SIP - Session Initiation Protocol

courtesy of

Marco Sommani

CNR-IIT, Pisa

marco.sommani@cnr.it

SIP

- SIP (Session Initiation Protocol) è un protocollo utilizzato per iniziare, modificare o terminare sessioni fra uno o più partecipanti. (RFC 3261)
  - permette ai partner di scoprire i rispettivi indirizzi e port number
  - permette ai partner di concordare le modalità con cui scambiarsi i dati
  - sessioni multimediali, ma anche chat, condivisione della lavagna ed altro
- Dove possibile, si appoggia su altri standard IETF:
  - Le modalità con cui scambiarsi i dati vengono concordate trasportando all'interno dei messaggi SIP informazioni SDP (Session Description Protocol, RFC 3227)
  - Per gli indirizzi si usano gli Uniform Resource Identifiers (URI, RFC 3986):
    - sip:marco.sommani@iit.cnr.it -- "sip:" ==> trasporto UDP o TCP
    - sips:+004437612234@sip-proxy.org:5062 -- "sips:" ==> trasporto TLS
  - Per il traffico multimediale si usa di preferenza RTP (RFC 3550, 5506)

Gli RFC 326x (giugno 2002)

- 3261 “SIP: Session Initiation Protocol”
  - l’RFC principale. Insieme agli altri rende obsoleto l’RFC 2543
- 3262 “Reliability of Provisional Responses in Session Initiation Protocol (SIP)”
- 3263 “Session Initiation Protocol (SIP): Locating SIP Servers”
  - come individuare il next-hop a cui inviare la richiesta
- 3264 “An Offer/Answer Model with Session Description Protocol (SDP)”
  - come utilizzare il protocollo SDP in combinazione con il SIP
- 3265 “Session Initiation Protocol (SIP)-Specific Event Notification”
  - definisce le richieste opzionali *SUBSCRIBE* e *NOTIFY*
- Aggiornamenti negli RFC 3835, 4320, 4916 e 5393
- 3581 “Symmetric Response Routing”
  - definisce parametro rport per seguire porte cambiate da firewall simmetrico

Messaggi SIP

- SIP si basa su **transazioni** in cui entità dette User Agents (UA) si scambiano messaggi ASCII
- Una **transazione** inizia con una *Request* inviata da uno User Agent Client (UAC) ad uno User Agent Server (UAS) e termina con una *Final Response* inviata in senso inverso
  - la *Final Response* può essere preceduta da una o più *Provisional Responses*
  - la *Request* "ACK" non prevede risposte
- I messaggi possono transitare attraverso uno o più Proxy Server
- La prima riga di una *Request* contiene il nome del *metodo* usato nella transazione
  - es: INVITE, CANCEL, ACK, BYE, REGISTER, OPTIONS...
- La prima riga di una *Response* contiene un *codice di stato*
  - codici 1xx: *Provisional Responses*
  - codici 2xx, 3xx, 4xx, 5xx, 6xx: *Final Responses*
- La prima riga dei messaggi è seguita da un certo numero di *Headers*
- L'ultimo Header è seguito da una riga vuota, dopo la quale può essere presente il *message body* (es.: SDP)

Esempio di transazione SIP

- *Request* “BYE”:


```
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
```

- *Final Response* alla *Request* “BYE”:

```
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
```

La SIP URI

- Identifica le entità SIP:

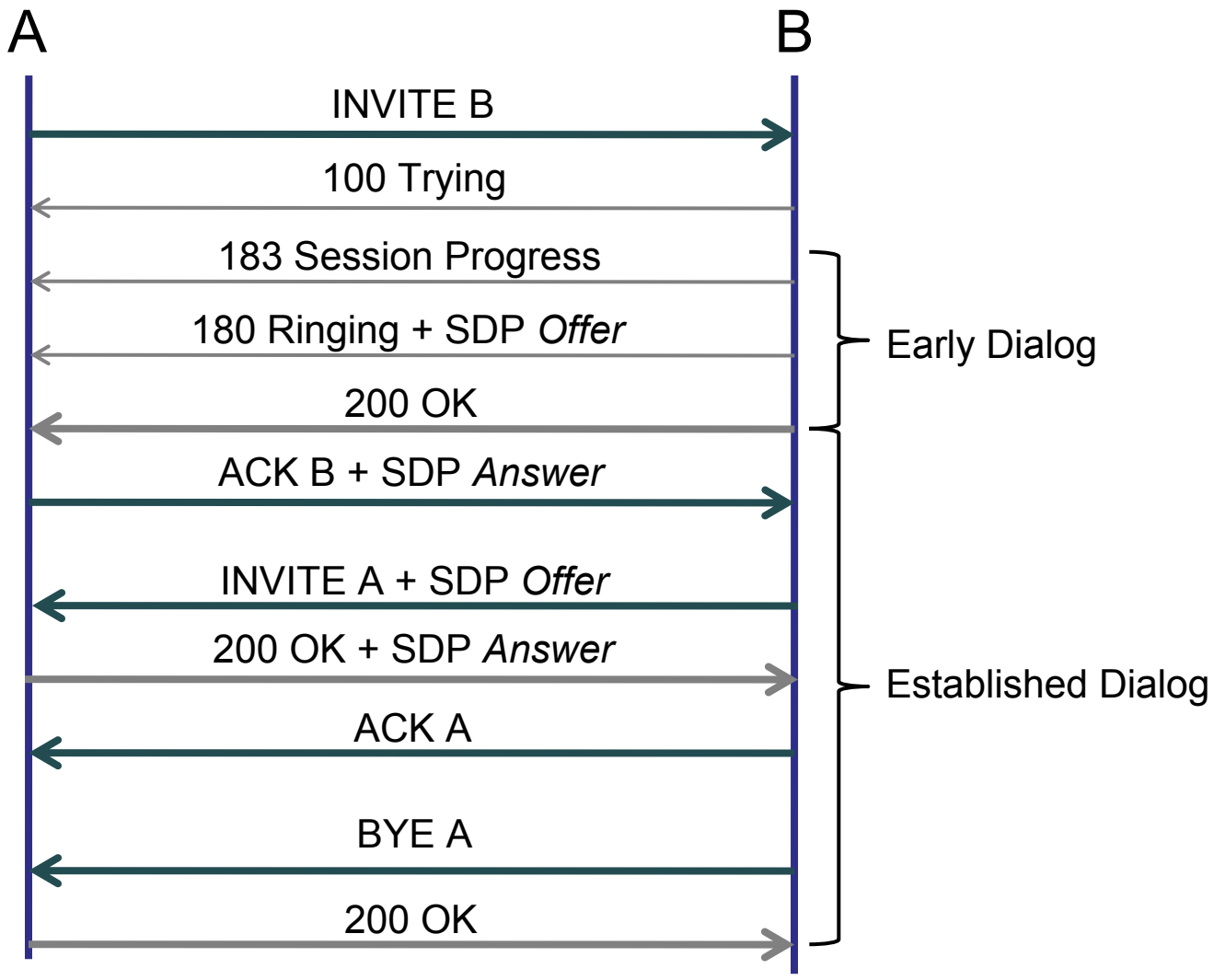
  {sip|sips}:[user-part@]domain-part[:port][;transport={UDP|TCP|TLS|...}]

  - le indicazioni su porta e/o trasporto sono consentite solo se la *domain-part* identifica un particolare host:
    - *fully qualified domain name* presente nel DNS con record A o AAAA
    - indirizzo IPv4 o IPv6
  - la *domain-part* può anche indicare un **dominio SIP**, convertibile in indirizzo di trasporto consultando i record NAPTR e SRV del DNS
- La URI prende il nome di:
  - ***contact-address*** (indirizzo reale): se permette di ricostruire gli indirizzi di trasporto dell'entità SIP stessa
  - ***address-of-record*** (indirizzo pubblico): se permette di ricostruire gli indirizzi di trasporto di un server in grado di localizzare l'entità SIP

