Universitatea
Transilvania
din Brașov
FACULTATEA DE MATEMATICĂ
ȘI INFORMATICĂ

Lucrare de licență

Autor: **Dan-Ștefan Arnăut**

Coordonator stiințific: **Lect. univ. dr. Vlad Monescu**

Brașov, 2021

Aplicație VoIP pentru apeluri audio în timp real

Autor: **Dan-Ștefan Arnăut**

Coordonator stiințific: **Lect. univ. dr. Vlad Monescu**

Brașov, 2021

Cuprins

Introducere

Odată cu accelerarea fenomenului de globalizare, comunicarea în cadrul companiilor prezintă noi provocări, iar infrastructura companiilor trebuie să țină pasul cu acestea. Nevoia de soluții care să permită comunicarea în timp real între oameni este în continuă creștere, iar contextul pandemic actual nu a făcut decât să accentueze aceste nevoi și să accelereze dezvoltarea a cât mai multor soluții.

Lucrarea de față își propune să pună la dispoziția utilizatorilor printr-o interfată simplă, posibilitatea de a realiza apeluri audio în timp real între doi participanți prin intermediul tehnologiei VoIP.

Tehnologia VoIP (Voice over Internet Protocol) oferă utilizatorilor aceleași servicii de comunicare (apeluri audio/video, SMS, mesaje vocale, etc.) precum o rețea clasică PSTN (Public Switch Telephony Network) însă, aceste servicii sunt asigurate prin intermediul internetului. Protocoalele standard care stau la baza acestei tehnologii sunt: SIP (Session Initiation Protocol), SDP (Session Description Protocol) și RTP (Real-Time Transport Protocol).

Această aplicație desktop pentru Windows este dezvoltată în limbajul C#.

Aplicația de față este alcătuită din două componente:

1. Componenta server – această componentă are rolul de a asigura conexiunea și transmiterea de mesaje între doi clienți
2. Componenta client – această componentă are rolul de a pune la dispoziția utilizatorului funcționalitățile necesare pentru realizarea unui apel audio (înregistrare audio și trimitere prin rețea a informațiilor, redare a informației audio venită din rețea, interfață grafică)

Pentru extragerea, manipularea și redarea informației audio am apelat la ajutorul librăriei externe "NAudio". Această librărie .NET este una open-source și a fost scrisă de către

Mark Heath la a cărei dezvoltare ulterioară au contribuit mulți alți dezvoltatori software. Această librărie se află sub licența Microsoft Public License (Ms-PL) ceea ce înseamnă că poate fi utilizată atât în scop personal cât și în dezvoltarea de aplicații comerciale.

Deși există numeroase librării open-source disponibile online care implementează protocoalele SIP, RTP, SDP cum ar fi PJSIP, Sippet, sipsorcery, etc. , pentru această aplicație am ales să dezvolt propria librărie care să implementeze aceste protocoale. Această librărie este folosită atât în componenta de client cât și în cea de server pentru crearea și parsarea mesajelor.

Validarea corectitudinii mesajelor produse de această aplicație cât și a flow-urilor apelurilor a fost realizată utilizând disectorii built-in din cadrul Wireshark.

Wireshark este un tool open-source de analiză a datelor dintr-o rețea și reprezintă standardul în materie de tool-uri folosite în domeniul protocoalelor și/sau al rețelelor.

Protocoalele care stau la baza acestei aplicații sunt:

- UDP
- SIP
- SDP
- RTP

Rolul și formatul acestora va fi descris în capitolele următoare.

Scenariile pe care această aplicație le suportă sunt:

1. Înregistrarea unui client la server
2. Realizarea unui apel
3. Respingerea unui apel
4. Renunțarea la apel
5. Apelarea unui utilizator neînregistrat

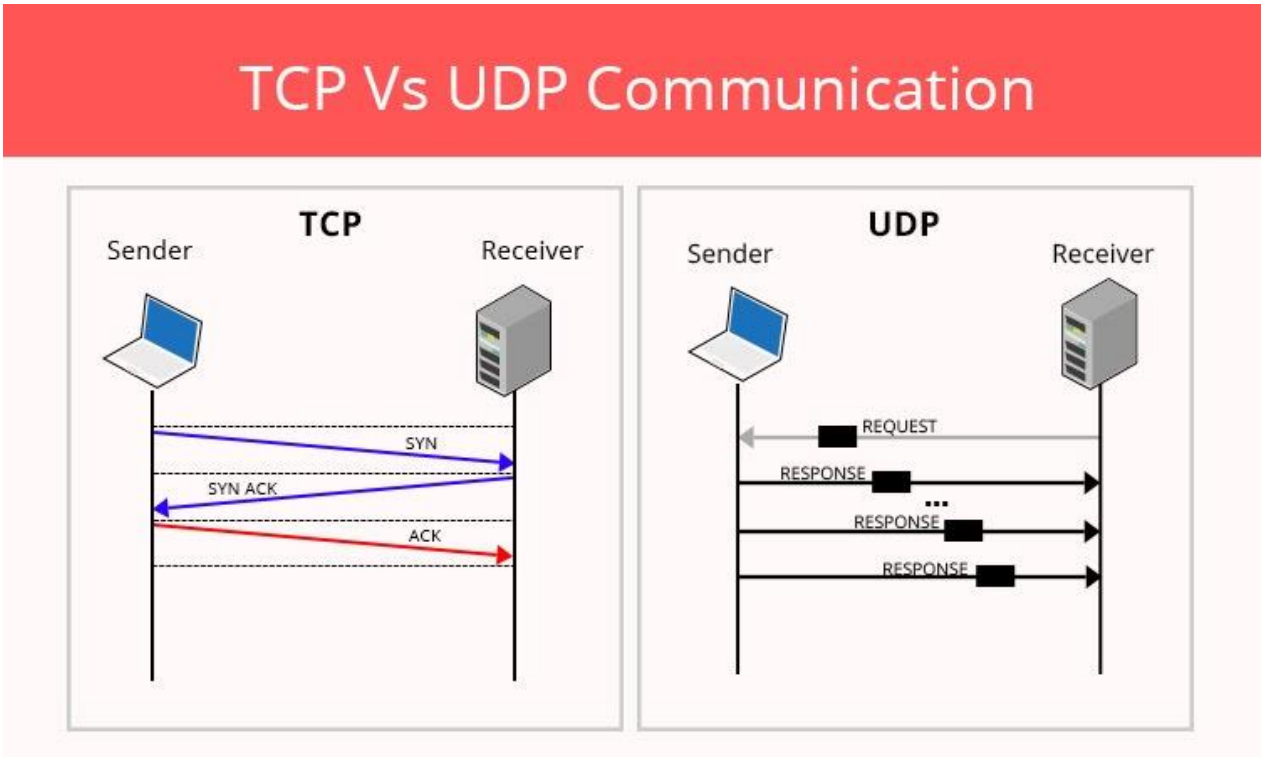
Pentru a dezvolta această aplicație am apelat la următoarele tool-uri:

- Microsoft Visual Studio 2012 – acesta este IDE-ul standard folosit în industria software pentru realizarea aplicațiilor desktop în limbajele C++, C, C#, etc. oferit de către Microsoft
- Oracle VM VirtualBox – acesta este un tool open-source de virtualizare pe care l-am folosit pentru a hosta componenta server a aplicației pe un sistem Windows Server 2016
- Wireshark – am folosit acest tool open-source pentru a analiza și valida datele din rețea

În continuare voi prezenta atât protocoalele care stau la baza aplicației cât și funcționalitățile acesteia.

UDP (User Datagram Protocol)

Protocolul UDP este un protocol de comunicare ce aparține nivelului 4 (nivelul de transport) în cadrul stivei OSI (figura 1). Acest protocol este utilizat în situațiile în care cel mai important aspect este viteza de transmitere. Principala sa calitate este rapiditatea, iar acest lucru este posibil deoarece, comparativ cu TCP, UDP nu necesită stabilirea unei conexiuni cu destinatarul mesajelor.



UDP transmite mesaje (numite datagrame utilizator) fără a verifica dacă acestea au ajuns la destinatar sau dacă au ajuns în ordine și nu are mecanisme prin care să realizeze controlul fluxului.

Principalele caracteristici ale protocolului UDP îl fac potrivit pentru protocoale precum: TFTP (Trivial File Transfer Protocol), SNMP (Simple Network Management Protocol), DHCP (Dynamic

Host Configuration Protocol) , DNS (Domain Name Sytems), dar și pentru aplicațiile în care se realizează schimbul de date media cum ar fi jocurile online, transmiterea de date video de pe diferite platforme de divertisment sau aplicațiile VoIP. Acest lucru se întâmplă deoarece, deși UDP nu asigură un procent de 100% al corectitudinii datelor transmise, pierderea câtorva pachete este de preferat în comparație cu întarzierea fluxului de date care poată să apară din cauza procesului prin care alte protocoale precum TCP transmit datele.

Header-ul UDP

Header-ul unui pachet UDP are o lungime fixă de 8 octeți și este compus din 4 câmpuri, fiecare având o dimensiune de 2 octeți după cum urmează:

- **Source port**: reprezintă portul de pe care s-a transmis mesajul. Dacă versiunea de IP de pe care s-a transmis mesajul este IPv4, atunci acest câmp este opțional, cazul în care acest câmp este gol reprezentând valoarea 0
- **Port of destination**: acest câmp este obligatoriu și reprezintă portul către care acest mesaj este transmis
- **Length**: acest câmp reprezintă lungimea mesajului (exprimată în octeți). Această lungime include atât lungimea header-ului cât și a datelor transmise, așadar lungimea minimă este egală cu 8
- **Checksum**: asigură imunitatea la erori. Se calculează ca fiind complementul față de 1, pe 16 biți, a pseudo-header-ului cu informații extrase din header-ul de IP, header-ul de UDP și a câmpului de date. În cazul în care lungimea este prea mică se completează cu 0 pentru a ajunge la lungimea stabilită.

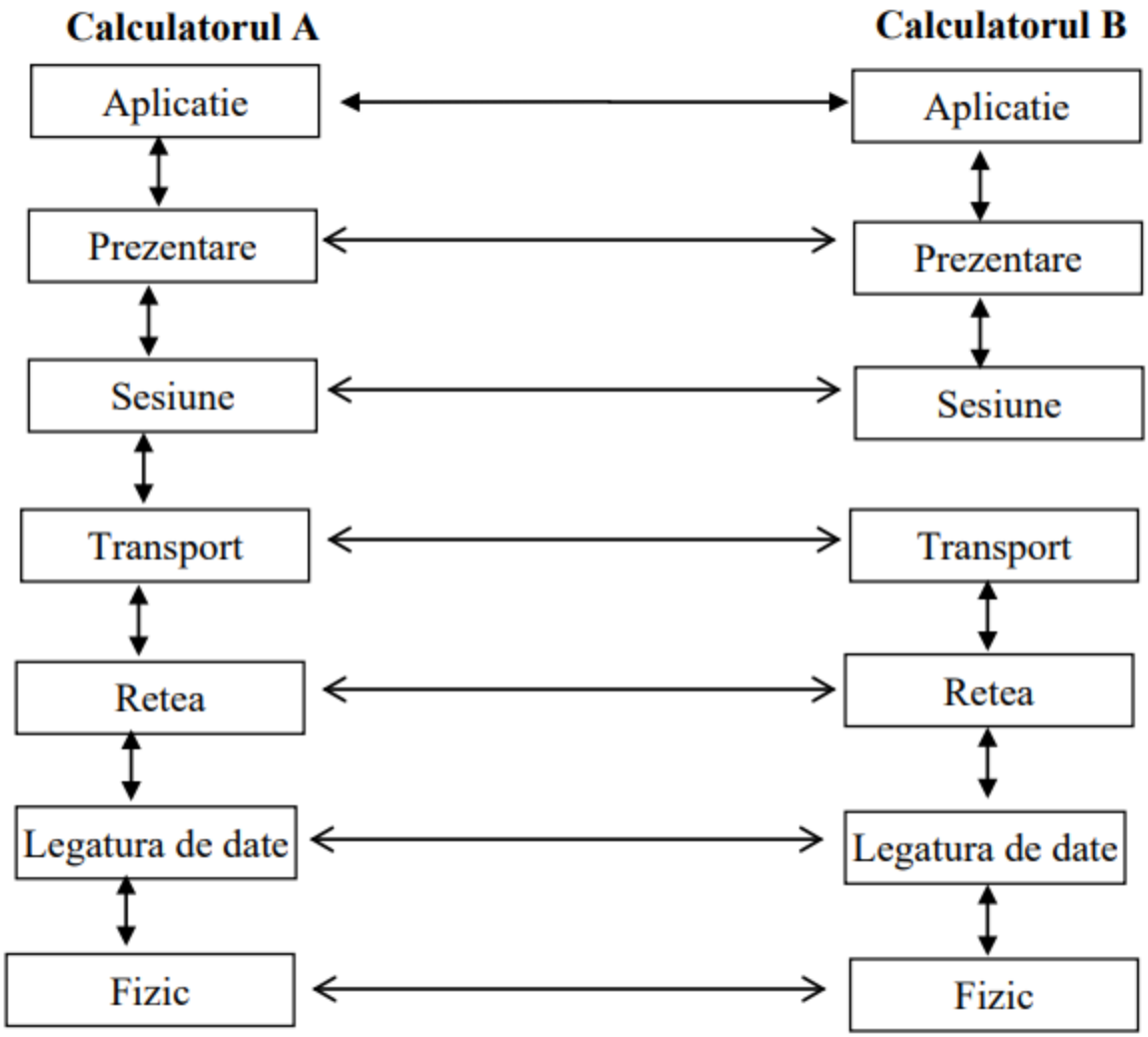
BIT	0 - 15	16 - 32
0	Source port	Port of destination
32	Length	Check amount
64	data	

(Header-ul unui pachet UDP)

Riscul ca pachetele UDP să fie pierdute este în general scăzut și de cele mai multe ori neglijabil. Însă, modul în care protocolul UDP se comportă îl face expus la atacuri cibernetice.

Unul din aceste riscuri este atacul de tip “flood” (engleză: a inunda). Din cauza faptului că protocolul UDP nu necesită o conexiune pentru a începe să transmită mesaje, atacatorii pot “inunda” server-ul țintă cu un număr mare de mesaje UDP trimise către porturi aleatoare. Acest lucru va forța server-ul țintă să răspundă fiecărui mesaj UDP printr-un mesaj ICMP prin care să transmită că portul respectiv nu este disponibil. Resursele consumate pentru a răspunde fiecărui mesaj transmis de atacator pot duce la epuizarea resurselor acestuia în momentul în care mesajele legitime de la alți utilizatori ajung la el.

Pentru a se proteja de astfel de atacuri organizațiile pot folosi o rețea intermediară care să se ocupe de primirea și transmiterea traficului UDP, astfel protejând server-ul de “inundarea” acestuia cu mesaje.



(figura 1 – modelul OSI)

SIP (Session Initiation Protocol)

SIP este un protocol de semnalizare utilizat pentru inițierea , menținerea și terminarea sesiunilor media în timp real. Protocolul SIP asigură semnalizarea și controlul sesiunilor în aplicațiile VoIP (Voice over IP), în aplicații de mesagerie instantă prin intermediul internetului, în rețele private de telefonie prin internet, dar și în cadrul apelurilor de telefonie realizate prin tehnologia VoLTE (Voice over LTE).

Acesta este un protocol construit la nivelul aplicației și a fost dezvoltat în așa fel încât să fie agnostic de protocolul de transport, mesajele SIP putând fi transmise prin UDP, TCP sau SCTP.

SIP este un protocol bazat pe text și incorporează multe elemente din protocolul HTTP (Hypertext Transfer Protocol), dar și din protocolul SMTP (Simple Mail Transfer Protocol).

Acesta definește formatul specific al mesajelor și funcționează împreună cu alte protocoale pentru a negocia parametrii media și a transporta datele. Cel mai adesea această negociere se realizează cu ajutorul protocolului SDP, mesajele SDP fiind încapsulate în corpul mesajului SIP.

```
INVITE sip:stefan@easySIP.com SIP/2.0
Via: SIP/2.0/UDP 192.168.1.9:5060;branch=e4d3c860-8190-4bb0-a51a-2fb7bfdf45f1;rport
Max-Forwards: 70
From: <sip:alexandra@easySIP.com>;tag=7f337f2a-dc64-4db3-b7df-6e4d97ff2632
To: <sip:stefan@easySIP.com>
Call-ID: 638147db-b49d-4c7b-9fc5-008920b13c7f
CSeq: 1 INVITE
Contact: <sip:alexandra@easySIP.com>
Allow: INVITE, ACK, BYE
User-Agent: WORKSTATION/4.0.0
Content-Type: application/sdp
Content-Length: 146

v=0
o=alexandra 123456 123456 IN IP4 192.168.1.9
s=-
c=IN IP4 192.168.1.9
t=0 0
m=audio 6666 RTP/AVP 0 8
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
```

Elementele de rețea

Elementele de rețea care folosesc SIP pentru comunicare poartă denumirea de "*SIP user agents*". Fiecare *user agent* se poate comporta atât ca un user-agent-client în momentul în care trimite o cerere, cât și ca user-agent-server în momentul în care răspunde unei cereri. Așadar, în teorie oricare două endpoint-uri SIP pot funcționa fără ajutorul niciunui alt element de rețea.

În practică însă, aproape toate rețelele SIP folosesc elemente de rețea specializate pentru a facilita comunicarea.

User agent

Un user agent reprezintă un endpoint în rețea, capabil să trimită și să recepționeze mesaje SIP. Spre deosebire de alte protocoale (precum HTTP) în care rolurile de client și server sunt fixate, în cadrul SIP aceste roluri durează doar în cadrul unei tranzacții.

Un telefon SIP este un telefon IP care implementează aceste două roluri pentru a putea comunica cu celelalte elemente în cadrul unei rețele SIP. Acesta oferă funcționalități de bază precum inițierea, acceptarea și terminarea unui apel. Există și versiuni mai performante care oferă mai multe funcționalități cum ar fi: hold/retrieve, call-forwarding (unconditional, on-busy, on-no answer, on-unreachable), call-waiting, conferințe, transmiterea și recepționarea de mesaje (atât SMS cât și mesaje text instante). Aceste telefoane pot fi fizice, fiind implementate la nivel hardware, dar și sub formă de programe software care pot rula de pe orice dispozitiv. Dispozitivele care oferă aceste funcționalități prin intermediul unui program software poartă denumirea de “softphone”. Așadar, aplicația de față este un softphone ce pune la dispoziția utilizatorilor funcționalități de bază precum inițierea, acceptarea, respingerea și terminarea apelurilor.

Proxy server

Un server proxy este un element de rețea care se comportă ca o entitate intermediară, cu scopul de a redirecționa mesajele SIP dintre alte elemente de rețea, facilitând comunicarea între acestea. Acesta interpretează mesajele recepționate și, la nevoie, le poate modifica, înainte de a le transmite. În cadrul rețelelor SIP, proxy server-ul are și rolul de a impune anumite politici (ex: de a restricționa anumite servicii).

Redirect server

Un server redirect este un server capabil să genereze răspunsuri de forma 3xx (redirection) pentru mesajele primite. Acesta permite astfel redirecționarea mesajelor pentru scenariile de forwarding.

