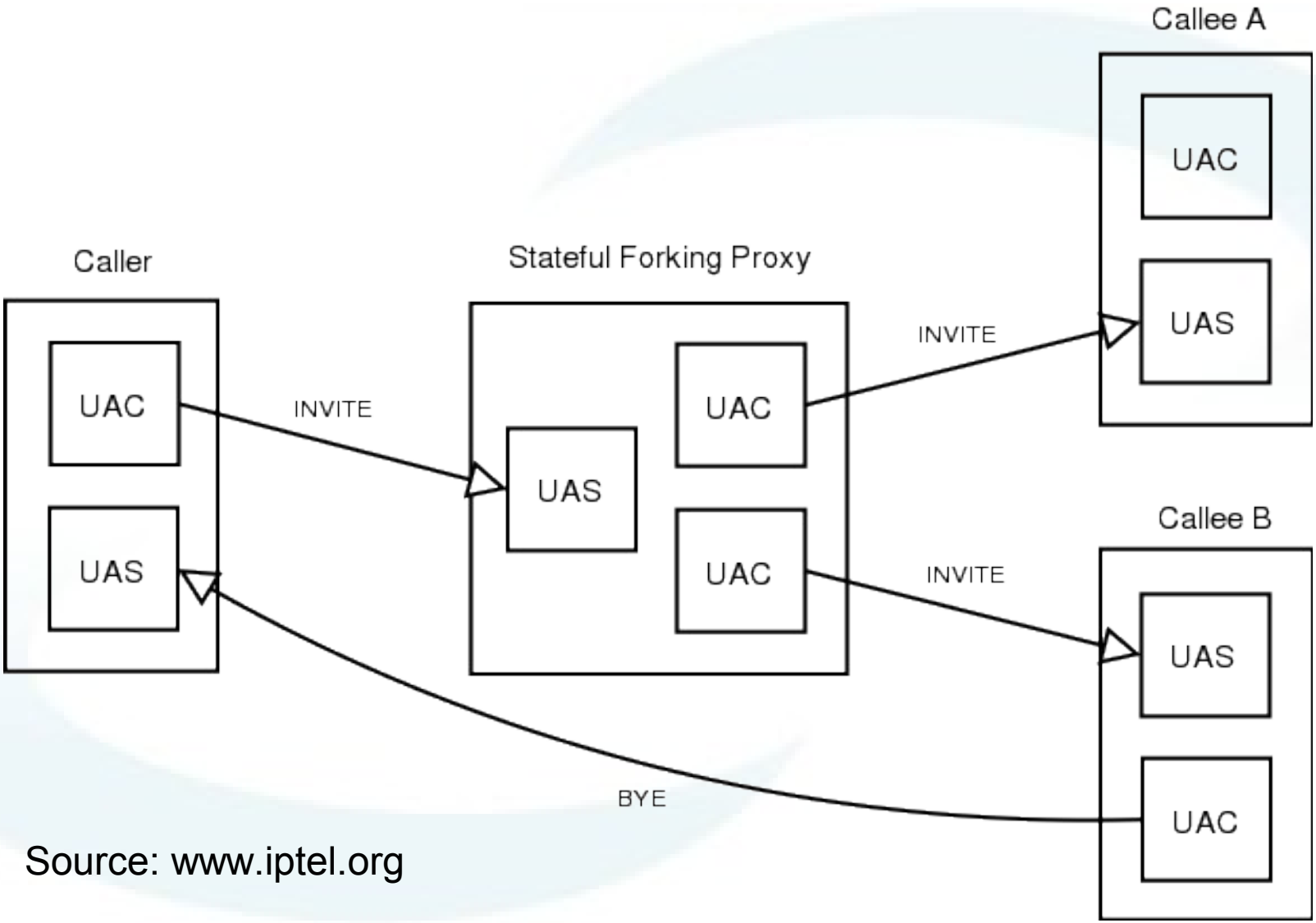
SIP Trapezoid



Transaction relationships

```
+---------+        +---------+        +---------+        +---------+
|      +-+|Request |+-+   +-+|Request |+-+   +-+|Request |+-+      |
|      |C||------->||S|   |C||------->||S|   |C||------->||S|      |
|      |l||        ||e|   |l||        ||e|   |l||        ||e|      |
|      |i||        ||r|   |i||        ||r|   |i||        ||r|      |
|      |e||        ||v|   |e||        ||v|   |e||        ||v|      |
|      |n||        ||e|   |n||        ||e|   |n||        ||e|      |
|      |t||        ||r|   |t||        ||r|   |t||        ||r|      |
|      | ||        || |   | ||        || |   | ||        || |      |
|      |T||        ||T|   |T||        ||T|   |T||        ||T|      |
|      |r||        ||r|   |r||        ||r|   |r||        ||r|      |
|      |a||        ||a|   |a||        ||a|   |a||        ||a|      |
|      |n||        ||n|   |n||        ||n|   |n||        ||n|      |
|      |s||Response||s|   |s||Response||s|   |s||Response||s|      |
|      +-+|<-------|+-+   +-+|<-------|+-+   +-+|<-------|+-+      |
+---------+        +---------+        +---------+        +---------+
     UAC               Outbound           Inbound               UAS
                          Proxy              Proxy
```

User Agent Graph (Fork example)



Source: www.iptel.org

Transaction – Dialog - Session

- **SIP Transaction:** occurs between a client and a server and comprises all messages from the first request sent from the client to the server up to a final (non-1xx) response sent from the server to the client. If the request is INVITE and the final response is a non-2xx, the transaction also includes an ACK to the response. The ACK for a 2xx response to an INVITE request is a separate transaction.
- **Dialog:** is a peer-to-peer SIP relationship between two UAs that persists for some time. A dialog is established by SIP messages, such as a 2xx response to an INVITE request. A dialog is identified by a call identifier, local tag, and a remote tag.
- **Session:**A multimedia session is a set of multimedia senders and receivers and the data streams flowing from senders to receivers.

