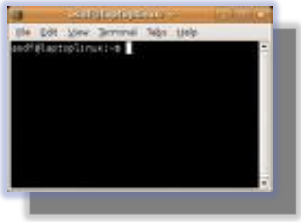
SER - Strumenti di management – parte 1

- Serctl - ***/usr/local/sbin/serctl***
  - Utility di gestione command-line, script shell
  - Comunica con SER tramite FIFO e con il database tramite il client mysql di sistema
  - Gestisce: Subscribers, Access Control Lists, Usrloc, Domains, Control & Diagnostic



- SERWeb - ***http://www.iptel.org/serweb***
  - Linguaggio PHP, modulare e scalabile
  - Comunica con SER tramite XML-RPC o FIFO
  - Gestisce il multi-dominio
  - Sezioni: Administrator e User

- Interfaccia Web custom - *Gestione utenti VoIP - Area della Ricerca Pisa*
  - Progettata per essere usata anche da amministratori/utenti che non hanno conoscenze sul VoIP e adattata alle esigenze dell'Area
  - Interagisce direttamente con il database mysql del SER
  - Scritta in Java Servlet/JSP - HTML/Ajax

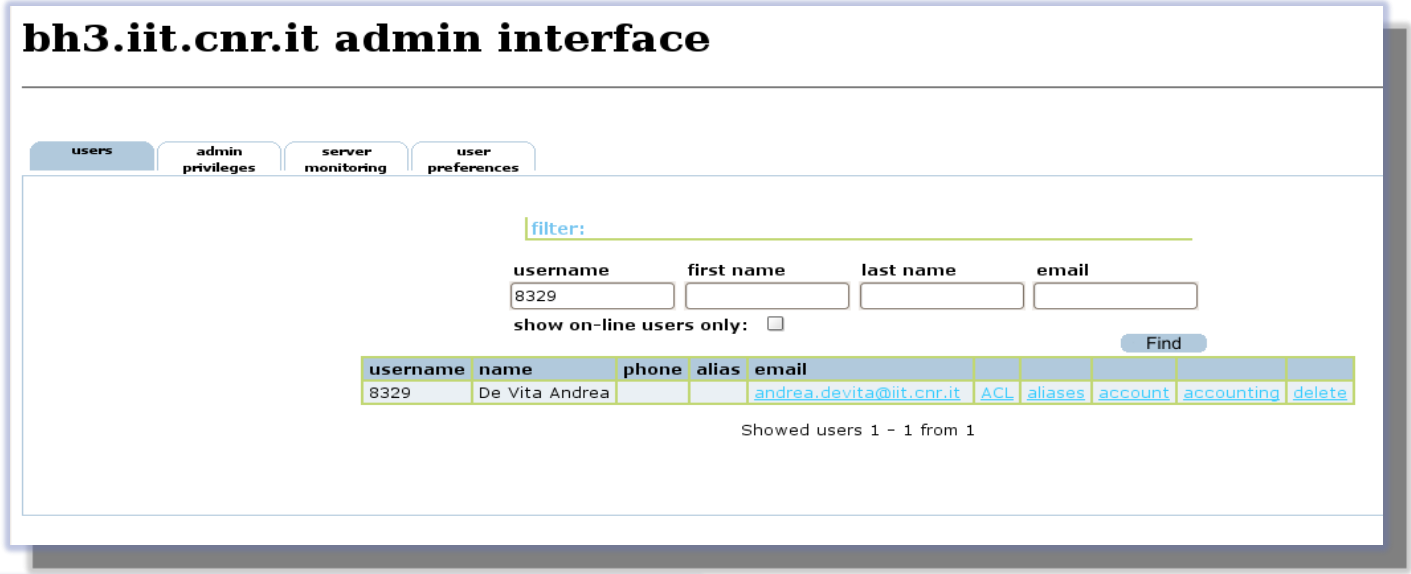
SER - Strumenti di management – parte 2

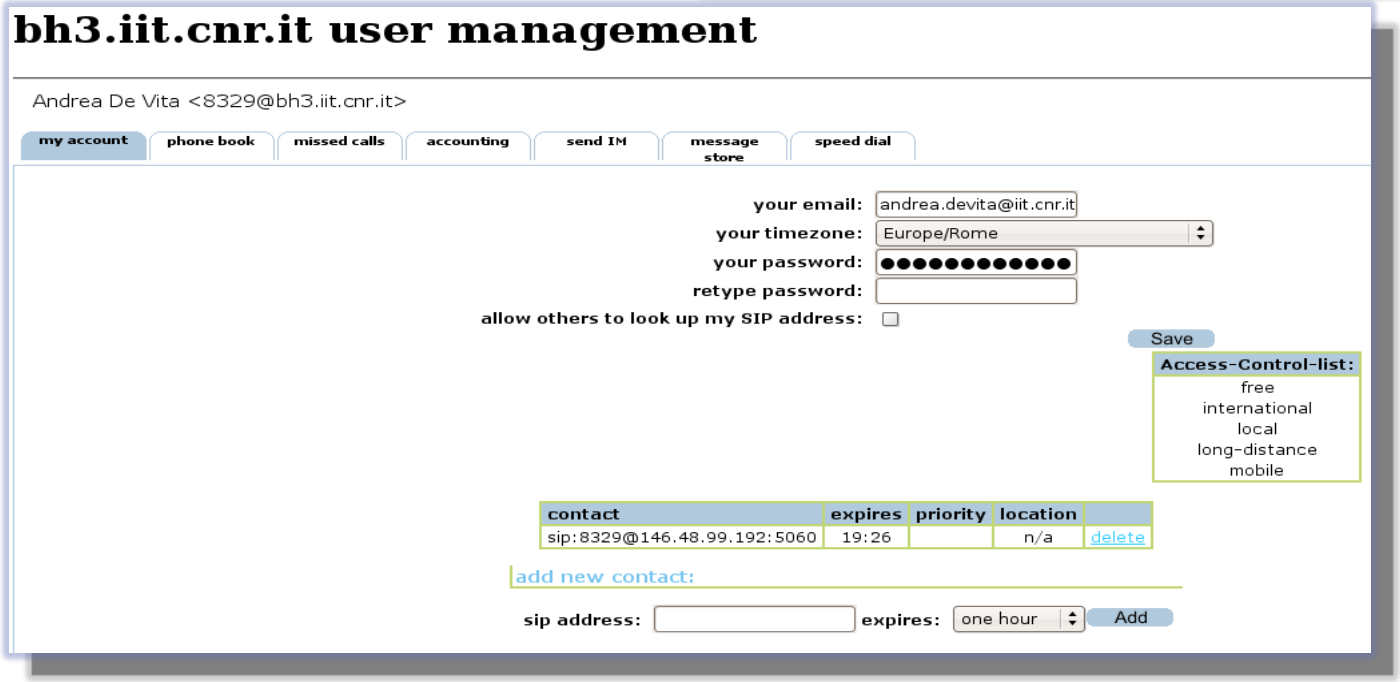
- Serctl - Screenshot

```
 serctl
/usr/local/sbin/serctl $Revision: 1.83.2.2 $
