UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA DE TELECOMUNICACIONES

PROYECTO FIN DE CARRERA

IMPLEMENTACIÓN DE MOVILIDAD DE SESIÓN CON EL PROTOCOLO SIP

Autora: Ainhoa Sesmero Fernández

Director: María Calderón Pastor

Co-Director: Víctor Sandonís Consuegra

Agradecimientos

Agradecer a todas aquellas personas que han convivido conmigo durante mi carrera, pero sobre todo porque me han ayudado en los momentos más difíciles. Me han dado todo su apoyo para no rendirme nunca y seguir adelante, haciendo posible que hoy me encuentre finalizando una etapa de mi vida. MUCHAS GRACIAS.

Primero quiero dar las gracias a mis padres, Victoriano y Mª Carmen, y a mi hermano Víctor Javier. Sin vosotros no habría llegado hasta aquí. Vosotros me habéis dado la fuerza suficiente para superar todos los obstáculos y me habéis apoyado en los momentos más duros de mi carrera. Gracias por saber qué decirme en los momentos adecuados con las palabras exactas. Pero sobre todo por aguantarme en mis días duros. Sin vosotros nunca hubiera sido ingeniera. Os quiero.

Pero no sólo quiero dar las gracias a mi familia directa, sino al resto de mi familia que siempre han estado ahí cuando los he necesitado y nunca me han dado la espalda. Se han preocupado todo este tiempo por mis estudios y me han dado los ánimos suficientes para seguir hacia adelante.

Agradecer tanto a María como a Víctor su dedicación y tiempo empleado en el proyecto. Me han ayudado en todo lo que he necesitado y han hecho posible que continuará hasta el final.

Gracias a mis amigas de la infancia, que me apoyaron para comenzar esta carrera y no me echará atrás en mi decisión. Han sabido hacerme reír y olvidar los estudios por unos momentos. No olvidéis que os quiero.

No puedo finalizar los agradecimientos sin nombrar a un grupo de personas peculiares, es broma, al mejor grupo de personas que he conocido nunca, mis amigos de la universidad. Con ellos he pasado de los mejores momentos de mi vida, pero también de los peores. Hemos discutido por prácticas, hemos llorado por los exámenes, hemos reído en el césped de la universidad… Han sido capaces de soportarme todos estos años pero sobre todo de sacarme una sonrisa. Por todo esto, y muchas cosas más que ellos saben, MUCHÍSIMAS GRACIAS por saber comprenderme y ayudarme en todo lo que he necesitado. Aunque nunca os lo digo, os quiero: Alba, Carmen, Dani, Sandra y Tania.

Resumen

Lo que se pretende, es poder transferir una sesión multimedia a otro dispositivo móvil sin necesidad de finalizar la sesión. Esta transferencia la puede iniciar otro terminal ajeno a la sesión (modo *pull*) o un terminal participante en la sesión (modo *push*).

El proyecto de fin de carrera ha consistido en implementar la movilidad de sesión mediante la utilización del protocolo SIP en una aplicación Android. Para ello se ha utilizado la aplicación IMSDroid, de software libre. Esta aplicación es el primer cliente IMS para dispositivos Android. IMSDroid utiliza la librería android-ngn-stack, que se apoya en las características de doubango para poder realizar todas las funciones del cliente SIP/IMS. Para el desarrollo de la implementación del proyecto, es necesario modificar, eliminar o incluir nuevo código no sólo de la aplicación IMSDroid, sino también en el código de doubango.

Adicionalmente, ha sido preciso disponer de una base de datos para almacenar información sobre la sesión multimedia y sobre el usuario registrado. Para ello se ha utilizado un servidor apache, con el fin de realizar la comunicación entre el terminal móvil y la base de datos.

Abstract

The main intention is to be able to transfer a multimedia session to another mobile device without terminating the session. This transfer can be started by a device which is not involved in the session (pull mode) or by a device which is taking part in the session (push mode).