Structure of the Protocol

- TCP/IP
  - Application Layer
- OSI
  - Session Layer

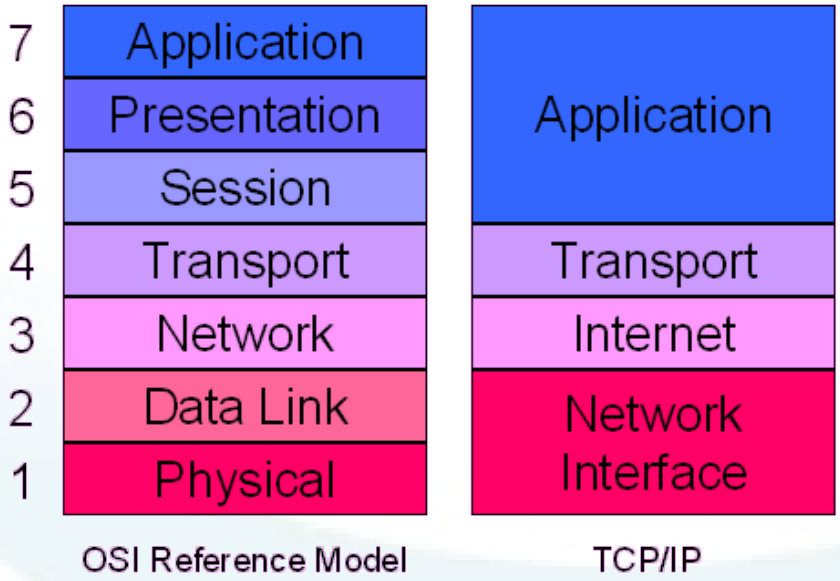


image source:http://www.hardwaresecrets.com/imageview.php?image=6731

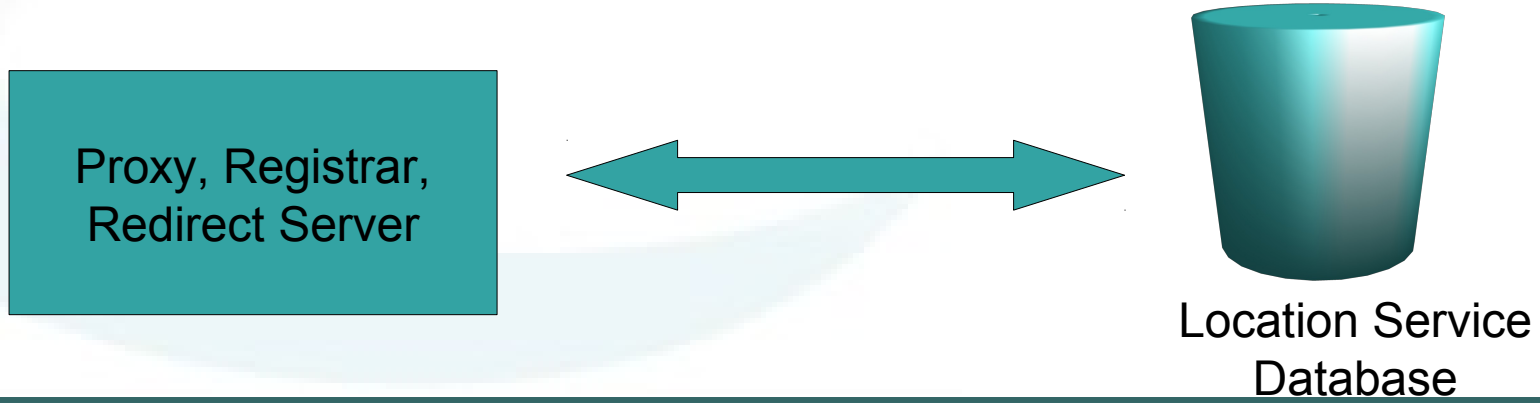
- SIP is an application layer protocol in TCP/IP reference model
- SIP Structure:
  - Transaction User (TU)
    - **Proxy core**
    - **UA Core**
  - Transaction
  - Transport
  - SIP syntax and encoding
- Good help to understand SIP Structure on tech-invite web site.
  - SIP Protocol Structure through an Example
    http://www.tech-invite.com/Ti-sip-archi.html

Terminology - Servers

- **Proxy, Proxy Server:** An intermediary entity that acts as both a server and a client for the purpose of making requests on behalf of other clients. A proxy server primarily plays the role of routing, which means its job is to ensure that a request is sent to another entity "closer" to the targeted user. Proxies are also useful for enforcing policy (for example, making sure a user is allowed to make a call). A proxy interprets, and, if necessary, rewrites specific parts of a request message before forwarding it.
- **Redirect Server:** A redirect server is a user agent server that generates 3xx responses to requests it receives, directing the client to contact an alternate set of URIs.
- **Registrar:** is a server that accepts REGISTER requests and places the information it receives in those requests into the location service for the domain it handles.

Terminology - Location Service

- **Location Service**:A location service is used by a SIP redirect or proxy server to obtain information about a callee's possible location(s). It contains a list of bindings of address-of-record keys to zero or more contact addresses.
  - Registrar is storing bindings in this DB service.
  - Forwarding Proxy is utilizing the DB to make routing decisions.
  - Redirect Server using this DB to redirect UA.



Terminology - Proxy

- **Stateless:**

  is a logical entity that does not maintain the client or server transaction state machines defined in this specification when it processes requests. A stateless proxy forwards every request it receives downstream and every response it receives upstream.

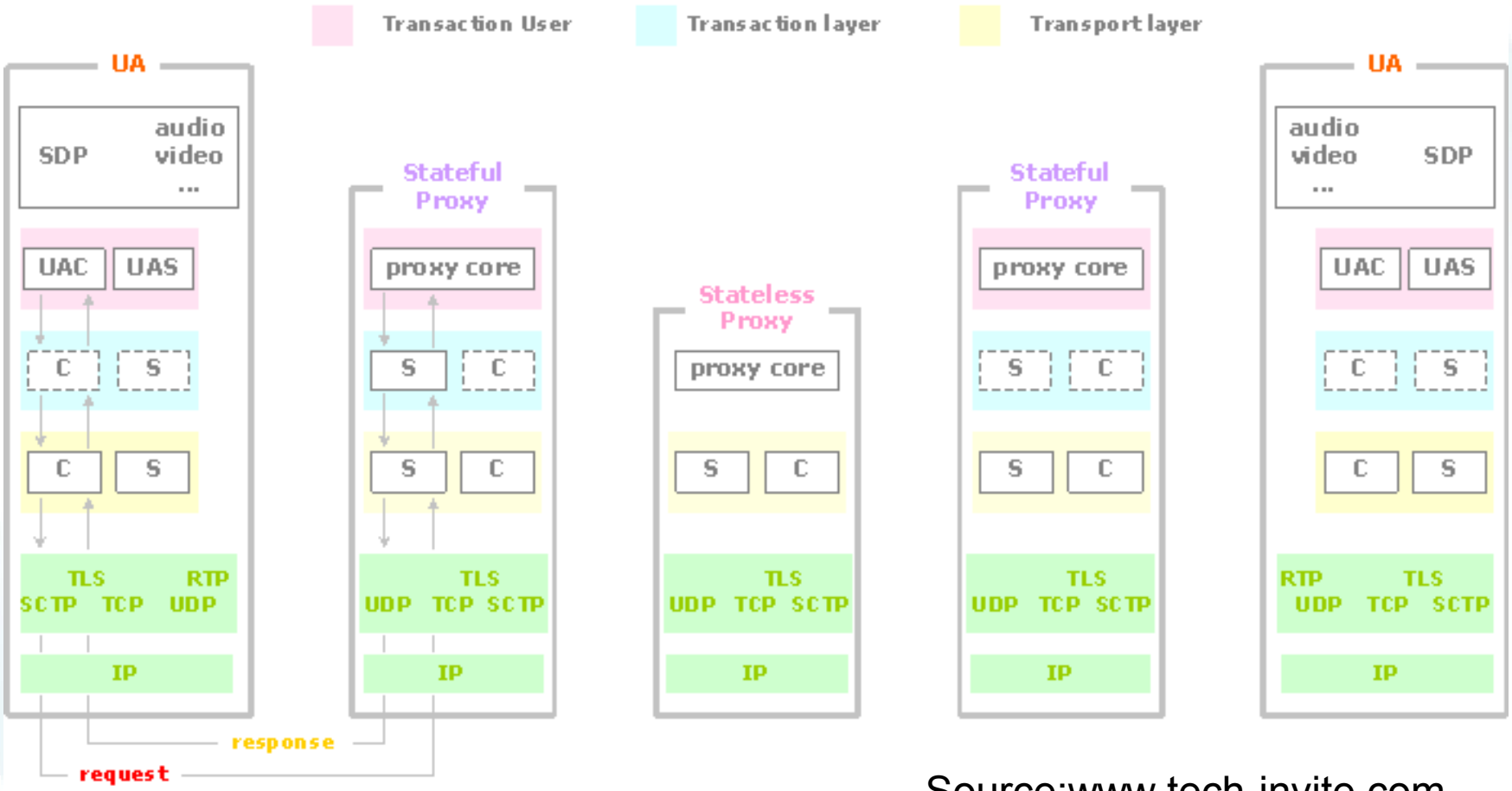
- **Transaction Stateful, Stateful Proxy:**

  is a logical entity that maintains the client and server transaction state machines defined by this specification during the processing of a request, also known as a transaction stateful proxy.

- **Call Stateful(!):**

  A proxy is call stateful if it retains state for a dialog from the initiating INVITE to the terminating BYE request. A call stateful proxy is always transaction stateful, but the converse is not necessarily true.

SIP Architecture



Source:www.tech-invite.com

Identifying endpoints

- URI:
  - Uniform Resource Identifier (URI) consists of a string of characters used to identify or name a resource on the Internet.
- UA URI schemes
  - sip/sips:
    - sip:alice@atlanta.test / sips:alice@atlanta.test
  - Tel (E.164):
    - tel:+3614503060
    - The tel: uri scheme has no “user” or “domain” part!
- ISN/ITAD
  - 1234*256

Terminology - AoR

- **Address-of-Record:** An address-of-record (AOR) is a SIP or SIPS URI that points to a domain with a location service that can map the URI to another URI where the user might be available. Typically, the location service is populated through registrations. An AOR is frequently thought of as the "public address" of the user.
  - Weel known uri
    - sip:misi@niif.hu