parameter usage:
            * subscribers *
 add <username> <password> <email> .. add a new subscriber (*)
 passwd <username> <passwd> ......... change user's password (*)
 rm <username> ....................... delete a user (*)
...
            * access control lists *
 acl show [<username>] ............... show user membership
 acl grant <username> <group> ....... grant user membership (*)
 acl revoke <username> [<group>] .... grant user membership(s) (*)
...
            * usrloc *
 ul show [<username>]................. show in-RAM online users
 ul rm <username> .................... delete user's UsrLoc entries
 ul add <username> <uri> ............. introduce a permanent UrLoc entry
 showdb [<username>] ................. show online users flushed in DB
...
            * control and diagnostics *
 moni ... show internal status      start .... start ser
 ps ..... show runnig processes     stop ..... stop ser
```

SER - Strumenti di management – parte 3

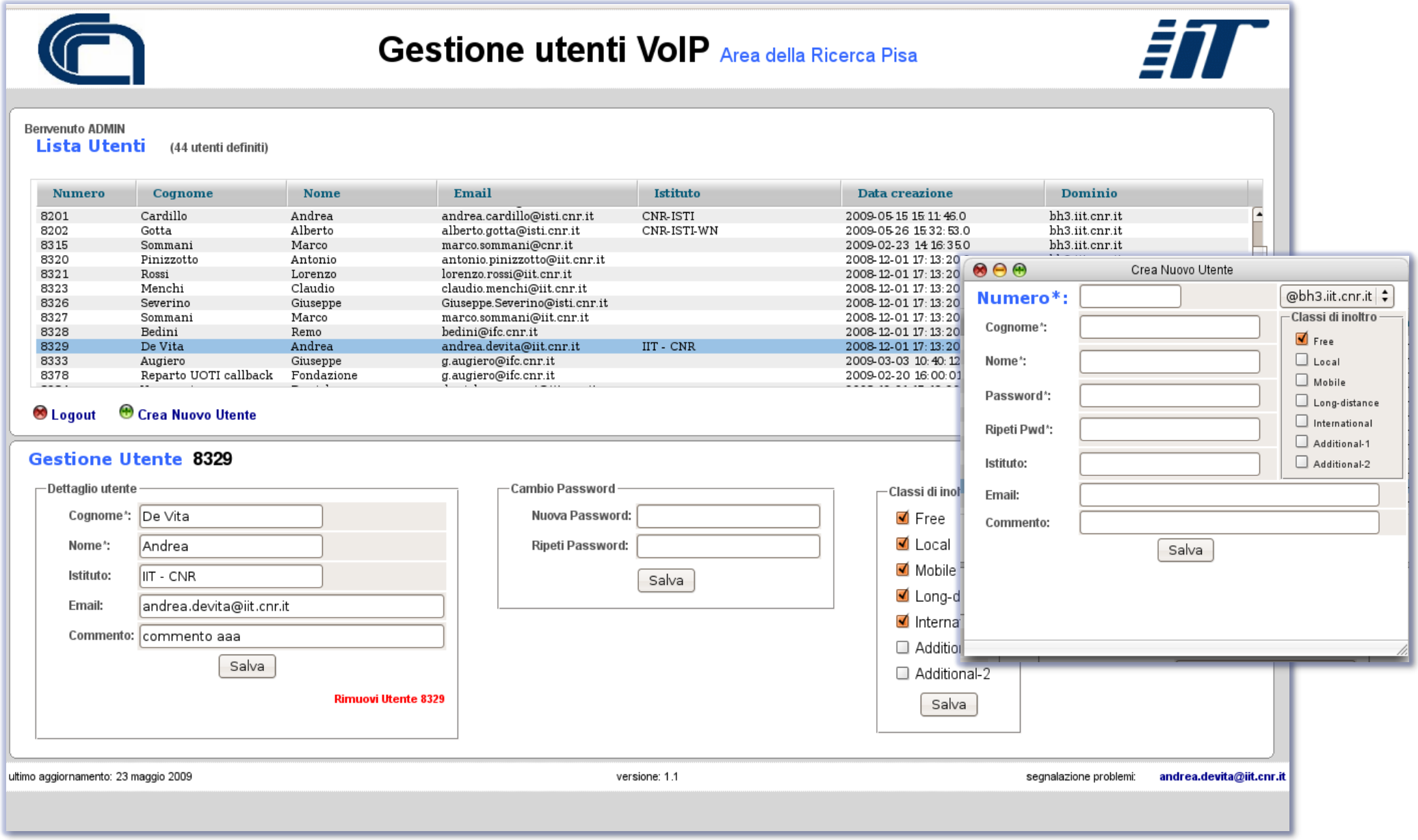
- SERWeb - Screenshot
  - SERWeb Admin
  - SERWeb User





