Universidad Autónoma Metropolitana
Unidad Azcapotzalco

Licenciatura en Ingeniería en Computación

Proyecto Terminal

Aplicación móvil para Telefonía IP

Cortés Díaz Fernando Javier 206309941

González Torres Jesús Alejandro 206307680

Trimestre 12-P

Asesor de Proyecto:
M. en C. Arturo Zúñiga López

Fecha de Entrega: 25 de Julio 2012

CONTENIDO

OBJETIVO GENERAL

Desarrollar una aplicación móvil para la plataforma *Android*[1] que permita realizar y recibir llamadas sobre el protocolo *SIP*[2].

OBJETIVOS PARTICULARES

- Diseñar e implementar el módulo de registro con el Servidor.
- Diseñar e implementar el módulo para realizar llamadas SIP.
- Diseñar e implementar el módulo para recepción de llamadas SIP.
- Diseñar e implementar el módulo de permisos (Wi-Fi[3], auricular, micrófono, SIP)
- Diseñar e implementar un módulo GUI[4] para la aplicación
- Integración de los módulos

[1] Android: Sistema Operativo de código abierto basado en Linux, diseñado para dispositivos móviles.
[2] SIP: Protocolo de señalización para establecer sesiones de comunicación en la telefonía IP.
[3] Wi-Fi. Tecnología inalámbrica que permite a un dispositivo electrónico el intercambio de datos. IEEE 802.11
[4] GUI Interfaz Gráfica de Usuario

INTRODUCCIÓN.

La comunicación es el proceso que da lugar al intercambio de información entre un emisor y un receptor. La información que se transmite recibe el nombre de mensaje y es enviado por el emisor desde un punto origen, en el destino se encuentra el receptor que se encargará de recibir y decodificar la información para interpretarla.

Si la información se intercambia entre humanos, ésta se puede transmitir en forma de sonidos, luz, texturas, señas, de manera tal que puedan ser detectadas por los sentidos primarios como oído, vista y tacto. Si la comunicación se realiza entre máquinas, antes se deben establecer los códigos y protocolos necesarios para asegurar que el mensaje pueda ser transmitido íntegro y libre de errores.

Desde el origen de la humanidad hemos tenido la necesidad de comunicarnos con el entorno y actualmente gracias al desarrollo tecnológico podemos encontrar una gama de opciones para hacerlo, desde telefonía convencional, la telefonía a través de internet Telefonía IP[5], o conectados a internet desde cualquier otro dispositivo.

La telefonía IP y las aplicaciones móviles son las tecnologías de interés para este proyecto, se aprovecharán el auge y los beneficios que ofrecen para establecer comunicación desde el hogar, el entorno empresarial o centros universitarios. Se propone una solución que implemente protocolos estandarizados como SIP (para establecer las sesiones en la Telefonía IP), una plataforma de código abierto como *Android* para telefonía móvil y lenguaje de programación *Java*[6].

En este proyecto se desarrolló una aplicación de software para la plataforma Android, que permite realizar y recibir llamadas de voz sobre SIP a teléfonos convencionales principalmente, también se puede establecer comunicación con *Softphones*[7] y otros dispositivos móviles que tengan instalada la aplicación. Una solución empresarial que sirve como una alternativa a las aplicaciones existentes que no permiten llamadas a teléfonos analógicos y otras más que son de software propietario.

Para el desarrollo de la aplicación se diseñaron módulos de programación con base en los objetivos particulares del proyecto. Los objetivos son

- Diseñar e implementar el módulo de registro con el Servidor.
- Diseñar e implementar el módulo para realizar llamadas SIP.
- Diseñar e implementar el módulo para recepción de llamadas SIP.
- Diseñar e implementar el módulo de permisos (Wi-Fi, auricular, micrófono, SIP)
- Diseñar e implementar un módulo GUI para la aplicación

[5] Telefonía IP. Servicio que ofrece intercambio de voz, video y datos a través de una red paquetes IP
[6] Java: Lenguaje de programación de alto nivel, diseñado por James Gosling y Sun Microsystems en 1995
[7] Softphone: Software que emula un teléfono convencional adaptado para entornos IP

- Integración de los módulos

Cada uno cumple una función específica y son trascendentales para establecer la comunicación con un servidor, que permitirá realizar una llamada o recibirla.

Para realizar las observaciones, pruebas y resultados del correcto funcionamiento de la aplicación móvil se implementó una Red LAN[8] para la interacción entre los diferentes dispositivos, formada por un *PBX*[9] implementado en software *Asterisk*[10] que cumple con la función de central telefónica en un Servidor, un Teléfono convencional, un *Softphone* que emule un teléfono IP[11] en una computadora, y nuestro dispositivo móvil conectado a la red a través de *Wi-Fi.*

[8] LAN: Red de Área Local, para intercambio de información en distancias cortas
[9] PBX: Dispositivo de telefonía que actúa como conmutador en una red telefónica.
[10] Asterisk: Plataforma telefónica implementada en software de código abierto, actúa como central telefónica.
[11] Teléfono IP. Teléfono similar a uno convencional, pero es adaptado para entornos IP.

JUSTIFICACIÓN

La evolución de hardware y el desarrollo de software en la actualidad se ve reflejada en las formas de comunicarnos, el desarrollo de internet gracias a la fibra óptica, las computadoras personales y portátiles, o los dispositivos móviles como *Smartphones*[12] o *Tablets*[13] son sin duda una muestra de los alcances del desarrollo tecnológico. En las empresas, universidades o en el hogar tenemos la necesidad de informarnos y comunicarnos con las personas sin importar donde nos encontremos, por tanto una de las alternativas para hacerlo son los dispositivos móviles, son portables y muy prácticos.

Por otra parte la telefonía IP es una tecnología que ha ido madurando de la mano de internet, ésta ofrece una gran variedad de servicios, la llamada de voz, vídeo, tres a la vez, llamada en espera, buzón de voz, envío de datos, todo a través de internet y con gran calidad. Estos servicios son una forma eficiente de comunicación y principalmente se aprovecha en el entorno empresarial. Las empresas cuentan con gran número teléfonos convencionales, se pensaría que para comunicarse deberían tener una línea telefónica contratada para cada uno, sin embargo la telefonía IP, hace posible que la voz viaje en una red como por ejemplo el internet, por otra parte los teléfonos tradicionales se integran a la red con dispositivos que convierten las señales analógicas a digitales. Además las empresas, tienen una central telefónica que sirve como conmutador, para asignar extensiones a cada teléfono convencional, sin necesidad de contratar muchas líneas telefónicas.

Para este proyecto terminal se eligió al sistema Android ya que actualmente hay una gran variedad de fabricantes que lo ofrecen en sus productos, además existe una página oficial de esta plataforma con la documentación para desarrolladores y proporciona una API específica para el protocolo *SIP*, el cual nos permitirá establecer la comunicación entre los dispositivos.

Existen aplicaciones gratuitas que permiten llamadas *SIP*, pero no son capaces de llamar a teléfonos tradicionales. Dentro del software privativo, esta *Mobile* de *CISCO*[14], que si permite llamar a teléfonos convencionales y utiliza diferentes protocolos para realizar llamadas.

Por lo anterior nace el interés de desarrollar una aplicación para la plataforma *Android* que permita implementar la telefonía IP con el protocolo SIP, para realizar llamadas de voz a través de una red LAN sin costo adicional, principalmente capaz de comunicarse con teléfonos convencionales, *softphones* y también permitiendo la comunicación entre dos dispositivos móviles que tengan esta aplicación instalada. En el entorno laboral si

[12] Smartphone: Teléfono inteligente con mayor capacidad de computación y conectividad que un teléfono móvil convencional
[13] Tablet: Dispositivo móvil similar a una computadora portátil, interacción con las aplicaciones a través de una pantalla táctil.
[14] Mobile CISCO: Aplicación móvil de CISCO que permite realizar llamadas sobre Internet

dos oficinistas cuentan con esta aplicación en su dispositivo Android podrán comunicarse dentro de la empresa marcando la extensión asignada por el servidor, y este se encarga de desempeñar funciones de conmutador y central telefónica, en nuestro caso se implementó en Asterisk.

ANTECEDENTES

Referencias Internas.

En la UAM Azcapotzalco.

"Implementación de *VoIP*[15] en una *VPN*[16] por los alumnos Muñoz Salgado Luis Roberto y Pérez Lovera Ulises Arturo y como asesor de proyecto el M. en C. Arturo Zúñiga López.

En general los alumnos implementan 2 redes Virtuales Privadas (*VPN*) con protocolos de seguridad diferentes *SSL*[17] e I*PSec*[18] respectivamente. Envían señales de voz en forma de paquetes de datos (*VoIP*), para la conexión usan el protocolo SIP y se compara bajo que protocolo de *VPN* se transmite la voz con mayor calidad y que tiempo requiere.

En este proyecto se usará el Protocolo de Inicio de Sesión (*SIP*). La aplicación permitirá realizar llamadas desde una aplicación móvil, a teléfonos IP y convencionales, sin usar protocolos de seguridad ni asegurar la calidad de la comunicación. Se implementará Telefonía IP para este tipo de comunicación, diferente a la *VoIP*. Las pruebas son en una Red *LAN* con componentes de funcionalidad diferente, como el *PBX* para conmutación de llamadas, *softphone*, y teléfonos convencionales.

"Programación de Administrador de Llamadas Telefónicas Vía *Modem*" por Arroyo Gómez José Roberto y como asesor de proyecto Dr. Oscar Herrera Alcántara

En la propuesta anterior se desarrolla una aplicación que administra las llamadas telefónicas entrantes, esto es posible a través de un módem hospedado en una computadora conectado a la línea telefónica, se extrae el número de teléfono para búsqueda de información del contacto almacenada en una base de datos, con el fin de desplegar en pantalla del equipo de cómputo, un mensaje con la información del contacto que realizó la llamada.

La diferencia es que la aplicación que se desarrolla en el presente proyecto no es un administrador de llamadas, es una aplicación móvil que permitirá realizar y recibir llamadas con otros dispositivos, teléfonos Convencionales y Teléfonos I

15 VoIP: Protocolo de Voz para la transferencia de mensajes de audio auxiliándose de direcciones IP.

16 VPN: Red Privada Virtual.

17 SSL: Protocolo de capa de conexión segura.

18 IPSec: Protocolo de seguridad de Internet.

Referencias Externas.

En 2003 el danés Janus Friis y el sueco Niklas Zennström desarrollan *Skype*. Software que permite comunicación de texto, voz y vídeo sobre Internet (*VoIP).* Los códigos y protocolos permanecen cerrados y propietarios. Los usuarios pueden descargar la aplicación de manera gratuita. La aplicación permite realizar llamadas a teléfonos convencionales pero implica un costo.

Fring una aplicación privativa para dispositivos móviles, que permite realizar llamadas de voz, mensajería instantánea, vídeo llamada. Se conecta a las principales redes de *VoIP,* como *Skype, Google Talk*[19] *y MSN*[20]*.*

Mobile de CISCO, una solución empresarial para dispositivos móviles que proporciona servicios de llamadas de voz a teléfonos fijos, inalámbricos y softphones, se creó para mejorar la agilidad y productividad de los empleados sacando provecho de la Telefonía IP y las redes *LAN y WLAN*[21], esta aplicación es software propietario, por lo tanto, no puede modificarse por los usuarios. La versión 8.1 fue lanzada en 2011 para *iPhone* de Apple.

Otros software que aprovechan los beneficios de la Telefonía IP desde un equipo de cómputo son los Softphones, aplicaciones que emulan las funciones de teléfonos IP, permiten realizar llamadas, vídeo llamadas a través de internet.
Algunos ejemplos de estos software: *Ekiga, Linphone, QuteCom, X-Lite.*

19 Google Talk: Cliente de mensajería instantánea y VoIP.
20 Colección de servicios de internet ofrecidos por Microsoft
21 WLAN: Sistema de comunicación de datos inalámbrico.

FUNDAMENTOS TEÓRICOS.

Android

Android es un entorno de software de código abierto, creado para dispositivos móviles. No es una plataforma de hardware. Incluye un sistema operativo basado en Linux, una completa interfaz de usuario, *middleware*, bibliotecas de código, estructuras para aplicaciones, compatibilidad multimedia y mucho más.

Núcleo de Linux

Android se basa en núcleo de Linux versión 2.6 para los servicios básicos del sistema como la seguridad, la gestión de memoria, gestión de procesos entre otros, El núcleo de Linux ofrece agilidad y portabilidad para aprovechar las numerosas opciones de hardware de los teléfonos equipados con Android.

Las aplicaciones dependen del núcleo de Linux para servicios como procesadores, memoria y administración de sistemas de archivos.

Bibliotecas

El conjunto de bibliotecas en C/C++ usadas por diferentes componentes del sistema de Android se describen a continuación:

- **System C Library** una implementación derivada de la distribución de software Berkeley de C (libc), diseñado para dispositivos embebidos basados en Linux.
- **Media Libraries** Las bibliotecas soportan reproducción y grabación de diferentes formatos de audio y video incluyendo *MPEG4, H.264, MP3, AAC, AMR, JPG y PNG*
- **Surface Manager** gestiona el acceso al subsistema de la pantalla, y sin problemas puede trabajar con las capas graficas 2D y 3D de las diferentes aplicaciones.
- **LibWebCore** un moderno motor de búsqueda web.
- **SGL** el motor de base de gráficos 2D
- **SQLite** un motor de base de datos relacional potente y ligero, disponible para todas las aplicaciones

En la siguiente figura se muestran los elementos de la arquitectura Android

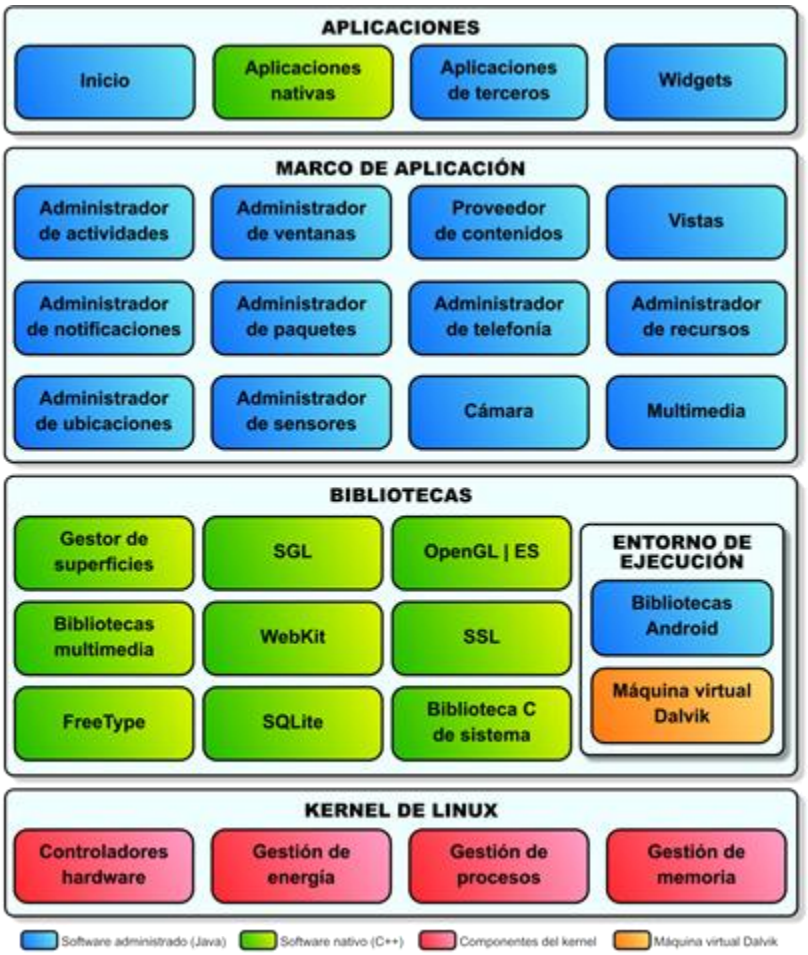


Figura 1. Arquitectura de Android.

Máquina Virtual Dalvik

Las aplicaciones se escriben en Java y se compilan en código de bytes, para después traducirse a una representación diferente denominada archivos dex. Estos archivos permiten que Android ejecute las aplicaciones en su propia máquina virtual. Después de la compilación se obtiene un archivo con extensión *.apk*, todo el código de este archivo es considerado una aplicación y es usado por Android para instalarla en el dispositivo.

Una vez instalada una aplicación *.apk,* cada aplicación de Android vive en un entorno seguro.

- El sistema operativo *Android* es un sistema *Linux* multiusuario en donde cada aplicación es un usuario diferente.
- Por default, el sistema asigna a cada aplicación un identificador de usuario. El sistema establece los permisos necesarios para que solo el identificador de usuario asignado a esa aplicación pueda acceder a estos archivos.
- Cada proceso tiene su propia Maquina Virtual (VM), de esta forma el código de una aplicación se ejecuta en forma aislada de otras aplicaciones.
- Por default cada aplicación se ejecuta en su propio proceso *Linux*. Android inicia un proceso cuando alguno de los componentes de la aplicación necesita ser ejecutado, después termina el proceso cuando ya no se necesita o cuando el sistema necesita memoria para otras aplicaciones.

De esta manera el sistema *Android* implementa el principio de privilegios mínimos, lo que significa que cada aplicación por defecto tiene acceso únicamente a los componentes que requiere para realizar su tarea y no más. Esto crea un ambiente muy seguro en el que una aplicación no puede acceder a partes del sistema a los cuales no se le han otorgado permisos.

Componentes de una aplicación.

Cada componente es un punto por el cual el sistema puede entrar a una aplicación. No todos los componentes son puntos de entrada para un usuario, algunos dependen el uno del otro, cada uno es una pieza única que ayuda a definir el comportamiento general de una aplicación.

Hay cuatro diferentes tipos de componentes en una aplicación. Cada tipo sirve para diferentes propósitos y tienen diferentes ciclos de vida que definen cuando la componente es creada y destruida.

Activities

Una actividad representa una sola pantalla con una interfaz de usuario. Por ejemplo, una aplicación de correo electrónico puede tener una actividad que muestra una lista de correos electrónicos nuevos, otra actividad para escribir un correo nuevo, así como otra actividad para leer un correo. Aunque las actividades trabajan juntas para dar una experiencia al usuario coherente en la aplicación de correo, cada una es independiente de las demás.
Una actividad es implementada como una subclase de una actividad.

Services

Un servicio es una componente que se ejecuta en segundo plano para realizar operaciones de larga duración o para realizar tareas para procesos remotos. Un servicio no cuenta con una interfaz de usuario. Por ejemplo, un servicio puede reproducir música en segundo plano mientras que el usuario está en una aplicación diferente. Otro componente como una actividad, puede iniciar un servicio y dejarlo en ejecución o vincularse a él para interactuar con este.
Los servicios son implementados como subclases de un servicio.

Content providers

Los proveedores de contenido gestionan un sistema compartido de datos de una aplicación, donde se puede almacenar datos en el sistema de archivos, en una base de datos *SQLite*, en internet, o en cualquier otro lugar de almacenamiento persistente que una aplicación puede tener acceso. A través de un proveedor de contenido otras aplicaciones pueden consultar o modificar datos. Por ejemplo, el sistema *Android* ofrece un proveedor de contenidos que gestiona la información de los contactos del usuario. Cualquier aplicación con los permisos necesarios puede consultar o modificar información de una persona en particular.
Un proveedor de contenido se implementa como una subclase de *ContentProvider.*

Broadcast Receivers

Un receptor de *Broadcast* es un componente que responde a mensajes de difusión en todo el sistema. Muchos de los *broadcast* se originan en el sistema. Por ejemplo, cuando se ha apagado la pantalla, cuando la batería se encuentra a punto de agotarse. Las aplicaciones también pueden iniciar un *Broadcast*. Por ejemplo, cuando otras actividades se enteran que cierta información ha sido descargada al dispositivo y esta lista para ser usada. Aunque un recibidor *Broadcast* no muestra una interfaz de usuario, puede crear una barra de notificación de estado para alertar a un usuario cuando un evento ha ocurrido.

Intents

Tres de los componentes básicos de una aplicación, *Activities*, Servicios y *Broadcast* Receivers, son activados por medio de mensajes, los cuales son llamados *Intents*. Estos mensajes facilitan la comunicación entre la misma aplicación y entre diferentes

aplicaciones en tiempo de ejecución. Un *Intent* es una estructura que contiene una descripción abstracta de una operación a ser realizada o en el caso de los *Broadcast*, es una descripción de que algo que ha ocurrido.

Existen mecanismos separados para la entrega de *Intents* para los diferentes tipos de componentes en Android:

- Un objeto *Intent* se pasa a este método *Context*.*startActivity*() para iniciar una actividad u obtener una actividad existente para realizar algo nuevo.
- Un objeto *Intent* se pasa a este método *Context*.*startService*() para iniciar un servicio o para pasar nuevas instrucciones a un servicio en ejecución.
- Un objeto *Intent* se pasa a diferentes tipos métodos *Broadcast* como: *Context*.*sendBroadcast*(), *Context*.*sendOrderedBroadcast*() o *Context*.*sendStickyBroadcast*(), el cual se enviado a todos los *broadcast* interesados.
  En cada caso el sistema Android encuentra la Actividad, Servicio o *Broadcast* Receiver que responda a cada intento, instanciándolos si es necesario. No hay superposición en este tipo de mensajes, los intentos *Broadcast* solo se entregan a los *Broadcast Receivers*, nunca a Actividades o Servicios. Un intento pasado al método *startActivity*() se entrega solo a una actividad, nunca a un *Broadcast Receiver* o a un servicio, y así sucesivamente.

Gestión del ciclo de Vida de una Actividad

El manejo del ciclo de vida de las actividades se realiza por medio de la implementación de métodos los cuales son cruciales para el desarrollo de aplicaciones fuertes y flexibles. El ciclo de vida de una actividad es directamente afectada por su asociación con otras actividades, sus tareas y la pila de retorno. En una actividad pueden existir esencialmente tres estados:

Resumed:

La actividad se encuentra en primer plano de la pantalla y es visible para el usuario.

Paused:

Otra actividad se encuentra en primer plano y es visible al usuario. Es decir, otra actividad es visible por encima de la actividad principal y esta actividad puede ser semitransparente o puede no cubrir toda la pantalla. Una actividad pausada se mantiene en la memoria, mantiene toda la información de su estado y se mantiene unida al gestor de ventanas. El sistema puede terminar esta actividad si no hay suficiente memoria.

Stopped:

La actividad principal se encuentra totalmente cubierta por otra actividad, la actividad principal se encuentra ahora en segundo plano. Una actividad detenida se mantiene en la memoria y mantiene toda la información de su estado pero no se encuentra unida al gestor de ventanas. Sin embargo, esta actividad ya no es visible para el usuario y pude ser destruida por el sistema cuando la memoria sea necesitada en otra parte.

