Fakulteten för ekonomi, kommunikation och IT

Datakommunikation II
SIP

Datum: 2007-10-17

Namn: Henrik Bäck
Mathias Andersson

Kurs: Kurs

Innehållsförteckning

Registrering

Vid registrering skickar klienten ett REGISTER-meddelande till servern som klienten skall registrera sig vid. Meddelandet innehåller adressen till servern, meddelandet innehåller också kontaktinformation för den aktuella klientens användare, exempelvis sip-adress.

Registreringsmeddelandet skickas ut på nätverket efter att ett DNS-uppslag för SIP-serverns namn har gjort.

Meddelandet tas emot på SER-servern och användarens platsinformation registreras. Efter att informationen registrerats skickas en bekräftelse, i form av 200 OK, tillbaka till klienten.

Efter detta är registreringen klar.

Samtal

Vid ett samtal skickar först klienten ett INVITE-meddelande till SER-servern, i detta fall vid *openiplab.net*. Serverb svarar på detta meddelandet med *100 Trying*. SER-servern skickar sedan vidare ett INVITE-meddelande, i detta fall till mottagaren. Mottagaren skickar, efter att ha tagit emot INVITE-meddelandet, tillbaka ett meddelande med *180 Ringing* till SER-servern. Härpå skickar SER-servern detta meddelande vidare till klienten.

När mottagaren svarar så skickas ett meddelande med *200 OK* ut till SER-servern, meddelandet innehåller också IP-dress och annan anslutningsinformation. SER-server skickar sedan vidare denna information till klienten.

När klienten har fått meddelandet med *200 OK* får den också information om var mottagaren befinner sig. Nu kan de båda kommunicera direkt med varandra och här tar ett annat protokoll över för själva innehållet i samtalat.

När samtalet är slut skickar en av deltagarna ett *BYE-meddelande* till den andre. Detta gör att samtalet kopplas ned.

Modifikation av SER

För att implementera den önskade telefonkön användes den befintliga konfigurationsfilen (ser.cfg) som grund. I den fil lades två fall till. Det enda fallet är då numret 020100100 rings och det andra är när 020100200 rings.

Filen ser.cfg finns som bilaga till detta dokument.

020100100 (route 2)

Om telefonnumret 020100100 påträffas vid en INVITE-metod så körs rutin för att hantera köuppbyggnad. Här sparas den uppringandes telefonnummer (URI) i en databasfil.

Därefter skickas meddelandet *600 Busy Everywhere* tillbaka till den uppringande med meddelandet att personen är placerad i kö och att denne kommer att bli återuppringd när en operatör är ledig.

020100200 (route 3)

När telefonnumret 020100200 påträffas vid en INVITE-metod så körs en rutin för att hämta ett nummer från den befintliga kön. Ett nummer hämtas från den sparade databasfilen och därefter vidarebefordras samtalet till detta nummer.

Detta gör att den person som ringer till 020100100, vid den modifierade SER-servern, kommer att bli kopplade till den person som står på tur i kön. Om ingen person finns i kön kommer samtalet att brytas eftersom någon omkoppling inte kan ske.