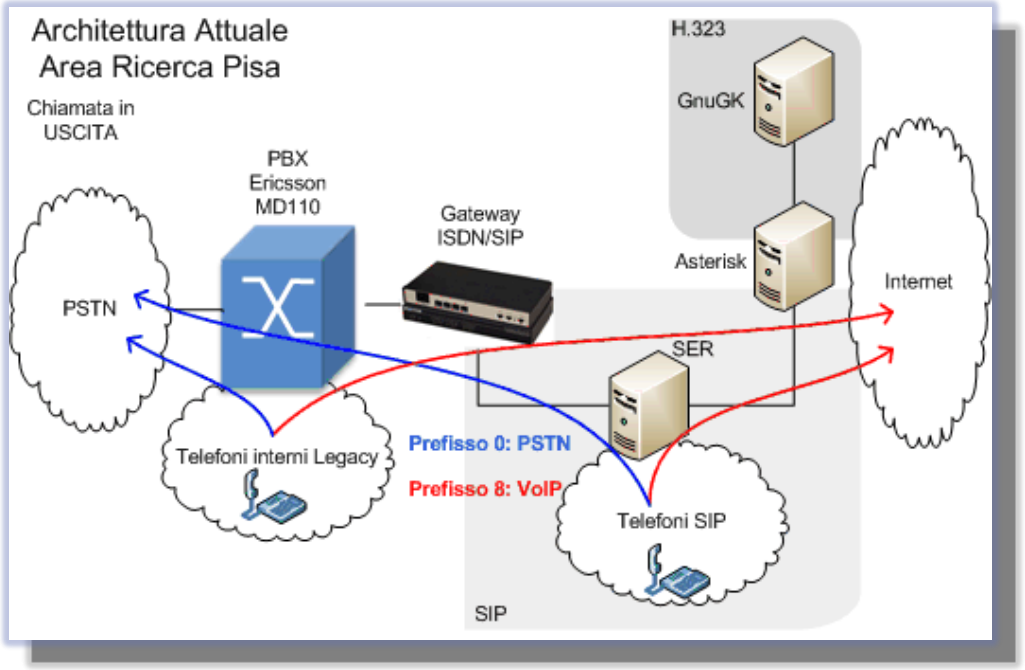
SER - Strumenti di management – parte 4

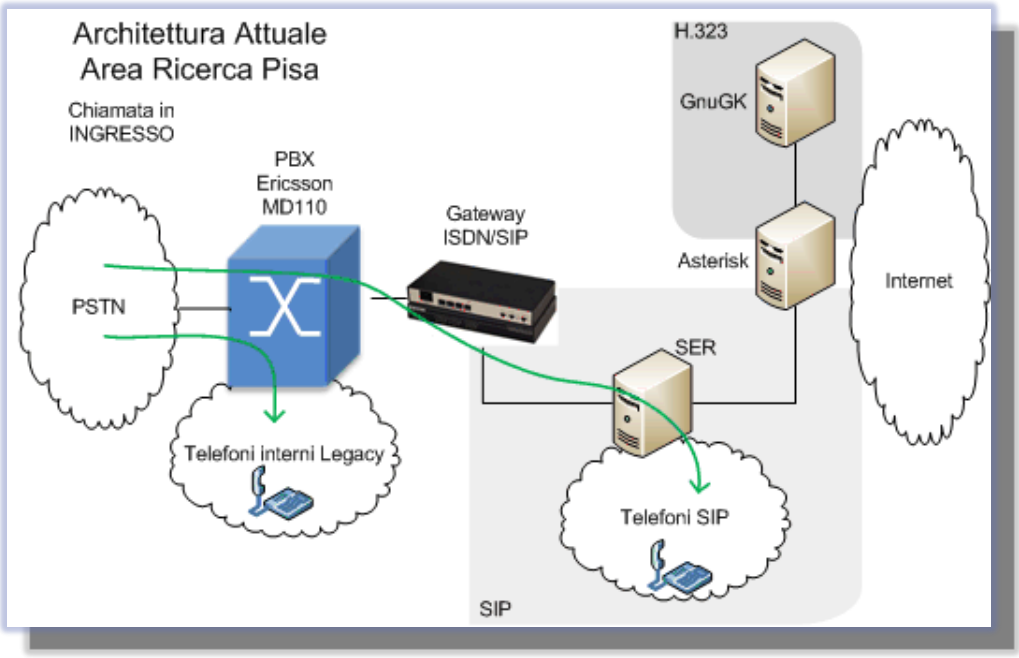
- Gestione utenti VoIP – Area della ricerca Pisa - Screenshot