Registrar

Un registrar este un endpoint SIP care oferă servicii de localizare. Acesta acceptă mesaje de tip REGISTER și înregistrează informații precum identitatea și adresa utilizatorului pentru ca mai apoi să le ofere proxy server-ului. Proxy server-ul folosește aceste informații pentru a distribui mesajele în mod corect în cadrul rețelei.

Session border controller

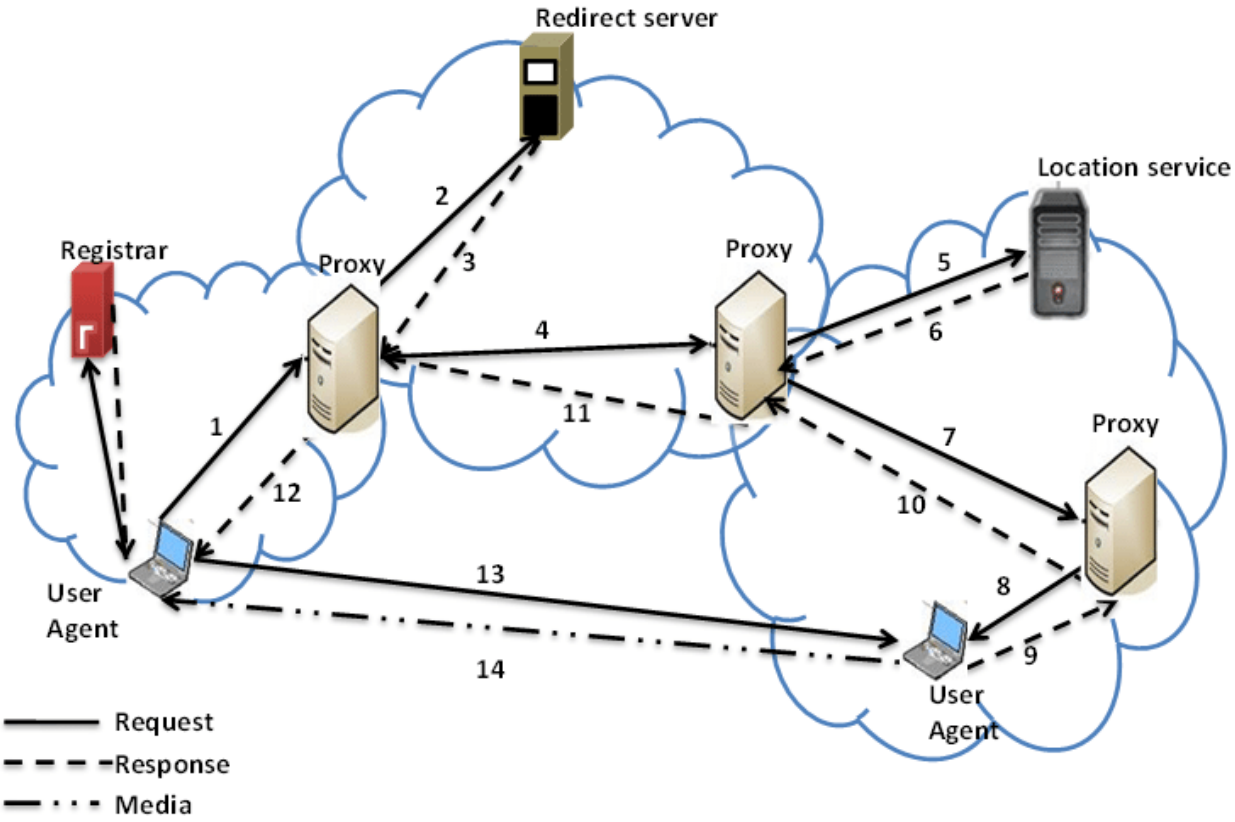
Un session border controller acționează ca un middle-man între user-agent și proxy server, în cadrul unei rețele SIP, oferind asistență în activități precum “network topology hiding” sau “NAT traversal”.

Gateway

Un gateway are rolul de a conecta o rețea SIP la o altă rețea exterioară care folosește alte tehnologii și protocoale pentru a comunica, cum ar fi rețelele clasice de telefonie PSTN (Public Switched Telephony Network).

În aplicația de față, componenta server joacă atât rolul de proxy server cât și cel de registrar.

În funcție de dimensiunile și rolul lor, rețelele SIP pot avea diferite topologii.



Mesajele SIP

Protocolul SIP definește două categorii de mesaje:

1. Requests
2. Responses

Prima linie a unui *request* conține o metodă care definește tipul de request, urmată de adresa (URI) unde mesajul trebuie să ajungă.

Prima linie a unui *response* conține un *response code* care indică rezultatul tranzacției (success/failure).

SIP Request

Prin intermediul unui mesaj request se inițiază o anumită funcționalitate a protocolului. Aceste mesaje sunt transmise de către user-agent-client către server, iar acesta răspunde prin unul sau mai multe mesaje de tip response.

Mesajele de tip request se diferențiază prin prima linie pe care se regăsesc metoda, request URI-ul, și versiunea de protocol, separate printr-un spațiu.

Metoda – reprezintă tipul de acțiune pe care mesajul o inițiază

Request-URI – acest câmp este un sip URI și indică user agent-ul căruia mesajul îi este adresat

Versiunea de protocol – acest câmp trebuie să aibă valoarea “SIP/2.0” pentru ca mesajul SIP să fie unul valid

Metodele SIP se împart în două categorii:

1. Metode de bază
2. Metode extinse

Există șase metode de bază, după cum urmează:

- **INVITE –** această metodă este folosită pentru a iniția o sesiune media între doi utilizatori. O sesiune este considerată ca fiind stabilită în momentul în care mesajului INVITE i se răspunde cu un cod de succes (de forma 2xx) sau un mesaj ACK a fost transmis. Odată ce sesiunea a fost stabilită aceasta continuă până când un mesaj BYE este transmis pentru a o termina. Un alt mesaj INVITE transmis în timpul unei sesiuni active poartă numele de re-INVITE și are rolul de a schimba anumite caracteristici ale sesiunii.

```
INVITE sip:stefan@easySIP.com SIP/2.0
Via: SIP/2.0/UDP 192.168.1.9:5060;branch=e4d3c860-8190-4bb0-a51a-2fb7bfdf45f1;rport
Max-Forwards: 70
From: <sip:alexandra@easySIP.com>;tag=7f337f2a-dc64-4db3-b7df-6e4d97ff2632
To: <sip:stefan@easySIP.com>
Call-ID: 638147db-b49d-4c7b-9fc5-008920b13c7f
CSeq: 1 INVITE
Contact: <sip:alexandra@easySIP.com>
Allow: INVITE, ACK, BYE
User-Agent: WORKSTATION/4.0.0
Content-Type: application/sdp
Content-Length: 146

v=0
o=alexandra 123456 123456 IN IP4 192.168.1.9
s=-
c=IN IP4 192.168.1.9
t=0 0
m=audio 6666 RTP/AVP 0 8
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
```

- **BYE** – acest tip de metodă este folosită pentru a încheia o sesiune activă și poate fi trimisă numai în aceste condiții. Mesajul BYE poate fi trimis de către oricare dintre participanții la apel.

```
BYE sip:alexandra@easySIP.com SIP/2.0
CSeq: 1 BYE
Via: SIP/2.0/UDP 192.168.1.6:5060;branch=658219b9-a244-4519-
babf-5b907cec7c29;rport
From: <sip:stefan@easySIP.com>;tag=7d6394b4-3d97-4eb1-9b13-deb2801e6e07
Call-ID: 0b19a606-1c41-41de-ad3d-487ed2fa24a0
To: <sip:alexandra@easySIP.com>
Content-Length: 0
Max-Forwards: 70
```

- **REGISTER** – metoda este folosită pentru a realiza înregistrarea unui utilizator la server-ul registrar.

```
REGISTER sip:192.168.1.8 SIP/2.0
CSeq: 1 REGISTER
Via: SIP/2.0/UDP 192.168.1.6:5060;branch=c4567a2a-7a70-4d7e-a572-a3eff4c83a8c;rport
User-Agent: RO-000088/4.0.0
From: <sip:stefan@easySIP.com>;tag=279924fa-d514-43e6-9211-4cca3564c343
Call-ID: 54077059-d6c5-4e76-bd42-c0005079a0b3
To: <sip:stefan@easySIP.com>
Contact: <sip:stefan@easySIP.com>
Allow: INVITE, ACK, BYE
Expires: 3600
Content-Length: 0
Max-Forwards: 70
```

- **CANCEL** - această metodă poate să fie transmisă doar de către inițiatorul unei sesiuni și are rolul de a renunța la stabilirea acesteia înainte ca celălalt participant să trimită un răspuns pozitiv sau negativ.

```
CANCEL sip:alex@easySIP.com SIP/2.0
CSeq: 1 CANCEL
Via: SIP/2.0/UDP
192.168.1.4:5060;branch=32b64146-70ca-40ed-83f4-7a1ab84f04e5;rport
From:
<sip:stefan@easySIP.com>;tag=96cc3bff-017e-4ba1-9af3-6aebe0b0fe83
Call-ID: ab1e8ad1-d6f7-4acf-a7cf-60cad2db26b1
To: <sip:alex@easySIP.com>
Content-Length: 0
Max-Forwards: 70
```

- **ACK** – este folosit pentru a confirma că un răspuns este final, iar după el nu mai urmează alte mesaje. Mesajul ACK poate conține în corpul său un mesaj SDP în cazul în care mesajul INVITE care a inițiat sesiunea nu a conținut astfel de informații.

```
ACK sip:stefan@easySIP.com SIP/2.0
CSeq: 1 ACK
Via: SIP/2.0/UDP 192.168.1.9:5060;branch=ac125527-9910-4643-a405-8e1e0e4e7334;rport
From: <sip:alexandra@easySIP.com>;tag=bbf9c55f-6b99-41b0-bc20-db25018d8eef
Call-ID: 638147db-b49d-4c7b-9fc5-008920b13c7f
To: <sip:stefan@easySIP.com>
Content-Length: 0
Max-Forwards: 70
```

- **OPTIONS** – este folosit de către clienți pentru a interoga un server în legătura cu funcționalitățile de care acesta este capabil

```
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
```

Metodele de extensie:

- **SUBSCRIBE** – este folosit de clienți pentru a iniția o subscripție cu scopul de a primi notificări în legatură cu un eveniment anume. Acest mesaj conține un header “Expires” care indică durata subscripției. Pentru a renunța, un client poate să trimită un nou mesaj SUBSCRIBE în care câmpul Expires să aibă valoarea 0. Subscripția se anulează automat în momentul în care durata setată expiră.
- **NOTIFY** – este folosit de către un user-agent pentru a notifica un altul în legătură cu evenimente la care acesta este abonat. Un mesaj NOTIFY va fi trimis de fiecare dată la începutul și sfârșitul unei subscripții**.**
- **PUBLISH** – este similar mesajului NOTIFY însă nu este nevoie ca un client să fie abonat pentru a recepționa astfel de mesaje
- **REFER** – este folosit pentru a îi cere destinatarului să transmită o solicitare (request) cu scopul de a transfera apelul
- **INFO** – acest mesaj este folosit de către un client pentru a transmite informații către un altul în momentul în care cei doi se află în cadrul unei sesiuni active
- **UPDATE**- este folosit pentru a schimba anumiți parametrii ai sesiunii (ex: codec-ul utilizat) dacă sesiunea nu este încă activă. Dacă sesiunea este deja activă atunci se va folosi un re-INVITE.
- **PRACK** – este transmis pentru a confirma primirea unui mesaj cu codul de răspuns de forma 1xx (provisional response)
- **MESSAGE** - utilizat în cadrul aplicațiilor de mesagerie instantă pentru a trimite un mesaj text

SIP Response

Mesajele de tip response sunt mesaje generate de către un user-agent-server sau de către un server SIP pentru a răspunde unei solicitări (SIP Request) trimise de către un client.

Acestea se diferențiază prin faptul că prima linie este compusă din următoarele câmpuri: *SIP-version*, *status-code* și *reason phrase*.

SIP-version – reprezintă versiunea de SIP (momentan aceasta are valaorea “SIP/2.0”)

Status-code – este un număr format din 3 cifre care indică rezultatul solicitării. Prima cifră a acestui numar indică clasa din care acesta face parte.

Reason-phrase - reprezintă un text, care are scopul de a oferi o scurtă descriere a status-code-ului

Mesajele de răspuns se împart în șase clase de forma:

- **1xx**: Provisional Responses – indică faptul că solicitarea este una validă și că se află în procesare
- **2xx**: Success Responses – indică faptul că solicitarea a fost completată cu succes
- **3xx**: Redirect Responses – în general aceste mesaje sunt transmise de către un redirect server ca răspuns pentru un mesaj INVITE. Rolul acestora este de a semnala faptul că, pentru ca solicitarea să fie procesată cu succes și sesiunea să fie stabilită, mesajul trebuie trimis către un alt destinatar (call redirection).
- **4xx**: Client Failure Responses – indică faptul că solicitarea nu poate fi realizată cu succes din cauza unor erori ce ține de client
- **5xx**: Server Failure Responses - semnalează faptul că solicitarea nu poate fi procesată cu succes din cauza unor erori în legătură cu server-ul
- **6xx**: Global Failure Responses – indică faptul că solicitarea nu va putea fi procesată cu success de către niciun server din rețea, așadar clientul nu are de ce să retransmită mesajul

Mesajele din clasa 1xx sunt considerate răspunsuri provizorii iar celelalte sunt considerate răspunsuri finale.

Următorul tabel prezintă toate răspunsurile SIP definite în standardele IETF RFC și înregistrate în registrele IANA.

100 Trying	180 Ringing	181 Call is Being Forwarded	182 Queued
183 Session Progress	199 Early Dialog Terminated		
200 OK	202 Accepted	204 No Notification	
300 Multiple Choices	301 Moved Permanently	302 Moved Temporarily	305 Use Proxy
380 Alternative Service			
400 Bad Request	401 Unauthorized	402 Payment Required	403 Forbidden
404 Not Found	405 Method Not Allowed	406 Not Acceptable	407 Proxy Authentication Required
408 Request Timeout	409 Conflict	410 Gone	411 Length Required
412 Conditional Request Failed	413 Request Entity Too Large	414 Request-URI Too Long	415 Unsupported Media Type
416 Unsupported URI Scheme	417 Unknown Resource-Priority	420 Bad Extension	421 Extension Required
422 Session Interval Too Small	423 Interval Too Brief	424 Bad Location Information	425 Bad Alert Message
428 Use Identity Header	429 Provide Referrer Identity	430 Flow Failed	433 Anonymity Disallowed

436 Bad Identity-Info	437 Unsupported Certificate	438 Invalid Identity Header	439 First Hop Lacks Outbound Support
440 Max-Breadth Exceeded	469 Bad Info Package	470 Consent Needed	480 Temporarily Unavailable
481 Call/Transaction Does Not Exist	482 Loop Detected	483 Too Many Hops	484 Address Incomplete
485 Ambiguous	486 Busy Here	487 Request Terminated	488 Not Acceptable Here
489 Bad Event	491 Request Pending	493 Undecipherable	494 Security Agreement Required
500 Internal Server Error	501 Not Implemented	502 Bad Gateway	503 Service Unavailable
504 Server Time-out	505 Version Not Supported	513 Message Too Large	555 Push Notification Service Not Supported
580 Precondition Failure			
600 Busy Everywhere	603 Decline	604 Does Not Exist Anywhere	606 Not Acceptable
607 Unwanted	608 Rejected		

(Tabel 1 – răspunsuri SIP)

înregistrarea la server

Pentru a putea iniția apeluri sau pentru a le putea recepționa, fiecare utilizator trebuie să fie în prealabil înregistrat la server. Acest lucru este necesar deoarece clienții nu își cunosc unul altuia locația, ei având nevoie de un proxy server care să le redirecționeze mesajele către destinația dorită.

În momentul în care se deschide aplicația, prima fereastră afișeaza un câmp în care se introduce adresa SIP la care utilizatorul va putea fi contactat de către ceilalți clienți prin intermediul proxy server-ului.

După apăsarea butonului “Register” un mesaj REGISTER este generat și transmis către server acesta având dublu rol (atât de proxy server cât și de registrar).

Adresa la care se află server-ul este configurabilă și este citită din fișierul “client_config.xml”.

client_config.xml

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <server_ip>192.168.1.9</server_ip>
    <server_port>5060</server_port>
    <ringtone>recorded.wav</ringtone>
</config>
```

Mesajul REGISTER este caracterizat prin prezența metodei “REGISTER“ urmată de adresa server-ului pe prima linie.

