Softphone con soporte IAX2 para Android

Víctor Peinó Díaz

- Memoría Proyecto Fin de Carrera -

ÍNDICE DE CONTENIDOS

Índice de contenidos 1
Índice de figuras 3
1 Introducción 5
2 Descripción del proyecto 5
3 Objetivos del proyecto 5
4 Plan de trabajo 6
4.1 Actividades 6
4.2 Calendario 9
5 El protocolo IAX2 10
5.1 Comparación entre SIP y IAX2 11
5.2 Tramas IAX2 13
5.2.1 Trama full 14
5.2.2 Trama mini 16
5.2.3 Trama Meta 16
5.3 Elementos de información (IE) 17
5.4 Mensajes IAX2 19
5.5 Operaciones IAX2 23
6 PBX Asterisk 26
6.1 Características generales 26
6.2 Instalación y configuración 27
6.3 Pruebas de funcionamiento 28
7 Android 33
8 Desarrollo de la aplicación 35
8.1 Pruebas con el soporte SIP 35
8.2 Interfaz gráfica de la aplicación 37
8.3 Busqueda de soporte IAX2 40
9 Ideas básicas para implementar soporte IAX2 en Android 41
9.1 Tramas, elementos de información y mensajes 41

9.2 Envío y recepción de datagramas 43
9.3 Captura y reproducción de audio 44
9.4 Operaciones Iax2 44
9.5 Eventos y operaciones relacionadas con las llamadas 45
10 Integración de contactos del teléfono con la aplicación 46
11 Estructura y composición de los ficheros de implementación 48
12 Conclusión 50
Glosario de términos 51
Bibliografía/Webgrafía 52

ÍNDICE DE FIGURAS

Ilustración 1 Calendario 9
Ilustración 2 Trama Full 14
Ilustración 3 Trama Mini 16
Ilustración 4 Elemento de información 17
Ilustración 5 Operaciones de suministro y descarga de firmware 23
Ilustración 6 Operación de registro 24
Ilustración 7 Establecimiento de llamada 24
Ilustración 8 Establecimiento fallido de llamada 25
Ilustración 9 Fin de llamada 25
Ilustración 10 Supervisión de llamada 25
Ilustración 11 Optimización de llamada 26
Ilustración 12 Administración FreePBX 28
Ilustración 13 Zoiper 29
Ilustración 14 Intercambio de registro 29
Ilustración 15 Esquema de registro 30
Ilustración 16 Intercambio de llamada 30
Ilustración 17 Esquema de llamada 31
Ilustración 18 Intercambio de registro (red externa) 31
Ilustración 19 Intercambio de llamada (red externa 32
Ilustración 20 Componentes fundamentales Android 33
Ilustración 21 Comprobación soporte SIP en el emulador 36
Ilustración 22 SipDemo registrada 36
Ilustración 23 SipDemo en llamada 37
Ilustración 24 Pantalla inicial Iax2Droid 38
Ilustración 25 Dialer Iax2Droid 38
Ilustración 26 Settings Iax2Droid 39
Ilustración 27 Mensaje de error Iax2Droid 39
Ilustración 28 Applet registrado 40

Ilustración 29 Applet en llamada 41
Ilustración 30 Lista de contactos 47
Ilustración 31 Marcado extension del contacto 47

1 INTRODUCCIÓN

La proliferación de dispositivos móviles de altas prestaciones (*smartsphones*) surgidos en los últimos años combinados con la contratación por parte de sus usuarios de tarifas de datos ha posibilitado la utilización de aplicaciones denominadas softphones con el propósito de registrarse a través de Internet en centralitas independientes de los proveedores de telefonía y realizar llamadas, consiguiendo de esta manera esquivar los costes que provocan la realización de las llamadas por medio de estas. Muchos de estos proveedores son reacios al empleo de sus redes de datos para trafico VoIP por lo que intentan frenar su uso con medidas como por ejemplo el uso de firewalls y de NAT en sus routers.

2 DESCRIPCIÓN DEL PROYECTO

El proyecto consiste a grandes rasgos en la realización de un softphone para Android empleando para ello el protocolo IAX2 soportado por la PBX Asterisk.

El pasado mes de diciembre fue liberada la versión 2.3 (Gingerbread) del Sistema Operativo para dispositivos móviles Android de Google. En esta versión una de las novedades era un nuevo API que permite realizar llamadas mediante SIP (Session Initiation Protocol).

El protocolo SIP emplea tres puertos de comunicaciones, una se dedica a la señalización, en concreto el puerto 5060, y otras dos para el flujo de audio RTP. Uno de los problemas más comunes que se pueden originar utilizando SIP es el producido cuando el tráfico VoIP debe pasar a través de routers que implementan NAT ya que estos no siempre son capaces de saber que conexiones empleadas para el tráfico de audio están asociadas a la conexión de señalización SIP. Mediante la sustitución por el protocolo IAX2 se consigue solucionar este problema con NAT y otros originados a la hora de enfrentarse con firewalls. El protocolo IAX2 utiliza un único puerto de comunicaciones, el UDP 4569, tanto para la señalización como para el flujo de audio, con lo que la señalización y el audio de las llamadas se multiplexa por el mismo canal, con lo que es menos susceptible a problemas con NAT.

3 OBJETIVOS DEL PROYECTO

El objetivo principal del proyecto es el estudio del protocolo IAX2 y su puesta en funcionamiento en dispositivos móviles donde los operadores emplean NAT y firewalls

para intentar impedir que se realicen comunicaciones VoIP a través de sus redes de datos.

Para llevar a cabo este propósito es necesario:

Estudio de las tecnologías implicadas

Instalación y configuración de una PBX Asterisk

Desarrollo de un softphone con soporte IAX2 para la plataforma Android

4 PLAN DE TRABAJO

Un requisito previo indispensable para la realización del proyecto consiste en el desarrollo de una planificación coherente para el mismo.

Las actividades que han sido incluidas en este plan de proyecto se definen a partir de unos objetivos y para unos tiempos concretos.

A partir de esta especificación se llevará a cabo la evaluación del progreso del proyecto.

4.1 ACTIVIDADES

Nombre	Definición del alcance del proyecto
Descripción	Definición del alcance general del proyecto y establecimiento de los objetivos finales detallando las tareas que se han de llevar a cabo así como el calendario de las mismas.
Duración	10 días
Resultado	Plan de trabajo

Nombre	Estudio del protocolo IAX2
Descripción	Descripción del protocolo IAX2, descripción de sus ventajas e inconvenientes y realización de una comparativa con otros protocolos similares como SIP.
Duración	7 días
Resultado	Capítulo en la memoria de trabajo

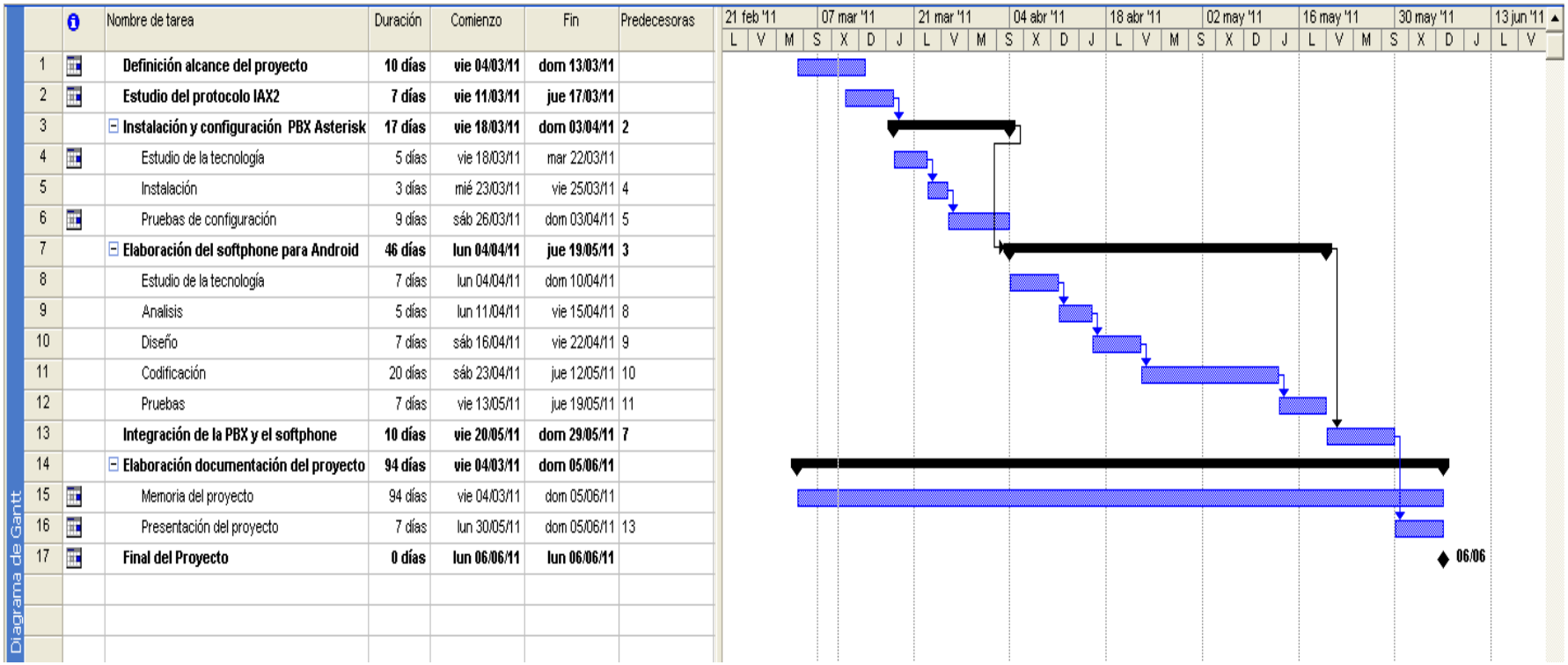
Nombre	Instalación y configuración PBX Asterisk
Descripción	Se divide en tres subtareas: Estudio de la tecnología ➔ Fijar los aspectos indispensables sobre la centralita Instalación ➔ Puesta en marcha de la centralita Pruebas de configuración ➔ Con el fin de familiarizarse con la administración del sistema
Duración	7 días
Resultado	Capítulo en la memoria de trabajo y PBX instalada

Nombre	Elaboración del softphone para Android
Descripción	Se divide en cinco subtareas: Estudio de la tecnología ➔ Fijar los aspectos indispensables para realizar la aplicación Análisis ➔ Se analizan las necesidades para determinar qué objetivos debe cubrir Diseño ➔ Se descompone y organiza el sistema en elementos que puedan elaborarse por separado Codificación ➔ Se implementa el programa Pruebas ➔ Se comprueba el correcto funcionamiento y que cumple con los requisitos
Duración	46 días
Resultado	Capítulo en la memoria de trabajo y softphone implementado

Nombre	Integración de la PBX y el softphone
Descripción	Realización de pruebas de integración entre la aplicación desarrollada y el PBX
Duración	10 días
Resultado	Capítulo en la memoria de trabajo y PBX instalada + softphone integrados

Nombre	Elaboración documentación del proyecto
Descripción	Esta actividad constará de dos subtareas: La elaboración de la memoria del proyecto ➔ Se prolongará a lo largo de toda la duración del proyecto La elaboración de la presentación del proyecto ➔ Con una duración especifica de siete días una vez haya finalizado la fase de integración
Duración	94 días
Resultado	Memoria de trabajo y presentación del proyecto

4.2 CALENDARIO



	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**Definición alcance del proyecto**	**10 días**	**vie 04/03/11**	**dom 13/03/11**	
2	**Estudio del protocolo IAX2**	**7 días**	**vie 11/03/11**	**jue 17/03/11**	
3	**Instalación y configuración PBX Asterisk**	**17 días**	**vie 18/03/11**	**dom 03/04/11**	**2**
4	Estudio de la tecnología	5 días	vie 18/03/11	mar 22/03/11	
5	Instalación	3 días	mié 23/03/11	vie 25/03/11	4
6	Pruebas de configuración	9 días	sáb 26/03/11	dom 03/04/11	5
7	**Elaboración del softphone para Android**	**46 días**	**lun 04/04/11**	**jue 19/05/11**	**3**
8	Estudio de la tecnología	7 días	lun 04/04/11	dom 10/04/11	
9	Analisis	5 días	lun 11/04/11	vie 15/04/11	8
10	Diseño	7 días	sáb 16/04/11	vie 22/04/11	9
11	Codificación	20 días	sáb 23/04/11	jue 12/05/11	10
12	Pruebas	7 días	vie 13/05/11	jue 19/05/11	11
13	**Integración de la PBX y el softphone**	**10 días**	**vie 20/05/11**	**dom 29/05/11**	**7**
14	**Elaboración documentación del proyecto**	**94 días**	**vie 04/03/11**	**dom 05/06/11**	
15	Memoria del proyecto	94 días	vie 04/03/11	dom 05/06/11	
16	Presentación del proyecto	7 días	lun 30/05/11	dom 05/06/11	13
17	**Final del Proyecto**	**0 días**	**lun 06/06/11**	**lun 06/06/11**	

