CONMUTACIÓN DE LLAMADAS DE VOZ IP ENTRE REDES 3G Y WIFI A TRAVÉS DE UN SERVIDOR SIP

NICOLÁS ALBERTO PATIÑO HERNÁNDEZ
JUAN PABLO ROBLES ALARCÓN

PONTIFICIA UNIVERSIDAD JAVERIANA
BOGOTA
2013

CONMUTACIÓN DE LLAMADAS DE VOZ IP ENTRE REDES 3G Y WIFI A TRAVÉS DE UN SERVIDOR SIP

NICOLÁS ALBERTO PATIÑO HERNÁNDEZ
JUAN PABLO ROBLES ALARCÓN

TRABAJO DE GRADO
PROYECTO DE APLICACIÓN PRÁCTICA

Director
Ing. GUSTAVO ADOLFO RAMÍREZ ESPINOSA
Profesor de planta

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULDAD DE INGENIERÍA
INGENIERÍA ELECTRÓNICA
BOGOTA
2013

CONTENIDO

TABLA DE FIGURAS

1. INTRODUCCIÓN

Según el boletín trimestral de las TIC en Colombia, correspondiente al cuarto trimestre del año 2012, el número de suscriptores a internet de banda ancha supera los seis millones de abonados, de los cuales se estima que el 36.9% de los accesos se realiza a través de redes móviles de tercera generación (3G), mientras que el 62.5% representa conexiones de banda ancha fijas. [1]

Las redes móviles 3G permiten la transmisión tanto de datos como de voz y los abonados a la telefonía celular dependen de la cobertura de red de cada operador al cual están suscritos y de los costos de comunicación establecidos por cada proveedor de servicios y verificados por la Comisión de Regulación de Telecomunicaciones (CRC).

Gracias a la conectividad de voz sobre IP nace una alternativa en la realización de llamadas de voz, aprovechando las características de redes como lo son 3G y WiFi, brindando la posibilidad de establecer conexiones de voz no solo por medio de redes de telefonía, sino también a través de redes de datos, incluyendo las redes celulares.

Las redes 3G permiten la movilidad de los usuarios por la conmutación entre celdas, siendo dependientes del radio de cobertura de cada una de las bases de transmisión (BTS[1]), con la ventaja que la distancia entre una y otra base permite mantener la conexión mientras el usuario está en movimiento. Las redes WiFi por su parte tienen un alcance limitado al radio de cobertura de la red inalámbrica propagada por dispositivos tales como Routers y Access Points, pero a diferencia de las redes 3G y las tarifas por consumo ofrecidas por los proveedores, las redes WiFi en la mayoría de los casos brindan conectividad y transferencia de datos ilimitada con costos de tarificación fijos.

Ahora bien, conociendo ambas redes de comunicaciones podemos obtener conexión de voz tanto por las redes celulares 3G como por las redes WiFi haciendo uso de servicios VoIP. Gracias a esto es posible conmutar una llamada previamente establecida por una u otra red sin perder la conectividad de la misma, pero este servicio no está disponible para los usuarios a través de los operadores celulares en Colombia, desaprovechando la capacidad de la interoperabilidad entre redes.

El Protocolo de Inicio de Sesiones (SIP, por sus siglas en inglés) nace como un protocolo de comunicaciones IP, encargado del control y señalización de mensajes a través de la red, desarrollado por la IETF[2] y definido en el RFC 3261 como el estándar para la iniciación, modificación y terminación de sesiones interactivas de usuario donde intervienen elementos multimedia como video, voz, mensajería instantánea, juegos en línea y realidad virtual [2].

SIP es un protocolo de control (señalización) bajo el modelo Cliente – Servidor, que opera de la forma 'request – response' y basa su funcionamiento en el registro de los usuarios en el servidor, identificándose de la forma usuario@dominio.

Gracias a la funcionalidad que brinda el protocolo SIP de realizar el registro de los usuarios en un servidor, se abre la posibilidad de utilizar ventajas como la modificación de las características de la comunicación mientras ésta se encuentra en progreso [3]. Esto nos permite por ejemplo, poder cambiar el identificador de los usuarios en el servidor, en este caso la dirección IP con la cual se han registrado, para de esta forma brindar la capacidad a un dispositivo de tener movilidad entre redes 3G y WiFi (Vertical Handover[3]).

[1] Base Transceiver Station
[2] Internet Engineering Task Force
[3] Vertical Handover: Se refiere a un nodo de red que modifica el tipo de conexión utilizada (redes de diferentes características) para acceder a una infraestructura soportada, usualmente para permitir movilidad.

El principal objetivo de este trabajo de grado es lograr realizar la conmutación de una llamada de voz entre redes 3G y WiFi, de forma automática y evitando perder la conexión durante el proceso.

En este documento se describe el proceso de implementación del servidor SIP para el registro y establecimiento de llamadas de voz y la implementación de la aplicación móvil que permite realizar una llamada entre dos usuarios y modificar los parámetros de la sesión manteniendo activa la comunicación. Se detalla la solución al problema planteado en el proyecto del presente trabajo de grado, las pruebas realizadas y los resultados obtenidos.

2. MARCO TEÓRICO

SIP es un protocolo de la capa de aplicación del stack del modelo TCP/IP (Figura 1), encargado del control y señalización de los mensajes a través de la red, utilizado para el establecimiento, modificación y finalización de sesiones, entre ellas VoIP. Permite movilidad por medio del registro y localización (en la red) de los usuarios, la descripción y negociación de las características de la sesión y de las capacidades de los participantes.

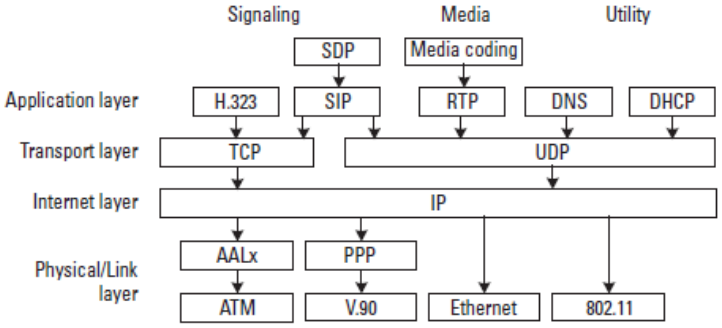


Figura 1. Stack del Protocolo Multimedia de Internet. Tomado de (1).

Según la referencia bibliográfica [4] sus principales características son:

- Basado en Texto
- Similar a HTTP o SMTP (petición/respuesta, encabezados, códigos de estado…)
- Uso de URIs (con esquemas sip, sips y tel)
- Métodos básicos: REGISTER, INVITE, ACK, BYE, CANCEL, OPTIONS
- Los mensajes se agrupan en Transacciones y llamadas.
- El cuerpo de los mensajes contiene descripciones de sesiones multimedia (SDP)
- Códigos de respuesta similares a los de HTTP (Ejemplo: 200 OK)
- Localización basada en DNS.

Para efectos del desarrollo de este proyecto, es fundamental entender el proceso de intercambio de mensajes para el establecimiento y terminación de una llamada mediante un servidor SIP, cómo se explica a continuación (ejemplos tomados de [5]).

2.1. Registro de Usuario

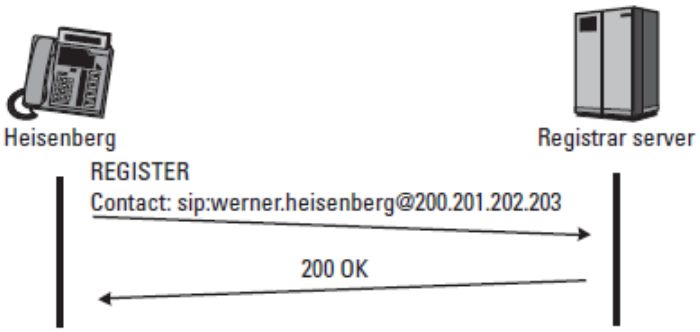


Figura 2. Ejemplo Registro SIP. Tomado de (2).

En la Figura 2 se muestra un ejemplo relacionado con el envío del mensaje de registro a un tipo de servidor SIP conocido como 'registrar server'. En el ejemplo el usuario Heisenberg envía el mensaje REGISTER al servidor, el cual lo recibe y utiliza la información contenida en la solicitud para actualizar la base de datos usada por los proxis para enrutar peticiones SIP.

La dirección SIP URI[4] de Heisenberg está contenida en el encabezado TO del mensaje de registro y el campo CONTACT URI, representa el dispositivo actual mediante un identificador único en una base de datos y la dirección IP del usuario. Cuando el servidor proxy recibe un mensaje INVITE (llamada entrante) dirigido a Heisenberg y accede a la base de datos, la solicitud será redirigida a la CONTACT URI del dispositivo actualmente registrado.

El mensaje REGISTER enviado por Heisenberg tiene la forma:

```
REGISTER sip:registrar.munich.de SIP/2.0
Via: SIP/2.0/UDP 200.201.202.203:5060;branch=z9hG4bKus19
Max-Forwards: 70
To: Werner Heisenberg <sip:werner.heisenberg@munich.de>
From: Werner Heisenberg <sip:werner.heisenberg@munich.de>
 ;tag=3431
Call-ID: 23@200.201.202.203
CSeq: 1 REGISTER
Contact: sip:werner.heisenberg@200.201.202.203
Content-Length: 0
```

Una vez recibido el mensaje de registro enviado por el cliente, el servidor confirma la operación exitosa mediante el envío de un mensaje de respuesta 200 OK. En el mensaje se incluye un TAG o identificador de la comunicación entre el servidor y el cliente y tiene la siguiente forma:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 200.201.202.203:5060;branch=z9hG4bKus19
To: Werner Heisenberg <sip:werner.heisenberg@munich.de>;tag=8771
From: Werner Heisenberg <sip:werner.heisenberg@munich.de>
 ;tag=3431
Call-ID: 23@200.201.202.203
CSeq: 1 REGISTER
Contact: <sip:werner.heisenberg@munich.de>;expires=3600
Content-Length: 0
```

[4] Uniform Resource Identifier: Cadena de caracteres corta que identifica inequívocamente un recurso (servicio, página, dispositivo, documento, dirección de correo electrónico, enciclopedia, etc.)

2.2. Establecimiento de Llamada a través de un Servidor Proxy

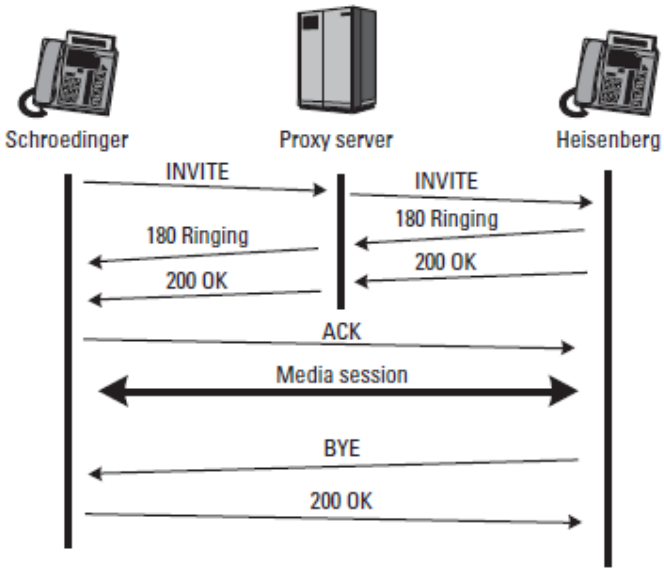


Figura 3. Ejemplo de llamada SIP con un servidor proxy. Tomado de (3).

Como se mencionó en el ejemplo anterior, el parámetro más importante para el contacto y ubicación del usuario en la red es la SIP URI. Esta es el nombre que se resuelve a una dirección IP gracias a un servidor SIP proxy y a consultas de DNS en el momento de la llamada (si es necesario).

En el flujo de mensajes de la Figura 3, Schroedinger llama a Heisenberg a través de un servidor SIP proxy. El servidor no inicia ni termina sesiones, simplemente actúa como un intermediario en la comunicación, encargándose de recibir y redirigir los mensajes.

El mensaje de INVITE que envía Schroedinger al servidor tiene la siguiente forma:

```
INVITE sip:werner.heisenberg@munich.de SIP/2.0
Via: SIP/2.0/UDP 100.101.102.103:5060;branch=z9hG4bKmp17a
Max-Forwards: 70
To: Heisenberg <sip:werner.heisenberg@munich.de>
From: E. Schroedinger <sip:schroed5244@aol.com>;tag=42
Call-ID: 10@100.101.102.103
CSeq: 1 INVITE
Subject: Where are you exactly?
Contact: <sip:schroed5244@pc33.aol.com>
Content-Type: application/sdp
Content-Length: 159

v=0
o=schroed5244 2890844526 2890844526 IN IP4 100.101.102.103
s=Phone Call
t=0 0
c=IN IP4 100.101.102.103
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

El mensaje INVITE es un tipo de mensaje SIP/SDP, ya que este no solo contiene la información de señalización de los mensajes SIP, sino que también contiene la información multimedia del usuario que genera la solicitud. Esta es enviada mediante el Protocolo de Descripción de Sesiones (SDP, por sus siglas en inglés), indispensable para el establecimiento de llamada una vez esta sea aceptada por la contraparte. El protocolo SIP encapsula la información del protocolo SDP dentro de su estructura y contiene información como codecs de audio/video soportados por el usuario y la dirección IP de contacto mediante la cual recibirá los datos multimedia (en este caso voz) y que será utilizada por el protocolo RTP[5] encargado de la transferencia de los datos multimedia una vez establecida la llamada.

Dado que Schroedinger no conoce la ubicación de Heisenberg el servidor proxy es el encargado de enrutar el mensaje INVITE. El proxy consulta la SIP URI (sip:werner.heisenberg@munich.de) en su base de datos con el fin de localizar la dirección IP de Heisenberg y envía el mensaje a la dirección IP resuelta agregando un campo VIA al mensaje, el cual identifica al servidor como quien redirige el mensaje INVITE.

```
INVITE sip:werner.heisenberg@200.201.202.203 SIP/2.0
Via: SIP/2.0/UDP proxy.munich.de:5060;branch=z9hG4bK83842.1
Via: SIP/2.0/UDP 100.101.102.103:5060;branch=z9hG4bKmp17a
Max-Forwards: 69
To: Heisenberg <sip:werner.heisenberg@munich.de>
From: E. Schroedinger <sip:schroed5244@aol.com>;tag=42
Call-ID: 10@100.101.102.103
CSeq: 1 INVITE
Contact: <sip:schroed5244@pc33.aol.com>
Content-Type: application/sdp
Content-Length: 159

v=0
o=schroed5244 2890844526 2890844526 IN IP4 100.101.102.103
s=Phone Call
c=IN IP4 100.101.102.103
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Los mensajes 180 RINGING son mensajes de respuesta que sirven como confirmación a la recepción de los mensajes de solicitud y se generan cuando el mensaje INVITE llega a la contraparte quien lo envía al servidor para que este lo dirija nuevamente al usuario que generó la solicitud. De esta forma, Schroedinger para el ejemplo, quien originó el primer mensaje en la comunicación, sabe que su solicitud fue recibida exitosamente. Cuando la llamada es aceptada por Heisenberg, éste envía un mensaje 200 OK al servidor:

[5] Real-Time Protocol

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP proxy.munich.de:5060;branch=z9hG4bK83842.1
 ;received=100.101.102.105
Via: SIP/2.0/UDP 100.101.102.103:5060;branch=z9hG4bKmp17a
To: Heisenberg <sip:werner.heisenberg@munich.de>;tag=314159
From: E. Schroedinger <sip:schroed5244@aol.com>;tag=42
Call-ID: 10@100.101.102.103
CSeq: 1 INVITE
Contact: <sip:werner.heisenberg@200.201.202.203>
Content-Type: application/sdp
Content-Length: 159

v=0
o=heisenberg 2890844526 2890844526 IN IP4 200.201.202.203
s=Phone Call
c=IN IP4 200.201.202.203
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

El mensaje 200 OK al igual que el mensaje INVITE es de tipo SIP/SDP, ya que mediante éste mensaje el usuario que recibe la solicitud de llamada envía su información multimedia, dando a conocer al igual que el usuario que origina la llamada, los codecs de audio/video soportados y su dirección IP de contacto, entre otros.

El servidor elimina el primer campo de VIA antes de reenviar el mensaje 200 OK a Schroedinger.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 100.101.102.103:5060;branch=z9hG4bKmp17a
To: Heisenberg <sip:werner.heisenberg@munich.de>;tag=314159
From: E. Schroedinger <sip:schroed5244@aol.com>;tag=42
Call-ID: 10@100.101.102.103
CSeq: 1 INVITE
Contact: <sip:werner.heisenberg@200.201.202.203>
Content-Type: application/sdp
Content-Length: 159

v=0
o=heisenberg 2890844526 2890844526 IN IP4 200.201.202.203
c=IN IP4 200.201.202.203
t=0 0
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

La presencia del campo CONTACT en el mensaje 200 OK con la SIP URI de Heisenberg, permite que Schroedinger conozca su ubicación y envíe un ACK directamente sin tener que pasar por el Proxy.

```
ACK sip:werner.heisenberg@200.201.202.203 SIP/2.0
Via: SIP/2.0/UDP 100.101.102.103:5060;branch=z9hG4bKka42
Max-Forwards: 70
To: Heisenberg <sip:werner.heisenberg@munich.de>;tag=314159
From: E. Schroedinger <sip:schroed5244@aol.com>;tag=42
Call-ID: 10@100.101.102.103
CSeq: 1 ACK
Content-Length: 0
```

Mediante el envío directo del mensaje entre los dos agentes usuarios clientes, se evidencia que el servidor SIP no está realmente presente durante la realización de la llamada, éste simplemente facilita la localización y el contacto entre las dos terminales, excluyéndose del camino de señalización una vez no agrega ningún valor en el intercambio de mensajes.

En la Figura 3 se aprecia también que como el servidor no participa en la llamada, la sesión de media se realiza punto-a-punto, usando el protocolo RTP para la trasmisión de los datos.

La sesión de media termina cuando Heisenberg envía un mensaje BYE:

```
BYE sip:schroed5244@pc33.aol.com SIP/2.0
Via: SIP/2.0/UDP 200.201.202.203:5060;branch=z9hG4bK4332
Max-Forwards: 70
To: E. Schroedinger <sip:schroed5244@aol.com>;tag=42
From: Heisenberg <sip:werner.heisenberg@munich.de>;tag=314159
Call-ID: 10@100.101.102.103
CSeq: 2000 BYE
Content-Length: 0
```

Schroedinger debe confirmar la finalización de la llamada, por lo tanto envía un mensaje 200 OK en respuesta al mensaje BYE. Nótese que los campos TO y FROM se mantienen igual que en el mensaje BYE, dado que fue Heisenberg quien originó la solicitud.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 200.201.202.203:5060;branch=z9hG4bK4332
To: E. Schroedinger <sip:schroed5244@aol.com>;tag=42
From: Heisenberg <sip:werner.heisenberg@munich.de>;tag=314159
Call-ID: 10@100.101.102.103
CSeq: 2000 BYE
Content-Length: 0
```

3. ESPECIFICACIONES

Tanto el servidor SIP como la aplicación utilizada en el dispositivo móvil están implementados basados en los parámetros establecidos por la RFC 3261 –'SIP: Session Initiation Protocol', con el fin de asegurar la uniformidad en el envío y recepción de los mensajes y la integridad, conformación y estructura de los mismos. Con esto se asegura que los mensajes que sean enviados por un agente usuario (servidor o cliente) sean "entendidos" por el agente usuario receptor y que el protocolo SIP sea utilizado basado en el estándar internacional, evitando modificaciones o alteraciones que pudiesen presentarse en software de código abierto.

3.1. Servidor SIP

El servidor SIP permite la creación y modificación de los usuarios y de sus características. En éste se asigna el número de extensión que utilizará cada usuario y su contraseña.

El SIP Register se encarga de la autorización y el registro de los clientes y el SIP Proxy realiza el enrutamiento de todos los mensajes hacia su destino, controlando la señalización durante el establecimiento de la llamada. El servidor solo genera mensajes de respuesta a las peticiones que realizan los agentes cliente (ej. 200 OK, 100 Trying).

Debido al esquema de red diseñado, mostrado más adelante en la sección 3.3, se implementa en el servidor SIP el RTP Proxy, el cual realiza el enrutamiento de todos los paquetes multimedia (voz) una vez se establece la conexión de una llamada. El RTP Proxy conoce la dirección de contacto de los usuarios mediante la información contenida en los mensajes INVITE y 200 OK que envía la aplicación durante el establecimiento de la comunicación.

3.1.1. ELASTIX®

Elastix es una distribución de software libre de un servidor de comunicaciones unificadas, el cual integra en un solo paquete:

- VoIP PBX
- Fax
- Mensajería Instantánea
- Email

Elastix implementa gran parte de su funcionalidad sobre cuatro programas de software muy importantes como son Asterisk, Hylafax, Postfix y Openfire. Estos programas brindan las funciones de PBX, Fax, Email y Mensajería Instantánea, respectivamente. El sistema operativo se basa en la popular distribución de Linux orientada a servidores llamada CentOS [6].

Asterisk es uno de los componentes más importantes de Elastix y quien provee la mayoría de las características telefónicas de la distro[6]. Dada la importancia de Asterisk para el sistema y específicamente para la solución del presente trabajo de grado, se destacan en la siguiente sección sus características más relevantes.

[6] Una distribución Linux (coloquialmente llamada distro) es una distribución de software basada en el núcleo Linux que incluye determinados paquetes de software para satisfacer las necesidades de un grupo específico de usuarios

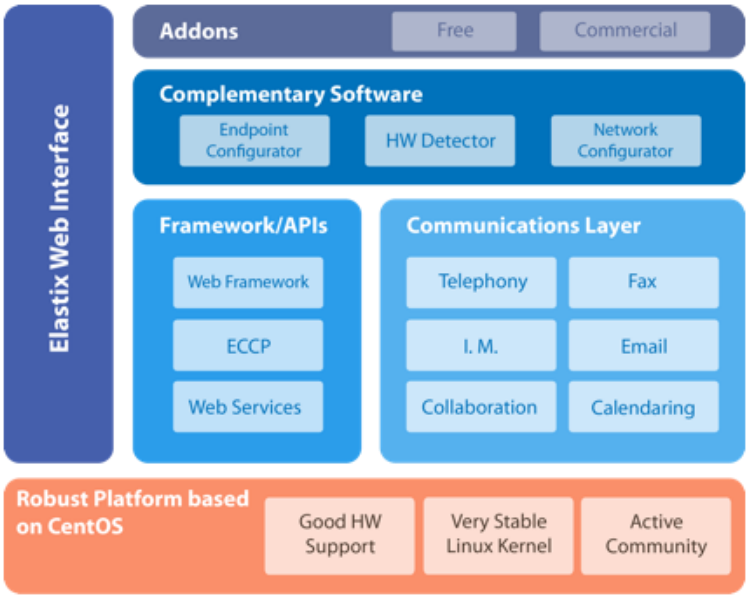


Figura 4. Esquema general de los componentes de Elastix. Tomada de (4).

En la Figura 4 se muestra la estructura de Elastix, sus componentes y su relación entre sí. Como se puede apreciar, la interface Web de Elastix, recoge todas las funcionalidades que éste provee y brinda un entorno común para la administración de los servicios y la integración de los mismos.

Los principales programas que conforman el núcleo de Elastix en su versión 2.4.0 y que brindan sus principales funcionalidades son:

- Asterisk (V. 1.8.20)
- vTigerCRM® and SugarCRM® – Sistemas de CRM.
- A2Billing® – Plataforma de tarjetas de llamadas y facturación para Asterisk.
- Flash Operator Panel – Consola de Operadora vía Web.
- Hylafax® – Sistemas de faxes.
- Openfire® – Servidor de mensajería instantánea.
- FreePBX® (V. 2.8.1-16) – Interface de administración Web de Asterisk y componente esencial en Elastix.
- Sistemas de Reportes – Información detallada de las operaciones de la PBX.
- OSLEC – Cancelador de Eco basado en Software.
- Postfix® – Servidor de correos.
- CentOS (V. 5.9) – Sistema Operativo.