Instradamento della chiamata – parte 1

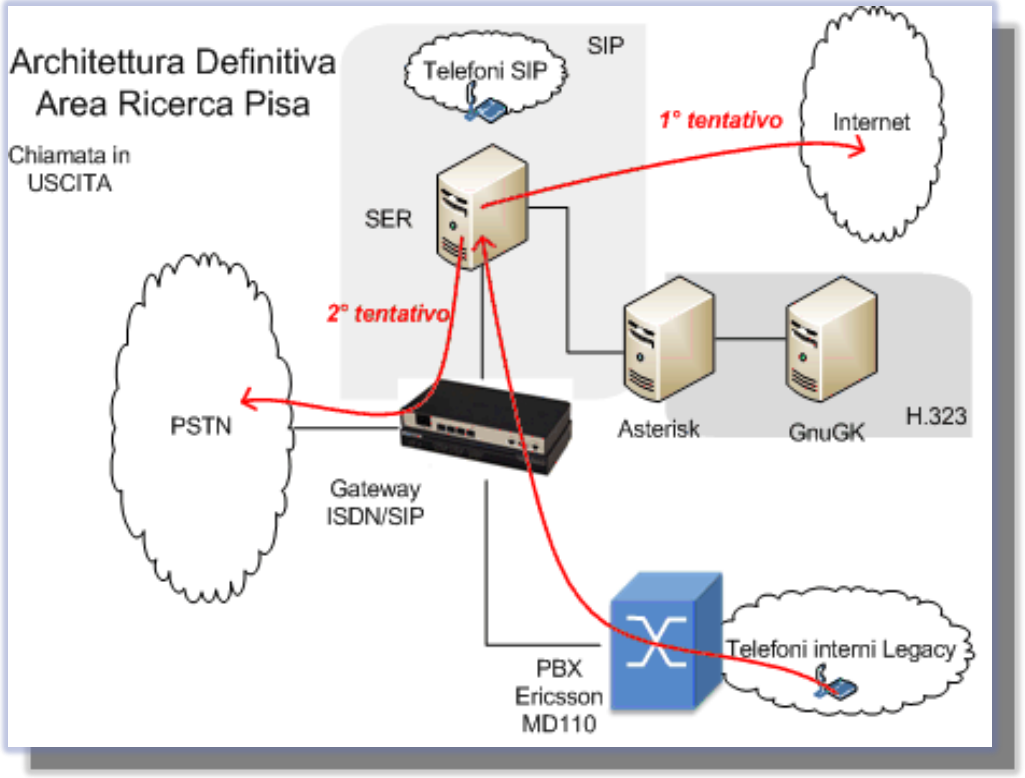
- Infrastruttura attuale Area della Ricerca di Pisa CNR
  - Schema di indirizzamento VoIP : numerico (050 315 8329 → 8329@mydomain)
  - Posizionamento del SER all'interno dell'infrastruttura a valle del PBX Legacy
  - Chiamate in ***uscita***: prefissi di preselezione sia dai telefoni legacy che dai telefoni VoIP
  - Chiamate in ***ingresso***: sottonumerazioni definite staticamente sul PBX legacy
    2XXX e 3XXX: Legacy
    8XXX: VoIP

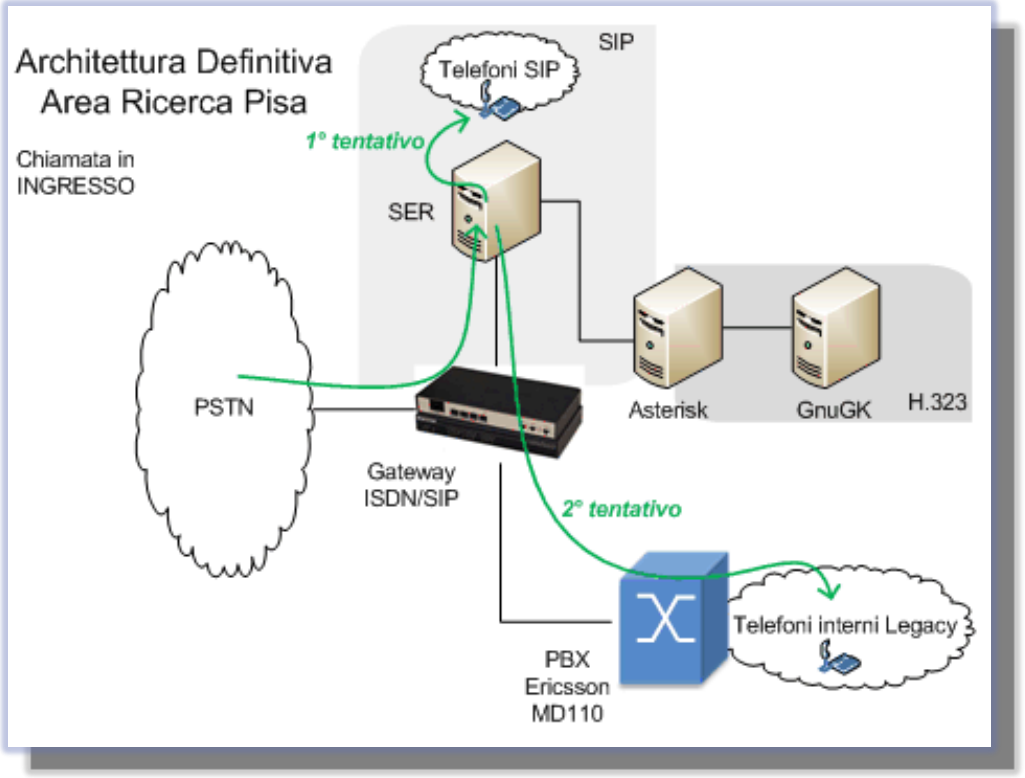




Instradamento della chiamata – parte 2

- Infrastruttura definitiva Area della Ricerca di Pisa CNR
  - Schema di indirizzamento VoIP : numerico (050 315 8329 → 8329@mydomain)
  - Posizionamento del SER all'interno dell'infrastruttura a monte del PBX Legacy
  - Chiamate in ***uscita***: abolizione dei prefissi di preselezione. Primo tentativo in VoIP con fallback su PSTN
  - Chiamate in ***ingresso***: un interno legacy può essere duplicato in VoIP. Si possono gestire facilmente Fork e trasferimenti