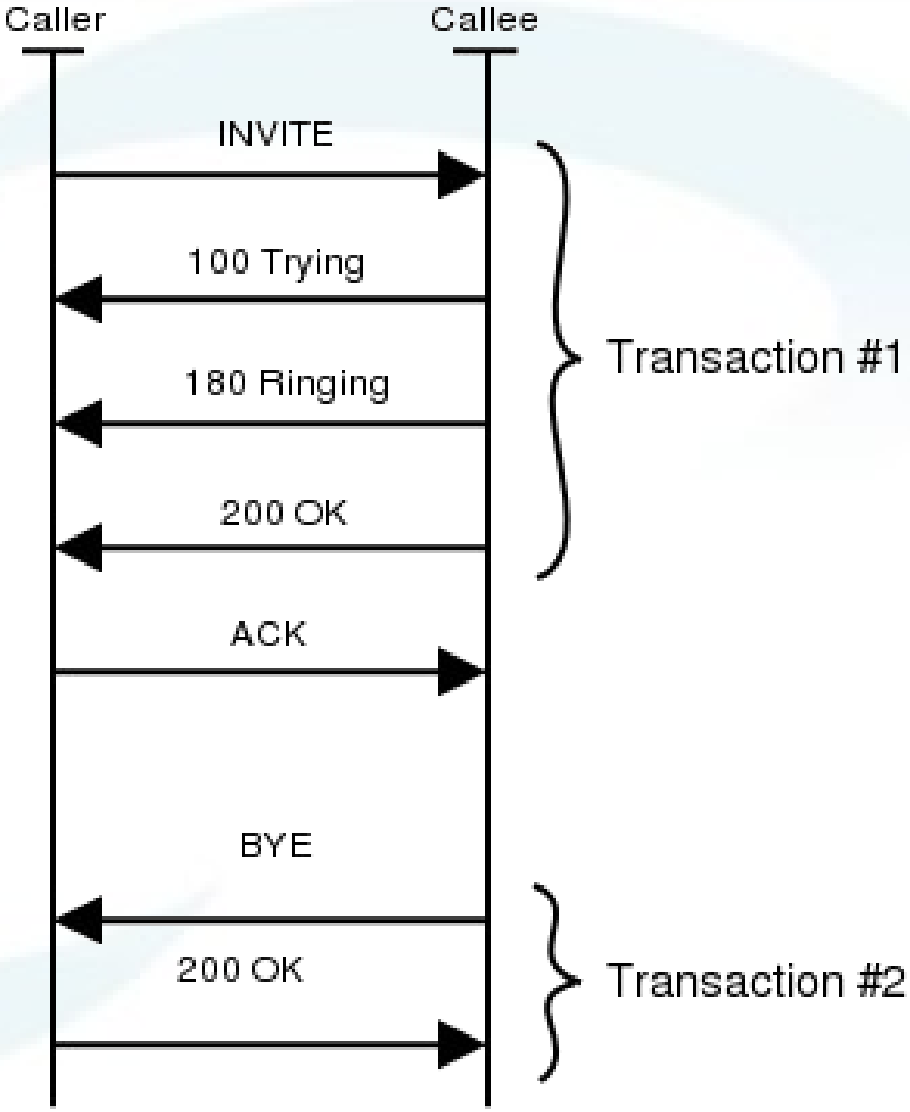
Message

```
generic-message = start-line
                  *message-header
                  CRLF
                  [ message-body ]

Start-line = Request-Line / Status-Line
```

SIP Methods

Methods	Reference
ACK	[RFC3261]
BYE	[RFC3261]
CANCEL	[RFC3261]
INFO	[RFC2976]
INVITE	[RFC3261]
MESSAGE	[RFC3428]
NOTIFY	[RFC3265]
OPTIONS	[RFC3261]
PRACK	[RFC3262]
PUBLISH	[RFC3903]
REFER	[RFC3515]
REGISTER	[RFC3261]
SUBSCRIBE	[RFC3265]
UPDATE	[RFC3311]



SIP request message

```
INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/TCP client.atlanta.example.com:5060
;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>
;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.atlanta.example.com
;transport=tcp>
Content-Type: application/sdp
Content-Length: 151

v=0
o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

SIP Response message

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP client.atlanta.example.com:5060
;branch=z9hG4bK74bf9
;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>
;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
;tag=8321234356
Call-ID: 384827629822018851 1@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.biloxi.example.com;transport=tcp>
Content-Type: application/sdp
Content-Length: 147

v=0
o=bob 2890844527 2890844527 IN IP4
client.biloxi.example.com
s=-
c=IN IP4 192.0.2.201
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Minimal request - Request vs. To

- Mandatory fields
  - start line
    - request/response
  - headers
    - Via
    - Max-Forwards
    - From
    - To
    - Call-ID
    - CSeq

```
OPTION sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/TCP client.atlanta.test:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
```

- Difference between Request-URI and To header!
  - To header
    - identifies the original recipient, but maybe URI differ from the UAS processing the request, due to call forwarding, or any other proxy operation.
    - Proxy can't rewrite
  - Request-URI
    - identifies the UAS that is processing the request
    - proxy can rewrite to a new URI what is different from original recipient

Framing SIP messages / Transport

- Unreliable datagram protocol (UDP)
  - One datagram is containing one message.
  - If a request is within 200 bytes of the path MTU, or if it is larger than 1300 bytes and the path MTU is unknown, the request MUST be sent using an RFC 2914 congestion controlled transport protocol, such as TCP.
  - If an element sends a request over TCP because of these message size constraints, and that request would have otherwise been sent over UDP, if the attempt to establish the connection generates either an ICMP Protocol Not Supported, or results in a TCP reset, the element SHOULD retry the request, using UDP.
  - If the transport user asks for a message to be sent over an unreliable transport, and the result is an ICMP error, the behavior depends on the type of ICMP error. Host, network, port or protocol unreachable errors, or parameter problem errors SHOULD cause the transport layer to inform the transport user of a failure in sending.

Framing SIP messages / Transport

- All SIP elements MUST implement UDP and TCP!
- Reliable stream protocol (TCP)
  - In the case of stream-oriented transports such as TCP, the Content-Length header field indicates the size of the body. The Content-Length header field MUST be used with stream oriented transports.
  - If the transport user asks for a request to be sent over a reliable transport, and the result is a connection failure, the transport layer SHOULD inform the transport user of a failure in sending.

Cseq

- Cseq
  - A CSeq header field in a request contains a single decimal sequence number and the request method.
  - The sequence number MUST be expressible as a 32-bit unsigned integer. The method part of CSeq is case-sensitive.
  - The CSeq header field serves to order transactions within a dialog, to provide a means to uniquely identify transactions, and to differentiate between new requests and request retransmissions. Two CSeq header fields are considered equal if the sequence number and the request method are identical.

Dialog

- Dialog ID
  - A dialog is identified at each UA with a **dialog ID**, which consists of a **Call-ID** value, a **local tag** and a **remote tag**.
- Dialog State
  - A dialog contains certain pieces of state needed for further message transmissions within the dialog. This state consists of the **dialog ID**, a **local sequence number** (used to order requests from the UA to its peer), a **remote sequence number** (used to order requests from its peer to the UA), a **local URI**, a **remote URI**, **remote target**, a boolean flag called **"secure"**, and a **route set**, which is an ordered list of URIs. The route set is the list of servers that need to be traversed to send a request to the peer.

Dialog example

- http://www.tech-invite.com/Ti-sip-dialog.html

Record-Route

- Route Set definition:
  - A route set is a collection of ordered SIP or SIPS URI which represent a list of proxies that must be traversed when sending a particular request. A route set can be learned, through headers like Record-Route, or it can be configured.
- Dialog
  - The route set MUST be set to the list of URIs in the Record-Route header field from the request, taken in order and preserving all URI parameters.
  - If no Record-Route header field is present in the request, the route set MUST be set to the empty set.
  - This route set, even if empty, overrides any pre-existing route set for future requests in this dialog.

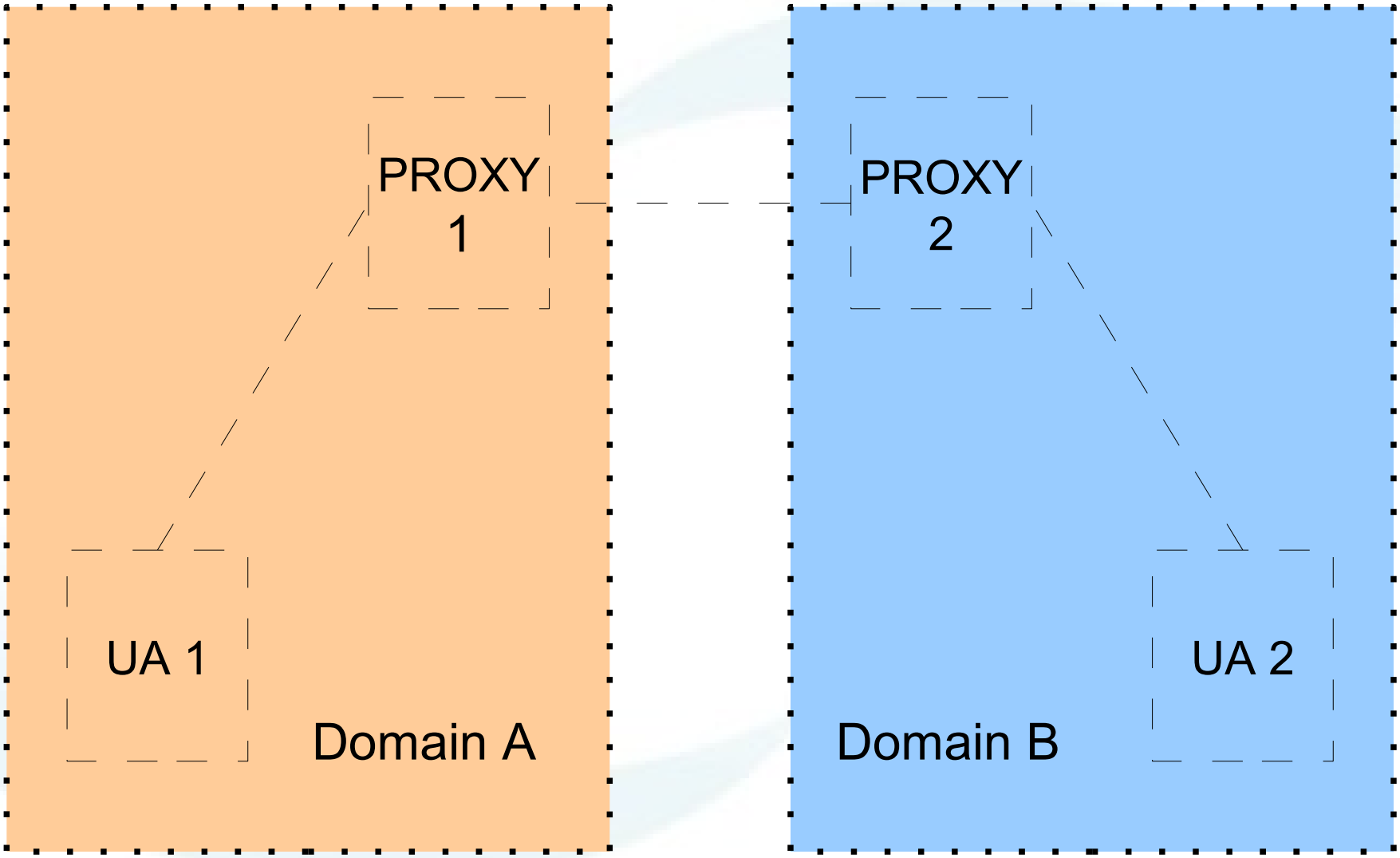
SIP Methods

- RFC3261
  - INVITE
  - ACK
  - OPTIONS
  - BYE
  - CANCEL
  - REGISTER
- RFC2976
  - INFO
- RFC3262
  - PRACK
- RFC3265
  - SUBSCRIBE
  - NOTIFY
- RFC3311
  - UPDATE
- RFC3428
  - MESSAGE
- RFC3515
  - REFER
- RFC3903
  - PUBLISH

SIP Response (Status Codes)

- 1XX Provisional
  - 100 Trying
  - 180 Ringing
  - 183 Session Progress
- 2XX Successful
  - 200 OK
- 3XX Redirection
  - 300 Multiple Choices
  - 301 Moved Permanently
  - 302 Moved Temporarily
- 4XX Request Failure
  - 400 Bad Request
  - 401 Unauthorized
  - 403 Forbidden
  - 404 Not Found
  - etc.
- 5XX Server Failure
  - 500 Server Internal Error
  - 501 Not Implemented
  - etc.
- 6XX Global Failure
  - 600 Busy Everywhere
  - etc.

SIP Trapezoid



SIP Trapezoid II.