Sessioni e dialoghi

- La **sessione** è un flusso di dati, generalmente multimediali, unidirezionale o bidirezionale, controllato da due UA SIP
  - due UA SIP concordano le sessioni scambiando messaggi SDP trasportati come *message-body* nei messaggi SIP
  - per attivare la sessione un UA invia un messaggio SDP di *Offer* e l'altro risponde con un *Answer*
    - caso 1: *Offer* nella INVITE *Request* e *Answer* in una*Response*
    - caso 2: *Offer* in una *Response* e *Answer* nella ACK *Request*
- Il **dialogo** è una associazione logica fra due UA SIP, che viene attivata o disattivata per mezzo di transazioni SIP
  - non tutte le transazioni SIP fanno parte di un dialogo
  - Gli scambi di informazioni SDP relative alle sessioni avvengono fra UA che hanno attivato o stanno attivando un dialogo

Dialogo con negoziazione di sessioni



Metodi SIP

- L'RFC 3261 definisce i seguenti metodi:

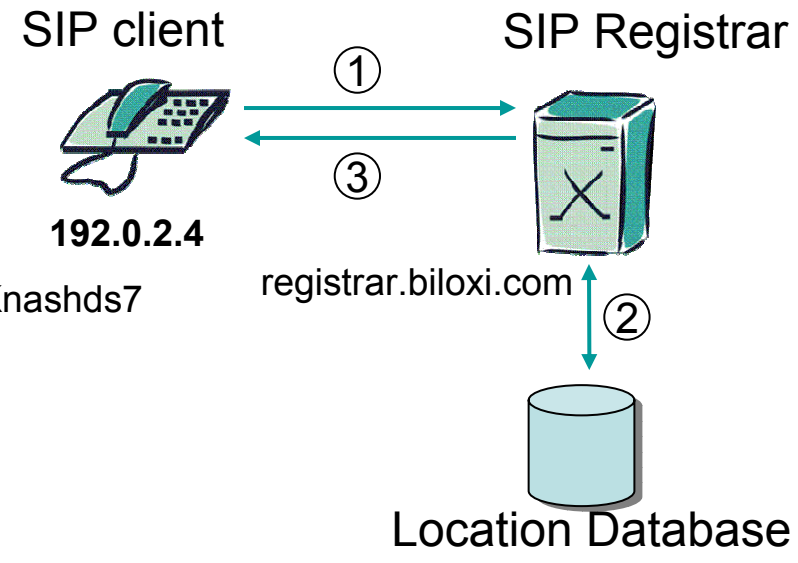
INVITE	inizia dialoghi con le relative sessioni o modifica le sessioni di un dialogo già iniziato
ACK	segue immediatamente la transazione INVITE (non è seguita da Response)
BYE	termina un dialogo e tutte le sessioni associate al dialogo
CANCEL	cancella una *Request* pendente (generalmente INVITE)
REGISTER	comunica a un SIP registrar uno o più contact-addresses da associare ad un address-of-record
OPTIONS	richiesta di informazioni su funzionalità e stato

- RFC successivi hanno introdotto altri metodi:
  - es.: SUBSCRIBE, NOTIFY, INFO, UPDATE, MESSAGE, ...

Elementi architetturali del SIP

- UserAgent Client (UAC)
  - entità che genera le *Request*
- UserAgent Server (UAS)
  - entità che riceve *Request* e può generare*Response* per accettarle, respingerle o indirizzarle altrove.
- Registrar Server
  - tipo speciale di UAS che accetta le *Request* REGISTER e memorizza le informazioni in esse contenute in un “location service”
- Proxy Server
  - entità che instrada le *Request* verso gli UAS e le *Response* verso gli UAC
  - può rispondere direttamente ad una *Request* (in tal caso opera come UAS)
  - può rimpiazzare la *Request URI* con una nuova
    - servendosi di informazioni configurate staticamente
    - consultando il location service di un registrar server, spesso co-locato
    - consultando il DNS (o ENUM)

SIP: esempio di registrazione



1. Il client SIP invia una *Request* REGISTER a registrar.biloxi.com

REGISTER sip:registrar.biloxi.com SIP/2.0
Via: SIP/2.0/UDP bobspc.biloxi.com:5060;branch=z9hG4bKnashds7
Max-Forwards: 70
To: Bob <**sip:bob@biloxi.com**>
From: Bob <sip:bob@biloxi.com>;tag=456248
Call-ID: 843817637684230@998sdasdh09
CSeq: 1826 REGISTER
Contact: <**sip:bob@192.0.2.4**>
Expires: 7200
Content-Length: 0

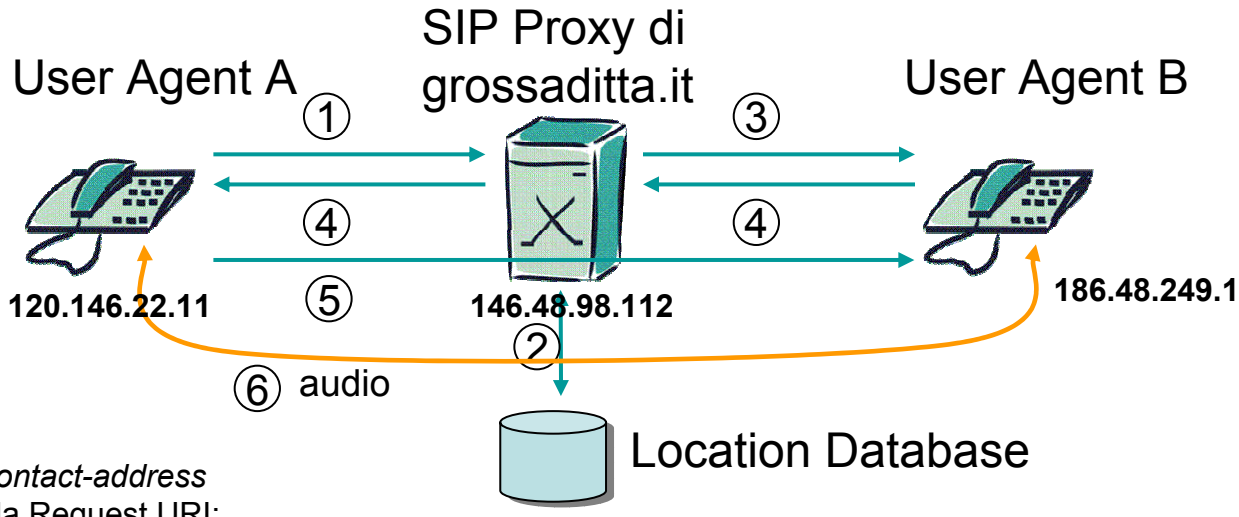
2. Il SIP registrar associa la URI del "Contact:" all'*address-of-record* presente nell'*Header* "To:"
3. Il SIP registrar risponde al client SIP comunicando la lista dei *contact-address* associati

SIP/2.0 200 OK
Via: SIP/2.0/UDP bobspc.biloxi.com:5060;.....
.....branch=z9hG4bKnashds7;received=192.0.2.4
To: Bob <**sip:bob@biloxi.com**>;tag=2493k59kd
From: Bob <sip:bob@biloxi.com>;tag=456248
Call-ID: 843817637684230@998sdasdh09
CSeq: 1826 REGISTER
Contact: <**sip:bob@192.0.2.4**>;expires=7200, ...
...<sip:robert@192.0.3.10>;expires=1320
Content-Length: 0

Attivazione di dialogo via SIP Proxy

1. l'UAC di A invia una INVITE al SIP Proxy "grossaditta.it" (l'indirizzo di trasporto viene ricavato dal DNS)

```
INVITE sip:bob@grossaditta.it SIP/2.0
Via: SIP/2.0/UDP 120.146.22.11:5060;....
To: sip:bob@grossaditta.it
From: sip:alice@grossaditta.it;tag=aaa...
Call-ID: ccc...
CSeq: 1 INVITE
Contact: sip:alice@120.146.22.11:5060
.....
```



2. Il SIP proxy cerca nel location database i *contact-address* associati all'*address-of-record*presente nella Request URI: sip:bob@grossaditta.it