Andrea.DeVita@iit.cnr.it

Instradamento della chiamata – parte 3

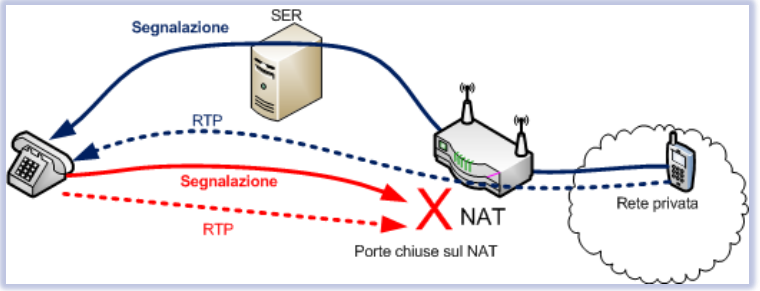
- Next-hop per le chiamate in uscita:
  - NAPTR+SRV lookup (RFC3263): ***username@otherDomain***
  - Enum lookup: ***+390503158329@mydomain***
    - Alberi e164.arpa, nrenum.net, e164.org
    - Chiave di ricerca: 9.2.3.8.5.1.3.0.5.0.9.3.nrenum.net
  - ISN (ITAD Subscriber Numbers) http://www.freenum.org/: ***1234*256@mydomain***
    - Chiede il record NAPTR per 4.3.2.1.256.freenum.org
- Per le chiamate in ingresso:
  - Location Lookup: ***username@mydomain***
    - Controlla se l'utente è attualmente registrato
  - Instradamento statico:
    - Effettutato per alcuni range di numeri definiti su altri server VoIP
  - SIP.edu: ***andrea.devita@iit.cnr.it*** → *8329@iit.cnr.it*
    - *http://www.internet2.edu/sip.edu/*
    - Consiste nel mappare indirizzi email in numeri telefonici

SER & NAT – parte 1

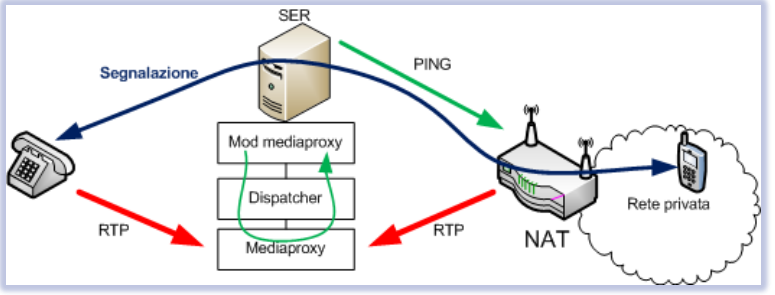
- Problematiche generali
  - L'uso di NAT nelle reti locali introduce gravi problemi nell'istraurazione delle chiamate SIP.
  - I pacchetti SIP contengono informazioni (indirizzi IP e Porte), negli Header e nell'SDP, essenziali per una corretta instaurazione della chiamata (segnalazione e media).
  - In generale uno User Agent dietro NAT non sa come si presenterà all'esterno in termini di IP pubblici e Porte
    - I client con funzionalità STUN non risentono di questo problema
  - I NAT traslano indirizzi IP privati in pubblici e viceversa lasciando inalterato il contenunto del pacchetto stesso. É evidente che i messaggi SIP generati dal client nattato conterranno alcune informazioni che su reti pubbliche sono prive di significato (es. indirizzo IP privato del client)
  - Tuttavia esistono NAT evoluti in grado di applicare le dovute modifiche all'interno dei messaggi SIP

SER & NAT – parte 2

- Sintomi
  - In presenza di User Agent dietro NAT si possono verificare questi inconvenienti:
    - Non si stabilisce la segnalazione (porte chiuse sul NAT o IP privato nella location del SER)
    - La chiamata si stabilisce ma il flusso audio è monodirezionale o totalmente assente



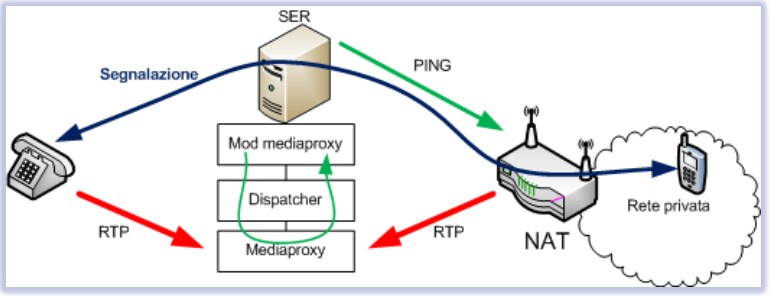
- Soluzioni
  - Il SER gestisce il problema introdotto dai NAT attraverso i moduli:
    - Nathelper, per scoprire se il client è nattato e per effettuare operazioni di riscrittura IP e porte
    - Rtpproxy o in alternativa Mediaproxy, per far transitare i pacchetti RTP (media) attraverso un proxy



Andrea.DeVita@iit.cnr.it

SER & NAT – parte 3

- Soluzione Nathelper/Mediaproxy
  - La soluzione Mediaproxy prevede oltre che l'omonimo modulo (distribuito con SER) un **mediaproxy server** prodotto da AG Projects che si occuperà di ruotare i flussi rtp.
- Il mediaproxy server è composto da:
  - ***Mediaproxy Dispatcher*** *(non obbligatorio)*, viene contattato dal modulo mediaproxy, determina quale relay usare ed inoltra la richiesta.