```
                     atlanta.com  . . . biloxi.com
                 .      proxy              proxy     .
               .                                       .
       Alice's  . . . . . . . . . . . . . . . . . . . .  Bob's
      softphone                                        SIP Phone
         |                |                |                |
         |    INVITE F1   |                |                |
         |--------------->|    INVITE F2   |                |
         |  100 Trying F3 |--------------->|    INVITE F4   |
         |<---------------|  100 Trying F5 |--------------->|
         |                |<-------------- | 180 Ringing F6 |
         |                | 180 Ringing F7 |<---------------|
         | 180 Ringing F8 |<---------------|     200 OK F9  |
         |<---------------|    200 OK F10  |<---------------|
         |    200 OK F11  |<---------------|                |
         |<---------------|                |                |
         |                       ACK F12                    |
         |------------------------------------------------->|
         |                   Media Session                  |
         |<================================================>|
         |                       BYE F13                    |
         |<-------------------------------------------------|
         |                     200 OK F14                   |
         |------------------------------------------------->|
         |                                                  |
```

Via, Routing

- Via
  - The Via header field indicates the path taken by the request so far and indicates the path that should be followed in routing responses. The branch ID parameter in the Via header field values serves as a transaction identifier, and is used by proxies to detect loops.
- Routing
  - Strict Routing (deprecated)
    - Request URI always contain the next hop URI. (the next hop proxy or a UA) Every hop is overwriting this with next hop.
  - Loose Routing (strongly preferred)
    - Request URI always contain the destination User Agent URI

Traversing a Strict-Routing Proxy (RFC3261)

In this scenario, a dialog is established across four proxies, each of which adds Record-Route header field values.

The third proxy implements the strict-routing procedures specified in RFC 2543 and many works in progress.

```
U1->P1->P2->P3->P4->U2
```

The INVITE arriving at U2 contains:

```
INVITE sip:callee@u2.domain.com SIP/2.0
Contact: sip:caller@u1.example.com
Record-Route: <sip:p4.domain.com;lr>
Record-Route: <sip:p3.middle.com>
Record-Route: <sip:p2.example.com;lr>
Record-Route: <sip:p1.example.com;lr>
```

Traversing a Strict-Routing Proxy II.

Which U2 responds to with a 200 OK. Later, U2 sends the following BYE request to P4 based on the first Route header field value.

```
BYE sip:caller@u1.example.com SIP/2.0
Route: <sip:p4.domain.com;lr>
Route: <sip:p3.middle.com>
Route: <sip:p2.example.com;lr>
Route: <sip:p1.example.com;lr>
```

P4 is not responsible for the resource indicated in the Request-URI so it will leave it alone. It notices that it is the element in the first Route header field value so it removes it. It then prepares to send the request based on the now first Route header field value of sip:p3.middle.com, but it notices that this URI does not contain the lr parameter, so before sending, it reformats the request to be:

```
BYE sip:p3.middle.com SIP/2.0
Route: <sip:p2.example.com;lr>
Route: <sip:p1.example.com;lr>
Route: <sip:caller@u1.example.com>
```

Traversing a Strict-Routing Proxy III.

P3 is a strict router, so it forwards the following to P2:

```
BYE sip:p2.example.com;lr SIP/2.0
Route: <sip:p1.example.com;lr>
Route: <sip:caller@u1.example.com>
```

P2 sees the request-URI is a value it placed into a Record-Route header field, so before further processing, it rewrites the request to be:

```
BYE sip:caller@u1.example.com SIP/2.0
Route: <sip:p1.example.com;lr>
```

P2 is not responsible for u1.example.com, so it sends the request to P1 based on the resolution of the Route header field value.
P1 notices itself in the topmost Route header field value, so it removes it, resulting in:

```
BYE sip:caller@u1.example.com SIP/2.0
```

Since P1 is not responsible for u1.example.com and there is no Route header field, P1 will forward the request to u1.example.com based on the Request-URI.

Fork

- Parallel search (fork)
  - http://www.tech-invite.com/Ti-sip-service-12.html
- Sequential search
  - http://www.tech-invite.com/Ti-sip-service-13.html

Options

- The SIP method OPTIONS allows a UA to query another UA or a proxy server as to its capabilities. This allows a client to discover information about the supported methods, content types, extensions, codecs, etc. without "ringing" the other party.
- That is, a 200 (OK) would be returned if the UAS is ready to accept a call, a 486 (Busy Here) would be returned if the UAS is busy, etc. This allows an OPTIONS request to be used to determine the basic state of a UAS, which can be an indication of whether the UAS will accept an INVITE request.
- A forked OPTIONS will only result in a single 200 (OK) response.