3.1.2.ASTERISK®

Asterisk es un programa de software libre bajo licencia GPL que proporciona funcionalidades de central telefónica PBX, capaz de convertir un computador ordinario en un servidor de comunicaciones.

Asterisk es un programa rico en características y funcionalidades, como se detalla en su página web [7], dentro de las cuales podemos destacar:

- Autenticación
- Registro de llamadas detallado
- Desvío de llamadas
- Monitoreo de llamadas

- Enrutamiento de llamadas (DID y ANI)
- Transferencia de llamadas
- Identificador de llamadas
- Integración de base de datos
- Marcado por nombre
- Canalización
- VoIP Gateways
- Correo de Voz

3.2. Aplicación Móvil

La aplicación móvil actúa como cliente para el servidor; ésta permite la creación de cuentas de usuario donde es posible configurar los datos de conexión al servidor y la información de autorización (usuario y contraseña).

La aplicación móvil es la encargada de realizar todo el proceso de registro del usuario en el servidor y es quien genera todos los mensajes de petición (ej. REGISTER, INVITE, etc.).

Para efectos de la conmutación de la llamada, es la aplicación quien se encarga de comunicar al servidor la modificación de los parámetros de la sesión, es decir, es la aplicación quien informa al servidor el cambio de red enviando su nueva dirección IP de contacto.

3.2.1. SIPDROID

SIPDroid es un cliente VoIP disponible para dispositivos Android que utiliza el protocolo SIP. Es un software de código abierto, bajo licencia GPL desarrollado completamente en Java (Android) y que brinda múltiples posibilidades para la comunicación, entre ellas:

- Formato de cambio de número
- Soporte de varios modos de tonos DTMF[7]
- Soporte para NAT (traducción de direcciones de red)
- Llamadas salientes simultáneas
- Enmascaramiento para llamadas anónimas
- Enrutamiento para llamadas entrantes basado en tiempo
- Transferencia de llamadas asistido
- Conferencias
- Recepción de vídeo

[7] Dual-tone multi-frequency signaling (DTMF) es usado para señalización en telecomunicaciones sobre líneas de teléfonos análogos.

3.3. Esquema de Red

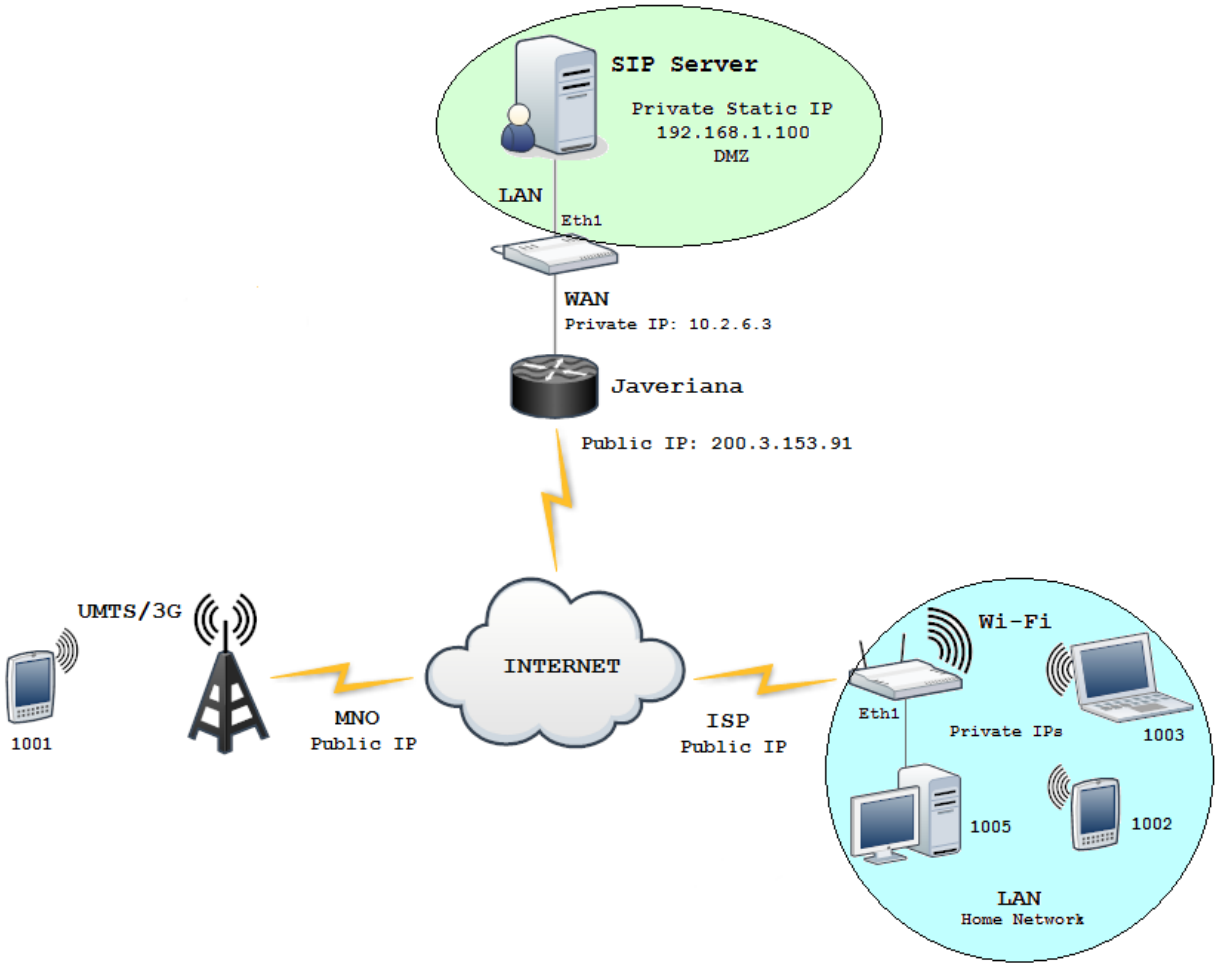


Figura 5. Diagrama de Red.[8]

La Figura 5, muestra el esquema de red que se implementa para el desarrollo del proyecto. El servidor SIP se encuentra ubicado dentro de una red local con dirección IP estática privada y es exhibido a la red externa en una zona desmilitarizada (DMZ), como se explicará más adelante en la sección 4.1.3 de éste documento.

Por medio de la interfaz WAN del enrutador, se conecta el servidor con la red interna de la Pontificia Universidad Javeriana, para después salir a internet por medio de una dirección IP pública estática, asignada para este desarrollo.

Los dispositivos en cualquier red externa acceden al servidor y establecen una conexión a través de internet, bien sea por medio de la red 3G de un MNO[9] o de una red doméstica provista por un ISP[10].

Es preciso notar que la comunicación entre el servidor SIP y cualquier dispositivo cliente utilizado para realizar la conexión, está bajo el efecto de múltiples traducciones de direcciones de red por medio de NAT[11].

[8] Autoría propia
[9] Mobile Network Operator
[10] Internet Service Provider
[11] Network Address Translation

4. DESARROLLO

El desarrollo de este proyecto de grado, como se ha venido mencionando en este documento, consta de la implementación del servidor SIP que permite la comunicación entre dos terminales y la aplicación móvil para el registro de los usuarios en la terminal. Dadas en el capítulo anterior las especificaciones alcanzadas durante el desarrollo de la solución al problema propuesto, se detalla en esta sección el proceso completo de desarrollo del trabajo de grado, las aproximaciones a la solución y la solución definitiva al problema.

4.1. Implementación KAMAILIO™

El principal requerimiento que debía cumplir el servidor, a parte del registro de los usuarios y el establecimiento de llamada, era que brindara soporte para NAT y que permitiera la configuración de la topología de acuerdo a la ubicación del servidor en la red; esto dado que siempre se debía tener presente la dificultad de transmisión de los datos multimedia (voz) en tiempo real, una vez establecida la llamada.

SIP al ser un protocolo de señalización, es independiente de la transmisión de la voz. Como se mencionó en los capítulos anteriores, SIP permite la inicialización, modificación y finalización de sesiones entre clientes, por medio del direccionamiento de mensajes de petición y respuesta, pero no se involucra directamente en el proceso de comunicación entre usuarios una vez se establece la conexión. Para esto, SIP se apoya específicamente en dos protocolos independientes, que a su vez interactúan entre sí, haciendo posible el envío de paquetes multimedia entre los usuarios.

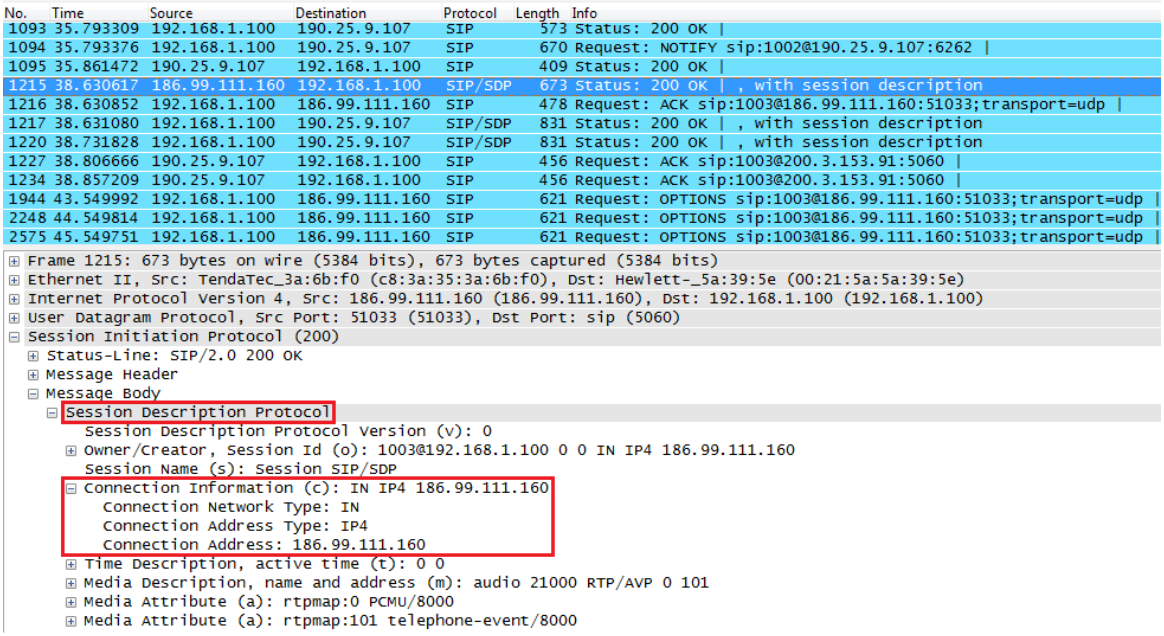


Figura 6. Información SDP.

Mediante el protocolo SDP los usuarios informan los parámetros de la sesión que se va a establecer, sus capacidades de comunicación (codecs) y su dirección IP de contacto, cómo se aprecia en la Figura 6. La información de contacto es fundamental e indispensable para el funcionamiento del protocolo RTP, ya que esta determina el correcto envío y recepción los mensajes multimedia.

Teniendo en cuenta lo mencionado anteriormente, se debía utilizar un servidor que permitiera el paso de los mensajes RTP a través de él, con el fin de solucionar el problema de la traducción de direcciones.

Kamailio es un servidor SIP de código abierto bajo licencia GPL y fue utilizado en primera instancia con el fin de comprobar su funcionamiento y sus posibilidades de configuración.

Se instaló la última versión estable del servidor (Kamailio V. 4.0) y adicionalmente, con el fin de atravesar el NAT presente en la red se instaló RTPproxy, un servidor que funciona como módulo interno en Kamailio y que se encarga de la recepción y el envío de los mensajes multimedia entre los usuarios, una vez establecida la llamada.

Kamailio 4.0 y RTPproxy 1.2.1 fueron instalados sobre Ubuntu 12.04 LTS[12] 32Bits, una de las distribuciones más utilizadas de Linux y con mayor soporte en la web.

El proceso de instalación de Kamailio 4.0 se detalla claramente en la referencia bibliográfica [8]. Una vez instalado el servidor, se debe proceder a configurarlo de acuerdo a la necesidad de los usuarios y la estructura de la red.

4.1.1. Configuración RTPproxy

El RTPproxy es configurado por medio del archivo `/etc/default/rtpproxy` de la siguiente forma:

```
# Defaults for rtpproxy

# The control socket.
CONTROL_SOCK="udp:127.0.0.1:7722"

# Additional options that are passed to the daemon.
EXTRA_OPTS="-l 200.3.153.91 -m 35000 -M 65000 -d DBUG:LOG_LOCAL5 -F"
```

CONTROL_SOCK es el parámetro mediante el cual se establece la comunicación entre el RTPproxy y Kamailio. En éste caso, se configuró para que la comunicación se realizara en la dirección local 127.0.0.1 a través del puerto 7722.

EXTRA_OPTS permite ligar la transmisión de los mensajes RTP a la dirección IP pública del servidor 200.3.153.91 a través de un rango de puertos, que en este caso se estableció entre el 35000 y el 65000. Además se crea un registro de depuración y se brindan permisos de administrador.

4.1.2. Configuración Kamailio

Kamailio se divide principalmente en tres archivos base que permiten modificar independientemente las opciones de arranque, el control y la configuración del sistema.

Para habilitar el arranque de Kamailio se configura el archivo `/etc/default/kamailio`

```
#
# Kamailio startup options
#

# Set to yes to enable kamailio, once configured properly.
RUN_KAMAILIO=yes
```

[12] Long-Term Support

```
# User to run as
USER=kamailio

# Group to run as
GROUP=kamailio

# Amount of shared memory to allocate for the running Kamailio server
(in Mb)
SHM_MEMORY=64

# Amount of private memory for each Kamailio process (in Mb)
PKG_MEMORY=4
…
…
```

Aquí se habilita el inicio, el usuario y la memoria compartida del servidor con el sistema operativo sobre el cual se ejecuta.

En el archivo `/etc/kamailio/kamctlrc` se configuran las variables de control del servidor. Uno de los módulos más importantes de Kamailio es el de administración de la base de datos, lugar donde se almacenan todos los registros del sistema, entre ellos, los datos de los clientes que son utilizados para la conexión y que identifican a cada usuario.

MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario y es usado por Kamailio como base central para la creación, modificación y administración de datos del servidor y de los clientes.

`Kamctlrc` es el archivo que contiene principalmente los parámetros de configuración de la base de datos MySQL y se configura de la siguiente forma:

```
# $Id$
#
# The Kamailio configuration file for the control tools.
#
…
…
## your SIP domain
SIP_DOMAIN=localhost

## chrooted directory
# $CHROOT_DIR="/path/to/chrooted/directory"
…
…
#
# If you want to setup a database with kamdbctl, you must at least
specify
# this parameter.
DBENGINE=MYSQL

## database host
DBHOST=localhost

## database name (for ORACLE this is TNS name)
DBNAME=kamailio

# database path used by dbtext, db_berkeley or sqlite
# DB_PATH="/usr/local/etc/kamailio/dbtext"
```

```
## database read/write user
DBRWUSER="kamailio"

## password for database read/write user
DBRWPW="kamailiorw"
…
…
```

SIP_DOMAIN es el parámetro donde se configura el dominio del servidor. Dado que en este caso el servidor no tiene una URL que se resuelva mediante un DNS y que se accede a éste únicamente a través de la dirección IP local, el SIP_DOMAIN se configura como "localhost".

En DBENGINE se establece el motor de la base de datos a ser usado por Kamailio, MySQL cómo se mencionó anteriormente. DBHOST indica la ubicación de la base de datos en la red, es decir, en el servidor local. DBNAME, DBRWUSER y DBRWPW son el nombre, el usuario y la contraseña para la administración de la base de datos.

Una vez establecida la configuración, se debe proceder a crear la base de datos en el sistema y así poder agregar los usuarios que podrán registrarse en el servidor. Para esto debe ejecutarse el comando `kamdbctl create` en una terminal, como se muestra en la Figura 7.

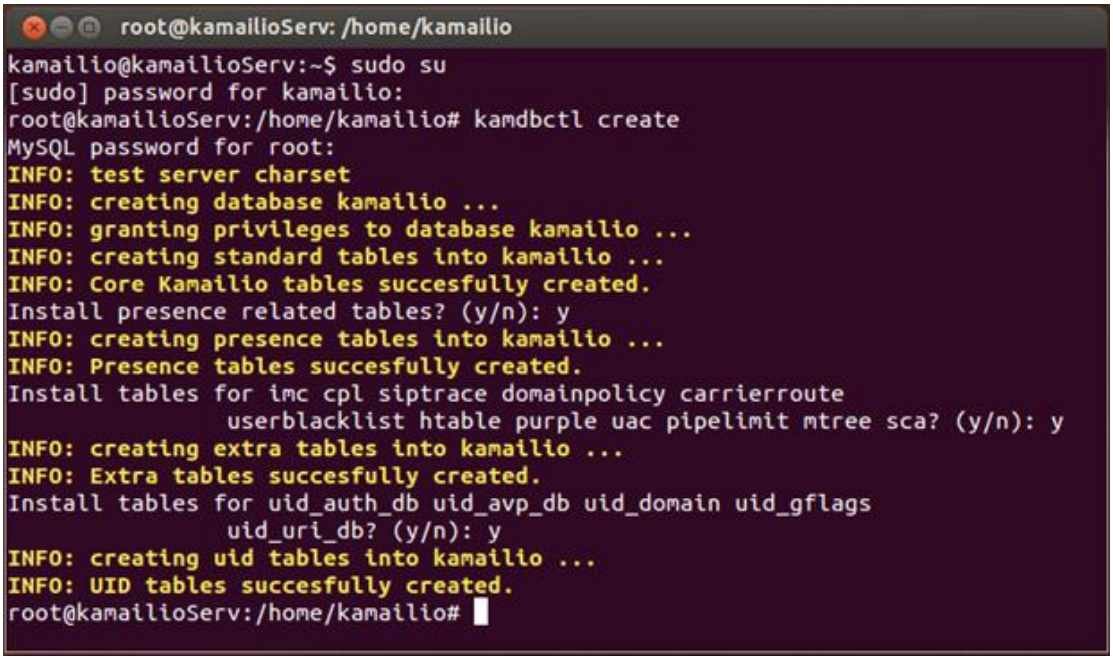


Figura 7. Creación de la base de datos.

Una vez creada la base de datos de Kamailio se debe continuar con la creación de los usuarios en el sistema para de esta forma dar acceso al servidor mediante el mensaje de registro que contiene los datos de autenticación y que se configuran en la aplicación cliente utilizada para realizar la conexión.

Para crear un usuario en la base de datos se ejecuta el comando `kamctl add userid password` en una terminal, como se muestra en la Figura 8.

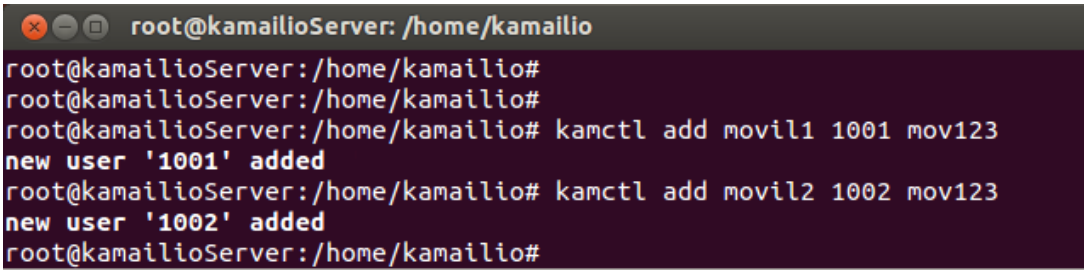


Figura 8. Creación de usuarios en la base de datos.

En la Figura 8 se aprecia la creación de los usuarios "1001" y "1002" con password "mov123" para ambos. Estos usuarios y contraseñas serán ingresados en la aplicación móvil para establecer el registro en el servidor.

Por último debemos configurar los módulos de Kamailio que serán utilizados durante su ejecución. Estos permiten dotar al servidor de múltiples funcionalidades, según sean requeridas y además establecer algunas variables para su adecuado funcionamiento.

El archivo `/etc/kamailio/kamailio.cfg` es el núcleo de configuración del servidor y en éste se establecen los siguientes parámetros:

```
#!KAMAILIO
#
# Kamailio (OpenSER) SIP Server v4.0 - default configuration script
#     - web: http://www.kamailio.org
#     - git: http://sip-router.org
#
…
…
####### Defined Values #########

#!define WITH_NAT
#!define WITH_MYSQL
#!define WITH_AUTH
#!define WITH_USRLOCDB
#!define WITH_XCAPSRV

# *** Value defines - IDs used later in config
#!ifdef WITH_MYSQL
# - database URL - used to connect to database server by modules such
#       as: auth_db, acc, usrloc, a.s.o.
#!ifndef DBURL
#!define DBURL "mysql://kamailio:kamailiorw@localhost/kamailio"
#!endif
#!endif
…
…
####### Global Parameters #########
…
…
fork=yes
children=4

/* uncomment the next line to disable TCP (default on) */
#disable_tcp=yes
```

```
/* uncomment the next line to disable the auto discovery of local
aliases based on reverse DNS on IPs (default on) */
#auto_aliases=no

/* add local domain aliases */
alias="200.3.153.91"

/* uncomment and configure the following line if you want Kamailio to
bind on a specific interface/port/proto (default bind on all available)
*/
listen=192.168.1.100:5060

/* port to listen to
* - can be specified more than once if needed to listen on many ports */
port=5060

#!ifdef WITH_TLS
enable_tls=yes
#!endif

#!ifdef WITH_XCAPSRV
tcp_accept_no_cl=yes
#!endif

# life time of TCP connection when there is no traffic
# - a bit higher than registration expires to cope with UA behind NAT
tcp_connection_lifetime=3605
…
…
####### Modules Section ########
…
…
#!ifdef WITH_MYSQL
loadmodule "db_mysql.so"
#!endif
…
…
#!ifdef WITH_AUTH
loadmodule "auth.so"
loadmodule "auth_db.so"
#!ifdef WITH_IPAUTH
loadmodule "permissions.so"
#!endif
#!endif
…
…
#!ifdef WITH_NAT
loadmodule "nathelper.so"
loadmodule "rtpproxy.so"
#!endif
…
…
#!ifdef WITH_XCAPSRV
loadmodule "xhttp.so"
#!endif

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