```
REGISTER sip:192.168.1.8 SIP/2.0
CSeq: 1 REGISTER
Via: SIP/2.0/UDP 192.168.1.6:5060;branch=c4567a2a-7a70-4d7e-a572-a3eff4c83a8c;rport
User-Agent: RO-000088/4.0.0
From: <sip:stefan@easySIP.com>;tag=279924fa-d514-43e6-9211-4cca3564c343
Call-ID: 54077059-d6c5-4e76-bd42-c0005079a0b3
To: <sip:stefan@easySIP.com>
Contact: <sip:stefan@easySIP.com>
Allow: INVITE, ACK, BYE
Expires: 3600
Content-Length: 0
Max-Forwards: 70
```

Acest mesaj este construit cu ajutorul următoarelor headere:

- Via: “UDP” – protocolul utilizat pentru a transporta mesajul, urmat de adresa IP a utilizatorului care încearcă să se înregistreze și de portul pe care acesta așteaptă să primească răspunsul; parametrul “branch” își păstrează valoarea în cadrul tuturor mesajelor din cadrul unei tranzacții. Acest parametru trebuie să înceapă întotdeauna cu secvența de caractere “z9hG4bk”, astfel încât server-ul să recunoască că acest parametru se supune acestor reguli (este unic în cadrul unei tranzacții)
- Max-Forwards: 70 - numărul maxim de retransmisii permis
- From: adresa SIP a utilizatorului care încearcă să se înregistreze, urmată de proprietatea “tag” care își păstrează valoarea în toate mesajele transmise de acest client pe parcursul acestei tranzacții. Această valoare este generată aleator.
- To: - în cazul mesajului REGISTER acesta are aceeași valoare ca și câmpul “From” , indicând astfel că tot el este cel care așteaptă să primească răspunsul final
- Call-ID: - acest header conține o valoare generată aleator și care identifică toate mesajele din cadrul unei sesiuni
- CSeq: în cazul mesajelor REGISTER sequence number-ul pornește întotdeauna de la 1, urmat de numele metodei “REGISTER”

- Content-Length: 0 – lungimea conținutului mesajului
- User-Agent: - conține informații despre clientul care se înregistrează (numele producătorului dispozitivului, modelul dispozitivului, versiunea software folosită, versiunea sistemului de operare, adresa MAC, etc.)
- Expires: durata pentru care înregistrarea rămâne valabilă
- Allow: înștiințează server-ul în legătură cu metodele SIP suportate de către acest client
- Contact: semnalează server-ului către cine să trimită răspunsurile

Server-ul recepționează mesajul și extrage informațiile necesare pentru a înregistra locația clientului. Acesta generează ca răspuns un mesaj SIP 200 OK și îl transmite către client.

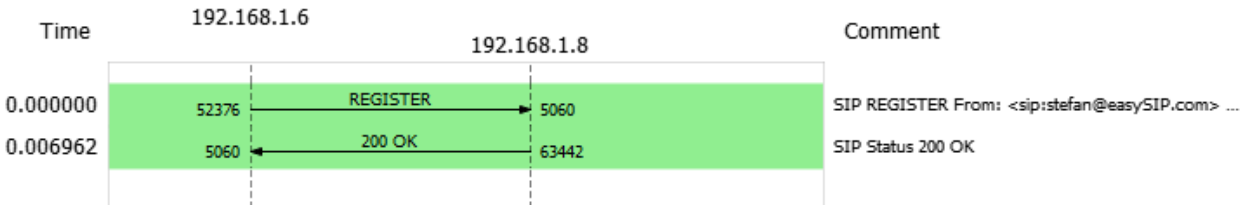
```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.1.6:5060;branch=c4567a2a-7a70-4d7e-a572-
a3eff4c83a8c;rport
From: <sip:stefan@easySIP.com>;tag=279924fa-d514-43e6-9211-4cca3564c343
To: <sip:stefan@easySIP.com>
Call-ID: 54077059-d6c5-4e76-bd42-c0005079a0b3
CSeq: 1 REGISTER
Contact: <sip:stefan@easySIP.com>
Content-Length: 0
```

Răspunsul 200 OK trimis de server este compus din următoarele headere:

- Via: “UDP” – protocolul utilizat pentru a transporta mesajul, urmat de adresa IP a utilizatorului care a transmis request-ul anterior și de portul la care acesta așteaptă să primească răspunsul; parametrul “branch” are aceeași valoare pe care a avut-o și în cadrul metodei REGISTER pe care utilizatorul A a transmis-o anterior
- From: adresa SIP a utilizatorului care a trimis request-ul anterior, urmată de proprietatea “tag” care își păstrează valoarea avută în mesajele SIP precendente ale acestei sesiuni
- To: - adresa SIP prezentă în mesajul SIP anterior
- Call-ID: - acest header își păstrează aceeași valoare ca și celalalte mesaje SIP transmise în cadrul sesiunii până în acel moment

- CSeq: sequence number-ul este unul generat aleator, urmat de metoda “CANCEL”
- Content-Length: 0 – lungimea conținutului mesajului

Clientul primește mesajul 200 OK, iar în acest moment tranzacția ia sfârșit.

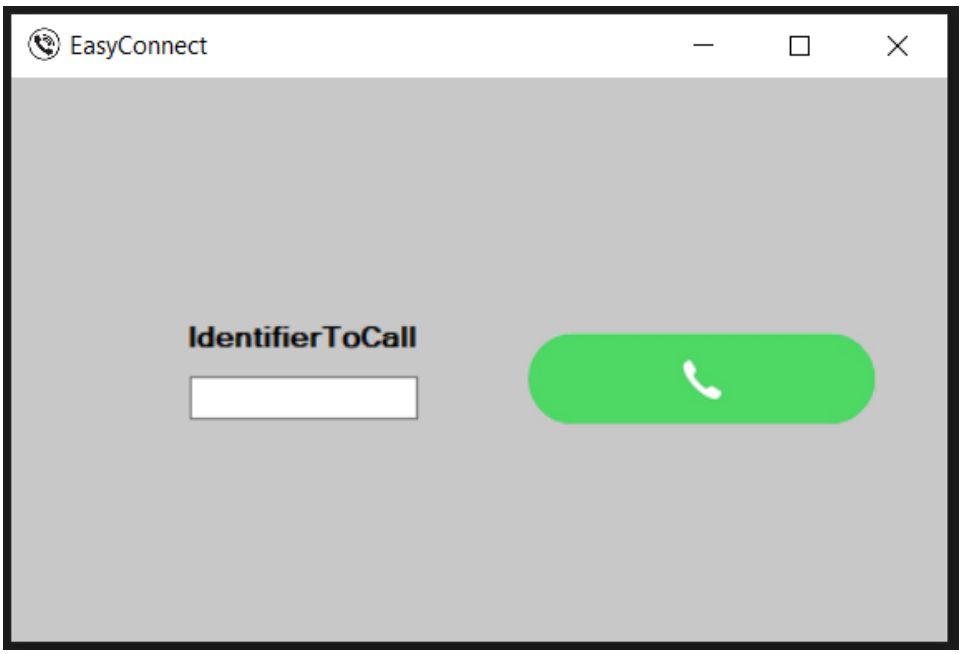


Inițierea și stabilirea cu succes a unei sesiuni

Odată înregistrat la server, clientul este capabil să inițieze sau să primească apeluri.

Din cea de-a doua fereastră a aplicației clientul are posibilitatea de a inița un apel.

Pentru a realiza acest lucru el trebuie să introducă în câmpul special destinat, adresa SIP a utilizatorului pe care încearcă să îl contacteze.



Orice sesiune din cadrul unei rețele VoIP este inițiată prin intermediul unui mesaj INVITE.

În momentul în care butonul “CALL” este apăsat, clientul (clientul A) construiește un mesaj SIP INVITE și îl transmite server-ului.

```
INVITE sip:stefan@easySIP.com SIP/2.0
Via: SIP/2.0/UDP 192.168.1.9:5060;branch=e4d3c860-8190-4bb0-a51a-2fb7bfdf45f1;rport
Max-Forwards: 70
From: <sip:alexandra@easySIP.com>;tag=7f337f2a-dc64-4db3-b7df-6e4d97ff2632
To: <sip:stefan@easySIP.com>
Call-ID: 638147db-b49d-4c7b-9fc5-008920b13c7f
CSeq: 1 INVITE
Contact: <sip:alexandra@easySIP.com>
Allow: INVITE, ACK, BYE
User-Agent: WORKSTATION/4.0.0
Content-Type: application/sdp
Content-Length: 146

v=0
o=alexandra 123456 123456 IN IP4 192.168.1.9
s=-
c=IN IP4 192.168.1.9
t=0 0
m=audio 6666 RTP/AVP 0 8
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
```

Mesajul INVITE se diferențiază prin prezența pe prima linie a metodei “INVITE”, urmată de adresa SIP a utilizatorului căruia acest mesaj îi este destinat.

Acest mesaj este alcătuit din următoarele headere:

- Via: “UDP” – protocolul utilizat pentru a transporta mesajul, urmat de adresa IP a utilizatorului care inițiază apelul și de portul la care acesta asteaptă să primească răspunsul; parametrul “branch” își păstrează valoarea în cadrul tuturor mesajelor din cadrul unei tranzacții. Acest parametru trebuie să înceapă întotdeauna cu secvența de caractere “z9hG4bk”, astfel încât server-ul să recunoască că acest parametru se supune acestor reguli (este unic în cadrul unei tranzacții)
- Max-Forwards: 70 - numărul maxim de retransmisii permis
- From: adresa SIP a utilizatorului care inițiază apelul, urmată de proprietatea “tag” care își păstrează valoarea în toate mesajele transmise de acest client pe parcursul acestei sesiuni. Această valoare este generată aleator.
- To: - adresa SIP a utilizatorului care este contactat

- Call-ID: - acest header conține o valoare generată aleator și care identifică toate mesajele din cadrul acestei sesiuni
- CSeq: sequence number-ul este unul generat aleator, urmat de metoda “INVITE”
- Content-Type: indică tipul media transmis în corpul acestui mesaj. În cazul în care corpul mesajului nu este gol, acest header este obligatoriu. Valoarea “application/sdp” semnalează prezența unui mesaj SDP în corpul acestui mesaj
- Content-Length: 146– lungimea conținutului mesajului transmis. Dacă niciun mesaj nu este transmis în corpul mesajului SIP atunci acest header ia valoarea 0
- User-Agent: - poate conține informații despre clientul care se înregistrează (numele producătorului dispozitivului, modelul dispozitivului, versiunea software folosită, versiunea sistemului de operare, adresa MAC, etc.)
- Allow: înștiințează server-ul în legătură cu metodele SIP suportate de către acest client
- Contact: semnalează server-ului către cine să trimită răspunsurile

Conținutul mesajului SDP transmis în interiorul mesajului INVITE are rolul de a oferi destinatarului informații în legătură cu formatele media suportate și cu detaliile despre conexiune. Acesta este detaliat mai pe larg în capitolul SDP (pagina 49).

Server-ul recepționează mesajul INVITE, extrage adresa SIP a destinatarului și încearcă să găsească informațiile legate despre adresa acestuia. Dacă un utilizator s-a înregistrat la server cu această adresă SIP, atunci acesta trimite mesajul INVITE mai departe către acel utilizator (clientul B).

Mesajul păstrează aceleași valori însă numărul maxim de retransmisii permise scade cu 1, iar în cadrul mesajului apare un nou header Via pentru a marca faptul că mesajul a fost transmis prin intermediul server-ului.

```
INVITE sip:stefan@easySIP.com SIP/2.0
Via: SIP/2.0/UDP 192.168.1.9:5060;branch=e4d3c860-8190-4bb0-a51a-2fb7bfdf45f1;rport
Via: SIP/2.0/UDP 192.168.1.8:5060;branch=dffddc62-367f-4589-aee5-ace28898918c;rport
Max-Forwards: 69
From: <sip:alexandra@easySIP.com>;tag=7f337f2a-dc64-4db3-b7df-6e4d97ff2632
To: <sip:stefan@easySIP.com>
Call-ID: 638147db-b49d-4c7b-9fc5-008920b13c7f
CSeq: 1 INVITE
Contact: <sip:alexandra@easySIP.com>
Allow: INVITE, ACK, BYE
User-Agent: WORKSTATION/4.0.0
Content-Type: application/sdp
Content-Length: 146

v=0
o=alexandra 123456 123456 IN IP4 192.168.1.9
s=-
c=IN IP4 192.168.1.9
t=0 0
m=audio 6666 RTP/AVP 0 8
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
```

După acest pas, server-ul trimite un răspuns 100 Trying către inițiatorul apelului pentru a-l înștiința că solicitarea lui a fost procesată și destinatarul mesajului INVITE a fost contactat.

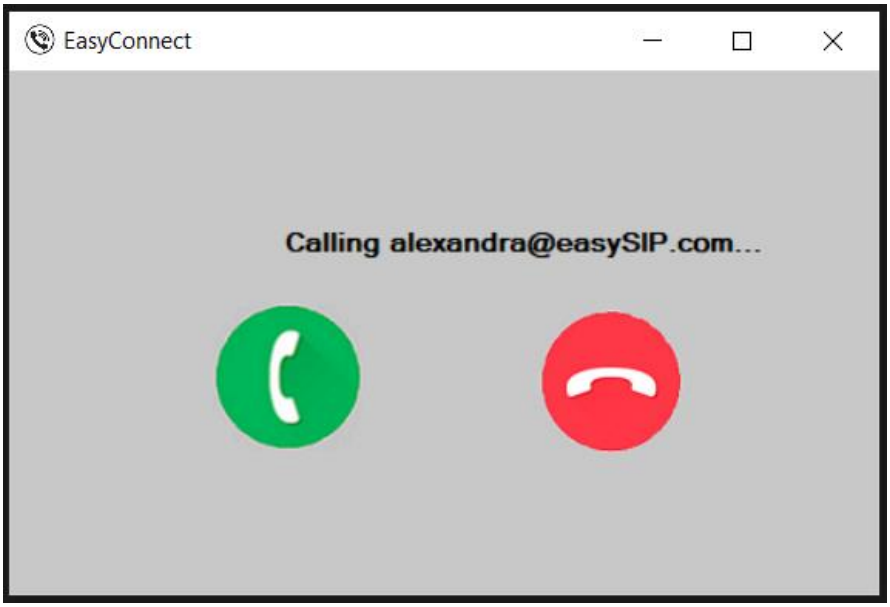
```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP 192.168.1.8:5060;branch=038bef61-2aa2-4429-a57e-92c25f37cb14;rport
From: <sip:alexandra@easySIP.com>;tag=7f337f2a-dc64-4db3-b7df-6e4d97ff2632
To: <sip:stefan@easySIP.com>
Call-ID: 638147db-b49d-4c7b-9fc5-008920b13c7f
CSeq: 1 INVITE
Content-Length: 0
```

Mesajul de răspuns 100 Trying este un răspuns din clasa 1xx ceea ce înseamnă că este un răspuns provizoriu, iar inițiatorul va mai aștepta și alte mesaje de răspuns.

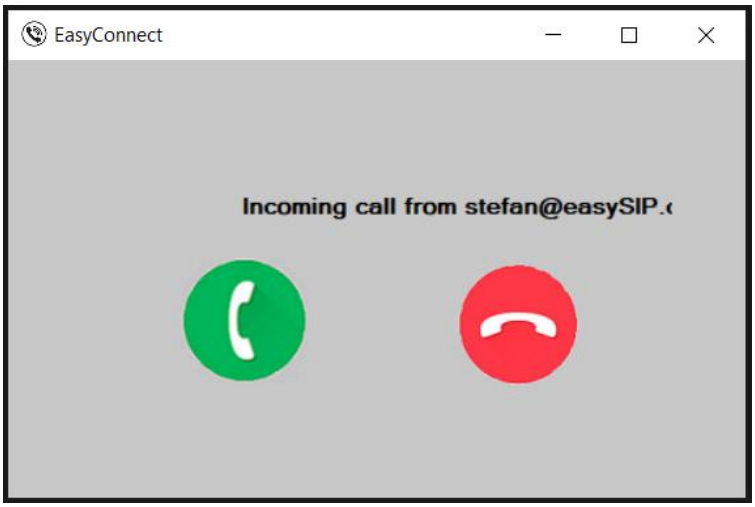
Acest mesaj este compus din header-ele: Via, From, To, Call-ID, CSeq și Content-Length. El se diferențiază prin faptul că pe prima linie se află codul de răspuns “100” urmat de textul “Trying” care descrie acțiunea executată de server.

Toate headere-ele își păstrează valorile avute în mesajul INVITE inițial, excepție făcând header-ul Via, care folosește de această dată adresa IP a server-ului (cel care a construit mesajul), și Content-Length care are acum valoarea 0, mesajul neavând nicio informație în corpul său.

Apelantul recepționează mesajul 100 Trying și deschide fereastra cu numărul trei, în care afișează mesajul “Calling <username>…” semnalând faptul că server-ul a transmis mai departe solicitarea acestuia.



În momentul în care clientul apelat primește mesajul INVITE de la server acesta deschide și el fereastra cu numărul 3 și afișează mesajul “Incoming call from <username…>” semnalând că a primit o solicitare pentru stabilirea unei sesiuni.



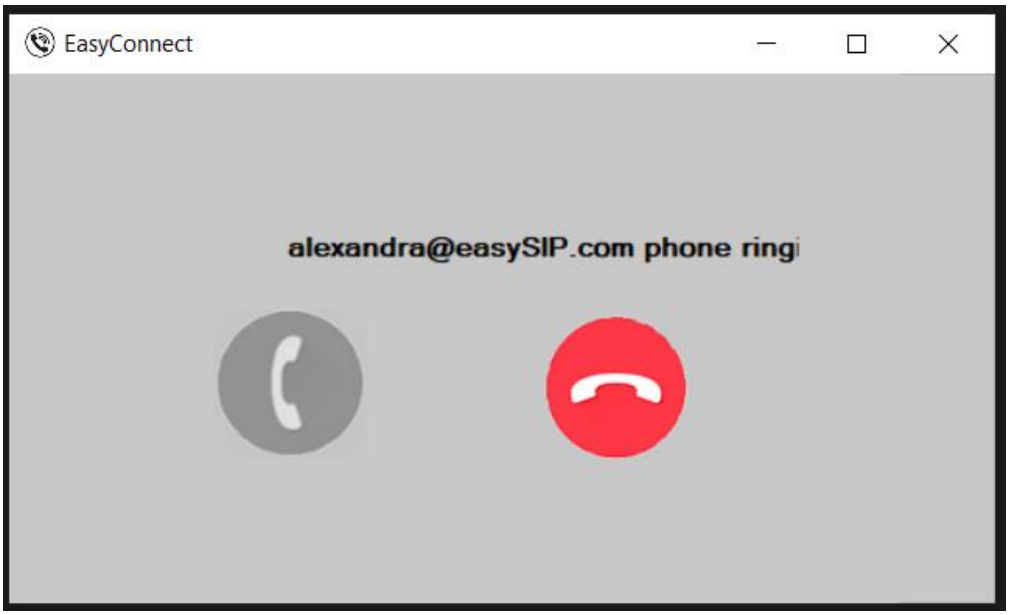
Totodată el generează un mesaj “180 Ringing” pentru a-l informa pe clientul A de acest lucru.

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP 192.168.1.6:5060;branch=3432770f-831f-46d7-a265-221517b48d81;rport
From: <sip:alexandra@easySIP.com>;tag=7f337f2a-dc64-4db3-b7df-6e4d97ff2632
To: <sip:stefan@easySIP.com>
Call-ID: 638147db-b49d-4c7b-9fc5-008920b13c7f
CSeq: 1 INVITE
Content-Length: 0
```

Headere-ele acestui mesaj păstrează aceleași valori pe care le-au avut și în cadrul mesajului INVITE. Pe prima linie a acestui mesaj se regăsesc codul de răspuns 180 urmat de cuvântul “Ringing“ care descrie acțiunea executată de clientul B.

Server-ul recepționează acest mesaj și îl transmite mai departe către clientul A.

Inițiatorul sesiunii primește acest răspuns și afișează în fereastră textul “<username> phone is ringing…” pentru a înștiința utilizatorul aplicației în legătură cu ceea ce se întamplă în rețea.



Clientul apelat are la dispoziție două opțiuni: de a accepta apelul sau de a-l respinge.