Ilustración 1 Calendario

5 EL PROTOCOLO IAX2

El protocolo IAX2 (Inter-Asterisk eXchange protocol) es un protocolo nativo de las PBX Asterisk desarrollado por la comunidad open source cuya función es la de proporcionar capacidad para crear, modificar y terminar sesiones multimedia a través de redes IP. Su creador Mark Spencer, el mismo que el de la PBX Asterisk, lo diseñó con el propósito en mente de posibilitar conexiones VoIP entre servidores. Debido a los inconvenientes que han ido surgiendo con la utilización de otros protocolos en las conexiones VoIP entre servidores y clientes se empezó a usar también profusamente en este ámbito. Las características más destacadas de IAX2 son:

- Comúnmente se suele hablar del protocolo IAX refiriéndose a su segunda versión la IAX2 ya que la original ha quedado obsoleta.
- Posee una gran flexibilidad que le permite ser usado para el streaming de cualquier tipo de media incluyendo video, aunque su principal función y para la que fue diseñado es su uso en conexiones de VoIP.
- IAX no es exclusivo para su uso en PBX Asterix. Puede ser utilizado independientemente de Asterisk. Además Asterisk puede trabajar con otros protocolos con SIP, H.323
- Permite el uso de cualquier códec para transmisión de voz. La negociación del códec se realiza al principio del establecimiento de la llamada y es posible cambiarlo durante la misma.
- No necesita un nuevo protocolo para el intercambio de los streams de audio, voz o el tipo de media para que se emplee. Es capaz de hacerlo por sí mismo.
- IAX define mensajes de tipo "confiable" y "no confiable". Los "no confiable" sirven para transportar el audio y en caso de pérdida no son retransmitidos. Tampoco necesitan el envío de ACK informando de su correcta recepción. Los mensajes confiables llevan información de control y si necesitan el envío de ACKs informando de su correcta recepción ya que si no son retransmitidos.
- Además IAX emplea otro tipo de mensajes para monitorear el estado de la red, que pueden ser intercambiados durante una llamada o sin que ninguna se esté produciendo.

- Es un protocolo binario que además está diseñado para minimizar la cantidad de información que se añade a cada paquete de voz lo que minimiza el ancho de banda necesario para su uso y lo hace más robusto ante ataques por desbordamiento del buffer.
- Proporciona soporte para transmisión de planes de marcación.
- Es muy adecuado para evitar los problemas derivados de NAT y de la existencia de firewalls ya que tanto la información de señalización como los datos viajan conjuntamente a través de datagramas UDP empleando una sola conexión establecida por defecto en el puerto 4569, por lo que permite "esconder" de manera más sencilla que otros protocolos el tráfico VoIP y así permitir su uso en países donde los ISP están filtrando este tipo de tráfico.
- Más fácil implementar seguridad añadida ya que emplea una sola conexión.
- La adición de nuevas características al protocolo es poco flexible ya que carece de un mecanismo de extensión genérica.
- El uso de un solo puerto provoca que pueda ser susceptible a ataques por denegación del servicio.
- IAX2 emplea un mecanismo de negociación del códec empleado punto a punto, lo que limita su extensibilidad ya que cada nodo por el que circule una llamada debe soportar hasta cierto nivel cada códec usado. Además la definición de los códec empleados se realiza en una máscara interna de 32 bits por lo que los codecs deben de estar definidos en el protocolo y el número que se podría emplear simultáneamente estaría limitado por la implementación del mismo.
- IAX2 proporciona autenticación basada en infraestructura de clave pública.
- IAX2 soporta trunking de manera que múltiples llamadas comparten la misma línea de comunicación.
- Incluye la posibilidad de usar un buffer para el jitter permitiendo mejorar notablemente la calidad de audio.

5.1 COMPARACIÓN ENTRE SIP Y IAX2

SIP (Session Initiation Protocol) es el protocolo para señalización VoIP más extendido. Sus diferencias con IAX2 son notables y dependiendo del entorno y el propósito que tengamos puede ser recomendable usar uno u otro. Por ejemplo, una de estas diferencias es que IAX2

emplea una codificación binaria en vez de una basada en ASCII lo que la hace más eficiente, más robusta y segura ante ataques por desbordamiento de buffer. Además el tamaño de los paquetes que se emplean para la señalización es significantemente más pequeño.

Otra diferencia notable entre estos dos protocolos radica en que SIP no necesita necesariamente que los mensajes de audio pasen a través del servidor sino que estos pueden ir directamente de un extremo a otro mientras que en IAX2 tanto la señalización como el audio deben pasar siempre a través del servidor, con lo que esto aumenta notablemente el ancho de banda que consume el servidor cuando hay llamadas simultaneas.

IAX2 proporciona un mecanismo por el cual en el caso de que una llamada termine de manera abrupta debido a que el emisor o el receptor hayan desaparecido repentinamente el tiempo necesario para cerrar la comunicación es mínimo. SIP, en cambio, carece de tal mecanismo.

Como ya se comentó anteriormente SIP es más proclive a problemas con NAT y firewalls ya que emplea tres conexiones, una para la señalización y dos para audio (una en cada sentido) mientras que IAX2 solo necesita una. Aunque esto se convierte en un inconveniente en las comunicaciones entre servidores a través de un IAX trunk ya que la mayoría de las pilas TCP/IP tiene problemas para manejar tantas comunicaciones simultaneas en un mismo puerto UDP limitando su escalabilidad. Esto no acurre en SIP ya que no está limitado el número de puertos UDP que puede emplear.

Estas y otras características vistas anteriormente se resumen en la siguiente tabla:

SIP	IAX2
Tres puertos: dos para audio (una en cada sentido) y una para señalización	Un solo puerto para audio y señalización
Codificación ASCI	Codificación binaria
Mayor tamaño paquetes señalización	Menor tamaño paquetes señalización
Mayor extensibilidad	Menor extensibilidad
Número de códecs simultáneos no limitado	Número de códecs simultáneos limitados
Menos proclive a ataques denegación del servicio	Más proclive a ataques denegación del servicio

Compatible con gran número de PBX	Compatible casi exclusivamente con PBX Asterisk
Compatible con mayor número de dispositivos	Compatible con menor número de dispositivos
Funciona peor con NAT y Firewalls	Funciona mejor con NAT y Firewalls
Mayor escalabilidad	Menor escalabilidad
Consume mayor ancho de banda	Consume menos ancho de banda
No tiene mensajes exclusivos para monitorear el estado de la red	Si tiene mensajes exclusivos para monitorear el estado de la red

5.2 TRAMAS IAX2

IAX2 diferencia entre tres distintos tipos de trama: mini, full y meta. Cada una de ellas con diferentes características y propósitos.

Cualquiera de los tres tipos puede ser encriptado. Para ello se recomienda el cifrado de los mensajes con AES, aunque este aspecto queda fuera del ámbito de este proyecto.

La siguiente tabla resume los tres tipos:

	MINI	**FULL**	**META**
USO	Para enviar voz u otro tipo de media	Para enviar mensajes confiables con info de control. También pueden incorporar media	Se usa para enviar video o múltiples mini tramas con una sola cabecera IAX
TAMAÑO	4 bytes	12 bytes	6 video / 8 trunk
CARACTERISTICAS PRINCIPALES	No requiere ACK del receptor	Si requiere ACK del receptor y si no se recibe hay retransmisión	No requiere ACK del receptor

5.2.1 TRAMA FULL

La trama full o completa se emplea generalmente para enviar mensajes de señalización. En esta trama también puede viajar info tipo media (audio, video,…) aunque no es recomendable. Es necesario el envío de un ACK por parte del receptor ya que se implementa el reenvío.

Las tramas full tienen una cabecera de 12 bytes y su formato es el siguiente:

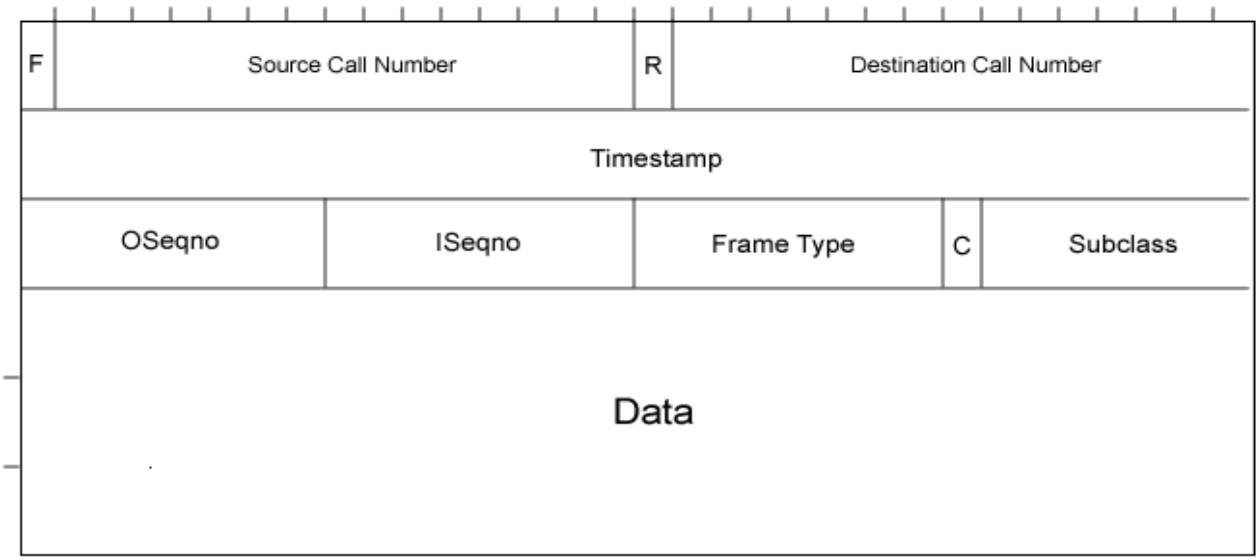


Ilustración 2 Trama Full

F: Este bit debe fijarse a 1 para indicar que se trata de una trama full (1 bit)

Source call number: El identificador de la llamada en el origen (15 bits)

R: indica si es una trama retransmitida (1 bit)

Destination Call Number: El identificador de la llamada en el nodo de destino (15 bits)

Time-stamp: Se utilizan para reordenar las tramas recibidas (32 bits)

OSeqno: Su propósito es identificar los streams de salida dentro de una llamada (8 bits)

Iseqno: Su propósito es identificar los streams recibidos dentro de una llamada (8 bits)

Frame Type: Indica el tipo de la trama (ver tabla x.x) (8 bits)

C: Indica el formato del campo SubClass (0 entero sin signo, 1 potencia de dos) (1 bit)

SubClass: Indica la subclase del mensaje enviado por la trama (ver tabla x.x) (7 bits)

Los tipos de trama full disponibles y sus sublclases son:

NOMBRE	DESCRIPCIÓN	SUBCLASE	VALOR
DTMF	Transporta un digito de DTFM (Dual Tone Multiple Frequency)	DTMF digit (i.e. 0-9, A-D, *, Full Frame Type, #)	0x01
Voice	Se usa para transportar voz	Indica el formato de audio. Por ej: G.711 [G711] o iLBC [ILBC]	0x02
Video	Para transportar video	Indica el formato de video. Por ej: H.263 [H263] o H.261 [H261]	0x03
Control	Transporta información sobre el estado de la llamada	Por ejemplo: 0x03: Ringing 0x05: Busy	0x04
Null		N/A	0x05
IAX2	Se usa para dar capacidad de gestión de la llamada	Por ejemplo: 0x01: NEW 0x03: PONG 0x04: ACK	0x06
Text	Para enviar un mensaje IAX	El valor de la subclase texto es 0	0x07
Image	Para enviar una imagen	Indica el formato de imagen. Por ej: JPEG, GIF …	0x08
HTML	Para enviar datos HTML	Ejemplo: 1: Sending URL 2: Data frame 8: End frame	0x09
Comfort Noise	Enviado para enviar 15confort noise	Indica el valor medido en dBov1	0x0A

5.2.2 TRAMA MINI

La longitud de la cabecera se encuentra limitada a 4 bytes y se usan exclusivamente para enviar voz una vez que la llamada ya ha sido establecida. Este tipo de mensajes no requiere de ACK y por lo tanto no hay reenvío en caso de pérdida. Se permite cambiar el códec utilizado en mitad de una comunicación pero para ello es necesario enviar una trama full.