```
#!ifdef WITH_NAT
# ----- rtpproxy params -----
modparam("rtpproxy", "rtpproxy_sock", "udp:127.0.0.1:7722")
…
…
```

En la sección `## Defined Values ##` se configuran los módulos que serán utilizados por el núcleo de Kamailio.

WITH_NAT indica la ejecución del módulo de detección de traducción de direcciones de red, que como se mencionó anteriormente, se soporta en el RTPproxy para el envío de los mensajes multimedia. En la sección `## setting module-specific parameters ##` se establece el socket de comunicación entre Kamailio y el RTPproxy, tal y como se configuró en el archivo `/etc/default/rtpproxy` a través de la dirección local 127.0.0.1 por medio del puerto 7722.

WITH_MYSQL establece el uso de MySQL como la base de datos utilizada y WITH_AUTH restringe la conexión únicamente a los usuarios que envíen los datos de autenticación en el mensaje de registro al servidor.

WITH_USRLOCDB permite mantener activa la ubicación del usuario en la red, actualizando sus datos de conexión si es necesario.

En `## Global Parameters ##` se establecen los parámetros globales, entre estos el alias del dominio local, es decir, la dirección IP pública del servidor 200.3.153.91. Además, se indica la dirección IP privada a la cual se debe ligar la recepción y envío de los mensajes 192.168.1.100 y el puerto SIP 5060 establecido en la RFC3261.

Una vez configurado correctamente el archivo `kamailio.cfg` se procede a iniciar el servidor SIP. Para esto, existen dos archivos de arranque mediante los cuales se da la posibilidad de iniciar, reiniciar o detener tanto Kamailio como RTPproxy. Estos archivos son `/etc/init.d/rtpproxy` y `/etc/init.d/kamailio`.

El inicio de ambos servicios se ejecuta de la siguiente forma desde una terminal:

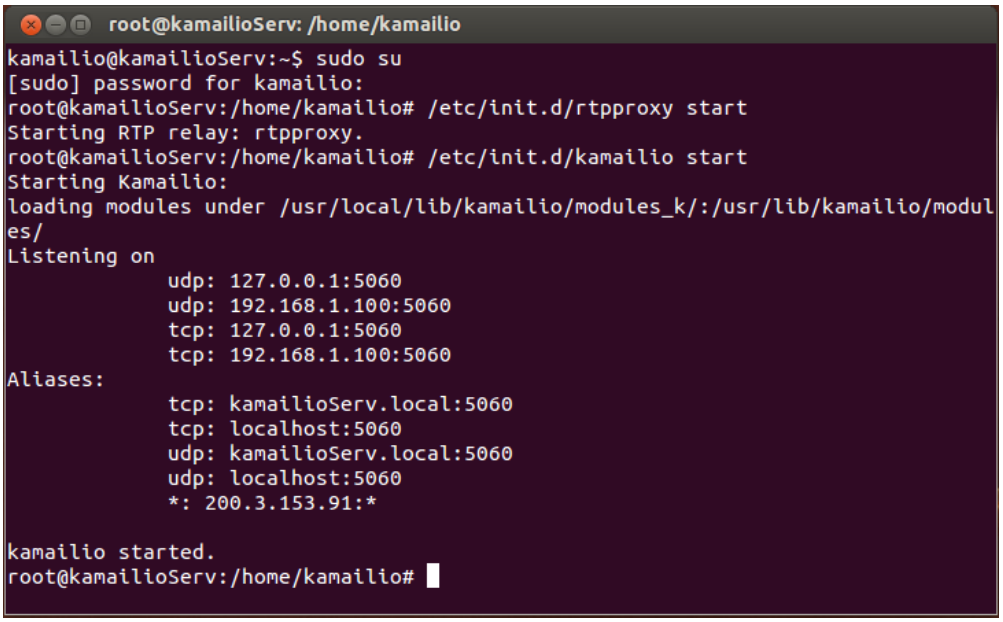


Figura 9. Inicio RTPproxy y Kamailio.

Mediante las opciones `start`, `stop` y `restart` se controla la ejecución de los servicios Kamailio y RTPproxy en el sistema. En la Figura 9 se muestra el correcto inicio del servidor, lo que indica que a partir de ese momento se encuentra listo para recibir conexiones entrantes.

4.1.3. DMZ del Servidor Kamailio

Dado que el servidor se encuentra dentro de una LAN y necesita ser expuesto completamente a la red externa para poder ser accedido a través de internet por los clientes, se debe configurar la dirección privada del servidor dentro de una "zona desmilitarizada" o DMZ[13], mediante la cual se establece una red perimetral, cuya función es permitir la conexión directa entre la red externa (a través de la dirección IP pública) y la red de área local (por medio de la IP privada del servidor).

El host que se encuentra dentro de la DMZ no se puede comunicar con los dispositivos de la red interna, de esta forma se protege el comprometer la seguridad de los demás dispositivos.

Al configurar la DMZ con la dirección IP privada del servidor dentro de la LAN, se indica al enrutador que debe dejar todos los puertos de comunicación abiertos, así se evitan posibles pérdidas de paquetes enviados por el servidor SIP o por el RTP.

Figura 10. Configuración DMZ.

En la Figura 10 se muestra la configuración de la DMZ en el enrutador, habilitando así la posibilidad de acceso de los clientes al servidor a través de Internet.

4.2. Implementación SIPDROID

Como se mencionó en el capítulo de especificaciones, SIPDroid es la aplicación utilizada como MMC[14]. De código abierto bajo licencia GPL, esta aplicación es desarrollada en Java para el sistema operativo Android y es un cliente VoIP para conexiones a servidores SIP.

4.2.1. SVN y AndroidSDK

Para poder trabajar sobre la aplicación y así verificar características como el flujo de datos durante una llamada, implementación de mensajes SIP y la posibilidad de modificación de los parámetros de la sesión

[13] Demilitarized Zone
[14] Mobility Management Client

durante una conexión de voz (conmutación de la llamada) se obtuvo el código fuente disponible en una base de datos de Google[15] de tipo SVN[16].

Mediante el uso del software DownloadSVN (Figura 11) y la ubicación de la troncal de SIPDroid dentro del repositorio disponible en Google, se obtiene el código fuente de la aplicación, completa y lista para ser utilizada en un software IDE[17].

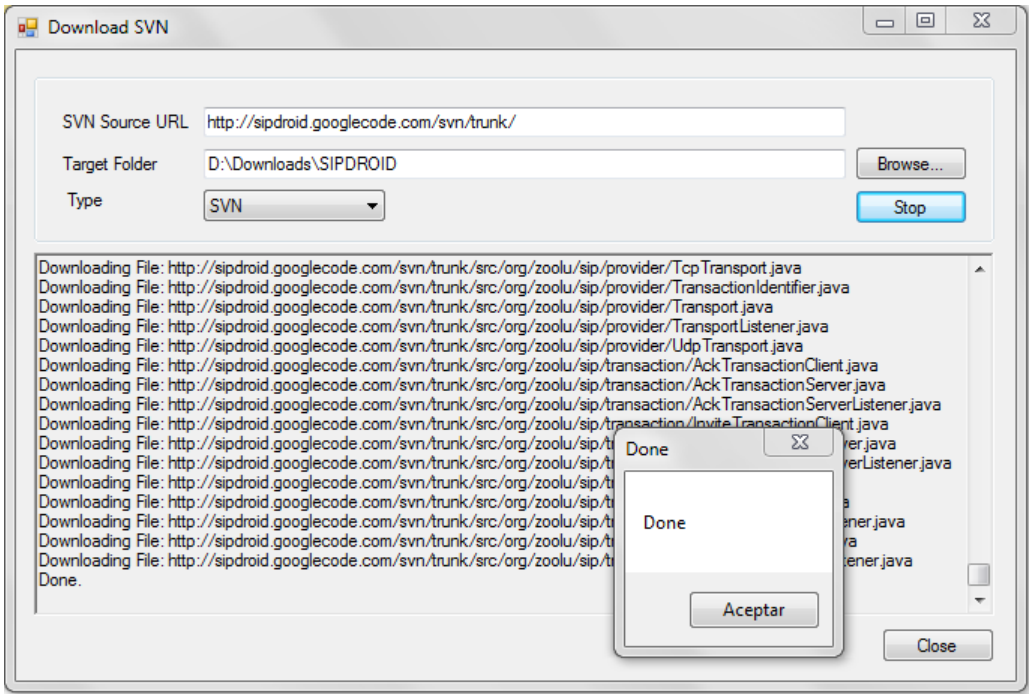


Figura 11. DownloadSVN para obtener el código fuente de la aplicación.

El software IDE es el programa que provee un conjunto de herramientas de programación y consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

Android provee todo un entorno de desarrollo de aplicaciones sobre su propia plataforma. AndroidSDK[18] proporciona las bibliotecas API[19] y las herramientas de desarrollo necesarias para crear, probar y depurar aplicaciones [9].

ADT Bundle provee todo el sistema de desarrollo necesario para comenzar a trabajar aplicaciones. Este incluye en un solo paquete:

- Eclipse + ADT plugin
- Android SDK Tools
- Android Platform-tools
- Android Emulator Images

Una vez obtenido el AndroidSDK, se debe proceder a importar el código fuente de la aplicación, descargado mediante el SVN.

[15] https://code.google.com/
[16] Sistema de control de versiones
[17] Integrated Development Environment
[18] Software Development Kit
[19] Application Programming Interface, es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

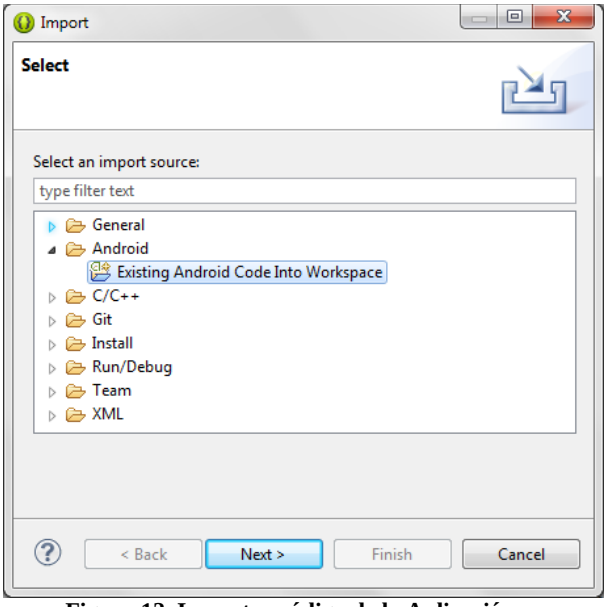


Figura 12. Importar código de la Aplicación.

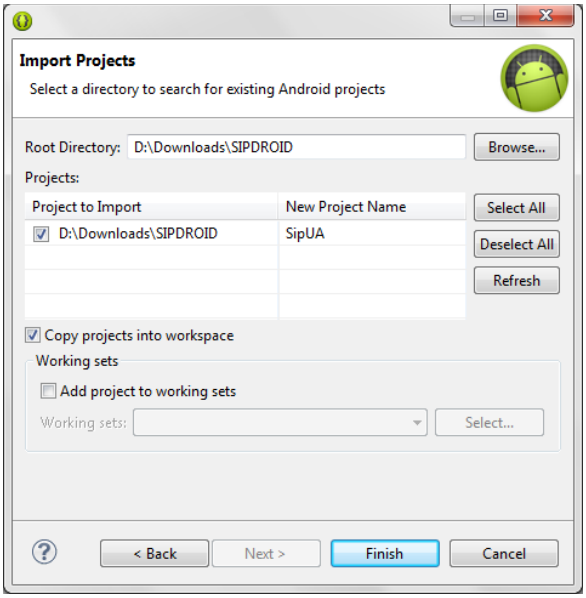


Figura 13. Creación del proyecto en el Workspace.

En la Figura 12 se muestra la pantalla de importación del código fuente, donde se elige importar al Workspace[20] un código Android existente (el anteriormente descargado en algún directorio del PC).

Cuando se selecciona el directorio donde se descargó el código fuente, AndroidSDK reconoce su contenido y genera el proyecto (Figura 13). Al finalizar solo resta configurar las propiedades de la aplicación y así poder explorar, modificar, compilar y depurar el código.

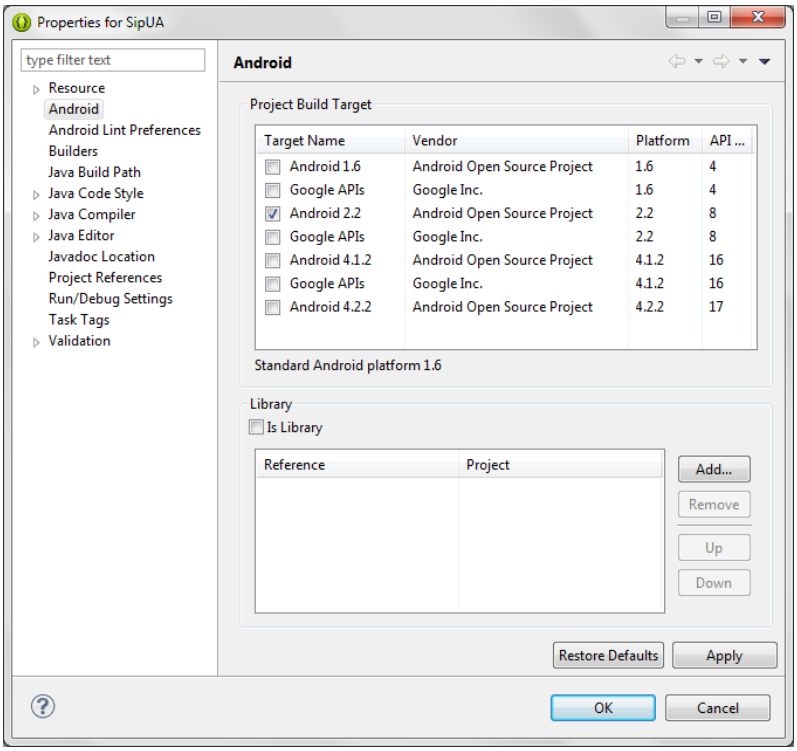


Figura 14. Configuración API Level.

[20] Workspace se refiere a un archivo o directorio que permite al usuario recopilar varios archivos y recursos de código fuente y trabajar con ellos como una unidad cohesiva.

En las propiedades del proyecto (clic derecho → properties), en la pestaña Android (Figura 14), se selecciona la mínima versión del sistema operativo con la cual es compatible la aplicación. La versión seleccionada será la base de compilación del proyecto, en este caso, Android 2.2.

El último paso es generar el archivo `R.java` el cual genera las variables que serán utilizadas y las traduce a posiciones de memoria hexadecimales. Para esto, se debe "limpiar" el proyecto (clic en la pestaña Proyect → Clean), lo que significa que se realiza una pre-compilación del código, se eliminan los errores que contenga el proyecto y se verifican los datos actuales de este.

4.2.2. Funcionamiento de la Aplicación

SIPDroid se divide fundamentalmente en tres capas de aplicación, éstas en conjunto interactúan para brindar la posibilidad de transmitir los mensajes a la capa de transporte del modelo TCP/IP (ver Figura 1), generar la sintaxis, operación y funcionalidades del protocolo SIP y dar la posibilidad al usuario de interactuar con una interfaz gráfica que gestione las actividades que se realizan dentro y fuera de la aplicación (ej. Realizar o recibir una llamada).

SipUA (SIP User Agent) es el proyecto que se genera dentro del IDE y contiene todos los paquetes[21] que conforman la aplicación y los recursos que ésta utiliza. La carpeta 'src' (Figura 15) contiene los paquetes que representan el código fuente de la aplicación. Estos están a su vez agrupados en tres secciones y representan el software base utilizado por SIPDroid y que en conjunto dotan a la aplicación de todas sus funcionalidades.

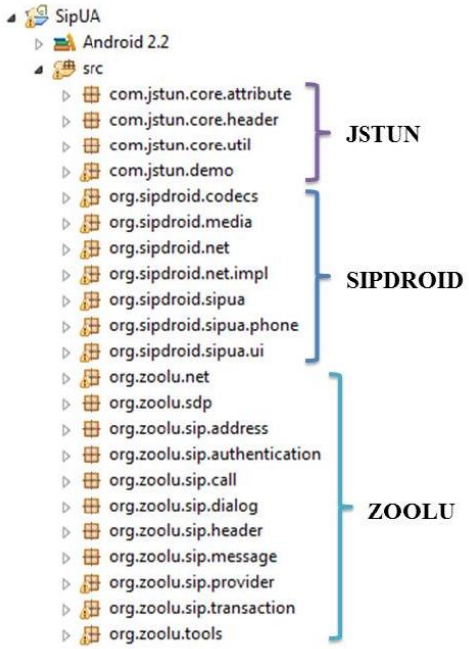


Figura 15. Paquetes base de SIPDROID.

JSTUN es el software de la capa más baja, en estos paquetes se generan los datagramas que se envían a través de los sockets de la aplicación y su principal funcionalidad es determinar condiciones de red tales como Firewalls o NATs.

[21] Agrupación de clases e interfaces asociadas

ZOOLU es el software más estrechamente ligado a todo el funcionamiento de SIPDROID. En estos paquetes se encuentran las clases que definen e implementan todo lo concerniente al protocolo SIP, según la RFC 3261. Cada clase representa una característica o funcionalidad de SIP. Entre estas se encuentran los protocolos de transporte (TCP y UDP), el protocolo SDP, la detección de direcciones IP, mensajes, llamada, etc.

SIPDROID es el software propio de la aplicación, estos paquetes contienen principalmente las clases que generan y controlan la interfaz gráfica. Sus principales funcionalidades se centran en ejercer control sobre las acciones externas del usuario (marcar una extensión), internas de la red (desplegar pantalla de timbrado al recibir una solicitud de llamada), interfaz con el sistema operativo, el servidor y otras aplicaciones cliente.

Para mayor información sobre las clases contenidas en los paquetes JSTUN, ZOOLU y SIPDROID consultar el anexo [1] SipUA.

4.2.3.Estados de la Aplicación

SIPDroid realiza sus funciones a partir de estados dentro de ciclos de interrupciones generadas por el usuario o por la interfaz de red mediante la cual se conecta (ej. Pulsación de la pantalla o invitación de llamada).

Al iniciar la aplicación en el móvil Android, cuando la cuenta del usuario ha sido correctamente configurada, se procede a realizar el registro en el servidor SIP establecido. SIPDroid toma la dirección del servidor y envía el mensaje de registro con los datos del usuario, así entra en un ciclo de estados durante toda su ejecución.

En la Figura 16 se muestra la máquina de estados que emula el proceso de registro que se realiza dentro de SIPDroid.

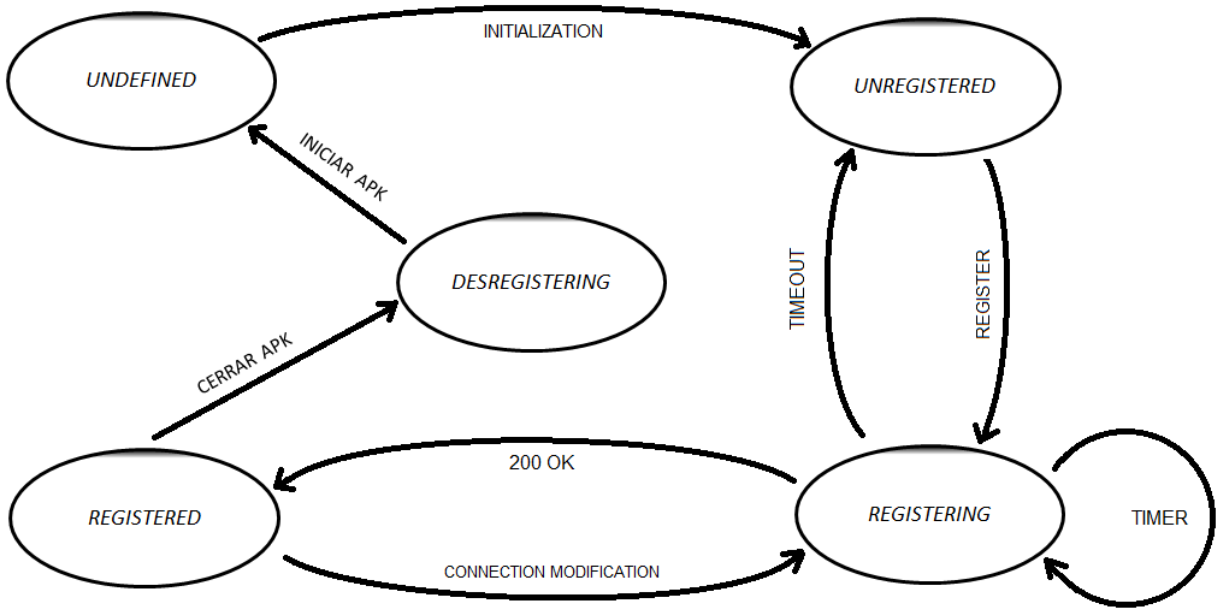


Figura 16. Estados de registro SIPDroid.[22]

[22] Autoría propia

En cuanto al proceso de llamada, este puede describirse también como una máquina de estados, donde estos son dependientes de los mensajes enviados y recibidos por la aplicación y las acciones del usuario durante la comunicación. En este caso se debe tener en cuenta que para el establecimiento y finalización de una llamada existen dos vías: que el usuario sea quien desee iniciar/finalizar una llamada o que este sea invitado a iniciarla o informado para terminarla.

En la Figura 17 se encuentra detallada la máquina de estados que emula el proceso de llamada dentro de la aplicación y las acciones que originan el movimiento del ciclo de comunicación.

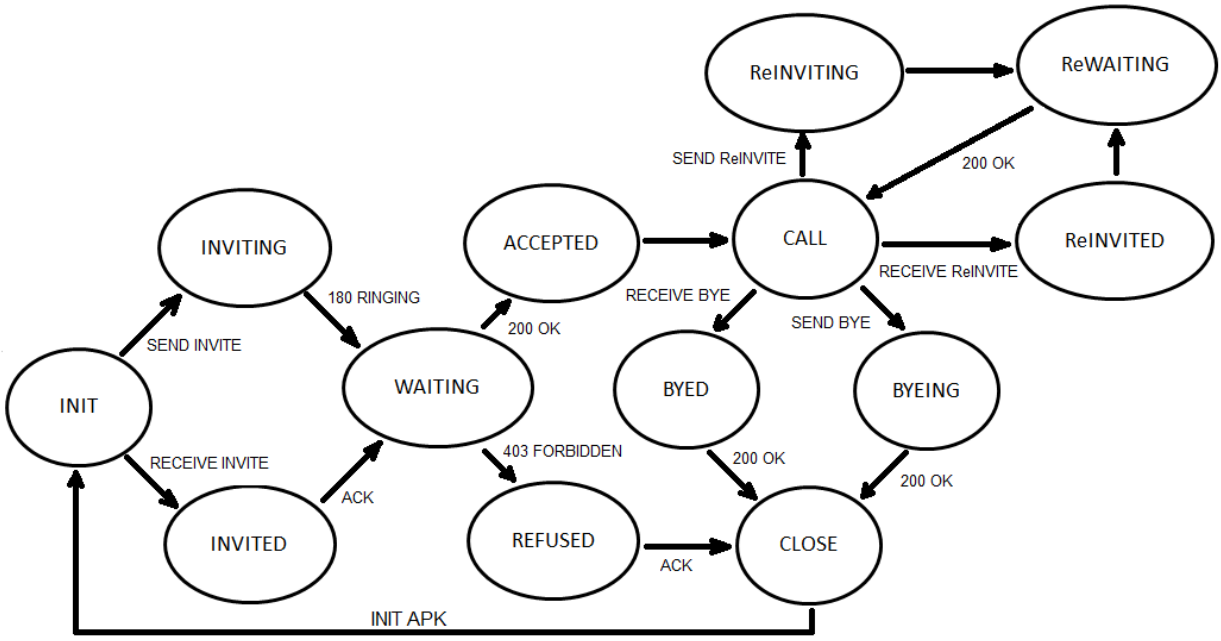


Figura 17. Estados de llamada SIPDroid.[23]

4.2.4. Conectividad por medio de 3G y WiFi

Uno de los puntos más importantes y que determina el desarrollo del presente trabajo de grado, es el soporte a la conectividad por medio de redes 3G y WiFi por parte de la aplicación móvil implementada. SIPDroid dentro de sus múltiples funcionalidades brinda esta posibilidad, permitiendo la transmisión y recepción de datos tanto por 3G como por WiFi, dependiendo de su disponibilidad.

Dentro del código de la aplicación, se implementó una prueba para verificar la conexión a una u otra red, haciendo uso de un hilo[24] dentro de la ejecución del programa que no afectaba su normal funcionamiento.

En primera instancia, se definieron las funciones mediante las cuales se verifica la conectividad a la red móvil o a la red WiFi de la siguiente forma:

```
private void WifiaMobile (){
       while (true) {
       final String DEBUG_TAG = "NetworkStatusExample";

    ConnectivityManager connMgr = (ConnectivityManager)
```

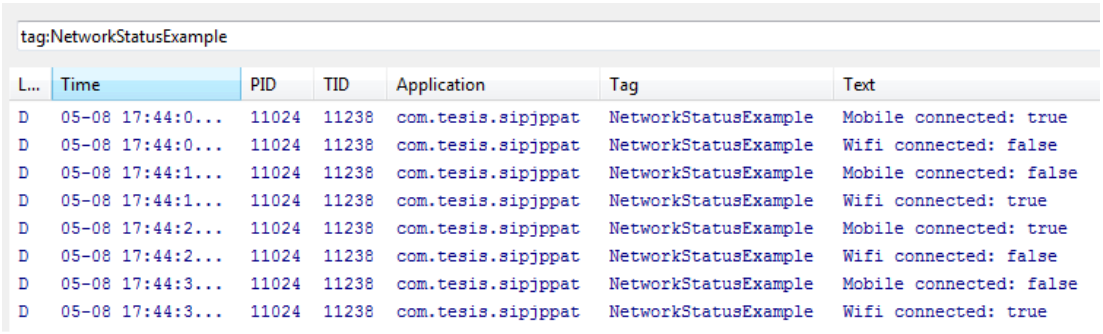
[23] Autoría propia
[24] Unidad de procesamiento más pequeña que puede ser planificada. Permite a una aplicación realizar varias tareas a la vez (concurrentemente).