Dacă alege să accepte apelul (apăsând pe butonul “ANSWER”) un mesaj de răspuns 200 OK este construit și transmis către server.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.1.9:5060;branch=e4d3c860-8190-4bb0-a51a-2fb7bfdf45f1;rport
Via: SIP/2.0/UDP 192.168.1.8:5060;branch=dffddc62-367f-4589-aee5-ace28898918c;rport
From: <sip:alexandra@easySIP.com>;tag=7f337f2a-dc64-4db3-b7df-6e4d97ff2632
To: <sip:stefan@easySIP.com>
Call-ID: 638147db-b49d-4c7b-9fc5-008920b13c7f
CSeq: 1 INVITE
Contact: <sip:alexandra@easySIP.com>
Content-Type: application/sdp
Content-Length: 143

v=0
o=stefan 123456 123456 IN IP4 192.168.1.6
s=-
c=IN IP4 192.168.1.6
t=0 0
m=audio 6666 RTP/AVP 0 8
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
```

Mesajul este construit având pe prima linie codul de răspuns 200 urmat de textul OK care are semnificația că sesiunea a fost acceptată. Acest mesaj poartă în corpul său un mesaj SDP prin care clientul B, în urma analizării mesajului SDP transmis în cadrul request-ului INVITE, îl înștiințează la rândul său pe clientul A referitor la capabilitățile sale media și la adresa la care acestea sunt disponibile (IP și port).

Server-ul retransmite mesajul către clientul A.

Analizând câmpul “CSeq” clientul A își dă seama că acest mesaj a sosit ca răspuns pentru solicitarea de inițiere a sesiunii și trimite un mesaj ACK către clientul B, prin intermediul server-ului, prin care confirmă primirea acestui mesaj.

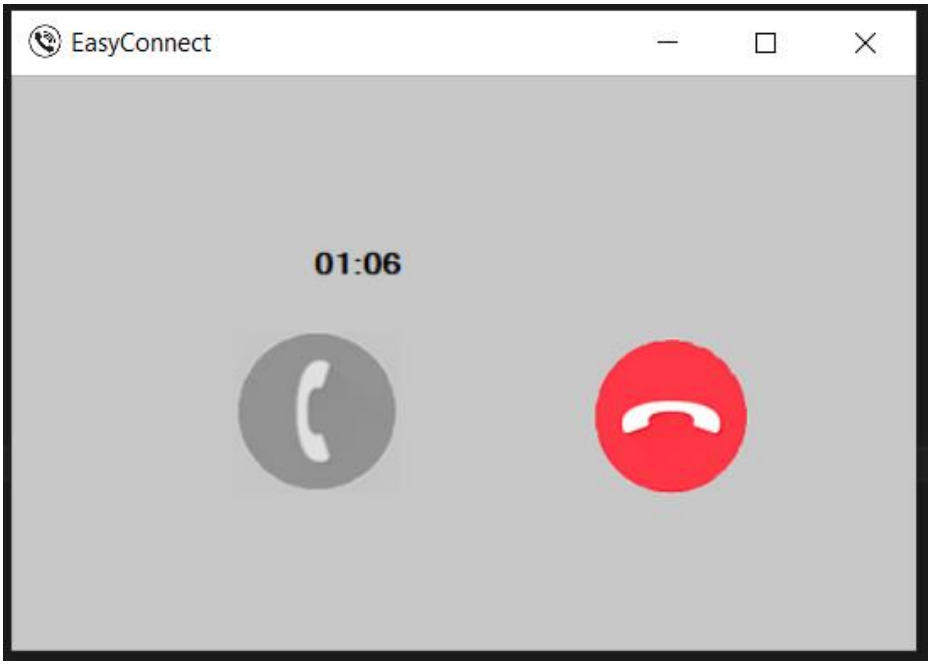
```
ACK sip:stefan@easySIP.com SIP/2.0
CSeq: 1 ACK
Via: SIP/2.0/UDP 192.168.1.9:5060;branch=ac125527-9910-4643-a405-8e1e0e4e7334;rport
From: <sip:alexandra@easySIP.com>;tag=bbf9c55f-6b99-41b0-bc20-db25018d8eef
Call-ID: 638147db-b49d-4c7b-9fc5-008920b13c7f
To: <sip:stefan@easySIP.com>
Content-Length: 0
Max-Forwards: 70
```

Din acest moment aplicația începe să înregistreze și să trimită în timp real către clientul B datele audio encodate în pachete RTP, conform informațiilor stabilite în cadrul mesajelor SDP și deschide portul configurat pentru a fi pregătit să asculte pachetele venite din rețea de la acesta.

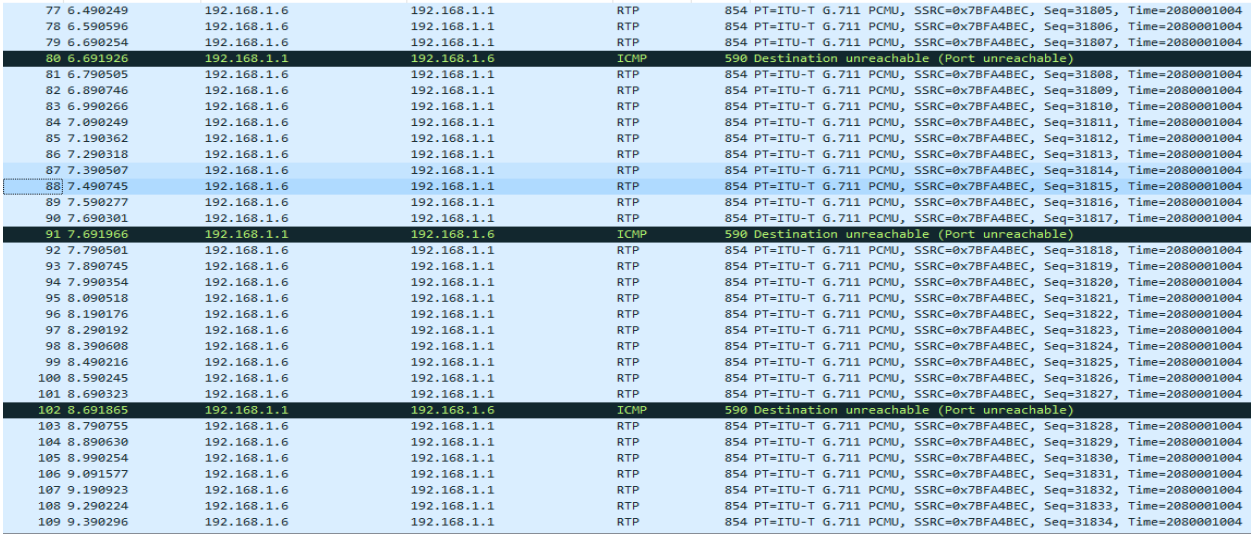
Server-ul trimite mai departe mesajul ACK către clientul B.

Clientul B recepționează mesajul ACK și la rândul său începe să înregistreze date audio și să le trimită către clientul A. De asemenea acesta deschide portul stabilit și începe să primească stream-ul de pachete RTP.

În acest moment sesiunea este stabilită și se realizează schimbul de pachete media între cei doi utilizatori. Pe ecran se afișează un timer care se incrementează cu fiecare secundă cu scopul de a măsura durata apelului.



În acest moment sesiunea rămâne activă până în momentul în care unul dintre cei doi utilizatori dorește să o închidă.



77	6.490249	192.168.1.6	192.168.1.1	RTP	854 PT=ITU-T G.711 PCMU, SSRC=0x7BFA4BEC, Seq=31805, Time=2080001004
78	6.590596	192.168.1.6	192.168.1.1	RTP	854 PT=ITU-T G.711 PCMU, SSRC=0x7BFA4BEC, Seq=31806, Time=2080001004
79	6.690254	192.168.1.6	192.168.1.1	RTP	854 PT=ITU-T G.711 PCMU, SSRC=0x7BFA4BEC, Seq=31807, Time=2080001004
80	6.691926	192.168.1.1	192.168.1.6	ICMP	590 Destination unreachable (Port unreachable)
81	6.790505	192.168.1.6	192.168.1.1	RTP	854 PT=ITU-T G.711 PCMU, SSRC=0x7BFA4BEC, Seq=31808, Time=2080001004
82	6.890746	192.168.1.6	192.168.1.1	RTP	854 PT=ITU-T G.711 PCMU, SSRC=0x7BFA4BEC, Seq=31809, Time=2080001004
83	6.990266	192.168.1.6	192.168.1.1	RTP	854 PT=ITU-T G.711 PCMU, SSRC=0x7BFA4BEC, Seq=31810, Time=2080001004
84	7.090249	192.168.1.6	192.168.1.1	RTP	854 PT=ITU-T G.711 PCMU, SSRC=0x7BFA4BEC, Seq=31811, Time=2080001004
85	7.190362	192.168.1.6	192.168.1.1	RTP	854 PT=ITU-T G.711 PCMU, SSRC=0x7BFA4BEC, Seq=31812, Time=2080001004
86	7.290318	192.168.1.6	192.168.1.1	RTP	854 PT=ITU-T G.711 PCMU, SSRC=0x7BFA4BEC, Seq=31813, Time=2080001004
87	7.390507	192.168.1.6	192.168.1.1	RTP	854 PT=ITU-T G.711 PCMU, SSRC=0x7BFA4BEC, Seq=31814, Time=2080001004
88	7.490745	192.168.1.6	192.168.1.1	RTP	854 PT=ITU-T G.711 PCMU, SSRC=0x7BFA4BEC, Seq=31815, Time=2080001004
89	7.590277	192.168.1.6	192.168.1.1	RTP	854 PT=ITU-T G.711 PCMU, SSRC=0x7BFA4BEC, Seq=31816, Time=2080001004
90	7.690301	192.168.1.6	192.168.1.1	RTP	854 PT=ITU-T G.711 PCMU, SSRC=0x7BFA4BEC, Seq=31817, Time=2080001004
91	7.691966	192.168.1.1	192.168.1.6	ICMP	590 Destination unreachable (Port unreachable)
92	7.790501	192.168.1.6	192.168.1.1	RTP	854 PT=ITU-T G.711 PCMU, SSRC=0x7BFA4BEC, Seq=31818, Time=2080001004
93	7.890745	192.168.1.6	192.168.1.1	RTP	854 PT=ITU-T G.711 PCMU, SSRC=0x7BFA4BEC, Seq=31819, Time=2080001004
94	7.990354	192.168.1.6	192.168.1.1	RTP	854 PT=ITU-T G.711 PCMU, SSRC=0x7BFA4BEC, Seq=31820, Time=2080001004
95	8.090518	192.168.1.6	192.168.1.1	RTP	854 PT=ITU-T G.711 PCMU, SSRC=0x7BFA4BEC, Seq=31821, Time=2080001004
96	8.190176	192.168.1.6	192.168.1.1	RTP	854 PT=ITU-T G.711 PCMU, SSRC=0x7BFA4BEC, Seq=31822, Time=2080001004
97	8.290192	192.168.1.6	192.168.1.1	RTP	854 PT=ITU-T G.711 PCMU, SSRC=0x7BFA4BEC, Seq=31823, Time=2080001004
98	8.390608	192.168.1.6	192.168.1.1	RTP	854 PT=ITU-T G.711 PCMU, SSRC=0x7BFA4BEC, Seq=31824, Time=2080001004
99	8.490216	192.168.1.6	192.168.1.1	RTP	854 PT=ITU-T G.711 PCMU, SSRC=0x7BFA4BEC, Seq=31825, Time=2080001004
100	8.590245	192.168.1.6	192.168.1.1	RTP	854 PT=ITU-T G.711 PCMU, SSRC=0x7BFA4BEC, Seq=31826, Time=2080001004
101	8.690323	192.168.1.6	192.168.1.1	RTP	854 PT=ITU-T G.711 PCMU, SSRC=0x7BFA4BEC, Seq=31827, Time=2080001004
102	8.691865	192.168.1.1	192.168.1.6	ICMP	590 Destination unreachable (Port unreachable)
103	8.790755	192.168.1.6	192.168.1.1	RTP	854 PT=ITU-T G.711 PCMU, SSRC=0x7BFA4BEC, Seq=31828, Time=2080001004
104	8.890630	192.168.1.6	192.168.1.1	RTP	854 PT=ITU-T G.711 PCMU, SSRC=0x7BFA4BEC, Seq=31829, Time=2080001004
105	8.990254	192.168.1.6	192.168.1.1	RTP	854 PT=ITU-T G.711 PCMU, SSRC=0x7BFA4BEC, Seq=31830, Time=2080001004
106	9.091577	192.168.1.6	192.168.1.1	RTP	854 PT=ITU-T G.711 PCMU, SSRC=0x7BFA4BEC, Seq=31831, Time=2080001004
107	9.190923	192.168.1.6	192.168.1.1	RTP	854 PT=ITU-T G.711 PCMU, SSRC=0x7BFA4BEC, Seq=31832, Time=2080001004
108	9.290224	192.168.1.6	192.168.1.1	RTP	854 PT=ITU-T G.711 PCMU, SSRC=0x7BFA4BEC, Seq=31833, Time=2080001004
109	9.390296	192.168.1.6	192.168.1.1	RTP	854 PT=ITU-T G.711 PCMU, SSRC=0x7BFA4BEC, Seq=31834, Time=2080001004

(pachete RTP schimbate între cei doi utilizatori în timpul sesiunii)

Pentru a închide sesiunea utilizatorul B apasă butonul “CLOSE”. În acel moment un mesaj SIP BYE este generat și transmis către server. Acesta are rolul de a termina o sesiune aflată în desfășurare.

Pe prima linie a acestuia se găsește metoda “BYE” urmată de adresa SIP a destinatarului, în acest caz clientul A.

```
BYE sip:alexandra@easySIP.com SIP/2.0
CSeq: 1 BYE
Via: SIP/2.0/UDP 192.168.1.6:5060;branch=658219b9-a244-4519-babf-5b907cec7c29;rport
From: <sip:stefan@easySIP.com>;tag=7d6394b4-3d97-4eb1-9b13-deb2801e6e07
Call-ID: 0b19a606-1c41-41de-ad3d-487ed2fa24a0
To: <sip:alexandra@easySIP.com>
Content-Length: 0
Max-Forwards: 70
```

Metoda din cadrul header-ului CSeq este “BYE” iar valoarea sequence number-ului este una generată aleator. Header-ul Call-Id păstrează valoarea pe care a avut-o și în celelalte mesaje ale sesiunii pe care acest mesaj o termină.

Server-ul trimite acest mesaj mai departe către destinatarul său.

În momentul în care clientul A primește acest mesaj se oprește din transmiterea și citirea datelor audio.

Un mesaj SIP 200 OK este trimis către clientul B, prin intermediul server-ului pentru a-l înștiința că solicitarea lui a fost procesată cu success. Aplicația revine la fereastra cu numărul doi și șterge informațiile despre această sesiune.

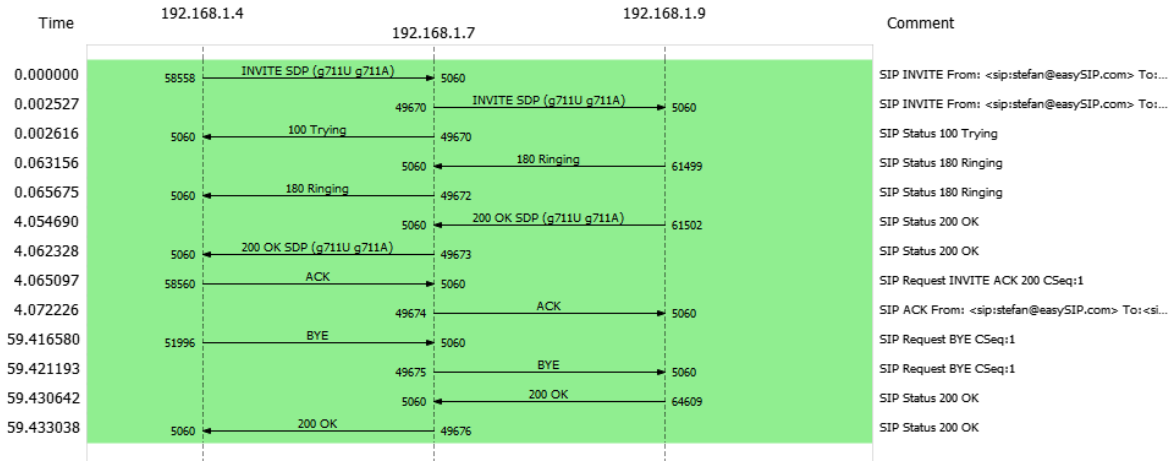
```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.1.6:5060;branch=658219b9-a244-4519-
babf-5b907cec7c29;rport
From: <sip:stefan@easySIP.com>;tag=7d6394b4-3d97-4eb1-9b13-deb2801e6e07
To: <sip:alexandra@easySIP.com>
Call-ID: 0b19a606-1c41-41de-ad3d-487ed2fa24a0
CSeq: 1 BYE
Contact: <sip:stefan@easySIP.com>
Content-Length: 0
```

De această dată metoda regăsită în cadrul header-ului CSeq a mesajului 200 OK este “BYE”.

Server-ul trimite acest mesaj mai departe către destinatarul său, clientul A.

Acesta procesează mesajul și, la rândul său, oprește recepționarea și transmiterea pacheteor RTP și revine la starea inițială.

În acest moment sesiunea a luat sfârșit.



(Diagrama unui apel VoIP)

Respingerea unui apel

În situația în care un utilizator primește o invitație la apel însă nu dorește să o accepte, acesta are opțiunea de a o respinge. Secvența inițială a mesajelor SIP schimbate între utilizatori și server este aceeași ca în cazul unui apel reușit până în acest punct.

Pentru a respinge apelul utilizatorul apelat (clientul B) trebuie să apese butonul “CLOSE” în momentul în care primește notificarea care semnalează că un alt client (clientul A) încearcă să îl contacteze.

În acel moment clientul B generează un mesaj de răspuns “480 Temporarily Unavailable” și îl transmite clientului A prin intermediul server-ului.

Acest mesaj se diferențiază prin faptul că prima linie este compusă din codul de răspuns 480 urmată de textul “Temporarily Unavailable” care semnalează faptul că generatorul acestui mesaj nu este disponibil pentru a participa la sesiune.

```
SIP/2.0 480 Temporarily Unavailable
Via: SIP/2.0/UDP 192.168.1.9:5060;branch=c5583715-30cf-4413-
bf6c-6eede09b4ce2;rport
From:
<sip:stefan@easySIP.com>;tag=4649b97f-0095-4177-9d0f-3219f4e1fa36
To: <sip:alex@easySIP.com>
Call-ID: 5cb376de-1552-472b-96d4-0735dcbc1e7c
CSeq: 1 INVITE
Content-Length: 0
```

Acest mesaj este alcătuit din următoarele headere:

- Via: “UDP” – protocolul utilizat pentru a transporta mesajul urmat de adresa IP a utilizatorului care a generat mesajul și de portul la care acesta așteaptă să primească răspunsul; parametrul “branch” își păstrează valoarea în cadrul tuturor mesajelor din cadrul unei tranzacții. Acest parametru trebuie să înceapă întotdeauna cu secvența de

caractere “z9hG4bk”, astfel încât server-ul să recunoască că acest parametru se supune acestor reguli (este unic în cadrul unei tranzacții)

- From: adresa SIP a utilizatorului care a inițiat apelul, urmată de proprietatea “tag” care își păstrează valoarea în toate mesajele transmise de acest client pe parcursul acestei sesiuni. Această valoare este generată aleator.
- To: - adresa SIP a utilizatorului care a fost contactat
- Call-ID: - acest header conține o valoare generată aleator și care identifică toate mesajele din cadrul acestei sesiuni
- CSeq: sequence number-ul păstrează aceeași valoare ca în mesajele precedente, urmat de metoda “INVITE”, deoarece acest mesaj este trimis ca răspuns pentru această metodă
- Content-Length: 0– lungimea conținutului mesajului transmis. Dacă niciun mesaj nu este transmis în corpul mesajului SIP atunci acest header ia valoarea 0

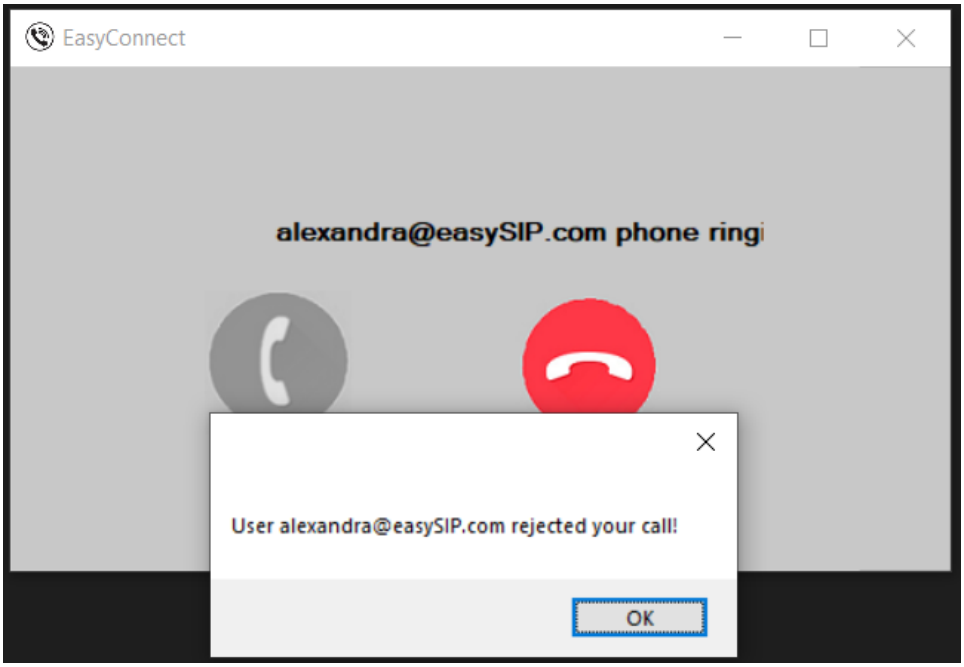
Server-ul recepționează răspunsul primit de la utilizatorul B și îi răspunde acestuia printr-un ACK pentru a-i confirma primirea mesajului. Din acest moment din punctul de vedere al clientului B sesiunea a luat sfârșit.