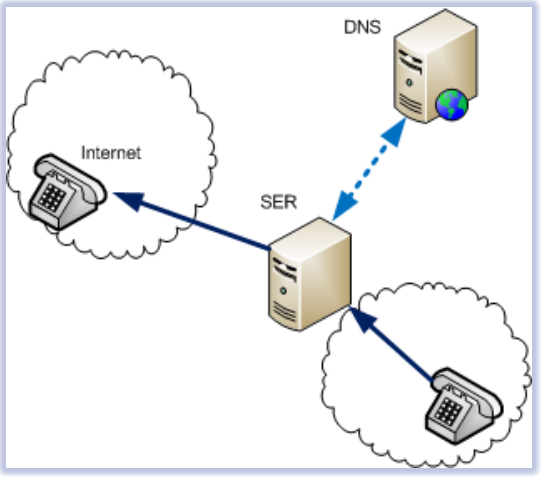


  - ***Mediaproxy Relay***, alloca una coppia di socket per ogni sessione (RTP e RTCP) e ritorna l'IP e le porte al modulo mediaproxy in modo che possano essere riscritti negli appositi campi dell'SDP
- Alcuni vantaggi:
  - Installazione componenti mediaproxy relay su più server (load balancing e ridondanza)
  - Monitoraggio dello stato del mediaproxy tramite script e interfaccia web
  - Accounting

Sessione Pratica

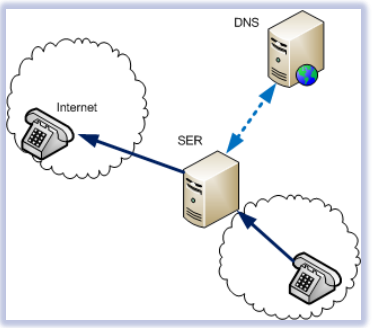
Enum lookup Esempio – parte 1

- Caricamento modulo
  - loadmodule "/usr/local/lib/ser/modules/enum.so"
- Configurazione modulo
  - modparam("enum", "domain_suffix", "nrenum.net.")
- Funzioni esportate
  - enum_query()
    enum_query("suffix")
    enum_query("suffix", "service")



Enum lookup Esempio – parte 2

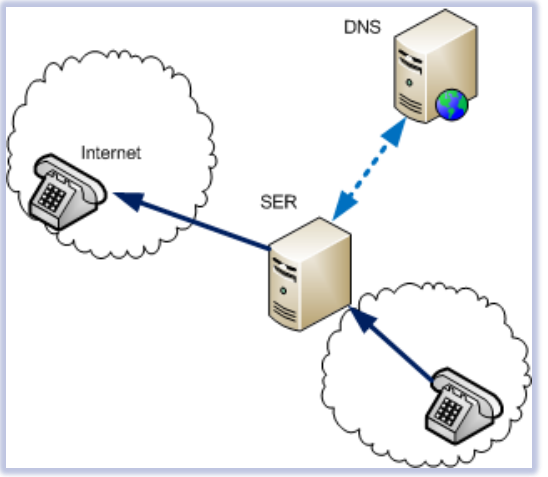
- Alberi consultati:
  - e164.arpa, nrenum.net, e164.org
- Numero chiamato e chiave di ricerca:
  - +390503158329
  - 9.2.3.8.5.1.3.0.5.0.9.3.nrenum.net
- Recor NAPTR ottenuto:
  - 9.2.3.8.5.1.3.0.5.0.9.3.nrenum.net. 3595 IN NAPTR 10 10 "u" "E2U+sip" "***!^\\+390503158(.*)$!sip:8\\1@bh3.iit.cnr.it!***" .




```
Route[7] {
# accertarsi che la parte user della request-uri contenga un numero nel
# formato E.164 (es. +390503158329) altrimenti fare le opportune modifiche
        if (    (enum_query("e164.arpa."))   ||
                (enum_query("nrenum.net."))  ||
                (enum_query("e164.org."))  )
                {
                        route(1);
                        break;
        };
        else { # provo con PSTN? }
}
```

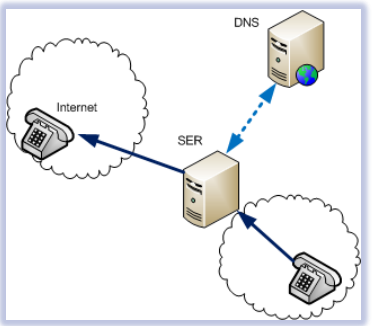
ISN lookup Esempio – parte 1

- ITAD Subscriber Numbers http://www.freenum.org/
  - Formato
    - numero*ITAD
  - ITAD associato al Garr
    - 400
  - Non esiste un modulo specifico per effettuare ISN lookup. Occorre utilizzare uno script esterno che:
    - Compone la chiave di ricerca
    - Effettua la query al DNS e se ottiene il record NAPTR, applica la regex e ritorna la nuova URI
  - Sarà la funzione **`exec_dset()`** ad eseguire lo script ed effettuare la riscrittura della request URI
  - Caricamento modulo
    - loadmodule "/usr/local/lib/ser/modules/exec.so"
  - Funzioni esportate
    - exec_dset("/usr/local/sbin/isn-resolv.pl")



ISN lookup Esempio – parte 2

- Albero consultati:
  - freenum.org
- Numero chiamato e chiave di ricerca:
  - +390503158329*400
  - 9.2.3.8.5.1.3.0.5.0.9.3.400.freenum.org
- Recor NAPTR ottenuto:
  - 9.2.3.8.5.1.3.0.5.0.9.3.400.freenum.org. 3600 IN NAPTR 100 10 "u" "E2U+sip" "***!^\\+*([^*]*)!sip:\\1@sipsrv1.dir.garr.it!***" .
- Lo script isn-resolv.pl si può scaricare dal seguente url
  - http://www.freenum.org/cookbook/#external-isn-resolver-for-ser




```
route[8] {
        #-----------------------------------------------------
        #       ISN Handler
        #-----------------------------------------------------
        if (!exec_dset("/usr/local/sbin/isn-resolv.pl")) {
                sl_send_reply("404", "Not Found");
                break;
        } else {
                route(1);
                break;
        };

}
```