```
        getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo networkInfo = connMgr.getNetworkInfo(ConnectivityManager.TYPE_WIFI);
        boolean isWifiConn = networkInfo.isConnected();
        networkInfo = connMgr.getNetworkInfo(ConnectivityManager.TYPE_MOBILE);
        boolean isMobileConn = networkInfo.isConnected();
        if (isMobileConn==true && isWifiConn==false){
            Log.d(DEBUG_TAG, "Mobile connected: " + isMobileConn);
           Log.d(DEBUG_TAG, "Wifi connected: " + isWifiConn);
            MobileaWifi();
                        }
           }
    }

    private void MobileaWifi (){
           while (true) {
           final String DEBUG_TAG = "NetworkStatusExample";

        ConnectivityManager connMgr = (ConnectivityManager)
        getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo networkInfo = connMgr.getNetworkInfo(ConnectivityManager.TYPE_WIFI);
        boolean isWifiConn = networkInfo.isConnected();
        networkInfo = connMgr.getNetworkInfo(ConnectivityManager.TYPE_MOBILE);
        boolean isMobileConn = networkInfo.isConnected();
        if (isWifiConn==true && isMobileConn==false){
           Log.d(DEBUG_TAG, "Mobile connected: " + isMobileConn);
           Log.d(DEBUG_TAG, "Wifi connected: " + isWifiConn);
            WifiaMobile();
                        }
           }
    }
```

Al definir las funciones `WifiaMobile` y `MobileaWifi` y almacenar el estado de la conexión en las variables `isWifiConn` e `isMobileConn` por medio del método `isConnected()`, era posible generar un hilo que se ejecutara dentro de la aplicación y que arrojara el estado de la conexión de la siguiente forma:

```
public class TareaParalelo extends AsyncTask<Object, Object, Object>  {

   @Override
   protected Object doInBackground(Object... params) {
           // TODO Auto-generated method stub
            while(true){
            //final String DEBUG_TAG = "NetworkStatusExample";
           ConnectivityManager connMgr = (ConnectivityManager)
            getSystemService(Context.CONNECTIVITY_SERVICE);
            NetworkInfo networkInfo= connMgr.getNetworkInfo(ConnectivityManager.TYPE_WIFI);
            boolean isWifiConn = networkInfo.isConnected();
            networkInfo = connMgr.getNetworkInfo(ConnectivityManager.TYPE_MOBILE);
            boolean isMobileConn = networkInfo.isConnected();
            /*Log.d(DEBUG_TAG, "Wifi connected: " + isWifiConn);
            Log.d(DEBUG_TAG, "Mobile connected: " + isMobileConn);  */
            if (isWifiConn==true && isMobileConn==false){
            WifiaMobile();
            // Log.d(DEBUG_TAG, "Wifi connected: " + isWifiConn);
                        }
            if (isMobileConn==true && isMobileConn==false){
            MobileaWifi();
            //  Log.d(DEBUG_TAG, "Mobile connected: " + isMobileConn);
                        }
            }
   }
```

El estado de la conexión de ambas redes es visible en el log[25] de la aplicación, cómo se muestra en la Figura 18.



tag:NetworkStatusExample

L...	Time	PID	TID	Application	Tag	Text
D	05-08 17:44:0...	11024	11238	com.tesis.sipjppat	NetworkStatusExample	Mobile connected: true
D	05-08 17:44:0...	11024	11238	com.tesis.sipjppat	NetworkStatusExample	Wifi connected: false
D	05-08 17:44:1...	11024	11238	com.tesis.sipjppat	NetworkStatusExample	Mobile connected: false
D	05-08 17:44:1...	11024	11238	com.tesis.sipjppat	NetworkStatusExample	Wifi connected: true
D	05-08 17:44:2...	11024	11238	com.tesis.sipjppat	NetworkStatusExample	Mobile connected: true
D	05-08 17:44:2...	11024	11238	com.tesis.sipjppat	NetworkStatusExample	Wifi connected: false
D	05-08 17:44:3...	11024	11238	com.tesis.sipjppat	NetworkStatusExample	Mobile connected: false
D	05-08 17:44:3...	11024	11238	com.tesis.sipjppat	NetworkStatusExample	Wifi connected: true

Figura 18. Log de estado de conexión.

Como resultado de la prueba de conexión de la aplicación, se evidenció una característica del sistema operativo móvil Android que modificaba directamente el planteamiento del desarrollo del presente trabajo de grado.

Al verificar el envío de datos por parte del dispositivo móvil cuando éste se encuentra en presencia de redes 3G y WiFi simultáneamente, se encontró que es el sistema operativo el encargado de decidir la red por medio de la cual envía los datos, dependiendo de su estabilidad y nivel de potencia. Esto quiere decir que la aplicación no tiene permisos de decisión sobre la red por medio de la cual transmite y recibe paquetes, sino que es el sistema operativo como se mencionó anteriormente, quien escoge la red a utilizar.

Las aplicaciones permiten o no la conexión a las redes que soporten, pero no deciden cuando conmutar el envío de datos por una u otra red.

Esta característica de Android (y de la mayoría de sistemas operativos móviles) es clara en cuanto a que si se diera la posibilidad de que cada aplicación instalada en el dispositivo decidiera la red por medio de la cual trasmite y recibe información, se necesitaría mantener activas ambas conexiones todo el tiempo, lo que implica mantener dos direcciones IP (una para datos 3G y otra para WiFi), incrementando ampliamente el consumo de batería, el nivel de procesamiento en el dispositivo y la transmisión de potencia de las señales para cada red.

Es preciso aclarar que este hallazgo no implicó ningún inconveniente en el desarrollo de la solución al problema planteado, sino que por el contrario evita tener que cargar a la aplicación de procesamiento innecesario, dejando la responsabilidad de la conexión al sistema operativo.

Android es entonces, quien al encontrar la presencia de una red WiFi conocida, medir su potencia y verificar su estabilidad a partir del envío de algunos paquetes, establece la conexión y conmuta el envío de datos de la red 3G a la red WiFi, dadas sus características y mayor ancho de banda.

4.2.5. Configuración de cuentas en SIPDroid

En la opción "Ajustes" de SIPDroid, el usuario puede configurar los parámetros más relevantes de la aplicación, entre estos se encuentra la creación de una cuenta SIP, calidad de audio/video y codecs.

Para su configuración la aplicación solicita el ingreso de unos datos mínimos, por medio de los cuales busca el servidor en la red y realiza el registro del cliente.

[25] Registro oficial de eventos durante un rango de tiempo en particular

En la Figura 19 se muestra la pantalla de configuración de una cuenta SIP dentro de la aplicación SIPDroid. Allí se ingresan los datos de identificación del usuario (usuario de autorización y contraseña) y la información de red del servidor (dirección IP pública y privada) y se habilita el uso tanto de redes 3G como de redes WLAN para el establecimiento de la conexión.

Figura 19. Configuración de cuenta SIP en la aplicación.

4.3. Implementación ELASTIX®

Como se mencionó en el capítulo de especificaciones del sistema, la solución alcanzada y mediante la cual se cumplieron los objetivos propuestos en el proyecto del presente trabajo de grado fue mediante el uso de Elastix como servidor SIP.

Dadas las limitaciones del servidor Kamailio que se presentan en el análisis de resultados (capítulo 5) y después de realizar múltiples intentos para solucionar el problema, se decidió implementar Elastix como servidor SIP, debido a su estructura y soporte como central telefónica PBX. El uso de FreePBX como sistema de administración de Asterisk, dio la posibilidad de establecer la configuración del servidor más claramente y el entorno gráfico permitía su fácil y rápida verificación.

4.3.1. Instalación del Servidor

Durante el proceso de instalación de Elastix se configura uno de los parámetros más importantes del servidor. La dirección IP que se asigna al sistema (Figura 20), no es solamente la dirección IP del servidor,

sino también la ruta de acceso a la consola de administración gráfica. Esta dirección debe ser configurada como estática con el fin de evitar su modificación durante el arranque o reinicio del sistema.

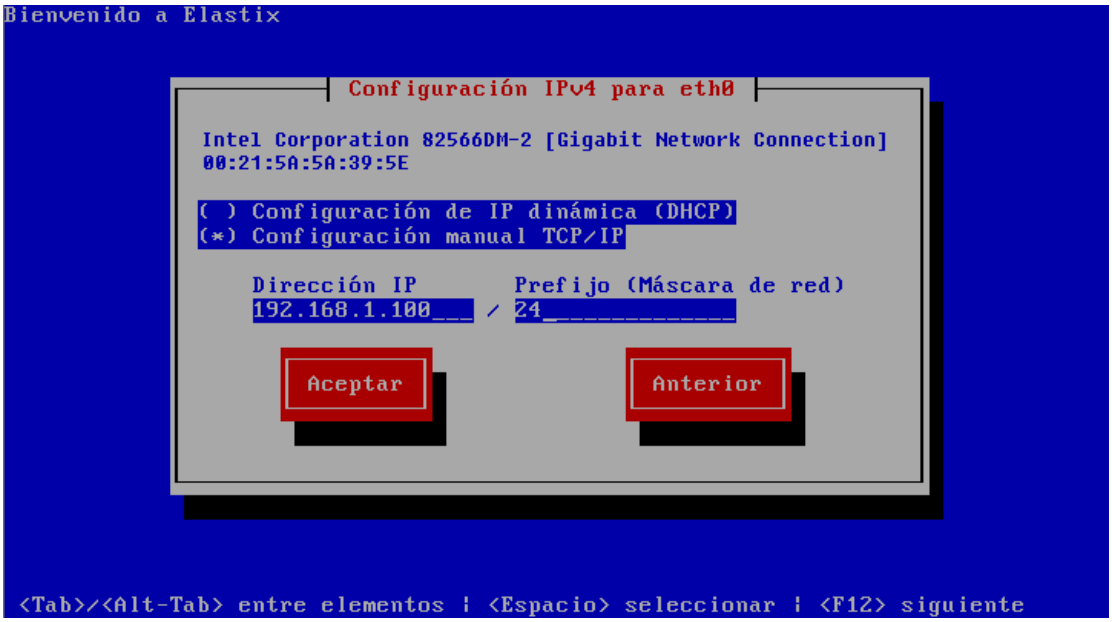


Figura 20. Asignación dirección IP estática del servidor Elastix.

Una vez finalizado el proceso de instalación, se accede al sistema a través del usuario de administración 'root'. Al iniciar la sesión, el servidor nos indica la dirección IP de conexión a la interfaz gráfica de configuración y administración, es decir, la dirección que configuramos anteriormente durante el proceso de instalación, como se muestra en la Figura 21.

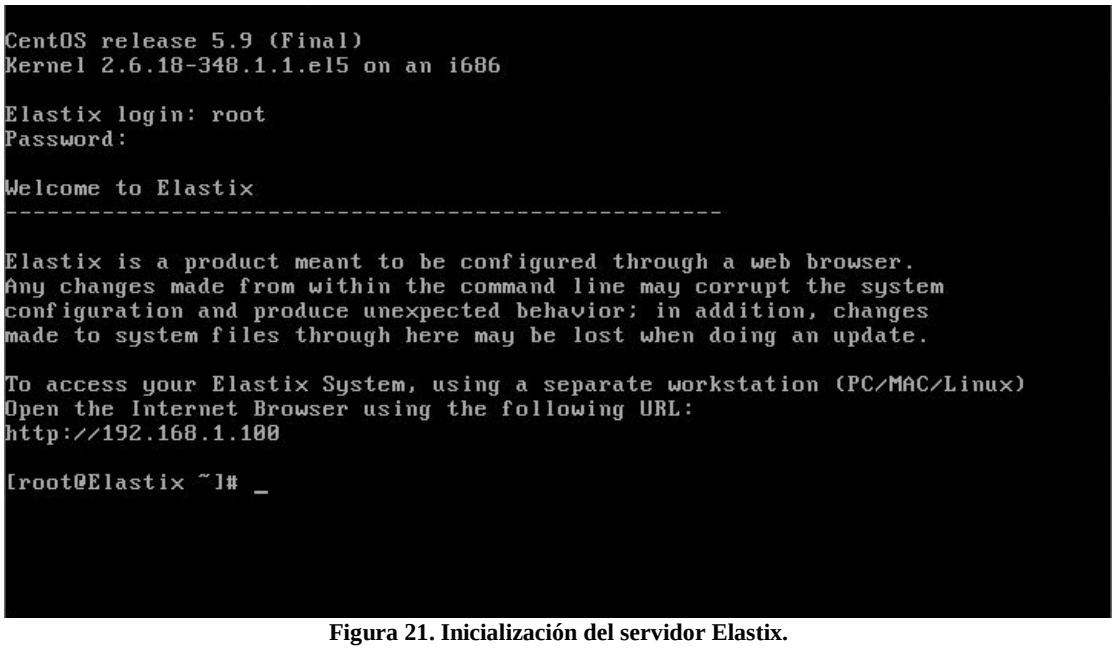


Figura 21. Inicialización del servidor Elastix.

4.3.2. Interfaz de Configuración y Administración

El acceso a la interfaz gráfica de Elastix se realiza mediante un navegador web, ya sea en un equipo dentro de la LAN del servidor por medio de la dirección IP privada, a través de internet por medio de la dirección IP pública o en el mismo sistema de Elastix instalando el entorno gráfico de CentOS.

La pantalla inicial de la consola de administración es un tablero de control donde se muestra la información más relevante del servidor. Aquí se puede ver la utilización de recursos del sistema como memoria RAM y CPU, almacenamiento, servicios y actividad de llamadas.

Esta pantalla es netamente informativa (Figura 22) y en ella se evidencia el comportamiento del servidor y su consumo de recursos, dando la posibilidad de prever inestabilidades o ralentizaciones cuando haya alto tráfico de llamadas.

Figura 22. Dashboard de Elastix.

4.3.3. Creación de usuarios y extensiones

El módulo de telefonía de Elastix, que como hemos mencionado en varias ocasiones a lo largo de este documento se soporta en la integración con Asterisk, brinda múltiples opciones de configuración para una central telefónica que se adapte a las necesidades de la red, de los usuarios y de su propósito. Por esta razón Elastix no es solamente un servidor SIP, ya que su sistema soporta la transmisión de mensajes y el establecimiento de comunicaciones por medio de múltiples protocolos. Dado el alcance de este proyecto y su fin, se hace uso expresamente de las funcionalidades como servidor SIP.

Elastix al igual que Kamailio, utiliza como motor de base de datos MySQL, pero a diferencia de ese servidor los módulos de MySQL se encuentran embebidos dentro del sistema y son administrados en su totalidad por Elastix.

El primer paso para la creación de un usuario en el sistema es indicar el tipo de dispositivo mediante el cual se configura la extensión en la terminal. Esto no es más que indicar al servidor el protocolo de comunicación que utilizará el cliente para realizar la conexión y el proceso de llamada. Como ya se dijo, todos los clientes creados en Elastix utilizan SIP (Figura 23).

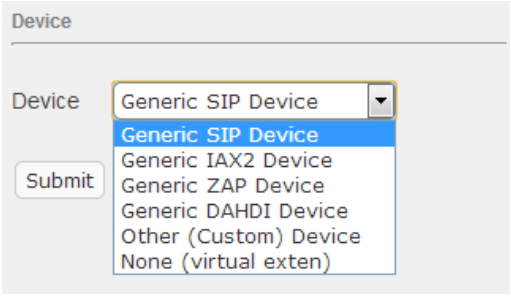


Figura 23. Tipos de dispositivo soportados por Elastix.