Cuando una actividad cambia entre los diferentes estados descritos anteriormente, se notifica por medio de diferentes métodos. Estos métodos se pueden reemplazar para realizar diferentes tareas cuando el estado de la actividad cambie.

La siguiente estructura incluye cada uno de los métodos fundamentales del ciclo de vida en una actividad.

```
public class ExampleActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // La actividad está siendo creada.
    }
    @Override
    protected void onStart() {
        super.onStart();
          // La actividad esta a punto de ser visible.
    }
    @Override
    protected void onResume() {
        super.onResume();
        // La actividad es visible.
    }
    @Override
    protected void onPause() {
        super.onPause();
        // Otra actividad es visible (esta a punto de ser pausada).
    }
    @Override
    protected void onStop() {
        super.onStop();
        // La actividad ya no es visible (Se encuentra detenida )
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
```

```
        // La actividad esta punto de ser destruida
    }
}
```

En conjunto, estos métodos definen el ciclo completo de una actividad. Implementando estos métodos, se puede monitorear los tres ciclos anidados en el ciclo de vida de la activad:

- El tiempo de vida completo de una actividad ocurre entre la llamada al método *onCreate()* y la llamada al método *onDestroy*(). La actividad debe realizar una configuración global de su estado (como la definición del *layout*) en el método *onCreate*(), y liberar todos los recursos que quedan en el método *onDestroy*().

- El tiempo visible de una actividad ocurre entre la llamada al método *onStart*() y la llamada al método *onStop*(). Durante este tiempo el usuario puede ver la actividad en la pantalla e interactuar con ella.

- El tiempo de vida en primer plano de una actividad ocurre entre la llamada al método *onResume*() y la llamada al método *onPause*(). Durante este tiempo la actividad se encuentra enfrente a todas las demás actividades en la pantalla y permite la entrada de datos del usuario.

La siguiente tabla muestra información adicional sobre las fases del ciclo de vida y los métodos principales de nivel superior de la clase *Activity*.

Método	**Función**
onCreate()	Se invoca al crear la actividad. Aquí se realiza la configuración.
onRestart()	Se invoca si se reinicia la actividad.
onStart()	Se invoca cuando la actividad es visible en pantalla para el usuario.
onResume()	Se invoca cuando la actividad comienza a interactuar con el usuario.
onPause()	Se invoca al detener la actividad o al reclamar CPU u otros recursos. En este método se guarda el estado para que al reiniciar una actividad pueda comenzar con el mismo estado con el que se finalizo.
onStop()	Se invoca para detener la actividad y pasar a una fase de invisibilidad y posteriores eventos de ciclo de vida.
onDestroy()	Se invoca al eliminar una actividad de la memoria del sistema. Se produce por invocar directamente onFinish()o cuando el sistema decide detener la actividad para liberar recursos.

Tabla 1. Los métodos principales de una Actividad

La figura 2 ilustra los ciclos y las trayectorias que una actividad podría tomar entre los diferentes estados. Los rectángulos representan el llamado de los métodos que se pueden implementar para llevar a cabo operaciones cuando la actividad cambie entre los diferentes estados.

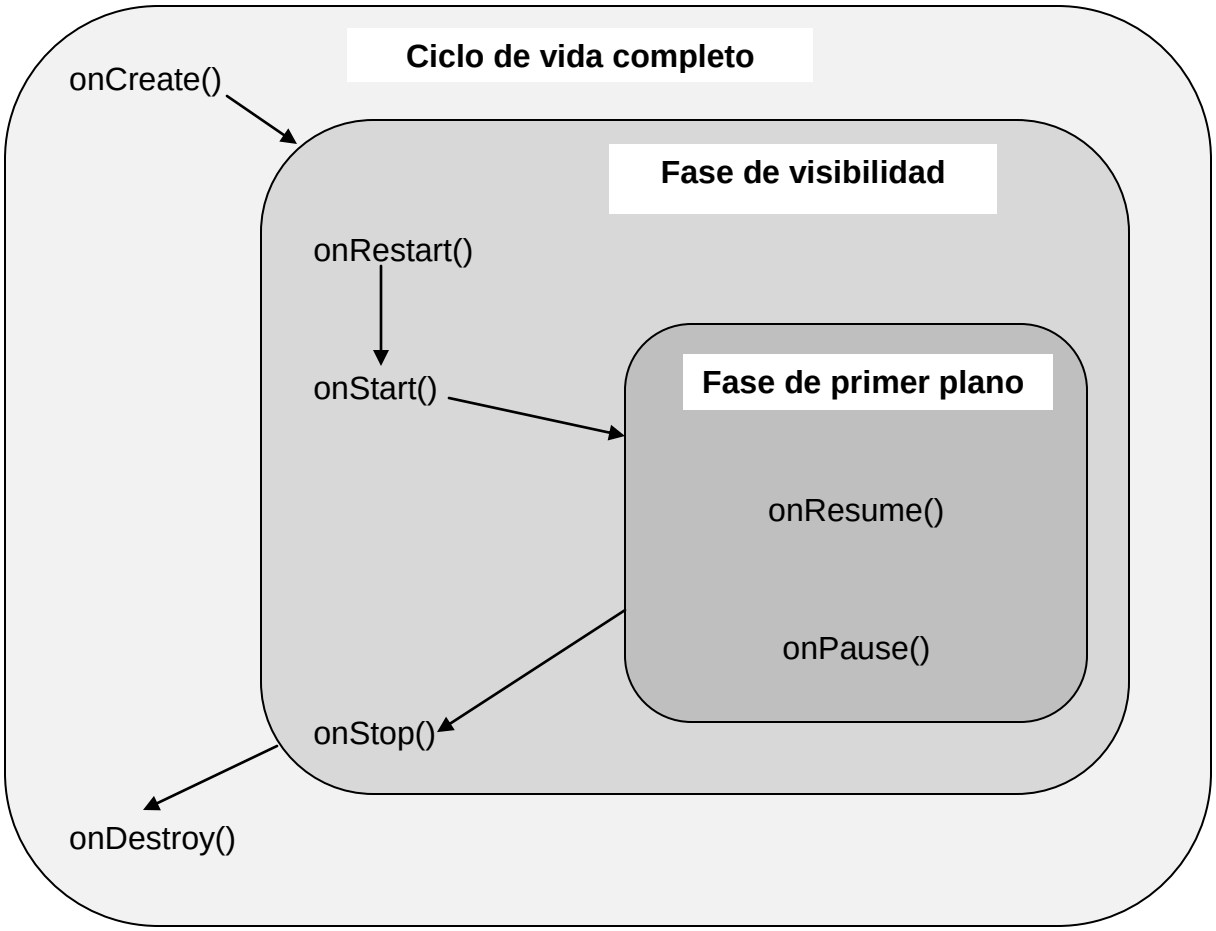


Figura 2. Los ciclos de vida de una Actividad

SIP (*Session Initiation Protocol*).

Es un protocolo de señalización usado en la telefonía
IP, para establecer el inicio, modificación y terminación de las sesiones entre los usuarios SIP. Este protocolo soporta sesiones de voz, vídeo, conferencias, mensajería instantánea y otras similares. Además dispone de los mecanismos necesarios para garantizar la fiabilidad de la comunicación.

Arquitectura.

SIP es un protocolo cliente/servidor basado en el paradigma de petición respuesta. Las peticiones son generadas por un cliente y son enviadas a un servidor. Este se encarga de procesar las peticiones y enviar una respuesta al cliente.

Características.

- SIP es capaz de conocer en todo momento la localización de los usuarios
- Permite negociar los parámetros necesarios para la comunicación: puertos para el tráfico SIP por default es el 5060, direcciones IP para el tráfico Media, el códec de audio a usar.
- Permite determinar si un usuario esta o no disponible para establecer la comunicación
- Permite la modificación, transferencia, finalización de la sesión activa. Informa del estado de la comunicación que se encuentra en progreso

Direccionamiento.

Las direcciones SIP identifican a un usuario de un determinado dominio y se les conoce habitualmente como URI (*Uniform Resource Indentifier*) y se puede especificar de la siguiente manera:

sip:usuario@dominio[port]

usuario: es el nombre de usuario de la cuenta SIP o el número de extensión
dominio: es la IP del dispositivo o un nombre asociado a la IP asignado por un DNS (Servidor de Nombres de Dominio)
port: el puerto para la comunicación SIP

De la misma forma el dispositivo móvil, el teléfono convencional y el softphone tienen su URI correspondiente en la red LAN, esa URI se utilizará para localizarlos y enviarles las peticiones o respuestas durante la comunicación.

Mensajes SIP

El protocolo define dos tipos de mensajes, las peticiones por parte de los clientes y las respuestas del servidor

En la Tabla 2 se muestran las Peticiones SIP más utilizadas:

PETICIÓN	DESCRIPCIÓN
INVITE	Petición que se envía a un usuario cuando se desea establecer una llamada con él.
ACK	Confirma que el cliente ha recibido un INVITE.
BYE	Para indicar al Servidor que finalice la llamada, puede ser usado por el usuario que envió el INVITE o el que la recibió.
REGISTER	Lo emplea el cliente para registrarse en el servidor.

Tabla 2. Las principales peticiones SIP

Las respuestas asociadas a las peticiones están formadas por un identificador numérico, dependiendo la petición enviada y el estado de la comunicación es el mensaje que se envía como respuesta. Las respuestas comunes se ilustran en la tabla 3.

TIPO	Id	SIGNIFICADO
Estado de la Comunicación	100	Trying – Intentando
Estado de la Comunicación	180	Ringing – Sonando
Estado de la Comunicación	183	Session progress – llamada en progreso
Informan del éxito de la comunicación	200	Ok
Informan errores en el Cliente	401	Unauthorized – No autorizado
Informan errores en el Cliente	404	Not Found – No encontrado

Tabla 3. Repuestas SIP que indican el estado de la comunicación

Registro con servidor SIP.

En seguida se describe el intercambio de mensajes para el registro de un dispositivo con el servidor de Registro

- El teléfono con extensión 200, envía una petición SIP *Register* al Servidor Proxy.

- El Proxy al estar configurado para exigir configuración envía una respuesta que el usuario no está autorizado por el momento y requiere mostrar información necesaria.
- El teléfono envía de nuevo la petición *Register* añadiendo a la petición campos donde indica su contraseña.
- Si la información es correcta, el servidor Proxy responde con *200 OK* indicando el éxito de la petición.

En la Figura 3 se muestra el intercambio de mensajes de un teléfono con número de extensión 200 que desea registrarse con el servidor.

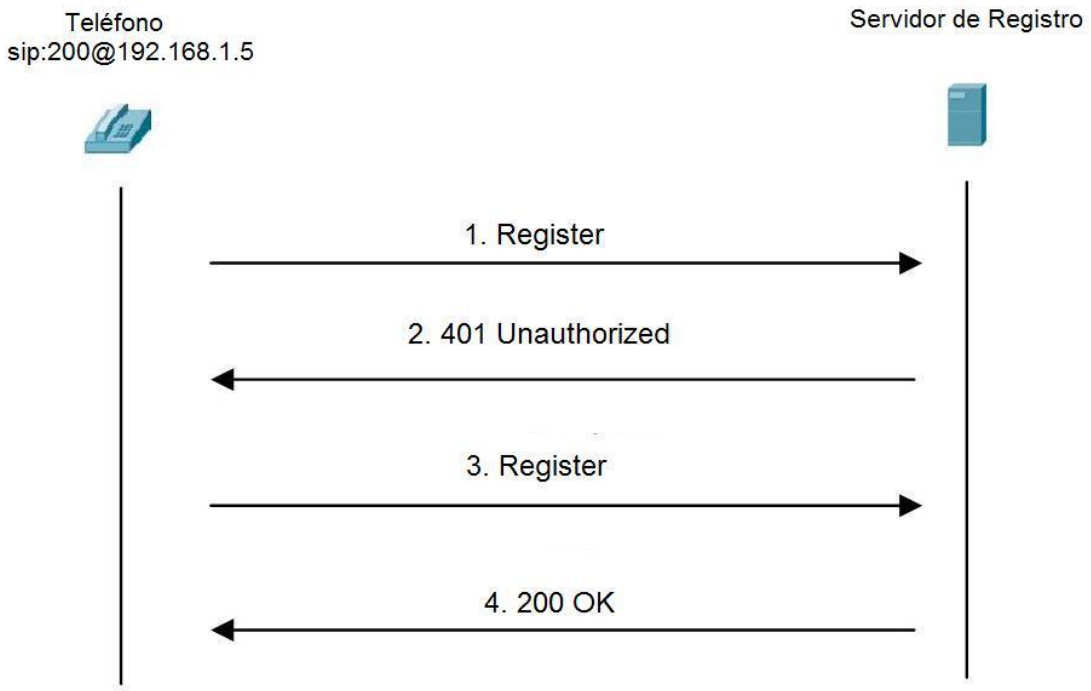


Figura 3. Proceso de registro de un usuario en un servidor de registro

Establecimiento de comunicación entre dos dispositivos.

- El llamante envía un mensaje *Invite* indicando que desea llamar al usuario destino
- El servidor busca la dirección destino en su base de datos, si la dirección destino está registrada, envía el mensaje *Invite* al usuario llamado.
- El servidor de registro envía un mensaje *Trying* (intentando) al usuario llamante

- El usuario llamado envía un mensaje *180 Ringing* al teléfono que llama para indicar que está sonando
- El usuario llamado envía una respuesta *200 OK* al servidor para indicar el fin de la transacción de manera exitosa
- Por último el servidor envía el mensaje recibido *200 OK* al usuario que llama.

- El usuario llamante envía un mensaje *ACK* para confirmar que el canal de comunicación está listo para transmitir voz
- Posteriormente se transmite la voz, siendo transmitida con el protocolo RTP.

En la Figura 4 se muestra como un teléfono con dirección SIP 200@192.168.1.5 envía una petición de llamada al teléfono con extensión 201.

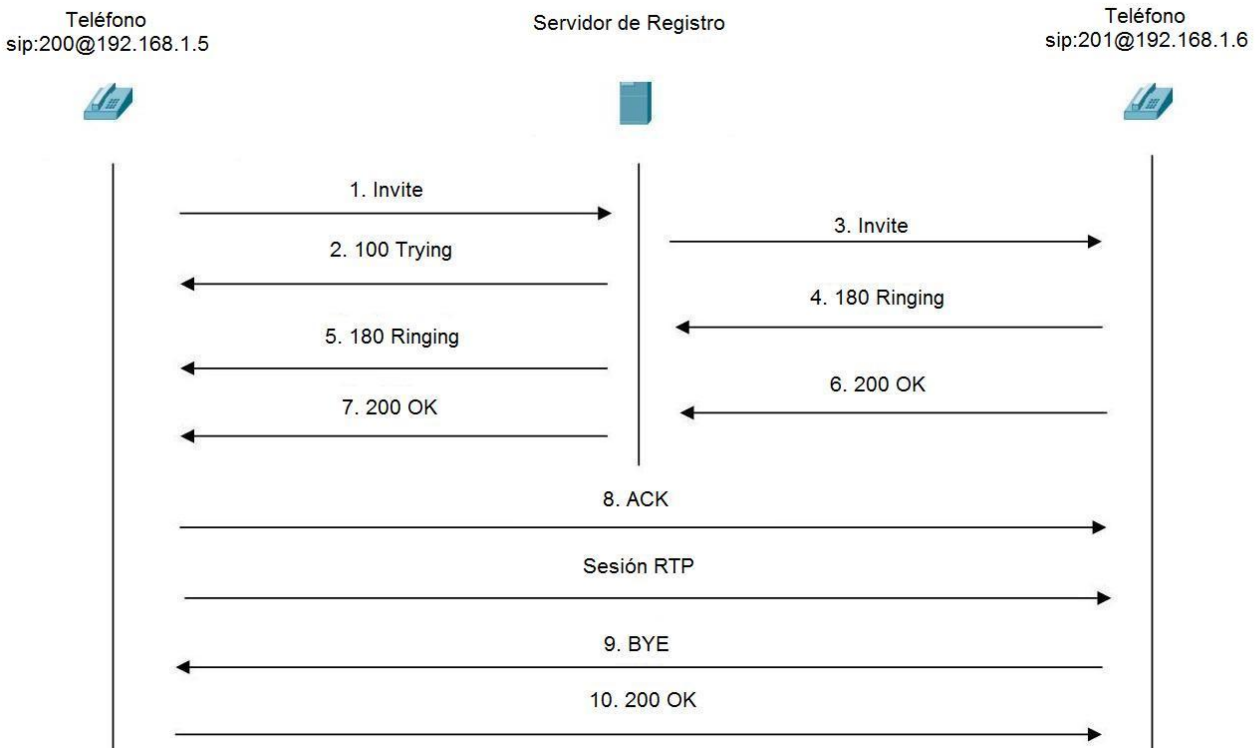


Figura 4. Establecimiento de llamada entre dos teléfonos

Pila de protocolos SIP

SIP se integra perfectamente con los protocolos de transporte como RTP y SDP, permitiendo así formar una completa arquitectura multimedia. SIP establece y gestiona las sesiones de usuarios, RTP se encarga de transportar el audio y SDP describe la configuración de las llamadas.

En la figura 5 se muestra la pila de protocolos de SIP.

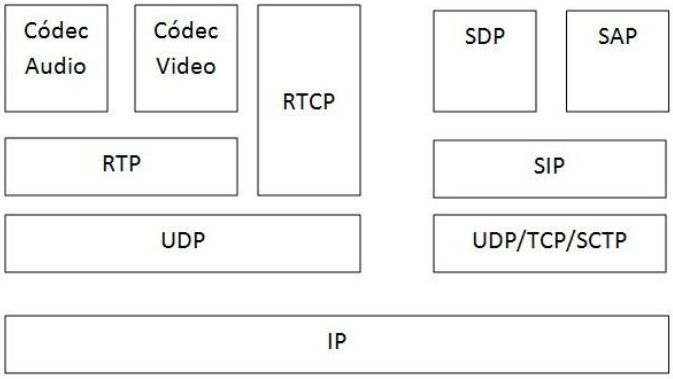


Figura 5. Pila de protocolos SIP

RTP (*Real Time Protocol)*

RTP es el protocolo que se encarga de transportar el flujo de audio y vídeo en tiempo real sobre una red de paquetes. Es decir cuando el protocolo SIP establece la correcta comunicación entre dos participantes, por ejemplo el dispositivo móvil y el teléfono convencional, el audio será transportado con el protocolo RTP.

El proceso de transporte implica dividir en paquetes el flujo de bits que proporciona el codificador de señal, enviar estos paquetes por la red y ensamblar el flujo de bits original en el destino. En este proceso existen pérdidas de paquetes, retraso en su viaje por la red e incluso una alteración en el orden en que llegan al receptor. Por tanto RTP permite al otro extremo detectar todas las pérdidas.

Funciones:

Fragmentación. Cada paquete RTP contiene un número de secuencia para detección de pérdidas durante el ensamblado del mensaje en recepción.

Sincronización media. Las aplicaciones usan buffers, para compensar el retardo que hay en la llegada de paquetes.

Identificación de tipo de Carga: RTP tiene este identificador en cada paquete, describe el tipo de codificación que sea usado para generar dicho paquete. Los codificadores de audio y vídeo se diferencian en su capacidad para trabajar de forma adecuada bajo las distintas condiciones de perdidas.

Identificación de trama. Es un bit de marca que se usa para indicar al receptor el inicio y fin de cada trama, para sincronización con niveles superiores. Las tramas son las unidades lógicas usadas para enviar las señales de audio y vídeo.

UDP (User Datagram Protocol).

Es un protocolo de la capa de transporte no orientado a conexión, proporciona a las aplicaciones una forma de enviar datagramas (pie) IP encapsulados sin tener que establecer una conexión.

Es usado por RTP como protocolo de transporte evitando un retardo prolongado en el envío de paquetes sobre la red.

TCP (Transmission Control Protocol).

Este es uno de los protocolos más importantes en internet, es un protocolo orientado a conexión y asegura que los datagramas IP enviados lleguen correctamente a su destino y los retransmite si es necesario. En la recepción de los datagramas, se encarga de ensamblarlos en mensajes con la secuencia apropiada.

Este protocolo garantiza a SIP la fiabilidad y retransmisión de información para la correcta comunicación entre los participantes.

ASTERISK

Asterisk es un proyecto de comunicación de código abierto, es gratuito y posibilita a una computadora convertirse en un servidor de comunicaciones de voz. Asterisk permite crear servicios y aplicaciones para telefonía. Incluyendo IP PBX, VoIP gateways.

Asterisk es un programa de software libre, bajo la licencia GNU *General Public License* (GPL), y está disponible para ser descargado sin ningún costo.

Asterisk incluye todos los bloques necesarios para crear un sistema PBX o cualquier otro tipo de comunicación.

Algunos de los bloques que incluye son:

- Controladores para diferentes protocolos de VoIP.
- Enrutamiento y gestión de llamadas entrantes.
- Generación de llamadas salientes y enrutamiento.
- Funciones de gestión de medios de comunicación (grabar, reproducir y generar el tono, etc.).
- Conversión de protocolos (convertir de un protocolo a otro).
- Integración de bases de datos para acceder a la información en bases de datos relacionales.
- Integración de servicios web para acceder a los datos mediante protocolos estándares de internet.
- Grabación de llamadas y funciones de monitoreo.

  La combinación de estos componentes permite a un desarrollador crear rápidamente aplicaciones con capacidad de voz.

Asterisk como PBX

Asterisk puede ser configurado como el núcleo de una central telefónica IP, permitiendo intercambio de llamadas, y permitiendo conexión de llamadas con el mundo exterior a través de IP, permite realizar llamadas analógicas con el Servicio Telefónico Tradicional así como realizar llamadas digitales.

Plataformas Soportadas

Asterisk funciona en una amplia variedad de sistemas operativos incluyendo Linux, *Mac OS X, Solaris* de *Sun*, entre otros. Ofrece todas las características de una PBX, incluyendo características avanzadas de gama alta de *PBX* privados. La arquitectura de Asterisk es compatible con Voz sobre IP en diversos protocolos y puede interoperar con casi todos equipos de telefonía.

Hardware Soportado

Asterisk no necesita de hardware adicional para Voz sobre IP. Para interacción con equipo telefónico digital y analógico, Asterisk es compatible con numerosos dispositivos de hardware.

Protocolos Soportados

Asterisk soporta una amplia gama de protocolos para el manejo de voz sobre interfaces de telefonía tradicional; entre ellas H323, SIP, MGCP y SCCP.

DESARROLLO

Investigación previa.

El primer paso fue hacer una revisión de la API SIP en la documentación oficial para desarrolladores del sistema operativo Android, se identificaron las clases e interfaces necesarias para construir una aplicación SIP, además de los permisos para uso de la API de Android y hardware del dispositivo.

IDE de desarrollo ECLIPSE.