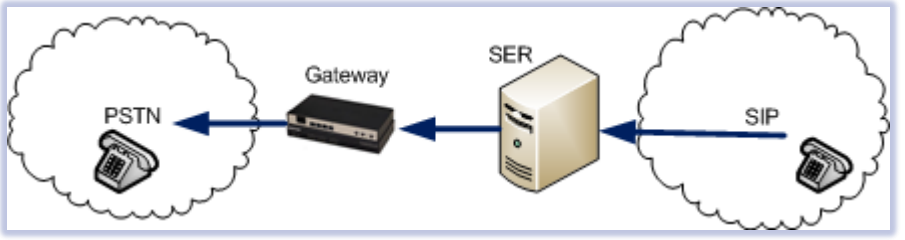
SIP.edu Lookup - http://www.internet2.edu/sip.edu/

- Anche in questo caso bisogna ricorrere alla funzione **exec_dset()** del modulo **exec.so** per eseguire uno script che effettua la riscrittura della request URI. Nel nostro caso lo script interroga una database. La chiave di ricerca è un indirizzo email.
- Caricamento modulo
  - loadmodule "/usr/local/lib/ser/modules/exec.so"
- Funzioni esportate
  - exec_dset("/usr/local/sbin/sipedu-rewrite.sh")
-

```
route[12] {
        #-------------------------------------
        # SIPEDU Handler
        #-------------------------------------
        if (!exec_dset("/usr/local/sbin/sipedu-rewrite.sh")) {
                sl_send_reply("404", "Not Found with SIP.EDU");
                break;
        } else {
                route(1);
                break;
        };
}
```

Instradamento gateway – parte 1

- La presenza di un gateway SIP/PSTN integra totalmente la telefonia VoIP a quella tradizionale
- Il gateway si occuperà di trasformare segnalazione e media fra i due standard
- Le chiamate verso PSTN possono essere instradate tramite un prefisso di preselezione (”0” nel nostro caso) o tramite faultback automatico
- Prima di inoltrare la chiamata sul gateway è buona norma controllare le autorizzazioni associate al chiamante
- L'IP del gateway può essere inserito fra gli agent ”trusted” per evitare che il SER richieda un’autenticazione per ogni chiamata proveniente dalla rete pubblica



Instradamento gateway – parte 2

- Caricamento moduli
  - loadmodule "/usr/local/lib/ser/modules/permissions.so "
  - loadmodule "/usr/local/lib/ser/modules/avpops.so "
  - loadmodule "/usr/local/lib/ser/modules/mysql.so"
- Configurazione
  - modparam("permissions", "db_url","mysql://user:pwd@localhost/ser ")
  - modparam("tm", "fr_inv_timer", 27)
  - modparam("tm", "fr_inv_timer_avp", "inv_timeout ")
  - modparam("permissions", "db_mode", 1)
  - modparam("permissions", "trusted_table", "trusted")

Instradamento gateway – parte 3

- Instradamento verso il gateway con prefisso ”0”
- Riscrittura della request-uri (parte host) per l'inoltro sul gateway
- Innalzamento del timer inv_timeout a 45 secondi
- ```
  ...parti omesse...
  if (uri=~"^sip:0[0-9]*@") {
   route(14);
   break;
  }
  ...parti omesse...
  ```
- ```
  route[14] {
   rewritehost("IP GATEWAY");
      avp_write("i:45", "inv_timeout");
      route(1);
  }
  ```

Trasferimenti e voicemail – parte 1

- Sui telefoni definiti sul SER è possibile applicare i seguenti trasferiementi:
  - ***Blind transfer***
    - Tutte le chiamate in ingresso sul telefono verranno dirottate incondizionatamente sulla destinazione del trasferimento
  - ***Forward On Busy***
    - Se un telefono risponde con ”486” (Busy here), il SER dirotta la chiamata sulla destinazione del trasferimento (es. Voicemail)
  - ***Forward No Answerd***
    - Se un telefono risponde con ”408” (Request Timeout), il SER dirotta la chiamata sulla destinazione del trasferimento (es. Voicemail)
- Il primo trasferimento consiste in una semplice riscrittura della request uri mentre gli altri trasferimenti vengono trattati in modo diverso perché il SER deve intercettare i codici di risposta dai client.
- Le informazioni sui trasferimenti vengono memorizzate sul database

Trasferimenti e voicemail – parte 2

- Caricamento moduli
  - loadmodule "/usr/local/lib/ser/modules/avpops.so”
  - loadmodule "/usr/local/lib/ser/modules/mysql.so"
- Configurazione
  - modparam("avpops", "avp_url", "mysql://user:pwd@localhost/ser”)
  - modparam("avpops", "avp_table", "usr_preferences")
- Tabella database ”usr_preferences”
username	attribute	value
8329	callfwd	sip:2125@miodominiovoip.zz
8123	fwdbusy	sip:vm-8123@voicemailserver.zz
8888	fwdnoanswer	sip:vm-8888@voicemailserver.zz

Trasferimenti e voicemail – parte 3

- Blocco di controllo per trasferimento incondizionato (Blind), effettuato prima della lookup in location

```
if (avp_db_load("$ruri/username", "s:callfwd")) {
        setflag(22);
        avp_pushto("$ruri", "s:callfwd");
        route(6);
        break;
};

if (!lookup("location")) {
        sl_send_reply("404", "User Not Found");
        xlog("L_ERR", "time [%Tf]  404 User Not Found \n");
        break;
};
```

- Blocco di controllo per eventuali trasferimenti (su occupato e su non risposta), da effettuare per gli utenti in location prima di inoltrare la chiamata. In tal caso setto il relativo flag e attivo il **`t_on_failure("1")`**

```
# controllo eventuali trasferimenti attivati
if (avp_db_load("$ruri/username", "s:fwdbusy")) {
        if (!avp_check("s:fwdbusy", "eq/$ruri/i")) {
                setflag(26);
        };
};

# controllo eventuali trasferimenti attivati
if (avp_db_load("$ruri/username", "s:fwdnoanswer")) {
        if (!avp_check("s:fwdnoanswer", "eq/$ruri/i")) {
                setflag(27);
        };
};

t_on_failure("1");
```