```
ACK sip:stefan@easySIP.com SIP/2.0
CSeq: 1 ACK
Via: SIP/2.0/UDP 192.168.1.9:5060;branch=ac125527-9910-4643-a405-8e1e0e4e7334;rport
From: <sip:alexandra@easySIP.com>;tag=bbf9c55f-6b99-41b0-bc20-db25018d8eef
Call-ID: 638147db-b49d-4c7b-9fc5-008920b13c7f
To: <sip:stefan@easySIP.com>
Content-Length: 0
Max-Forwards: 70
```

După ce trimite confirmarea (prin mesajul ACK) clientului B, proxy server-ul trimite mesajul 480 Temporarily Unavailable mai departe către clientul A.

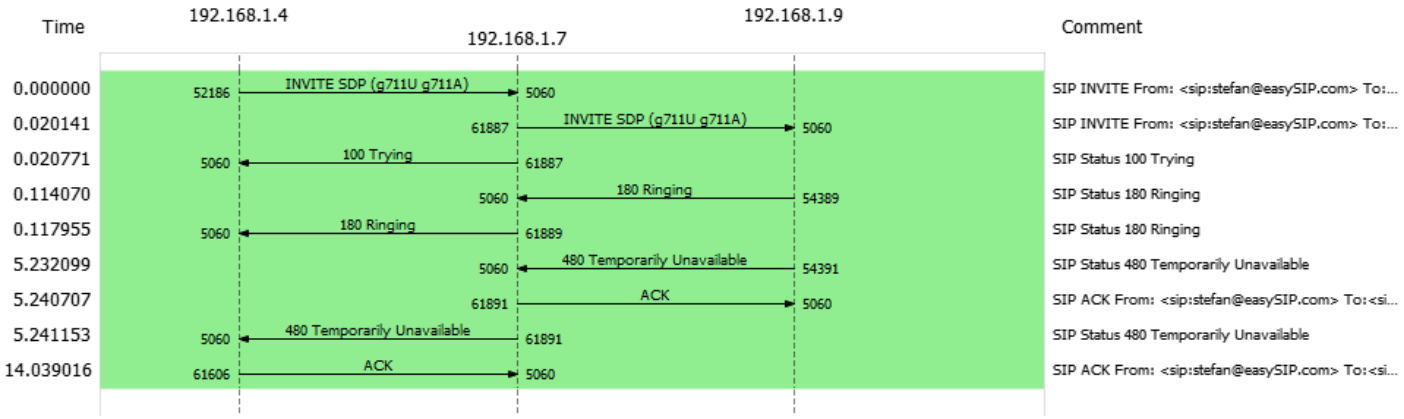
Clientul A recepționează răspunsul transmis de clientul B și trimite la rândul său un mesaj ACK către server pentru a confirma primirea răspunsului.

Totodată acesta afișează un mesaj prin care informează utilizatorul că persoana apelată a respins solicitarea.



În acest moment sesiunea ia sfârșit și din perspectiva clientului A și acesta este disponibil pentru a iniția sau pentru a participa în alte sesiuni.

Diagrama de mai jos prezintă grafic cazul descris.



Renunțarea la inițierea unui apel

După inițierea unui apel , utilizatorul care a realizat această acțiune (utilizatorul A) poate alege să termine apelul înainte ca utilizatorul apelat (utilizatorul B) să răspundă (înainte ca B să trimită mesajul SIP 200 OK). Această acțiune se realizează cu ajutorul metodei SIP CANCEL.

În momentul în care utilizatorul A alege să renunțe la inițierea apelului, acesta trimite un mesaj SIP CANCEL către server prin care cere renunțarea la stabilirea sesiunii. Mesajul SIP CANCEL se identifică prin prezența pe primul rând a metodei “CANCEL” urmată de adresa SIP a utilizatorului B.

```
CANCEL sip:alex@easySIP.com SIP/2.0
CSeq: 1 CANCEL
Via: SIP/2.0/UDP
192.168.1.4:5060;branch=32b64146-70ca-40ed-83f4-7a1ab84f04e5;rport
From:
<sip:stefan@easySIP.com>;tag=96cc3bff-017e-4ba1-9af3-6aebe0b0fe83
Call-ID: ab1e8ad1-d6f7-4acf-a7cf-60cad2db26b1
To: <sip:alex@easySIP.com>
Content-Length: 0
Max-Forwards: 70
```

Mesajul CANCEL este construit cu ajutorul următoarelor headere:

- Via: “UDP” – protocolul utilizat pentru a transporta mesajul, urmat de adresa IP a utilizatorului A și de portul la care acesta așteaptă să primească răspunsul. Parametrul “branch” are aceeași valoare pe care a avut-o și în cadrul metodei INVITE pe care utilizatorul A a transmis-o pentru a iniția sesiunea
- Max-Forwards: 70 - numărul maxim de retransmisii
- From: adresa SIP a utilizatorului A, urmată de proprietatea “tag” care își păstrează valoarea avută în mesajele SIP precedente ale acestei sesiuni
- To: - adresa SIP a utilizatorului B
- Call-ID: - acest header își păstrează aceeași valoare ca și celelalte mesaje SIP transmise în cadrul sesiunii până în acel moment

- CSeq: sequence number-ul este unul generat aleator, urmat de metoda “CANCEL”
- Content-Length: 0 – lungimea conținutului mesajului

Server-ul recepționează acest mesaj și îi răspunde utilizatorului A printr-un mesaj SIP 200 OK.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP
192.168.1.4:5060;branch=32b64146-70ca-40ed-83f4-7a1ab84f04e5;rport
From:
<sip:stefan@easySIP.com>;tag=96cc3bff-017e-4ba1-9af3-6aebe0b0fe83
To: <sip:alex@easySIP.com>
Call-ID: ab1e8ad1-d6f7-4acf-a7cf-60cad2db26b1
CSeq: 1 CANCEL
Contact: <sip:stefan@easySIP.com>
Content-Length: 0
```

Răspunsul 200 OK trimis de server este compus din următoarele header:

- Via: “UDP” – protocolul utilizat pentru a transporta mesajul, urmat de adresa IP a utilizatorului A și de portul la care acesta așteaptă să primească răspunsul; parametrul “branch” are aceeași valoare pe care a avut-o și în cadrul metodei INVITE pe care utilizatorul A a transmis-o pentru a inița sesiunea
- From: adresa SIP a utilizatorului A, urmată de proprietatea “tag” care își păstrează valoarea avută în mesajele SIP precendente ale acestei sesiuni
- To: - adresa SIP a utilizatorului B
- Call-ID: - acest header își păstrează aceeași valoare ca și celelalte mesaje SIP transmise în cadrul sesiunii până în acel moment
- CSeq: sequence number-ul este unul generat aleator, urmat de metoda “CANCEL”
- Content-Length: 0 – lungimea conținutului mesajului

După ce server-ul trimite răspunsul 200 OK către utilizatorul A, acesta transmite mesajul SIP CANCEL mai departe către utilizatorul B. Utilizatorul B recepționează mesajul primit de la server și îi răspunde, la rândul lui acestuia printr-un mesaj SIP 200 OK.

După aceasta utilizatorul B oprește notificarea primirii apelului și trimite către utilizatorul A, prin intermediul server-ului, mesajul SIP 487 Request Terminated.

```
SIP/2.0 487 Request Terminated
Via: SIP/2.0/UDP
192.168.1.9:5060;branch=666d8fe0-9674-4ea8-842d-130a892d877f;rport
From:
<sip:stefan@easySIP.com>;tag=96cc3bff-017e-4ba1-9af3-6aebe0b0fe83
To: <sip:alex@easySIP.com>
Call-ID: ab1e8ad1-d6f7-4acf-a7cf-60cad2db26b1
CSeq: 1 INVITE
Content-Length: 0
```

Mesajul SIP 487 Request Terminated este alcătuit din următoarele headere:

- Via: “UDP” – protocolul utilizat pentru a transporta mesajul, urmat de adresa IP a utilizatorului A și de portul la care aceasta așteaptă să primească răspunsul; parametrul “branch” are aceeași valoare pe care a avut-o și în cadrul metodei INVITE pe care utilizatorul A a transmis-o pentru a iniția sesiunea
- From: adresa SIP a utilizatorului A, urmată de proprietatea “tag” care își păstrează valoarea avută în mesajele SIP precendente ale acestei sesiuni
- To: - adresa SIP a utilizatorului B
- Call-ID: - acest header își păstrează aceeași valoare ca și celalalte mesaje SIP transmise în cadrul sesiunii până în acel moment
- CSeq: sequence number-ul este unul generat aleator, urmat de metoda “INVITE”
- Content-Length: 0 – lungimea conținutului mesajului

Server-ul primește acest mesaj și îi răspunde utilizatorului B printr-un mesaj SIP ACK pentru a-l anunța pe acesta că a recepționat mesajul trimis.

Prima linie a mesajului SIP ACK este alcătuită din metoda “ACK” urmată de adresa destinatarului, în acest caz adresa SIP a utilizatorului B.

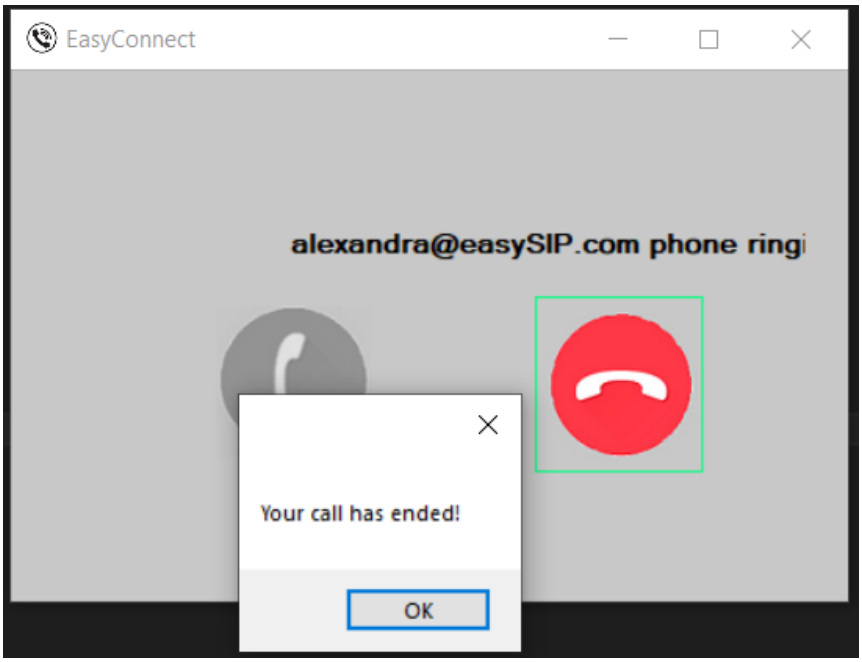
```
ACK sip:alex@easySIP.com SIP/2.0
CSeq: 1 ACK
Via: SIP/2.0/UDP 192.168.1.7:5060;branch=6dc3d99d-33a3-4b72-8c4f-
c82a2a391712;rport
From: <sip:stefan@easySIP.com>;tag=ce05c7c7-3d38-4458-9735-
d753f415f58f
Call-ID: ab1e8ad1-d6f7-4acf-a7cf-60cad2db26b1
To: <sip:alex@easySIP.com>
Content-Length: 0
Max-Forwards: 70
```

- Via: “UDP” – protocolul utilizat pentru a transporta mesajul, urmat de adresa IP a utilizatorului A și de portul la care acesta așteaptă să primească răspunsul, parametrul “branch” are aceeași valoare pe care a avut-o și în cadrul metodei INVITE pe care utilizatorul A a transmis-o pentru a iniția sesiunea
- Max-Forwards: 70 - numărul maxim de retransmisii
- From: adresa SIP a utilizatorului A, urmată de proprietatea “tag” care își păstrează valoarea avută în mesajele SIP precendente ale acestei sesiuni
- To: - adresa SIP a utilizatorului B
- Call-ID: - acest header își păstrează aceeași valoare ca și celalalte mesaje SIP transmise în cadrul sesiunii până în acel moment
- CSeq: sequence number-ul este unul generat aleator, urmat de metoda “CANCEL”
- Content-Length: 0 – lungimea conținutului mesajului

Fiind destinat utilizatorului A, mesajul SIP 487 este transmis de către server către acesta.

Utilizatorul A recepționează mesajul SIP 487 și trimite răspunsul SIP ACK către sever. În acest moment procesul de renunțare la această sesiune ia sfârșit și ambii utilizatori sunt liberi pentru a participa la alte sesiuni.

Utilizatorul A primește o notificare prin care i se transmite acest lucru.



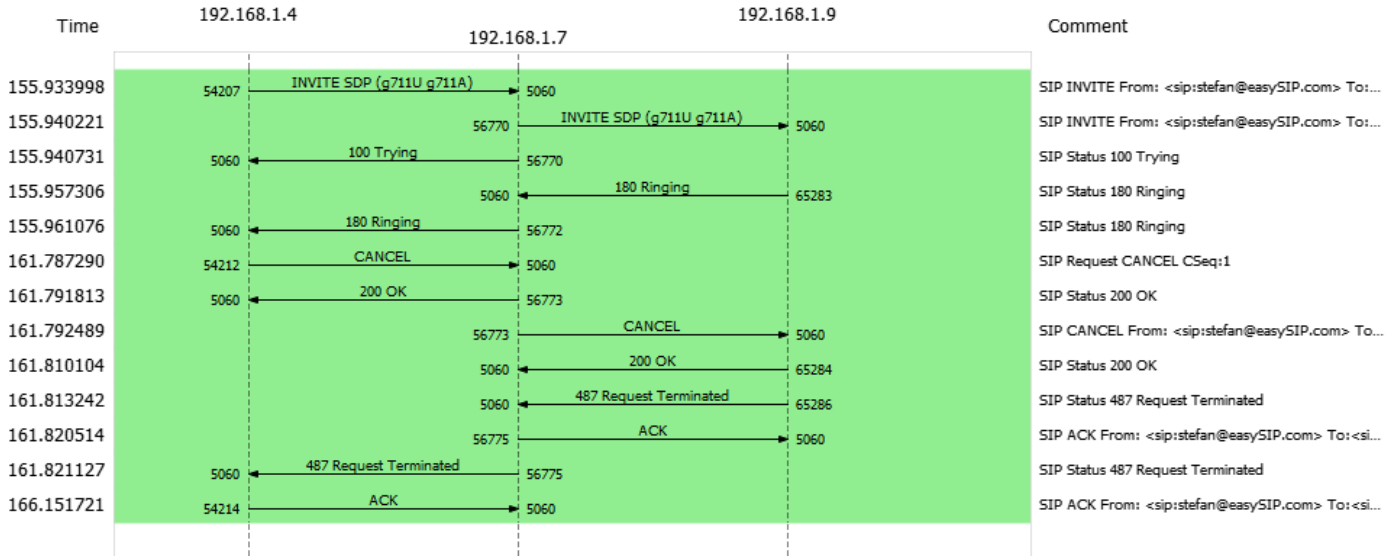


Diagrama acestui scenariu

Apelarea unui utilizator neinregistrat

În situația în care un utilizator (clientul A) inițiază un apel către un utilizator neînregistrat la server, stabilirea unei sesiuni nu este posibilă.

În momentul în care mesajul INVITE ajunge la proxy server acesta încearcă să găsească o înregistrare care să corespundă cu adresa SIP a utilizaturului căruia îi este destinat mesajul.

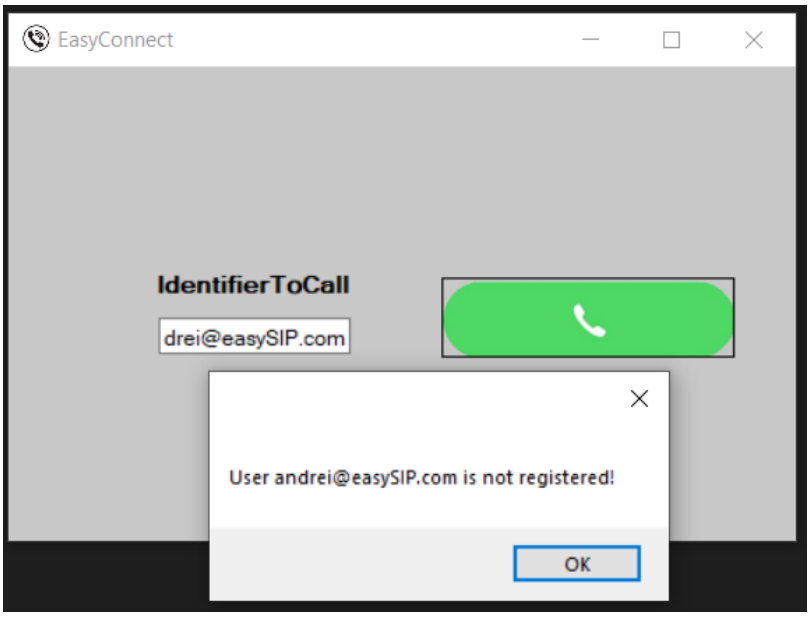
Dacă server-ul nu reușește să termine această acțiune cu succes, acesta generează un mesaj de eroare “404 User Not Found” pe care îl trimite ca răspuns clientului A.

```
SIP/2.0 404 Not Found
Via: SIP/2.0/UDP 192.168.1.7:5060;branch=b74e633c-9db3-4f72-
b2f0-3d42452ad256;rport
From:
<sip:stefan@easySIP.com>;tag=86874fe5-578e-4461-897a-37008c581dfa
To: <sip:andrei@easySIP.com>
Call-ID: 62c664b2-f937-4aff-8b32-6d53ab187b53
CSeq: 1 INVITE
Content-Length: 0
```

Mesajul “SIP 404 User Not Found” este alcătuit din următoarele headere:

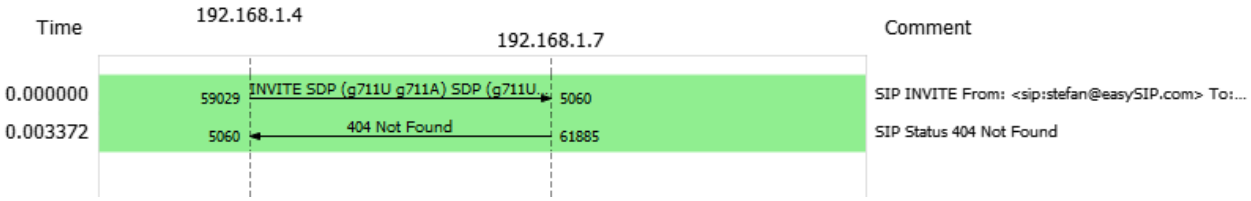
- Via: “UDP” – protocolul utilizat pentru a transporta mesajul, urmat de adresa IP a server-ului și de portul la care acesta așteaptă să primească răspunsul; parametrul “branch” are aceeași valoare pe care a avut-o și în cadrul metodei INVITE pe care utilizatorul A a transmis-o pentru a iniția sesiunea
- From: adresa SIP a utilizatorului A, urmată de proprietatea “tag” care își păstrează valoarea avută în mesajele SIP precendente ale acestei sesiuni
- To: - adresa SIP a utilizatorului B
- Call-ID: - acest header își păstrează aceeași valoare ca și celalalte mesaje SIP transmise în cadrul sesiunii până în acel moment
- CSeq: sequence number-ul este unul generat aleator, urmat de metoda “INVITE”
- Content-Length: 0 – lungimea conținutului mesajului

În momentul în care clientul A primește răspunsul de la server,acesta trimite un mesaj ACK pentru a confirma primirea răspunsului. Utilizatorul A primește o notificare prin care i se transmite că utilizatorul apelat nu este înregistrat.