This project consists on implementing a session transfer by using the SIP protocol in an Android application. For that the IMSDroid application was used which is open source and free software. This application is the first IMS client for Android devices. IMSDroid makes use of the android-ngn-stack library which employs doubango's characteristics to be able to carry out all of the functionality of a SIP/IMS client. To be able to develop the implementation of the project, it is necessary to modify, delete or add new code, not only for the IMSDroid application but also in doubango's code.

In addition, it was necessary to make use of a data base to store information about the multimedia session and about the registered user. For that an Apache server was used, so its purpose was to enable the communication between the mobile device and the data base.

Índice

Capítulo 1. Introducción..... 23

1.1- Motivación del proyecto 23

1.2- Objetivos del proyecto 24

1.3- Contenido de la memoria..... 25

Capítulo 2. Estudio del Arte..... 28

2.1- IPTV 28

2.1.1- Arquitectura basada en no NGN 29

2.1.2- Arquitectura basada en NGN 30

2.1.3- Arquitectura basada en IMS..... 31

2.2- IMS 31

2.2.1- Arquitectura IMS 32

2.3- SIP 34

2.3.1- Clientes y Servidores SIP 34

2.3.1.1- User Agents 34

2.3.1.2- Back-To-Back User Agents (B2BUA) 35

2.3.1.3- Servidores SIP 35

2.3.1.3.1- Proxy 35

2.3.1.3.2- Redirect 35

2.3.1.3.3- Registrar 36

2.3.2- Mensajes SIP 36

2.3.2.1- Principales métodos SIP 37

2.3.3- Operaciones principales en SIP 37

2.3.3.1- Registro de un usuario SIP 37

2.3.3.2- Establecimiento de una sesión 38

2.4- Movilidad SIP 41
2.4.1- Movilidad de Terminal 41
2.4.2- Movilidad Personal 41
2.4.3- Movilidad de Servicio 41
2.4.4- Movilidad de Sesión 42
2.4.4.1- MNC 42
2.4.4.2- SH 43
Capítulo 3. Solución a la movilidad de sesión basada en SIP 46
3.1- Movilidad de sesión 47
3.2- Application Server 48
3.2.1- Mobility Manager 48
3.2.2- Transparency Manager 48
3.3- Señalización concreta para los distintos tipos de transferencia de sesión 49
3.3.1- Modo pull 49
3.3.2- Modo push 52
3.3.2.1- GRUUs 54
Capítulo 4. Librería Doubango y aplicación IMSDroid 56
4.1- Doubango 56
4.1.1- Introducción a la librería nativa de IMSDROID 56
4.1.2- TinySIP 58
4.2- Aplicación IMSDroid 60
Capítulo 5. Implementación de la movilidad de sesión con el protocolo SIP en la aplicación IMSDroid 65
5.1- Error al colgar una sesión multimedia entrante 66
5.2- CAS 67
5.2.1- Modo Pull 68
5.2.2- Modo Push 69
5.3- Modo Pull 70
5.4- Modo Push 74
5.4.1- Globally Routable User Agent Uris (GRUUs) 74
5.4.2- Implementación del envío del REFER en la aplicación 76
5.5- Cambio de la interfaz 79
Capítulo 6. Evaluación de la implementación 83
6.1- Entorno de pruebas 84