```
OPTIONS sip:carol@chicago.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKhjhs8ass877
Max-Forwards: 70
To: <sip:carol@chicago.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 63104 OPTIONS
Contact: <sip:alice@pc33.atlanta.com>
Accept: application/sdp
Content-Length: 0
```

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKhjhs8ass877
To: <sip:carol@chicago.com>;tag=93810874
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 63104 OPTIONS
Contact: <sip:carol@chicago.com>
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE
Accept: application/sdp
Accept-Encoding: gzip
Accept-Language: en
Supported: foo
Content-Type: application/sdp
Content-Length: 274

(SDP not shown)
```

ACK

- General
  - Cseq equal the requests being acknowledged
  - The ACK MUST contain a single Via header field, and this MUST be equal to the top Via header field of the original request.
  - Servers MUST NOT attempt to challenge an ACK
  - Different handling of Successful and Failure ACK!!
- 2xx
  - end-to-end
  - ACK is a new transaction
  - Body presents in delayed offer case.
  - 2xx retransmission ACK generation is handled by UA core.
- 3xx-6xx
  - hop-by-hop
  - ACK belongs the transaction created by the request
  - Placement of bodies in ACK for non-2xx is NOT RECOMMENDED.
  - Retransmission is handled by transaction layer.

Cancel

- Cancel in progrss request
- A UAS that receives a CANCEL request for an INVITE, but has not yet sent a final response, would "stop ringing", and then respond to the INVITE with a specific error response (a 487).
- CSeq equal the requests being.
- CANCEL is best for INVITE requests, which can take a long time to generate a response.
- If no provisional response has been received, the CANCEL request MUST NOT be sent; rather, the client MUST wait for the arrival of a provisional response before sending the request.
- A stateful proxy responds to a CANCEL, rather than simply forwarding a response it would receive from a downstream element. For that reason, CANCEL is referred to as a "hop-by-hop" request, since it is responded to at each stateful proxy hop.
- hop-by-hop and cannot be resubmitted, cannot be challenged.
- CANCEL request SHOULD be accepted by a server if it comes from the same hop that sent the request being canceled.

Transaction Layer

- FSM (Finite State Machine)
  - Client INVITE
  - Client non-INVITE
  - Server INVITE
  - Server non-INVITE
- Look for all state machines in RFC3261
- Timers
  - T1,T2,T4
  - Timer A,B,C,D,E,F,G,H,J,K
  - Recommended default values
  - timeout, retransmit

```
                                  |INVITE from TU
                Timer A fires     |INVITE sent
                Reset A,          V                      Timer B fires
                INVITE sent +-----------+                or Transport Err.
                  +---------|           |---------------+inform TU
                  |         |  Calling  |               |
                  +-------->|           |-------------->|
                            +-----------+ 2xx           |
                               |  |       2xx to TU     |
                               |  |1xx                  |
         300-699 +---------------+  |1xx to TU            |
            ACK sent |               |                     |
         resp. to TU |  1xx          V                     |
                     |  1xx to TU  -----------+            |
                     |  +---------|           |            |
                     |  |         |Proceeding |------------->|
                     |  +-------->|           | 2xx        |
                     |            +-----------+ 2xx to TU  |
                     |       300-699    |                  |
                     |       ACK sent,  |                  |
                     |       resp. to TU|                  |
                     |                  |                  |      NOTE:
                     |  300-699         V                  |
                     |  ACK sent  +-----------+Transport Err. |  transitions
                     |  +---------|           |Inform TU   |  labeled with
                     |  |         | Completed |------------->|  the event
                     |  +-------->|           |            |  over the action
                     |            +-----------+            |  to take
                     |              ^   |                  |
                     |              |   | Timer D fires    |
                    +--------------+   | -                |
                                       |                  |
                                       V                  |
                                 +-----------+            |
                                 |           |            |
                                 | Terminated|<-------------+
                                 |           |
                                 +-----------+
```

Authentication

- Basic authentication obsoleted and is not allowed
- Modified HTTP digest (MD5)
  - How it works?
    - HA1=MD5(A1)=MD5(username:relam:password)
    - HA2=MD5(A2)=MD5(method:digestURI)
    - response=(HA1:nonce:HA2)
  - The password on AAA server can be stored only in from of clear text or HA1 hash.
- ACK
  - Problems come up for any requests that take no response, including ACK. Any credentials in the INVITE that were accepted by a server MUST be accepted by that server for the ACK.
  - Servers MUST NOT attempt to challenge an ACK
- Cancel
  - Although the CANCEL method does take a response (a 2xx), servers MUST NOT attempt to challenge CANCEL requests since these requests cannot be resubmitted.

Auth & Identity

- http://www.tech-invite.com/Ti-sip-identity.html

Locating endpoint

- How to find an UA from AoR?
- Location lookup sources
  - Location Service / Location database
    - Register
    - used by proxies
  - DNS
    - RFC3263
    - SRV, NAPTR
    - Can be used by UA or proxy
  - E164 URI mapping (ENUM)
    - NAPTR
    - Can be used by UA or Proxy
  - Local database / Local policy
    - Alias
    - Local rewrite policy (normalization)
    - white list / black list
    - etc.

Registration

```
                                                  bob
                                                +----+
                                                | UA |
                                                |    |
                                                +----+
                                                   |
                                                   |3)INVITE
                                                   |   carol@chicago.com
      chicago.com        +--------+               V
      +---------+ 2)Store|Location|4)Query +-----+
      |Registrar|=======>| Service|<=======|Proxy|sip.chicago.com
      +---------+        +--------+=======>+-----+
            A                       5)Resp      |
            |                                   |
            |                                   |
  1)REGISTER|                                   |
            |                                   |
        +----+                                  |
        | UA |<---------------------------------+
cube2214a|    |                            6)INVITE
        +----+                     carol@cube2214a.chicago.com
         carol
```

Registration (binding)

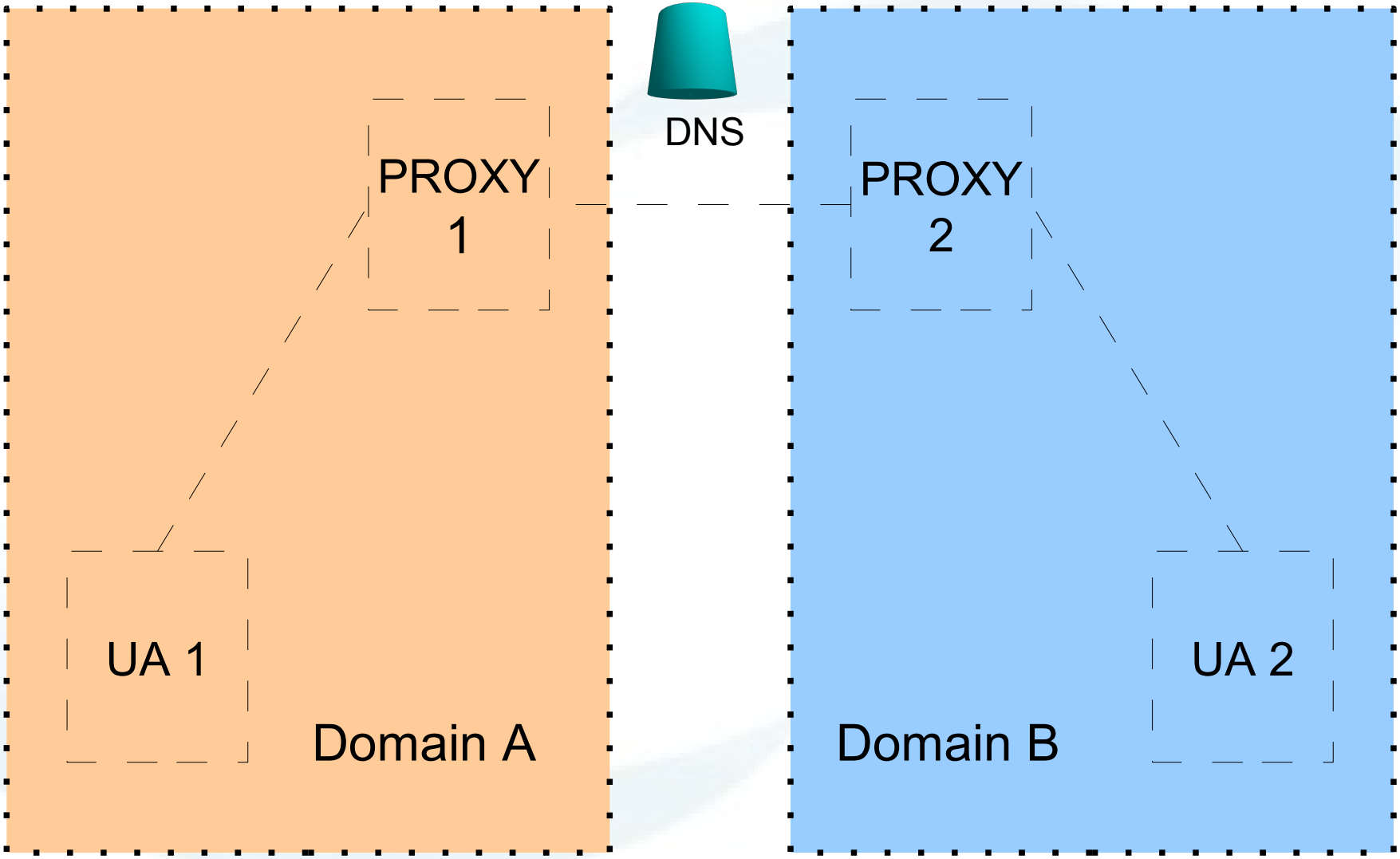
- Registration creates bindings in a location service for a particular domain that associates an address-of-record URI with one or more contact addresses.
- expires: The "expires" parameter indicates how long the UA would like the binding to be valid. The value is a number indicating seconds.
- REGISTER requests add, remove, and query bindings.
  - **Add** One or more contact header