Eclipse es el entorno de programación de aplicaciones que se uso para desarrollar la aplicación móvil, incluye un editor de texto, un compilador, un intérprete y un depurador. Este entorno es compatible con varios lenguajes de programación, sin embargo nosotros trabajaremos con el compilador de Java. Podemos descargar *Eclipse* en el sitio web oficial y de forma gratuita, la versión para trabajar es el *Eclipse Classic.*

Android ofrece un *plugin* para Eclipse llamado *Android Development Tools* (ADT), diseñado para dar un completo refuerzo para el desarrollo de aplicaciones Android. También es necesario instalar el *SDK* de Android para poder usar las bibliotecas y herramientas que nos permitirán crear, probar y depurar aplicaciones.

API SIP

El uso de esta API permite agregar a una aplicación Android, características de telefonía IP basadas en SIP. Android cuenta con una pila completa del protocolo SIP y servicios integrados que permiten a las aplicaciones realizar y recibir llamadas, sin tener que administrar las sesiones, el transporte o reproducción de audio directamente.

Requerimientos

- Es necesario un dispositivo con Android 2.3 o superior
- SIP corre sobre una conexión de datos inalámbrica, el dispositivo debe contar con Wi-Fi o servicio de datos móviles.
- Cada participante en la sesión de comunicación debe tener una cuenta SIP, hay diferentes proveedores SIP, en nuestro caso las cuentas serán configuradas en el Servidor Asterisk (ver Apéndice)

Clases e interfaces de la API SIP.

En la Tabla 4 se muestran las clases e Interfaces de la API SIP usadas para la aplicación móvil.

Clase / Interface	Descripción
SipAudioCall	Se encarga de una llamada de audio de internet sobre SIP
SipAudioCall.Listener	Escucha los eventos relacionados a una llamada SIP, llamadas entrantes (“on ringing”) o llamadas salientes (“on calling”)
SipErrorCode	Define códigos de error durante las acciones SIP
SipManager	Gestiona inicio de conexiones SIP y proporciona acceso a los servicios relacionados con SIP
SipProfile	Define un perfil SIP, incluye una cuenta SIP, dominio y servidor
SipProfile.Builder	Clase que ayuda a construir un perfil SIP
SipRegistrationListener	Interfaz que escucha los eventos de registro SIP

Tabla 4. Las clases e interfaces de la API SIP de Android

La aplicación desarrollada permite establecer comunicación con teléfonos convencionales y softphones. Además puede comunicarse con otros dispositivos móviles que tengan instalada la misma aplicación.

Características Mínimas

Las características mínimas que debe cumplir la aplicación son

- Establecer una comunicación entre la aplicación móvil y el teléfono convencional
- Permitir realizar y recibir llamadas de voz sobre SIP entre la aplicación móvil y el teléfono convencional, sin importar situaciones de eco o retardo
- Incluir una GUI para poder interactuar con la aplicación

Módulos de programación.

El desarrollo de la aplicación móvil se descompone en módulos de programación

- Módulo de permisos (para uso de *Wi-Fi, SIP*, micrófono, auricular)
- Módulo de registro con servidor
- Módulo de realizar llamada SIP
- Módulo de recepción de llamada SIP
- Módulo GUI para la aplicación

Estos módulos corresponden a los objetivos particulares del proyecto. En la figura 6 se muestra un diagrama general de los módulos mencionados.

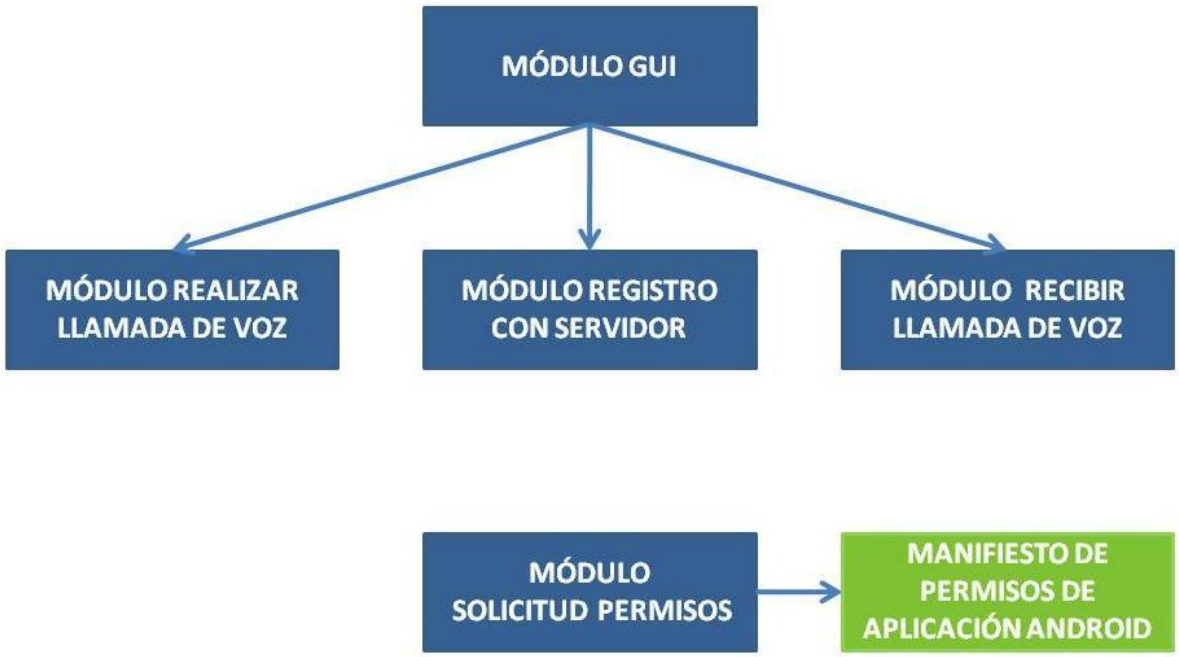


Figura 6. Diagrama que muestra los módulos de la aplicación móvil

Módulo Solicitud de Permisos.

Toda aplicación debe tener un archivo con extensión XML llamado Manifest.xml, este se encuentra el directorio raíz de la aplicación. Este manifiesto presenta información esencial sobre la aplicación para el Sistema Android y se encarga de establecer los permisos necesarios dentro de la aplicación. Nuestro manifiesto hace lo siguiente

Creando el Manifiesto

El encabezado del archivo es generado cuando se crea el proyecto en el IDE Eclipse, indicando la versión de XML, la codificación, y el nombre del paquete del proyecto

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.sip"
    android:versionCode="1"
    android:versionName="1.0" >
```

Reiteramos que el uso de la API SIP necesita un dispositivo Android con versión 2.3 o superior para dar soporte a esta característica. Por lo tanto garantizamos que la aplicación solo sea instalada en dispositivos que soporten SIP añadiendo la siguiente línea en el manifiesto:

```
    <uses-sdk android:minSdkVersion="10" />
```

Para solicitar los permisos necesarios como el uso de la API SIP, acceso a internet, uso de Wi-Fi, permiso para registro y modificación del audio, uso del modo vibrador del dispositivo y uso del micrófono, se debe agregar al manifiesto las siguientes líneas:

```
…
    <uses-permission android:name="android.permission.USE_SIP"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
    <uses-permission android:name="android.permission.WAKE_LOCK"/>
    <uses-permission android:name="android.permission.RECORD_AUDIO"/>
    <uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS"/>
    <uses-permission android:name="android.permission.VIBRATE"/>

        <uses-feature android:name="android.hardware.sip.voip" android:required="true"/>
        <uses-feature android:name="android.hardware.wifi" android:required="true"/>
        <uses-feature android:name="android.hardware.microphone" android:required="true"/>
  …
```

En el manifiesto se describen los componentes de la aplicación, como los nombres de las actividades, servicios, receptores de broadcast para llamadas entrantes y filtros de intento que usaremos para ejecutar otras actividades.

La actividad principal se llama *SipActivity* y será la primer vista que se mostrará al ejecutar la aplicación en el dispositivo, además se describe otra actividad *SipProfileSettings*, esta es la vista para un menú donde se configura la cuenta de usuario SIP.

```
    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name"
```

```
        android:screenOrientation="portrait">

        <activity
            android:name=".SipActivity"
            android:label="@string/app_name"
            android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity
        android:name="com.utilities.SipProfileSettings"
        android:label="@string/app_settings"
        android:screenOrientation="portrait">
        </activity>
```

Si en la aplicación se recibirán llamadas, es necesario definir un receptor.

```
        <receiver
            android:name=".IncomingCallReceiver"
            android:label="Incoming Call Receiver.">
        </receiver>

    </application>
</manifest>
```

Módulo Registro con Servidor.

Para poder hacer la gestión de las sesiones *SIP* entre los participantes, se creó una clase que se encarga de todas las acciones para este propósito. Lo primero que se debe hacer, es crear un objeto de la clase *SipManager*, este objeto nos permitirá:

- Inicio de las sesiones *SIP*
- Iniciar y recibir las llamadas de voz
- Registro y baja con el proveedor de servicios *SIP*, en este caso nuestro Servidor Asterisk

El siguiente fragmento de código muestra como se realiza la instanciación en un constructor.

```
public static final void setManager(SipActivity sipActivity) {
        .
        .
        sipManager = SipManager.newInstance(sipActivity);
        .
        .
```

Perfil de Usuario

Una aplicación SIP involucra uno o más usuarios, cada uno tiene una cuenta SIP, en este tipo de aplicaciones cada cuenta SIP se representa con un objeto *SipProfile.*

Un *SipProfile* define un perfil de usuario *SIP*, está formado por una cuenta SIP, su IP o dominio y la información del servidor. El perfil que está asociado al dispositivo donde se ejecuta la aplicación se le llama Perfil Local, al usuario que se desea llamar se le conoce como Perfil Par. Cuando la aplicación móvil se registre con el servidor *Asterisk*, quedara almacenada su localización para enviarle llamadas *SIP*.

Aquí se muestra como crear el *SipProfile* y dar seguimientos a los eventos de registro con el servidor:

```
builder = new SipProfile.Builder(username, domain);
                    builder.setPassword(password);
                    builder.setDisplayName(displayName);
                    builder.setPort(port);
                    builder.setProtocol(protocol);
                    sipLocalProfile = builder.build();
```

La siguiente sección abre el perfil local para realizar o recibir llamadas genéricas SIP. También se establece la acción "*com.sip.INCOMING_CALL"*, que se usa con un filtro de intentos cuando se recibe una llamada. En la siguiente sección se muestra el código para realizar el registro con el servidor y la inicialización del *SipProfile* para realizar o recibir llamadas.

```
intent = new Intent();
intent.setAction("com.sip.INCOMING_CALL");
incomingCallPendingIntent =
PendingIntent.getBroadcast(sipActivity.getBaseContext(),0,intent,Intent.FILL_IN_ACTIO
N);

sipManager.open(SipLocalProfile.getSipProfile(),    incomingCallPendingIntent,    new
RegistrationListener(sipActivity));
```

También es necesario establecer un escuchador de eventos de registro, que nos indique el estatus en todo momento, para los casos de registro exitoso, falló el registro o registrando con el servidor. En el siguiente código se muestra la implementación de la interfaz *SipRegistrationListener*

```
public class RegistrationListener implements SipRegistrationListener{
      public RegistrationListener(SipActivity sipActivity){
```

```
            …
      }

      public void onRegistering(String localProfileUri) {
            //se pueden mostrar mensajes como (…Registrando con servidor)
      }

      public void onRegistrationDone(String localProfileUri, long expiryTime) {
            //se pueden mostrar mensajes como (…Registrado)
      }

      public void onRegistrationFailed(String localProfileUri, int errorCode,
                  String errorMessage) {
            //se pueden mostrar mensajes como (… fallo el registro)
      }

}
```

Módulo Realizar Llamada.

Para realizar una llamada es necesario contar con:

- Un objeto *SipProfile* que será el que realizará la llamada, en este caso será el perfil local.
- Una dirección *SIP* valida del Servidor *SIP*, en este caso será el perfil par.

Para realizar una llamada de audio, antes se debe establecer un *SipAudioCall.Listener*. Es el encargado de escuchar los eventos cuando se realiza una llamada de audio, los estados posibles como, llamando, ocupado, llamada en espera, finalizando llamada, entre otros. Los métodos que se implementan de esta clase se sobreescriben añadiendo código que necesitemos dependiendo la situación,

```
public class AudioCallListener extends SipAudioCall.Listener{
      public AudioCallListener(SipActivity sipActivity){
      }

      @Override
      public void onCallBusy(SipAudioCall call){
            // usuario ocupado, agregar código aqui
      }

      @Override
      public void onCallEnded(SipAudioCall call){
            // finaliza llamada, agregar código aqui
      }
```

```
@Override
public void onCallEstablished(SipAudioCall call){
	// llamada establecida, agregar código aqui
}


@Override
public void onCallHeld(SipAudioCall call){
	// llamada en espera, agregar código aqui
}

@Override
public void onCalling(SipAudioCall call){
	// llamando, agregar código aqui
}

@Override
public void onError(SipAudioCall call, int errorCode, String errorMessage){
	// ocurrió error, agregar código aqui
}


@Override
public void onReadyToCall(SipAudioCall call){
	// listo para llamar, agregar código aqui
}

@Override
public void onRinging(SipAudioCall call, SipProfile caller){
	// llamando, agregar código aqui
}
```

Después de configurar el *SipAudioCall.Listener* se puede realizar la llamada. El método *makeAudioCall()* de la clase *SipManager* toma los siguientes parámetros:

- Un perfil local SIP.
- Un perfil par SIP.
- Un *SipAudioCall.Listener* el cual escuchará los eventos en la llamada.
- Tiempo de espera en segundos.

El código para realizar la llamada de audio es el siguiente:

```
sipManager.makeAudioCall(SipLocalProfile.getUriString(),

SipPeerExtension.getUri(), new AudioCallListener(sipActivity), 60);
```

Módulo Recibir Llamada.

Para recibir llamadas, la aplicación necesita incluir una subclase de *BroadcastReceiver*, que tiene la habilidad de identificar y responder a una llamada de entrada. Para esto no hay que olvidar agregar el receptor en el Manifiesto de la aplicación como se indicó en la sección Módulo solicitud Permisos, además de implementar ese receptor con una subclase de *BroadcastReceiver*, que en nuestra aplicación se llama IncomingCallReceiver.java y por ultimo establecer un filtro de intentos que filtra la acción que representa una llamada de entrada, la acción se lla*ma com.sip.INCOMING_CAll*

Subclase de BroadcastReceiver.

Aquí se muestra el código para implementar el receptor de llamadas entrantes, en el cual se activa el *ringtone* para notificar la llamada entrante, además de permitir a la clase Manager tomar la llamada para que posteriormente el usuario a través de la interfaz decida si contesta o rechaza dicha llamada.

```
public class IncomingCallReceiver extends BroadcastReceiver {
	private static final String TAG = "IncomingCallReceiver-->";

	@Override
	public void onReceive(Context context, Intent intent) {
		Log.d(TAG, "Hay una llamada entrante");

		…

		AudioManager.setMode(android.media.AudioManager.MODE_RINGTONE);
		sipActivity.startMediaPlayer();
		sipActivity.vibratorOn();
		Manager.takeAudioCall(intent);

	}
}
```

Establecer un filtro para recibir llamadas.

Cuando el servicio SIP recibe una nueva llamada, este envía un intento con la acción proporcionada por la aplicación. En esta aplicación es una cadena que se nombró "*com.sip.INCOMING_CALL*", este se menciona cuando se construye el *SipPrfile*.
El objeto *PendingIntent* realiza un *broadcast* cuando el *SipProfile* recibe una llamada

```
intent = new Intent();
```

```
intent.setAction("com.sip.INCOMING_CALL");
incomingCallPendingIntent =
PendingIntent.getBroadcast(sipActivity.getBaseContext(),0,intent,Intent.FILL_IN_ACTIO
N);
sipManager.open(SipLocalProfile.getSipProfile(),    incomingCallPendingIntent,    new
RegistrationListener(sipActivity));
```

Módulo GUI.

Todos los elementos de interfaz de usuario en una aplicación Android se construyen usando vistas (*View*) y grupos de vistas (*ViewGroup*). Un View es un objeto que dibuja algo en la pantalla del dispositivo para que el usuario interactúe con él.

Para comenzar se necesita de un diseño (*Layout*) que es la arquitectura de la interfaz de usuario en una *Activity*. Sirve para definir la estructura y manejar todos los elementos que aparezcan para el usuario. Hay dos formas de declarar el *Layout*:

- Declarar los elementos de la interfaz en XML. Android proporciona el vocabulario *XML* que corresponde a sus clases View, como para *widgets* y *layouts*. Esto se comporta como un árbol de vistas y la comprensión del documento es más sencilla.
- Instanciar elementos de un *Layout* en tiempo de ejecución. En la aplicación se pueden crear objetos View o *ViewGroup* para manipular las propiedades de la interfaz, es decir, se realiza con programación.

En esta aplicación se utilizan las dos formas, para la ubicación y estructura de los botones, cajas de texto visuales, cajas de texto editables y asociarle una imagen por default se usa la declaración *XML* y para iniciar los botones, establecer un escuchador de eventos que permita detectar cuando se pulse un botón y asignarle una acción se usó código de programación.

Archvivo XML.

El archivo donde se hace la declaración de botone se llama main.xml y se encuentra en el directorio *R.layout*.

Cada etiqueta que declara una vista en *XML*, tiene sus atributos específicos como descripción, ubicación, largo, ancho, acción, entre otros que sirven para añadir características al objeto.

Para comenzar se declara un *LayoutLinear* que describe un lienzo principal donde se colocaran todos los objetos de la interfaz.
Se coloca un *TextView* que permitirá visualizar texto con el estatus de la comunicación, por ejemplo: esperando a que conteste llamada, hablando con usuario…, la llamada se encuentra en espera. En el siguiente párrafo se muestra la declaración

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    …

    <TextView
        android:id="@+id/textViewStatus"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello" />
      …
```

Para modificar los mensajes de este objeto TextView se implemento la clase TextViewRunnable para establecer el texto que se quiere mostrar y actualizar el TextView.

```
…
textView = (TextView) this.sipActivity.findViewById(R.id.textViewStatus);
this.textViewRunnable.setText("Se termino la llamada.");
```

Para visualizar el número de extensión que se está marcando, se usa un <EditText> que servirá para mostrar el número de extensión a marcar.

```
   <LinearLayout
            …
        <LinearLayout
                …
            <TableRow
                …

                <EditText
                      android:id="@+id/editTextExtension"
                      android:layout_height="wrap_content"
                      android:layout_width="270dp"
                      android:textSize="40dp"
                      android:textColor="#FFFFFF"
                      android:background="@null"
                      android:ems="6"
                      android:inputType="phone" >
                 </EditText>
                 …
```

Botones.

Para colocar los botones primero se definió el diseño y ubicación de los mismos, para esto se declara una tabla con <*TableLayout*> a su vez anidaremos los <*TableRow*> necesarios que servirán como contenedores para colocar botones, en este caso se colocaron 3 botones por renglón donde se anidaran los botones del teclado numérico, botón llamar, colgar y registrar con servidor.

Para cada declarar cada botón se usa <*ImageButton*> ya que este nos servirá para asociar una imagen diseñada por nosotros a cada botón, este tiene la propiedad de que puede cambiar la imagen que le asociemos en tiempo de ejecución, esto es de utilidad para mostrar diferentes estados de botón, dependiendo que acción realicemos durante la comunicación. Por ejemplo el efecto de pulsación de botón, dar el efecto de que se habilita el botón para realizar llamada, etc.
Aquí se muestra la sección de *XML* para declarar los botones, en el atributo *background* se le asocia las imágenes diseñadas, y se colocan en el directorio *R.res.drawable*

```
<TableLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:stretchColumns="*"
            >

            <TableRow
                   …                                          >

                <ImageButton
                    android:id="@+id/imageButton1"
                    android:contentDescription="@string/desc"
                    android:layout_width="wrap_content"
                   android:layout_height="wrap_content"
                    android:background="@drawable/button1" />
                    …

            </TableRow>
<TableLayout
```

Para dar el efecto de pulsación de botón, se diseñaron dos imágenes para cada botón, el botón por default tiene un diseño con efecto de brillo de luz y el pulsado un color liso.
Para establecer el estado de pulsado y default a un botón y asociarle una imagen a cada estado se crea un archivo xml para cada botón, asignando un nombre alusivo como por ejemplo boton1_custom.xml

Aquí se muestra la sección XML y el nombre del archivo es el que se asigna en la etiqueta *android:background="@drawable/boton1_custom"* como fondo de botón.

```
        <item android:state_pressed="true"
              android:drawable="@drawable/p0"></item>
        <item android:drawable="@drawable/sp0"></item>
```

Para el botón llamar, y registrar se usan de la misma forma <*ImageButton*> para asignar una imagen de acuerdo al estado de la comunicación en la aplicación móvil.
Sin embargo para el botón registrar se usa un <*ToggleButton*> que permite mostrar notificación en el botón cuando se encuentra en “checked” o “unchecked”, Asi podemos darnos cuenta si la aplicación esta o no registrada en el servidor.

```
<TableRow
      android:id="@+id/tableRow5"
                                        >
                <ImageButton
                    android:id="@+id/imageButtonCall"
                    …
"
                    android:background="@drawable/talk" />

                <ToggleButton
                    android:id="@+id/toggleButtonHome"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:textOff="Desconectado"
                    android:textOn="Registrado"
                    android:textColor="#FFFFFF"
                    android:background="@drawable/led"    />

                <ImageButton
                    android:id="@+id/imageButtonHang"
                    …
                   android:background="@drawable/hang"    />
 </TableRow>
```

OnClickListener.

Para establecer un escuchador de eventos para detector cuando se pulsa cada botón se implemento la clase *OnClickListener*, e indicara que símbolo enviar cuando se pulse dado botón. En el siguiente fragmento de código se muestra como se realiza:

```
public class ClickListener implements OnClickListener {


      public void onClick(View view) {
```

```
            switch (view.getId()){
            case R.id.imageButton0:
                  SipPeerExtension.setExtension("0"); //Asocia un caracter 0
                  break;
            case R.id.imageButton1:
                  SipPeerExtension.setExtension("1"); //Asocia un caracter 1
                  break;
            case R.id.imageButton2:
                  SipPeerExtension.setExtension("2"); //Asocia un caracter 2
                  break;

           …

            case R.id.imageButtonAsterisk:
                  SipPeerExtension.setExtension("*");  //Asocia un caracter *
                  break;
            case R.id.imageButtonSharp:
                  SipPeerExtension.setExtension("#");  //Asocia un caracter #
                  break;
            case R.id.imageButtonCall:
                  // insertar codigo para hacer llamada
                  break;
            case R.id.toggleButtonHome:
                  // insertar codigo para registrar o dar debaja de servidor
            break;
            case R.id.imageButtonClear:
                  // Insertar codigo para terminar llamada
                  break;
            }

      }

}
```

Para iniciar los botones se construyó una clase *Buttons.java*, en el siguiente fragmento se muestra un ejemplo

```
public class Buttons {

      public static void setButtons(Context context ){
            final SipActivity sipActivity = (SipActivity)context;
            ClickListener clickListener = new ClickListener ();

      ImageButton imageButton0 =
      (ImageButton)sipActivity.findViewById(R.id.imageButton0);
       imageButton0.setOnClickListener(clickListener);

      …
}
```

Diseño de los botones.