3. Il SIP proxy inoltra l'INVITE all'unico *contact-address* di B trovato nel location database: sip:bob@186.48.249.1

```
INVITE sip:bob@186.48.249.1 SIP/2.0
Via: SIP/2.0/UDP 146.48.98.112:5060;....
Via: SIP/2.0/UDP 120.146.22.11:5060;....
To: sip:bob@grossaditta.it
From: sip:alice@grossaditta.it;tag=aaa...
Call-ID: ccc...
CSeq: 1 INVITE
Contact: sip:alice@120.146.22.11:5060
.....
```

4. Quando l'utente B alza la cornetta, l'UAS di B invia una *Response "*200 OK", che raggiunge l'UAC di A, transitando da tutti gli hop presenti nei "Via:" *Header*. Lo UserAgent A apprende dalla *Response* il *contact-address* dello UserAgentB

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 146.48.98.112:5060;....
Via: SIP/2.0/UDP 120.146.22.11:5060;....
To: sip:bob@grossaditta.it;tag=bbb...
From: sip:alice@grossaditta.it;tag=aaa...
Call-ID: ccc...
CSeq: 1 INVITE
Contact: sip:bob@186.48.249.1:5060
.....
```

5. L'UAC di A invia un ACK all'UAS di B mandandolo direttamente al *contact-address*presente nel "Contact:" Header della *FinalResponse*