    example: Contact: sip:misi@alma.ki.niif.hu
    - **Expire** is not 0
      example: expire: 3600
    - **q parameter** (default 1)
  - **Remove** expire: 0
    - **Remove all** expire: 0 + Contact: “*”
  - **Query** A success response to any REGISTER request contains the complete list of existing bindings, regardless of whether the request contained a Contact header field. If no Contact header field is present in a REGISTER request, the list of bindings is left unchanged.

Registration example message

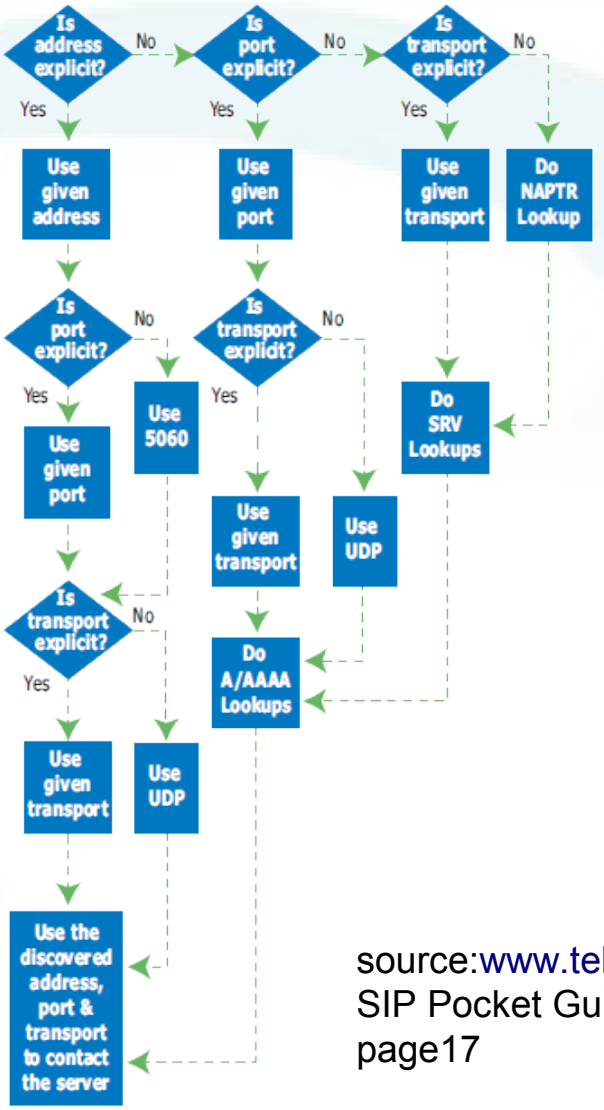
- http://www.tech-invite.com/Ti-sip-ex3261.html#figa

SIP Trapezoid



RFC3263

- From uri domain part discovery of transport protocol and socket informations
  - transport
  - ip
  - port
- Defaults
  - default port: 5060
  - default transport: udp
- DNS
  - NATPR
    - DDS
  - SRV
  - A/AAA



source:www.tekelc.com
SIP Pocket Guide
page17

DNS SRV, NAPTR

- bind
  - named-checkconf
- Dynamic Delegation Discovery System (DDDS)
  - rfc3401, rfc3402, rfc3403, rfc3404
  - Naming Authority Pointer (NAPTR) Resource Record (RR)
  - key domain-names
- Example picece from DNS ZONE file
  - NAPTR

```
;;                  order pref flags service    regexp replacement
atlanta.test IN NAPTR 10 50   "s"   "SIPS+D2T" ""     _sips._tcp
atlanta.test IN NAPTR 30 50   "s"   "SIP+D2U"  ""     _sip._udp
```

  - SRV

```
;;   priority          Weight  Port     Target
_sip._tcp IN SRV       10      0        5060     a1
_sip._udp IN SRV       10      0        5060     a1
```

DDDS Algorithm RFC3402

```
+--------- Application Unique String
|                 +-----+
|                 |input|
|         +-------+     +---------+
|         | First Well Known Rule |
|         +-------+      +--------+
|                 |output|
|                 +------+
|                First Key
|                    |
|                    +----<--------------<--------------+
|                    |                                  |
|                   key     (a DDDS database always     |
|                 +-----+    takes a key and returns    |
|                 |input|    a rule)                   ^
|       +---------+     +------------+                  |
|       | Lookup key in DDDS Database|                  |
|       +---------+      +-----------+                  |
|                 |output|                              |
|                 +------+                              |
|                 rule set                              |
|                    |                                  |
|                    |      (the input to a rule        |
|                 rule set  is the rule and the AUS.    ^
|                 +-----+   The output is always        |
+---------------->|input|   either a key or the result) |
  +---------------+     +------------------+            |
  | Apply Rules to Application Unique String|           |
  | until non-empty result are obtained     |           |
  | that meet the applications requirements |           |
  +---------------+      +-----------------+            |
                  |output|                              |
                  +------+                              ^
                   key                                  |
                    |                                   |
                    v                                   |
     +-------------------------------------+            |
     | Was the last matching rule terminal? | No >------+
     +-------------------------------------+
                   Yes     (if the rule isn't terminal then
                    |      its output is the new key which
                    |      is used to find a new rule set)
     +-----------------------------------+
     | The output of the last rule is the |
     | result desired by the application  |
     +-----------------------------------+
```

1. The First Well Known Rule is applied to the Application Unique String which produces a Key.
2. The Application asks the Database for the ordered set of Rules that are bound to that Key (see NOTE below on order details).
3. The Substitution Expression for each Rule in the list is applied, in order, to the Application Unique String until a non-empty string is produced. The position in the list is noted and the Rule that produced the non-empty string is used for the next step. If the next step rejects this rule and returns to this step then the Substitution Expression application process continues at the point where it left off. If the list is exhausted without a valid match then the application is notified that no valid output was available.
4. If the Service description of the rule does not meet the client's requirements, go back to step 3 and continue through the already retrieved list of rules. If it does match the client's requirements then this Rule is used for the next step. If and only if the client is capable of handling it and if it is deemed safe to do so by the Application's specification, the client may make a note of the current Rule but still return to step 3 as though it had rejected it. In either case, the output of this step is one and only one Rule.
5. If the Flags part of the Rule designate that this Rule is NOT Terminal, go back to step 2 with the substitution result as the new Key.
6. Notify the Application that the process has finished and provide the Application with the Flags and Services part of the Rule along with the output of the last Substitution Expression.

RFC3263 DDDS

- The following is the information that DDDS requests an application to provide:
  - Application Unique String:
    The Application Unique String (AUS) is the input to the resolution service. For this application, it is the URI to resolve.
  - First Well Known Rule:
    The first well known rule extracts a key from the AUS. For this application, the first well known rule extracts the host portion of the SIP or SIPS URI.
  - Valid Databases:
    The key resulting from the first well known rule is looked up in a single database, the DNS.
  - Expected Output:
    The result of the application is an SRV record for the server to contact.

NAPTR example

```
;;                          order pref flags service    regexp replacement
atlanta.test          IN NAPTR 10 10   ""    "SIPS+D2T" ""     redirect.atlanta.test
biloxi.test           IN NAPTR 10 10   ""    "SIPS+D2T" ""     redirect.atlanta.test
redirect.atlanta.test IN NAPTR 5 10    ""    "SIPS+D2T" "!^(biloxi\\.test)$!\1!"     .
redirect.atlanta.test IN NAPTR 10 10   ""    "SIPS+D2T" ""     redirect2.atlanta.test
redirect2.atlanta.test IN NAPTR 10 10  "s"   "SIPS+D2T" "!^.*$!_sip._tcp.\1!"     .

;;                            priority          Weight     Port      Target
_sip._tcp IN SRV    10                 0         5060      a1
_sip._udp IN SRV    10                 0         5060      a1
```

NAPTR

- **ORDER**

  A 16-bit unsigned integer specifying the order in which the NAPTR records MUST be processed in order to accurately represent the ordered list of Rules. The ordering is from lowest to highest. If two records have the same order value then they are considered to be the same rule and should be selected based on the combination of the Preference values and Services offered.