El diseño se realizo en el software *Libre Office*.
Todas las imágenes son formato *PNG* y se encuentran en el directorio *R.drawable*

En la tabla 5 se muestra el diseño de los botones por defecto (sin pulsar) en la interfaz de usuario.

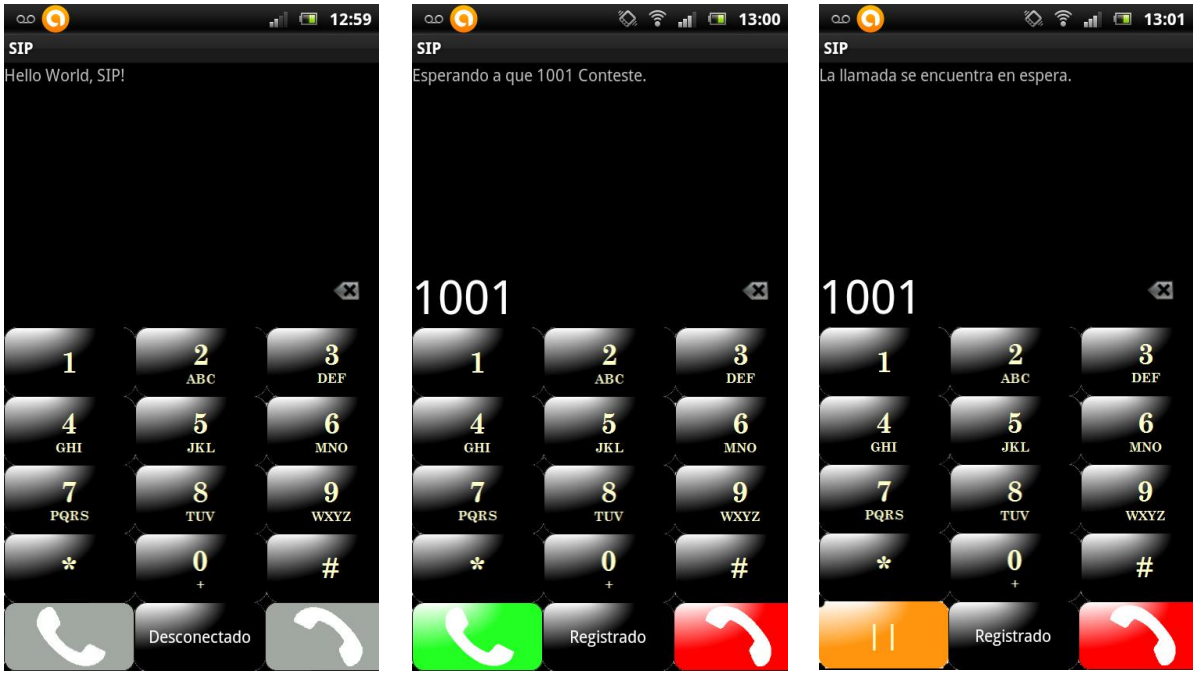
Tabla 5. Diseño de botones por defecto de la interfaz de usuario

Ahora en la tabla 6, se muestra el diseño de los botones para dar el efecto de “botón pulsado”.

Tabla 6. Diseño de los botones por defecto de la interfaz de usuario

Vista general de la GUI.

La vista general de la GUI terminada se muestra en la figura 7 a continuación



Desconectado Realizando llamada Llamada en espera

Figura 7. Captura de pantalla de la interfaz de usuario en el dispositivo, mostrando tres estados durante la llamada, desconectado, realizando llamada y en espera

Implementación de Red LAN para pruebas de comunicación.

Para realizar las pruebas de comunicación entre la aplicación móvil y el teléfono convencional se implementó una red LAN que permite la interconexión entre los participantes para poder realizar las llamadas sobre SIP.

Componentes

La red está formada por un

- Servidor con Asterisk.
- Teléfono convencional (analógico),
- Softphone
- Convertidor Analógico Digital ATA

- Access Point
- Dispositivo móvil con la aplicación SIP.
- Switch genérico (Opcional)

Implementación

A continuación se describe la implementación de la red LAN

El servidor *Asterisk* cumplirá con la función de una central telefónica y se encarga de conmutar las llamadas. En él se configura la información relacionada a los perfiles de usuario *SIP* del Softphone, teléfono convencional y el dispositivo móvil, para hacer posible su registro. *Asterisk* recibe las peticiones de los usuarios SIP y se encarga de resolverlas y enviar respuestas para la gestión de la comunicación.
Para la configuración de *Asterisk* (*ver apéndice…)*

El *softphone Zoiper*, se configuró en una computadora portátil para realizar pruebas de llamadas salientes y entrantes con el teléfono convencional y con dispositivo móvil que tiene instalada la aplicación. Para más detalles acerca de la configuración de Zoiper (*ver apéndice …)*

El teléfono convencional por ser analógico, no podrá integrarse a la red LAN directamente, para ello será necesario el ATA, que se encargará de convertir las señales analógicas a señales digitales (y viceversa), haciendo posible su conexión a la red.
El ATA se debe configurar previamente en un explorador web, para asignar los datos de la cuenta del perfil SIP para el teléfono convencional. Para más detalles de la configuración del ATA *(ver apéndice…)*

El dispositivo móvil con la aplicación instalada, se integra a la red por medio de Wi-Fi al Access Point, a excepción de la aplicación, el *softphone* en una computadora, el teléfono convencional y el servidor se conectan vía cable *Ethernet* a los puertos del Access Point, en caso de no tener puertos suficientes, entonces es viable el uso del *Switch* genérico para interconectarlos.

Esta red es fundamental para verificar que la comunicación entre la aplicación móvil, el teléfono convencional y el *softphone* se establezca de forma correcta.

Maqueta de la Red LAN

La maqueta de la red LAN con todos sus componentes se muestra en la figura 8

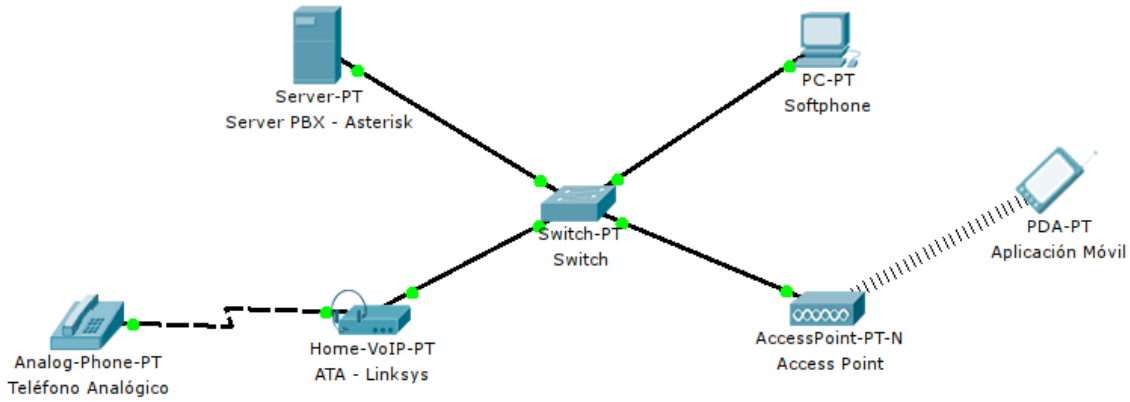


Figura 8. Maqueta de red LAN para pruebas de comunicación

PRUEBAS Y RESULTADOS.

Para realizar las pruebas de comunicación entre la aplicación y el teléfono convencional, se implementó la red LAN con los siguientes componentes:

- Servidor *Asterisk*
  - Computadora portátil
  - 500 Gb disco duro
  - 4 Gb Memoria RAM
  - Sistema operativo *Debian* 6 de Linux
  - Procesador Core I7 de Intel

- *ATA PAP2 Linksys de Cisco*
  - 2 puertos RJ11 (para teléfono convencional)
  - 1 puerto Ethernet

- *Linksys Wireless*-G *Broadband* Router
  - 4 puertos Ethernet
  - *Wi-Fi* de hasta 54 Mbps

- Teléfono convencional

- Softphone Zoiper
  - Computadora Portátil
  - 120 Gb disco duro
  - 2 Gb Memoria RAM
  - Procesador Centrino Duo

- Dispositivos *Android*
  - *Galaxy Note*
    - Android 2.3.5
  - *Xperia* R800 Sony
    - Android 2.3.4

Herramientas adicionales utilizadas
Wireshark: Para la captura de tráfico en la red, instalado en el servidor de la red LAN.

En la figura 9 se muestra la implementación de la red LAN con los componentes arriba descritos

Figura 9. Implementación de Red LAN con el servidor, *router*, ATA, teléfono convencional, dispositivos Android y *softphone*

Cuentas SIP de los dispositivos.

Xperia R800

Nombre de usuario: xperia
Número de extensión: 1004

Galaxy Note

Nombre de usuario: galaxy
Número de extensión: 1001

Teléfono convencional

Nombre de usuario: ata
Número de extensión: 3001

Softphone

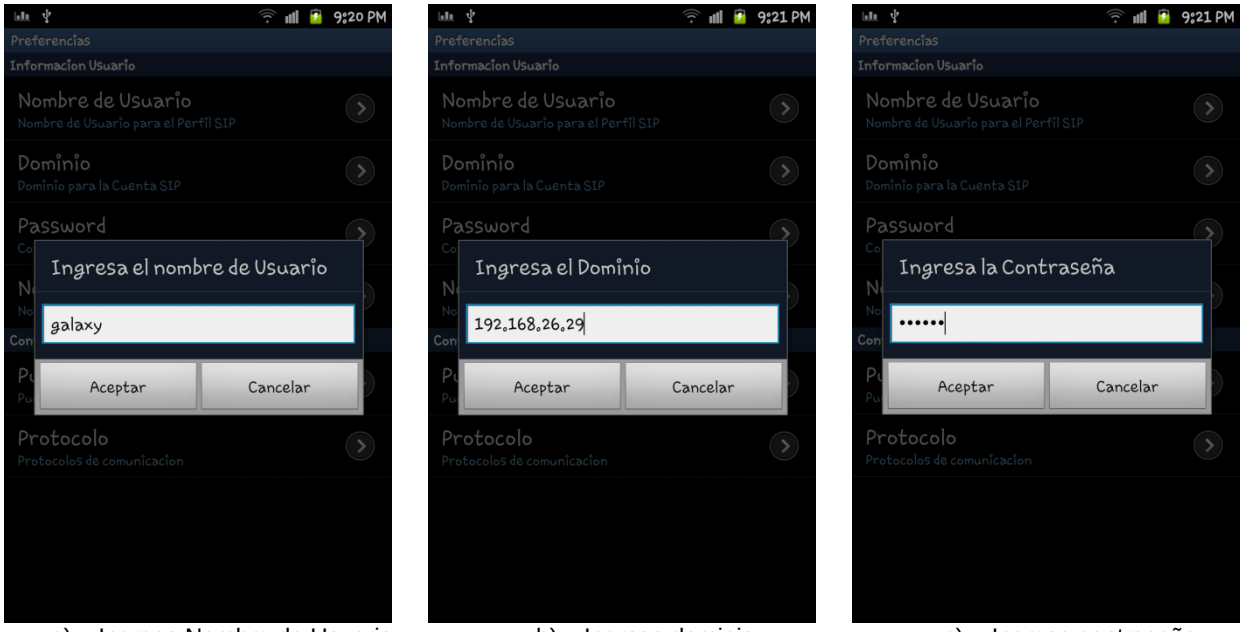
Nombre de usuario: zoiper

Número de extensión: 2001

Pruebas Realizadas

Registro de Aplicación móvil con el servidor Asterisk

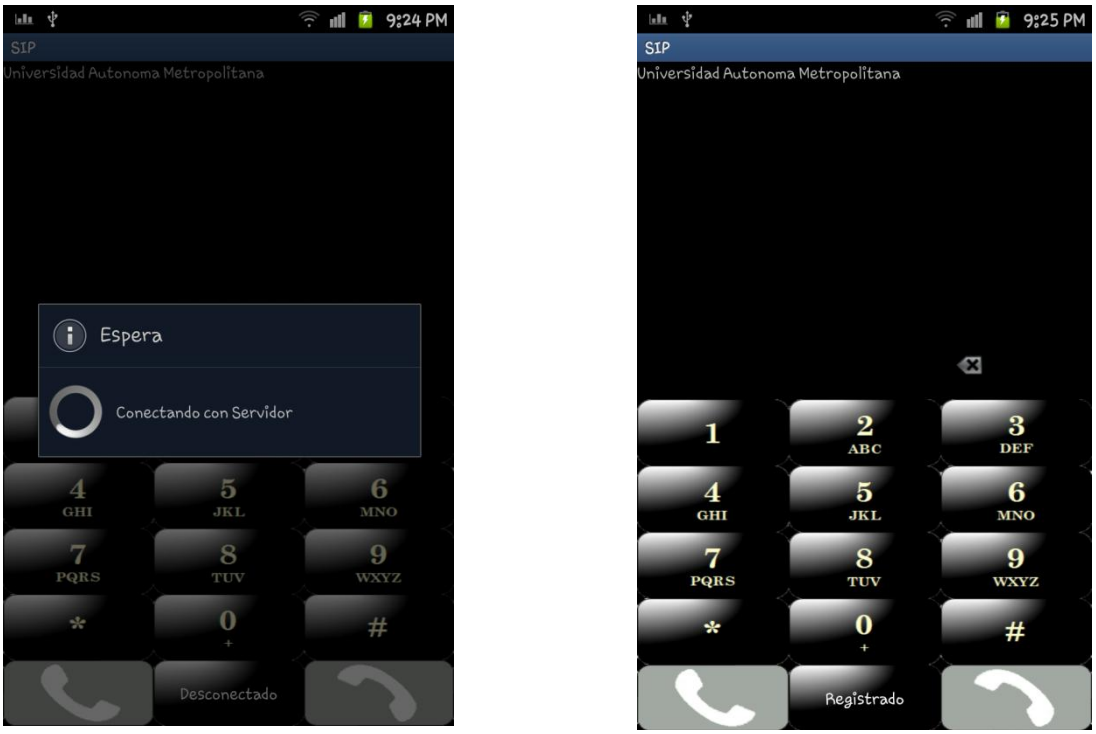
La figura muestra como se ingresa la información de la cuenta SIP en la aplicación, en la figura 10(a) se ingresa el nombre de usuario, 10(b) se ingresa el dominio y en la 10(c) se ingresa la contraseña



a) Ingresa Nombre de Usuario b) Ingresa dominio c) Ingresa contraseña

Figura 10. Ingreso de datos de la cuenta SIP para registro con el servidor

Ahora se muestra en la figura la captura de pantalla del proceso de registro con servidor, en la figura 11(a) se muestra la espera del proceso de registro, en la figura 11(b) se muestra el botón de registrado mostrando el estado exitoso de registro



(a) **(b)**

Figura 11. Interfaz de usuario mostrando el proceso de registro con el servidor

En la figura 12 se muestra la secuencia del intercambio de mensajes para el proceso de registro, como lo define el protocolo SIP.

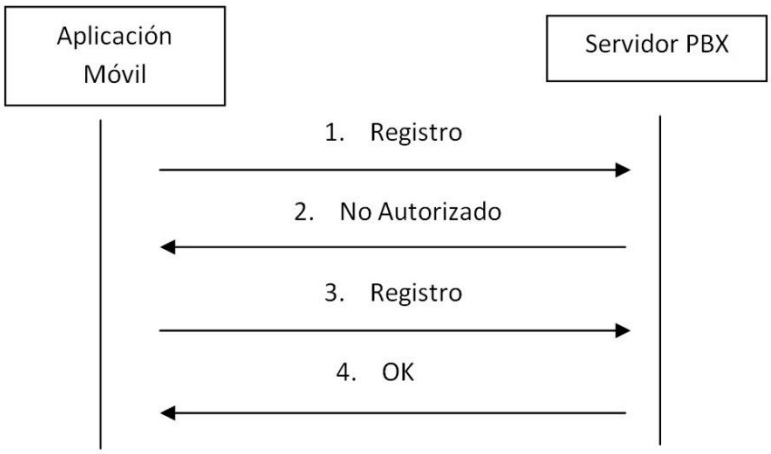
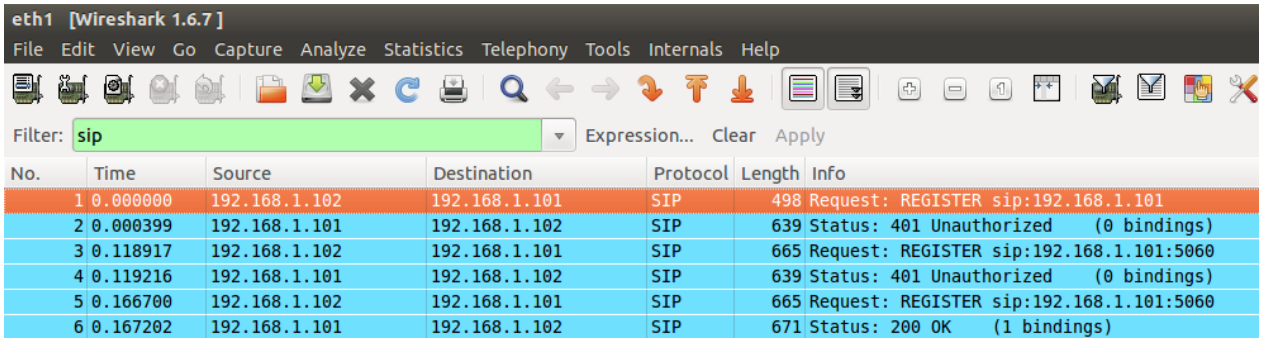


Figura 12. Proceso de registro de la aplicación móvil con el servidor

En la figura 13 se muestra una captura de trafico con wireshark para verificar que el intercambio de mensajes entre la aplicación y el servidor se llevo correctamente.

El servidor tiene la IP 192.168.1.101 y la aplicación móvil en el dispositivo tiene la IP 192.168.1.102.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.102	192.168.1.101	SIP	498	Request: REGISTER sip:192.168.1.101
2	0.000399	192.168.1.101	192.168.1.102	SIP	639	Status: 401 Unauthorized (0 bindings)
3	0.118917	192.168.1.102	192.168.1.101	SIP	665	Request: REGISTER sip:192.168.1.101:5060
4	0.119216	192.168.1.101	192.168.1.102	SIP	639	Status: 401 Unauthorized (0 bindings)
5	0.166700	192.168.1.102	192.168.1.101	SIP	665	Request: REGISTER sip:192.168.1.101:5060
6	0.167202	192.168.1.101	192.168.1.102	SIP	671	Status: 200 OK (1 bindings)

Figura 13. Wireshark. Captura de tráfico de red, en el proceso de registro de la aplicación móvil con el servidor

En la figura 13, haciendo la lectura de izquierda a derecha, la línea No.1, verifica que en el tiempo 0, el dispositivo móvil con IP 192.168.1.102 envía una petición de *Register.*
En la línea número 2, el servidor envía una respuesta no autorizado (401 *Unauthorized*), en la línea 3 el dispositivo móvil envía de nuevo una petición Register y de nueva cuenta el servidor notifica que no está autorizado, finalmente en la línea 5 se envía por tercera ocasión la petición *Register* a lo que el servidor autoriza y envía una respuesta *200 OK.* A pesar de que se envió la petición de registro una vez más comparado con la figura 12, sigue respetando la secuencia y negociación del registro.

Aplicación móvil realiza llamada a Teléfono Convencional.

En la figura 14 se muestra el intercambio de mensajes en el establecimiento de una llamada desde la aplicación móvil al teléfono convencional, tal como lo define SIP.