6.1.1- CAS 86
6.1.2- Modo Pull 87
6.1.3- Modo Push 87
6.2- Evaluación y resultados. 87
6.2.1- Modo Pull 88
6.2.1- Modo Push 89
Capítulo 7. Conclusiones 92
Apéndice A. Planificación del proyecto 95
A.1- División en actividades y tareas del proyecto. 95
A.2- Cuadro resumen de la planificación temporal del proyecto 97
Apéndice B. Presupuesto 99
Apéndice C. Manual de Instalación 103
C.1- Eclipse, SDK y NDK 103
C.2- Apache, MySQL y PHP 105
Apéndice D. Manual de la librería Doubango 108
D.1- Descarga del código 108
D.2- Ajuste del fichero root.mk 108
D.3- Compilación y construcción de la librería 109
Apéndice E. Manual de la aplicación IMSDroid 111
E.1- Configuración 111
E.1.1- Identidad 112
E.1.2- Ajuste de red 113
Apéndice F. Manual de MySQL 115
F.1- Conectarse y desconectarse del servidor 115
F.2- Crear y utilizar una base de datos 115
F.3- Crear y utilizar las tablas 116
Apéndice G. Código del CAS 119
G.1- Ficheros de comunicación 119
G.1.1- Modo pull 120
G.2.1.1- Establecimiento de la sesión 120
G.1.1.2- Finalización de la sesión 121
G.1.1.3- Petición de todas las sesiones establecidas 121
G.1.1.4- Información de la sesión 121
G.1.2- Modo Push 122

G.1.2.1- Registro de un usuario 122
G.1.2.2- Desregistro de un usuario 123
G.1.2.3- Petición de todos los registros 123
Bibliografía 125

Índice de Figuras

Figura 1: Arquitectura IPTV basada en no NGN 29

Figura 2: Arquitectura IPTV basada en NGN 30

Figura 3: Arquitectura IPTV basada en IMS 31

Figura 4: Arquitectura IMS 33

Figura 5: Registro de un usuario 37

Figura 6: Establecimiento de sesión 39

Figura 7: Modo MNC para la transferencia a un dispositivo 43

Figura 8: Modo SH para transferir una sesión a un dispositivo 44

Figura 9: Elementos de la solución 46

Figura 10: Componentes del IPTV Continuity Agent 48

Figura 11: Señalización para el modo pull 50

Figura 12: Señalización para el modo push 52

Figura 13: Arquitectura IMS de Doubango 57

Figura 14: Esquema de la aplicación IMSDroid 60

Figura 15: Esquema de la librería android-ngn-stack 61

Figura 16: Cabecera Record-Route del mensaje 200 OK 66

Figura 17: Cabecera Record-Route del mensaje INVITE 67

Figura 18: Cabecera Route del mensaje BYE 67

Figura 19: Tabla user_sip 68

Figura 20: Tabla information 68

Figura 21: Tabla register 69
Figura 22: Menú de inicio con el nuevo icono 71
Figura 23: Actividad Modo Pull 72
Figura 24: Erros al realizar la petición 72
Figura 25: Mensaje OK como respuesta al INVITE 73
Figura 26: Cabecera Replace del nuevo INVITE 73
Figura 27: Mensaje REFER 76
Figura 28: Pantalla para transferir una sesión 77
Figura 29: Menú en la pantalla cuando se ha establecido una sesión de audio/video 78
Figura 30: Pantalla cuando se recibe el INVITE 79
Figura 31: Menú principal 79
Figura 32: Pantalla para establecer una sesión 80
Figura 33: Pantalla para establecer una llamada de audio 80
Figura 34: Pantalla al recibir una llamada 81
Figura 35: Pantalla para rechazar una sesión que todavía no se ha establecido 81
Figura 36: Menú de opciones 81
Figura 37: Escenario de pruebas 84
Figura 38: Modo Pull: Establecimiento de la sesión 85
Figura 39: Modo Pull: Transferencia de sesión 86
Figura 40: Esquema de pruebas para el modo pull 88
Figura 41: Esquema de pruebas para el modo push 89
Figura 42: Cuadro de diálogo 103
Figura 43: Android SDK Manager 104
Figura 44: Web de información del servidor 106
Figura 45: Pantalla de identidad 112
Figura 46: Pantalla de ajuste de red 113
Figura 47: Mostrar las bases de datos disponibles 116