```
ACK sip:bob@186.48.249.1:5060 SIP/2.0
Via: SIP/2.0/UDP 120.146.22.11:5060;....
To: sip:bob@grossaditta.it;tag=bbb...
From: sip:alice@grossaditta.it;tag=aaa...
Call-ID: ccc...
CSeq: 1 ACK
.....
```

6. Le sessioni audio/video fluiscono come concordato nella negoziazione SDP

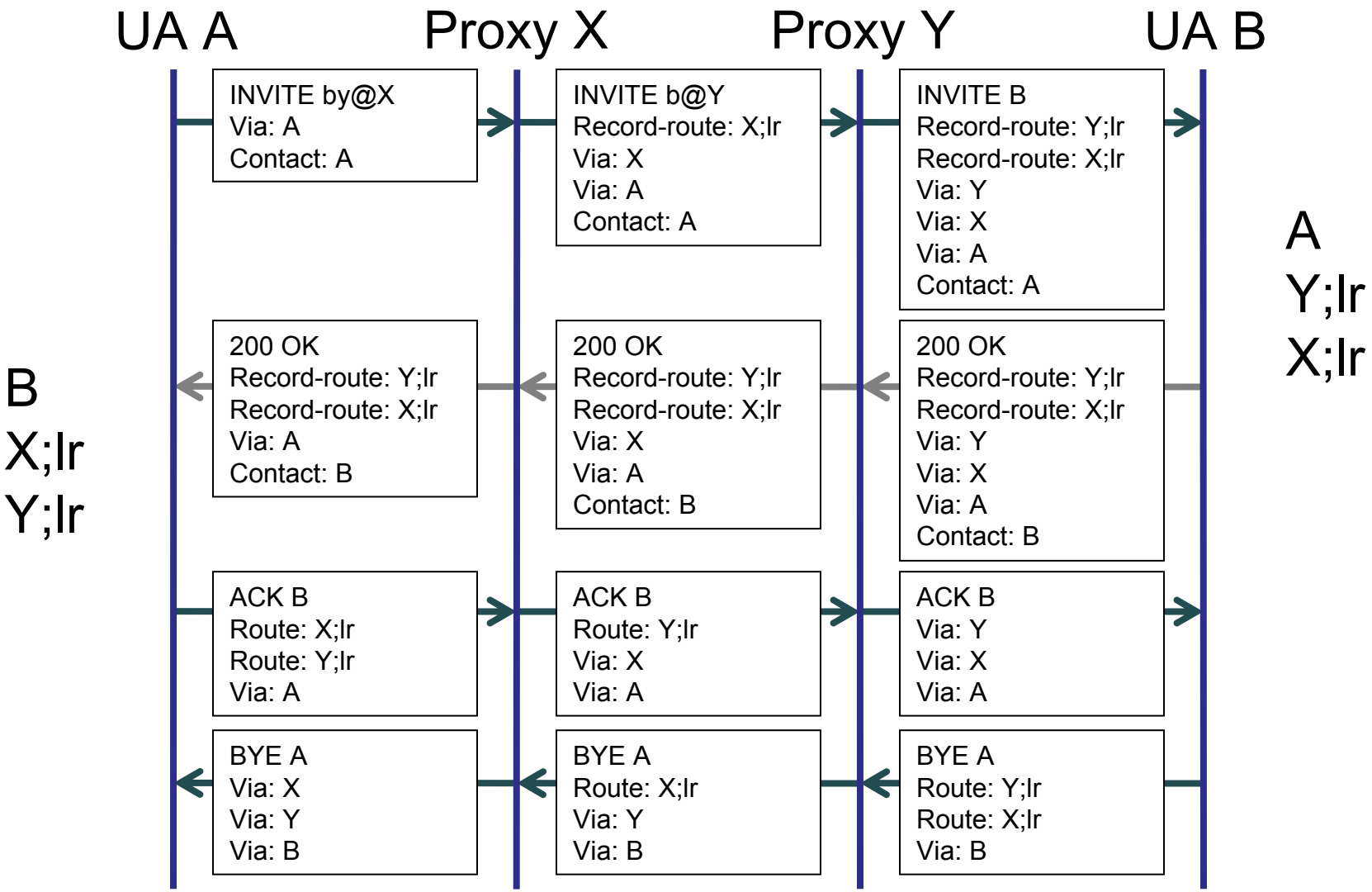
SIP URI nei diversi campi dei messaggi

Riga iniziale della Request (Request URI)	• *domain-part* =dominio SIP locale ➔processare la *Request* localmente, eventualmente sostituendo la *Request URI* • *domain-part* ≠ dominio SIP locale ➔ in mancanza di *Header* "Route:", la *domain-part* della*Request URI* indica il *next-hop*
Route:	• Il primo *Header* "Route:" indica il *next-hop* della *Request* • Chi riceve una *Request* cancella il primo *Header* "Route:" se la sua *domain-part* è un dominio SIP locale
From:	• Identifica l'originatore della *Request* • Dovrebbe giungere immutato al destinatario
To:	• Mostra al destinatario della Request quale era la URI originaria • La URI dovrebbe giungere immutata al destinatario • Nel REGISTER contiene l'address-of-record da (de)registrare
Contact:	• Nelle INVITE il campo è usato da ciascun UA per comunicare il proprio *contact-address* per le successive *Request* del dialogo • Non dovrebbe essere modificato (eccezione: indirizzi NAT-tati)

Ancora sui dialoghi

- Un dialogo è identificato da “Call-ID:”, *tag* del “From:” e *tag* del “To:”
- L’UAC mette nella *Request* iniziale il “Call-ID” e Il *tag* del “From:”
- L’UAS mette il *tag* del “To:” nella prima *Response* diversa da “100 Trying”
- Ogni *Proxy* intermedio può inserire nella *Request* iniziale un *Header* “Record-route:” contenente la propria URI
- L’UAS che riceve una *Request* iniziale deve:
  - memorizzare le URI dei “Record-route:” per costruire la lista dei “Route:” da inserire nelle successive richieste del dialogo
  - Copiare tutti i “Record-route”, così come erano presenti nella *Request,* nella *Final Response* che rende il dialogo *established* (codice 2xx)
- L’UAC che riceve la *Final Response* deve:
  - memorizzare in ordine inverso le URI dei “Record-route” per costruire la lista dei “Route:” da inserire nelle successive richieste del dialogo

“Route:” e “Record-route:” al lavoro



Il Forking

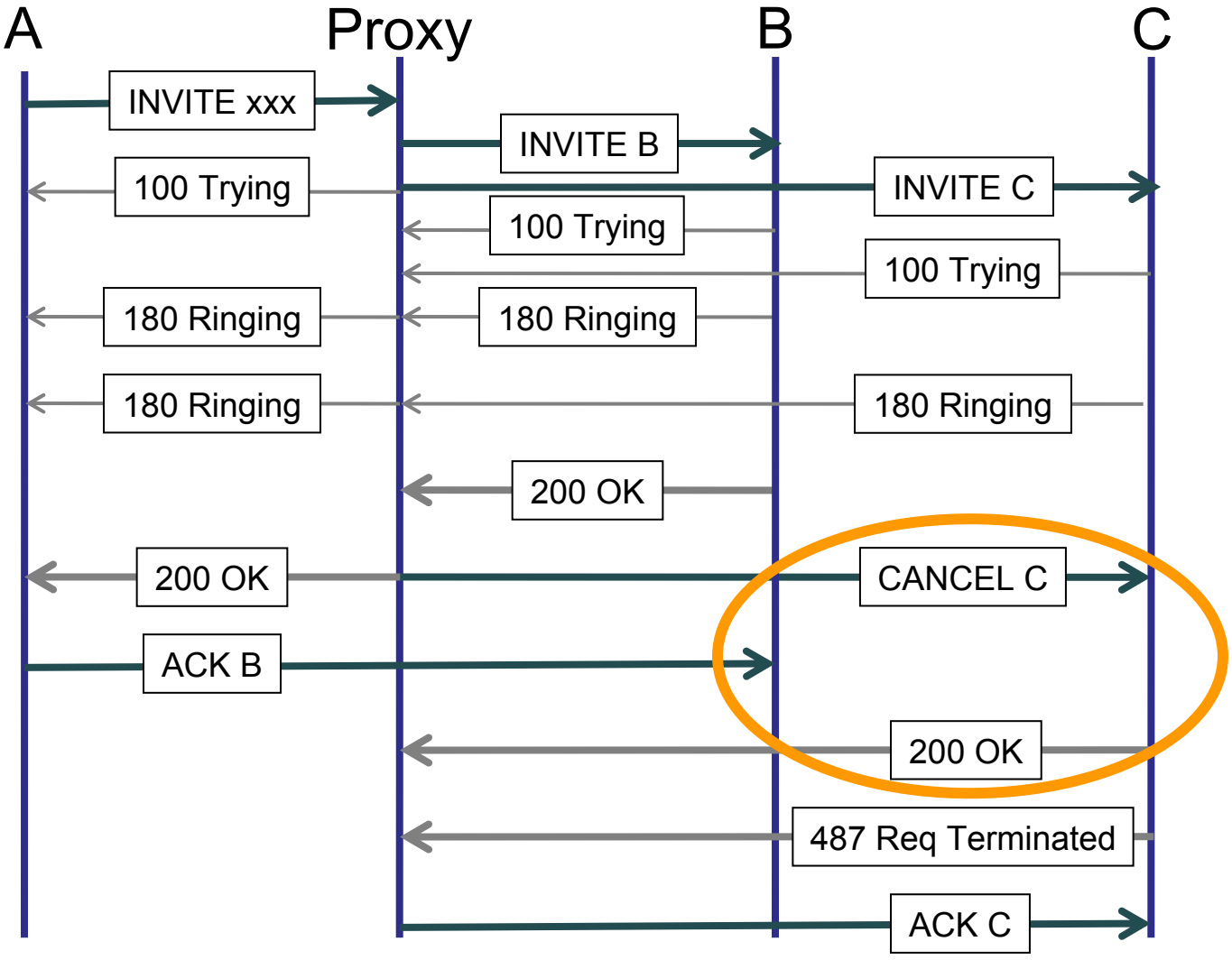
- Quando un SIP Proxy riceve una *Request* in cui la *domain-part* della*Request URI* indica il dominio locale, determina la nuova*Request URI* a cui inoltrare il messaggio
  - Consultando il *Location Database*
  - Usando altri meccanismi (es.: ENUM)
- Se le nuove *Request URI* sono più di una, il Proxy invia a ciascuna una copia della *Request*: ha luogo il **Forking**
- Il Proxy cerca di fare in modo che l'UAC riceva solo una*Final Response* (massimo un *established dialogue)*
- Prima della*Final Response,* l'UAC può ricevere *Provisional Responses* da più UAS, con conseguente attivazione di più *early dialogues*

Forking: trattamento delle *Response* sul Proxy

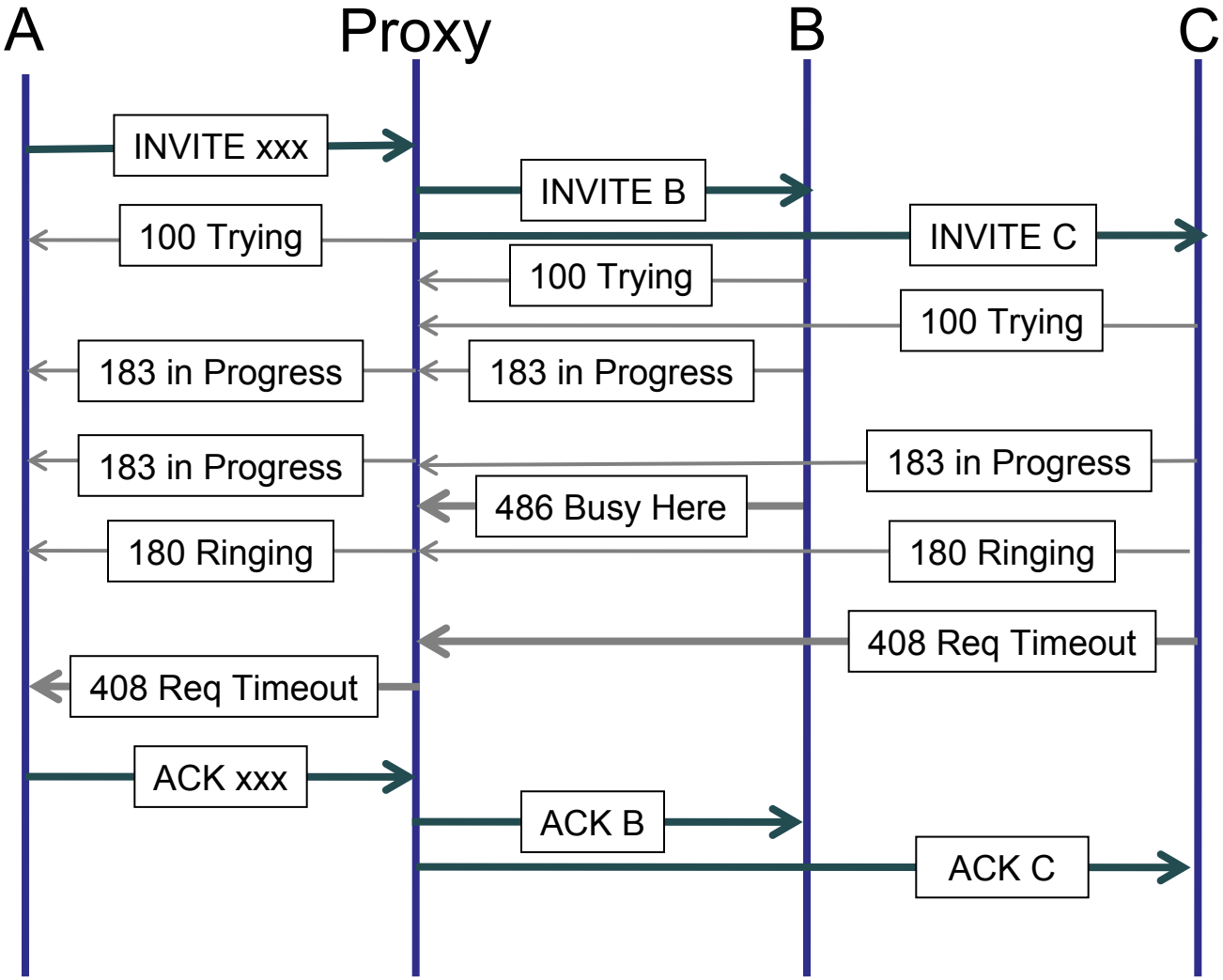
Response code	**Azioni**
100	• La *Response* si ferma al Proxy
1xx ≠ 100	• La *Response* prosegue verso l'UAC
2xx	• La Response prosegue verso l'UAC • il Proxy invia una CANCEL a tutti gli UAS che non hanno ancora inviato una *Final Response*

3xx, 4xx, 5xx, 6xx	• Vengono conservate dal Proxy fino a quando:
	  • Arriva una 2xx ➔tutte le altre*Final Response* in cache sono eliminate
	  • Tutti gli UAS hanno inviato una *Final Response* diversa da 2xx ➔il Proxy deve inviare una *Final Response* all'UAC, scegliendola fra quelle ricevute
	  • scade un timeout ➔all'UACviene inviata una "408 Request Timeout" e agli UAS che non hanno risposto una CANCEL

Forking con *Response* 200 OK



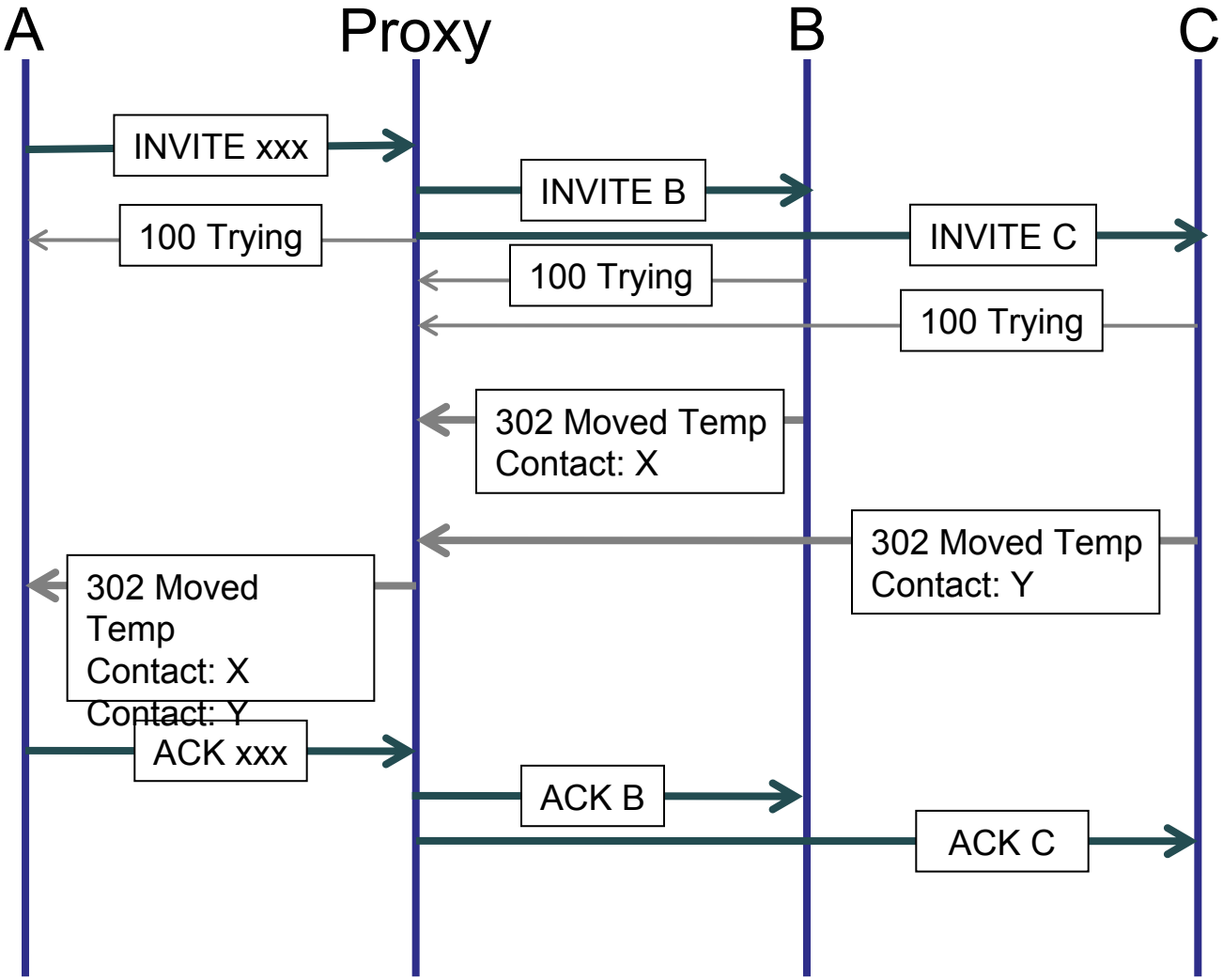
Forking con *Response* negative



Le *Final Response* con codice 3xx

- Con la *Response* 3xx l'UAS suggerisce all'UAC una lista di destinazioni (URI) alternative
  - le URI alternative sono inviate nell'*Header* "Contact:" della *Response*