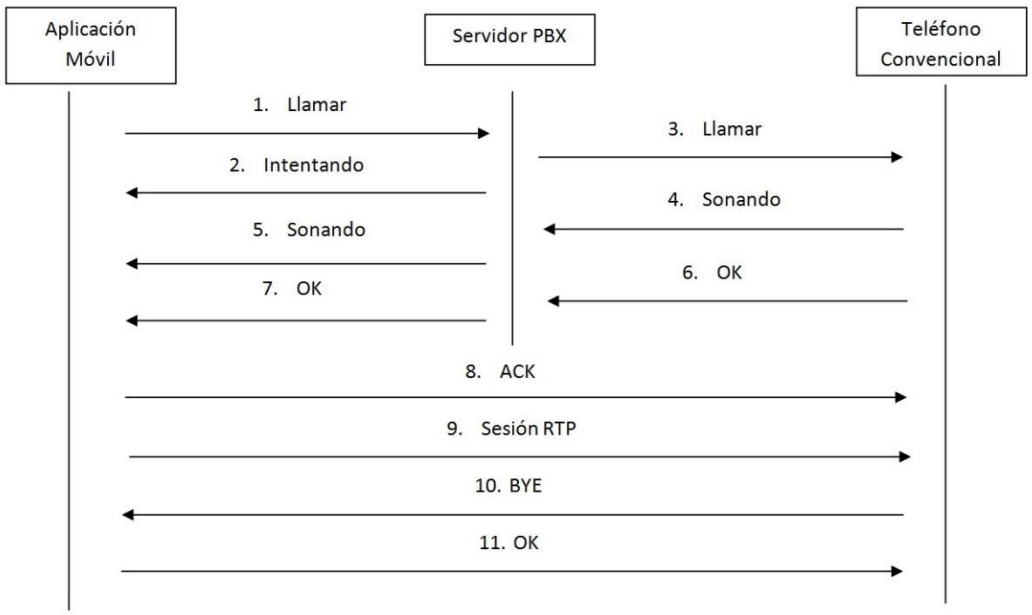
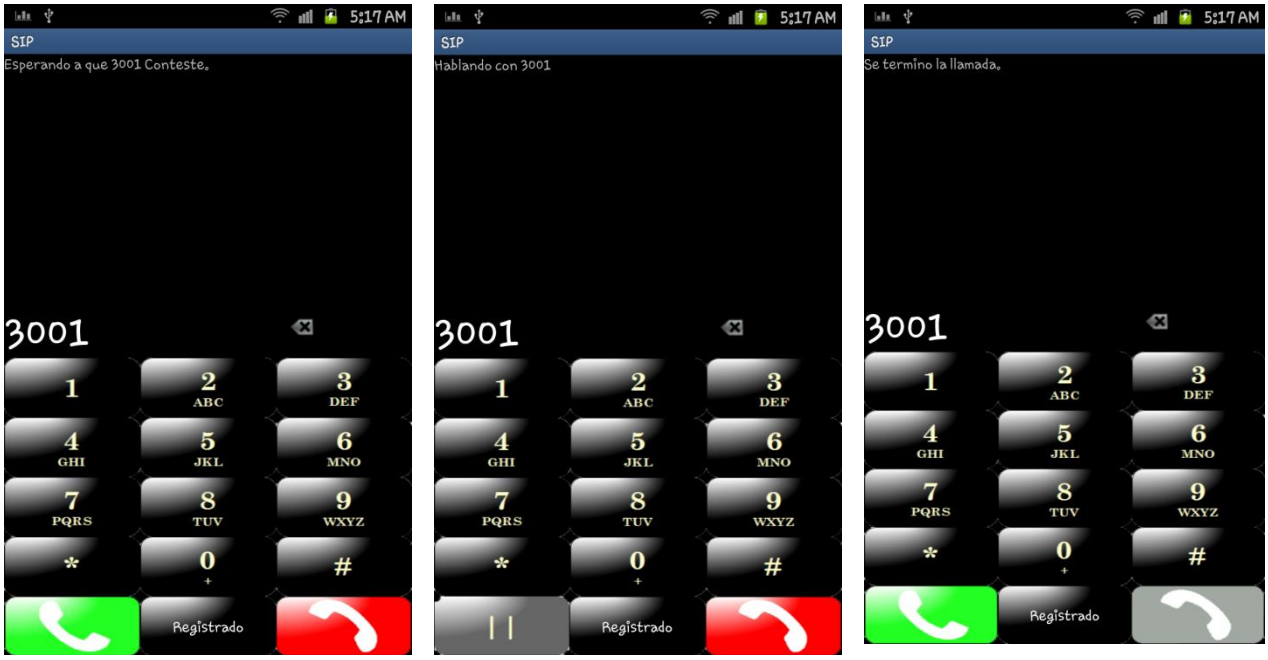


Figura 14. Establecimiento de llamada desde la aplicación móvil al teléfono convencional

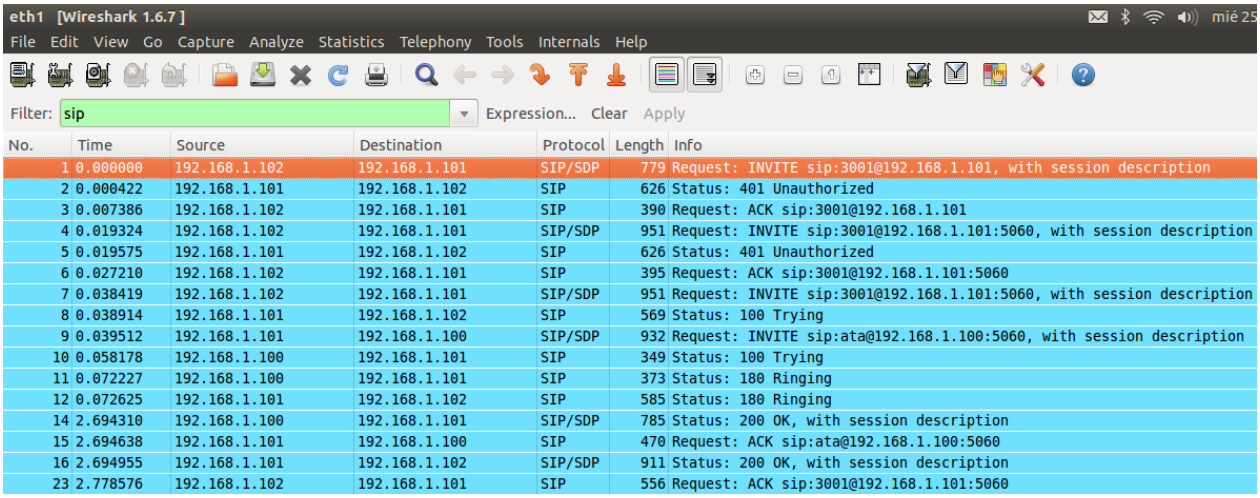
En la figura 15 se muestran 3 estados de la GUI, en la figura 15(a) Se marca al teléfono convencional con la extensión 3001, en la figura 15(b) el teléfono convencional toma la llamada y se establece el audio, en la figura 15(c) se finaliza la llamada.



(a) (b) (c)

Figura 15. Establecimiento de llamada de Aplicación móvil a Teléfono convencional

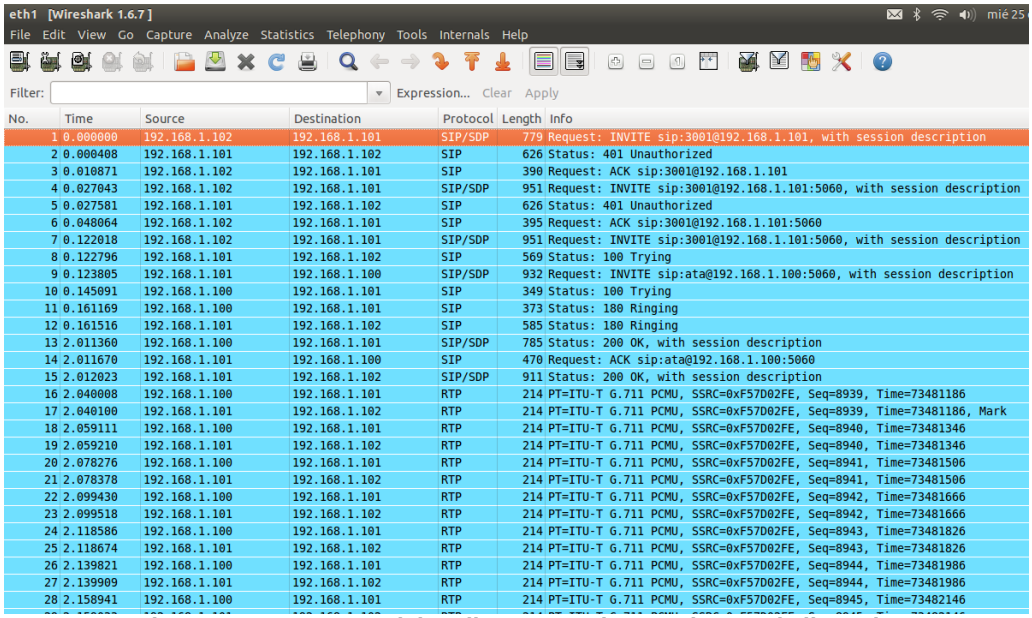
En la figura 16, con ayuda de Wireshark se observa el intercambio de mensajes cuando la aplicación móvil realiza la llamada al teléfono convencional. Haciendo la lectura de izquierda a derecha:
Línea 8, la aplicación con IP 192.168.1.102 envía una petición *Invite* al servidor 192.168.1.101, este responde con un *Trying* de intentando
Línea 10, el servidor le envía la invitación al teléfono convencional con IP 192.168.1.100
Línea 11, el teléfono convencional envía un mensaje al servidor de *Trying*,
Línea 12, el teléfono envía una respuesta de sonando (*Ringing*)
Línea 15, el teléfono convencional responde la llamada indicando un *200 OK*



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.102	192.168.1.101	SIP/SDP	779	Request: INVITE sip:3001@192.168.1.101, with session description
2	0.000422	192.168.1.101	192.168.1.102	SIP	626	Status: 401 Unauthorized
3	0.007386	192.168.1.102	192.168.1.101	SIP	390	Request: ACK sip:3001@192.168.1.101
4	0.019324	192.168.1.102	192.168.1.101	SIP/SDP	951	Request: INVITE sip:3001@192.168.1.101:5060, with session description
5	0.019575	192.168.1.101	192.168.1.102	SIP	626	Status: 401 Unauthorized
6	0.027210	192.168.1.102	192.168.1.101	SIP	395	Request: ACK sip:3001@192.168.1.101:5060
7	0.038419	192.168.1.102	192.168.1.101	SIP/SDP	951	Request: INVITE sip:3001@192.168.1.101:5060, with session description
8	0.038914	192.168.1.101	192.168.1.102	SIP	569	Status: 100 Trying
9	0.039512	192.168.1.101	192.168.1.100	SIP/SDP	932	Request: INVITE sip:ata@192.168.1.100:5060, with session description
10	0.058178	192.168.1.100	192.168.1.101	SIP	349	Status: 100 Trying
11	0.072227	192.168.1.100	192.168.1.101	SIP	373	Status: 180 Ringing
12	0.072625	192.168.1.101	192.168.1.102	SIP	585	Status: 180 Ringing
14	2.694310	192.168.1.100	192.168.1.101	SIP/SDP	785	Status: 200 OK, with session description
15	2.694638	192.168.1.101	192.168.1.100	SIP	470	Request: ACK sip:ata@192.168.1.100:5060
16	2.694955	192.168.1.101	192.168.1.102	SIP/SDP	911	Status: 200 OK, with session description
23	2.778576	192.168.1.102	192.168.1.101	SIP	556	Request: ACK sip:3001@192.168.1.101:5060

Figura 16. Envío de petición de llamada desde la aplicación móvil al teléfono convencional

Cuando el teléfono convencional toma la llamada el canal RTP se establece para el transporte de audio, en la figura 17 se muestra la captura de wireshark



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.102	192.168.1.101	SIP/SDP	779	Request: INVITE sip:3001@192.168.1.101, with session description
2	0.000408	192.168.1.101	192.168.1.102	SIP	626	Status: 401 Unauthorized
3	0.010871	192.168.1.102	192.168.1.101	SIP	390	Request: ACK sip:3001@192.168.1.101
4	0.027043	192.168.1.102	192.168.1.101	SIP/SDP	951	Request: INVITE sip:3001@192.168.1.101:5060, with session description
5	0.027581	192.168.1.101	192.168.1.102	SIP	626	Status: 401 Unauthorized
6	0.048064	192.168.1.102	192.168.1.101	SIP	395	Request: ACK sip:3001@192.168.1.101:5060
7	0.122018	192.168.1.102	192.168.1.101	SIP/SDP	951	Request: INVITE sip:3001@192.168.1.101:5060, with session description
8	0.122796	192.168.1.101	192.168.1.102	SIP	569	Status: 100 Trying
9	0.123805	192.168.1.101	192.168.1.100	SIP/SDP	932	Request: INVITE sip:ata@192.168.1.100:5060, with session description
10	0.145091	192.168.1.100	192.168.1.101	SIP	349	Status: 100 Trying
11	0.161169	192.168.1.100	192.168.1.101	SIP	373	Status: 180 Ringing
12	0.161516	192.168.1.101	192.168.1.102	SIP	585	Status: 180 Ringing
13	2.011360	192.168.1.100	192.168.1.101	SIP/SDP	785	Status: 200 OK, with session description
14	2.011670	192.168.1.101	192.168.1.100	SIP	470	Request: ACK sip:ata@192.168.1.100:5060
15	2.012023	192.168.1.101	192.168.1.102	SIP/SDP	911	Status: 200 OK, with session description
16	2.040008	192.168.1.100	192.168.1.101	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0xF57D02FE, Seq=8939, Time=73481186
17	2.040100	192.168.1.101	192.168.1.102	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0xF57D02FE, Seq=8939, Time=73481186, Mark
18	2.059111	192.168.1.100	192.168.1.101	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0xF57D02FE, Seq=8940, Time=73481346
19	2.059210	192.168.1.101	192.168.1.102	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0xF57D02FE, Seq=8940, Time=73481346
20	2.078276	192.168.1.100	192.168.1.101	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0xF57D02FE, Seq=8941, Time=73481506
21	2.078378	192.168.1.101	192.168.1.102	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0xF57D02FE, Seq=8941, Time=73481506
22	2.099430	192.168.1.100	192.168.1.101	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0xF57D02FE, Seq=8942, Time=73481666
23	2.099518	192.168.1.101	192.168.1.102	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0xF57D02FE, Seq=8942, Time=73481666
24	2.118586	192.168.1.100	192.168.1.101	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0xF57D02FE, Seq=8943, Time=73481826
25	2.118674	192.168.1.101	192.168.1.102	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0xF57D02FE, Seq=8943, Time=73481826
26	2.139821	192.168.1.100	192.168.1.101	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0xF57D02FE, Seq=8944, Time=73481986
27	2.139909	192.168.1.101	192.168.1.102	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0xF57D02FE, Seq=8944, Time=73481986
28	2.158941	192.168.1.100	192.168.1.101	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0xF57D02FE, Seq=8945, Time=73482146

Figura 17. Transporte del audio a cargo de RTP durante la llamada

En la figura 18 se muestra la finalización de la llamada por parte de la aplicación.

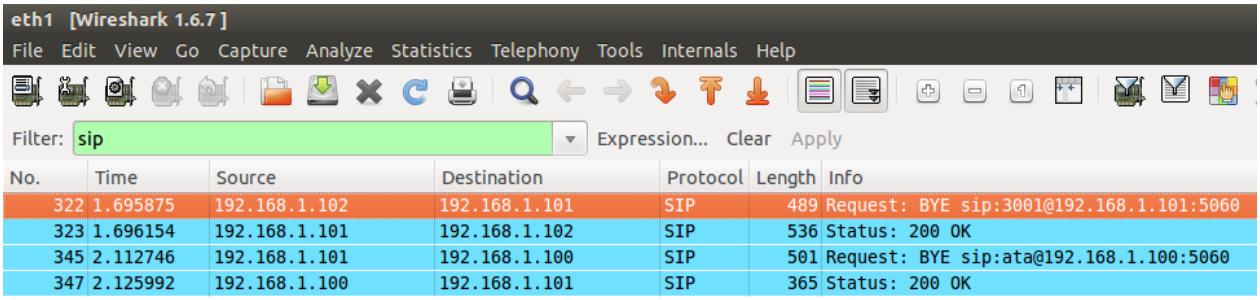


Figura 18. Aplicación finaliza llamada, enviando una petición BYE

Por último se muestra la finalización de la llamada por parte del teléfono convencional (ver figura 19)

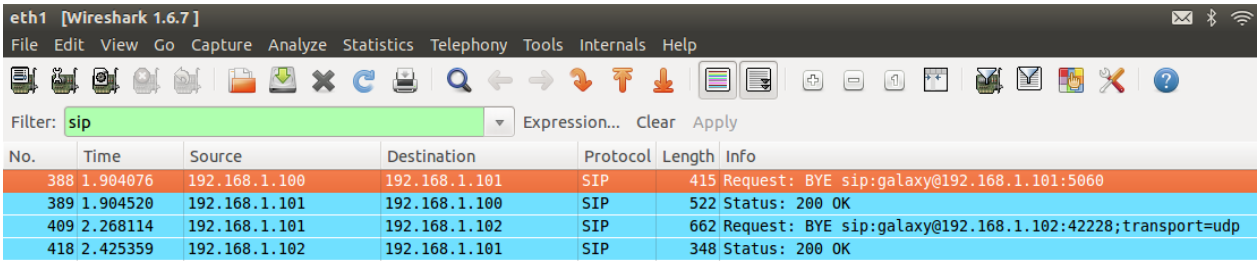
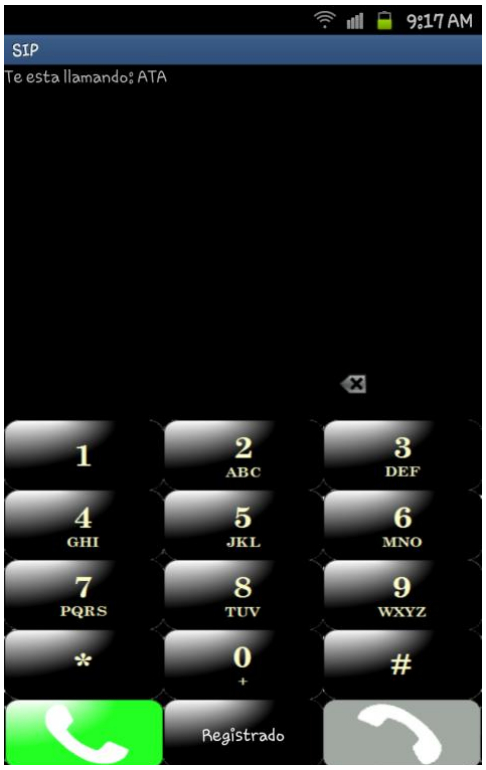


Figura 19. Teléfono convencional finaliza la llamada, enviando una petición BYE

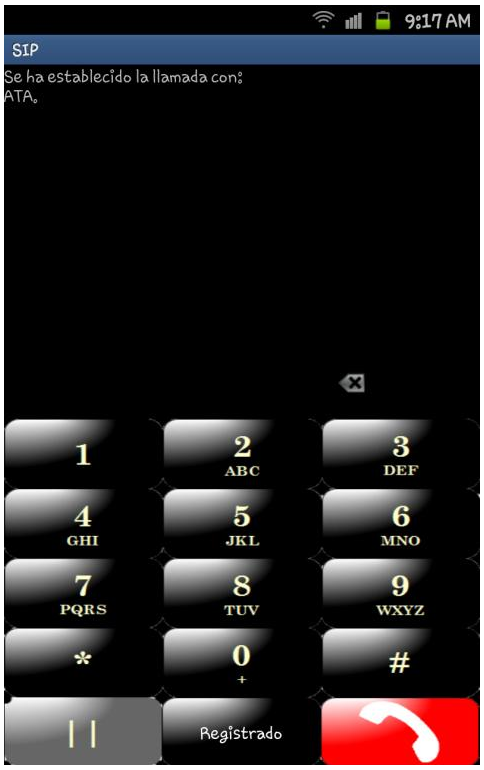
Con ayuda de *wireshark* se verificó que el intercambio de mensajes durante el establecimiento de una llamada realizada desde la aplicación al teléfono convencional, respeta la secuencia de mensajes y la negociación, como lo describe la figura 14. Se puede confirmar el transporte de audio por medio del protocolo RTP.

Aplicación móvil recibe llamada del Teléfono Convencional.

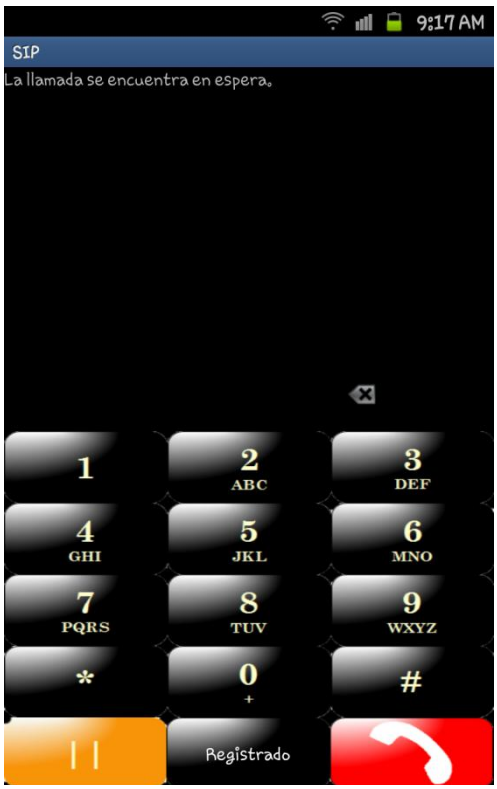
En la figura 20 se muestra la GUI de la aplicación cuando recibe una llamada del teléfono convencional. En la figura 20(a) se notifica que el teléfono convencional está llamando, fig. 20(b) Muestra el momento cuando el usuario contesta la llamada, fig. 20(c) Muestra la llamada en espera, fig. 20(d) Finaliza la llamada.



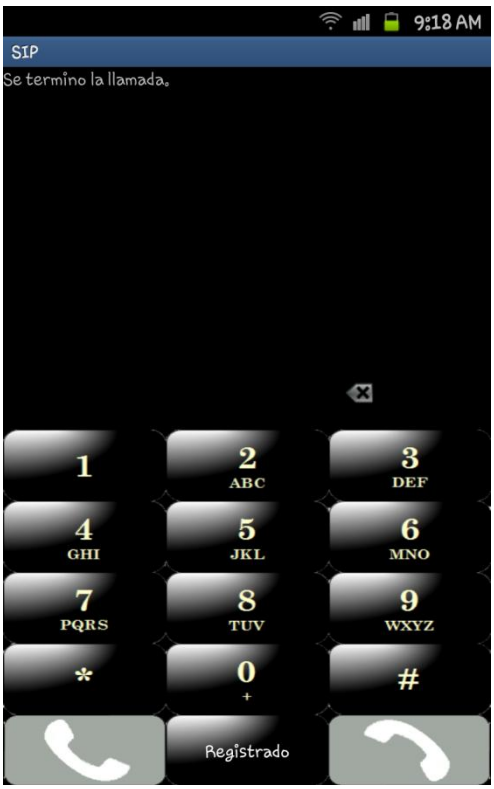
(a) Aplicación recibiendo llamada



(b) Aplicación contesta llamada



(c) Llamada en espera



(d) Finaliza la llamada

Figura 20. GUI de la aplicación, cuando recibe una llamada del teléfono convencional

Las capturas correspondientes en *wireshark* muestran el intercambio de mensajes cuando el teléfono convencional realiza una llamada al dispositivo móvil.

El teléfono convencional con la IP 192.168.1.100, envía una petición *Invite* al dispositivo móvil con IP 192.168.1.102, en la línea 6 se puede verificar que el servidor ha enviado la petición Invite del dispositivo ATA al dispositivo móvil.

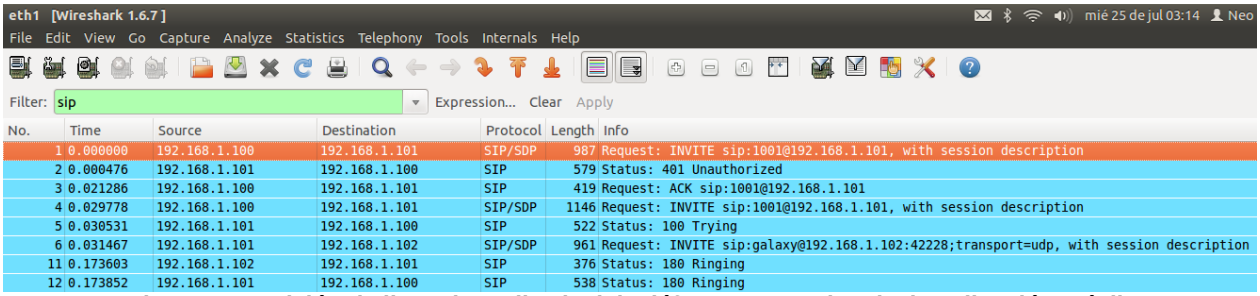


Figura 21. Petición de llamada realizada del teléfono convencional a la aplicación móvil

En la figura 22 se muestra la captura de *wireshark* en el momento que la aplicación móvil contesta la llamada realizada por el teléfono convencional.
En la línea 11 se puede observar el mensaje de aceptación de la llamada.