Figura 48: Sentencia para acceder a una base de datos 116

Figura 49: Sentencia para crear una base de datos 116

Figura 50: Sentencia para mostrar las tablas 116

Figura 51: Ejemplo de creación de una tabla 116

Índice de Tablas

Tabla 1: Planificación temporal del proyecto..... 97

Acrónimos

SIP:	Session Initiation Protocol
IMS:	IP Multimedia Subsystem
IPTV:	Internet Protocol Television
HSS:	Home Subscriber Server
CSCF:	Call Session Control Function
P-CSCF:	Proxy CSCF
I-CSCF:	Interrogating CSCF
S-CSCF:	Serving CSCF
AS:	Application Servers
URI:	Uniform Resource Identifier
UAC:	User Agent Client
UAS:	User Agent Server
B2BUA:	Back-to-back User Agent
MNC:	Mobile Node Control
SH:	Session Handoff
MN:	Mobile Node
CAS:	Context-Awareness System
GRUU:	Global Routable User Agent URI
3GPP:	3rd Generation Partnership
NGN:	Next Generation Networking
UMTS:	Universal Mobile Telecommunications System

GRPS:	General Packet Radio Service
LTE:	Long Term Evolution
RAC:	Resource and Admission Control Subsystem
NAS:	Network Attached Storage
IP:	Internet Protocol
VoIP:	Voice over IP
UDP:	User Datagram Protocol
SDP:	Session Description Protocol
RTP:	Real-time Transport Protocol
RTCP:	RTP Control Protocol
RTSP:	Real Time Streaming Protocol
HTTP:	Hypertext Transfer Protocol
SMTP:	Simple Mail Transfer Protocol
DNS:	Domain Name System
XML:	Extensible Markup Language
XCAP:	XML Configuration Access Protocol
MSRP:	Message Session Relay Protocol
QoS:	Quality of Service
TDT:	Televisión Digital Terrestre

Capítulo 1

Introducción

En este capítulo se exponen los principales objetivos del proyecto, así como su motivación. Además se explicará brevemente la estructura de la memoria del proyecto.

1.1- Motivación del proyecto

En estos últimos años ha aumentado la penetración de la banda ancha en España. Este hecho se debe a que los usuarios tienen la necesidad de estar conectados continuamente a través de diferentes dispositivos y poder estar comunicados con sus amigos, informados de las noticias que ocurren a su alrededor o simplemente visualizando cualquier vídeo.

Este proyecto se centra en ofrecer a los clientes registrados movilidad de sesión para servicios IPTV (Internet Protocol Television). IPTV permite visualizar la televisión en directo, vídeos bajo demanda o incluso televisión diferida, es decir, ver películas o programas que han sido emitidos pero que el cliente no los ha podido visualizar. Para ofrecer este servicio se utiliza el protocolo IP mediante una red de conmutación de paquetes, Internet.

Hoy en día los operadores están apostando la adopción de la plataforma IMS (IP Multimedia Subsystem) para proporcionar, a través de sus redes IP, servicios multimedia tales como VoIP o IPTV. Por este motivo, se va a utilizar la arquitectura IMS en la implementación de este proyecto.

La gran ventaja de IMS reside en el lado de los operadores. Permitiendo a estos provisionar los servicios de una manera flexible, así como controlar la autenticación y autorización de los usuarios para acceder a los servicios.

Será el protocolo SIP (Session Initiation Protocol) el que se utilice para poder establecer sesiones multimedia entre dos o más dispositivos.

Sin embargo la motivación de este proyecto no reside, solamente, en establecer una comunicación entre un dispositivo móvil y un Servidor IPTV, sino en poder transferir dicha sesión multimedia a otro terminal diferente.

Resulta interesante poder estar viendo cualquier canal televisivo o un vídeo bajo demanda en el teléfono móvil y poder transferirlo a la televisión por cualquier motivo, como por ejemplo que el móvil se ha quedado sin batería o simplemente el usuario se ha cambiado de habitación y hay disponible una televisión que prefiere usar. A este tipo de movilidad se denomina movilidad de sesión.