Los parámetros básicos que se deben configurar durante la creación de un usuario son el número de extensión, el nombre de visualización y el alias SIP. El número de la extensión es el número de marcado por medio del cual se contacta al usuario; el nombre de visualización es el texto que identifica al usuario cuando realice una llamada (ej. NN); el SIP alias, es un texto alternativo al número de marcado, si se desea contactar al usuario por un nombre en lugar de un número.

Figura 24. Creación de una extensión.

En la Figura 24 se muestra la creación básica del usuario y en la Figura 25 se muestran las opciones de configuración para la extensión creada en el sistema.

Las opciones más importantes que se configuran a cada dispositivo son la contraseña de autenticación y el soporte para NAT. Además se da la posibilidad de configurar los codecs que serán permitidos o denegados para el establecimiento de las llamadas y un control de acceso a ciertas direcciones de red, si así se requiere.

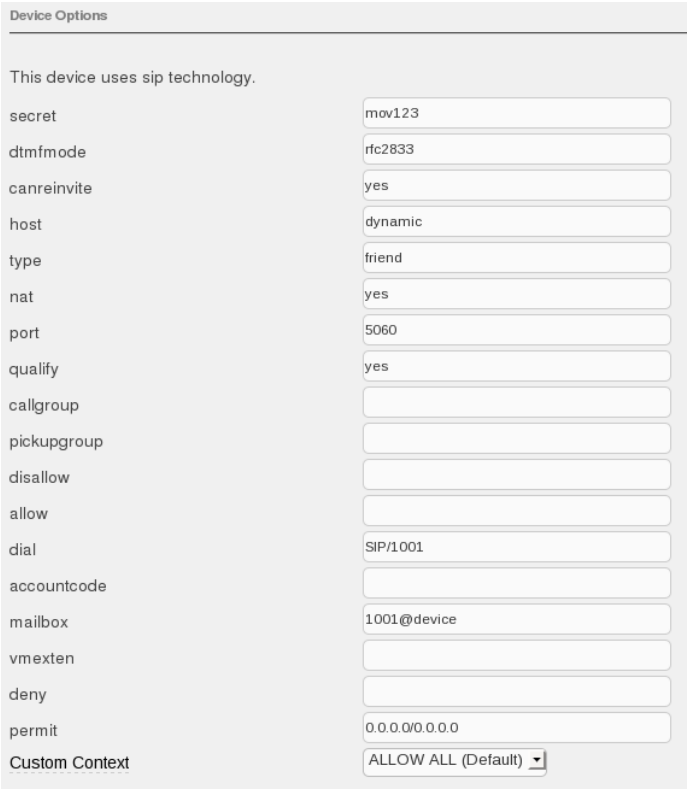


Figura 25. Opciones del dispositivo.

Una vez se han creado los usuarios en el sistema, Elastix permite observar su comportamiento en tiempo real a través del Panel Operador como se muestra en la Figura 26. Allí se pueden observar todos los usuarios creados en el sistema, su estado de conexión al servidor y su estado de llamada.

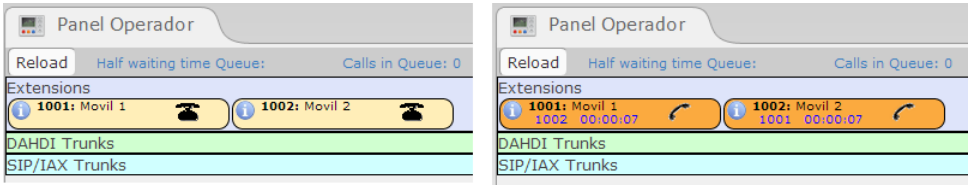


Figura 26. Panel Operador.

4.3.4. Configuración FreePBX® (ASTERISK®)

FreePBX es una interfaz gráfica GUI[26] para el control y administración de Asterisk. Por defecto, ésta se encuentra embebida dentro de todo el entorno de Elastix; esto hace más sencilla la administración del sistema, pero limita muchas de las características de Asterisk.

Existe la posibilidad de extraer solamente la interfaz gráfica de FreePBX, sin desligarlo de Elastix para tener acceso a todas las opciones de configuración sobre Asterisk de forma independiente.

Al extraer la interfaz gráfica de FreePBX, nos encontramos con una pantalla de estado del sistema, similar al tablero de control de Elastix, pero exclusivo del módulo de Asterisk (Figura 27). Allí encontramos la información más relevante sobre el funcionamiento de la central telefónica, las llamadas y la utilización de recursos del sistema.

[26] Graphical User Interface

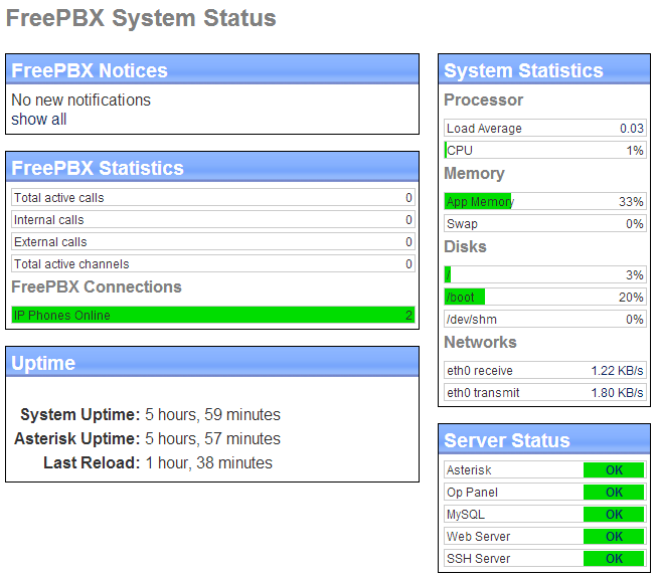


Figura 27. Estado del sistema FreePBX.

Para realizar la configuración de la red y el NAT e indicar al sistema la necesidad de utilizar un proxy RTP para la transmisión de datos multimedia entre clientes, se accede a 'Asterisk SIP Settings' dentro de las herramientas de FreePBX. Allí se configuran las direcciones de red (Interna y Externa), se habilita el uso de NAT para todas las comunicaciones, se establecen los parámetros del protocolo RTP y se habilitan los codecs permitidos por el servidor para el empaquetamiento de la voz (Figura 28).

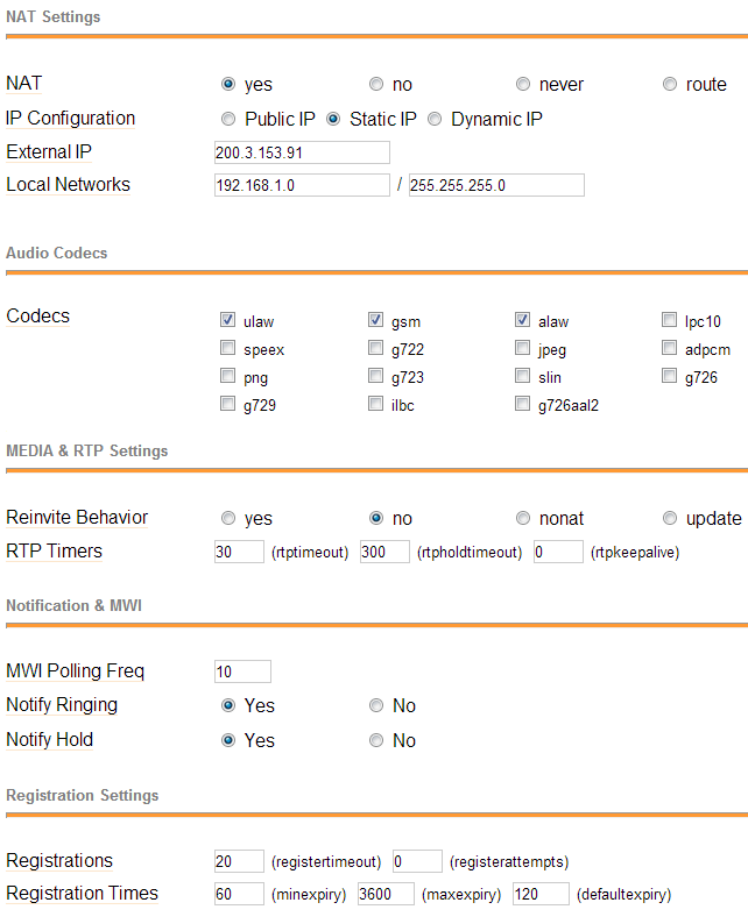


Figura 28. Configuración Asterisk SIP Settings.

4.3.5. DMZ del Servidor Elastix

Al igual que se realizó con Kamailio, para poder acceder al servidor SIP desde internet, realizar el registro de los usuarios y establecer conexiones de voz, era necesario introducir la dirección IP privada de Elastix dentro de una zona desmilitarizada en el enrutador al cual se conecta el equipo.

Dado que la configuración de red de Elastix se realizó de la misma forma como se configuró con Kamailio (ver Figura 5), la DMZ del servidor no varía y por lo tanto es la misma a la presentada en la sección 4.1.3.

4.4. Conmutación de llamadas de voz entre redes 3G y WiFi a través de Elastix

Gracias a la integración del proxy RTP dentro de la central telefónica de Elastix y su soporte para NAT, se encontró la solución al problema planteado en el proyecto y se logró conmutar una llamada de voz establecida en una red, a otra de diferentes características (3G/WiFi), es decir, realizar un Handover Vertical sin que se cortara la comunicación.

La conmutación de la llamada se efectúa mediante un nuevo mensaje de registro por parte del cliente que realiza el cambio de red. Dado que la totalidad de los paquetes multimedia se envían al servidor y es éste quien los redirige a su destino, el cambio de IP no afecta a la contraparte en la comunicación, ya que ésta seguirá enviando sus paquetes RTP al servidor.

En el flujo de mensajes SIP que se muestra en la Figura 29, se aprecia el esquema del proceso de conmutación logrado, el cual será comprobado más adelante en el análisis de resultados (capítulo 5).

Como se ha dicho ya anteriormente, es el servidor SIP quien se encarga por medio del proxy RTP, de actuar como enrutador de los paquetes multimedia, recibiéndolos y redirigiéndolos a su destino. Es por esto que los clientes en los dispositivos terminales nunca se enteran realmente de la ubicación en la red de su contraparte en la llamada, puesto que la conexión y el intercambio de mensajes y paquetes de voz lo realizan únicamente con el servidor.

Mediante el envío del mensaje REGISTER durante la llamada, el cliente que conmuta de red actualiza su dirección de contacto en el servidor, indicándole de esta forma la nueva IP por medio de la cual realizará la comunicación.

La aplicación móvil actualiza la dirección IP y realiza el envío del mensaje REGISTER al servidor, en el instante en el que el sistema operativo se conecta o desconecta de una red WiFi y modifica la conexión para el envío de datos.

En las Figuras 30 y 31 se aprecia el cambio de dirección IP en el estado del sistema de un dispositivo Android. La Figura 30 muestra la conexión del móvil a la red de datos 3G y la Figura 31 el cambio a una red WiFi.

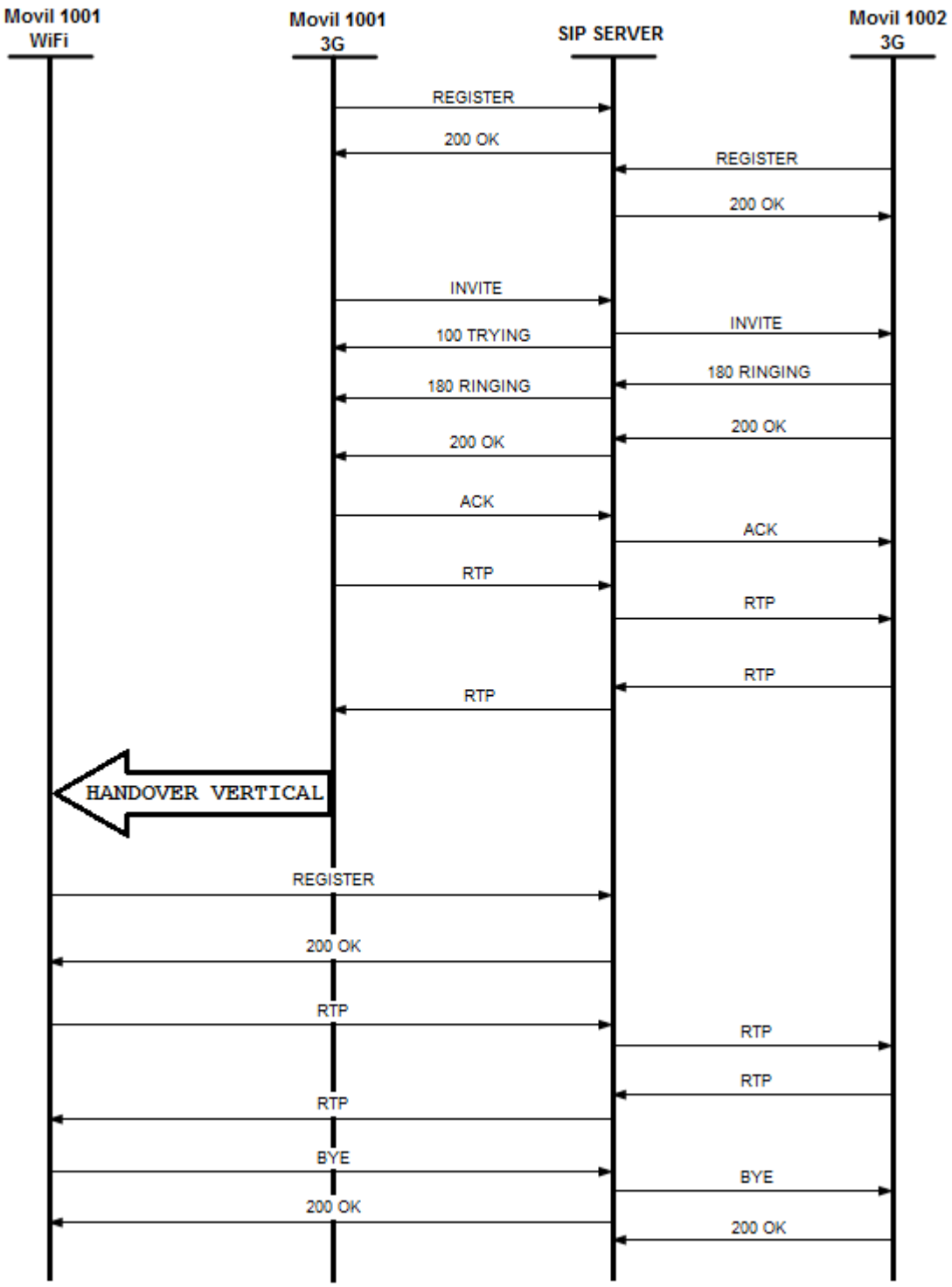


Figura 29. Flujo de mensajes SIP durante la conmutación de red.[27]

[27] Autoría propia

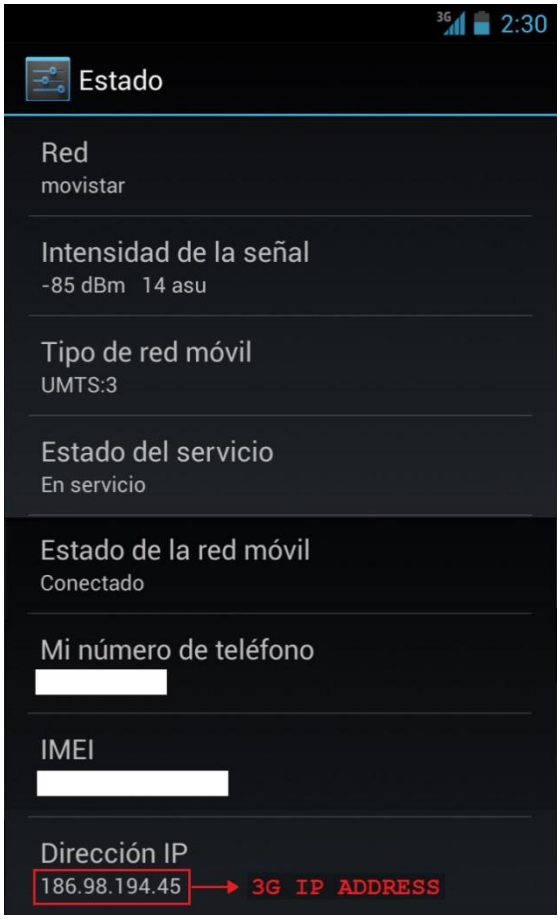


Figura 30. Conexión red 3G.

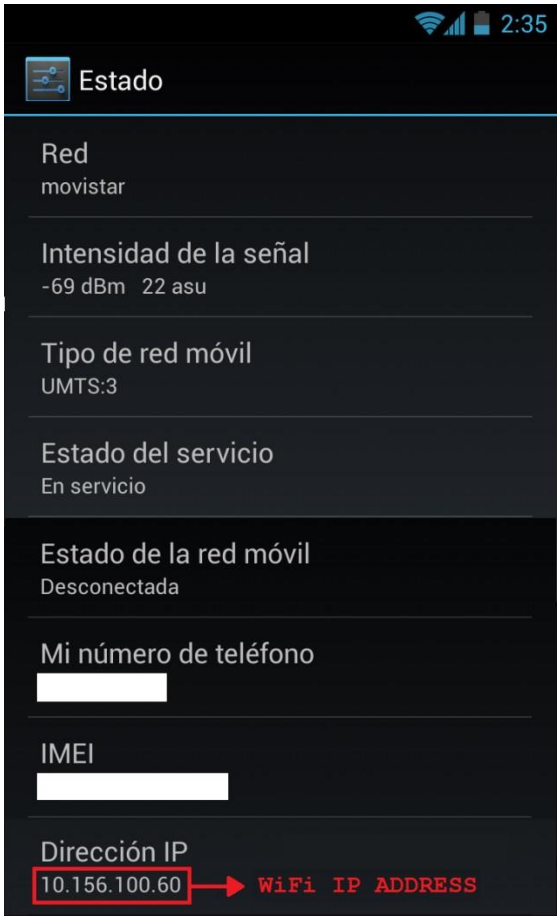


Figura 31. Conexión red WiFi.

5. ANÁLISIS DE RESULTADOS

En cuanto a los resultados obtenidos a lo largo del desarrollo del presente trabajo de grado, es preciso evidenciar la principal diferencia entre la conexión de un dispositivo a una red 3G y a una red WiFi.

Cuando un dispositivo se conecta a una red 3G, éste se identifica con una dirección IP pública que le es asignada exclusivamente para su comunicación y acceso a internet. Cuando el dispositivo establece la conexión a una red WiFi, necesariamente adquiere una dirección IP privada, la cual es traducida por medio de NAT a una dirección IP pública para acceder a internet.

Esta diferencia determinó en gran medida los resultados que se detallan en éste capítulo y fue concluyente para la realización exitosa de la conmutación de una llamada de voz entre una red 3G y una red WiFi.

Las capturas de las pruebas mostradas a continuación y los logs de las llamadas se encuentran en los anexos [2] y [3].

5.1. Resultados obtenidos utilizando el servidor Kamailio

Una vez instalado y configurado el servidor Kamailio tal y como se explica en la sección 4.1, se realizaron las correspondientes pruebas para verificar su correcto funcionamiento.

5.1.1. Comunicación entre redes 3G

Kamailio permitió establecer una comunicación estable y de buena calidad cuando ambos dispositivos se encontraban conectados a redes 3G, como se muestra a continuación.

El servidor recibe el mensaje INVITE originado por el usuario 1002 y lo redirige al usuario 1001. Dado que este mensaje es de tipo SIP/SDP, en él se envía la información de conexión del cliente, indicando la dirección IP por medio de la cual recibirá los paquetes multimedia (voz). El servidor reenvía el mensaje INVITE, así quien recibe la solicitud de llamada obtiene la dirección IP de contacto de quien origina la comunicación.

La Figura 32 muestra el envío del mensaje INVITE al servidor por parte del usuario 1002, siendo este el que origina la llamada y la Figura 33 la redirección del mensaje al usuario 1001, quien recibe la solicitud para el establecimiento de la comunicación. En ambas figuras se destaca la información de contacto contenida en el SDP del mensaje enviado por 1002 y recibido por 1001.

El resultado de la llamada entre dos clientes conectados a redes 3G se aprecia en el flujo de mensajes que atraviesan el servidor durante todo el proceso de establecimiento, comunicación y finalización de la conexión, mediante el uso de Wireshark[28].

[28] Programa de captura de las tramas de una red. Analizador de paquetes y protocolos.

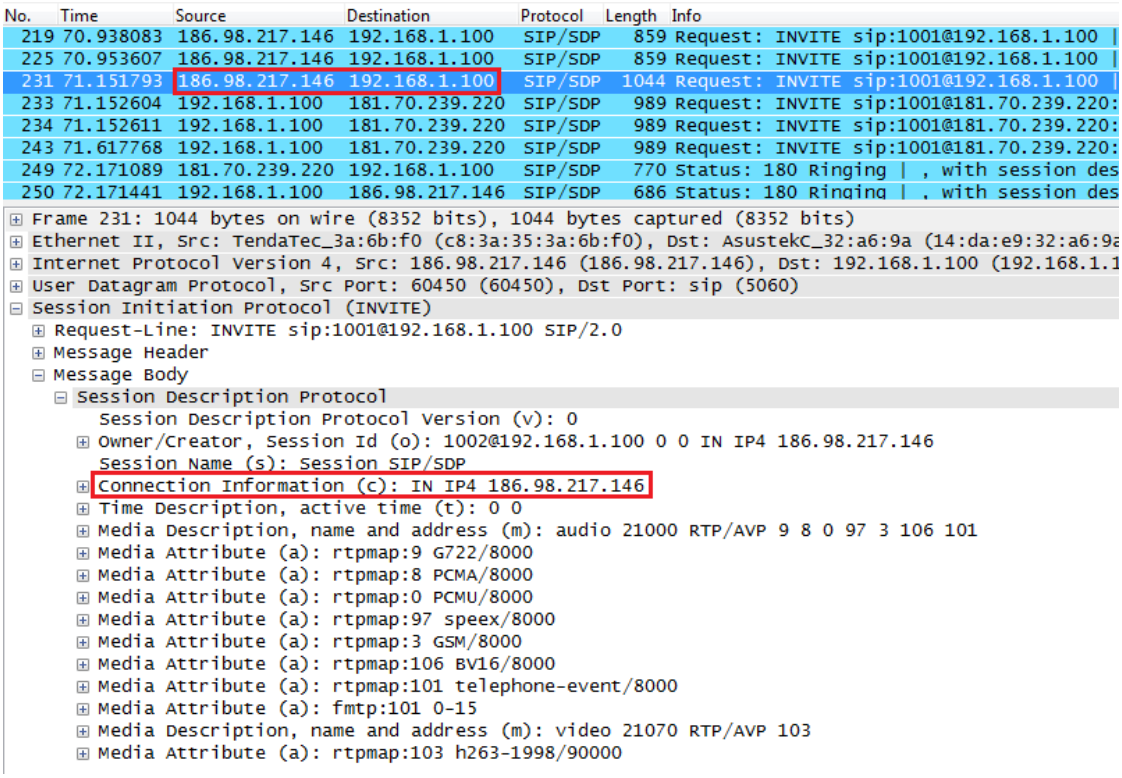


Figura 32. INVITE 3G de 1002 al Servidor Kamailio.

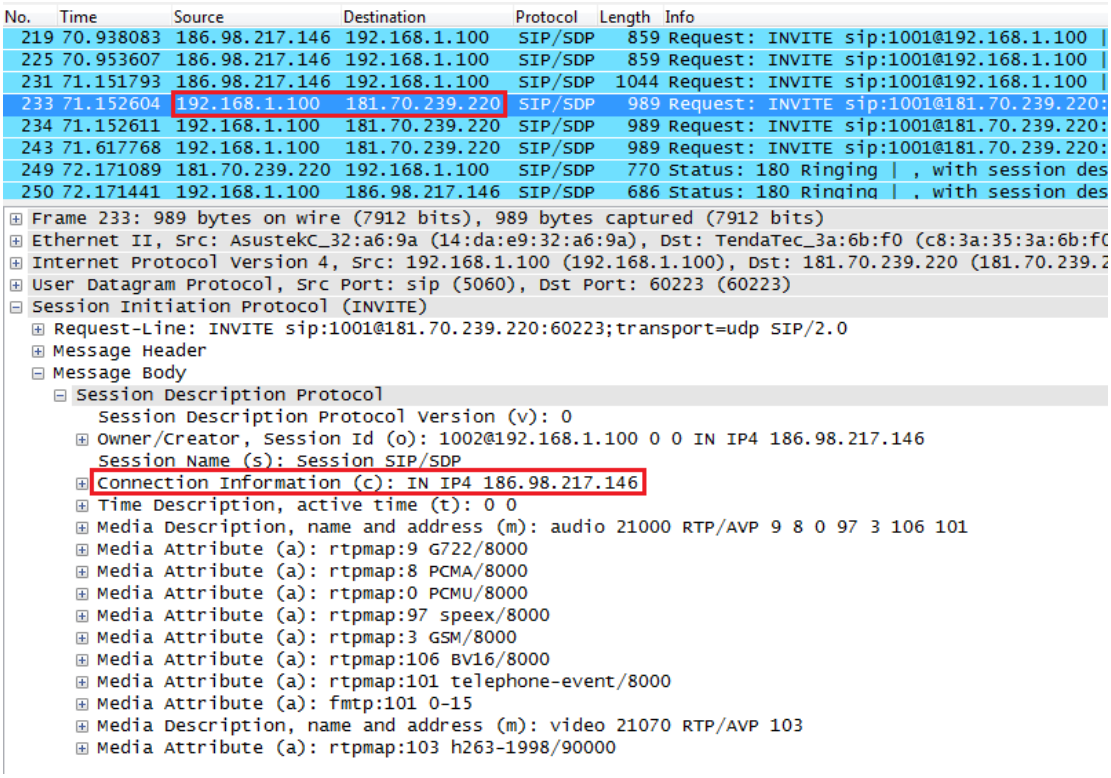


Figura 33. INVITE 3G del Servidor Kamailio a 1001.

Cuando el usuario 1001 recibe la solicitud y contesta la llamada, envía un mensaje 200 OK de tipo SIP/SDP de la misma forma como el usuario 1002 envió el mensaje INVITE. En el momento en el cual el usuario 1002 recibe el mensaje 200 OK por parte del servidor, extrae la información de contacto del

usuario 1001 contenida en el SDP, permitiéndole iniciar la transmisión de paquetes de voz. El proceso es el mismo al explicado anteriormente y el flujo de mensajes se muestra en las Figuras 34 y 35.