- **PREFERENCE**

  Although it is called "preference" in deference to DNS terminology, this field is equivalent to the Priority value in the DDDS Algorithm. It is a 16-bit unsigned integer that specifies the order in which NAPTR records with equal Order values SHOULD be processed, low numbers being processed before high numbers. This is similar to the preference field in an MX record, and is used so domain administrators can direct clients towards more capable hosts or lighter weight protocols. A client MAY look at records with higher preference values if it has a good reason to do so such as not supporting some protocol or service very well.

  DDS Priority: Simply a number used to show which of two otherwise equal rules may have precedence. This allows the database to express rules that may offer roughly the same results but one delegation path may be faster, better, cheaper than the other.

- The important **DIFFERENCE between ORDER and PREFERENCE** is that once a match is found the client MUST NOT consider records with a different Order but they **MAY** process records with the same Order but different Preferences.

- **REPLACEMENT** field and the **REGEXP** field together make up the Substitution Expression in the DDDS Algorithm. It is simply a historical optimization specifically for DNS compression that this field exists. **The fields are also mutually exclusive.**

SRV (RFC2782)

- **PRIORITY:** A client MUST attempt to contact the target host with the lowest-numbered priority it can reach; target hosts with the same priority SHOULD be tried in an order defined by the weight field. The range is 0-65535.
- **WEIGHT:** The weight field specifies a relative weight for entries with the same priority. Larger weights SHOULD be given a proportionately higher probability of being selected. The range of this number is 0-65535.
  Domain administrators **SHOULD use Weight 0** when there isn't any server selection to do, to make the RR easier to read for humans (less noisy). In the presence of records containing weights greater than 0, records with weight 0 should have a very small chance of being selected.
- **Ordering algorithm**
  In the absence of a protocol whose specification calls for the use of other weighting information, a client arranges the SRV RRs of the same Priority in the order in which target hosts, specified by the SRV RRs, will be contacted. **The following algorithm SHOULD be used to order the SRV RRs of the same priority:**
  To select a target to be contacted next, arrange all SRV RRs (that have not been ordered yet) in any order, except that all those with **weight 0 are placed at the beginning** of the list. Compute the sum of the weights of those RRs, and with each RR associate the running sum in the selected order. Then choose a uniform **random number** between 0 and the sum computed (**inclusive**), and **select** the RR whose running sum value is the **first in the selected order which is greater than or equal to the random number selected.** The target host specified in the selected SRV RR is the next one to be contacted by the client. **Remove this SRV RR from the set of the unordered SRV RRs and apply the described algorithm to the unordered SRV RRs to select the next target host.** Continue the ordering process until there are no unordered SRV RRs. **This process is repeated for each Priority.**

RFC3263 (Details of RFC2782 process)

RFC 2782 spells out the details of how a set of SRV records are sorted and then tried. However, it only states that the client should "try to connect to the (protocol, address, service)" without giving any details on what happens in the event of failure. Those details are described here for SIP.

For SIP requests, failure occurs if the transaction layer reports a 503 error response or a transport failure of some sort (generally, due to fatal ICMP errors in UDP or connection failures in TCP). Failure also occurs if the transaction layer times out without ever having received any response, provisional or final (i.e., timer B or timer F in RFC 3261 [1] fires). If a failure occurs, the client SHOULD create a new request, which is identical to the previous, but has a different value of the Via branch ID than the previous (and therefore constitutes a new SIP transaction). That request is sent to the next element in the list as specified by RFC 2782.

RFC3263

- NAPTR
  - First, a client resolving a **SIPS** URI MUST discard any services that do not contain "SIPS" as the protocol in the service field. The converse is not true, however. A client resolving a SIP URI SHOULD retain records with "SIPS" as the protocol, if the client supports TLS.
  - Second, a client MUST discard any service fields that identify a resolution service whose value is not "D2X", for values of X that indicate **transport protocols supported by** the **client**.
  - For NAPTR records with **SIPS** protocol fields, (if the server is using a site certificate), the **domain name in the query** and the domain name in the replacement field MUST both be **valid** based on the **site certificate** handed out by the server in the TLS exchange. Similarly, the domain name in the SRV query and the domain name in the target in the SRV record MUST both be valid based on the same site certificate.
  - **fallback:** If no NAPTR records are found, the client constructs SRV queries for those transport protocols it supports, and does a query for each. Queries are done using the service identifier "_sip" for SIP URIs and "_sips" for SIPS URIs.
- SRV
  - If no SRV records are found, the client SHOULD use TCP for a SIPS URI, and UDP for a SIP URI.

SRV weigth example

- zone file

```
;;                priority Weight Port Target
_sip._tcp IN SRV  10       0      5060 a1
_sip._tcp IN SRV  10      10      5060  a2
_sip._tcp IN SRV  10      10      5060  a3
_sip._tcp IN SRV  10      20      5060  a4
```

- Step0: create a list with result. (Repeat this algorithm foreach priority!)
- Step1: Put RR-s with weight 0 to beginning of the list
- Step2: Sum weight 10+10+20=40
- Step3: Compute running sum associated each record
  - a1 0
  - a2 1-10
  - a3 11-20
  - a4 20-40
- Step4: random number between 0- sum(weight) /inclusive!/
- Step5: Find the (lucky) winner add to the end of the final result list
- Step6: Remove the winner from the list
- Step7: If we need the next RR go to step1 until list is not empty.

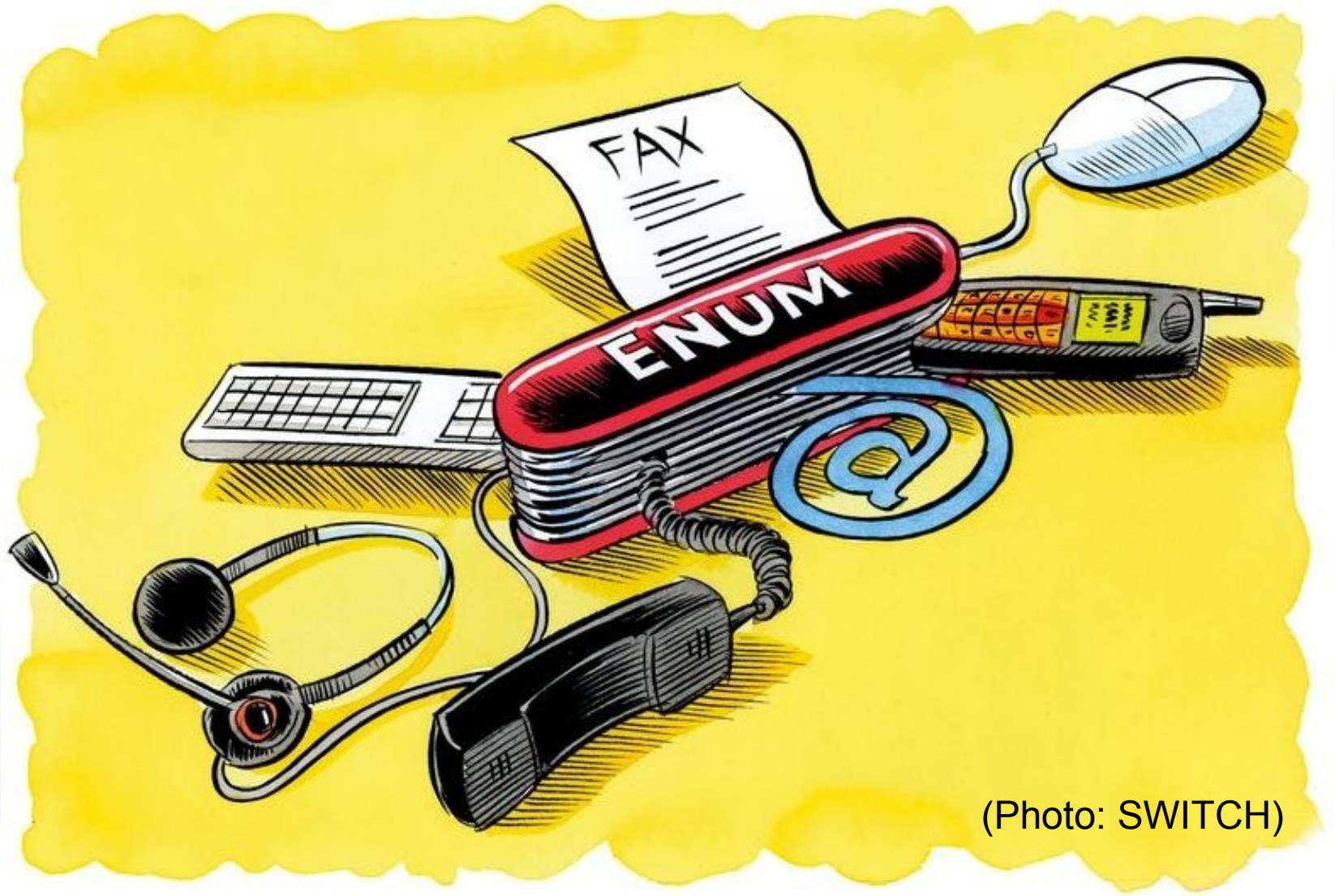
DNS Zone Example

```
$TTL    60
$ORIGIN atlanta.test.
@       IN      SOA     a1.atlanta.test. root.a1.atlanta.test. (
                     2009071301         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@               IN      NS      a1
@               IN      NS      a2
a1              IN      A       195.111.158.41
a2              IN      A       195.111.158.42

;;                              order   pref    flags   service         regexp replacement
@               IN      NAPTR   10      10      "s"     "SIPS+D2T"      ""     _sips._tcp
@               IN      NAPTR   20      10      "s"     "SIP+D2T"       ""     _sip._tcp
@               IN      NAPTR   30      10      "s"     "SIP+D2U"       ""     _sip._udp

;;                              Priority        Weight  Port    Target
_sips._tcp      IN      SRV     10              0       5061    a1
_sips._tcp      IN      SRV     20              0       5061    a2
_sip._tcp       IN      SRV     10              0       5060    a1
_sip._tcp       IN      SRV     20              0       5060    a2
_sip._udp       IN      SRV     10              0       5060    a1
_sip._udp       IN      SRV     20              0       5060    a2
```