Este proyecto se basa en implementar dos tipos de movilidad de sesión mediante el protocolo SIP. Estos dos tipos son: modo pull y modo push. En el modo pull, el usuario indica desde otro terminal, ajeno a la comunicación establecida, que desea transferir la sesión a este terminal. Sin embargo para el modo push, el usuario indica desde el terminal que tiene establecida la sesión a que dispositivo quiere transferirla.

Como se ha mencionado antes se va a utilizar un dispositivo móvil, que iniciará una sesión multimedia con un Servidor IPTV. Los dispositivos móviles son los encargados de soportar e iniciar la movilidad de sesión. Para ello se pretende realizar una aplicación móvil que soporte la transferencia de sesión.

Hoy en día triunfan dos grandes sistemas operativos para los móviles, Android e iOS. Se eligió Android no sólo porque existen más dispositivos con dicho sistema operativo, sino por una aplicación Android de código abierto, IMSDroid. Esta aplicación se comporta como un cliente IMS/SIP sin movilidad, con lo que se debe de implementar la movilidad de sesión en esta aplicación.

1.2- Objetivos del proyecto

Este proyecto tiene un gran objetivo: poder dotar a la aplicación IMSDroid de la capacidad de movilidad de sesión. Por lo que no es necesario finalizar e iniciar una sesión completamente nueva entre dos participantes, ya que un terminal es capaz de transferir dicha sesión a otro dispositivo móvil.

Uno de los extremos de la comunicación no tiene ser porque soportar los procedimientos de movilidad de sesión, sólo es necesario que los soporten los terminales que deseen llevar a cabo la transferencia de sesión. Para solucionar este problema es preciso introducir un nuevo elemento en la comunicación, Application Server (AS). Éste se encargará de gestionar la señalización SIP y poder realizar la transferencia de sesión. Además el extremo de la comunicación permanece ignorante de la movilidad de sesión.

Hay dos tipos de movilidad de sesión, modo *pull* y modo *push.* Estos dos tipos se desarrollarán en la implementación del proyecto.

Estos tipos de movilidad se utilizan por ejemplo en la siguiente situación: se ha establecido una llamada mediante un dispositivo móvil y, en cierto momento de la

conversación, uno de los participantes llega a la oficina y desea transferir la llamada a su PC.

Especialmente, se centra en que un dispositivo de la sesión sea un Servidor IPTV, el cual es capaz de dar acceso en tiempo real a canales de televisión o poder visualizar vídeos almacenados en el servidor. El Servidor IPTV no realizará ningún tipo de movilidad, simplemente será el destino de la sesión multimedia y, serán los dispositivos móviles los que realicen dichos tipos de movilidad de sesión.

Otro objetivo secundario del proyecto es familiarizarse con el entorno de las bases de datos y las peticiones a un servidor, ya que son funciones que se van a necesitar para la realización del proyecto. Para poder realizar la transferencia de sesión es necesario tener cierta información de la sesión establecida o de un participante de la comunicación. Por ello es preciso tener una base de datos para almacenar esta información.

El último objetivo del proyecto es modificar la interfaz de la aplicación. Se debe a que IMSDroid es un cliente SIP/IMS que se utiliza normalmente para realizar llamadas de voz o para realizar videoconferencias, pero el proyecto se basa en un cliente IPTV.

1.3- Contenido de la memoria

La memoria del proyecto está constituida por siete capítulos y apéndices. A continuación se describe brevemente cada parte de la memoria.

Capítulo 1: Introducción.

Capítulo 2: Estado del Arte. En este capítulo se presentan las tecnologías en las que se sustenta este proyecto.

Capítulo 3: Solución a la movilidad de sesión basada en SIP. Se describen los componentes utilizados para la realización del proyecto y las soluciones que se implementarán detalladamente.