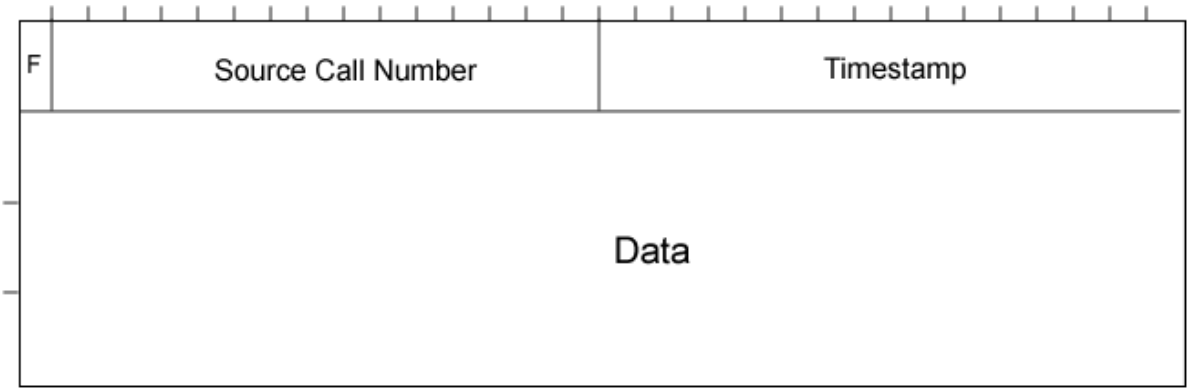


Ilustración 3 Trama Mini

F: Este bit debe fijarse a 0 para indicar que se trata de una trama mini (1 bit)

Source call number: El identificador de la llamada en el origen (15 bits)

Time-stamp: Se utilizan para reordenar las tramas recibidas y su tamaño es la mitad que el empleado en las tramas full (16 bits)

5.2.3 TRAMA META

Este tipo de trama, menos común, se puede emplear en dos casos:

➔ En el intercambio de streams de video utilizando para ello una cabecera optimizada.

➔ Para permitir que múltiples streams de diferente media sean incluidos en la misma trama con una sola cabecera y de esta manera optimizar el consumo de bando de ancha.

Estas tramas viajan marcadas con el campo F a 0 para indicar que no son una trama full.

Después llevan un campo llamado meta indicador con 16 ceros indicando que se trata de una trama meta.

Además incorporan, entre otros, un nuevo campo V que si lleva el valor 0 indica que no es una trama meta de video.

5.3 ELEMENTOS DE INFORMACIÓN (IE)

Una de las particularidades del protocolo IAX2 es el uso de elementos de información cuya función es la de contener información requerida para la gestión de las llamadas IAX. Los IE viajan en tramas de tipo Full y tienen la siguiente estructura:

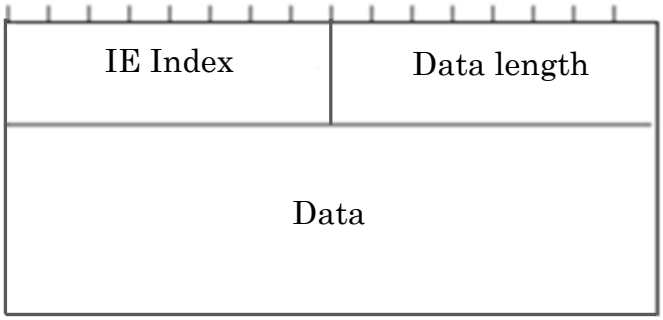


Ilustración 4 Elemento de información

IE Index: Almacena el identificador del IE que se está enviando.

Data Lenght: Especifica el tamaño de los datos.

Data: De tamaño variable contiene los datos codificados en UTF-8.

Los tipos de IE que se pueden enviar son:

NOMBRE	DESCRIPCIÓN
CALLED NUMBER	Se emplea para almacenar la uri del destino
CALLING NUMBER	Se emplea para almacenar la uri del origen
CALLING ANI	Se emplea para almacenar el ani
CALLING NAME	Almacena el nombre del origen de la llamada
CALLED CONTEXT	Información sobre el dial plan
USERNAME	Identidad del usuario
PASSWORD	La contraseña (puede ir cifrada)
CAPABILITY	Información relativa a las capacidades códec empleado
FORMAT	El códec empleado
LANGUAGE	El lenguaje empleado
VERSION	La versión de IAX soportada
ADSICPE	Indica si el dispositivo usado tiene Analog Display

	Services Interface
DNID	Informa sobre el identificador del número marcado
AUTHMETHODS	Métodos de autenticación soportados
CHALLENGE	Desafío RSA o MD5
MD5RESULT	Resultado del desafío MD5
DEVICETYPE	Tipo de dispositivo que solicita el registro
SERVICEIDENT	Se emplear para el transporte de identificadores
RDNIS	Indica el DNIS referido
PROVISIONING	Para propósitos de suministro
AESPROVISIONING	Para propósitos de suministro de AES
RSARESULT	Resultado del desafío RSA
APPARENT ADDR	La ip y el puerto empleado
REFRESH	Tiempo de refresco expresado en segundo
DPSTATUS	Indica el estado de un número llamada en un dial plan
CALLNO	Indica el número de la entidad que realiza la llamada
CAUSE	Info sobre un suceso
UNKNOWN	Informar del uso de un método IAX no soportado
MSGCOUNT	Info sobre el número de mensajes en el buzón de voz
AUTOANSWER	En caso de ser posible responder la llamada automáticamente
MUSICONHOLD	En caso de ser posible reproducir música mientras la llamada en espera
DATETIME	Indica la hora en que el mensaje es enviado
FIRMWAREVER	Indica la versión del firmware de un determinado dispositivo
ENCRYPTION	Info sobre los métodos de encriptación soportados
CODEC PREFS	Preferencia de códec a emplear
FWBLOCKDESC	Relativos al firmware del dispositivo
FWBLOCKDATA	Relativos al firmware del dispositivo

SAMPLINGRATE	Envía el sampling rate que se va a usar para enviar audio
RR JITTER	Información sobre el jitter de una llamada
RR LOSS	Porcentaje de tramas perdidas
RR PKTS	Número total de tramas recibidas para una llamada
RR DELAY	Información sobre el retraso en la recepción de tramas
RR DROPPED	Información sobre las tramas descartadas
RR OOO	Información sobre el número de tramas con datos corruptos

5.4 MENSAJES IAX2

El protocolo IAX define una serie de mensajes cuya función es la de inicializar, controlar o finalizar llamadas. La siguiente tabla especifica estos mensajes y resume sus características:

IAX MENSAJE	DESCRIPCIÓN	Ies RELACIONADOS
MWI	Indica que hay mensaje en espera	MSGCOUNT
DPREP	Se envía como respuesta a un DPREQ	CALLEDNUMBER, DPSTATUS, DP REFRESH
DIAL	Se emplea en el caso de que no se mantenga un dial plan	CALLEDNUMBER and/or CALLEDCONTEXT
DPREQ	Para determinar el número de llamadas mantenidas por un servidor	CALLEDNUMBER
FWDOWNL	Para requerir la descarga de un firmware	DEVTYPE and FWBLOCKDATA
FWDATA	El firmware a descargar como respuesta a un FWDOWNL	FWBLOCKDESC and FWBLOCKDATA
FLASH	Notifica el suceso de un evento	N/A
HOLD	Requiere la parada del envío de streams con audio	N/A

UNHOLD	Para volver a enviar streams de audio	N/A
QUELCH	Similar a HOLD con la diferencia que solo se puede usar en llamadas iniciadas mediante el mensaje NEW	N/A
UNQUELCH	Para volver a enviar streams de audio	N/A
TRANSFER	Para especificar el nuevo numero	CALLEDNUMBER and/or CALLEDCONTEXT
NEW	Para iniciar una llamada	VERSION, CALLEDNUMBER, AUTOANSWER, CODECPREF, CALLINGPRES, CALLINGNUMBER, CALLINGTON, CALLINGTNS, CALLINGNAME, ANI, LANGUAGE, DNID, CALLEDCONTEXT, USERNAME, RSARESULT, MD5RESULT, FORMAT, CAPABILITY, ADSICPE and/or DATETIME
ACCEPT	Para aceptar el establecimiento de una llamada	FORMAT
REJECT	Para rechazar un mensaje NEW,AUTHREP, DIAL o ACCEPT	CAUSE and/or CAUSECODE
HANGUP	Para colgar una llamada	CAUSE and/or CAUSECODE
AUTHREP	Se envía como respuesta a un	RSARESULT or

	AUTHREQ	MD5RESULT
AUTHREQ	Se envía como respuesta a un mensaje NEW si se requiere autenticación	USERNAME, CHALLENGE and/or AUTHMETHODS
PROCEEDING	Se emplea por un nodo IAX intermedio para informar al solicitante de una llamada que la solicitud ha sido enviada pero la respuesta no ha sido recibida aun	N/A
RINGING	Para informar que se está en proceso de aceptar o rechazar la llamada	N/A
ANSWER	Para indicar que se ha aceptado el establecimiento de una llamada y se empieza a enviar streams	N/A
PING	Realiza pruebas de conectividad entre dos peers	N/A
POKE	Similar a PING pero con la diferencia que se usa solo cuando no hay una llamada establecida entre las dos peers	N/A
PONG	Se envía como respuesta a un PING o POKE	RRJITTER, RRPKTS, RRDELAY and/ or RRDROPPED
LAGRQ	Para evaluar el lag	N/A
LAGRP	Se envía como respuesta a un LAGRQ	N/A
PROVISION	Para enviar información de provisioning	PROVISIONING and/or AESPROVISIONING
REGREL	Para borrar un registro	MD5RESULT, RSARESULT, CAUSE and/or CAUSECODE
REGREQ	Para registrarse	USERNAME,

		MD5RESULT, RSARESULT and/ or REFRESH
REGAUTH	Se envía como respuesta a un REGREQ o a REGREL	USERNAME, AUTH-METHODS and/or CHALLENGE
REGACK	Ack de un REGREQ	USERNAME, DATE-TIME, APPAREN-TADDR, MSGCOUNT, CALL-INGNUMBER CALL-INGNAME, FIRM-WAREVER and/or REFRESH
REGREJ	Para rechazar una solicitud de registro	CAUSE and/or CAUSECODE
TXREQ	Para solicitar un TRANSFER	APPARENTADDR, CALL-NUMBER and/or TRANDERID
VNAK	Solicita la retransmisión de los mensajes después a partir de uno especificado por medio de su identificador	N/A
ACK	Para informar de la recepción de un mensaje	N/A
INVAL	Para solicitar la destrucción del Call Context	N/A
TXACC	Se envía como respuesta a un TXCNT	TRANSFERID
TXCNT	Para verificar la conectividad	TRANSFERID
TXREADY	Una vez verificada la conectividad para	TRANSFERID

	informar de ella	
TXREJ	Durante una operación de transferencia para indicar que uno de los extremos no puede realizar la operación	N/A
TXREL	Para indicar que una operación de transferencia se ha realizado satisfactoriamente	CALLEDNUMBER
UNSUPPORT	Para indicar a un peer que no es soportado	N/A

5.5 OPERACIONES IAX2

A continuación se describen las operaciones más comunes que se llevan a cabo en el protocolo IAX2 y los mensajes implicados:

➔ Operaciones de suministro y descarga de firmware para dispositivos. Permite el envío de información y la descarga de firmware por parte de ciertos dispositivos. Un dialogo de descarga de firmware sería similar al siguiente:

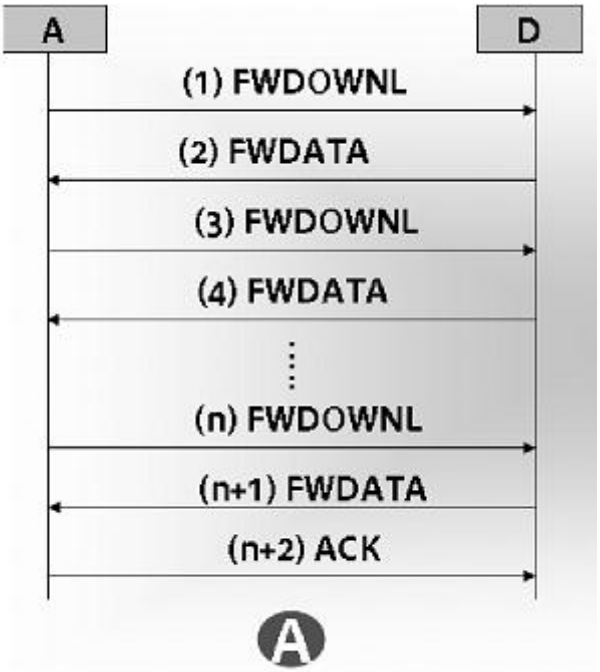


Ilustración 5 Operaciones de suministro y descarga de firmware

➔ Registro: Esta operación es opcional para aquellas peer con direcciones ip estáticas. Para aquellas con direcciones dinámicas es obligatorio registrarse en un servidor para permitir que otras peer puedan acceder a ellas. Por ejemplo una operación de registro satisfactoria sin autenticación implica el siguiente dialogo:

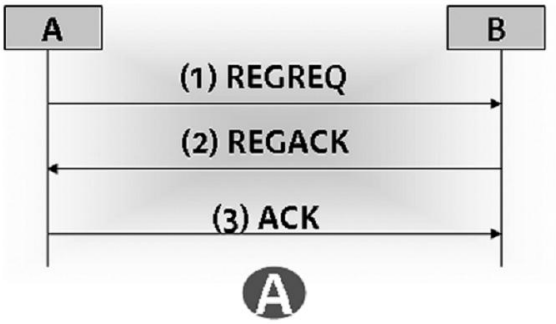


Ilustración 6 Operación de registro

➔ Establecimiento de llamada: Esta es la función principal del protocolo. Un establecimiento exitoso con autenticación llevado a cabo directamente entre dos nodos incluiría los siguientes mensajes:

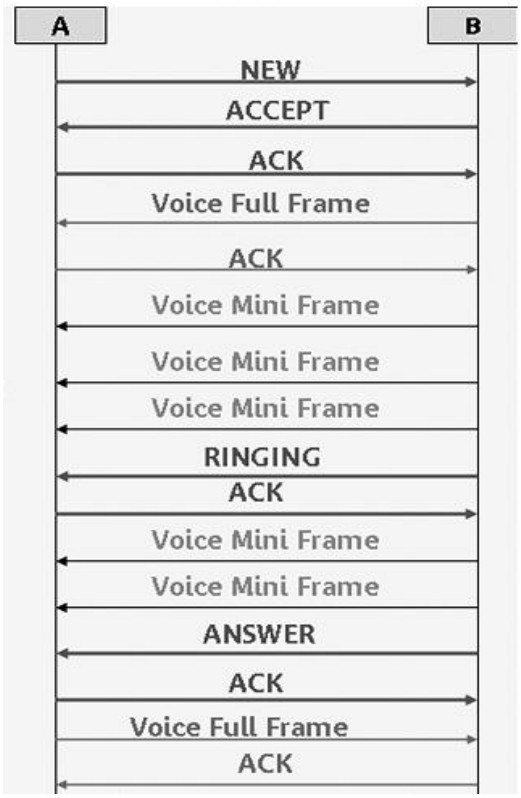


Ilustración 7 Establecimiento de llamada

Por su parte un intento fallido de establecimiento de llamada implicaría:

Ilustración 8 Establecimiento fallido de llamada

➔ Fin de llamada: Mensajes necesarios para finalizar una llamada:

Ilustración 9 Fin de llamada

➔Supervisión de llamada: a diferencia de SIP permite reconocer si una determina peer está conectado o no.

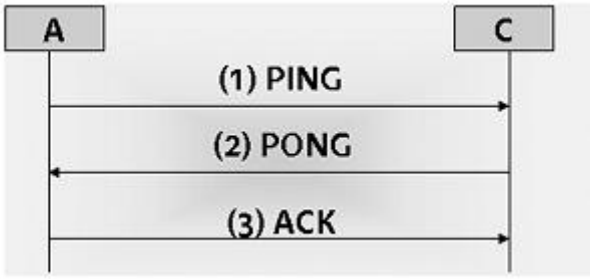


Ilustración 10 Supervisión de llamada

➔Optimización de llamada: Su función principal es la de permitir a un nodo intermedio desaparecer de la comunicación entre emisor y receptor de una llamada permitiéndoles conectarse directamente. En el siguiente ejemplo B desaparece de la comunicación entre A y C.

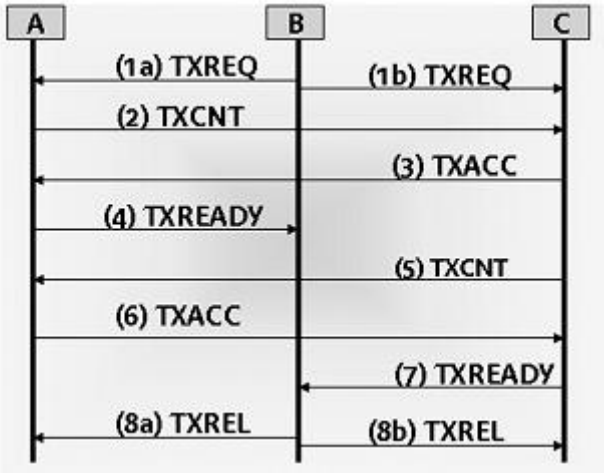


Ilustración 11 Optimización de llamada

6 PBX ASTERISK

En este apartado se trataran las características generales de la PBX Asterisk para posteriormente tratar aspectos relacionados con su instalación, configuración así como la realización de diversas pruebas cuyo propósito es el de familiarizarse con su funcionamiento.

6.1 CARACTERÍSTICAS GENERALES

Asterisk es un software que implementa una centralita telefónica (private branch exchange – PBX). Fue creada por Mark Spencer, el mismo del protocolo IAX, en 1999 y liberada bajo una doble licencia: GPL para los componentes básicos, y privativa para algunos componentes adicionales.

En un principio solo estaba disponible para sistemas GNU-Linux aunque actualmente se pueden encontrar implementaciones para otros S.O. como MacOS, FreeBSD, OpenBSD, etc.

Asterisk soporta gran variedad de protocolos de comunicaciones VoIP y es compatible con la mayor parte de fabricantes del hardware empleado para telefonía IP (teléfonos, adaptadores, routers,…).

Otros aspectos destacables son:

- Permite crear respuestas interactivas personalizadas
- Es posible la distribución automática de llamadas
- Permite la creación de conferencias
- Permite la conexión con la red pública conmutada

- Autenticación
- Respuesta automatizada
- Listas negras
- Transferencia no supervisada
- Registros de llamada detallados
- Desvío de llamada si la extensión está ocupada
- Desvío de llamada si la extensión no responde
- Monitorización de llamadas
- Grabación de llamadas
- Transferencia de llamadas
- Llamada en espera
- Identificación del llamante
- Integración con Base de Datos
- Marcación por nombre
- Recepción y transmisión de Fax
- Música en espera
- Conversión de protocolos
- Mensajería SMS
- Trunking
- Pasarelas VoIP
- Envío de mensajes del buzón al correo electrónico

6.2 INSTALACIÓN Y CONFIGURACIÓN

La instalación de una PBX Asterisk varía dependiendo de la distribución concreta en la que se vaya a realizar. Actualmente es posible utilizar distribuciones específicas para su uso donde Asterisk viene pre-instalado. Algunas de estas distribuciones son: Elastix o AsteriskNow! De cualquier modo se puede llevar a cabo en la distro que prefiramos.
En pro de facilitar la realización de este proyecto se ha seleccionado una distro con Asterisk pre-instalado. En concreto se utilizará AsteriskNow! corriendo en una máquina virtual. Para ello primeramente es necesario realizar la descarga de la imagen ISO:
http://www.asterisk.org/downloads/asterisknow/i386/asterisknow32.iso

Una vez descargada, valiéndonos de un software de virtualización tal como VMWare o VirtualBox se realiza su instalación. Este proceso es trivial y no requiere de grandes conocimientos. Una vez finalizada ya es posible pasar a su configuración.

Para configurar Asterisk debemos acceder a un navegador e introducir la dirección IP asignada a la máquina donde está corriendo AsteriskNow! De esta forma accederemos a una interfaz gráfica muy sencilla donde podremos realizar la configuración.

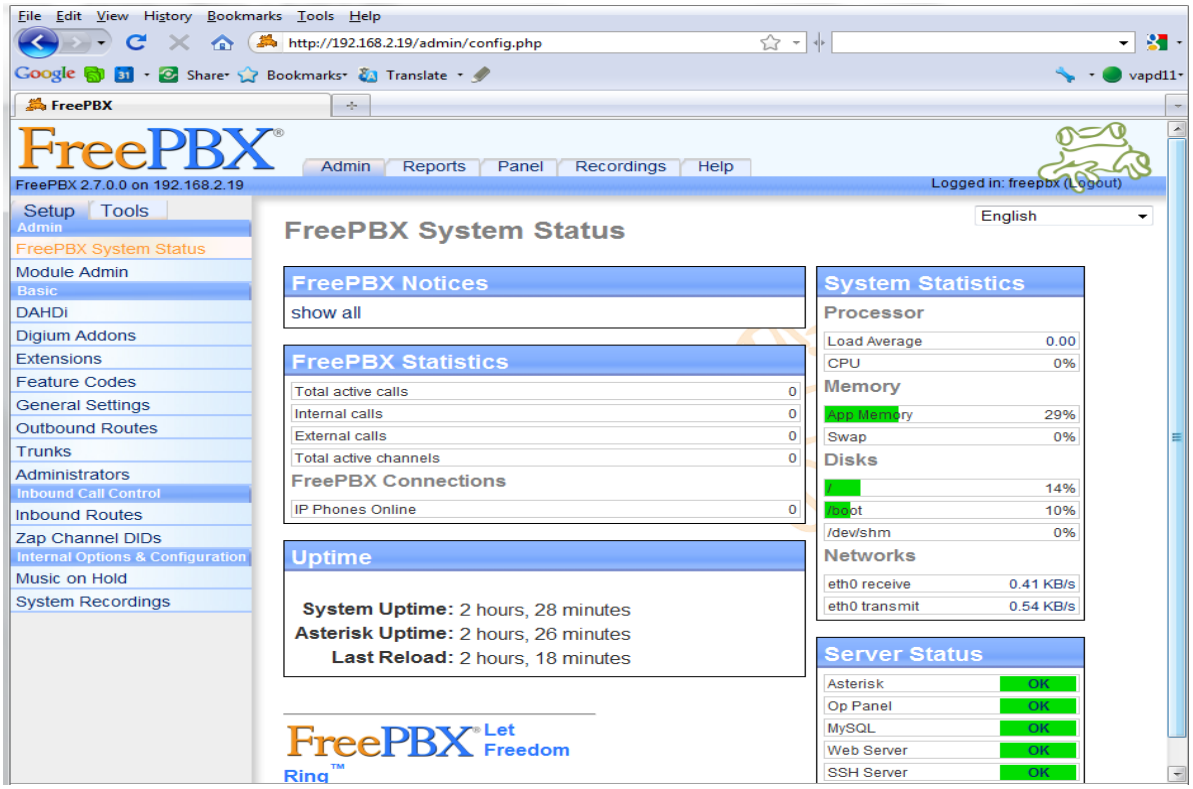


Ilustración 12 Administración FreePBX

Valiéndonos de esta interfaz podremos añadir extensiones IAX2 que son únicamente las que necesitaremos para este proyecto en concreto. Simplemente con indicar un número para la extensión, un nombre y una contraseña es suficiente.

6.3 PRUEBAS DE FUNCIONAMIENTO

Valiéndonos de un softphone recomendado por el proyecto Asterisk (http://www.zoiper.com/) y habiendo creado un par de extensiones IAX2 en nuestra PBX vamos a comprobar la correcta conectividad entre las extensiones dentro y fuera de una red local.

Dentro de la misma red local el proceso de configuración de los softphones es idéntico y consiste en configurar una cuenta IAX accediendo a las opciones del programa. Para ello se debe indicar la ip en la que se encuentra la PBX y proporcionar los datos para la extensión concreta.