- Lo standard lascia all'UAC ampie libertà di scelta su come utilizzare le informazioni contenute nel "Contact":
  - inviare la *Request* a tutte le nuove URI contemporaneamente
  - inviare le *Request* in successione
  - Mostrare I destinatari all'utente umano (display del telefono, messaggio vocale...)
  - non fare niente
- Talvolta i Proxy creano loro stessi le *Response* 3xx
  - così facendo si comportano da UAS e non da Proxy

Forking con *Response* 3xx



HTTP Authentication (1 di 2)

- L'RFC 3261 dà la possibilità ai Proxy ed agli UAS di accettare *Request* solo se l'UAC fornisce le sue credenziali
- Il meccanismo usato in SIP imita la "Digest Access Authentication" dell'HTTP (RFC 2617)
- Ad una *Request* con credenziali mancanti, invalide o scadute si risponde con
  - "401 Unauthorized" + *Header* "WWW-Authenticate:" (caso UAS)
  - "407 Proxy Authentication Required" + *Header* "Proxy-Authenticate:" (caso Proxy)
- Entrambi gli *Header* "*x*-Authenticate:" comunicano all'UAC
  - Il *realm* di chi chiede l'autenticazione

    (il realm e' una stringa che indica il dominio di protezione a cui si cerca di accedere; serve solo per ricordare all'utente quale username deve usare im quel dominio)
  - un *nonce* per calcolare la "Digest Authentication Response"

HTTP Authentication (2 di 2)

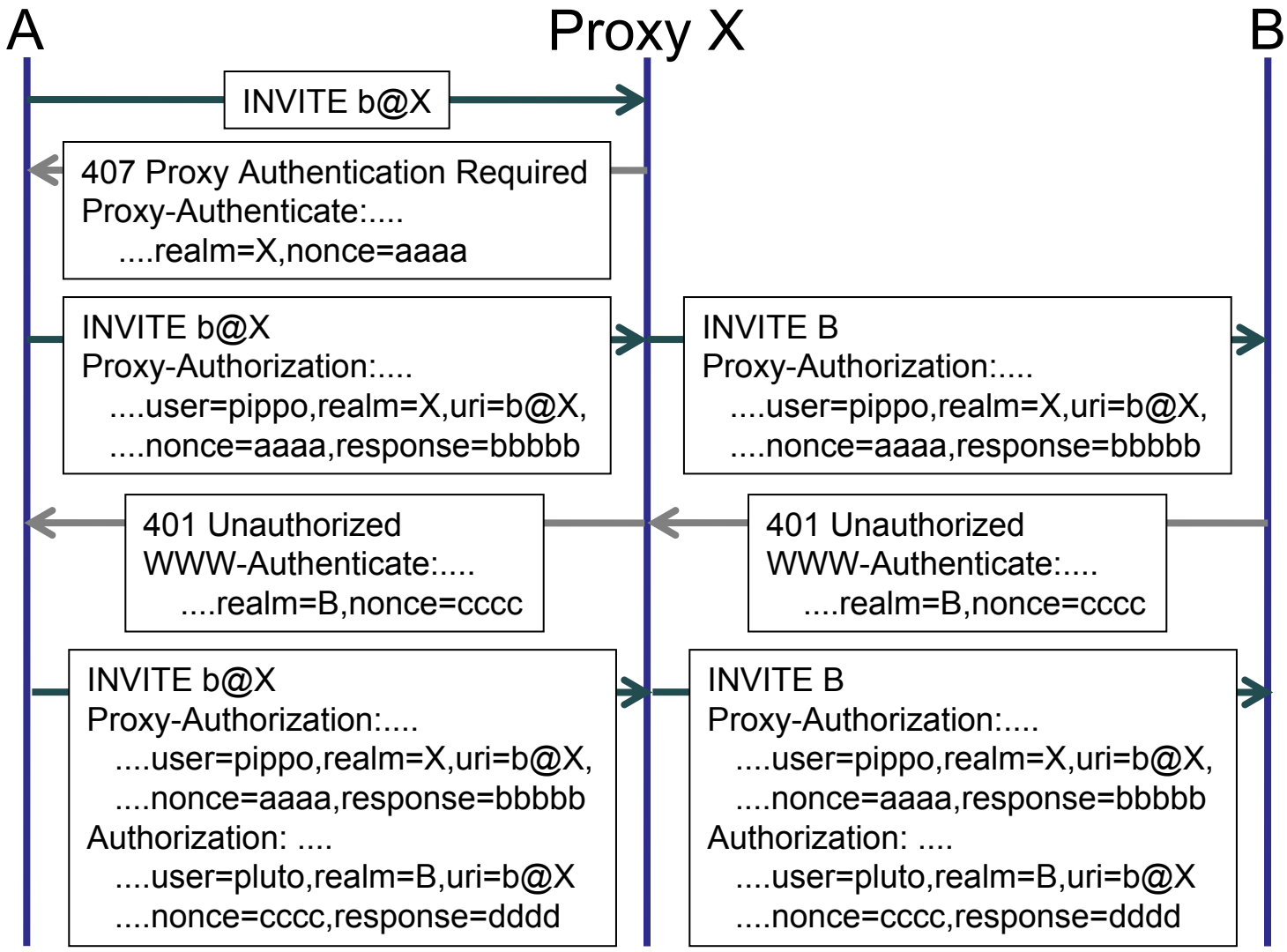
- L'UAC che riceve una *Response* 401 o 407 può rinunciare o ripetere la *Request* aggiungendo un *Header* "Authorization:" o "Proxy-Autorization" contenente
  - Il nome del *realm* per cui si forniscono le credenziali

    (il realm e' una stringa che indica il dominio di protezione a cui si cerca di accedere; serve solo per ricordare all'utente quale username deve usare im quel dominio)
  - lo *username* che l'UAC possiede in quel *realm*
  - la "Request URI" della*Request*
  - Il *nonce* ricevuto da chi ha inviato la *challenge*
  - Una stringa di 32 caratteri esadecimali, detta "Digest Authentication Response", calcolata a partire dai valori degli altri parametri e dalla password dello *username*
- In tutte le successive *Request* inviate allo stesso destinatario l'UAC continua a inserire gli *Header* "Authorization" e "Proxy-Authorization" fino alla ricezione di una nuova *challenge*

Proxy-Authenticate / Proxy-Authorization

- Dall'UAC parte la *Request*:
  - INVITE sip:0800202020@bh3.iit.cnr.it SIP/2.0
- Il Proxy bh3.iit.cnr.it risponde con un:
  - SIP/2.0 407 Proxy Authentication Required
- Nella *Response* 407 c'è il seguente *Header*:
  - Proxy-Authenticate: Digest realm="bh3.iit.cnr.it", nonce="4a210e6c72263543d1e916f211827b1084f2c330"
- Nelle successive *Request* dirette alla stessa destinazione:
  - Proxy-Authorization: Digest username="8327",realm="bh3.iit.cnr.it", nonce="4a210e6c72263543d1e916f211827b1084f2c330", uri="sip:0800202020@bh3.iit.cnr.it", response="129ceed44ef731662acac6f01574e080",algorithm=MD5

HTTP Authentication: flussodeimessaggi



Indirizzo di trasporto di un SIP domain

- Se *la domain-part* della *Request* URI è un *SIP domain*, le infomazioni per determinare l'indirizzo di trasporto del next-hop a cui inviare la *Request* si trovano su record DNS di tipo **SRV**
- La chiave (nome a dominio) del record SRV da cercare si ottiene così:
  - chiedere al DNS i record con dominio uguale alla *domain-part* e con tipo NAPTR e fra questi scegliere quelli con campo "service" uguale a:
    - "SIP+D2U" se come trasporto si vuole usare UDP
    - "SIP+D2T" se si vuole usare TCP
    - "SIPS+D2T" se si vuole usare TLS
  - in caso di successo, la chiave si trova nel campo "replacement" del NAPTR
  - altrimenti, il nome a dominio del record SRV si costruisce anteponendo alla *domain-part*le stringhe:
    - "_sip._udp." per il trasporto UDP, "_sip._tcp." per TCP, "_sips._tcp." per TLS
- In mancanza di SRV, la *domain-part* della URI è un nome a dominio del next-hop stesso (cercare A e/o AAAA)

Record SRV

- I record SRV (RFC 2782) hanno la seguente forma:

```
Domain TTL Class SRV Priority Weight Port Target
_sip._udp.bigu.edu. 43200 IN SRV 10 10 5060 proxy.bigu.edu.
```

- ***Domain*** è la chiave (nome a dominio) del Resource Record
- **SRV** è il tipo di record in esame
- ***Priority*** è la priorità: 10. Nel caso di più record SRV indica l'ordine di interrogazione (prima I più bassi)
- ***Weight*** è il peso: 10. Nel caso di più record SRV con stessa priorità, indica la frequenza con cui va utilizzato.
- ***Port*** è la porta: 5060.
- ***Target*** è il nome a dominio del next-hop: proxy.bigu.edu.

Formato del record NAPTR

- Il record NAPTR ha molteplici usi ed è descritto nello RFC 3403
- I campi usuali “Domain TTL Class NAPTR” sono seguiti da altri 6 campi:

  order preference flags services regexp replacement
  - I primi due sono numeri, gli altri stringhe racchiuse fra “”
  - *regexp* e *replacement* sono mutuamente esclusivi
  - con *flags* = **“s”** c’è solo il campo *replacement*
  - con *flags* = **“u”** c’è solo il campo *regexp*
  - al posto di un campo stringa assente ci vuole un punto
- Nei record contenenti le chiavi per la ricerca dei record SRV
  - *services* è **“SIP+D2x” o “SIPS+D2x”**, dove possibili valori di “x” sono “U” o “T”
  - il campo *flags* ha il valore **“s”**
- Esempi di NAPTR con chiavi di record SRV:

```
example.com. IN NAPTR 50   50  "s"  "SIPS+D2T"  .  _sips._tcp.example.com.
example.com. IN NAPTR 90   50  "s"  "SIP+D2T"   .  _sip._tcp.example.com.
example.com. IN NAPTR 100  50  "s"  "SIP+D2U"   .  _sip._udp.example.com.
```

Esempio di ricerca come da RFC 3263

- Devo mandare una INVITE a sip:marco.sommani@iit.cnr.it
- Chiedo al DNS i NAPTR per iit.cnr.it e ottengo:

```
iit.cnr.it. 300 IN NAPTR   90 50 "s" "SIP+D2T" . sip_tcp.iit.cnr.it.
iit.cnr.it. 600 IN NAPTR   90 50 "s" "SIP+D2U" . sip_udp.iit.cnr.it.
```

- Siccome voglio usare UDP, scelgo il secondo record
- Chiedo al DNS i record SRV per sip_udp.iit.cnr.it e ottengo:

```
sip_udp.iit.cnr.it. 43200 IN SRV   10 10 5060 bh3.iit.cnr.it.
```

- Ora so che il next-hop della richiesta è la porta 5060 del server bh3.iit.cnr.it
- In mancanza del record NAPTR, avrei cercato il record SRV con nome _sip._udp._iit.cnr.it.
- In mancanza dei record SRV, avrei tentato di mandare la richiesta alla porta 5060 di un host di nome “iit.cnr.it”, sperando di trovarlo

Different Categories of NATs and Firewalls

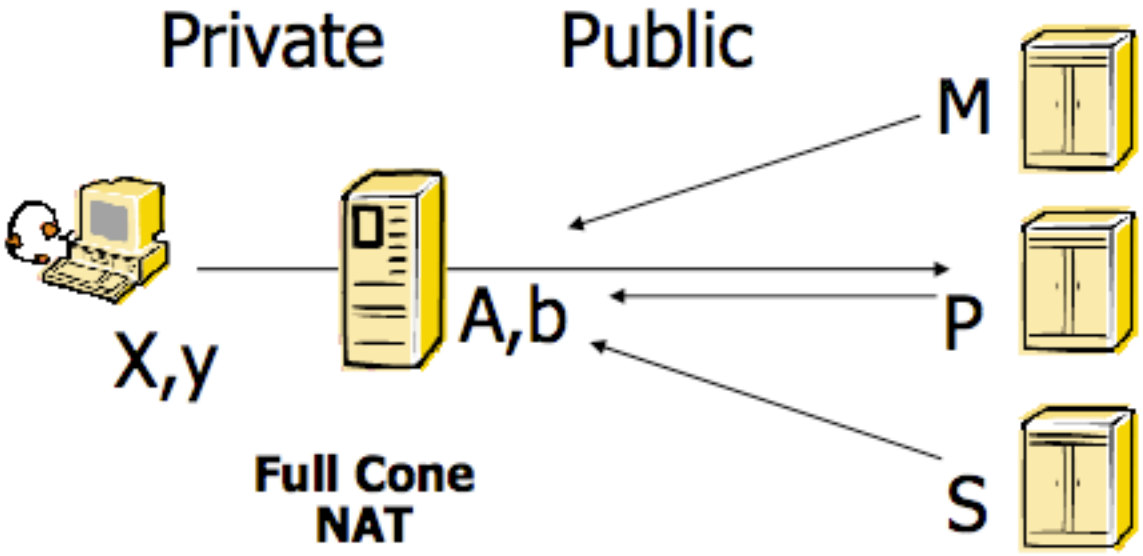
- **Full Cone NAT** is one where all requests from the same internal IP address and port are mapped to the same external IP address and port. Furthermore, any external host can send a packet to the internal host, by sending a packet to the mapped external address.
- **Restricted Cone** is one where all requests from the same internal IP address and port are mapped to the same external IP address and port. Unlike a full cone NAT, an external host (with IP address X) can send a packet to the internal host only if the internal host had previously sent a packet to IP address X.
- **Port Restricted Cone** is like a restricted cone NAT, but the restriction includes port numbers. Specifically, an external host can send a packet, with source IP address X and source port P, to the internal host only if the internal host had previously sent a packet to IP address X and port P.
- **Symmetric Nat** is one where all requests from the same internal IP address and port, to a specific destination IP address and port, are mapped to the same external IP address and port. If the same host sends a packet with the same source address and port, but to a different destination, a different mapping is used. Furthermore, only the external host that receives a packet can send a UDP packet back to the internal host.

You can find out which one you are using by trying a stun client:

SIP in presenza di NAT e firewall

- Non esistono soluzioni universali per risolvere il problema dell'attraversamento dei firewall
  - come minimo, è sempre necessaria la cooperazione del gestore del firewall
- Nell'ottobre 2008 è uscito l'RFC 5389 "Session Traversal Utilities for NAT (STUN)", che rende obsoleto il precedente RFC 3489 e che è stato pensato per risolvere il problema dell'attraversamento dei NAT
- Siccome la maggior parte degli UserAgent attuali sono conformi al vecchio RFC, nel seguito si fa riferimento a quello
- IL'RFC 3489 permette a uno UserAgent di comunicare correttamente anche trovandosi su una rete con indirizzi privati, purché
  - il NAT sia del tipo detto "a cono pieno" (full cone NAT) (cioè NON SIMMETRICO)
  - il Client si attenga a procedure ben precise

NAT a “cono pieno”



- Il mapping fra la coppia<indirizzo:porta>privata (X,y) e la pubblica (A,b),è lo stesso per tutti i partner pubblici di (X,y)
- Il NAT passa a (X,y) qualunque pacchetto TCP o UDP destinato a (A,b)
- Una volta quasi tutti i router per uso domestico e quelli usati nei WiFi Hot spot erano a cono pieno, ora non più.
- N.B.: in caso di saturazione delle tabelle di mapping, il comportamento del NAT è imprevedibile e sono possibili malfunzionamenti

Requisiti per il Client SIP

- Uno User Agent SIP a valle di un **NAT a "cono pieno"**, opera correttamente se:
  - è in grado di interrogare un server STUN (RFC 3489) per conoscere il mapping esterno delle porte su cui è in ascolto (anche quelle per RTP)
  - usa il parametro "**rport**" nell'*Header* "**Via**:", come da RFC 3581
  - usa come "source port" per i flussi RTP da lui prodotti le stesse porte su cui intende ricevere i flussi del partner
  - completata la negoziazione SDP, emette costantemente i flussi RTP
  - se vuole ricevere *Request*, mantiene attivo sul NAT il mapping della porta su cui il suo UAS sta in ascolto, inviando al Registrar e/o al Proxy messaggi di "keepalive" (tipicamente uno ogni 20 secondi)
- Anche in presenza di tutti i requisiti, non è garantita la consegna del flusso RTCP, inviato alla porta RTP+1

Attraversamento NAT: STUN e rport (1 di 2)

```
INVITE sip:662127@bh3.iit.cnr.it SIP/2.0
Via: SIP/2.0/UDP 192.168.1.64:7498;branch=z9hG4bK-d87543-7072ef1c0daa6b0d-1--
    d87543-;rport
Max-Forwards: 70
Contact: <sip:3827@84.220.118.200:12345>
To: "662127"<sip:662127@bh3.iit.cnr.it>
From: "Marco Sommani"<sip:3827@bh3.iit.cnr.it>;tag=64fb0350
Call-ID: MmFjMmEzNTRmYWYxMWZiZDE3ODM5ZTc0OGFhOTA3MTU.
CSeq: 2 INVITE
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY, MESSAGE, SUBSCRIBE, INFO
Content-Type: application/sdp
User-Agent: eyeBeamrelease 1010i stamp 39384
Authorization: Digestusername="3827",realm="bh3.iit.cnr.it",nonce="4617....
Content-Length: 394

v=0
o=- 62 IN IP4 84.220.118.200
s=CounterPatheyeBeam 1.5
c=IN IP4 84.220.118.200
t=0 0
m=audio 35590 RTP/AVP 100 106 60 105 8 18 35 101
a=x-rtp-session-id:20EA85E324FA73EF6C6D97075C735303
a=fmtp:18 annexb=yes
a=fmtp:101 0-15
a=rtpmap:100 SPEEX/16000
a=rtpmap:106 SPEEX-FEC/16000
a=rtpmap:105 SPEEX-FEC/8000
a=rtpmap:18 G729/8000
a=rtpmap:101 telephone-event/8000
a=sendrecv
```

Per porta SIP pubblica provocato dal NAT e individuata da STUN

Per porta RTP pubblica provocato dal NAT e individuata da STUN

SDP

Attraversamento NAT: STUN e rport (2 di 2)

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.1.64:7498;received=84.220.118.200;branch=z9hG4bK-
    d87543-7072ef1c0daa6b0d-1--d87543-;rport=12345
Record-Route: <sip:146.48.98.112;ftag=64fb0350;lr=on>
From: "Marco Sommani"<sip:3827@bh3.iit.cnr.it>;tag=64fb0350
To: "662127"<sip:662127@bh3.iit.cnr.it>;tag=as34a82cc4
Call-ID: MmFjMmEzNTRmYWYxMWZiZDE3ODM5ZTc0OGFhOTA3MTU.
CSeq: 2 INVITE
User-Agent: Asterisk PBX
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY
Contact: <sip:662127@146.48.96.150>
Content-Type: application/sdp
Content-Length: 263

(v): 0
(o): root 4722 4723 IN IP4 146.48.96.150
(s): session
(c): IN IP4 146.48.96.150
(t): 00
(m): audio 18834 RTP/AVP 803 101
(a): rtpmap:8 PCMA/8000
(a): rtpmap:0 PCMU/8000
(a): rtpmap:3 GSM/8000
(a): rtpmap:101 telephone-event/8000
(a): fmtp:101 0-16
(a): silenceSupp:off - - - -
```