Capítulo 4: Librería Doubango y aplicación IMSDroid. Se explica la estructura de esta librería centrándose en el paquete tinySIP, ya que éste es el que realiza todas las funciones del protocolo SIP. Además se describe el código más relevante de la aplicación.

Capítulo 5: Implementación de la movilidad de sesión con el protocolo SIP en la aplicación IMSDroid. Es en este capítulo donde se describen los pasos a seguir para poder conseguir la movilidad de sesión en la aplicación IMSDroid. Así como la identificación de los cambios realizados en la interfaz de la aplicación y algunos problemas encontrados en la aplicación.

Capítulo 6: Evaluación de la implementación. Se describen las pruebas realizadas para comprobar el funcionamiento del código implementado y se evalúa el tiempo en establecer la transferencia de sesión.

Capítulo 7: Conclusiones. Analiza si se han cumplido los objetivos del proyecto y valora los resultados obtenidos. También comenta los trabajos futuros que se podrían realizar.

En los apéndices se incluyen los manuales de instalación de los programas necesarios, la planificación en tareas y temporal del proyecto, el presupuesto, un pequeño manual para poder construir y compilar la librería nativa, unas pautas para el manejo de la aplicación IMSDroid, un manual con las instrucciones básicas para manipular una base de datos y el código utilizado para la comunicación entre la base de datos y el terminal móvil.

Capítulo 2

Estudio del Arte

2.1- IPTV

IPTV es una tecnología que permite hacer llegar a los clientes la señal digital de un servicio como es el de la televisión y el vídeo, utilizando conexiones de banda ancha sobre el protocolo IP. Como se ha mencionado en la introducción, permite a los usuarios ver la televisión en directo, vídeos almacenados o vídeos bajo demanda. El abonado decide qué contenidos quiere ver y cómo los quiere visualizar. Por lo que se consigue personalizar la oferta a cada cliente y olvidarnos de los horarios de los programas televisivos.

IPTV dispone de un conjunto de servidores, en los cuales se guardan los contenidos, para ofrecer dichos servicios. De este modo se podrán visualizar los vídeos cuando el cliente lo solicite.

Para la distribución de los programas televisivos se utiliza la técnica multicast. De esta forma la señal de los canales de televisión es alcanzada por los usuarios que se encuentren sintonizando un canal en concreto.

Pero no sólo utiliza la técnica multicast, sino que también usa la unicast. Esta técnica se usa para el caso de los contenidos en vídeo bajo demanda ya que el cliente tiene el absoluto control de la reproducción del programa.

A parte de las ventajas ya mencionadas de IPTV como son el vídeo bajo demanda o los contenidos almacenados, también dispone de una publicidad personalizada. Los clientes pueden determinar cuáles son sus aficiones o gustos para poder recibir ofertas publicitarias de dichas áreas. Otra ventaja es que ofrece servicios de valor añadido, gracias a la conexión de banda ancha se puede tener acceso al correo electrónico, buscadores, etc.

La arquitectura IPTV ha ido evolucionando. A continuación se muestran las diferentes arquitecturas:

- **IPTV basada en no NGN:** Es posible hacer algún interfuncionamiento con las NGN, pero en general es un servicio de control independiente. Se utiliza para servicios de IPTV basados en una propiedad de middleware IPTV.
- **IPTV basada en NGN pero no en IMS:** Permite la interacción y el interfuncionamiento sobre puntos de referencia entre funciones específicas de IPTV y algunos elementos existentes de NGN (Next Generation Networking), como por ejemplo elementos de control de transporte para la admisión de recursos y subsistema de control (RAC) o el subsistema de conexión a la red (NAS).
- **IPTV basada en IMS:** Permite la reutilización de la funcionalidad IMS, permitiendo a los operadores ofrecer servicios de voz y de IPTV, así como la implementación de servicios que combinan los servicios convencionales de televisión con funciones de telefonía.