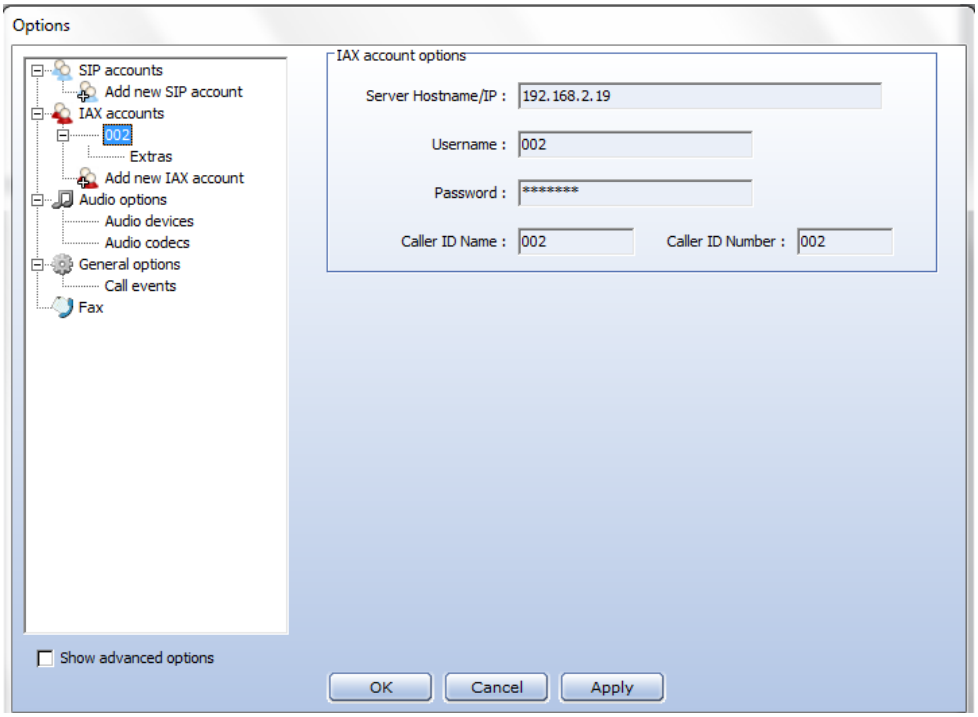


Ilustración 13 Zoiper

Una vez hecho esto se puede proceder al registro de la extensión el cual, si todo transcurre satisfactoriamente, intercambiara los siguientes mensajes entre el nodo y el servidor:

Source	Destination	Protocol	Info
192.168.2.4	192.168.2.19	IAX2	IAX, source call# 66, timestamp 1ms REGREQ
192.168.2.19	192.168.2.4	IAX2	IAX, source call# 1378, timestamp 19ms REGAUTH
192.168.2.4	192.168.2.19	IAX2	IAX, source call# 66, timestamp 4ms REGREQ
192.168.2.19	192.168.2.4	IAX2	IAX, source call# 3946, timestamp 3ms POKE
192.168.2.19	192.168.2.4	IAX2	IAX, source call# 1378, timestamp 43ms REGACK
192.168.2.4	192.168.2.19	IAX2	IAX, source call# 66, timestamp 43ms ACK
192.168.2.4	192.168.2.19	IAX2	IAX, source call# 1, timestamp 3ms ACK
192.168.2.4	192.168.2.19	IAX2	IAX, source call# 1, timestamp 3ms PONG
192.168.2.19	192.168.2.4	IAX2	IAX, source call# 3946, timestamp 3ms ACK

Ilustración 14 Intercambio de registro

En el que se puede apreciar un esquema similar al teórico:

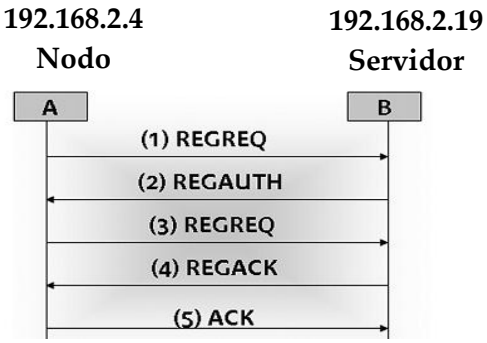


Ilustración 15 Esquema de registro

Después de haber registrado otra extensión podemos proceder a realizar una llamada y apreciar los paquetes que se intercambia una de ellas con el servidor comparándolo con el dialogo teórico:

Source	Destination	Protocol	Info
192.168.2.19	192.168.2.4	IAX2	IAX, source call# 14987, timestamp 7ms NEW
192.168.2.4	192.168.2.19	IAX2	IAX, source call# 165, timestamp 7ms ACK
192.168.2.4	192.168.2.19	IAX2	IAX, source call# 165, timestamp 2ms ACCEPT
192.168.2.4	192.168.2.19	IAX2	Control, source call# 165, timestamp 3ms RINGING
192.168.2.19	192.168.2.4	IAX2	IAX, source call# 14987, timestamp 2ms ACK
192.168.2.19	192.168.2.4	IAX2	IAX, source call# 14987, timestamp 3ms ACK
192.168.2.4	192.168.2.19	IAX2	IAX, source call# 164, timestamp 9ms ACK
192.168.2.4	192.168.2.19	IAX2	Control, source call# 165, timestamp 1601ms ANSWER
192.168.2.19	192.168.2.4	IAX2	IAX, source call# 14987, timestamp 1601ms ACK
192.168.2.4	192.168.2.19	IAX2	IAX, source call# 165, timestamp 1685ms ACK
192.168.2.4	192.168.2.19	IAX2	Voice, source call# 165, timestamp 1665ms, Raw mu-law data (G.711)
192.168.2.19	192.168.2.4	IAX2	IAX, source call# 14987, timestamp 1665ms ACK
192.168.2.19	192.168.2.4	IAX2	Voice, source call# 14987, timestamp 1740ms, Raw mu-law data (G.711)
192.168.2.4	192.168.2.19	IAX2	IAX, source call# 165, timestamp 1740ms ACK
192.168.2.4	192.168.2.19	IAX2	Mini packet, source call# 165, timestamp 1685ms, Raw mu-law data (G.711)
192.168.2.4	192.168.2.19	IAX2	Mini packet, source call# 165, timestamp 1705ms, Raw mu-law data (G.711)
192.168.2.19	192.168.2.4	IAX2	Mini packet, source call# 14987, timestamp 1760ms, Raw mu-law data (G.711)
192.168.2.4	192.168.2.19	IAX2	Mini packet, source call# 165, timestamp 1725ms, Raw mu-law data (G.711)
192.168.2.19	192.168.2.4	IAX2	Mini packet, source call# 14987, timestamp 1780ms, Raw mu-law data (G.711)
192.168.2.4	192.168.2.19	IAX2	Mini packet, source call# 165, timestamp 1745ms, Raw mu-law data (G.711)

Ilustración 16 Intercambio de llamada

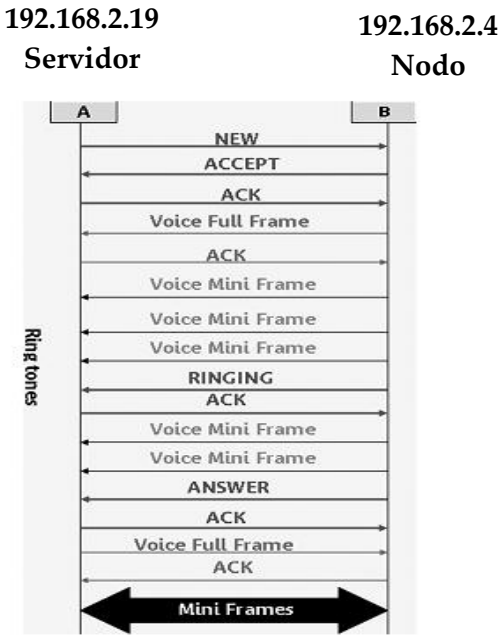


Ilustración 17 Esquema de llamada

El registro de una extensión desde fuera de la red local precisa de la apertura del puerto UDP 4569 y, en caso de que se esté empleando NAT, la redirección del mismo a la máquina concreta en que reside la PBX dentro de nuestra red local. Una vez hecho esto y conociendo nuestra dirección IP pública el proceso de configuración del softphone es prácticamente idéntico, simplemente varía en la dirección del servidor que en este caso se corresponderá con la IP pública que tenemos asignada.

En la siguiente figura podemos ver el intercambio de paquetes entre el servidor en la IP pública 84.78.60.188 y el equipo dentro de nuestra red local 192.168.2.15 en el proceso de registro:

Source	Destination	Protocol	Info
192.168.2.15	84.78.60.188	IAX2	IAX, source call# 2, timestamp 1ms REGREQ
84.78.60.188	192.168.2.15	IAX2	IAX, source call# 1, timestamp 1ms unknown (0x28)
192.168.2.15	84.78.60.188	IAX2	IAX, source call# 2, timestamp 16ms REGREQ
84.78.60.188	192.168.2.15	IAX2	IAX, source call# 5693, timestamp 14ms REGAUTH
192.168.2.15	84.78.60.188	IAX2	IAX, source call# 2, timestamp 141ms REGREQ
84.78.60.188	192.168.2.15	IAX2	IAX, source call# 1845, timestamp 12ms POKE
84.78.60.188	192.168.2.15	IAX2	IAX, source call# 5693, timestamp 152ms REGACK
192.168.2.15	84.78.60.188	IAX2	IAX, source call# 2, timestamp 152ms ACK
192.168.2.15	84.78.60.188	IAX2	IAX, source call# 1, timestamp 12ms ACK
192.168.2.15	84.78.60.188	IAX2	IAX, source call# 1, timestamp 12ms PONG
84.78.60.188	192.168.2.15	IAX2	IAX, source call# 1845, timestamp 12ms ACK

Ilustración 18 Intercambio de registro (red externa)

En la realización de una llamada entre una extensión localizada dentro de nuestra red local y la externa que acabamos de registrar el intercambio de paquetes es el siguiente:

Source	Destination	Protocol	Info
84.78.60.188	192.168.2.15	IAX2	IAX, source call# 4637, timestamp 3ms NEW
192.168.2.15	84.78.60.188	IAX2	IAX, source call# 16, timestamp 3ms ACK
192.168.2.15	84.78.60.188	IAX2	IAX, source call# 16, timestamp 2ms ACCEPT
192.168.2.15	84.78.60.188	IAX2	Control, source call# 16, timestamp 2ms RINGING
84.78.60.188	192.168.2.15	IAX2	IAX, source call# 4637, timestamp 2ms ACK
84.78.60.188	192.168.2.15	IAX2	IAX, source call# 4637, timestamp 2ms ACK
192.168.2.15	84.78.60.188	IAX2	IAX, source call# 1, timestamp 14ms ACK
192.168.2.15	84.78.60.188	IAX2	Control, source call# 16, timestamp 2518ms ANSWER
84.78.60.188	192.168.2.15	IAX2	IAX, source call# 4637, timestamp 2518ms ACK
192.168.2.15	84.78.60.188	IAX2	Voice, source call# 16, timestamp 2612ms, Raw mu-law data (G.711)
192.168.2.15	84.78.60.188	IAX2	Mini packet, source call# 16, timestamp 2632ms, Raw mu-law data (G.711)
84.78.60.188	192.168.2.15	IAX2	Voice, source call# 4637, timestamp 3000ms, Raw mu-law data (G.711)
84.78.60.188	192.168.2.15	IAX2	IAX, source call# 4637, timestamp 2612ms ACK
84.78.60.188	192.168.2.15	IAX2	Mini packet, source call# 4637, timestamp 3020ms, Raw mu-law data (G.711)
84.78.60.188	192.168.2.15	IAX2	Mini packet, source call# 4637, timestamp 3040ms, Raw mu-law data (G.711)
84.78.60.188	192.168.2.15	IAX2	Mini packet, source call# 4637, timestamp 3060ms, Raw mu-law data (G.711)
84.78.60.188	192.168.2.15	IAX2	Mini packet, source call# 4637, timestamp 3080ms, Raw mu-law data (G.711)
84.78.60.188	192.168.2.15	IAX2	Mini packet, source call# 4637, timestamp 3100ms, Raw mu-law data (G.711)
84.78.60.188	192.168.2.15	IAX2	Mini packet, source call# 4637, timestamp 3120ms, Raw mu-law data (G.711)
84.78.60.188	192.168.2.15	IAX2	Mini packet, source call# 4637, timestamp 3140ms, Raw mu-law data (G.711)
84.78.60.188	192.168.2.15	IAX2	Mini packet, source call# 4637, timestamp 3160ms, Raw mu-law data (G.711)

Ilustración 19 Intercambio de llamada (red externa

7 ANDROID

Android es un sistema operativo basado en Linux para dispositivos móviles. Actualmente pertenece a Google y es el principal producto de la Open Handset Alliance, un conglomerado de fabricantes y desarrolladores de hardware, software y operadores de servicio.

La arquitectura interna de la plataforma Android, está formada por cuatro componentes fundamentales:

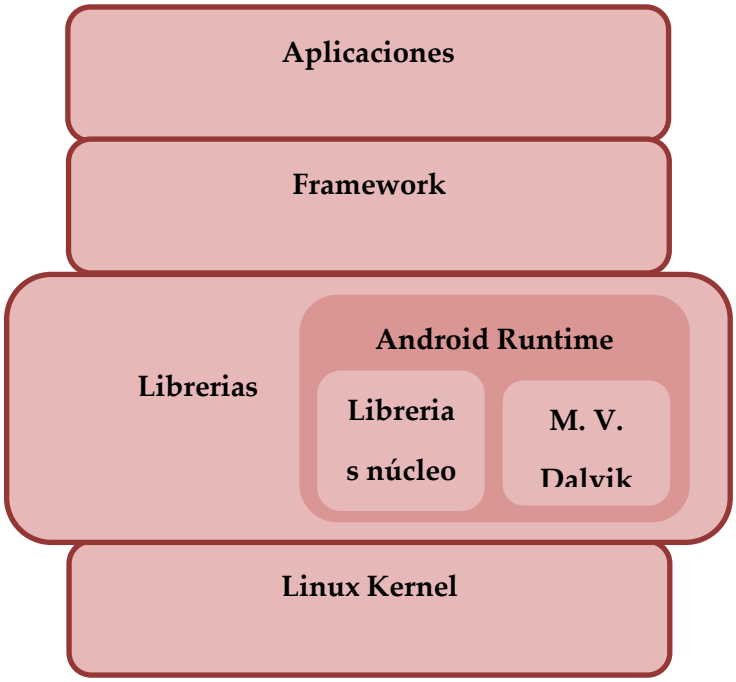


Ilustración 20 Componentes fundamentales Android

Aplicaciones ➔ Android cuenta con una amplia comunidad de desarrolladores que han realizado más de 200.000 aplicaciones. Estos programas están escritos en el lenguaje de programación Java.