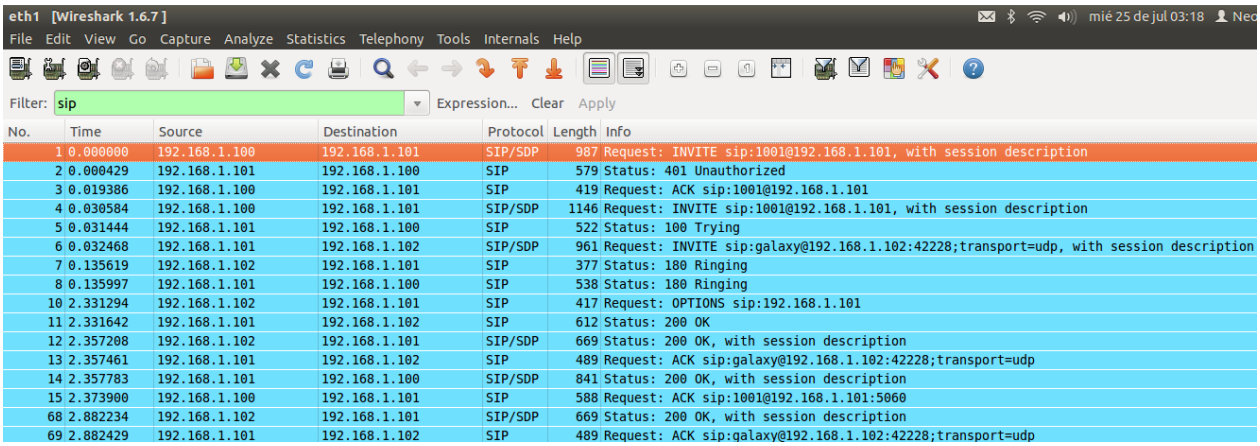


Figura 22. Aplicación móvil contesta llamada realizada por el teléfono convencional

En la figura 23 se muestra el transporte de audio a cargo de RTP después de haber respondido la aplicación móvil.

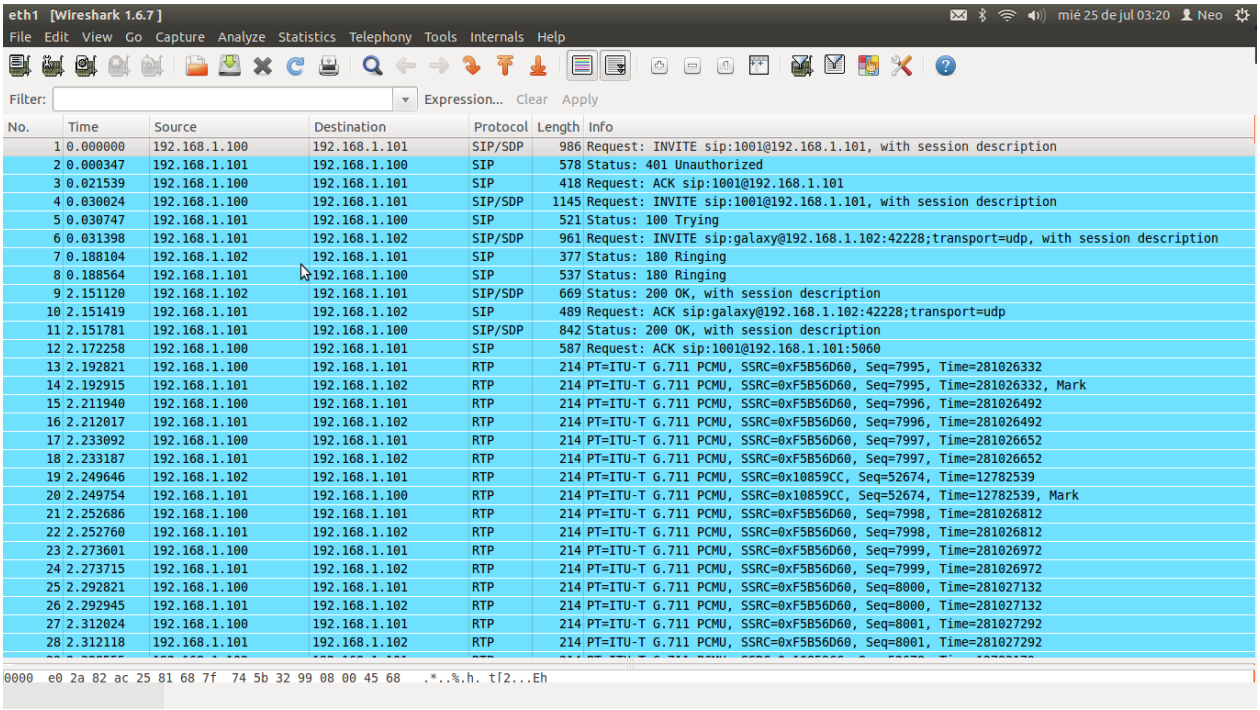


Figura 23. Transmisión de audio con el protocolo RTP durante la llamada realizada por el teléfono convencional

En la figura 24 se muestra como el dispositivo Android finaliza la llamada con una petición *BYE* que es enviada al servidor y este se encarga de enviarla al teléfono convencional

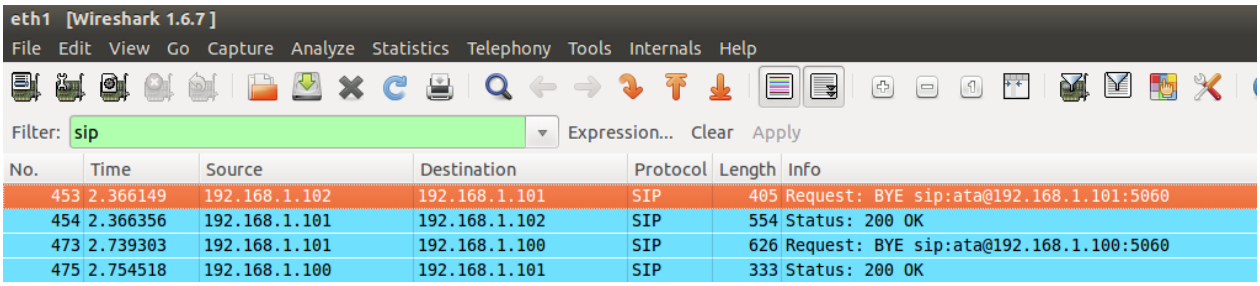


Figura 24. Aplicación móvil finaliza llamada, enviando una petición BYE

En la figura 25 se muestra como el dispositivo Android finaliza la llamada con una petición *BYE* que es enviada al servidor y este se encarga de enviarla al teléfono convencional

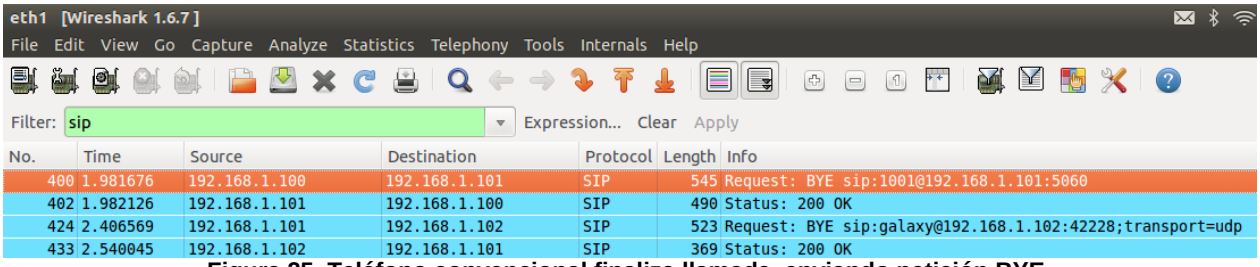


Figura 25. Teléfono convencional finaliza llamada, enviando petición BYE

Aplicación móvil realiza llamada a un Softphone

En las figuras 26 se muestra la GUI de la aplicación durante una llamada realizada desde la aplicación al Softphone.

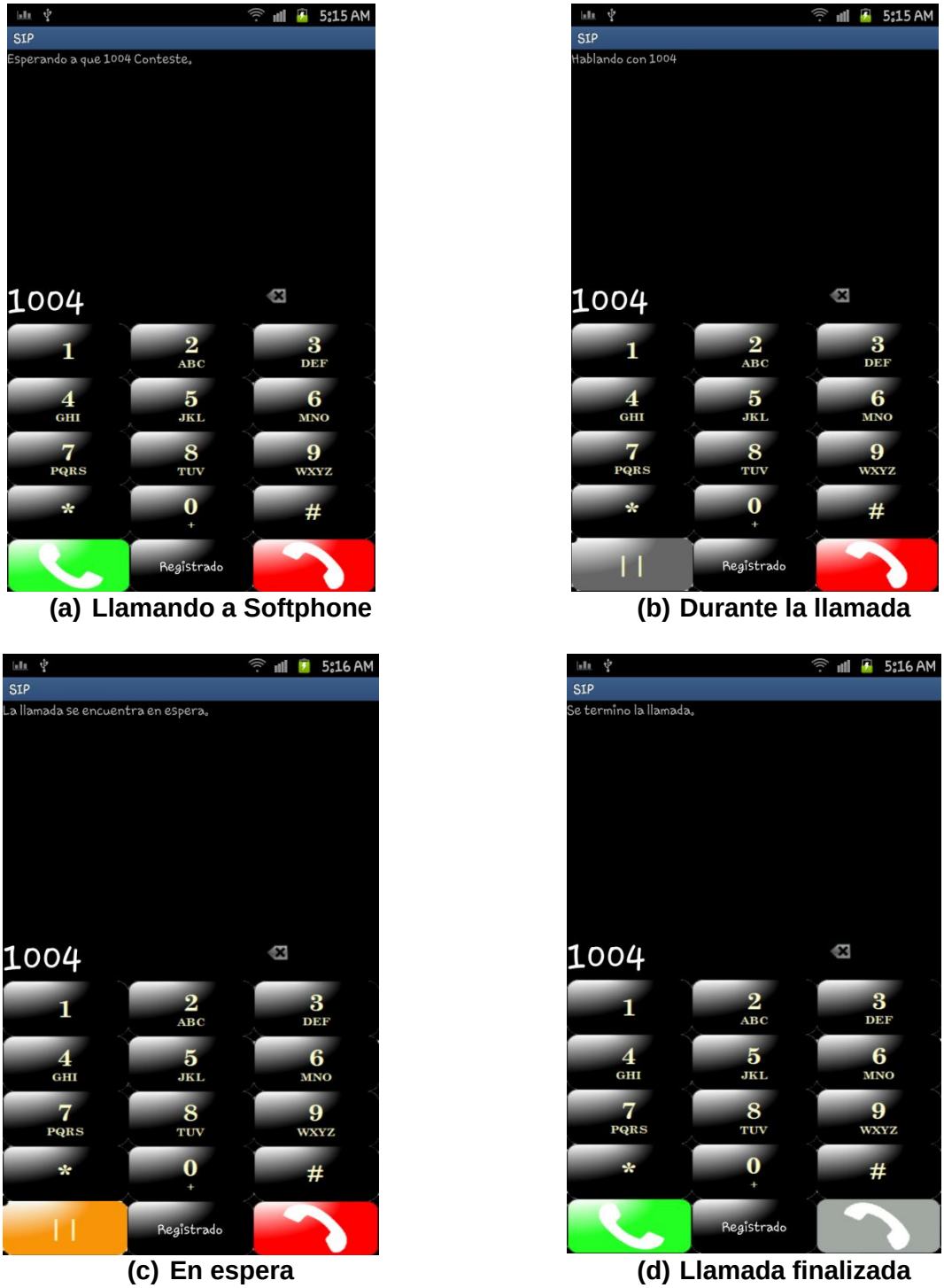


Figura 26. GUI de aplicación durante una llamada realizada al Softphone.

En la figura se muestra el envío de peticiones y respuestas para establecer comunicación desde la aplicación móvil con IP 192.168.1.101 al Softphone con IP 192.168.1.104, en este caso el servidor ahora tiene la IP 192.168.1.102.

En la línea 9 se observa como el dispositivo móvil envía la petición de realizar llamada al softphone. En la línea 15 el softphone envía una respuesta de aceptación y en la línea 18 se observa cómo se envía el audio en el protocolo RTP.

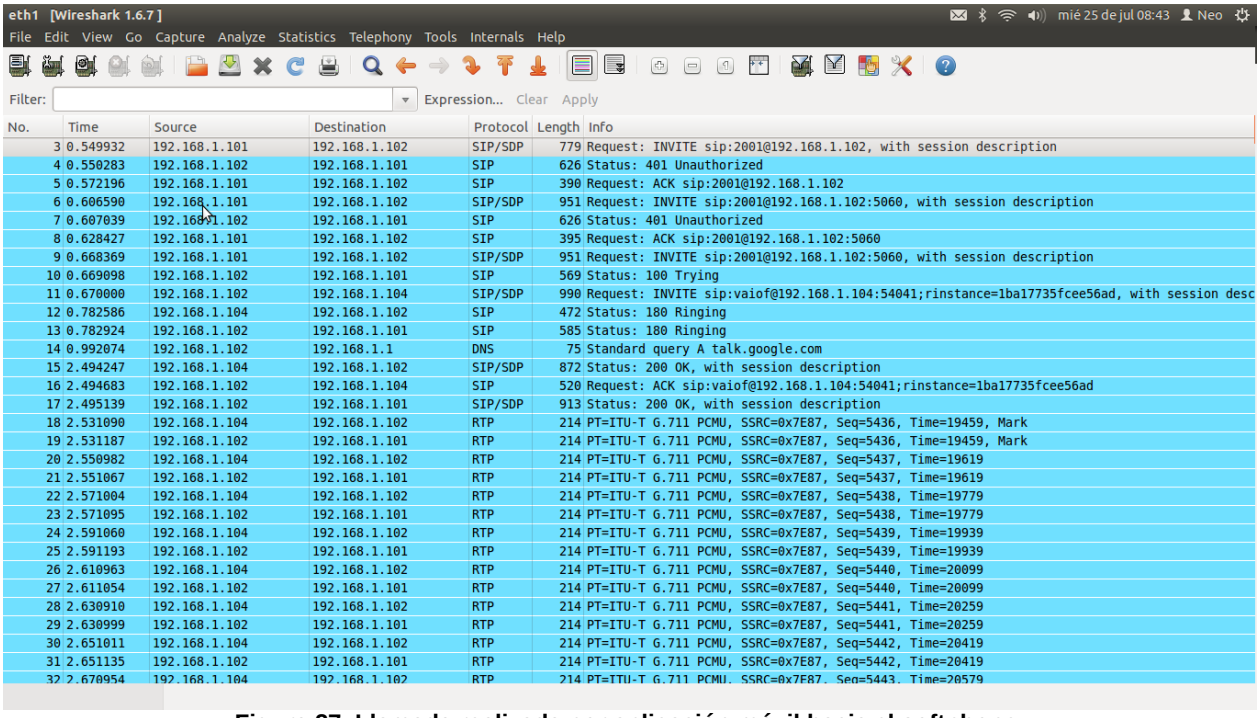
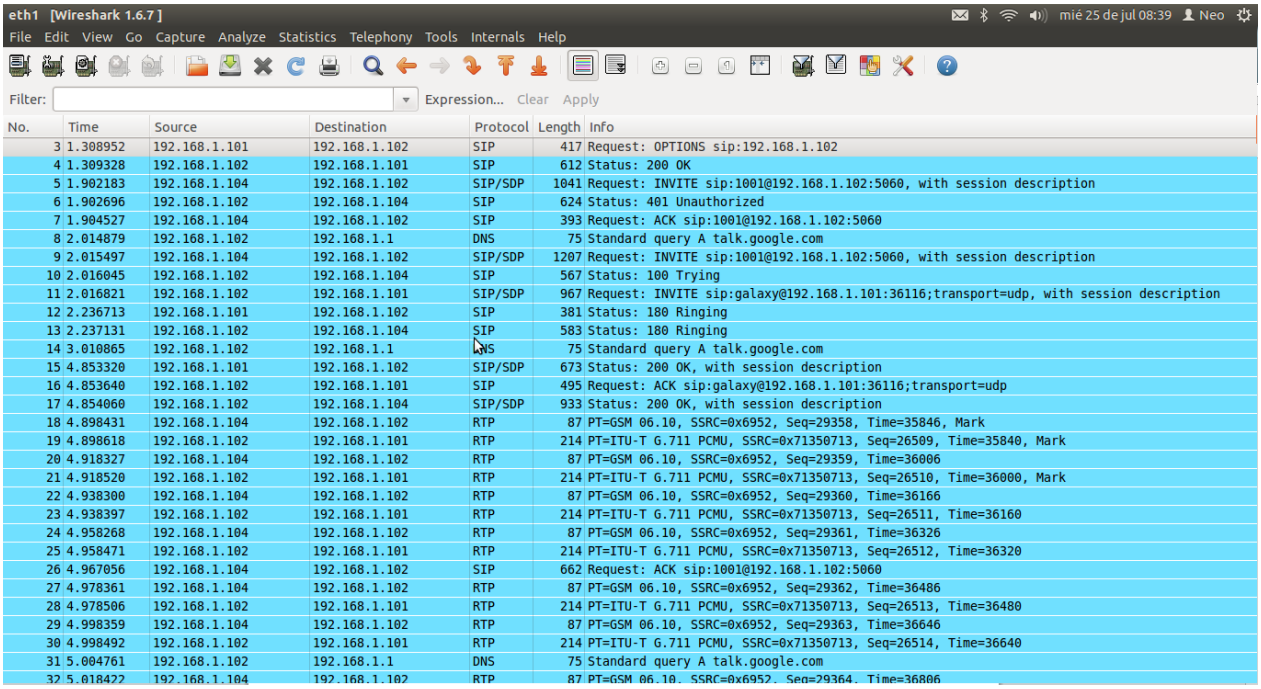


Figura 27. Llamada realizada por aplicación móvil hacia el softphone

Aplicación móvil recibe llamada de un Softphone

En la figura 28 se muestra el intercambio de mensajes cuando un Softphone realiza una llamada a la aplicación.
Leyendo de izquierda a derecha, en la línea 9 el que envía la petición Invite es el Softphone con Ip 192.168.1.104, en la línea 15 la aplicación contesta la llamada y notifica con una respuesta *200 OK.* En la línea 18 se observa como el audio es transportado por el protocolo RTP.



No.	Time	Source	Destination	Protocol	Length	Info
3	1.308952	192.168.1.101	192.168.1.102	SIP	417	Request: OPTIONS sip:192.168.1.102
4	1.309328	192.168.1.102	192.168.1.101	SIP	612	Status: 200 OK
5	1.902183	192.168.1.104	192.168.1.102	SIP/SDP	1041	Request: INVITE sip:1001@192.168.1.102:5060, with session description
6	1.902696	192.168.1.102	192.168.1.104	SIP	624	Status: 401 Unauthorized
7	1.904527	192.168.1.104	192.168.1.102	SIP	393	Request: ACK sip:1001@192.168.1.102:5060
8	2.014879	192.168.1.102	192.168.1.1	DNS	75	Standard query A talk.google.com
9	2.015497	192.168.1.104	192.168.1.102	SIP/SDP	1207	Request: INVITE sip:1001@192.168.1.102:5060, with session description
10	2.016045	192.168.1.102	192.168.1.104	SIP	567	Status: 100 Trying
11	2.016821	192.168.1.102	192.168.1.101	SIP/SDP	967	Request: INVITE sip:galaxy@192.168.1.101:36116;transport=udp, with session description
12	2.236713	192.168.1.101	192.168.1.102	SIP	381	Status: 180 Ringing
13	2.237131	192.168.1.102	192.168.1.104	SIP	583	Status: 180 Ringing
14	3.010865	192.168.1.102	192.168.1.1	DNS	75	Standard query A talk.google.com
15	4.853320	192.168.1.101	192.168.1.102	SIP/SDP	673	Status: 200 OK, with session description
16	4.853640	192.168.1.102	192.168.1.101	SIP	495	Request: ACK sip:galaxy@192.168.1.101:36116;transport=udp
17	4.854060	192.168.1.102	192.168.1.104	SIP/SDP	933	Status: 200 OK, with session description
18	4.898431	192.168.1.104	192.168.1.102	RTP	87	PT=GSM 06.10, SSRC=0x6952, Seq=29358, Time=35846, Mark
19	4.898618	192.168.1.102	192.168.1.101	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x71350713, Seq=26509, Time=35840, Mark
20	4.918327	192.168.1.104	192.168.1.102	RTP	87	PT=GSM 06.10, SSRC=0x6952, Seq=29359, Time=36006
21	4.918520	192.168.1.102	192.168.1.101	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x71350713, Seq=26510, Time=36000, Mark
22	4.938300	192.168.1.104	192.168.1.102	RTP	87	PT=GSM 06.10, SSRC=0x6952, Seq=29360, Time=36166
23	4.938397	192.168.1.102	192.168.1.101	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x71350713, Seq=26511, Time=36160
24	4.958268	192.168.1.104	192.168.1.102	RTP	87	PT=GSM 06.10, SSRC=0x6952, Seq=29361, Time=36326
25	4.958471	192.168.1.102	192.168.1.101	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x71350713, Seq=26512, Time=36320
26	4.967056	192.168.1.104	192.168.1.102	SIP	662	Request: ACK sip:1001@192.168.1.102:5060
27	4.978361	192.168.1.104	192.168.1.102	RTP	87	PT=GSM 06.10, SSRC=0x6952, Seq=29362, Time=36486
28	4.978506	192.168.1.102	192.168.1.101	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x71350713, Seq=26513, Time=36480
29	4.998359	192.168.1.104	192.168.1.102	RTP	87	PT=GSM 06.10, SSRC=0x6952, Seq=29363, Time=36646
30	4.998492	192.168.1.102	192.168.1.101	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x71350713, Seq=26514, Time=36640
31	5.004761	192.168.1.102	192.168.1.1	DNS	75	Standard query A talk.google.com
32	5.018422	192.168.1.104	192.168.1.102	RTP	87	PT=GSM 06.10, SSRC=0x6952, Seq=29364, Time=36806

Figura 28. Llamada realizada por el Softphone hacia la aplicación móvil

CONCLUSIONES

En este proyecto todos los objetivos particulares se cumplieron satisfactoriamente, además se construyeron los módulos que permitieron realizar una aplicación para enviar y recibir llamadas de audio sobre el protocolo SIP, cumpliendo con los requerimientos mínimos establecidos para dar por concluido el proyecto, refiriéndonos a que el audio en la comunicación es muy aceptable y la comunicación se establece de forma correcta.

En cuanto al módulo de solicitud de permisos que se implementó para poder usar la característica de VoIP, cumple con las especificaciones descritas en la documentación de Android, pero los dispositivos utilizados para las pruebas, de cierta forma tienen limitada o bloqueada la característica de VoIP, debido a que nuestro proyecto utilizó la API SIP que proporciona Android se tiene una desventaja con los dispositivos móviles de la marca Samsung y Sony ya que estas dos compañías restringen el uso de VoIP y SIP, esto fue una limitante ya que al momento de hacer uso de la API, no contábamos con los permisos necesarios para utilizar estas dos funcionalidades con las que cuenta nativamente Android, para ello fue necesario liberar los teléfonos para forzar su uso.