SDP

SIP URI e utente umano

- Spesso l'utente umano non fornisce una *Request URI*, ma solo un identificatore (numero di telefono, username ...)
- In questi casi l'UAC costruisce una URI appendendo all'identificatore fornito dall'utente "@" seguito da
  - un "sip domain name", se configurato nel client
  - altrimenti l'indirizzo e la porta del Proxy
  - es.: l'utente digita "662127", e l'UAC crea la *Request URI*:
    - sip:662127@iit.cnr.it (sip domain name)
    - sip:662127@bh3.iit.cnr.it:5060 (indirizzo e porta del Proxy)
- La *Request URI* deve essere fornita dall'utente stesso quando il Client lavora in modalità "SIP Direct" (senza Outbound Proxy)

A cosa serve ENUM

- ENUM (rfc3761) stabilisce come usare il DNS per costruire degli Universal ResourceIdentifiers (URI, rfc3986) partendo da numeri E.164
- Esempio
  - partire dal numero +390123456789 (Application Unique String o AUS)
  - chiedere al DNS i record NAPTR relativi al FQDN (Fully Qualified Domain Name) 9.8.7.6.5.4.3.2.1.0.9.3.e164.arpa
  - costruire gli URI scegliendo fra i NAPTR ricevuti il migliore fra quelli relativi alla applicazione ENUM e che corrispondono alla AUS ed al servizio desiderato (sip, h323, http, mail, address,....)

Record NAPTR (rfc3403)

- A destra della keyword NAPTR ci sono 6 campi:
  - order preference flags services regexp replacement
  - I primi due sono numeri, gli altri stringhe racchiuse fra “”
  - regexp e replacement sono mutuamente esclusivi
  - Al posto di un campo stringa assente ci vuole un punto
- Nell’applicazione ENUM
  - i services, se presenti, hanno la forma “E2U+servizi”
  - i flags, se presenti, hanno il valore “u”
- Negli usi pratici di ENUM, flags, services e regexp sono presenti e replacement è assente

Uso di regexp con flags="u"

- Il campo regexp è composto da due sottocampi ("ere" e "repl") delimitati dallo stesso carattere (generalmente "!") e opzionalmente seguiti da "i"
- La regexp, applicata al numero E.164 originario (AUS o Application Unique String), fornisce l'URI
- Esempio:
  - da +390503158315
  - e !^\+39(0(50)315)([238].*)$!sip:\3@area\2.cnr.it!
  - si ottiene l'URI "sip:8315@area50.cnr.it"
- Un eventuale carattere "i" dopo il terzo delimitatore indica che la regular expression può "matchare" la AUS in modalità case insensitive

Esempi di record NAPTR (1 di 2)

```
;; QUESTION SECTION:
;5.1.8.3.5.1.3.0.5.0.9.3.e164.org. IN   NAPTR

;; ANSWER SECTION:
5.1.8.3.5.1.3.0.5.0.9.3.e164.org. 403 IN NAPTR  100 10 "u" "E2U+ADDRESS"
   "!^.*$!ADDRESS:CN=MarcoSommani\;STREET=Via Giuseppe Moruzzi
   1\;L=Pisa\;ST=Pisa\;C=Italy!" .
5.1.8.3.5.1.3.0.5.0.9.3.e164.org. 403 IN NAPTR  100 10 "u" "E2U+SIP"
   "!^\\+390503153815$!sip:3815@voipgw1.iit.cnr.it!" .
5.1.8.3.5.1.3.0.5.0.9.3.e164.org. 403 IN NAPTR  100 10 "u" "E2U+EMAIL"
   "!^.*$!mailto:marco.sommani@iit.cnr.it!" .
```

```
;; QUESTION SECTION:
;1.0.0.0.0.0.0.5.9.6.9.3.e164.namex.it. IN NAPTR

;; ANSWER SECTION:
1.0.0.0.0.0.0.5.9.6.9.3.e164.namex.it. 86400 IN NAPTR 10 10 "u" ”E2U+sip"
   "!^.*$!sip:389001@bh3.iit.cnr.it!" .
```

Esempi di record NAPTR (2 di 2)

```
;; QUESTION SECTION:
;0.0.3.6.8.4.4.4.6.0.9.3.e164.namex.it. IN NAPTR

;; ANSWER SECTION:
0.0.3.6.8.4.4.4.6.0.9.3.e164.namex.it. 86400 IN       NAPTR 10 10 "u" "E2U+SIP"
     "!^\\+390644486(.*)$!sip:\\1@caspur.it!" .

;; QUESTION SECTION:
;1.0.5.1.8.6.2.4.4.1.4.e164.arpa. IN       NAPTR

;; ANSWER SECTION:
1.0.5.1.8.6.2.4.4.1.4.e164.arpa. 600 IN    NAPTR      100 10 "u" "E2U+sip"
     "!^.*$!sip:x501@switch.ch!" .
1.0.5.1.8.6.2.4.4.1.4.e164.arpa. 600 IN    NAPTR      100 30 "u" "E2U+h323" "!^.*$!h323:501@switch-
     gk.switch.ch!" .

;; QUESTION SECTION:
;4.3.2.1.256.freenum.org.       IN         NAPTR

;; ANSWER SECTION:
4.3.2.1.256.freenum.org. 60      IN        NAPTR      100 10 "u" "E2U+sip"
     "!^\\+*([^\\*]*).*$!sip:\\1@204.91.156.10!" .
4.3.2.1.256.freenum.org. 60      IN        NAPTR      100 10 "u" "E2U+iax2"
     "!^\\+*([^\\*]*)!iax2:@204.91.156.10/\\1!" .
4.3.2.1.256.freenum.org. 60      IN        NAPTR      100 10 "u" "E2U+web:http:"
     "!^.*$!http://www.loligo.com/!" .
```

Funzioni di un ENUM resolver

- Generalmente un ENUM resolver è una procedura che
  - riceve in input un numero E.164, il nome del servizio desiderato e il nome del sottoalbero DNS in cui fare la ricerca
  - restituisce, se esiste, l'URI per il servizio desiderato
- ENUM resolvers sono presenti nei server VoIP più evoluti (SER, OpenSer, Asterisk, GnuGK,...)
- ENUM resolvers possono essere inclusi anche negli apparati VoIP terminali

ISN: una numerazione alternativa

- Gli Internet SubscriberNumbers (ISN, http://www.freenum.org) sono una numerazione controllata da IANA anziché dalla ITU
- Hanno la forma “mmmmmm*nnnn”. Il numero a destra del “*” si chiama Internet Telephony Administrative Domain (ITAD)
- Gli ITAD usabili per ISN sono numeri compresi fra 256 e 2^{16}-1. Sono assegnati da IANA a chi ne fa richiesta. Al GARR è stato assegnato l’ITAD 400
- Le ricerche ENUM relative a numeri ISN si fanno cercando il numero mmmmmm sull’albero nnnn.freenum.org
- Esempio. Per cercare gli URI relativi al numero 1234*256 si richiedono i NAPTR di 4.3.2.1.256.freenum.org

L’albero nrenum.net

- Per sviluppare progetti coordinati facenti uso della tecnologia ENUM senza attendere la piena funzionalità del goldentree, le National Research and EducationNetworks (NREN) hanno attivato l’albero nrenum.net
- L’intendimento è che nrenum.net sia consultato solo se fallisce la consultazione di e164.arpa
- LaName Server primario di nrenum.net è gestito da Switch (NREN svizzera), su incarico di TERENA
- Le radici dei sottoalberi nazionali sono delegate alle NREN
- Il sito del progetto è http://www.nrenum.net/
- Il GARR (NREN italiana) controlla il sottoalbero 9.3.nrenum.net