Framework ➔ Los desarrolladores de aplicaciones Android tienen acceso a los mismos APIs del framework usados por las aplicaciones base.

Librerías ➔ Android incluye un conjunto de librerías de C/C++ usadas por varios componentes del sistema. Entre ellas destacan: System C library, bibliotecas de medios, bibliotecas de gráficos, 3D, etc

Android runtime ➔ Cada aplicación Android corre su propio proceso, con su propia instancia de la máquina virtual Dalvik. Además se incluye un set de bibliotecas base que

proporcionan la mayor parte de las funciones disponibles en las bibliotecas base del lenguaje Java.
Linux Kernel ➔ Android depende del núcleo de Linux para los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de controladores.

La versión de Android que se empleará en este proyecto será la 2.3 (en concreto la 2.3.3) conocida con el nombre en clave de GingerBread. Sus características más destacables son:

- Actualización del diseño de la interfaz de usuario
- Soporte para pantallas extra grandes y resoluciones WXGA y mayores
- Soporte nativo para telefonía VoIP SIP
- Soporte para reproducción de videos WebM/VP8 y decodificación de audio AAC
- Nuevos efectos de audio como reverberación, ecualización, virtualización de los auriculares y refuerzo de graves
- Soporte para Near Field Communication
- Funcionalidades de cortar, copiar y pegar disponibles a lo largo del sistema
- Teclado multi-táctil rediseñado
- Soporte mejorado para desarrollo de código nativo
- Mejoras en la entrada de datos, audio y gráficos para desarrolladores de juegos
- Recolección de elementos concurrentes para un mayor rendimiento
- Soporte nativo para más sensores (como giroscopios y barómetros)
- Un administrador de descargas para descargar archivos grandes
- Administración de la energía mejorada y control de aplicaciones mediante la administrador de tareas
- Soporte nativo para múltiples cámaras
- Cambio de sistema de archivos de YAFFS a ext4

Para comenzar a desarrollar aplicaciones en Android primeramente es necesario instalar el software necesario en el S.O. que vayamos a emplear. Los pasos que se deben realizar para una correcta instalación son:

- Instalar el Oracle Java SE SDK (JDK).
- Instalar el SDK de Android
- Instalar Eclipse (opcional)
- Instalar el plugin ADT para Eclipse (opcional)
- Configurar el emulador

8 DESARROLLO DE LA APLICACIÓN

Dividiremos el desarrollo de la aplicación en cuatro partes. En la primera llevaremos a cabo diversas pruebas con la plataforma y el soporte SIP. En la segunda nos centraremos en la interfaz gráfica del programa, en la tercera en la búsqueda y adaptación de alguna librería que de soporte a IAX2 y en la cuarta y última en conectar la librería con la interfaz creada en el segundo punto.

8.1 PRUEBAS CON EL SOPORTE SIP

Dentro de los ejemplos que se proporcionan con el SDK encontramos, para la versión de Android 2.3.3, la aplicación "SIPDemo". Este ejemplo es lo suficientemente sencillo y claro, por lo que nos permite apreciar las principales características del soporte SIP proporcionado en el API de Android GingerBread. La ejecución y observación del código de esta aplicación nos va a servir de gran ayuda a la hora de desarrollar nuestro softphone.

Antes de ejecutar la aplicación en nuestro emulador debemos configurarlo para permitir emplear la API que da soporte SIP, el empleo de VoIP así como suprimir la restricción por la cual SIP solo se puede emplear a través de la conexión WIFI (no de 3G). Estas tres condiciones no se cumplen en un principio por lo que debemos realizar los siguientes cambios:

1.- Permitir la escritura en la partición del sistema

2.- Copiar android.software.sip.xml, android.software.sip.voip.xml en /system/etc/permission

3.- Deshabilitar la opción config_sip_wifi_only en el fichero config.xml

Una vez hechos estos pasos escribimos un sencillo programa para comprobar que los cambios han surtido efecto. La clase SipManager nos proporciona tres métodos para este menester: isApiSupported, isSipWifiOnly, isVoipSupported que nos devuelven un booleano. Nuestro programa consistirá en mostrar por pantalla el resultado de la llamada a estos

métodos. Si todo ha ido correcto debemos ver en la pantalla del dispositivo algo parecido a lo siguiente:

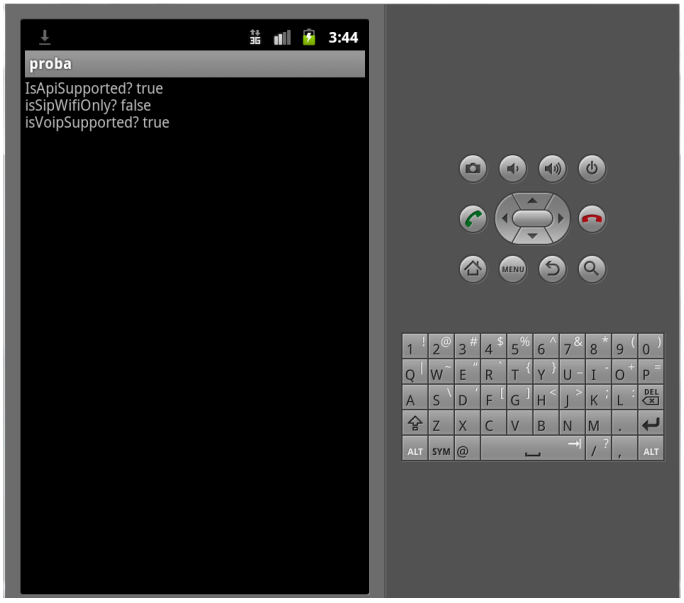


Ilustración 21 Comprobación soporte SIP en el emulador

Una vez ya hemos comprobado que disponemos de soporte para SIP ya podemos instalar y ejecutar SIPDemo. Lo primero que nos pedirá será que definamos las propiedades de nuestra conexión: La extensión, la contraseña y la dirección del servidor al que nos conectaremos. Una vez hecho si la aplicación logra registrarse veremos un mensaje "Ready" informándonos de ello:

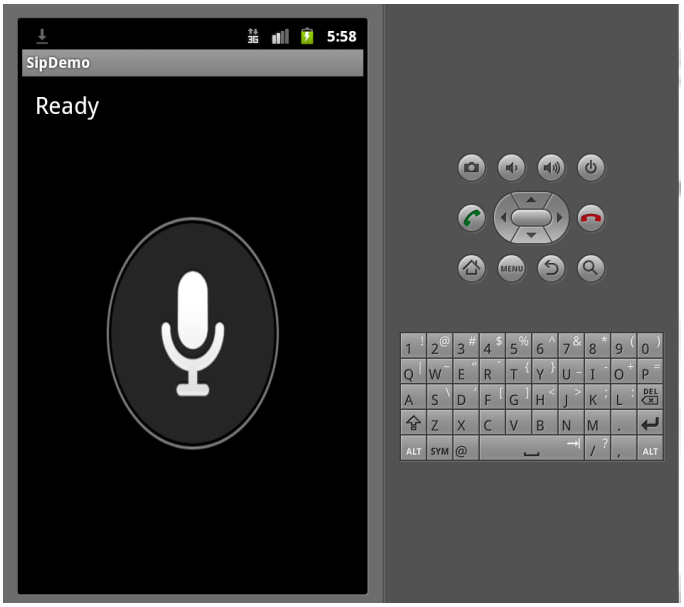


Ilustración 22 SipDemo registrada

Desde otra extensión realizamos una llamada a la extensión del emulador y comprobamos que el audio se reproduce y captura adecuadamente. El programa no permite realizar llamadas solo recibirlas. En el margen superior izquierdo se muestra el origen de la llamada:

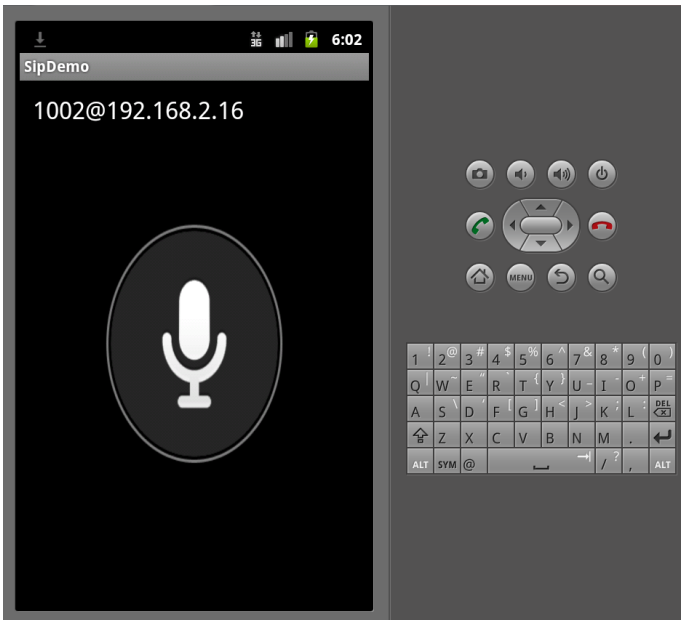


Ilustración 23 SipDemo en llamada

8.2 INTERFAZ GRÁFICA DE LA APLICACIÓN

En esta parte nos centramos en la elaboración de la interfaz gráfica de usuario para el softphone. La idea es la de desarrollar una GUI sencilla y a la vez manejable que nos permita desarrollarla rápidamente y así poder dedicar más tiempo a la búsqueda de librerías Java que proporcionen soporte IAX2. En el desarrollo hemos fijado nuestro objetivo en no separar la configuración del teléfono en sí, en vez de implementar layouts diferentes se encuentras implementados en el mismo gracias al uso de un tabulador que los alterna en la pantalla. Así mismo el dialpad es diferente al convencional pero más adaptado a la realidad de las extensiones que emplearemos donde no se emplean ni asteriscos ni almohadillas.
Partiremos de una imagen inicial donde se mostrará un logo de la aplicación:

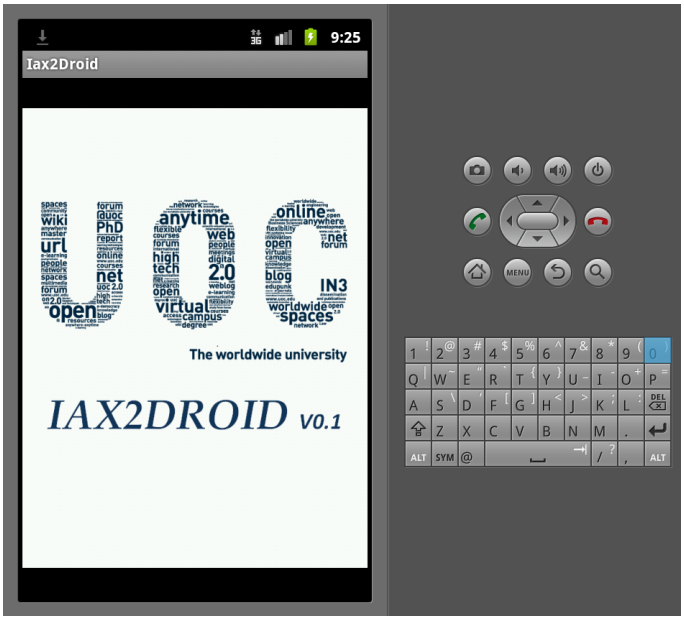


Ilustración 24 Pantalla inicial Iax2Droid

Para pasados unos pocos segundos mostrar automáticamente la pantalla principal que incluya los botones que permitan marcar los números, llamar, colgar, una pantalla, etc. Así como un selector de tabulación para pasar a la ventana de configuración de las propiedades de la conexión:

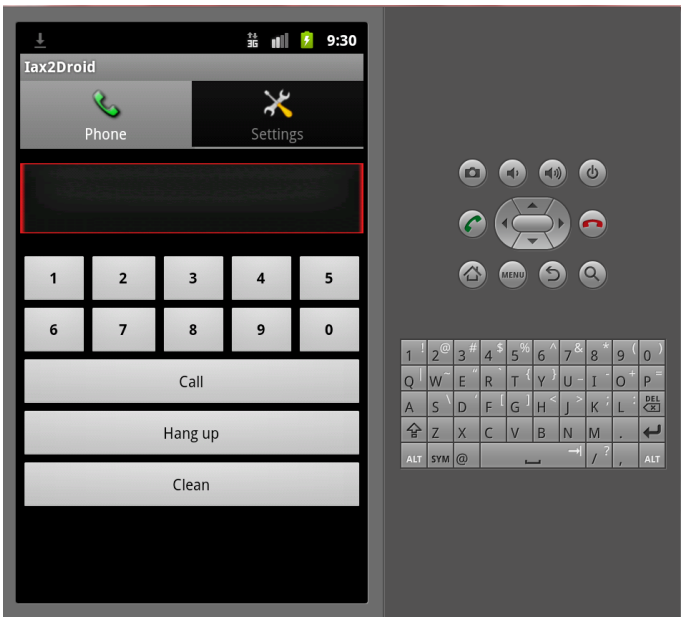


Ilustración 25 Dialer Iax2Droid

Las propiedades para la conexión que se pueden configurar se limitan a la dirección del PBX, el nombre de usuario y la contraseña para la extensión:

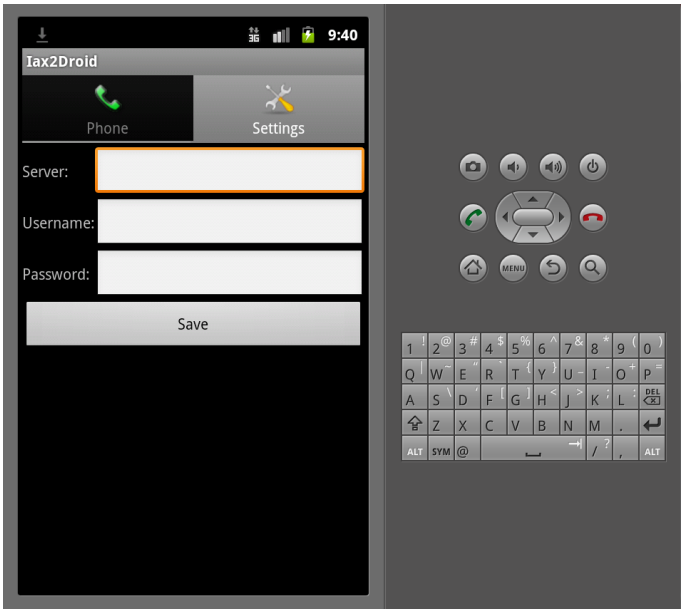


Ilustración 26 Settings Iax2Droid

En caso de que se intente realizar una llamada y las propiedades de la conexión no hayan sido fijadas se mostrará un mensaje alertando de ello:

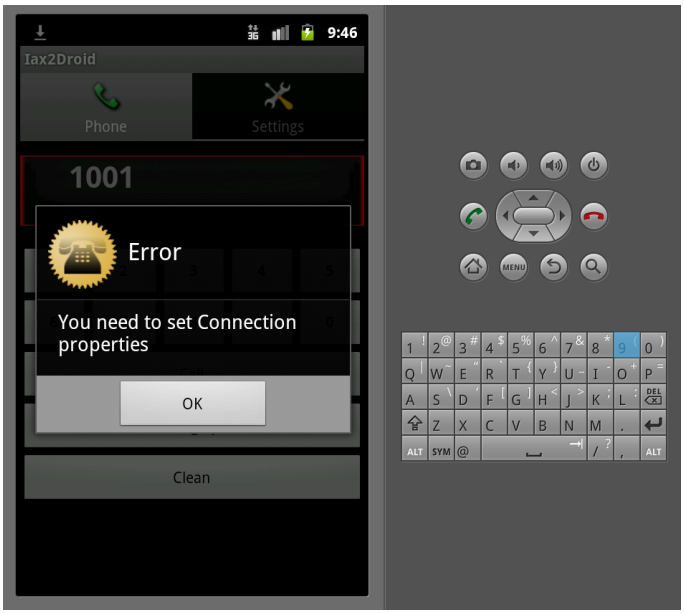


Ilustración 27 Mensaje de error Iax2Droid

8.3 BUSQUEDA DE SOPORTE IAX2

A diferencia del protocolo SIP donde encontramos un buen número de implementaciones en Java (además del soporte proporcionado para Android a partir de su versión GingerBread) en IAX2 las librerías son escasas destacando principalmente dos: njiax y Asterisk-Java IAX La primera de ellas, njiax, es un proyecto de la empresa española Nomasystems. Una vez descargada intentamos la ejecución de un sencillo ejemplo consistente en el registro de una extensión en nuestra PBX. Desafortunadamente, aunque se consigue registrar una extensión esta se cae inmediatamente. Enviamos un correo al responsable del proyecto, Enrique Marcote Peña, donde nos informa de que el proyecto ha sido abandonado y de que nunca ha llegado a estar operativo al 100% por lo que desistimos en nuestro empeño de usarla en el nuestro.

La segunda, Asterisk-Java IAX consiste en un applet donde la configuración de los parámetros de la conexión se debe hacer en el mismo código. Una vez registrada una extensión podemos observar un mensaje en el margen inferior izquierdo informándonos de ello:

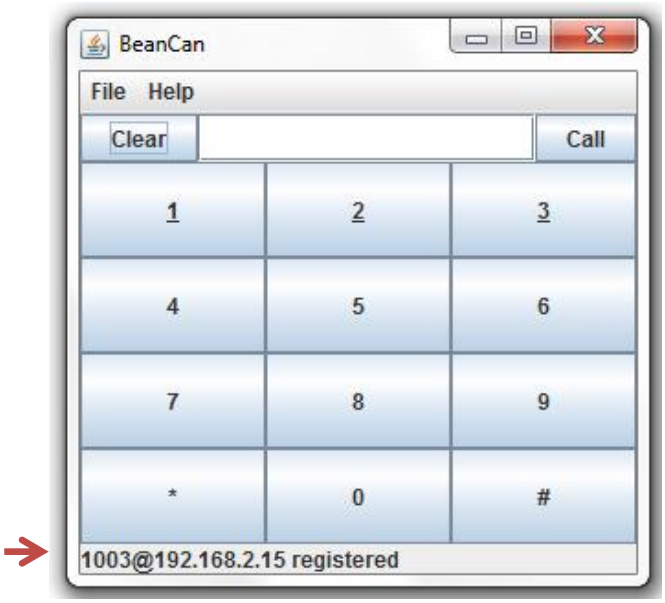


Ilustración 28 Applet registrado

Desde otra extensión realizamos pruebas y comprobamos que la captura y reproducción del audio funcionan adecuadamente así como que nos permite realizar y recibir llamadas. En el

margen inferior izquierdo nos informa en todo momento de las acciones que llevamos a cabo.

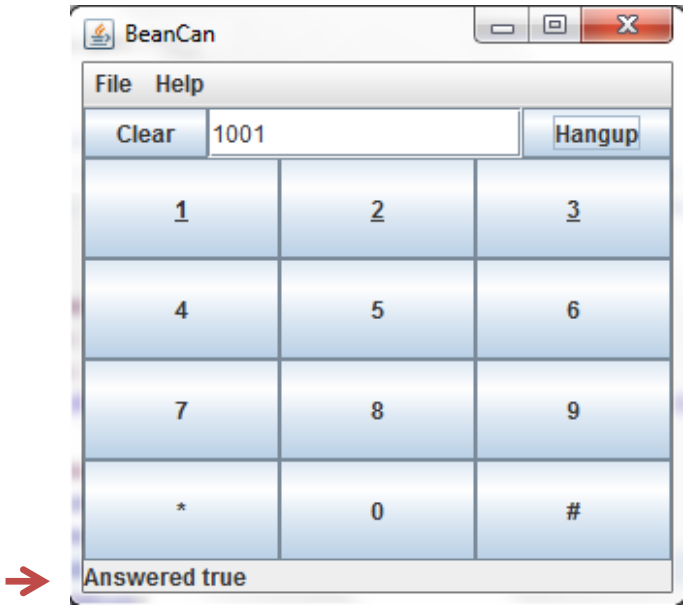


Ilustración 29 Applet en llamada

Desafortunadamente la codificación del protocolo se encuentra entremezclada con la interfaz gráfica y resulta imposible su migración a la plataforma Android.

Finalmente intentamos emplear una librería en C que implemente el protocolo IAX2 mediante el Android NDK pero desistimos debido a la extrema dificultad y dudosa viabilidad de tal solución.

9 IDEAS BÁSICAS PARA IMPLEMENTAR SOPORTE IAX2 EN ANDROID

Debido a la falta de una librería que nos proporcione el soporte necesario para el empleo del protocolo IAX2 en Android procederemos, a partir de este punto, a describir algunas ideas básicas que una implementación de este protocolo debería incorporar.

9.1 TRAMAS, ELEMENTOS DE INFORMACIÓN Y MENSAJES

En primer lugar necesitaríamos una **implementación de las tramas/frames** que se emplean en el protocolo IAX2. Como ya se estudió en el punto 5.2 de la memoria las dos principales tramas empleadas son la full y la mini (opcionalmente se podría contemplar la posibilidad de crear una superclase de la que heredarán las dos donde se recojan las características comunes). Por lo que como mínimo serían necesarias dos clases que las implementen.

Dentro de estas clases se implementaran los constructores, getters y setters de todos los campos incluidos en las correspondientes cabeceras así como del resto de la info contenida en la trama. Otra aproximación podría consistir en implementar clases para frames más específicos como de DTMF, voz o control que hereden de los anteriores.

FullFrame	
Atributo	**Descripción**
F	Indica que se trata de una trama full
SourceCallNumber	Identificador de la llamada en el origen
R	Indica si es una trama retransmitida
DestinationCallNumber	Identificador de la llamada en el nodo de destino
TimeStamp	Se utilizan para reordenar las tramas recibidas
Oseqno	Su propósito es identificar los streams de salida dentro de una llamada
Iseqno	Su propósito es identificar los streams recibidos dentro de una llamada
FrameType	Indica el tipo de la trama
C	Indica el formato del campo SubClass
SubClass	Indica la subclase del mensaje enviado por la trama
Data	Normalmente no suelen utilizarse para enviar el audio sino solo para señalización. Puede viajar info tipo media (audio, video,…) aunque no es recomendable

MiniFrame	
Atributo	**Descripción**
F	Indica que se trata de una trama full
SourceCallNumber	Identificador de la llamada en el origen
TimeStamp	Se utilizan para reordenar las tramas recibidas
Data	Normalmente audio

Las tramas en si se podrían implementar como un array de bytes donde a la cabecera, de un tamaño determinado (12 bytes full y 4 bytes mini), se le puede ir aumentando de tamaño para incluir los bytes de datos.

Otra clase que podría necesitarse es aquella que nos implemente los **elementos de información (IE)** que nos ayudan en la señalización del protocolo y viajan dentro de las tramas.
Constará de tres campos: En el primero nos indica el tipo de IE que se está enviando (se estudiaron en el punto 5.3 de esta memoria), el segundo la longitud del campo de datos y el tercero serán los datos en sí.

IE	
Atributo	**Descripción**
Type	Identificador del tipo de IE
Size	Tamaño del campo de datos
Data	(CALLED NUMBER, CALLING NUMBER, CALLED CONTEXT…)

Finalmente para este primer paquete de clases se incluiría una clase donde se implementaran los **mensajes** que se intercambian en una comunicación IAX2: IAX Requests, Responses, Signalling, Media IAX Messages… (Se estudiaron en el punto 5.4). Para implementar esta clase nos valdríamos de las anteriormente descritas. Esto quiere decir que un determinado mensaje, dependiendo de sus características, encapsularía determinados IE. IAX2 define una cantidad ingente de mensajes aunque no todos son indispensable para nuestro propósito por lo que no es necesario definirlos todos sino solo los más importantes, aquellos que se emplearían posteriormente en las operaciones más básicas: HANGUP, RINGING, ANSWER, BUSY, PROCEEDING, NEW, PING, PONG, ACK, HANGUP, REJECT, ACCEPT, AUTHREQ, AUTHREP, REGREQ, REGAUTH, POKE...

9.2 ENVÍO Y RECEPCIÓN DE DATAGRAMAS

Necesitaríamos **una clase que sea capaz de enviar/recibir los datagramas** UDP al PBX Asterisk. Llegados a este punto somos muy dependientes del soporte de red que

proporciona Android. En un principio Android permite el uso de paquete java.net por lo que mediante el uso de las clases ahí representadas, fundamentalmente:

DatagramPacket – Clase que implementa un datagrama UDP

DatagramSocket – Clase que implementa un socket UDP para enviar y recibir DatagramPackets

Deberíamos ser capaces de construir nuestros propios métodos que al menos nos proporcionaran las funcionalidades necesarias para enviar y recibir datagramas. Además deberíamos definir el tamaño del buffer, el puerto UDP y otras características relacionadas con el soporte de red.

9.3 CAPTURA Y REPRODUCCIÓN DE AUDIO

Llegados a este punto necesitaríamos acceso a nuestro hardware, concretamente a nuestro micrófono y altavoces para la captura y la reproducción del audio respectivamente. Para la captura disponemos de la clase MediaRecorder mediante la cual podemos especificar que solo capturaremos audio (no video), la codificación, etc. Para la reproducción disponemos de MediaPlayer con la que se nos facilitan los medios para ser capaces de reproducir streaming.

La idea general sería la de que cuando se realice una llamada o se atienda una, tanto la captura como la reproducción funcionen simultáneamente y este trabajo fuera en conjunción con las clases que se explicaron anteriormente de forma que el audio que nos llega en forma de información dentro de los miniframes se fuera reproduciendo y el audio que nosotros generemos se fuera enviando.