Server-ul recepționează mesajul iar în acest moment sesiunea ia sfârșit.

Acest scenariu este prezentat grafic în diagrama de mai jos.



Protocolul SDP (Session Description Protocol)

Protocolul SDP reprezintă un format pentru a descrie comunicarea multimedia din cadrul unei sesiuni. De cele mai multe ori acesta este folosit pentru a completa aplicațiile de streaming media, cum sunt aplicatiile VoIP sau cele de video conferințe. SDP nu transmite date media ci este utilizat doar pentru a negocia și a stabili detaliile media legate de o sesiune (tipurile media, media codecs folosite, adrese IP și porturi, etc.).

SDP a fost inițial o parte componentă a SAP (Session Announcement Protocol), însă astăzi este folosit în principal împreună cu SIP și RTP în cadrul aplicațiilor VoIP.

Protocolul SDP descrie o sesiune printr-un grup de câmpuri în format text, fiecare câmp fiind situat pe un nou rând. Formatul este următorul:

<type>=<value><CR><LF>

Unde “<type>” reprezintă un singur caracter case-sensitive iar “<value>” reprezintă un text într-un format care depinde de valoarea “<type>”. Spațiile nu sunt permise imediat lângă semnul “=”.

Prezența unui mesaj SDP în corpul unui mesaj SIP este semnalizată prin valoarea “application/sdp” în dreptul câmpului “Content-Type”, urmată de lungimea acestuia în dreptul câmpului “Content-Length”.

Începutul unui mesaj SDP în corpul unui mesaj SIP este marcat prin prezența secvenței “0x0d0a0d0a”.

SDP este compus din trei secțiuni:

1. Descrierea sesiunii
2. Descrierea timpului

3. Descrierea media

Fiecare mesaj SDP poate să conțină mai multe secțiuni de descriere a timpului și mai multe secțiuni de descriere media însă doar o secțiune de descriere a sesiunii.

Descrierea sesiunii

- v= (versiunea de protocol)
- o= (inițiatorul sesiunii și identificatorul sesiunii: utilizator, id, numărul de versiune, adresa IP)
- s= (numele sesiunii)
- i=* (informații despre sesiune)
- u=* (URI – ul sesiunii)
- e=* (adresa e-mail)
- p=* (numărul de telefon)
- c=* (informațiile despre conexiune)
- b=* (niciuna sau mai multe informații despre lățimea de bandă)
- Una sau mai multe secțiuni de descriere a timpului
- z=* (informații legate despre time-zone)
- k=* (cheia de encripție)
- a=* (niciuna sau mai multe atribute de sesiune)
- Niciuna sau mai multe secțiuni de descriere media

Descrierea timpului

- t= (timpul scurs de când sesiunea este activă)
- r=* (niciunul sau mai multe informații despre de câte ori se repetă sesiunea)

Descrierea media

- m= (tipul media și adresa de transport)
- i=* (titlul media)
- c=* (informațiile de conexiune – opțional dacă acestea au fost prezente în secțiunea de descriere a sesiunii)
- b=* (niciuna sau mai multe informații despre lățimea de bandă)
- k=* (cheia de encripție)
- a=* (niciuna sau mai multe atribute media)

Câmpurile marcate cu '*' sunt opționale, toate celelalte fiind obligatorii. Toate câmpurile trebuie să fie prezente exact în ordinea prezentată mai sus pentru ca mesajul SDP să fie unul valid.

Câmpurile care iau valori text (ex: numele sesiunii, informațiile despre sesiune) pot conține orice valori cu excepția 0x00 (Nul), 0x0a (ASCII newline), 0x0d (ASCII carriage return). Secvența CRFL (0x0d0a) este folosită pentru a marca sfârșitul unui câmp, însă parser-ele ar trebui să fie suficient de flexibile astfel încât să accepte și intrări care se termină cu un singur rând nou (0x0a).

Un exemplu de mesaj SDP produs și utilizat de aplicație în cadrul unui mesaj SIP INVITE pentru a negocia parametrii media și informațiile de conexiune:

```
Content-Type: application/sdp
Content-Length: 143

v=0
o=stefan 123456 123456 IN IP4 192.168.1.6
s=-
c=IN IP4 192.168.1.6
t=0 0
m=audio 6666 RTP/AVP 0 8
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
```

v=0

câmpul v=0 reprezintă versiunea de protocol. În acest moment singura versiune este “0”.

o=stefan 123456 123456 IN IP4 192.168.1.6

câmpul o= reprezintă inițiatorul sesiunii și identificatorul sesiunii

“stefan” – indică numele utilizatorului.

“123456” – indică identificatorul sesiunii.

“123456” – reprezintă versiunea sesiunii. Aceasta trebuie să fie incrementată în momentul în care apar modificări în sesiune.

“IN” – este un text care reprezintă tipul de rețea. Valoarea “IN” reprezintă “Internet”

“IP4” – textul indică tipul de adresă folosită (adresă de tipul IPv4)

“192.168.1.6” – reprezintă adresa IP a emițătorului acestui mesaj

Tuplul compus din numele utilizatorului, identificatorul sesiunii, tipul de rețea, tipul de adresă și adresa IP trebuie să formeze un identificator unic global pentru sesiune.

s=-

câmpul s= reprezintă numele sesiunii.

Valoarea “-“ este o valoare standard ce denotă că sesiunea nu are un nume. Este folosit acest caracter datorită faptului că parametrul “s=” este unul obligatoriu iar caracterul imediat următor după “=” nu poate fi un spațiu gol conform regulilor SDP.

c=IN IP4 192.168.1.6

câmpul c= conține informațiile legate despre conexiune. Un mesaj SDP trebuie să conțină un câmp “c=” fie cel puțin în fiecare secțiune de descriere media, fie unul singur în secțiunea de descriere a sesiunii.

“IN” – este primul sub-câmp și este un text care reprezintă tipul de rețea. Valoarea “IN” reprezintă “Internet”

“IP4” – cel de-al doilea sub-câmp indică tipul de adresă folosită (adresă de tipul IPv4)

“192.168.1.6” – reprezintă adresa IP la care destinatarul acestui mesaj SDP trebuie să trimită pachetele media

t=0 0

câmpul t= indică timpul de început și timpul de sfârșit al unei sesiuni

Primul sub-câmp reprezintă timpul de început al sesiunii iar cel ce-al doilea timpul de sfârșit al acesteia. Dacă sub-câmpul de sfârșit al sesiunii are valoarea “0” atunci sesiunea nu este delimitată. Dacă și timpul de început al sesiunii are valoarea “0” atunci sesiunea este considerată permanentă.

m=audio 6666 RTP/AVP 0 8

câmpul m= marchează prezența unei secțiuni de descriere media. Aceasta se termină fie în momentul în care se întâlnește o nouă secțiune media, fie când mesajul SDP se termină.

Sub-câmpul “audio” reprezintă tipul de date media. Valorile posibile sunt: “audio”, “video”, “text”, “application”, și “message”.

6666 - este portul pe care emițătorul acestui mesaj așteaptă să primească datele media

RTP/AVP - indică protocolul prin care se dorește schimbul de date media

0 8 - reprezintă formatele media în care se așteaptă ca informația să fie transmisă. Semnificația acestor valori depinde de protocolul prin care se realizează comunicarea. Dacă protocolul este “RTP/AVP” atunci formatele media reprezintă valorile “payload type”.

a=rtpmap:0 PCMU/8000

“a=rtpmap” indică prezența unui atribut SDP care descrie formatul media semnalizat în secțiunea de descriere media. Acesta face maparea dintre un payload type RTP și un nume de encodare care reprezintă formatul media. De asemenea oferă și informații legate de rata de eșantionare a fluxului de date. Aceste informații sunt indispensabile în special când este vorba de media codecs dinamici (Payload type > 96).

În acest caz este vorba de codec-ul static PCMU cu o rată de eșantionare de 8000 Hz.

a=rtpmap:8 PCMA/8000

Similar “a=rtpmap:0 PCMU/8000”, în acest caz fiind vorba de codec-ul static PCMA cu o rată de eșantionare de 8000 Hz.

Real-time Transport Protocol

Real-time Transport Protocol este un protocol de transmitere a fluxurilor de date de tip media (sunete, imagini, video) în timp real în rețelele de internet. Acest protocol este utilizat cu precădere în cadrul sistemelor de comunicații sau al sistemelor care se ocupă cu transmiterea de conținut de divertisment.

Protocolul RTP a fost dezvoltat de către Audio-Video Transport Working Group din cadrul IETF (Internet Engineering Task Force) și a fost publicat prima dată în anul 1996. Acesta reprezintă fundația, din punct de vedere tehnic, pe baza căreia a luat naștere tehnologia VoIP (Voice over IP), iar în acest context protocolul RTP este adesea folosit împreună cu alte protocoale precum SIP (Session Inițiation Protocol) pentru a stabili o sesiune și SDP (Session Description Protocol) pentru a stabili parametrii sesiunii.

De obicei protocolul RTP este implementat peste protocolul UDP datorită caracteristicilor acestuia cum ar fi viteza crescută de transmitere. Datorită acestui lucru RTP pune la dispoziție informații care să permită detecția pachetelor lipsă, sau a celor recepționate neordonat prin intermediul câmpului numărului de secvență (sequence number).

RTP a fost dezvoltat în asa fel încât să permită utilizarea a multiple formate multimedia, astfel că acest protocol este indepent de formatul media folosit. Pentru fiecare clasă multimedia (audio/video), protocolul RTP definește un profil și valorile parametrului “payload format” asociat. Acest parametru este folosit pentru a descrie modul în care informația transportată este encodată în cadrul pachetului.

RTP este implementat la nivel de aplicație ci nu la nivelul pachetului de protocoale a sistemului de operare.

Fiecare pachet RTP este compus dintr-un header urmat de datele propriu-zise.

Header-ul unui pachet RTP

Fiecare header al unui pachet RTP are o lungime minimă de 12 octeți. După header, poate fi prezentă, opțional, o extensie a acestuia, iar apoi restul pachetului conține datele propriu-zise codate conform informatiilor prezente în header.

Un header de pachet RTP este compus din următoarele câmpuri:

- **Versiunea** (2 biți): reprezintă versiunea RTP folosită. În prezent aceasta este versiunea "2"
- **P (padding)** (1 bit): acest flag este folosit pentru a indica faptul că pachetul este completat cu informație suplimentară (ex: în cazul în care informația transmisă trebuie să aibă o lungime anume)
- **X (extension)** (1 bit): indică prezența header-ului de extensie, opțional, aflat la sfârșitul header-ului
- **CC (CSRC count)** (4 biți): reprezintă numărul de identificatori CSRC (opționali) care urmează după câmpul SSRC
- **M (marker)** (1 bit): este rezervat pentru nivelul aplicației. Dacă este setat, înseamnă că datele din acest pachet au o semnificație specială pentru aplicație
- **PT (payload type)** (7 biți): indică formatul după care datele sunt encodate. Valorile statice pe care acest parametru le poate lua sunt conforme tabelului (Tabel 1) iar cele dinamice sunt asignate dinamic luând valori din intervalul 96-127
- **Numărul de secvență (sequence number)** (2 octeți) – acest câmp este incrementat cu fiecare pachet RTP transmis și are rolul de a ajuta destinatarul să detecteze probleme cum ar fi pierderea pachetelor sau sosirea lor neordonat. Valoarea inițială a acestui parametru trebuie să fie una aleasă aleator
- **Timestamp** (4 octeți): este folosit de către destinatar pentru a putea reda informația primită la viteza potrivită

- **SSRC (Synchronization source identifier)** (4 octeți): este folosit pentru a identifica expeditorul fluxului de date. Acest câmp va avea o valoare unică pentru fiecare participant în cadrul unei sesiuni
- **CSRC (Contributing source identifiers)** (4 octeți fiecare): reprezintă identitatea celorlalți participanți care au contribuit la crearea fluxului de date în cazul în care acesta este unul generat din mai multe surse
- **Header extension** (opțional): primii 4 octeți reprezintă un identificator specific profilului (2 octeți) și lungimea (2 octeți), care reprezintă lungimea header-ului de extensie exprimată în unități de 4 octeți. Header-ul de extensie urmează după aceste valori.

Offsets	Octet	0								1								2								3							
Octet	Bit [a]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version			P	X		CC		M				PT				Sequence number															
4	32	Timestamp																															
8	64	SSRC identifier																															
12	96	CSRC identifiers ...																															
12+4×CC	96+32×CC	Profile-specific extension header ID																Extension header length															
16+4×CC	128+32×CC	Extension header ...																															

(Header-ul unui pachet RTP)

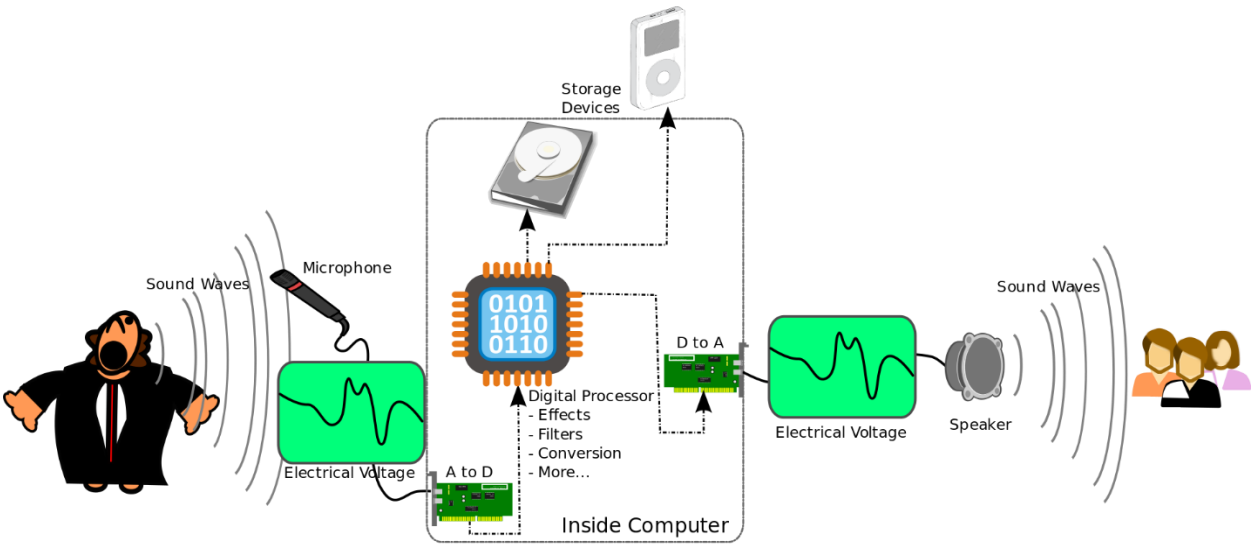
PT	Encoding Name	Audio/Video (A/V)	Clock Rate (Hz)	Channels
0	PCMU	A	8000	1
1	Reserved			
2	Reserved			
3	GSM	A	8000	1
4	G723	A	8000	1
5	DVI4	A	8000	1
6	DVI4	A	16000	1
7	LPC	A	8000	1
8	PCMA	A	8000	1
9	G722	A	8000	1
10	L16	A	44100	2
11	L16	A	44100	1
12	QCELP	A	8000	1
13	CN	A	8000	1
14	MPA	A	90000	
15	G728	A	8000	1
16	DVI4	A	11025	1
17	DVI4	A	22050	1
18	G729	A	8000	1
19	Reserved	A		
20	Unassigned	A		
21	Unassigned	A		
22	Unassigned	A		
23	Unassigned	A		
24	Unassigned	V		
25	CelB	V	90000	
26	JPEG	V	90000	
27	Unassigned	V		
28	nv	V	90000	
29	Unassigned	V		
30	Unassigned	V		
31	H261	V	90000	
32	MPV	V	90000	
33	MP2T	AV	90000	
34	H263	V	90000	
35-71	Unassigned	?		
72-76	Reserved for RTCP conflict avoidance			
77-95	Unassigned	?		
96-127	dynamic	?		

(Tabel 1 – payload types)

Media Codecs

Sunetul digital este o reprezentare digitală a sunetului din natură care ne permite înregistrarea, stocarea, reproducerea sau editarea acestuia. Anumite caracteristici ale undelor sonore analoge, cum ar fi frecvența sau amplitudinea, sunt codificate sub forma unor informații binare astfel încât sistemele informatice să poate înțelege informația.