De esta forma se logró realizar una llamada exitosa, a través del servidor Kamailio, transmitiendo los paquetes de voz directamente de un cliente a otro. Como ambos usuarios obtienen la dirección pública de su contraparte y ésta es un identificador único para un dispositivo en una red 3G, el servidor RTP nunca entra en funcionamiento, haciendo que los clientes envíen directamente el contenido multimedia de un extremo de la comunicación al otro.

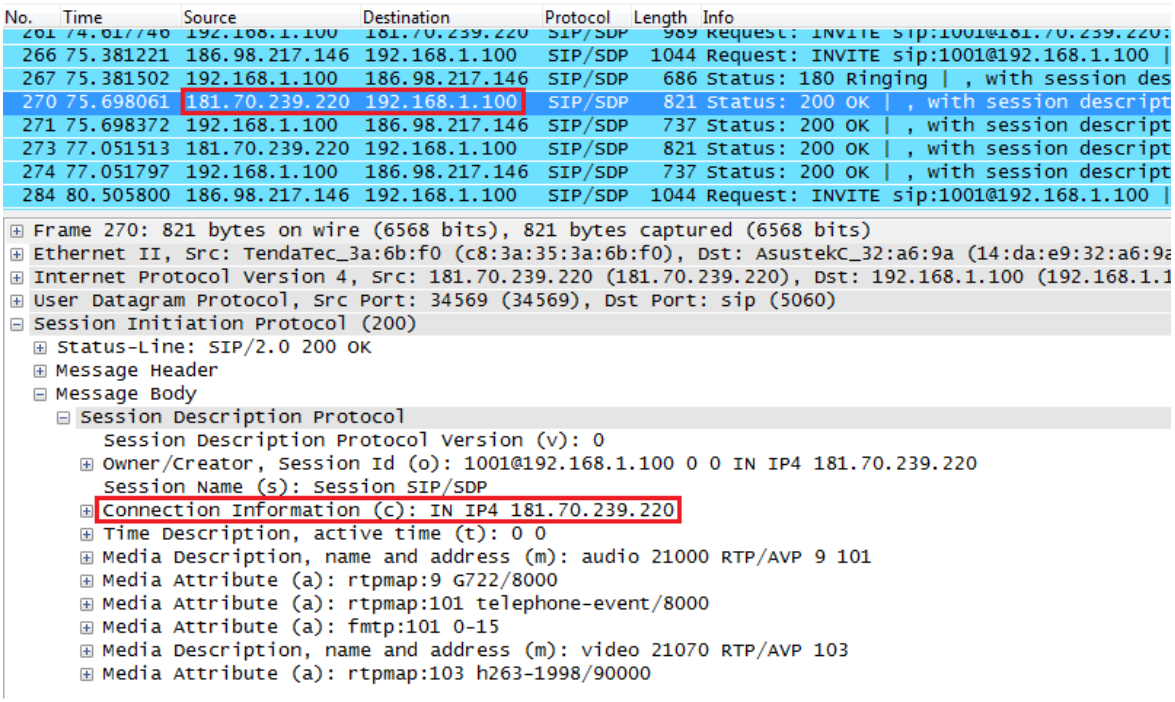


Figura 34. 200 OK 3G de 1001 al Servidor Kamailio.

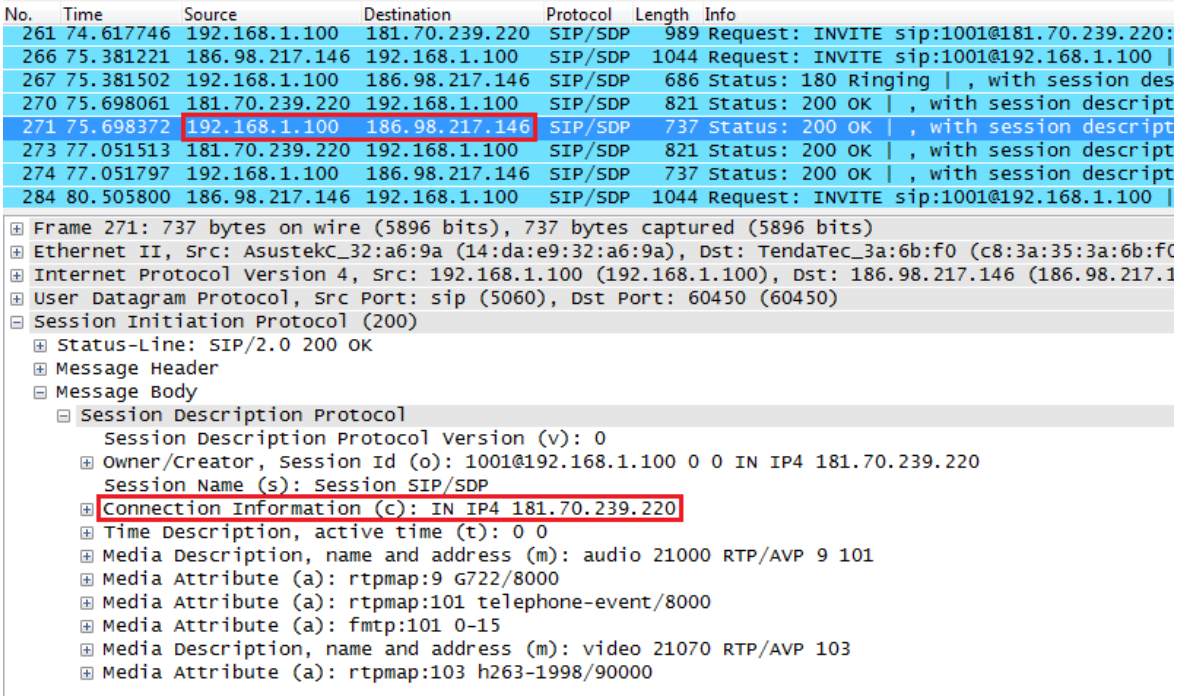


Figura 35. 200 OK 3G del Servidor Kamailio a 1002.

En la Figura 36 se puede ver un análisis gráfico de la comunicación VoIP capturada por Wireshark donde se muestran los paquetes SIP que pasaron por el servidor para la realización de la llamada. Allí se muestran los paquetes en una línea de tiempo desde que se envía el primer mensaje INVITE, hasta que finaliza la comunicación con un mensaje BYE. El esquema muestra las direcciones IP de los clientes en ambos extremos del gráfico y la dirección IP del servidor en el medio actuando como proxy.

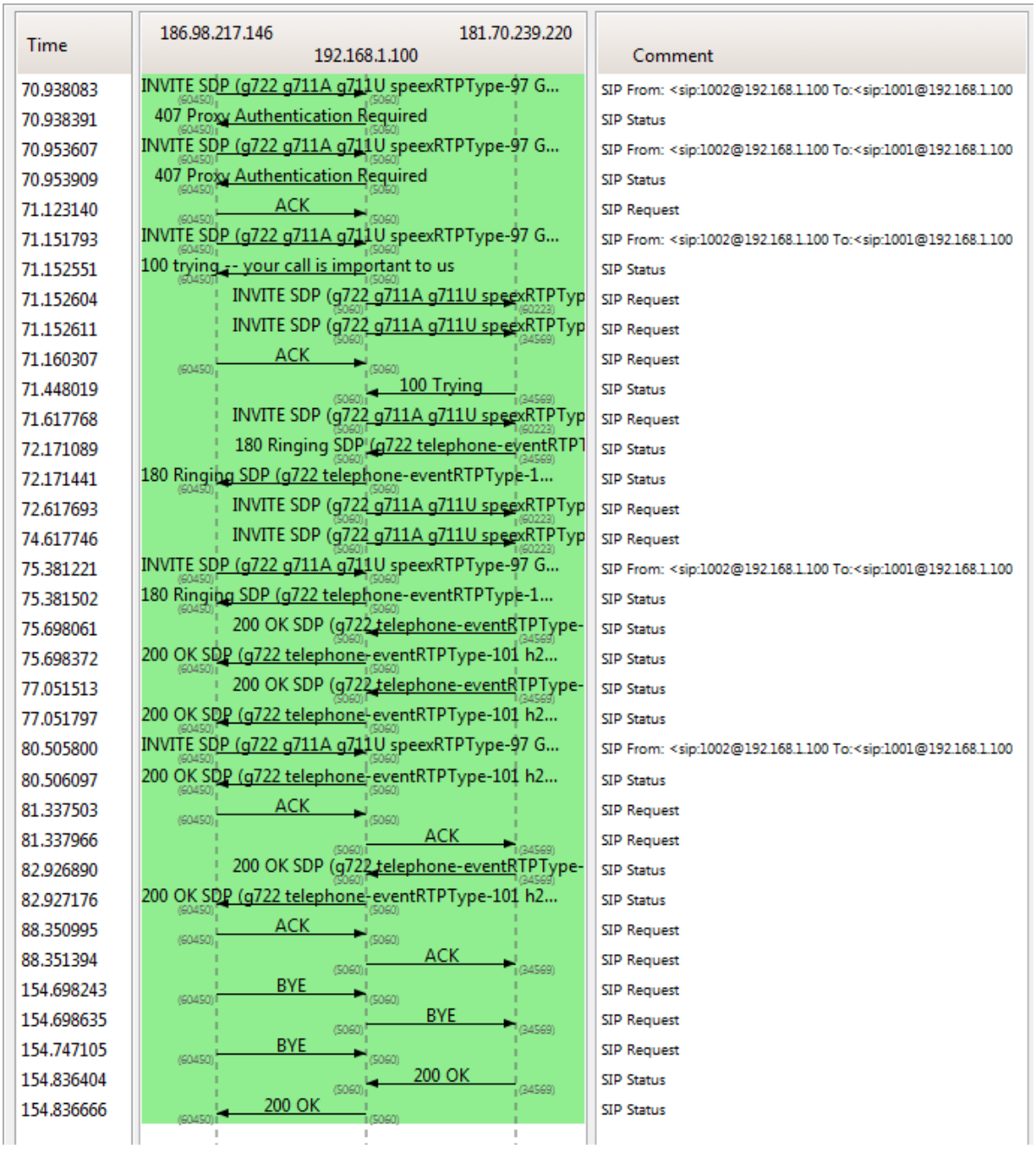


Figura 36. Análisis gráfico llamada VoIP 3G – 3G Kamailio.

Como análisis de la calidad percibida en el dispositivo móvil durante la comunicación, se obtuvo un log de las variables medidas por SIPDroid una vez se establece la conexión, las cuales permiten tener una representación cuantitativa del estado y estabilidad de la llamada.

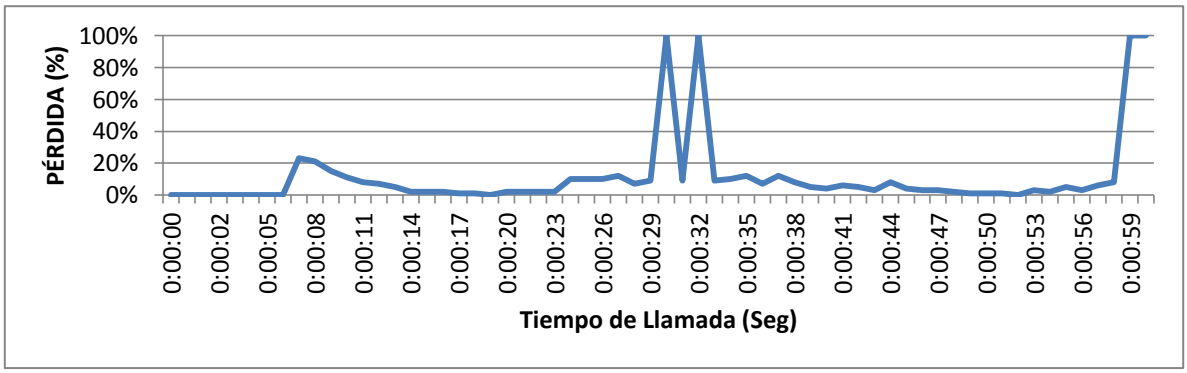


Figura 37. Porcentaje de Pérdida llamada 3G – 3G Kamailio.

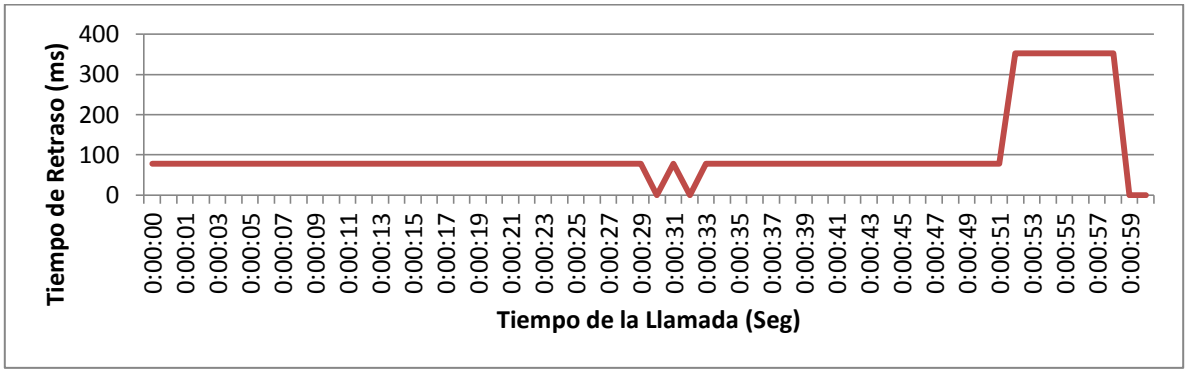


Figura 38. Tiempo de Retraso llamada 3G – 3G Kamailio.

La Figura 37 muestra el porcentaje de perdida de datos en la comunicación. En ella se puede observar que la perdida se mantiene por debajo del 20% durante casi la totalidad del tiempo de llamada y que en algunos puntos se perdió la recepción de paquetes (silencio de voz), los cuales representan un pico del 100% durante unos pocos segundos.

La Figura 38 es la representación en tiempo (milisegundos) del retraso que sufren los paquetes en ser recibidos por el cliente. Tiempos menores a 200 milisegundos son imperceptibles por el usuario; tiempos mayores ocasionan silencios esporádicos que entrecortan la voz y hacen incomoda la comunicación.

5.1.2. Comunicación entre redes WiFi

En este punto del desarrollo del proyecto se encontró el principal inconveniente, que derivó en la búsqueda de una nueva solución al problema.

En la prueba de comunicación entre redes 3G (sección 5.1.1) se mostró que el RTPproxy no entra en funcionamiento y que Kamailio envía la información de contacto de los usuarios sin realizar ninguna modificación. Ya que la dirección IP de los dispositivos cuando estos se conectan a una red 3G es una dirección pública, asignada exclusivamente a cada una de las terminales móviles, esta situación no representa ningún problema en el establecimiento de la comunicación entre los dos usuarios.

En el momento en el que un dispositivo se conecta a una red WiFi, como se explicó al inicio de este capítulo, adquiere una dirección IP privada que es traducida a una dirección IP pública por medio de NAT. Esta dirección IP pública, a diferencia de la asignada a través de 3G no representa a un solo dispositivo, ya que ésta identifica la red WiFi en su totalidad, es decir, por medio de ella se conectan a internet todos los dispositivos que se encuentren dentro de la LAN de la red WiFi (ver Figura 5).

En la prueba realizada, tanto el usuario 1001 como el usuario 1002 se encuentran dentro de la LAN de diferentes redes WiFi. Al enviar un mensaje INVITE al servidor para establecer una llamada, el usuario 1002 envía su dirección IP privada en la información de contacto dentro del SDP, por lo que el RTPproxy debe traducirla a la dirección del servidor para recibir los paquetes de voz y redirigirlos a cada cliente según su destino.

En la Figura 39, el servidor recibe el mensaje INVITE proveniente del usuario 1002 quien origina la llamada y lo reenvía al usuario 1001 sin que el RTPproxy modifique la dirección de contacto, como se ve en la Figura 40. El usuario 1001 recibe en la solicitud de llamada una dirección IP privada que no conoce y de igual forma sucede cuando éste contesta la llamada y envía el mensaje 200 OK, en el cual envía su dirección IP privada al servidor (Figura 41) que posteriormente es enviada al usuario 1002 sin ser traducida (Figura 42).

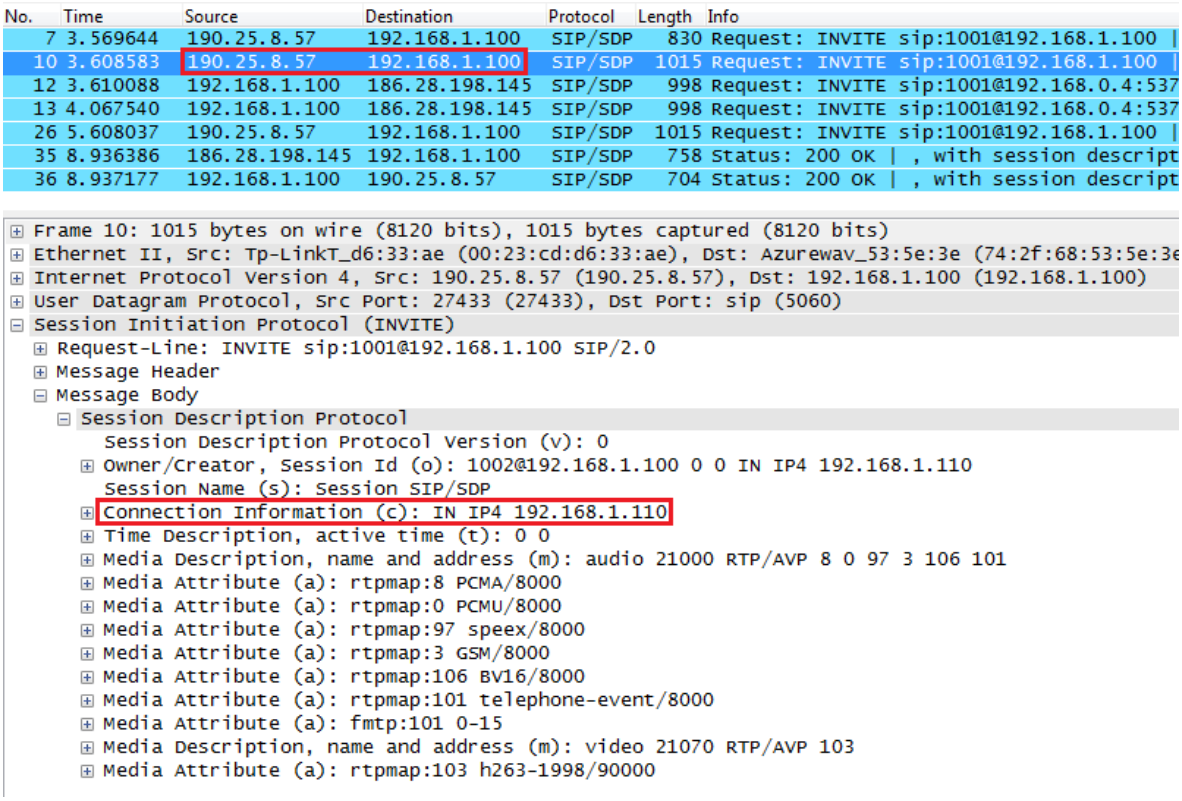


Figura 39. INVITE WiFi de 1002 al Servidor Kamailio.

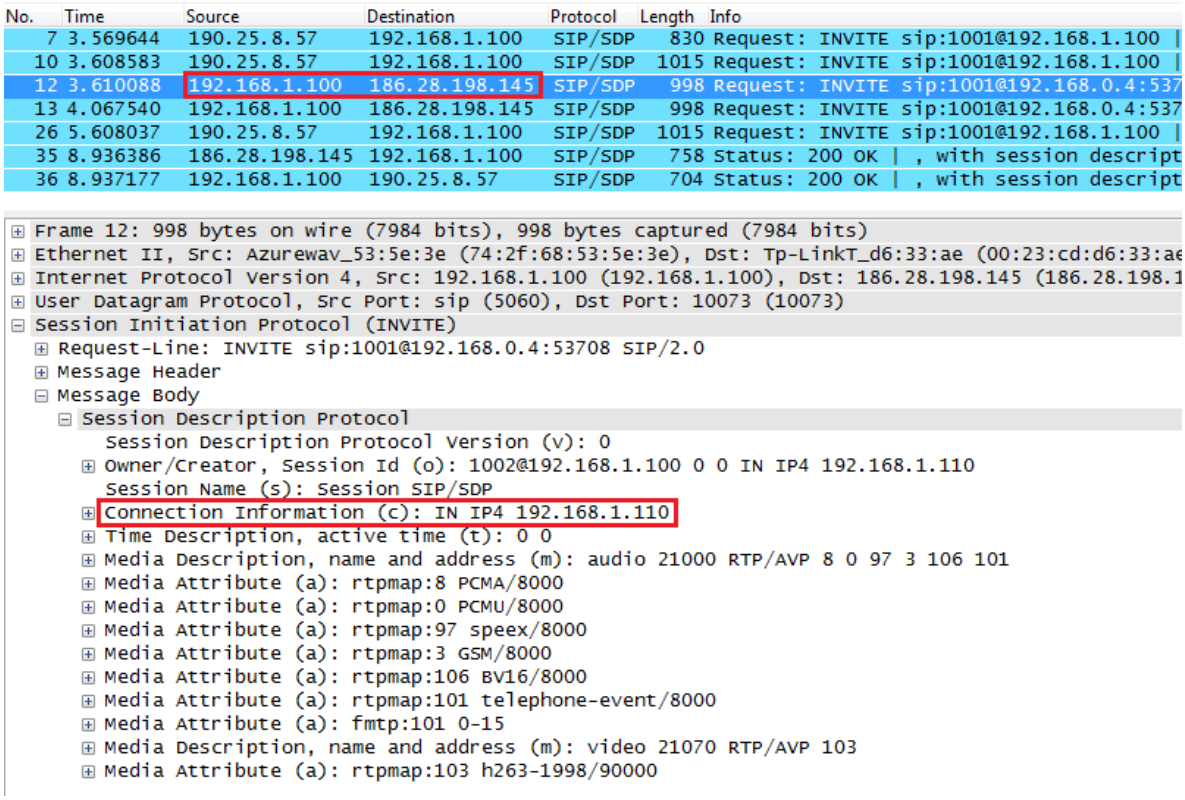


Figura 40. INVITE WiFi del Servidor Kamailio a 1001.

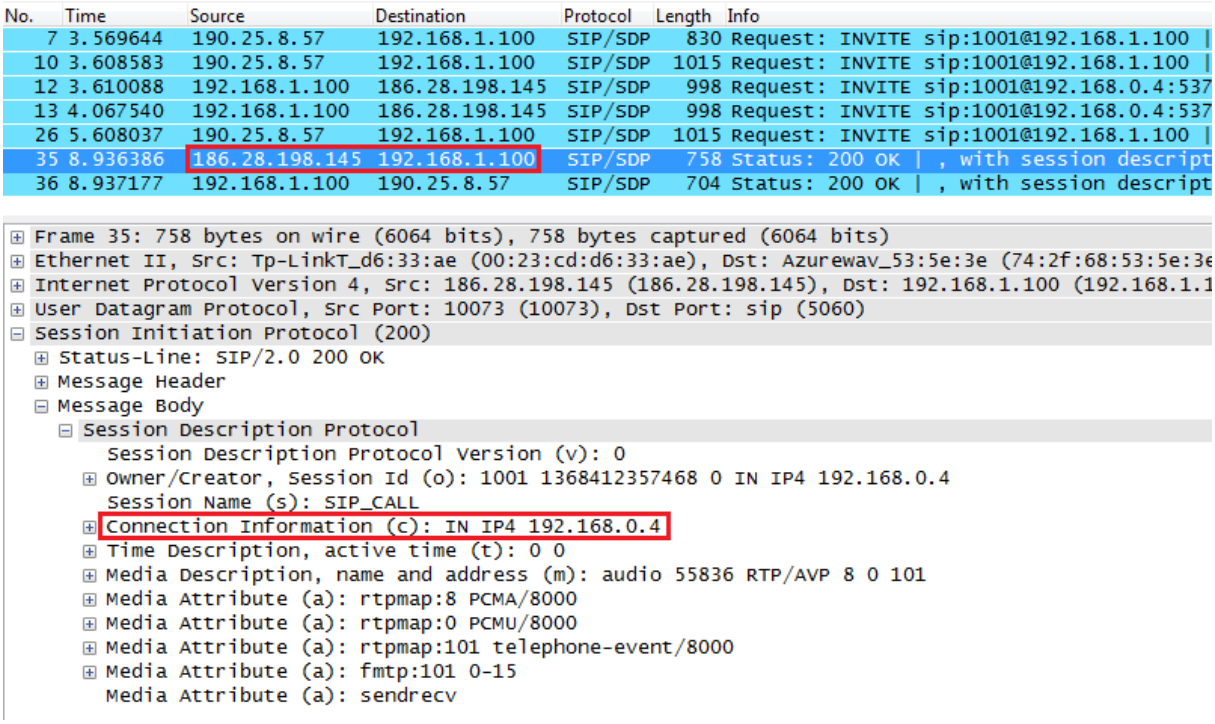


Figura 41. 200 OK WiFi de 1001 al Servidor Kamailio.

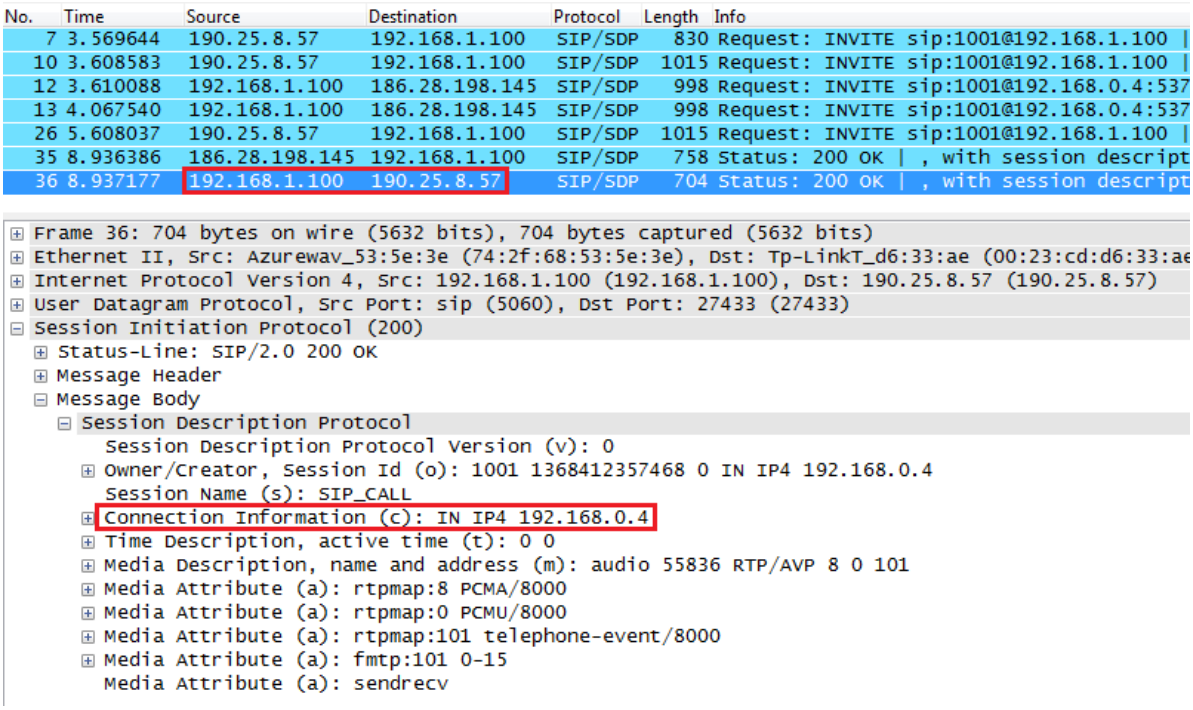


Figura 42. 200 OK WiFi del Servidor Kamailio a 1002.

Así, cuando se inicia la comunicación los paquetes de voz que envían ambos usuarios se pierden dentro de la red, ya que las direcciones IP privadas no existen en internet.

A pesar de los múltiples intentos de configuración del RTPproxy, que no se detallan en este documento, fue imposible acoplar su funcionamiento al del servidor Kamailio, por lo tanto se exploró la alternativa del servidor Elastix, obteniendo resultados satisfactorios que se muestran a continuación.

5.2. Resultados obtenidos utilizando el servidor Elastix

Elastix es el servidor SIP que permitió superar los problemas inherentes a la conexión de un dispositivo a través de un NAT, de tal forma que los paquetes de voz RTP sean enviados exitosamente a los usuarios, sin importar su ubicación dentro de la red.

Las pruebas realizadas muestran el funcionamiento del servidor durante el intercambio de mensajes SIP y durante la comunicación de una llamada de voz mediante el direccionamiento de los paquetes RTP. Además, se realiza la conmutación desde una red 3G a una red WiFi y viceversa, mostrando el envío del mensaje REGISTER por parte del cliente y la continuación de la llamada durante el Handover.

5.2.1. Conmutación de llamada – 3G a WiFi

En esta prueba, el usuario 1001 y el usuario 1002 se encuentran conectados a través de la red de datos 3G de dos operadores móviles. El usuario 1002 inicia la llamada por medio del envío al servidor de un mensaje INVITE que contiene en el SDP su dirección IP pública de contacto (Figura 43). El servidor modifica la información suministrada por el cliente 1002 y establece la dirección de contacto como su propia IP pública (Figura 44). Cuando el usuario 1001 recibe la solicitud de llamada, toma la IP pública del servidor como la dirección a la cual debe enviar los paquetes de voz durante la comunicación. El mensaje 200 OK que se envía al contestar la llamada (Figura 45) es igualmente modificado por el servidor, estableciendo la dirección de contacto del usuario 1001 como su propia IP pública (Figura 46).

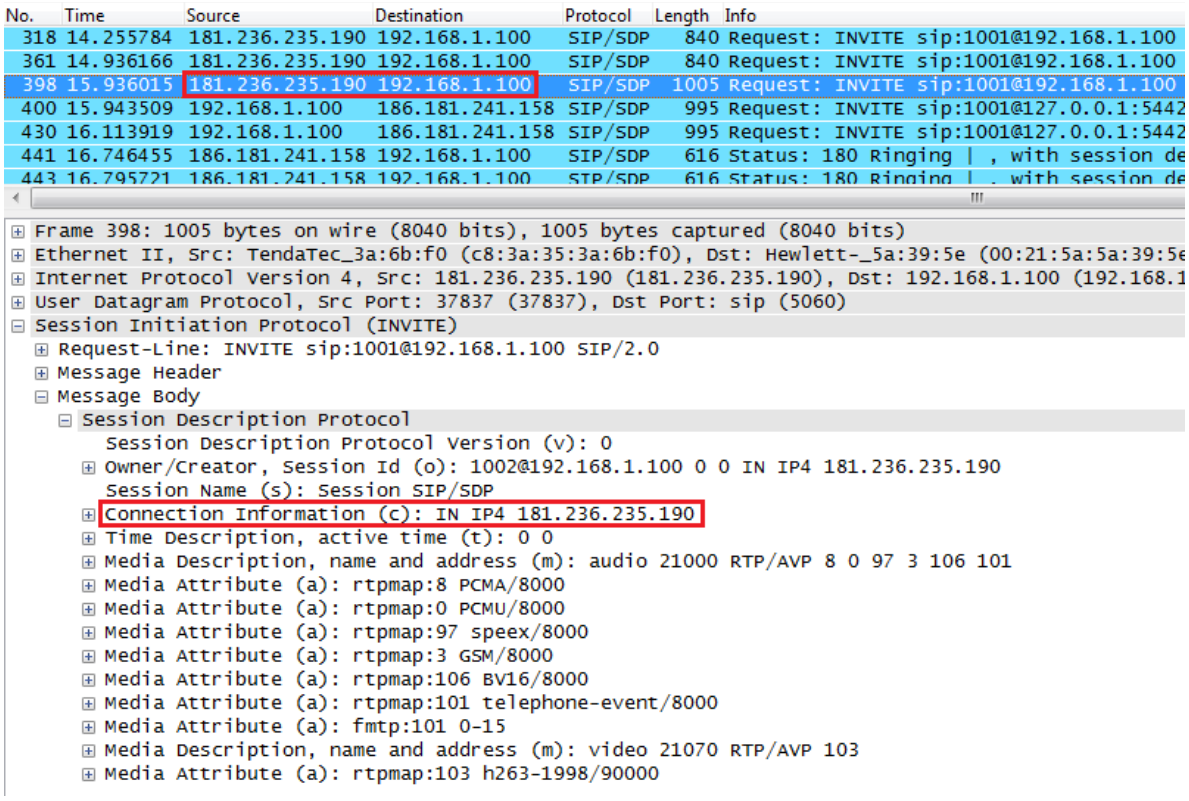


Figura 43. INVITE 3G de 1002 al Servidor Elastix.

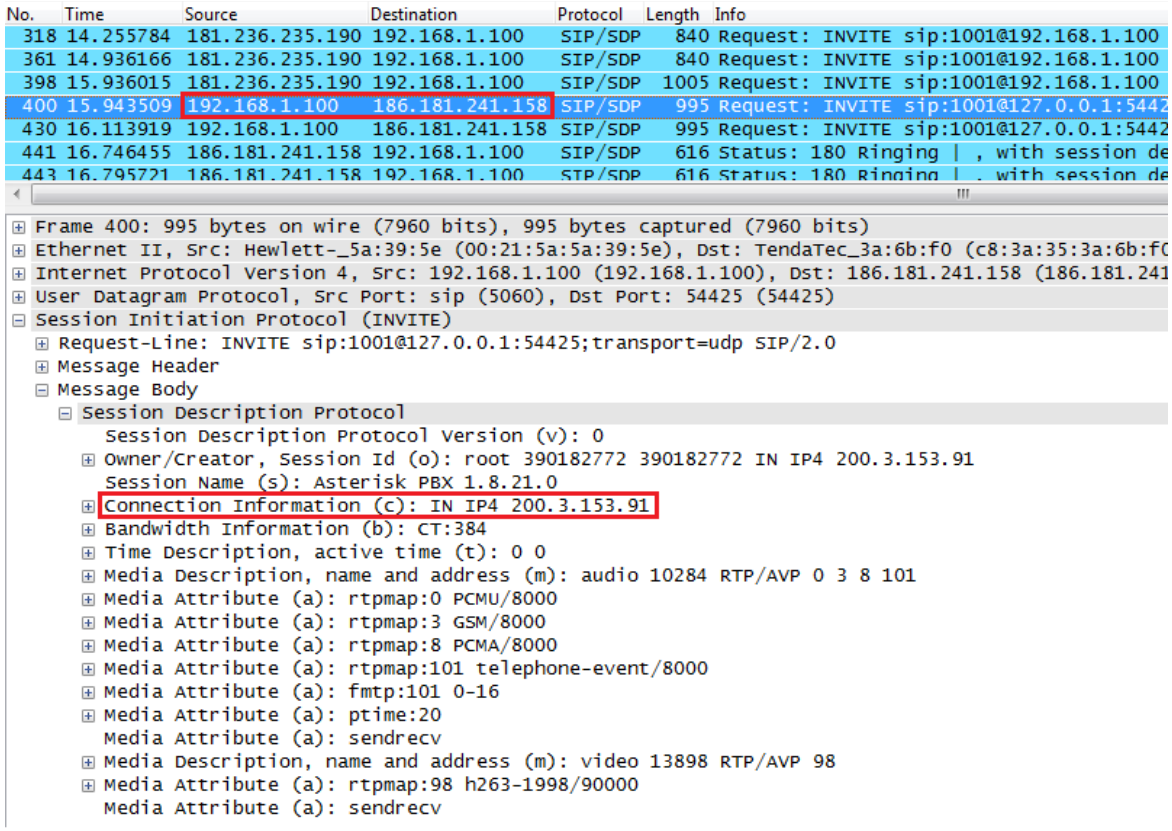


Figura 44. INVITE 3G del Servidor Elastix a 1001.

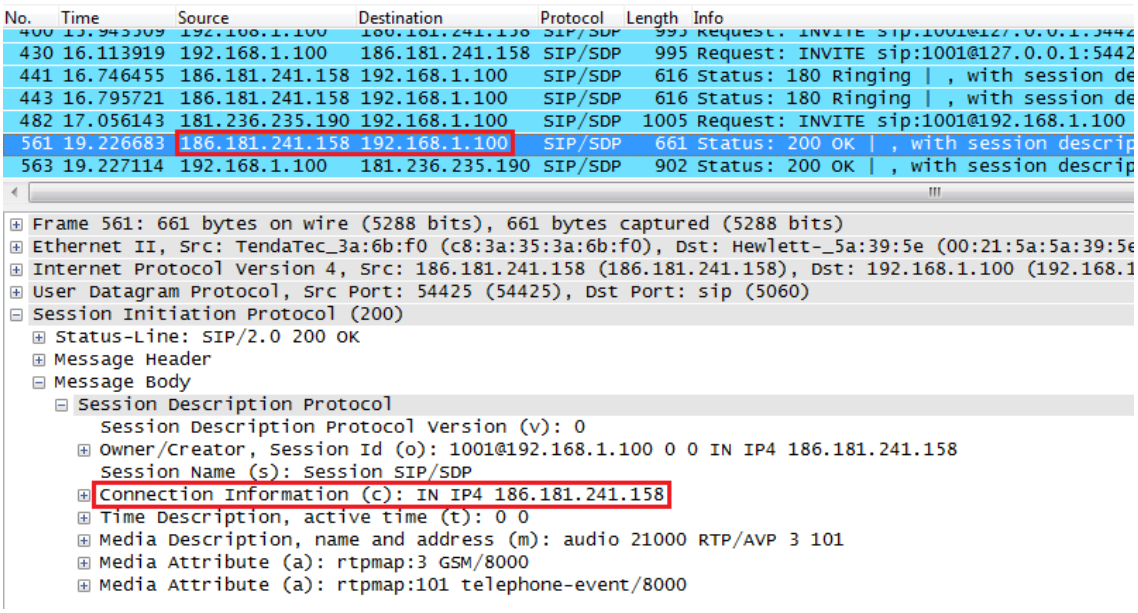


Figura 45. 200 OK 3G de 1001 al Servidor Elastix.

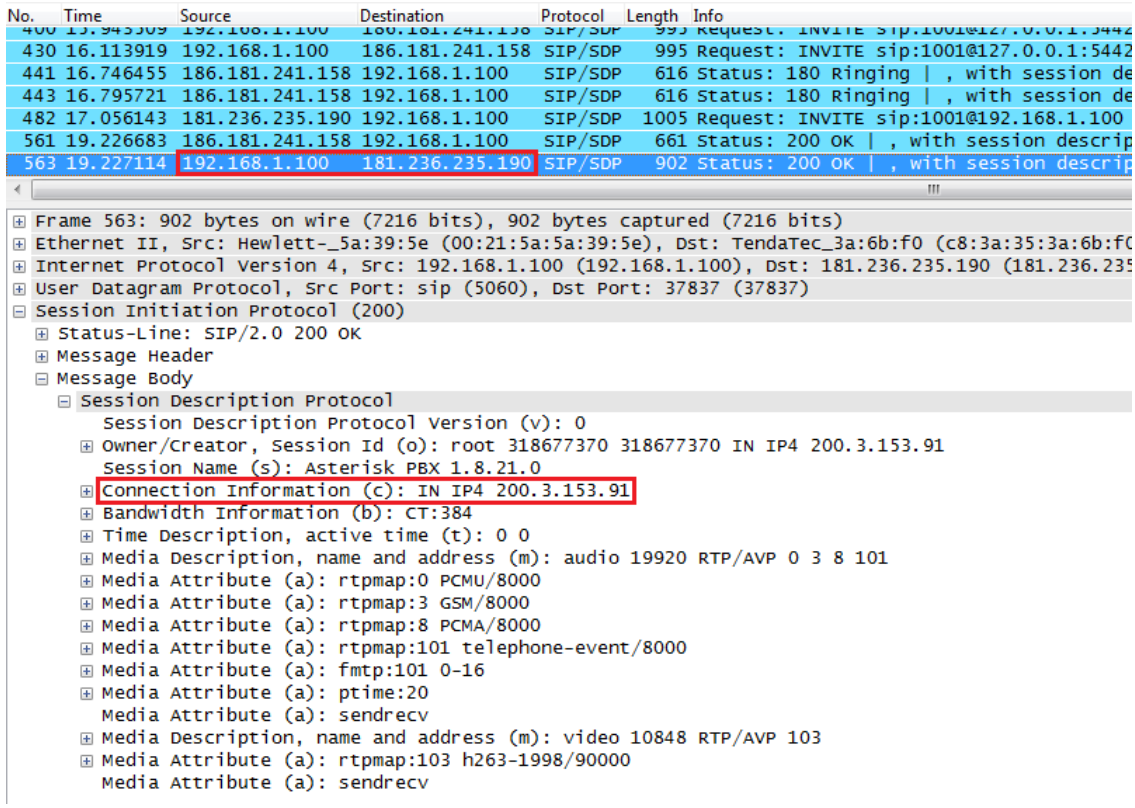


Figura 46. 200 OK 3G del Servidor Elastix a 1002.

Cuando la conexión se ha establecido y los paquetes de voz se están enviando correctamente desde un extremo de la comunicación al otro, el usuario 1001 conmuta y se conecta a una red WiFi.

La Figura 47 muestra el envío del mensaje REGISTER por parte del usuario 1001 al servidor, actualizando la dirección IP de contacto con la que había establecido la conexión a través de la red 3G. Al conectarse a una red WiFi, el usuario adquiere una dirección IP privada, pero dado que el otro usuario está enviando los paquetes de voz al servidor, este nunca se entera de la conmutación de red que efectúa el cliente 1001 y por lo tanto la llamada se mantiene activa.

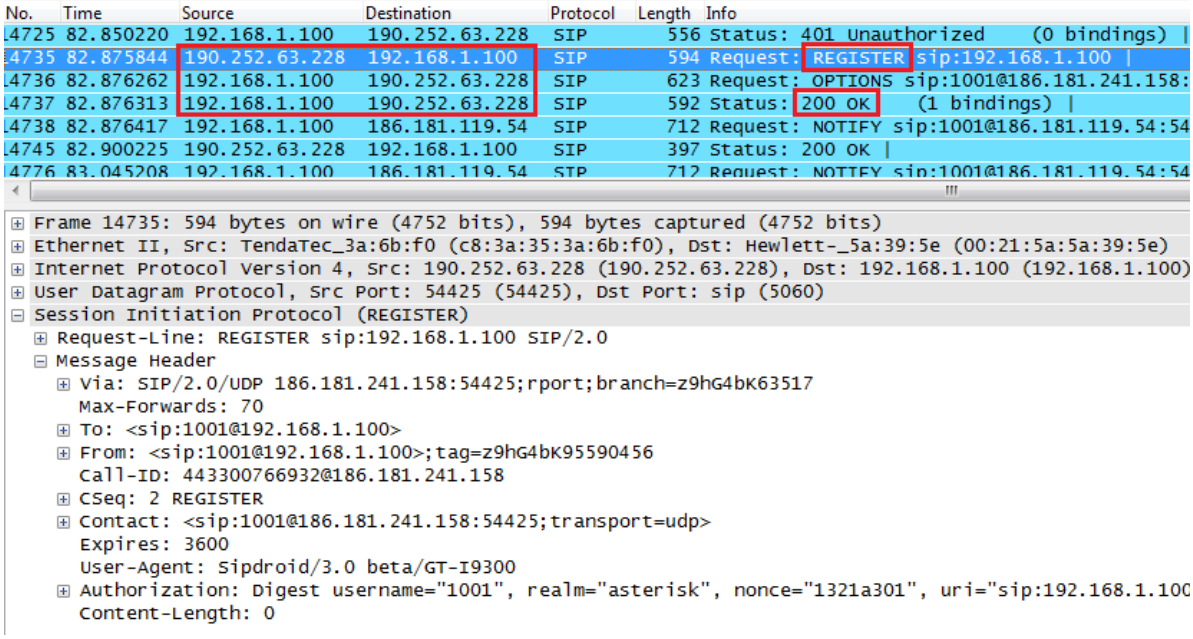


Figura 47. Mensaje REGISTER WiFi al conmutar de red.

El análisis gráfico del registro de la comunicación VoIP capturado en el servidor y generado por Wireshark, muestra el flujo de mensajes SIP durante el establecimiento, modificación y finalización de la llamada, comprobando el cambio de red por parte del usuario 1001 y el envío de los paquetes RTP al servidor (Figura 48).

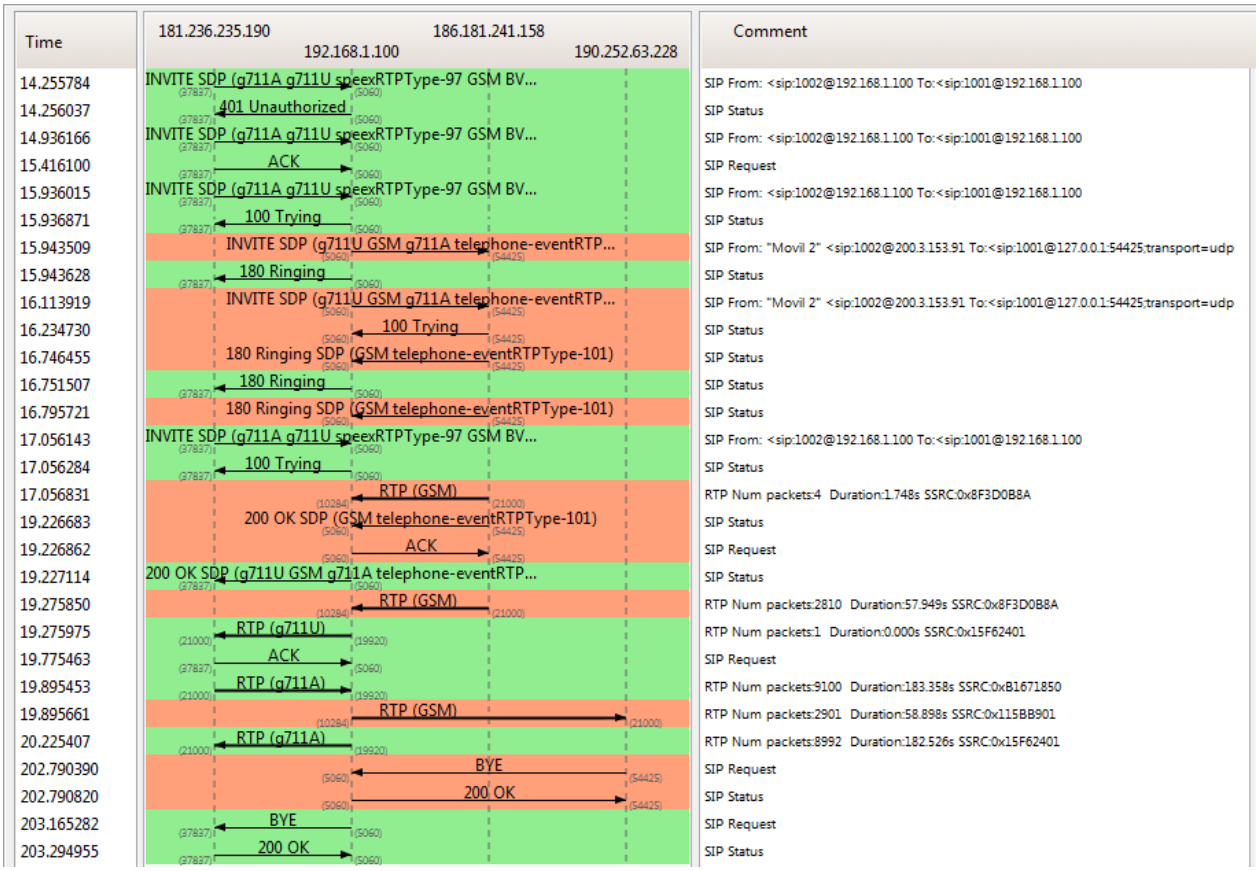


Figura 48. Análisis gráfico conmutación de red (3G a WiFi) Elastix.

Durante la llamada se obtuvo el log de calidad de la comunicación, capturado por SIPDroid por medio de las variables porcentuales de retraso y pérdida y el estimado del tiempo que tardan los paquetes en llegar al cliente. El resultado se observa en las siguientes figuras, en las cuales se puede observar el corto tiempo de Handover que es prácticamente imperceptible por el usuario.

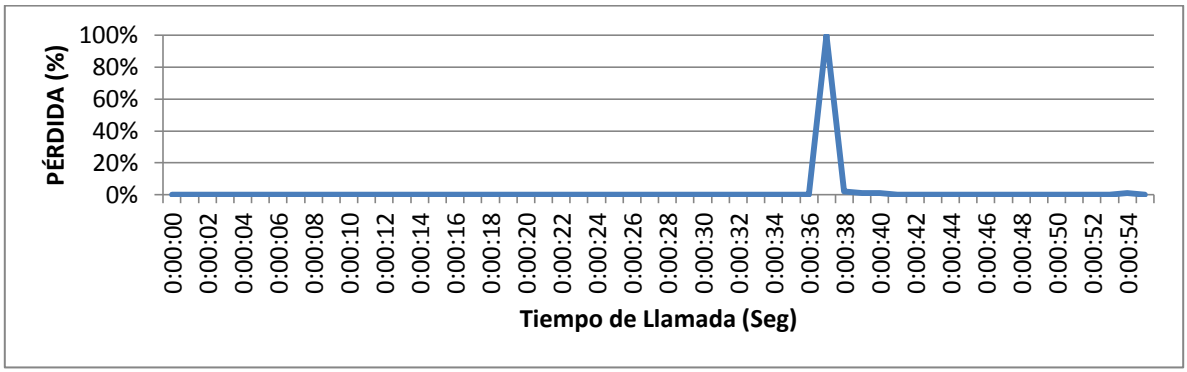


Figura 49. Porcentaje de Pérdida conmutación 3G – WiFi Elastix.

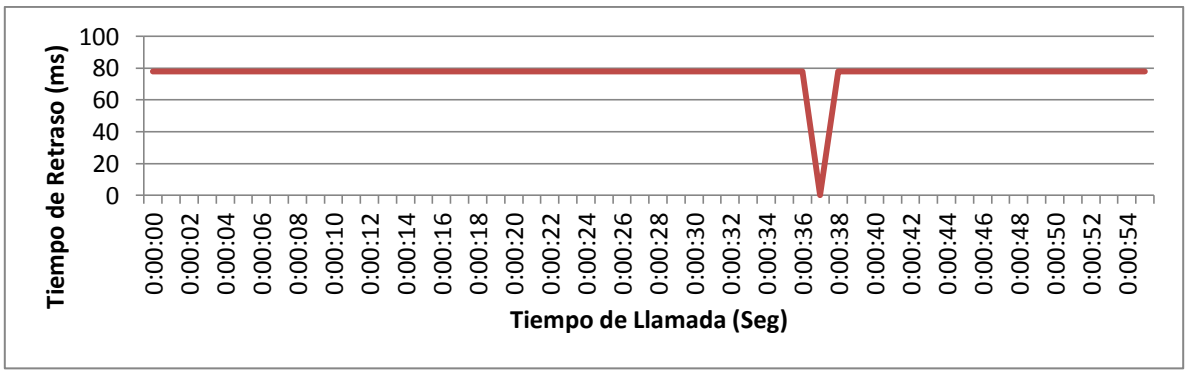


Figura 50. Tiempo de Retraso conmutación 3G – WiFi Elastix.

La Figura 49 muestra el porcentaje de pérdida de paquetes durante la comunicación. Allí se aprecia que este valor se mantiene en cero durante toda la llamada, con un pico que se presenta al momento de realizar la conmutación de una red a otra y en la Figura 50 se aprecia el tiempo de retraso, cuyo valor siempre es inferior a 80 milisegundos, mostrando un silencio de voz durante la conmutación.