Como se forzó el uso de los dispositivos móviles para hacer uso de VoIP y SIP la aplicación desarrollada en este proyecto no es portable en otros dispositivos de las mismas compañías que utilicen Android, para lograr la portabilidad de la aplicación, es probable que al diseñar nuestra propia API SIP en lenguaje Java se pueda tener la misma funcionalidad de SIP que proporciona Android. Dado que este no es el objetivo principal de este proyecto, queda como una posible mejora a futuro de este proyecto.

En este proyecto se implementaron y aprovecharon las características de tecnologías muy poderosas, la programación de alto nivel, plataformas de código abierto y uso de protocolos estándares, éstos nos permiten crear herramientas poderosas y útiles para dar solución a problemas cotidianos como la forma de comunicarnos y de transmitir datos. Esto es posible ya que este tipo de tecnologías ofrecen a los usuarios y desarrolladores acceso a su documentación para poder desarrollar o implementarlas.

BIBLIOGRAFÍA

[1]J. Gómez López y F. Gil Montoya, *VoIP y Asterisk: redescubriendo la Telefonía,* Almería: Alfaomega Ra-ma, 2008, ISBN 978-84-7897-902-8

[2]W. Frank Ableson y Robi Sen, *Android Guía para Desarrolladores,* segunda ed. España: Anaya Multimedia, 2011, ISBN 978-84-415-2958-8

[3]J.M Huidrobo, *Integración de voz y datos: Call Centers, Tecnología y Aplicaciónes,* Madrid: Mac Graw Hill, 2003, ISBN 84-481-3850-3

[4]Reto Meier, *Professional Android 2 Application Development,* Indianapolis: Wiley Publishing, Inc, 2010, ISBN 978-0-470-56552-0

[5]Mark L. Murphy, *Beginning Android 2:* Apress, 2010, ISBN, 978-1-4302-2629-1

[6]Sayed Y. Hashimi y Satya Komatineni, *Pro Android 2:* Apress, 2010, ISBN 978-1-4302-2659-8

[7]S. Tanenbaum , *Redes de computadoras,* cuarta edición.Mexico: Pearson Education, 2003, ISBN 970-26-0162-2

APÉNDICE

Configuración de Asterisk

Instalación de Asterisk en Debian 6

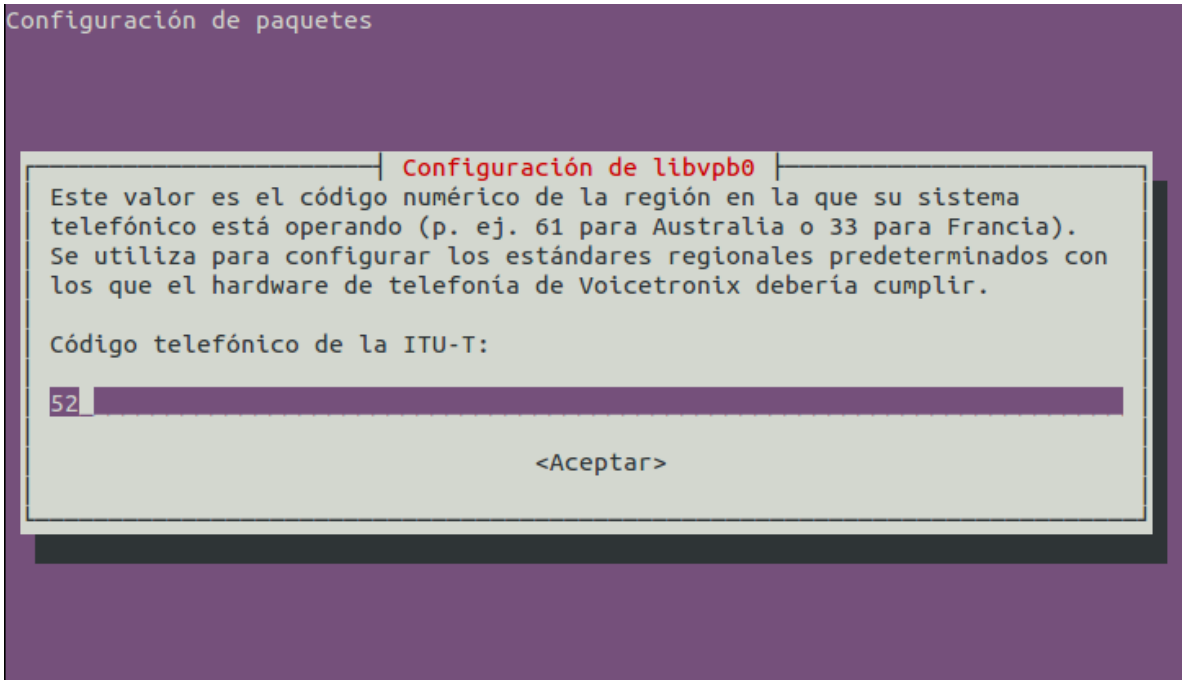
Desde la terminal buscar el paquete de Asterisk

```
sudo apt-cache search asterisk
```

Para instalar Asterisk

```
sudo apt-get install asterisk
```

Durante la instalación se configura el prefijo telefónico internacional. Estos números son asignados por la ITU (Unión Internacional de Telecomunicaciones), los cuales se encuentran divididos en zonas del 1 al 9, para el caso de México es la zona 5 (Hispanoamericana, Brasil y El Caribe) y el prefijo es el 52.



Al instalar Asterisk se crean muchos directorios y cada uno contiene una parte de Asterisk, el directorio más importante es /etc/asterisk, contiene archivos de

configuración, así como el archivo asterisk.conf, donde se indica la ubicación de los demás directorios.

Configuración de Canales SIP

La configuración de los dispositivos SIP se realiza en el fichero *sip.conf*. Dentro de este archivo se definen los usuarios SIP, para ello Asterisk cuenta con 3 tipos de usuarios diferentes:

- **peer**: los peers son los usuarios a los que Asterisk manda llamadas, es decir, Asterisk llama A un peer.
- **user:** los user son los usuarios de los que Asterisk recibe llamadas, es decir, Asterisk recibe llamadas DE un user.
- **friend:** los friend son la agrupación de los dos usuarios anteriores, es decir, un friend, es un peer y un user a la vez.

En esta aplicación lo usuarios SIP serán de tipo **friend,** ya que nos interesa tanto recibir como hacer llamadas a diferentes usuarios dentro de nuestra red.

```
[usuario1]
type=friend
callerid=Usuario 1
username=usuario1
secret=pass
context=extensions
host=dynamic
```

El ejemplo anterior muestra como se define un usuario dentro de Asterisk, además del tipo de usuario, se agregan otros parámetros, a continuación se explicarán cada uno de ellos:

- **secret**: Indica la contraseña que se utilizará para la autentificación.
- **context**: Indica el contexto que se aplicará a este usuario.
- **callerid:** Fija el identificador al momento de llamar, es decir, cuando *usuario1* llame a alguien, el que recibe la llamada vera “Usuario 1” en la pantalla de su terminal.
- **host**: Indica la IP del usuario. En nuestra configuración utilizaremos ‘dynamic’, esto permite que el usuario pueda registrarse con la IP asignada por el DHCP de la red.
- **canreinvite:** deshabilita re-invites.

Cuando se realice un cambio en el fichero *sip.conf,* es necesario actualizar la configuración SIP de Asterisk, para ello primero iniciamos Asterisk desde una terminal.

Arrancar Asterisk

Para arrancar Asterisk basta con ejecutar el siguiente comando como root en una terminal

```
sudo asteriks
```

Asterisk arrancará en segundo plano, es decir, no se mostrará ningún mensaje al usuario, y éste podrá seguir trabajando, mientras Asterisk sigue funcionando.

Conectarse a una instancia Arrancada

En la sección anterior, al arrancar Asterisk, éste se inicia en segundo plano para conectarse la interface de línea de comandos (CLI) es necesario utilizar el modificador –r. Además del modificador –r podemos conectarnos con otras dos opciones de verbose y debug, para que nos muestre más información detallada del funcionamiento de Asterisk, mientras que el debug muestra información a más bajo nivel, que es útil para diagnosticar fallos en Asterisk.

Para añadir verbose es necesario arrancar Asterisk con –v y para añadir debug con –d. Entre más v y d se añadan más información se mostrará.

A continuación se muestra un ejemplo:

```
sudo asteriks -rvvvvvvvvvvvvvvddddd
```

Consola de línea de comandos (CLI)

Después de realizar un cambio en el archivo *sip.conf*, es necesaria actualizar la configuración SIP de Asterisk, para que los comandos tengan efecto. Para ello se ejecuta sip reload en la interface de línea de comandos.

```
root*CLI> sip reload
Reloading SIP
 == Parsing '/etc/asterisk/sip.conf':   ==
Found
 == Parsing '/etc/asterisk/users.conf':   ==
Found
 == Using SIP CoS mark 4
```

Para consultar los usuarios que existen en el sistema, así como su estado, podemos ejecutar *sip show peers* y *sip show users desde la línea de comandos (CLI)*

```
root*CLI> sip show peers
Name/username      Host            Dyn   ACL  Port   Status
usuario1/usuario1  192.168.1.10    D           5060
usuario2/usuario2  192.168.1.11    D           5060
2 sip peers [Monitored: 0 online, 0 offline Unmonitored: 0 online, 2 offline]
```

```
localhost*CLI> sip show users
Username  Secret    Accountcode  Def.Context  ACL  ForcerPort
usuario1  usuario1              extensions   No   No
usuario2  usuario2              extensions   No   No
```

Contextos, extensiones y prioridades

Una vez configurados los dispositivos SIP, para que los usuarios puedan llamarse, es necesario establecer un dialplan. El dialplan de Asterisk se encuentra en el archivo *extensions.conf*

Las extensiones son números que el usuario es capaz de marcar.

Los contextos son agrupaciones lógicas de extensiones, y se utilizan para dividir el dialplan en diversos entes lógicos. Esta división es necesaria para disponer de un dialplan mantenible, escalable y con posibilidades de ofrecer diversos entornos de mercado aislados.

Sintaxis

La sintaxis para las extensiones de un contexto es la siguiente

```
exten  => numero de extensión, prioridad, aplicación
```

Seguido de la palabra clave *exten* => se expresa el número de extensión.

Una aplicación se le denomina aquellos módulos que realizan algún tipo de acción sobre algún canal.

```
exten=>1002,1,Dial(SIP/usuario1)
```

Al marcar la extensión 1002 se ejecutará la aplicación Dial, al a que indicamos que llame el usuario1, que es un usuario SIP, para ellos indicamos primero la tecnología (SIP/) y después el nombre del usuario que se ha definido en el archivo *sip.conf*

El siguiente ejemplo muestra un dialplan completo que incluye tres contextos, el contexto1 y el contexto2 los cuales incluyen las extensiones 1001,1002, 2001 y 2002. El contexto3, incluye los otros 2 contextos, y fue el que se asigno a las extensiones en el archivo *sip.conf.*

```
[contexto1]
exten=>1001,1,Dial(SIP/usuario1)
exten=>1002,1,Dial(SIP/usuario2)

[contexto2]
exten=>2001,1,Dial(SIP/usuario3)
exten=>2002,1,Dial(SIP/usuario4)

[contexto3]
include=>contexto1
include=> contexto2
```

Configuración Linksys Wireless-G Broadband Router

El router permite acceder a Internet a través de una conexión inalámbrica, transmite a velocidades de hasta 54 Mbps, o por medio de sus 4 puertos conmutados. También puede usar el router para compartir recursos tales como computadores impresoras y archivos. La configuración del router es fácil utilizando para ello un explorador.

1, 2, 3, 4 Estos LEDs numerados, corresponden a los puertos numerados en el panel posterior del router, tienen dos finalidades. Si el LED permanece encendido, el router está correctamente conectado a un dispositivo a través de ese puerto. Un LED que parpadea indica la actividad de red en ese puerto.

Boton de Configuracion Wi-Fi Protected si se cuenta con dispositivos inalámbricos que soportan configuración Wi-Fi Protected se puede configurar automáticamente la seguridad inalámbrica de la red.

Wireless El led Wireless se ilumina cuando la función inalámbrica está habilitada. Si el LED parpadea el router está enviando o recibiendo datos a través de la red.

Internet El LED de Internet se enciende cuando hay conexión a través del puerto de Internet. Si el LED parpadea indica actividad de rede en el puerto de internet.

Power El LED de encendido se ilumina y se mantendrá mientras el router está encendido.

Internet Este puerto es donde se conecta el cable o la conexión DSL a Internet.

1, 2, 3, 4 Estos puertos Ethernet conectan el router a los equipos de computo de la red cableada y otros dispositivos de red Ethernet.

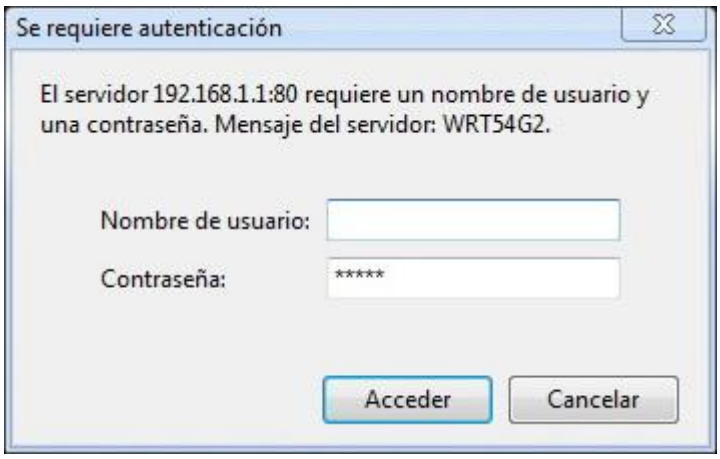
Reset Permite restaurar la configuración de fabrica del equipo.

Power El puerto de alimentación es donde se conecta el adaptador de corriente.

Configuración

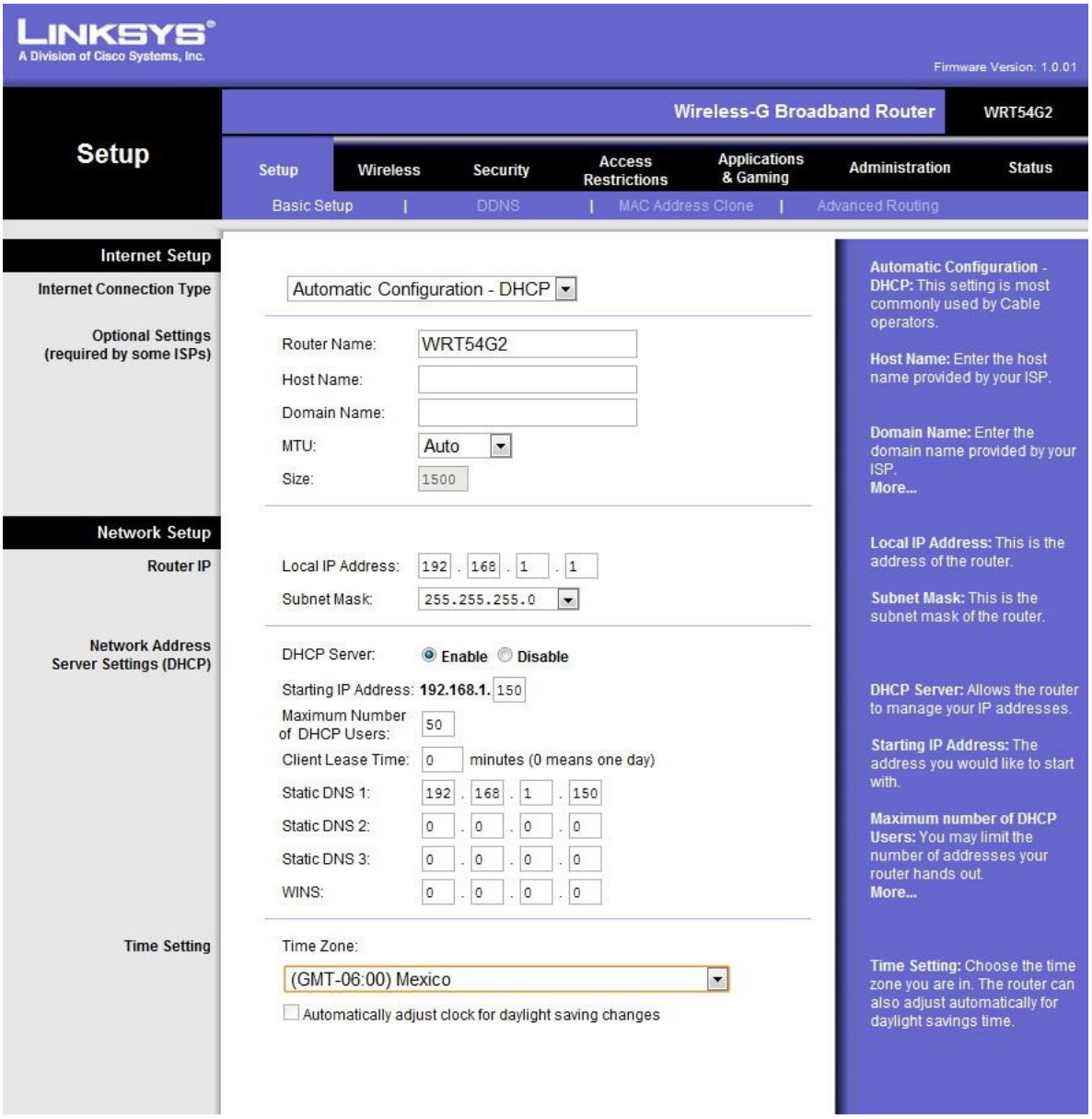
Para acceder a la utilidad basada en Web, inicie el navegador web en su computadora, e introduzca la dirección del router IP por defecto, 192.168.1.1, en el campo Dirección. A continuación, pulse Intro.

Al momento de acceder una pantalla le solicitará una contraseña. El nombre de usuario debe estar en blanco. La primera vez que se accede a la utilidad basada en Web, la contraseña predeterminada es “admin”.



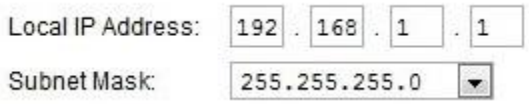
Configuración de la Red

La primera pantalla que aparece es la pantalla de configuración básica. Esto le permite cambiar la configuración general del router.



IP del Router

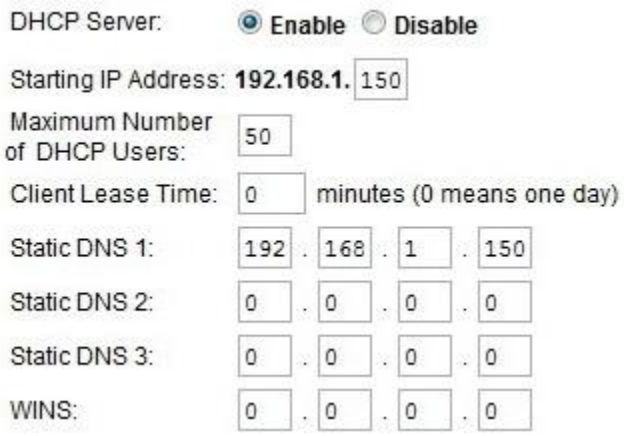
A continuación se muestra la dirección del Router y la máscara de subred del router.



Configuración del servidor de direcciones de Red (DHCP)

Las siguientes opciones permiten configurar la función de servidor DHCP (Dynamic Host Configuration Protocol). El router puede ser utilizado como un servidor

DHCP para la red. Un servidor DHCP asigna automáticamente una dirección IP a cada computadora en la red.



DHCP Server El servidor DHCP está habilitado por defecto de fábrica.

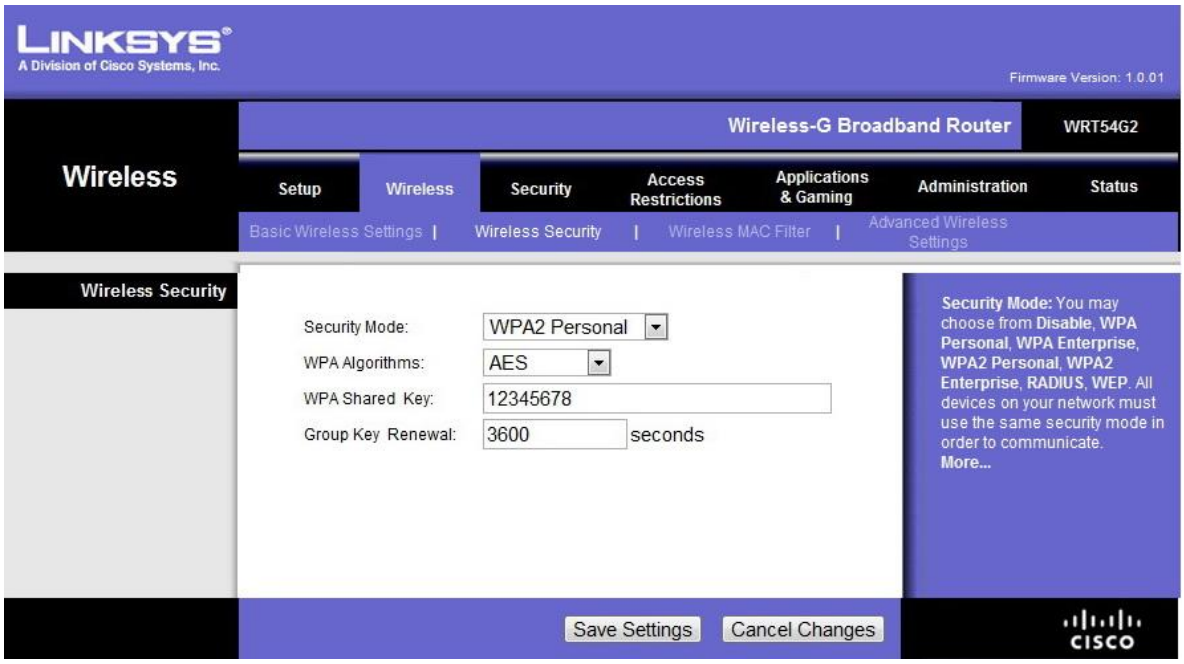
Starting IP Address Valor de inicio de las direcciones IP. El valor diferentes valores que se pueden utilizar van desde 192.168.1.2 hasta 192.168.1.254

Maximum Number of DHCP Users El número máximo de equipos que se desea que el servidor DHCP asigne direcciones IP.

Client Lease Time es la cantidad de tiempo que un usuario de la red podrá estar conectado al router con su dirección IP dinámica actual.

Static DNS El DNS (Domain Name System) permite traducir nombres de domino o paginas Web en direcciones de Internet.

Configuración de la seguridad Wi-Fi



En la configuración de la seguridad Wi-Fi se uilizo WPA2

WPA Algorithm WPA admite dos métodos de encriptación, TKIP y AES, con claves de codificación dinámicas. AES es un método de encriptación más fuerte que TKIP.