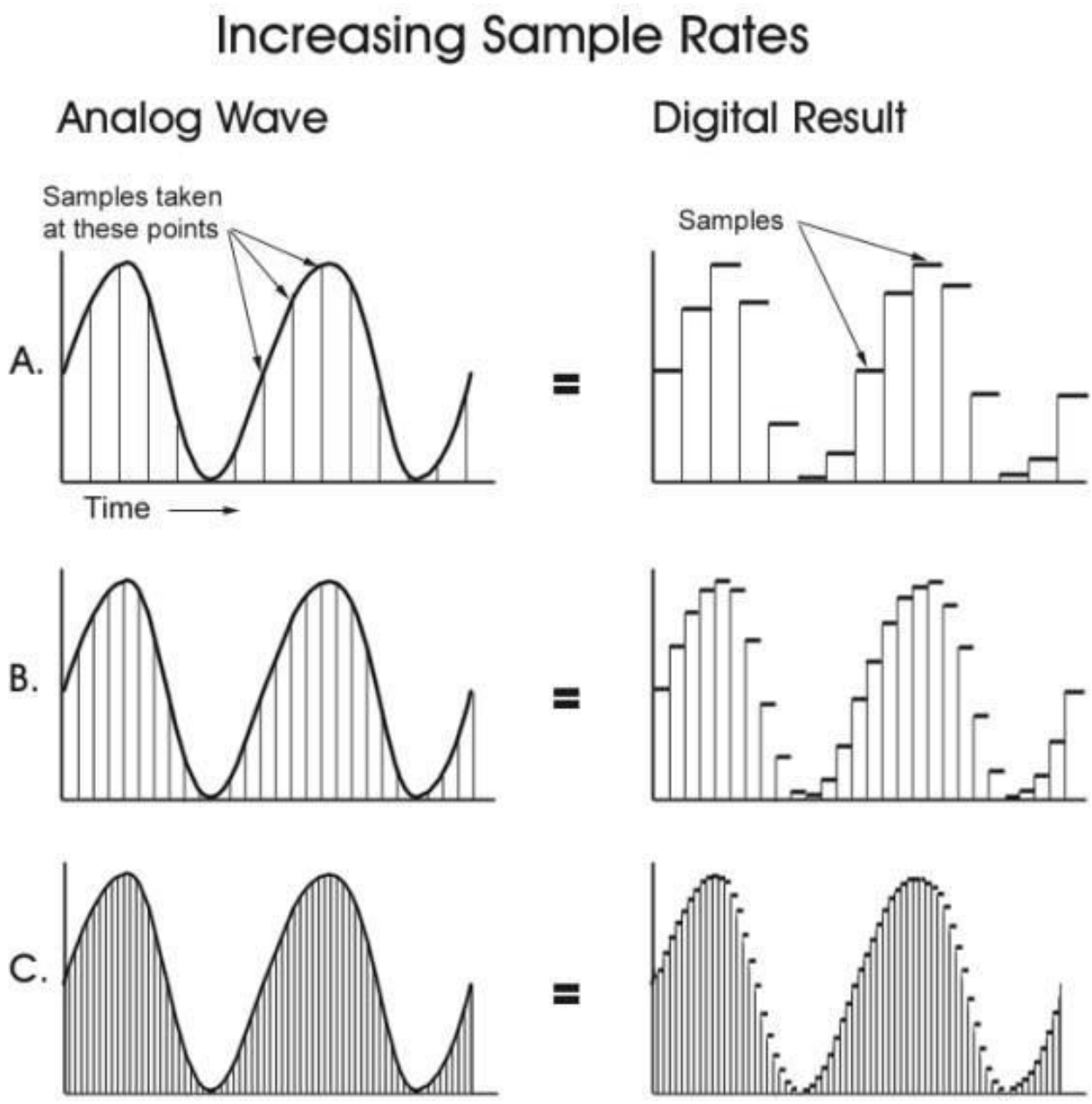
În sistemele audio digitale, sunetul reprezentat de un semnal electric analog este convertit în semnale digitale, prin intermediul unui convertor analog-la-digital (ADC – analog to digital converter), de obicei folosind metoda pulse-code modulation (PCM). În mod similar, în momentul în care dorim să redăm un fișier audio digital, sau un flux de date primit din rețea, un convertor digital-la-analog (DAC – digital to analog converter) este folosit pentru a reda sunetul.



Metoda pulse-code modulation este o metodă standard de reprezentare a semnalelor sonore analoge în semnale digitale, folosită de majoritatea calculatoarelor, CD-urilor, industriei de telecomunicații sau a altor aplicații audio.

Aceasta presupune ca undele sonore să fie convertite în informație binară printr-o serie de măsuratori. O măsurare a undei sonore, realizată la un anumit punct în timp, înregistrează

amplitudinea semnalului. Un sistem realizează mii de astfel de măsurători pe secundă, iar rata la care aceste măsurători sunt făcute se numește rată de eșantionare sau frecvență. Dacă putem realiza aceste măsurători cu o frecvență suficient de crescută și putem atribui amplitudinii înregistrate valori dintr-un interval suficient de mare, atunci putem reconstrui undele sonore într-un format digital cu suficientă acuratețe.

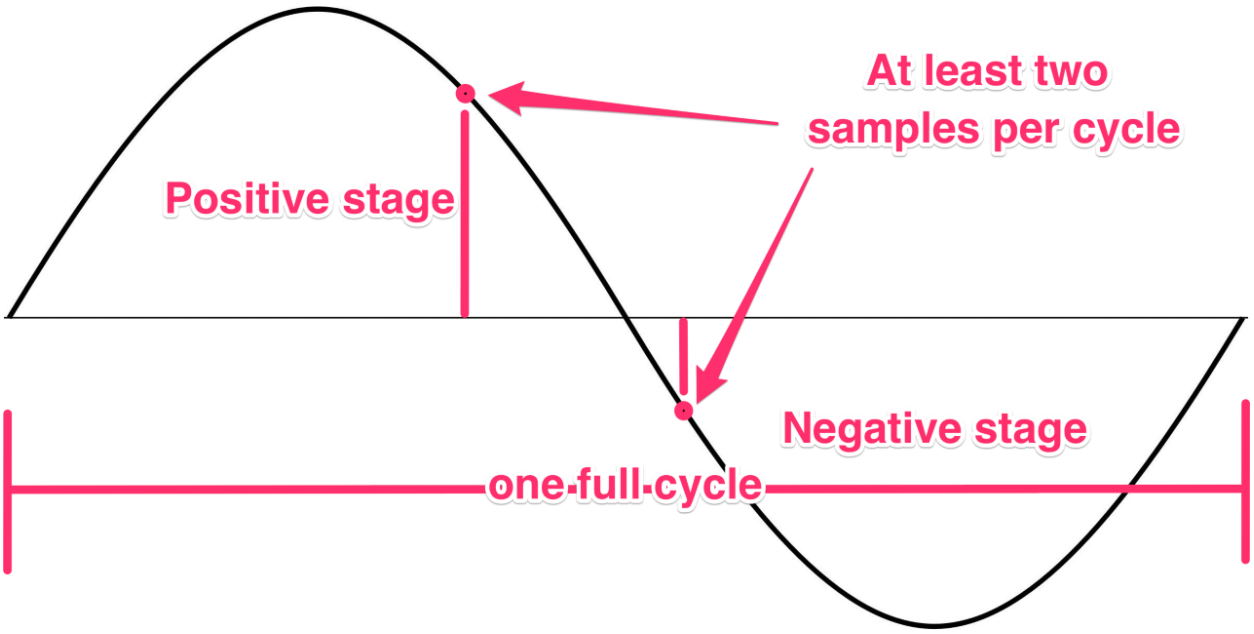


Frecvența (rata de eșantionare -sample rate-)

Numărul de măsuratori pe care un sistem îl realizează într-un interval de o secundă este denumit rată de eșantionare. Rata de eșantionare este măsurată în Hertzi (Hz).

Toate sunetele din natură pot fi reprezentate sub forma unor unde sinusoidale. Pentru a măsura frecvența unei unde sinusoidale este necesar să se definească un ciclu.

Un ciclu complet al oricărei unde de tip sinusoidal conține atât o etapă pozitivă cât și una negativă. Pentru a afla lungimea acestui ciclu (lungimea de undă- cu ajutorul căreia putem deduce frecvența acesteia prin intermediul formulei $\boldsymbol{c = f * \lambda}$) este necesar să putem detecta ambele etape. Așadar, pentru fiecare ciclu al undei sunt necesare două măsuratori (câte una pentru fiecare etapă).



Din acest lucru rezultă faptul că pentru a putea înregistra frecvența undei sonore în mod corect, este necesar ca rata de eșantionare să fie egală cu cel puțin de două ori frecvența undei sonore. Această rată poartă denumirea de “rata Nquist”. În mod invers, un sistem poate să înregistreze

și să redea în mod corect doar frecvențe egale cu jumătâte din rata de eșantionare, această limită purtând denumirea de “frecvența Nquist”.

Semnalele cu o valoare peste frecvența Nquist nu sunt înregistrate corect de către convertoarele de tip analog-la-digital, producând distorsiuni. Pentru a preveni aceste distorsiuni convertoarele de tip analog-la-digital, sunt adesea precedate de filtre care elimină frecvențele cu valori peste prag, înainte ca acestea să ajungă în convertor.

Cea mai des întâlnită rată de eșantionare este de 44,1 kHz (44100 Hertzi sau 44100 măsurători pe secundă). Această valoare nu este aleasă aleator, ci este aleasă datorită faptului că majoritatea oamenilor pot percepe sunetele aflate în intervalul 20 Hz – 20kHz. Deși o rată de eșantionare de 40 kHz ar fi fost suficientă, pentru a preveni nevoia de a folosi filtrele de eliminare a frecvențelor cu valori peste frecvența Nquist, s-a optat pentru lărgirea acestui interval de la 20 kHz la 22.05 kHz.

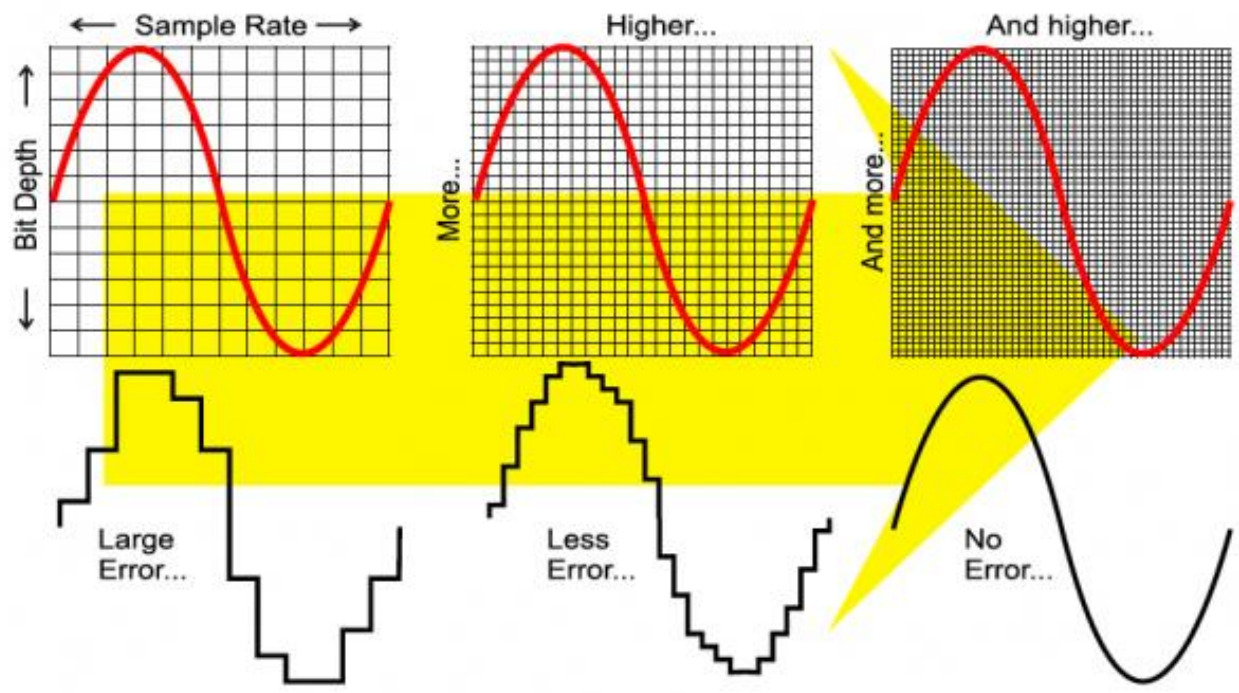
În baza acestui fapt și a ratei Nquist, rata de eșantionare de 44,1 k Hz reprezintă rata de eșantionare standard în domeniul digital audio.

Adâncimea de biți (-bit depth-)

Semnalele audio din natură pot avea amplitutidini cu un număr infinit de valori. Pentru a putea atribui o valoare acestor amplitudini este necesar să definim un interval în care amplitudinile pot lua valori. Extremitățile acestui interval sunt date de numărul de biți pe care îl alegem pentru a stoca fiecare semnal analogic. Numărul de biți utilizat poartă denumirea de adâncime de biți. Cu cât adâncimea de biți este mai mare, cu atât există mai multe posibile valori care pot fi atribuite amplitudinii. Acest lucru rezultă într-o reprezentare mai fidelă a realității, deci o calitate superioară. Această îmbunătățire a calitătii înseamnă totodată, și un volum mai mare de date, iar raportul optim între calitate și dimensiune trebuie ales în funcție de specificul situației.

Cele mai comune adâncimi de biți folosite sunt de 8-biți, 16-biți, 32-biți, acestea permițând 256, 65 536, respectiv 4 294 967 296 valori posibile.

Crescând atât adâncimea de biți cât și rata de eșantionare se obține o calitate mai bună a sunetului după cum se poate vedea mai jos.



Codecs

Un codec reprezintă un program software sau o componentă hardware al cărui scop este de a coda și de a decoda informația. Cuvântul reprezintă un acronim din limba engleză (encoding/decoding). În general, termenul este utilizat atunci când se face referire la procesul de codare și/sau decodare al unui flux de date audio sau video.

În funcție de natura informației pe care o procesează un codec poate fi:

- Audio – ex: G.711 , AMR, EVS
- Video – ex: MPEG-4 , H.264

Fiecare codec utilizează o metodă diferită de a procesa informația și prezintă anumite avantaje (viteză crescută, rată de compresie crescută, etc.).

Unul dintre cele mai cunoscute codec-uri audio utilizate în aplicațiile VoIP este codecul G.711.

G.711 a fost introdus de către International Communication Union (ITU) în 1972 cu scopul de a fi folosit în aplicațiile de telefonie. Acesta procesează semnalele audio în intervalul 300-3400 Hz (narrowband) și le măsoară cu o rată de eșantionare de 8000 Hz. Adâncimea de biți folosită este de 8 biti.

Codecul G.711 realizează conversia unei mostre cu o adâncime de biți 16 într-o mostră cu adâncimea de biți 8. Acest lucru duce la o rată de compresie de 1:2, fapt ce asigură un bit-rate de 64kbit/s (8 kHz rată de eșantionare × 8 biți pe mostră). Acest lucru este necesar pentru a limita intervalul în care o mostră poate lua valori, deoarece în domeniul telecomunicațiilor în timp real, viteza de transmitere a datelor joacă un rol esențial în calitatea serviciilor.

Codecul G.711 vine în două variante:

1. G.711 A-law
2. G.711 μ-law

G.711 A-law (PCMA)

Algoritmul A-law este folosit în toate țările din lume, mai puțin SUA și Japonia.

Pentru orice x , ecuația algoritmului A-law pentru codare este următoarea:

$$F(x) = \operatorname{sgn}(x) \begin{cases} \frac{A|x|}{1+\ln(A)}, & |x| < \frac{1}{A} \\ \frac{1+\ln(A|x|)}{1+\ln(A)}, & \frac{1}{A} \leq |x| \leq 1, \end{cases}$$

unde $\boldsymbol{A}$ reprezintă parametrul de compresie. În Europa $\boldsymbol{A = 87.6}$

Pentru decodare algoritmul A-law folosește funcția inversă:

$$F^{-1}(y) = \operatorname{sgn}(y) \begin{cases} \frac{|y|(1+\ln(A))}{A}, & |y| < \frac{1}{1+\ln(A)} \\ \frac{\exp(|y|(1+\ln(A))-1)}{A}, & \frac{1}{1+\ln(A)} \leq |y| < 1. \end{cases}$$

G.711 μ-law (PCMU)

Algoritmul μ-law este folosit în SUA și Japonia. Fiind o versiune a codec-ului G.711, acesta are rolul de a micșora intervalul în care o mostră poate lua valori, atunci când acesta realizează codarea, și de a-l mări atunci cand realizează decodarea, după cum urmează:

Pentru codare, pentru orice $\boldsymbol{x}$ **,** ecuația algoritmului μ-law este următoarea:

$$F(x) = \operatorname{sgn}(x) \frac{\ln(1+\mu|x|)}{\ln(1+\mu)} \qquad -1 \leq x \leq 1$$

unde $\mu = \boldsymbol{255}$.

Iar pentru decodare algoritmul μ-law folosește ecuația inversă:

$$F^{-1}(y) = \operatorname{sgn}(y)(1/\mu)((1+\mu)^{|y|} - 1) \quad -1 \leq y \leq 1$$

În cadrul header-ului pachetelor RTP (Real-Time Transport Protocol), valoarile parametrului ***Payload Type*** pentru codecul G.711 sunt următoarele:

- G.711 A-law: ***Payload Type*** – 8
- G.711 μ-law: ***Payload Type*** – 0

În cadrul aplicației prezentate sunetul este înregistrat și redat într-un format PCM cu o adâncime de biți 16 și rata de eșantionare 8000 Hz. În momentul transmiterii în rețea fluxul de date este codat prin intermediul unuia dintre cei doi codecs suportați de către aplicație (G.711 PCMA sau G.711 PCMU), urmând ca atunci când este recepționat, să fie decodat în formatul PCM (adâncime de biți 16 și rata de eșantionare 8000 Hz) pentru a fi redat.

Wireshark

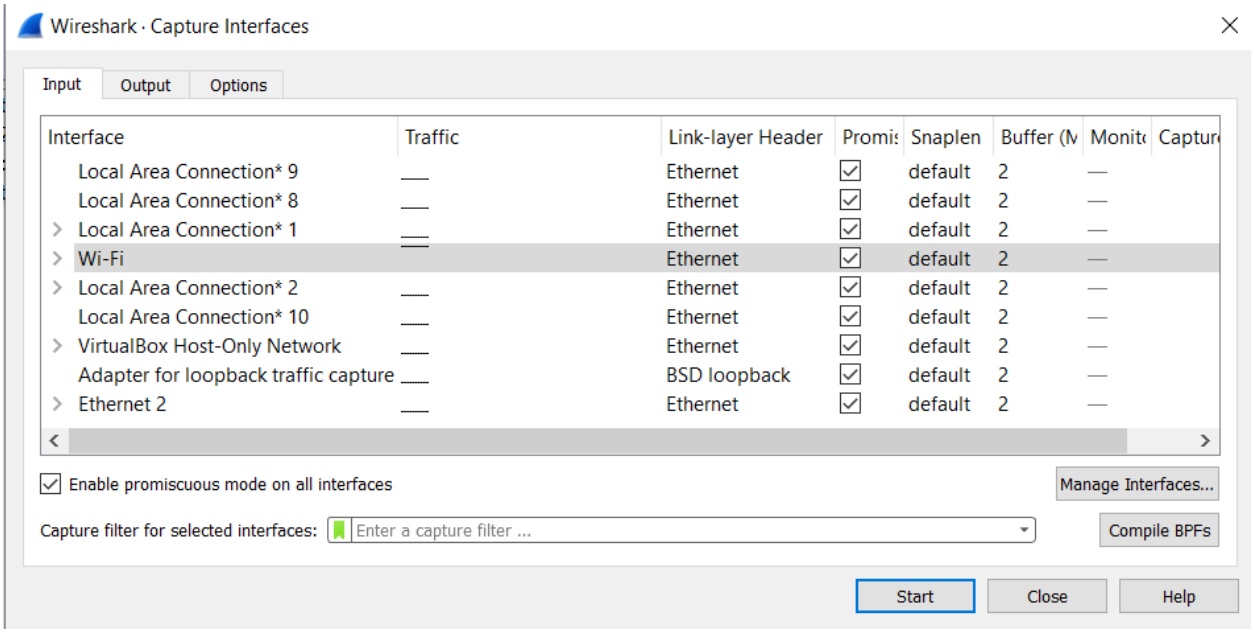
Wireshark este un tool gratuit, care oferă posibilitatea utilizatorilor de a analiza traficul dintr-o anumită rețea. Acesta este cel mai utilizat program destinat acestui scop și este open source (https://gitlab.com/wireshark/wireshark.git).