ENUM



(Photo: SWITCH)

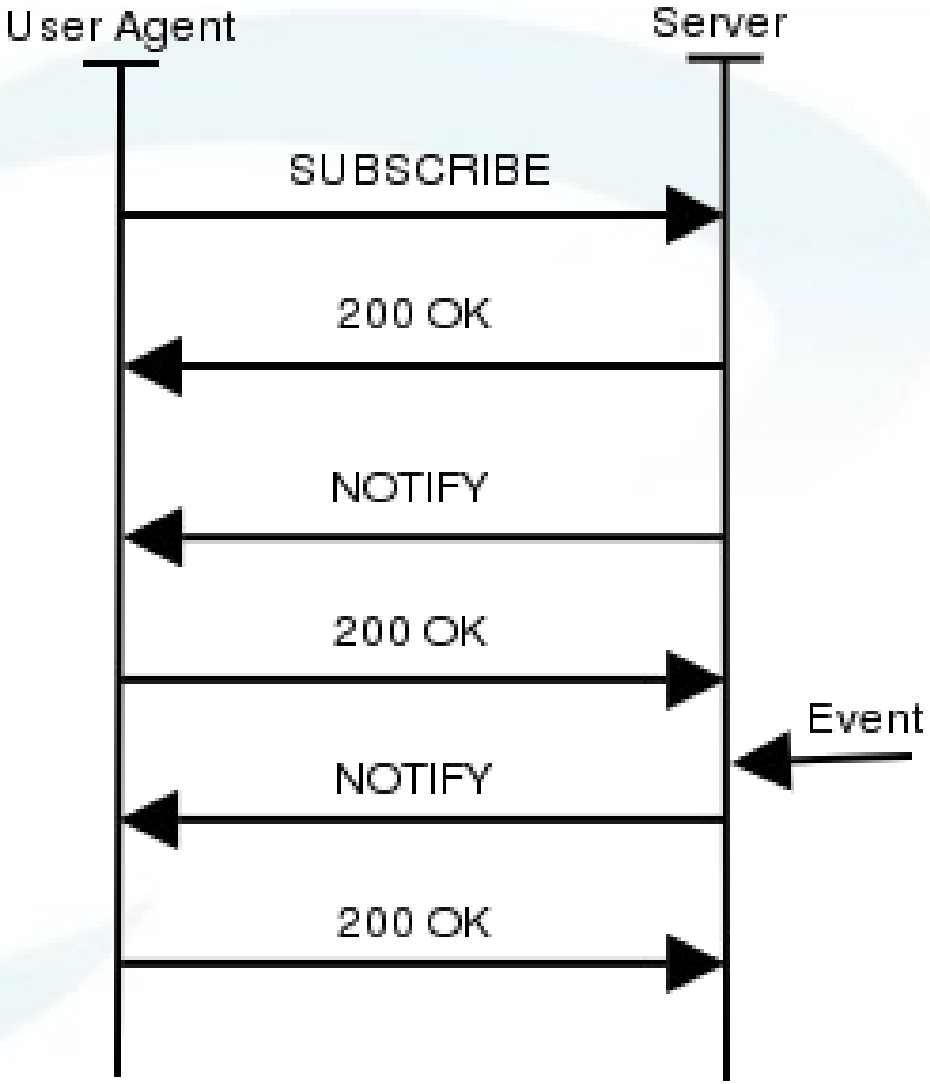
ENUM example

- Steps From E.164 to URI
  - +36-1-4503060
  - 3614503060
  - 0.6.0.3.0.5.4.1.6.3
  - 0.6.0.3.0.5.4.1.6.3.e164.arpa
- NAPTR

```
;;                            order pref flags service   regexp                            replacement
6.8.4.6.1.1.4.1.6.3 IN NAPTR 10    10   "u"   "e2u+sip" "!^(.*)$!sip:\\1@195.111.104.22!" .
```

SUBSCRIBE-NOTIFY

- Subscription to asynchronous evenet
- Notfiy everytime the event occurs
- Limited lifetime, must be preiodically refreshed.
- Usage
  - Refer
  - Presence
  - Voicemail



SIMPLE (Presence)

```
                 +---------------------------+
                 |     PRESENCE SERVICE      |
                 +---------------------------+
                     ^                     |
                     | PRESENCE PROTOCOL |
                     |                     v
               +------------+       +------------+
               | PRESENTITY |       |  WATCHER   |
               +------------+       +------------+
                     ^                    ^
                     |                    |
                     |                    |
      o      +--------------+      +--------------+      o
     /|\  -->| PRESENCE UA  |      | WATCHER UA  |<--  /|\
      X      +--------------+      +--------------+      X

    (PRINCIPAL)                                       (PRINCIPAL)
```

SIMPLE (Messaging)

```
                 +---------------------------+
                 |  INSTANT MESSAGE SERVICE  |
                 +---------------------------+
                     ^                       |
                   IM|   INSTANT MESSAGE     |IM
                     |       PROTOCOL        v
              +------------+        +---------------+
              |   SENDER   |        | INSTANT INBOX |
              +------------+        +---------------+
                    ^                       ^
                    |                       |
                    |                       |
   o        +-------------+        +------------------+      o
  /|\   -->|  SENDER UA  |        |  INBOX UA        |<--  /|\
   X        +-------------+        +------------------+      X

 (PRINCIPAL)                                            (PRINCIPAL)
```

Service Examples RFC3665

- http://www.tech-invite.com/Ti-sip-CF3665.html

References

- http://www.tech-invite.com/
- http://www.iptel.org/files/sip_tutorial.pdf
- http://tools.ietf.org/html/rfc3261
- http://tools.ietf.org/html/rfc3263
- http://ftp.iptel.org/pub/ser/0.8.14/doc/html/sip_introduction.html
- http://www.switch.ch
- http://www.voip-info.org
- http://www.cs.columbia.edu/sip/
- http://www.jdrosen.net/
- http://en.wikipedia.org/wiki/Session_Initiation_Protocol
- http://www.iana.org/assignments/sip-parameters
- http://www.crt.realtors.org/projects/rets/variman/support/digest.php
- http://www.rfc3261.net/
- http://www.unc.edu/~cschlatt/docs/SIP_failover_and_redundancy.pdf
- http://nrenum.net/
- http://enumdata.org/
- http://www.iana.org/assignments/enum-services
- http://tools.ietf.org/html/rfc2778

Questions ?

misi@niif.hu