WPA Shared Key Introduzca la clave compartida por el router y los dispositivos de red. Se debe tener 8-63 caracteres.

Group Key Renewal Introduzca un período de renovación de claves, que le dice al routercon qué frecuencia se debe cambiar las claves de cifrado. El valor predeterminado es 3600 segundos.

Configuración de ATA PAP2 de Linksys CISCO

EL ATA PAP2 de Linksys cuenta con dos puertos para conectar teléfonos convencionales, un puerto Ethernet para conectar el ATA a la red y la salida de alimentación de corriente.

Conexión.

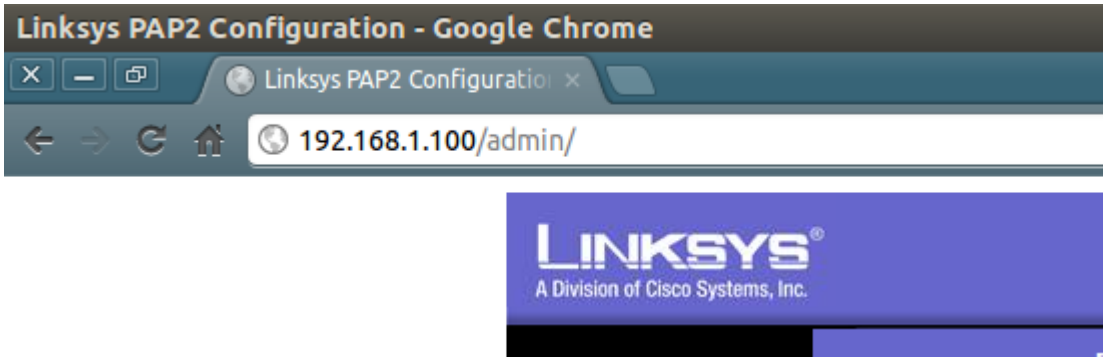
1. Conectar el puerto Ethernet hacia el Access Point o routeador

2. Conectar uno de los puertos RJ11 al teléfono convencional, en este caso al puerto Phone1

3. Conectar a corriente eléctrica

Configuración.

1. Para obtener la IP asignada por el DHCP del Access Point o Router, marca desde el teléfono convencional 4 asteriscos (****), una voz grabada en ingles anuncia el menú de configuración
2. Marca la opción 110#. Posteriormente una voz te menciona la IP que tiene asignada en el ATA, generalmente es la dirección 192.168.1.100
3. Abre un explorador web y en la barra de búsqueda ingrese la dirección IP

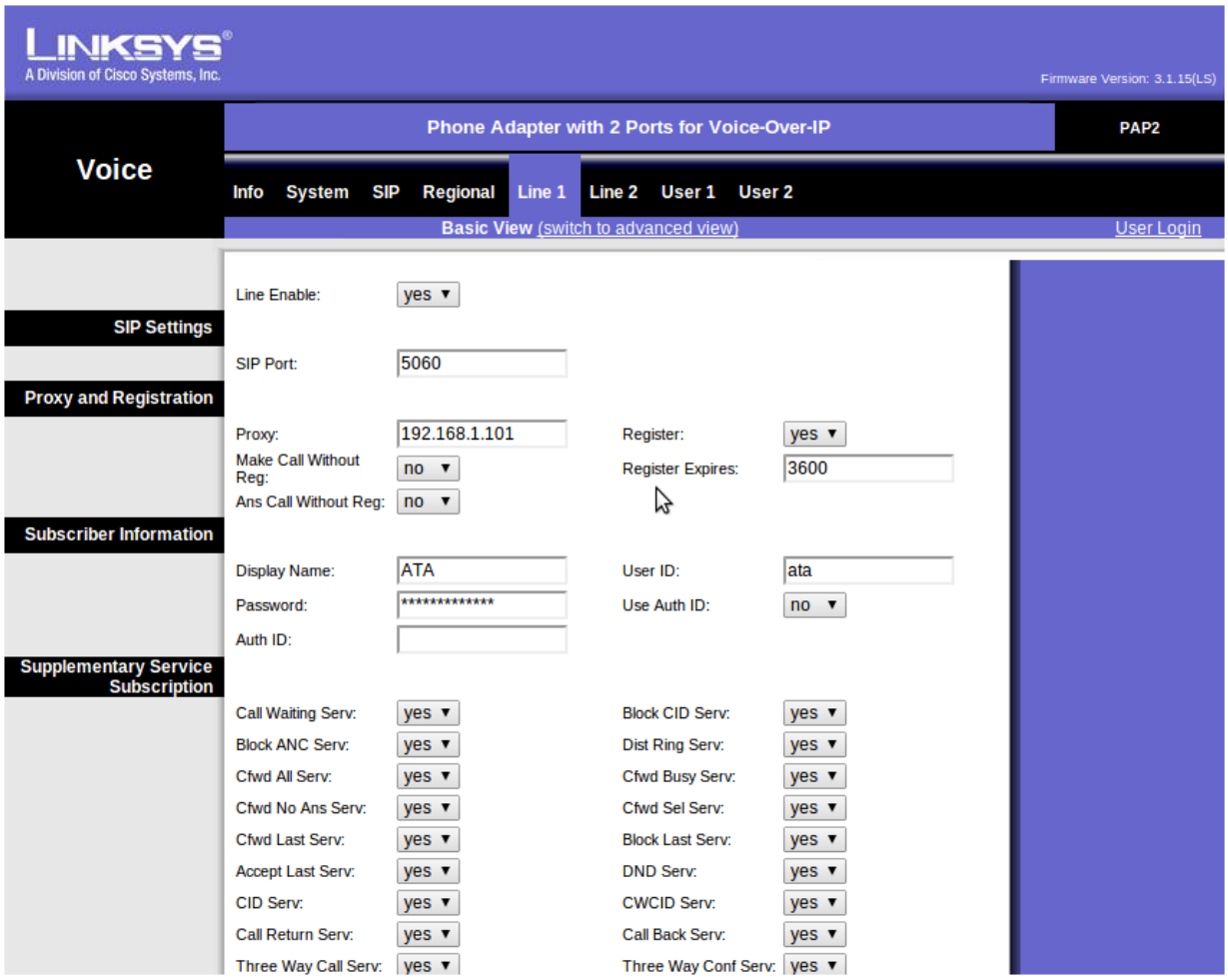


4. Se debe abrir la interfaz de configuración. Selecciona la opción *UserLogin*, aparece arriba a la derecha de la interfaz.
5. Selecciona *Line1* para que aparezca la interfaz donde se configuran los datos de la cuenta SIP para el teléfono convencional
6. Hay que marcar la línea como habilitada y asegurar que el puerto SIP es 5060

7. Ingresa la dirección IP del Servidor Asterisk

8. Ingresa el nombre de usuario, la contraseña y nombre a mostrar de la cuenta SIP, esta información debe estar también debe estar almacenada en Asterisk.

Interfaz para configuración de ATA PAP2 de Linksys usando un explorador web



Al terminar guardamos los cambios. Para esto el ATA se reinicia automáticamente y debe detectar el teléfono convencional conectado, el servidor valida la información y de ser correcta se realiza el registro del ATA en el servidor y al levantar el auricular debe haber tono de marcado.

Manual de Usuario de la Aplicación

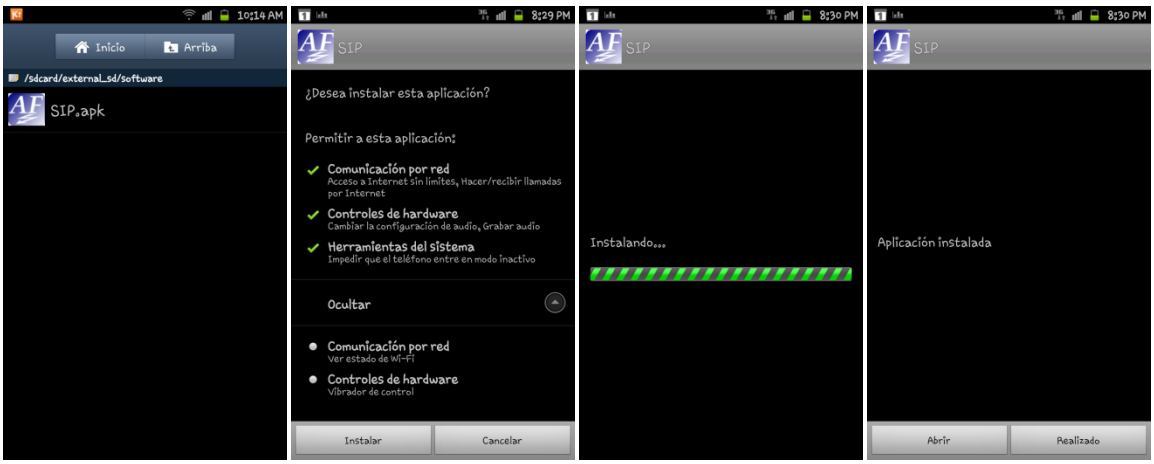
Requerimientos

El dispositivo móvil donde se instalará la aplicación debe contar con:

- Sistema operativo Android 2.3 o superior.
- Contar con conexión de datos Wi-Fi.
- Una cuenta SIP, previamente configurada en Asterisk.

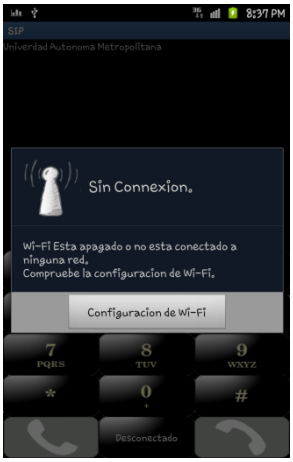
Instalación de la aplicación

A) Para realizar la instalación debe copiar al dispositivo móvil el archivo SIP.apk, el cual le permitirá instalar la aplicación. B) Al abrir el archivo se le solicita que autorice los permisos necesarios para que la aplicación sea instalada. C) Al dar siguiente la aplicación será instalada en el dispositivo D) y al finalizar se mostrará el siguiente mensaje, “Aplicación instalada” lo que indica que la instalación se realizó exitosamente.

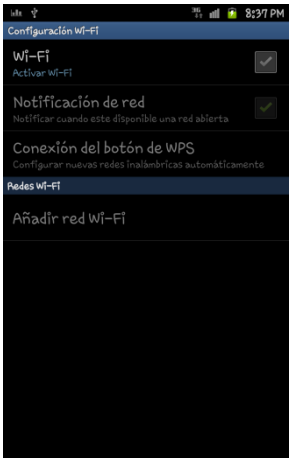


A) B) C) D)

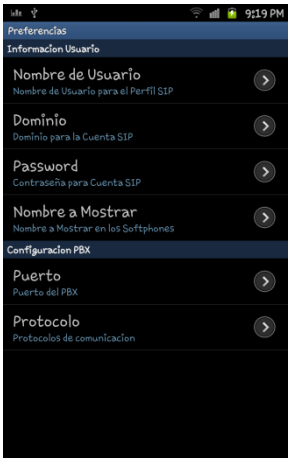
Primeros pasos dentro de la aplicación



Después de haber instalado la aplicación si no se encuentra encendido el Wi-Fi del dispositivo o si no se encuentra conectado a ninguna red inalámbrica, se le indica que revise la configuración inalámbrica del dispositivo.



Para poder hacer uso de la aplicación es necesario tener encendido el Wi-Fi y estar conectado a la misma red que el servidor Asterisk, de lo contrario no se podrá recibir ni realizar llamadas.

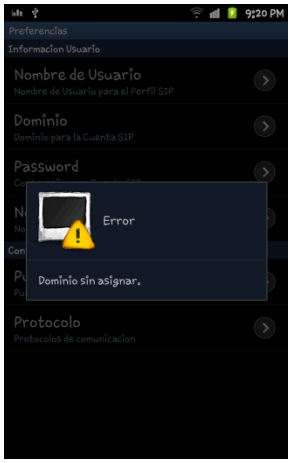


Cuando termine de configurar el Wi-Fi del dispositivo se le indica que ingrese Nombre, Dominio y Password de la cuenta SIP que se tenga asignada en el Servidor Asterisk. A continuación se muestra la tabla descripción de los botones de configuración del Usuario y del Servidor PBX en este caso será Asterisk.

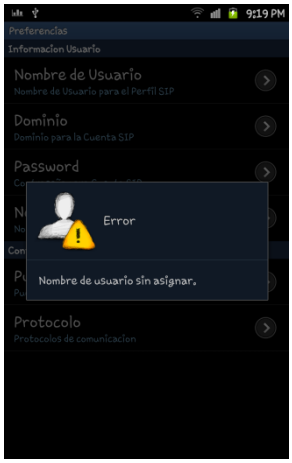
Tabla de descripción de botones de configuración

Descripción botones Configuración Usuario y Configuración Asterisk	
Nombre de Usuario	Nombre de usuario del perfil SIP dado de alta en Asterisk.
Domino	Dirección IP o Nombre del servidor Asterisk.
Password	Contraseña del Perfil SIP.
Nombre a Mostrar	Nombre que se mostrará a los dispositivos cuando se realice una llamada.
Puerto	Número de puerto del Servidor Asterisk
Protocolo	Protocolo usado para conectar con el Servidor Asterisk

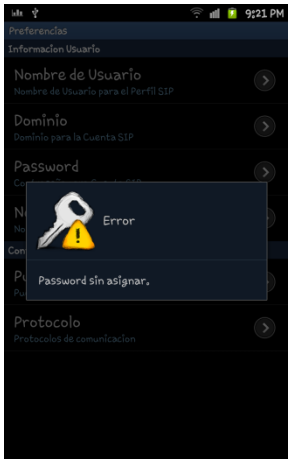
Si alguno de los campos, Nombre de E) Usuario, F) Domino y G) Password es omitido se mostrarán los siguientes mensajes informando que campo falta por completar.



E)

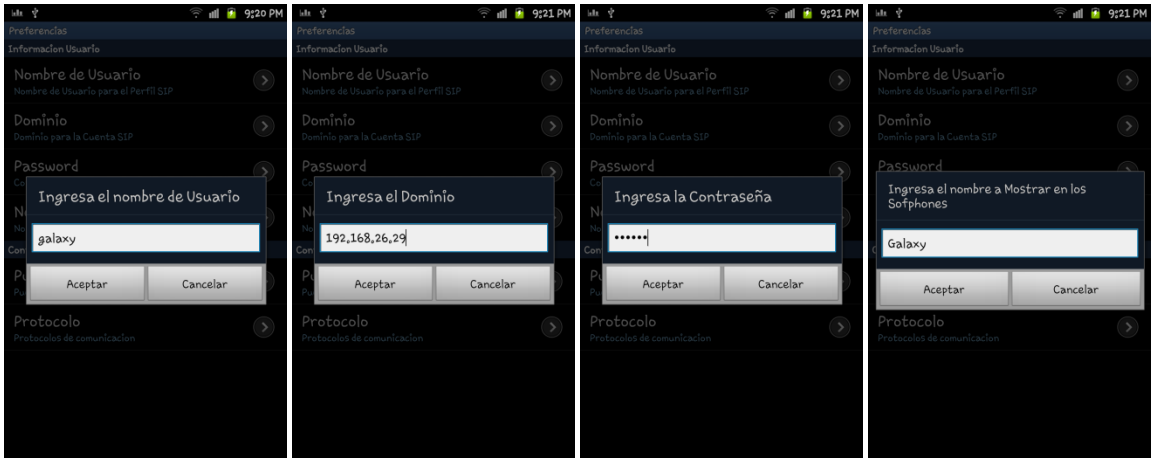


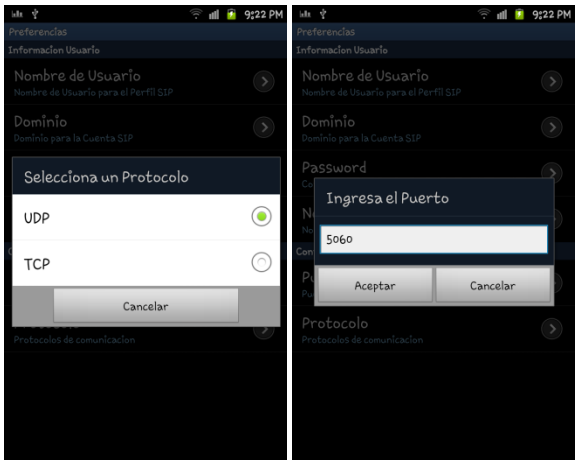
F)



G)

A continuación se muestra un ejemplo de la forma correcta de llenar los campos de configuración del Usuario y del Servidor Asterisk.



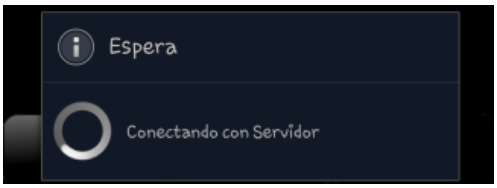


Registro con el Servidor

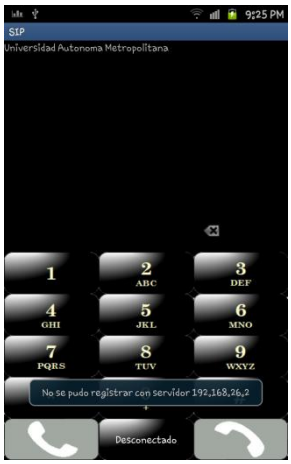
Al momento que presione el botón *back* del dispositivo, con la información proporcionada en la sección anterior el dispositivo intentará conectarse con el Servidor Asterisk, esto le permitirá realizar o recibir llamadas.

Durante el proceso de Registro con el Servidor se muestran los siguientes diálogos indicando el progreso del Registro

Cuando el dispositivo intenta conectarse con el servidor, se muestra un dialogo de espera, indicandole que se está intentando registrar con el servidor Asterisk.



Cuando se realiza el registro con el servidor Asterisk exitosamente se muestra un dialogo que le indica, el registro exitoso con el servidor Asterisk.



En caso contrario de que no se pueda realizar un registro con el servidor se muestra el siguiente mensaje que le indica el fallo del registro con el servidor Asterisk

Realizar una llamada

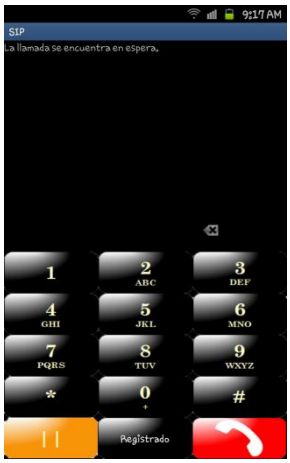
Para realizar una llamada es necesario que conozca la extensión a la que se desea comunicarse, las extensiones están asignadas en el servidor Asterisk.

Para comenzar a marcar presione un digito del 0 al 9 incluidos los caracteres * y #. Una vez presionado un digito o un carácter el botón de llamar se cambia a color verde Indicando que se puede realizar una llamada a la extensión que se marco, por ejemplo 1002.

La longitud de la extensión no debe ser mayor a seis dígitos.

Si ha ingresado una extensión y la aplicación se ha registrado previamente al servidor Asterisk, presione el botón verde de llamar, esto iniciara la llamada a la extensión indicada. Si la extensión existe se muestra en la pantalla el siguiente mensaje: Esperando a que 1004 conteste, y el botón de colgar se enciende a color rojo.

Cuando la extensión par conteste a la llamada se muestra el siguiente mensaje. Hablando con 1004. Esto indica que se ha establecido la comunicación con la extensión par exitosamente, El botón de llamar se cambia por un botón de pausa en color gris, esto le permitirá puede poner en pausar la llamada actual si así lo desea.

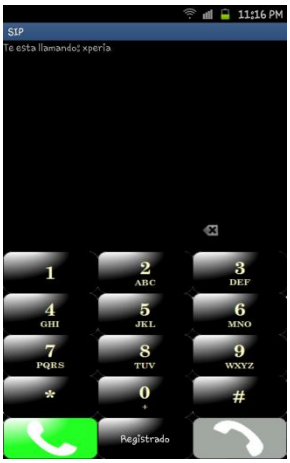


Para poner en espera la llamada actual debe presionar el botón de pausa que está en color gris, la llamada actual se pone pondrá en espera y el botón de espera cambia a color amarillo indicando que la llamada se encuentra en espera, además se muestra el siguiente mensaje. La llamada se encuentra en espera.

Para terminar la llamada actual debe presionar el botón rojo, esto termina la llama actual y le permite realizar o recibir llamadas entrantes nuevamente.

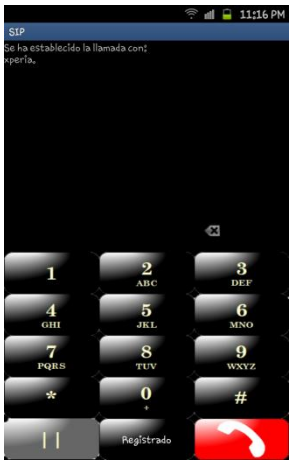
Recibir una llamada

Cuando se registra el dispositivo al Servidor Asterisk, puede recibir llamadas entrantes de otros perfiles registrados al servidor Asterisk.

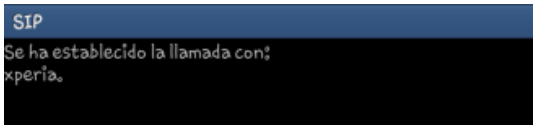


Si se recibe una llamada de otro perfil, se mostrará en la pantalla un mensaje indicando que ha entrado una llamada con el nombre del perfil que está llamando, además se reproducirá un sonido y el teléfono comenzará a vibrar.

Para contestar a la llamada solo se debe presionar el botón de contestar que se encuentra en color verde, esto apaga el sonido y deja de vibrar para permitir la comunicación.

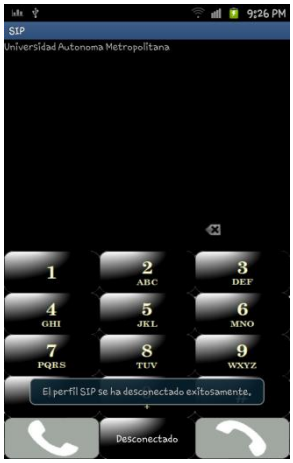


Cuando se recibe la llamada entrante se cambiará el botón de contestar a un botón de pausa en color gris y un mensaje indicando que se ha establecido la llamada con el nombre del perfil que está llamando.



Desconectarse del Servidor

El registro al servidor le permite dar de alta al dispositivo móvil con el servidor Asterisk, esto le da la posibilidad de realizar y recibir llamadas con perfiles registrados previamente con el servidor Asterisk. Cuando ya no se desee realizar o recibir llamadas debe presionar el botón Registrado el cual desconecta al perfil previamente registrado en el servidor.



Cuando se desconecta el dispositivo del Servidor Asterisk se muestra el siguiente mensaje, “El perfil Sip se ha desconectado exitosamente” indicando la desconexión exitosa del servidor Asterisk.