Fiind un program cross-platform acesta rulează atât pe Windows cât și pe alte sisteme de operare precum macOS sau Linux și este folosit în majoritatea organizațiilor private și guvernamentale.

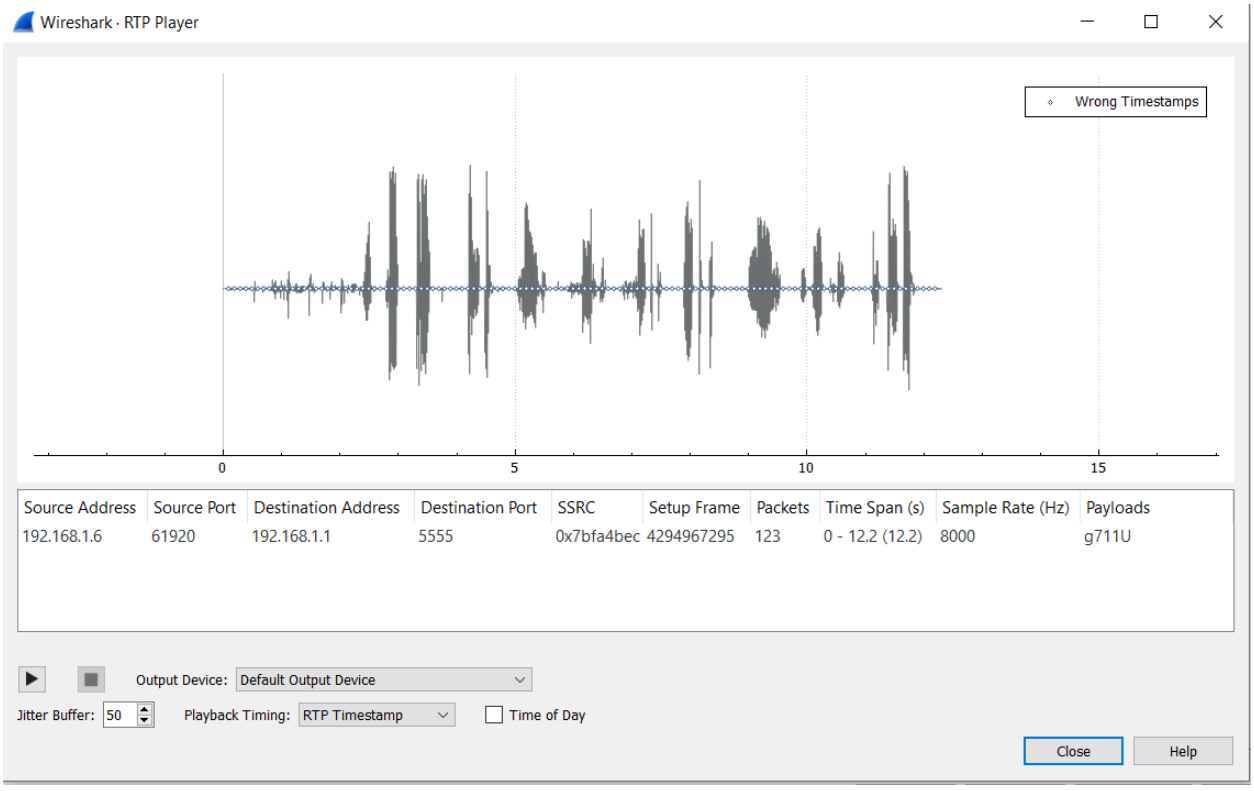
Deoarece este un program atât de popular, cunoștințele minime de utilizare ale acestui program nu ar trebui să lipsească din bagajul tehnic al niciunui profesionist care își desfașoară activitatea în domeniul rețelelor și/sau al protocoalelor .

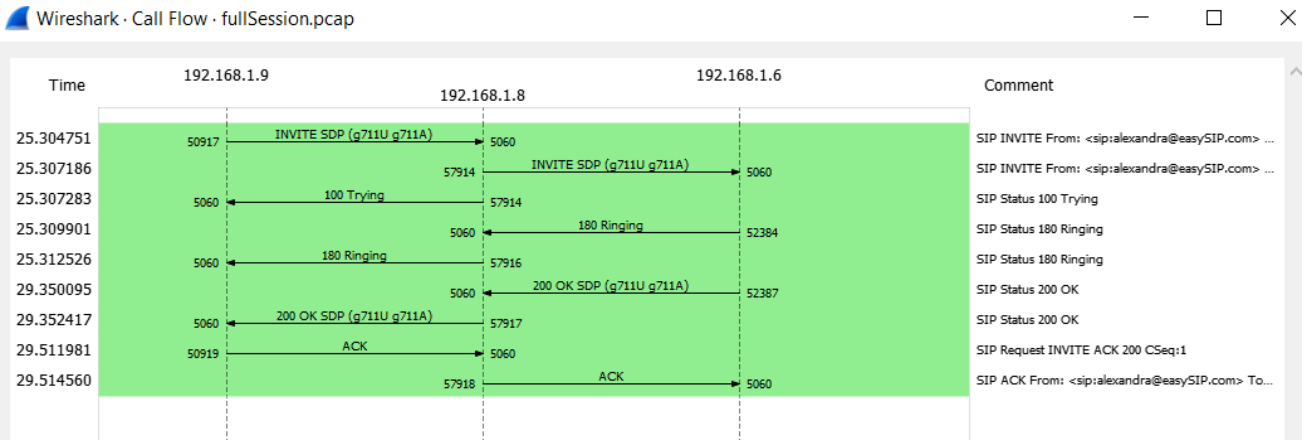
Programul este scris în limbajele C și C++ iar pentru interfața grafică este folosit Qt. Pentru a filtra și a captura pachete, acesta folosește librăriile libpcap și npcap.

Wireshark este capabil atât să deschidă fișiere care conțin pachete deja capturate în formate .pcap , .pcapng, .cap, .snoop, cât și să captureze traficul în timp real, dintr-o anumită rețea. În anumite condiții, dacă interfața este setată în modul promiscuu, Wireshark poate să captureze tot traficul din rețeaua respectivă chiar dacă acesta nu este îndreptat către mașina de pe care Wireshark rulează sau nu este transmis de pe aceasta.



Apelurile VoIP înregistrate pot fi detectate și, dacă codec-ul folosit pentru a coda pachetele în cadrul sesiunii este unul pe care Wireshark îl recunoaște, acesta poate să redea conținutul audio.

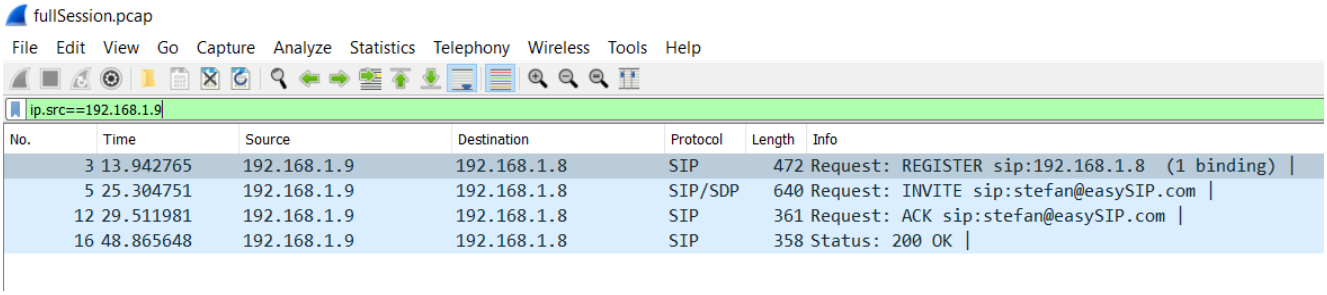




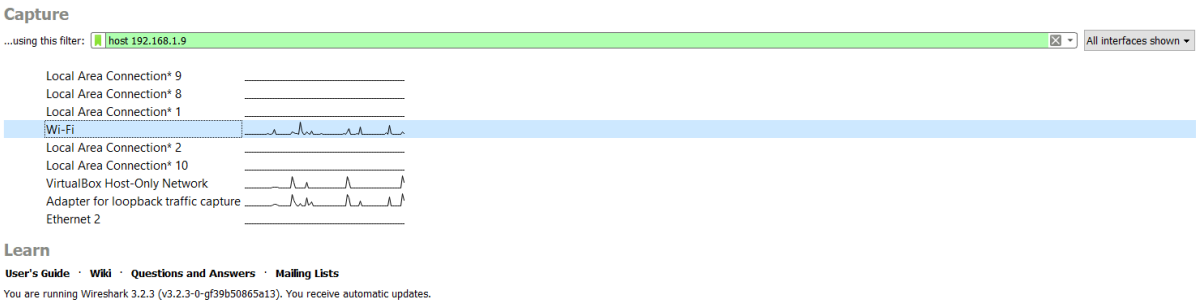
(pachete produse de către aplicație capturate cu Wireshark)

Filtrele pe care Wireshark le are la dispoziție pot fi folosite în 2 moduri:

1. Astfel încât să captureze toate pachetele dintr-o rețea și să le afișeze doar pe acelea care corespund filtrului



2. Astfel încât să captureze doar pachetele care corespund filtrului



Acest lucru este util deoarece oferă foarte multă flexibilitate utilizatorului.

Wireshark pune la dispoziția utilizatorilor capacitatea de a interpreta zeci de protocoale în detaliu și posibilitatea ca aceștia să își scrie propriile librării de interpretare a protocoalelor (numiți disectori) care să fie mai apoi folosiți în cadrul Wireshark pentru a extinde funcționalitățile acestuia. Această capabilitate este foarte importantă deoarece în anumite industrii, cum este cea a telecomunicațiilor, unele protocoale sunt private, iar disectorii standard puși la dispoziție de Wireshark nu sunt suficienți.

Acești disectori pot fi scrișii atât în limbajul C cât și în limbajul LUA. Deși disectorii scrișii în C au o viteză de procesare a pachetelor superioară celor scrișii în LUA, aceștia din urmă sunt mai ușor de folosit și de scris, fiind compatibili cu toate versiunile de Wireshark, pe când cei scrișii în C sunt compatibili numai cu o anumită versiune de Wireshark, cea împreună cu care au fost compilați. Ținând cont de frecvența crescută cu care comunitatea open-source lansează noi versiuni pentru Wireshark, această limitare este una de care trebuie să se țină cont.

LUA

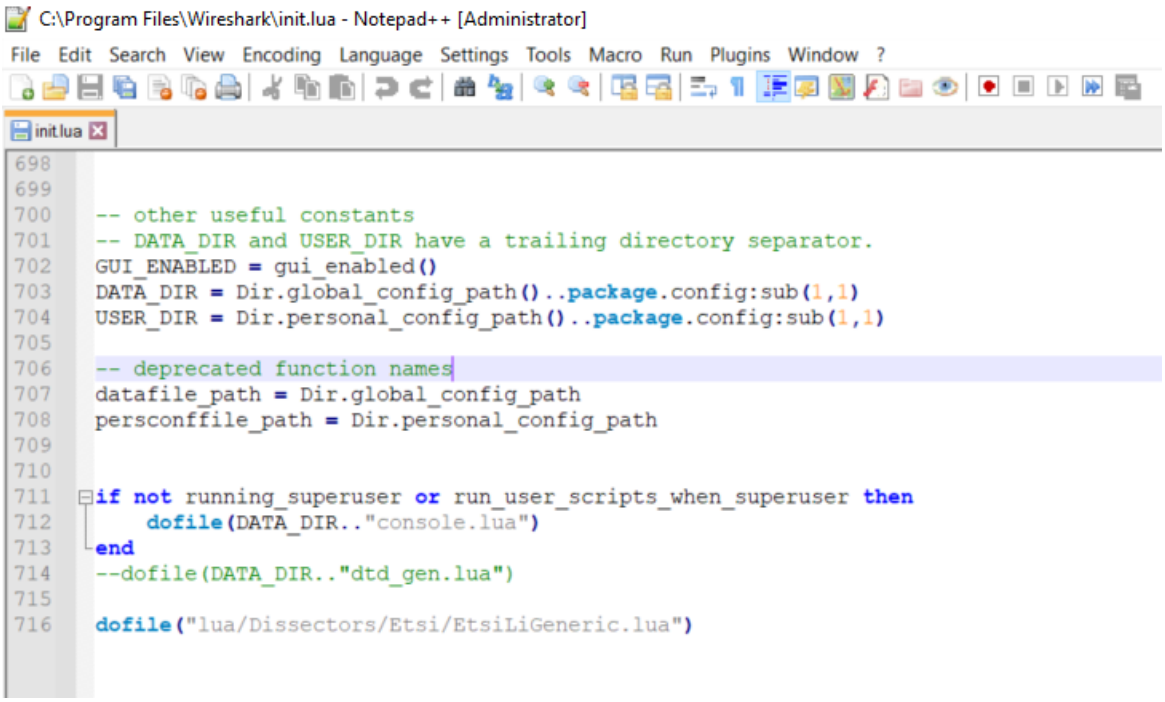
LUA este un limbaj de scripting care oferă suport pentru programarea procedurală, programarea orientată pe obiecte sau programarea funcțională. LUA este cross-platform și are o sintaxă ușor de folosit.

```lua
function Hello(n)
    for i = 1, n do
        print("Hello World!")
    end
end
```

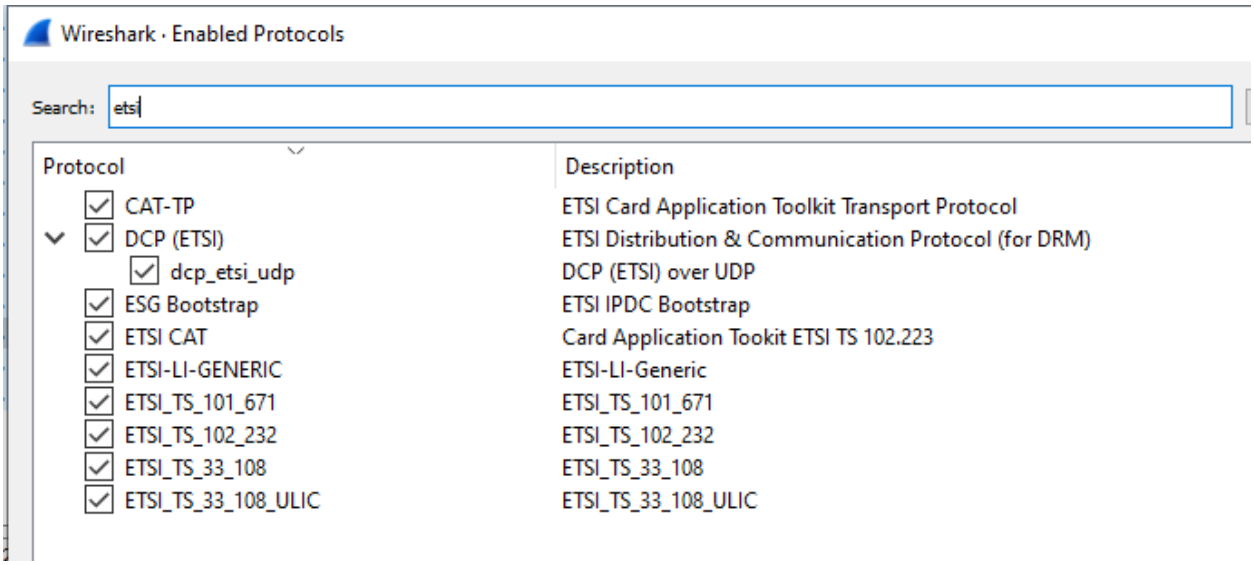
(exemplu funcție LUA)

În general LUA este folosit pentru a extinde funcționalitățile unui program, în cazul de față Wireshark.

Wireshark conține un interpretator integrat pentru LUA iar de fiecare dată când pornește acesta încearcă să încarce un fișier numit “init.lua” , aflat în directorul în care Wireshark a fost instalat. Pentru a extinde funcționalitățile Wireshark cu propriul nostru disector tot ceea ce trebuie să facem, după ce ne-am scris disectorul în LUA conform instrucțiunilor (https://www.wireshark.org/docs/wsdg_html_chunked/wslua_dissector_example.html) este să modificăm fișierul “init.lua” astfel încât acesta să încarce fișierele noastre.



Dacă fișierele noastre au fost încărcate cu succes, în momentul în care pornim aplicația, disectorii noștri trebuie să fie vizibili în fereastra “Enabled protocols” (Analyze -> Enabled protocols / Ctrl+Shift+E).



Considerații finale

Conform cu cele prezentate putem conchide că procesul prin care se stabilește cu succes o sesiune între doi clienți, prin intermediul tehnologiei VoIP este unul complex iar protocoalele care stau la baza acestei tehnologii sunt unele de bază în industria comunicațiilor și au fost rafinate de-a lungul timpului odată cu dezvoltarea la un nivel din ce în ce mai avansat a acestor sisteme.

Odată cu accelerarea procesului de globalizare și adoptarea la o scară cât mai largă a metodologiilor work-from-home și online-schooling, sistemele de comunicare în timp real vor trebui să țină pasul cu nevoia crescută de a asigura conectarea utilizatorilor.

În sprijinul acestor nevoi vine și tehnologia 5G care, prin vitezele superioare și capacitatea crescută de utilizatori pe care le suportă va asigura baza din punct de vedere logistic pentru ca această dezvoltare accelerată să continue.

Limitări

Din punct de vedere al securității aplicația nu oferă protecție, datele transmise nefiind criptate. Din această cauză mesajele sunt vizibile pentru orice utilizator conectat în aceeași rețea în care este folosită aplicația.

Perspective de dezvoltare

Există mai multe direcții în care aplicația poate fi dezvoltata în continuare:

1. Prin adăugarea de noi funcționalități
2. Prin îmbunătățirea funcționalităților existente

Printre noile funcționalități ce pot fi implementate amintim:

- apeluri video
- call forwarding (unconditional, on busy, on no-answer, on unreachable)
- hold/retrieve
- conferință
- mesaje text și media

Functionalitățile actuale pot fi îmbunătățite prin:

- adăugarea opțiunii de criptare a mesajelor
- adăugarea unui istoric al apelurilor și a unei agende de contacte
- implementarea mai multor media codecs

Bibliografie și webografie

- “Packet Guide to Voice over IP: A system administrator's guide to VoIP technologies 1st Edition”, Bruce Hartpence
- https://tools.ietf.org/html/rfc3261
- https://www.iana.org/
- https://en.wikipedia.org/wiki/
- https://markheath.net/category/naudio
- https://github.com/naudio/NAudio
- https://www.wireshark.org/
- https://blog.wildix.com/
- https://datatracker.ietf.org/doc/html/rfc3665
- https://www.izotope.com/en/learn/digital-audio-basics-sample-rate-and-bit-depth.html