Trasferimenti e voicemail – parte 4

- Route di inoltro trasferimenti
- Failure route

```
route[6] {

        ...parte omessa...

        if (!isflagset(22)) {
                append_branch();
        };

        ...parte omessa...

        route(1);
}
```

```
failure_route[1] {
        if (t_check_status("487")) {
                break;
        };

        if (isflagset(26) && t_check_status("486")) {
                if (avp_pushto("$ruri", "s:fwdbusy")) {
                        avp_delete("s:fwdbusy");
                        resetflag(26);
                        route(6);
                        break;
                };
        };

        if (isflagset(27) && t_check_status("408")) {
                if (avp_pushto("$ruri", "s:fwdnoanswer")) {
                        avp_delete("s:fwdnoanswer");
                        resetflag(27);
                        route(6);
                        break;
                };
        };
}
```

Nathelper & Mediaproxy – parte 1

- Installazione mediaproxy:
  - Scaricare da sito http://www.ag-projects.com/content/view/479/1/
  - Scompattare l'archivio in ”/urs/local/mediaproxy/” e rinominare il file `mediaproxy.ini.sample` in `mediaproxy.ini` quindi editarlo
- Caricamento moduli
  - loadmodule "/usr/local/lib/ser/modules/uri_db.so”
  - loadmodule "/usr/local/lib/ser/modules/domain.so”
  - loadmodule "/usr/local/lib/ser/modules/mediaproxy.so”
  - loadmodule "/usr/local/lib/ser/modules/nathelper.so”
  - loadmodule "/usr/local/lib/ser/modules/textops.so"
- Configurazione
  - modparam("nathelper", "rtpproxy_disable", 1)
  - modparam("nathelper", "natping_interval", 0)
  - modparam("mediaproxy","natping_interval", 30)
  - modparam("mediaproxy","mediaproxy_socket", "/var/run/mediaproxy.sock”)
    oppure se si usa il dispatcher
    modparam("mediaproxy","mediaproxy_socket", "/var/run/proxydispatcher. sock”)
  - modparam("registrar", "nat_flag", 6)

Nathelper & Mediaproxy – parte 2

- Main route

```
# ----------------------------------------
# Record Route Section
# ----------------------------------------
if (method=="INVITE" && client_nat_test("3")) {
        record_route_preset("SER IP ADDRESS:5060;nat=yes");
} else if (method!="REGISTER") {
        record_route();
};
# ----------------------------------------
# Call Tear Down Section
# ----------------------------------------
if (method=="BYE") {
        end_media_session();
};
if (method=="CANCEL") {
        end_media_session();
};
```

```
if (loose_route()) {
        if ((method=="INVITE" || method=="REFER") && !has_totag()) {
                sl_send_reply("403", "Forbidden");
                break;
        };

        if (method=="INVITE") {
                if (!allow_trusted()) {
                        if (!proxy_authorize("bh3.iit.cnr.it","subscriber")) {
                                proxy_challenge("bh3.iit.cnr.it","0");
                                break;
                        } else if (!check_from()) {
                                sl_send_reply("403", "Use From=ID");
                                break;
                        };
                        consume_credentials();
                };

                if (client_nat_test("3")||search("^Route:.*;nat=yes")){
                        setflag(6);
                        use_media_proxy();
                };
        };


        route(1);
        break;
};
```

Nathelper & Mediaproxy – parte 3

- Gestione REGISTER e INVITE

```
route[2] {
        # ----------------------------------------------------------
        # REGISTER Message Handler
        # ----------------------------------------------------------
        sl_send_reply("100", "Trying");
        if (!search("^Contact:[ ]*\*") && client_nat_test("7")) {
                setflag(6);
                fix_nated_register();
                force_rport();
        };
        if (!www_authorize("bh3.iit.cnr.it","subscriber")) {
                www_challenge("bh3.iit.cnr.it","0");
                break;
        };
        if (!check_to()) {
                sl_send_reply("401", "Unauthorized");
                break;
        };
        consume_credentials();
        if (!save("location")) {
                sl_reply_error();
        };
}
```

```
route[3] {
# ----------------------------------------------
# INVITE Message Handler - CHECK NATTED CLIENT
# ----------------------------------------------
        if (client_nat_test("3")) {
                setflag(7);
                force_rport();
                fix_nated_contact();
        };
        ...parte omessa...
        if (!lookup("location")) {
                sl_send_reply("404", "User Not Found");
                break;
        };
        route(4);
        route(1);
}
```

Nathelper & Mediaproxy – parte 4

- Gestione NAT traversal, onreply_route e default route

```
route[4] {
# ----------------------------------------
# NAT Traversal Section
# ----------------------------------------
        if (isflagset(6) || isflagset(7)) {
                if (!isflagset(8)) {
                        setflag(8);
                        use_media_proxy();
                };
        };
}
```

```
onreply_route[1] {
        if ((isflagset(6) || isflagset(7)) &&
                (status=~"(180)|(183)|2[0-9][0-9]")) {
                if (!search("^Content-Length:[ ]*0")) {
                        use_media_proxy();
                };
        };
        if (client_nat_test("1")) {
                fix_nated_contact();
        };
}
```

```
route[1] {
        # ----------------------------------------------
        # Default Message Handler
        # ----------------------------------------------
        t_on_reply("1");
        if (!t_relay()) {
                if (method=="INVITE" || method=="ACK") {
                        end_media_session();
                };
                sl_reply_error();
        };
}
```


Andrea.DeVita@iit.cnr.it

Fine

- Grazie per l'attenzione