Un aspecto a tener en cuenta en el desarrollo es que el emulador no dispone de micrófono y por tanto es incapaz de capturar audio por lo que se debería disponer de un dispositivo real para poder probar todo aquello que se vaya desarrollando.

9.4 OPERACIONES IAX2

Sería conveniente aglutinar las operaciones/tareas IAX2 en una clase que nos permite acceder a tales servicios. Esta clase actuaría como **punto de inicio** para cualquier acción, de manera que pudiéramos crear sesiones, registrarnos en un servidor, realizar llamadas o permanecer a la escucha por si se produce alguna entrante etc. A modo de ejemplo se muestran algunos métodos que serían interesantes incluir:

IAX2Manager	
Método	**Descripción**
estaRegistrado	Método que nos informara de si se está registrado con la PBX
llamar	Inicializaría una llamada de manera que se necesitaría indicarle el origen, destino y activar los Listeners determinados para controlar el flujo de la llamada
registrarse	Método para registrarse en una PBX
escucharPBX	Controlaría los mensajes de control que de intercambiaran con la PBX
descolgar	Para aceptar una llamada que se está produciendo a nuestra extensión. Por descontado se debería tener activado algún listener encargado de activarse en caso de llegada de mensaje de llamada entrante
cancelarRegistro	Cancela el registro en una PBX

9.5 EVENTOS Y OPERACIONES RELACIONADAS CON LAS LLAMADAS

La inclusión de una clase cuya función sería la de la escucha de eventos sucedidos durante las llamadas IAX2 sería muy beneficiosa. Dentro de estos eventos podríamos incluir: La finalización de la llamada, el establecimiento de la llamada, número ocupado, llamada en espera, error en la llamada, etc.

Para la implementación de las operaciones relacionadas con las llamadas IAX2 se recomienda incluir los siguientes métodos:

IAX2AudioCall	
Método	**Descripción**
responderLlamada	Método cuya función sería la de responder a una llamada
mantenerEnEspera	Mantendría en espera una llamada sin finalizarla
continuarLlamada	Continuaría una llamada que previamente se hubiera puesto en espera
silenciarMicro	Desactivaría el micrófono
activarMicro	Activaría el micrófono

finalizarLlamada	Finalizaría una llamada
estaEnLlamada	Nos informaría de si una llamada se ha establecido
estaEnEspera	Nos informaría de si una llamada se ha pasado a estado de espera
estaSilenciado	Nos informaría de si el micrófono se ha silenciado

Finalmente solo indicar que la implementación de una librería que de soporte a un protocolo de VoIP como IAX2 es una tarea harto complicada y requiere de unos conocimientos y experiencia en la materia muy avanzados así como de mayor tiempo para poderlo llevar a cabo.

Las ideas aquí expuestas son solo algunas ideas basadas en la revisión de las librerías implementadas para otros lenguajes de programación

10 INTEGRACIÓN DE CONTACTOS DEL TELÉFONO CON LA APLICACIÓN

Una manera elegante de facilitar la realización de llamadas consiste en permitir a los usuarios acceder a sus contactos y seleccionando el número deseado proceder con el marcado.

En Android el acceso a los contactos se realiza por medio de ContactsContract que permite acceder a los datos de la agenda desde diferentes puntos de vista.

Para realizar la integración de los contactos de nuestro dispositivo en nuestra aplicación primeramente debemos otorgar los permisos necesarios para poder acceder a los mismos en el fichero AndroidManifest.xml.

A la hora de añadir un nuevo contacto en Android uno de los campos opcionales que se nos permite cumplimentar es el de "Internet Call", este campo al que se puede acceder a traves de CommonDataKinds.SipAddress representa un dirección/extensión SIP para el contacto pero para nuestro caso podemos emplearlo perfectamente para extensiones/direcciones IAX2.

Aprovechando esta característica vamos a listar aquellos contactos que tengan una "Internet Call" y su nombre permitiendo seleccionar uno y procediendo al marcado de su extensión.

La nueva pestaña que hemos añadido en la aplicación para mostrar los contactos y la lista de contactos tendrían el siguiente aspecto:

Ilustración 30 Lista de contactos

Una vez se ha clickeado en el contacto deseado la aplicación cambia automáticamente a la pestaña "phone" y marcaría la extensión asociada mostrándola en la pantalla:

Ilustración 31 Marcado extension del contacto

11 ESTRUCTURA Y COMPOSICIÓN DE LOS FICHEROS DE IMPLEMENTACIÓN

El proyecto recibe el nombre de "Iax2Droid". Dentro del paquete "uoc.pfc" podemos encontrar los siguientes archivos:

uoc.pfc	
Archivo	**Descripción**
Contact.java	Clase empleada para almacenar los contactos que se mostrarán en nuestra tercera pestaña
Iax2Droid.java	Está definida en AndroidManifest.xml como la actividad principal y contiene la imagen inicial tras la que unos breves segundos salta a la pantalla principal de la aplicación
Properties.java	Contiene una clase empleada para almacenar las propiedades de la conexion con la PBX
Mainmenu.java	Contiene la pantalla principal de la aplicación y la lógica de programación

Como se ha mencionado el fichero más importante es Mainmenu.java ya que en el reside toda la lógica de la aplicación:

Mainmenu		
ContactAdapter	Clase	Clase creada para contener la lógica para las filas adaptadas de nuestra lista de contactos
ContactHolder	Clase	Clase que rellena una fila con la información de un contacto determinado
onCreate	Método	Método llamada al comienzo de la actividad
onCalling	Listener	Escucha cuando se presiona el botón de llamada y en caso de que las propiedades esten vacias lanza el método Alert()
onDeleting	Listener	Escucha cuando se presiona el botón de borrado y vacia la pantalla

onHangingUp	Listener	Escucharía cuando se presiona el botón de descolgar
onListClick	Listener	Escucha cuando se selecciona un contacto y recoje la extension para pasarsela a la pantalla del telefono
onPressed0	Listener	Escucha cuando se presiona el botón y marca el número en la pantalla
onPressed1	Listener	Escucha cuando se presiona el botón y marca el número en la pantalla
onPressed2	Listener	Escucha cuando se presiona el botón y marca el número en la pantalla
onPressed3	Listener	Escucha cuando se presiona el botón y marca el número en la pantalla
onPressed4	Listener	Escucha cuando se presiona el botón y marca el número en la pantalla
onPressed5	Listener	Escucha cuando se presiona el botón y marca el número en la pantalla
onPressed6	Listener	Escucha cuando se presiona el botón y marca el número en la pantalla
onPressed7	Listener	Escucha cuando se presiona el botón y marca el número en la pantalla
onPressed8	Listener	Escucha cuando se presiona el botón y marca el número en la pantalla
onPressed9	Listener	Escucha cuando se presiona el botón y marca el número en la pantalla
onSave	Listener	Escucha cuando se presiona el botón y guarda las propiedades de la conexión con la PBX
alert	Método	Se encarga de presentar un mensaje informando de que es necesario rellenar las propiedades de conexión

Ya por ultimo con respecto al Layout tenemos tres ficheros:

Layout	
Archivo	**Descripción**
splash.xml	Contiene una imageView que será la que se muestre durante unos segundos en la pantalla al iniciarse la aplicación
row.xml	Contiene la estructura de cada una de las filas que van a contener a nuestros contactos
main.xml	Contiene la estructura para la pantalla principal, incluyendo la del dialpad y su pantalla así como la de la recogida de datos para las propiedades

12 CONCLUSIÓN

La aplicación desarrollada no ha podido cumplir al 100% los objetivos que en un principio nos habíamos marcado, la falta de soporte ha sido un impedimento demasiado grande que no hemos podido superar. De hecho una gran cantidad del tiempo ha sido empleado en la búsqueda y prueba de librerías en JAVA que pudieran ofrecernos un soporte valido para ser empleadas en Android. Una vez se ha descartado este punto por la pobre oferta existente y la baja calidad y/o portabilidad hemos intentado abrir una nueva vía mediante el empleo de implementaciones en lenguaje C del protocolo y Android ndk pero lo hemos descartado por su dudosa viabilidad.

IAX2 no es todavía un protocolo demasiado empleado si lo contraponemos a SIP y es de esperar que en los próximos meses/años comience a aumentar la oferta de software que lo ofrezca así como la de librerías que lo implementen con el fin de saltarse los impedimentos por parte de los proveedores telefónicos.

Actualmente para la plataforma Android existen dos aplicaciones disponibles a través del AndroidMarket llamadas: IaxAgent Beta y MPhoneGG, las cuales no han podido ser probadas debido a que el emulador no permite la instalación de aplicaciones del AndroidMarket como medida para prevenir la libre distribución de aplicaciones de pago, y aunque las aplicaciones parecen funcionar correctamente a través de las conexiones WiFi de los teléfonos, las críticas de los usuarios en sus respectivas páginas se centran en los cortes

en las comunicaciones cuando las llamadas se realizan a través de las conexiones de datos con los operadores de telefonía. Evidentemente como ya se habló al comienzo del proyecto los operadores telefónicos no están dispuestos a permitir que las comunicaciones VoIP fluyan libremente por sus redes de datos mermando su parte del negocio por lo que aspectos como la eficiencia de la librería que proporcione el soporte se antoja crítico.
En el momento en el que estuviera disponible una implementación libre y funcional para Android del protocolo, la integración con la interfaz desarrollada no debería llevar asociada una gran dificultada y el aspecto en el que tendríamos que centrarnos sería el relativo a la codificación/descodificación del audio.

GLOSARIO DE TÉRMINOS

SIP – (Session Initiation Protocol) es un protocolo desarrollado por el grupo de trabajo MMUSIC del IETF con la intención de ser el estándar para la iniciación, modificación y finalización de sesiones interactivas de usuario donde intervienen elementos multimedia como el video, voz, mensajería instantánea, juegos en línea y realidad virtual.
VoIP – (Voz sobre Protocolo de Internet) es un grupo de recursos que hacen posible que la señal de voz viaje a través de Internet empleando para ello un protocolo IP (Protocolo de Internet)
ACK – Un ack es un señal pasada a través de un proceso de comunicación para significar la recepción o respuesta como parte de un protocolo de comunicación.
Dialpad – se refiere al teclado empleado para la marcación de los números en las llamadas telefónicas realizadas con móviles o softphones.
Applet – componente de una aplicación que se ejecuta en el contexto de otro programa, por ejemplo un navegador web. El applet debe ejecutarse en un contenedor, que lo proporciona un programa anfitrión, mediante un plugin, o en aplicaciones como teléfonos móviles que soportan el modelo de programación por 'applets'
RTP – Siglas de Real-time Transport Protocol (Protocolo de Transporte de Tiempo real). Protocolo de nivel de sesión utilizado para la transmisión de información en tiempo real, como por ejemplo audio y vídeo en una video-conferencia.
UDP – protocolo del nivel de transporte basado en el intercambio de datagramas (Paquete de datos). Permite el envío de datagramas a través de la red sin que se haya establecido

previamente una conexión.

PBX – Central telefónica capaz de gestionar llamadas internas, las entrantes y salientes con autonomía sobre cualquier otra central telefónica

Layout – Calculo en el que los distintos objetos que ocupan la pantalla se distribuyen en ella dependiendo de sus distintas dimensiones.

URI – cadena de caracteres corta que identifica inequívocamente un recurso (servicio, página, documento, dirección de correo electrónico, etc.). Normalmente estos recursos son accesibles en una red o sistema.

BIBLIOGRAFÍA/WEBGRAFÍA

RFC 5456 "IAX: Inter-Asterisk eXchange Version 2"
http://tools.ietf.org/search/rfc5456

Inter-Asterisk Exchange (IAX): Deployment Scenarios in SIP-Enabled Networks
Mohamed Boucadair. Publisher: Wiley (March 23, 2009)
http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0470770724.html

Asterisk Reference Information Version SVN-branch-1.6.2-r311140
http://www.asterisk.org/docs

Wireshark User's Guide 36237 for Wireshark 1.5
http://www.wireshark.org/download/docs/user-guide-a4.pdf

How to enable SIP support in Android 2.3 (Gingerbread) emulator
http://xilard.hu/

Asterisk-Java (IAX Java implementation IAX2 protocol)
http://asterisk-java.org/iax/development/

Installing the SDK
http://developer.android.com/sdk/installing.html

The Busy Coders Guide to Android Development

http://commonsware.com/books

Android Programming Tutorials

http://commonsware.com/books

Asterisk For Dummies

Mark Spencer, Stephen P. Olejniczak and Brady Kirby (Feb 9, 2007)

http://www.dummies.com/store/product/Asterisk-For-Dummies.productCd-0470098546.html

njiax (IAX Java implementation IAX2 protocol)

http://code.google.com/p/njiax/

A reference for VOIP, primarily Asterisk

http://www.voip-info.org/

Hello, Android: Introducing Google's Mobile Development Platform

Ed Burnette (Aug 4, 2010)

http://pragprog.com/titles/eband/hello-android