Bilagor

Bilaga A - ser.cfg

```
#
# $Id: ser.cfg,v 1.21.4.1 2003/11/10 15:35:15 andrei Exp $
#
# simple quick-start config script
#

# ----------- global configuration parameters
------------------------

#debug=3         # debug level (cmd line: -ddddddddddd)
#fork=yes
#log_stderror=no   # (cmd line: -E)

/* Uncomment these lines to enter debugging mode
debug=7
fork=no
log_stderror=yes
*/

check_via=no  # (cmd. line: -v)
dns=no           # (cmd. line: -r)
rev_dns=no      # (cmd. line: -R)
#port=5060
#children=4
fifo="/tmp/ser_fifo"

# ------------------ module loading
----------------------------------

# Uncomment this if you want to use SQL database
#loadmodule "/usr/lib/ser/modules/mysql.so"

loadmodule "/usr/lib/ser/modules/sl.so"
loadmodule "/usr/lib/ser/modules/tm.so"
loadmodule "/usr/lib/ser/modules/rr.so"
loadmodule "/usr/lib/ser/modules/maxfwd.so"
loadmodule "/usr/lib/ser/modules/usrloc.so"
loadmodule "/usr/lib/ser/modules/registrar.so"
loadmodule "/usr/lib/ser/modules/exec.so"

# Uncomment this if you want digest authentication
# mysql.so must be loaded !
#loadmodule "/usr/lib/ser/modules/auth.so"
#loadmodule "/usr/lib/ser/modules/auth_db.so"
```

```
# ----------------- setting module-specific parameters
---------------

# -- usrloc params --

modparam("usrloc", "db_mode",   0)

# Uncomment this if you want to use SQL database
# for persistent storage and comment the previous line
#modparam("usrloc", "db_mode", 2)

# -- auth params --
# Uncomment if you are using auth module
#
#modparam("auth_db", "calculate_ha1", yes)
#
# If you set "calculate_ha1" parameter to yes (which true in this
config),
# uncomment also the following parameter)
#
#modparam("auth_db", "password_column", "password")

# -- rr params --
# add value to ;lr param to make some broken UAs happy
modparam("rr", "enable_full_lr", 1)

# -------------------------  request routing logic
-------------------

# main routing logic

route{

     # initial sanity checks -- messages with
     # max_forwards==0, or excessively long requests
     if (!mf_process_maxfwd_header("10")) {
          sl_send_reply("483","Too Many Hops");
          break;
     };
     if ( msg:len > max_len ) {
          sl_send_reply("513", "Message too big");
          break;
     };

     # we record-route all messages -- to make sure that
     # subsequent messages will go through our proxy; that's
     # particularly good if upstream and downstream entities
     # use different transport protocol
     record_route();
     # loose-route processing
     if (loose_route()) {
          t_relay();
          break;
     };


     if (method=="INVITE" && uri =="sip:020100100@192.168.0.107")
     {
          route(2);
          break;
     }
```

```
     if (method=="INVITE" && uri =="sip:020100200@192.168.0.107")
     {
          route(3);
          break;
     }

     # if the request is for other domain use UsrLoc
     # (in case, it does not work, use the following command
     # with proper names and addresses in it)
     if (uri==myself) {

          if (method=="REGISTER") {

# Uncomment this if you want to use digest authentication
#             if (!www_authorize("iptel.org", "subscriber")) {
#                  www_challenge("iptel.org", "0");
#                  break;
#             };

               save("location");
               break;
          };

          # native SIP destinations are handled using our USRLOC DB
          if (!lookup("location")) {
               sl_send_reply("404", "Not Found");
               break;
          };
     };


     route(1);
}

route[1] {
     # forward to current uri now; use stateful forwarding; that
     # works reliably even if we forward from TCP to UDP
     if (!t_relay()) {
          sl_reply_error();
     };
}

#route to add caller to queue
route[2] {
     sl_send_reply("600", "Du placerad i kö&#65533;och du kommer att
bli uppringd när&#65533;en operatör&#65533;är ledig.");

     exec_msg("perl /home/lab/queue.pl enqueue");

}

#route to conect operator to queue user
route[3] {

     exec_dset("perl /home/lab/queue.pl dequeue");
     if (!t_relay()) {
          sl_reply_error();
     };

}
```

Bilaga B - queue.pl

```
#!/usr/local/bin/perl
#


     $inputFile = "/home/lab/queue.db";         # DBFIle

     if($ARGV[0] eq "enqueue")
     {
          open(QUEUE, ">>$inputFile"); # Open the file

          while (<STDIN>) {
               if (/^From:.*?<(.*?)>/) {
                    print QUEUE "$1\n";
               }
          }
     }
     elsif($ARGV[0] eq "dequeue")
     {
          open(QUEUE, $inputFile);     # Open the file
          @content = <QUEUE>;

          $newQueue = '';
          $i = 1;
          foreach $line (@content)
          {
               if($i)
               {
                    $i = 0;
                    print $line;
               }
               else
               {
                    $newQueue .= $line;
               }

          }
          open(QUEUE, ">$inputFile");
          print QUEUE $newQueue;
     }
```