En las dos figuras anteriores se puede notar el resultado de la conmutación de la red 3G a la red WiFi, cuya duración produce un efecto menor a los 2 segundos. La estabilidad en la llamada a través del servidor Elastix es considerablemente buena y el tiempo de retraso que se mantiene por debajo de los 80 milisegundos es el mínimo tiempo que la aplicación puede determinar, dados los efectos inherentes a la transmisión de paquetes por la red.

5.2.2. Conmutación de llamada – WiFi a 3G

El proceso de conmutación de una llamada desde la red WiFi a una red 3G es igual al explicado anteriormente en la sección 5.2.1, donde se detalla el proceso de conmutación de la llamada desde una red 3G a una red WiFi.

En este caso, el dispositivo que se encuentra dentro de la LAN de una red WiFi con una dirección IP privada, conmuta a una red 3G la cual le asigna una IP pública que identifica exclusivamente a esa terminal en internet.

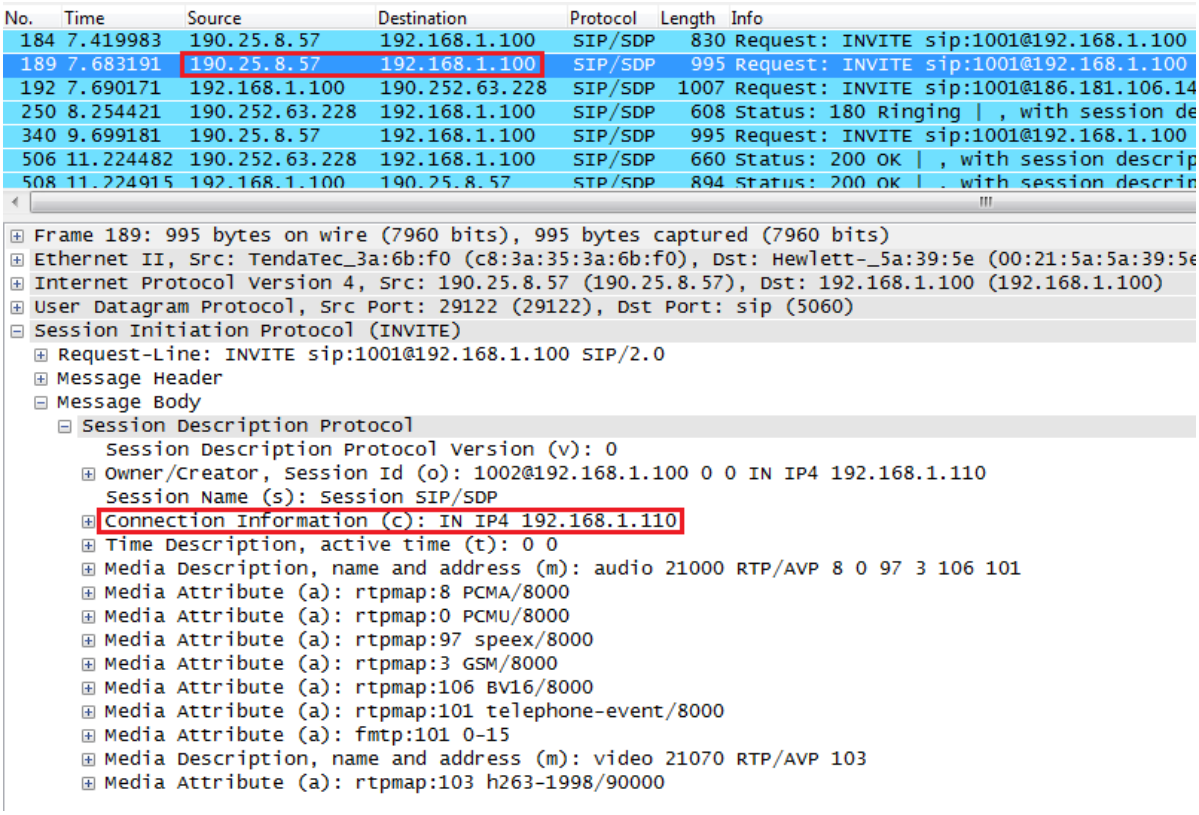


Figura 51. INVITE WiFi de 1002 al Servidor Elastix.

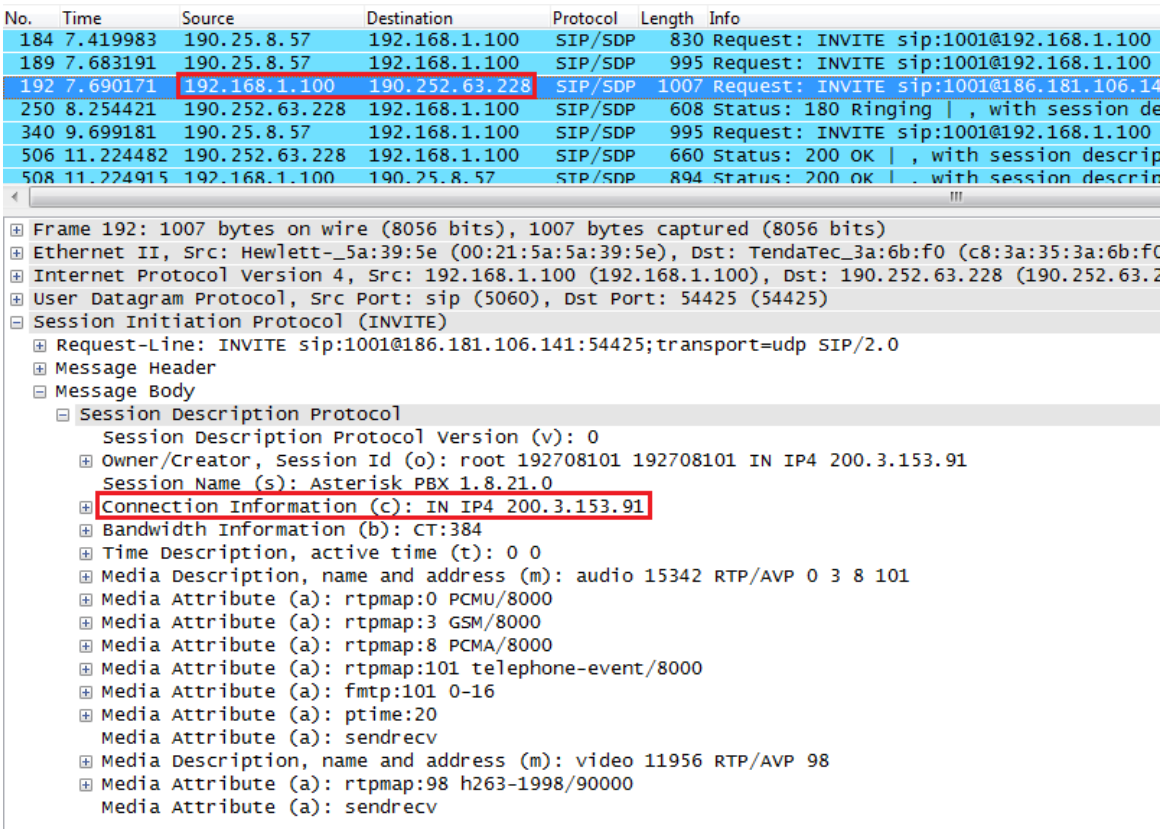


Figura 52. INVITE WiFi del Servidor Elastix a 1001.

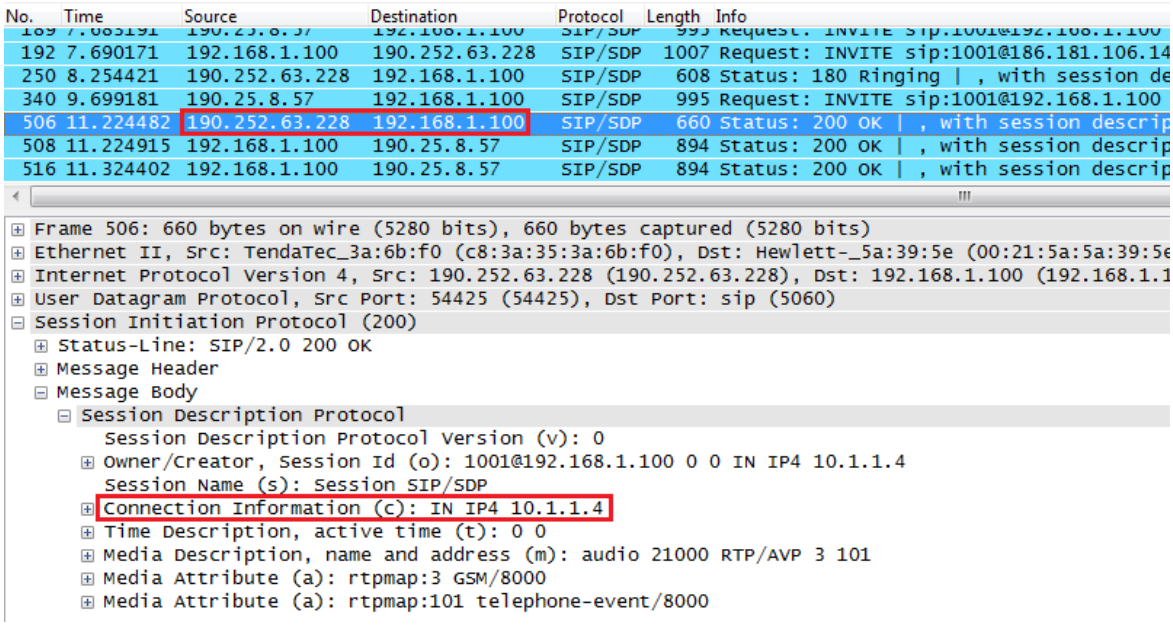


Figura 53. 200 OK WiFi de 1001 al Servidor Elastix.

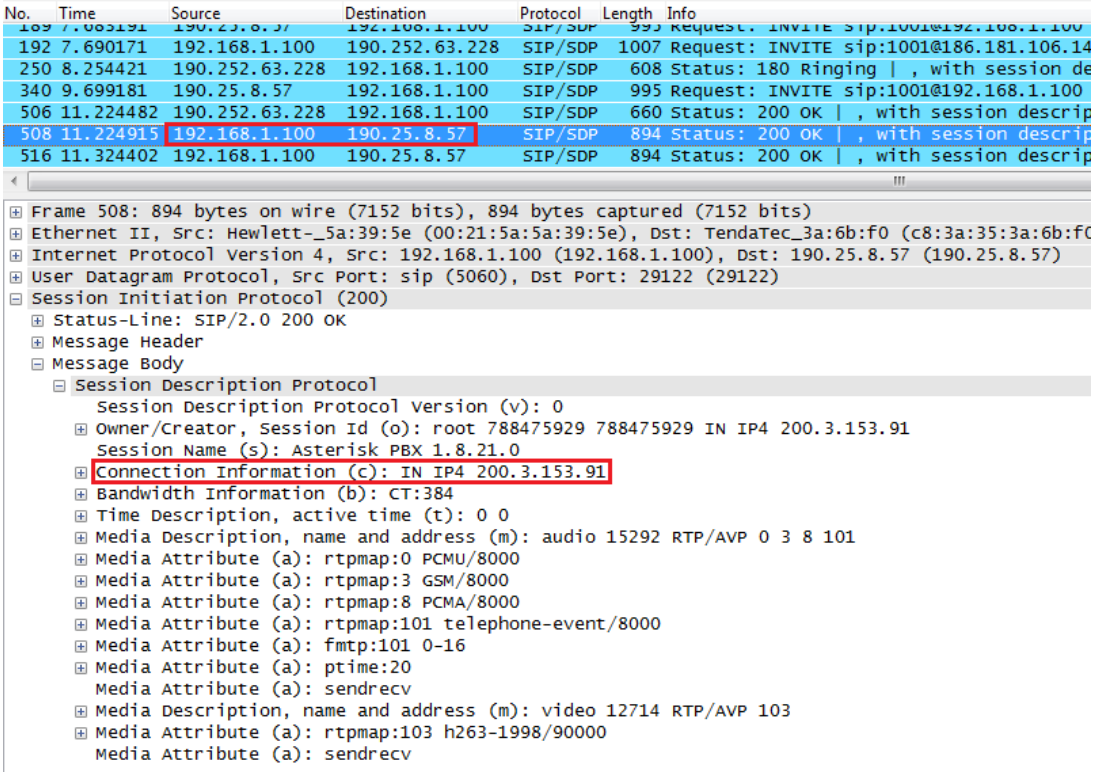


Figura 54. 200 OK WiFi del Servidor Elastix a 1002.

Las Figuras 51 y 52 son equivalentes a las Figuras 42 y 43 mostradas en la sección anterior. Su diferencia radica en que el mensaje INVITE que envía el usuario 1002 al servidor es enviado desde una red WiFi con su IP privada, la cual es traducida al igual que en la prueba de la sección 5.2.1, a la dirección IP pública del servidor antes que el mensaje sea redirigido al usuario 1001.

De igual forma, las Figuras 53 y 54 son equivalentes a las Figuras 44 y 45, donde se muestra el mensaje 200 OK que se envía al contestar la llamada.

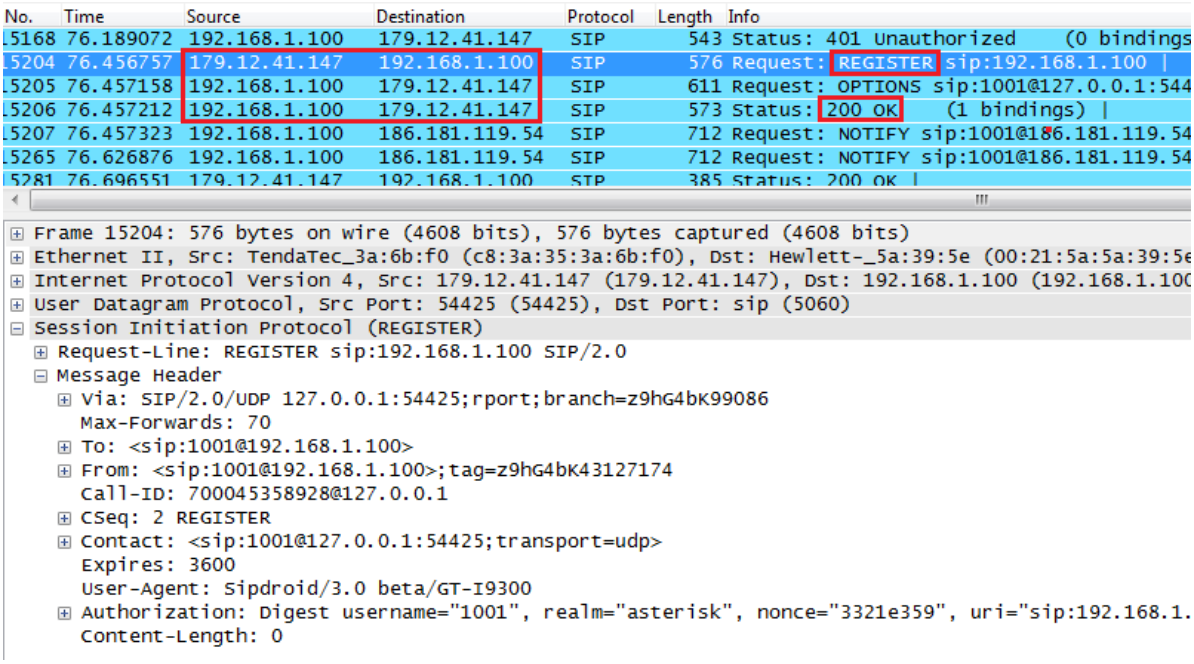


Figura 55. Mensaje REGISTER 3G al conmutar de red.

En la Figura 55 se muestra el mensaje REGISTER que envía el usuario 1001 al conmutar y conectarse a la red 3G. De esta forma, el servidor actualiza la información de contacto por la nueva dirección IP y continua enviando los paquetes RTP, sin que el usuario 1002 se entere de la nueva ubicación del usuario 1001 ni se afecte la comunicación.

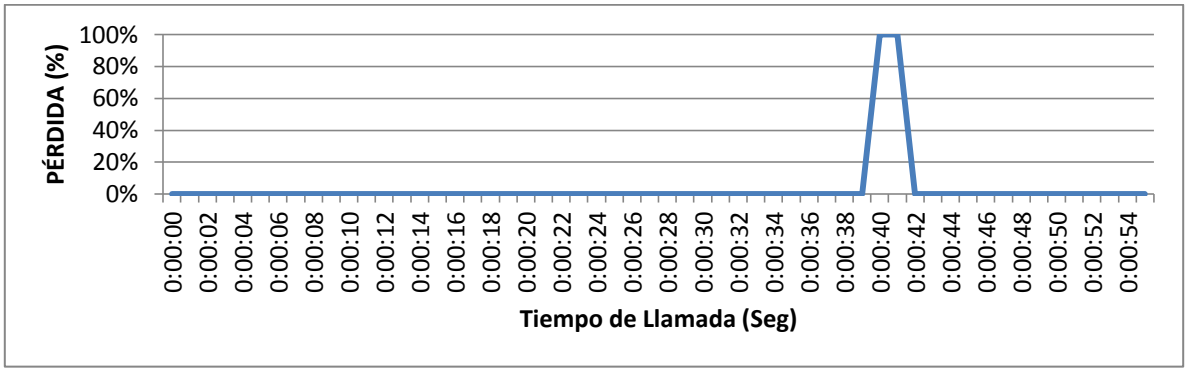


Figura 56. Porcentaje de Pérdida conmutación WiFi – 3G Elastix.

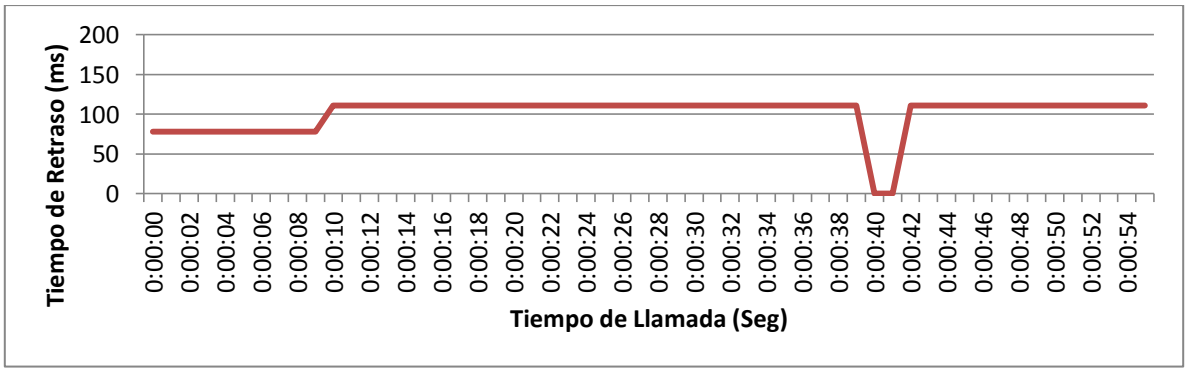


Figura 57. Tiempo de Retraso conmutación WiFi – 3G Elastix.

La Figura 56 muestra el porcentaje de pérdida de paquetes durante la llamada y la Figura 57 el tiempo aproximado que tarda la voz en llegar al cliente. En las dos figuras se puede apreciar el momento en el cual el usuario 1001 realiza la conmutación, provocando un efecto en la transmisión de datos cuya duración se mantiene inferior a los 2 segundos.

5.3. Limitaciones Red WiFi PUJ

Dado que el servidor SIP fue implementado dentro de la red interna de la Pontificia Universidad Javeriana, como se muestra en el diagrama de la Figura 5, se encontraron ciertas limitantes que impiden realizar la conmutación de una llamada de voz de forma automática, entre una red 3G y cualquier red inalámbrica dentro de la WAN de la universidad.

La dirección IP pública que fue dispuesta para la realización de este proyecto de grado, está conectada a una dirección IP privada dentro de la WAN de la universidad; por configuración de Firewall, al conectar un cliente a una red inalámbrica, no se permite la salida de conexiones desde la WAN a la IP pública suministrada y por esta razón no se puede establecer la comunicación entre el servidor y el cliente a través de internet.

Por otra parte, al conectar cualquier dispositivo móvil a una red WiFi de la PUJ, se requiere el registro del usuario en el portal cautivo que vigila el tráfico por la red desde y hacia internet, haciendo que la conmutación de la llamada no se efectué hasta que el usuario no realice el registro en el portal.

Para poder realizar la conmutación de una llamada de una red 3G a una WiFi o viceversa, es necesario contar con una red inalámbrica que no pertenezca a la WAN de la universidad.

6. CONCLUSIONES

En primera instancia es preciso mencionar que el desarrollo del presente trabajo de grado se llevó a cabo de forma satisfactoria y la solución al problema propuesto cumplió en su totalidad con los objetivos planteados en el proyecto.

La conmutación de una llamada de voz a través de un servidor SIP entre redes de diferentes características (Handover Vertical), como lo son las redes 3G y WiFi, se logró mediante la implementación de la central telefónica PBX que forma parte de la herramienta de comunicaciones del software Elastix. Mediante la implementación de la aplicación SIPDroid, su pudo realizar el registro de los clientes en el servidor y la inicialización, modificación y terminación de una llamada de voz.

El diseño de la estructura de red implementada permitió realizar el seguimiento de la señalización SIP para la verificación de la información contenida en el envío y recepción de mensajes por parte de los clientes y el servidor y el redireccionamiento de puertos permitió la transversalidad de NAT y el uso de una red privada para la ubicación del servidor mediante el establecimiento de la zona desmilitarizada (DMZ) en el enrutador al cual se conecta el servidor.

La comunicación entre los clientes y el servidor SIP a través de internet, represento en gran medida el principal problema a solucionar durante el desarrollo del proyecto, dadas las limitaciones que interpone el uso de NAT y la necesidad de forzar la transmisión de mensajes RTP desde y hacia el servidor por medio de la implementación de un proxy.

Mediante el reenvío del mensaje REGISTER y el enrutamiento de los paquetes RTP por parte del servidor, se logró actualizar la información de contacto del cliente que realiza la conmutación, sin perder la llamada y sin afectar la comunicación con el cliente en el otro extremo de la conexión.

La conmutación de la red en el dispositivo móvil se facilitó debido a las características del sistema operativo Android, el cual se encarga de la verificación de la estabilidad y el nivel de potencia de las señales inalámbricas y decide la red de transmisión de datos, disminuyendo el procesamiento dentro de la aplicación y permitiendo que ésta se limite exclusivamente a la comunicación.

El porcentaje de Handovers realizados exitosamente fue del 100%, esto dado que en ninguno de los procesos de conmutación se perdió la comunicación; el tiempo de conmutación durante la totalidad de las pruebas realizadas se mantuvo por debajo de los 2 segundos y los tiempos de retraso, inherentes a la red, nunca superaron los 200ms, brindando una experiencia del usuario confortable durante la comunicación.

Como continuación del presente trabajo de grado se propone implementar la comunicación por medio del protocolo IPv6 en la capa de red, con el fin de eliminar limitaciones del protocolo IPv4 como la traducción de direcciones por medio de NAT, facilitando así la transmisión de los datos multimedia y realizando el envío de los paquetes RTP directamente entre los clientes. Además se propone utilizar la red móvil de cuarta generación 4G, aumentando la velocidad de conexión a internet por parte de los usuarios y reduciendo los retardos provocados por la red.

7. BIBLIOGRAFÍA Y FUENTES DE INFORMACIÓN

http://www.kamailio.org/w/
http://www.elastix.org/
http://sipdroid.com/

[1] Ministerio de Tecnologías de la Información y las Comunicaciones; "Boletín trimestral de las TIC. Cuarto trimestre de 2012". Marzo 2013.Marzo 2013.

[2] Network Working Group; Request for Comments: 3261. "SIP: Session Initiation Protocol". Junio 2002.

[3] CAMARILLO, Gonzalo. SIP Demystified, p. 94-95. McGraw-Hill, 2002.

[4] Nopal, G; Grupo de Trabajo VoIP; "Resumen del Protocolo SIP", p. 1.

[5] JOHNSTON, Alan. SIP understanding the Session Initiation Protocol, p 17, 2. Introduction to SIP. Artech House, second edition, November 2003.

[6] LANDÍVAR, Edgar. Comunicaciones Unificadas con Elastix, p 93, 6. Elastix Overview. Second edition, May 2011.

[7] Asterisk Features; [Online]. Available: http://www.asterisk.org/get-started/features.

[8] MIERLA, Daniel; "Install And Maintain Kamailio v4.0.x Version From GIT" [Online]. Available: http://www.kamailio.org/wiki/install/4.0.x/git.

[9] Developer Tools. "Get AndroidSDK" [Online]. Available: http://developer.android.com/sdk/index.html.

FIGURAS

(1) JOHNSTON, Alan. SIP understanding the Session Initiation Protocol. [Imagen], p. 29, Figura 1.1: The Internet Multimedia Protocol stack.

(2) JOHNSTON, Alan. SIP understanding the Session Initiation Protocol. [Imagen], p. 31, Figura 2.3: SIP registration example.

(3) JOHNSTON, Alan. SIP understanding the Session Initiation Protocol. [Imagen], p. 27, Figura 2.2: SIP call example with proxy server.

8. ANEXOS

[1] Código fuente aplicación SIPDroid – SipUA. (CD del proyecto).
[2] Trazas Wireshark (CD del proyecto).
[3] Logs de Llamadas (CD del proyecto).