2.1.1- Arquitectura basada en no NGN

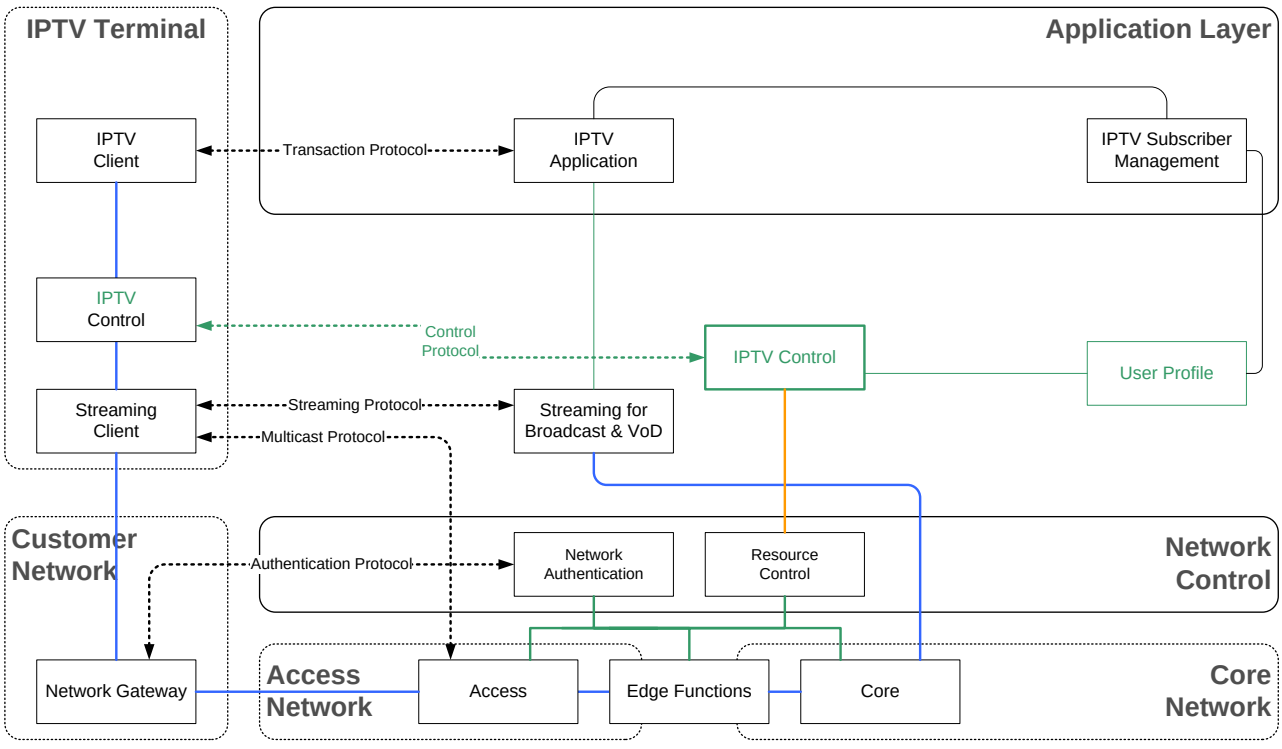


Figura 1: Arquitectura IPTV basada en no NGN [12].

Como se observa en la figura anterior, la primera arquitectura de IPTV no tiene estandarizado nada en el control de la red, son los propios vendedores los que implementan esta capa. Además la modularidad y los protocolos que se pueden utilizar son escasos por el mismo motivo.

Al no tener integrada la red de nueva generación no cuenta con las funciones de control y de entrega. Tampoco permite la integración con el usuario NGN y las funciones de las interfaces RAC y NAS.

Es por este motivo por el que se da un paso más y se integra la red de nueva generación a la arquitectura de IPTV.

2.1.2- Arquitectura basada en NGN

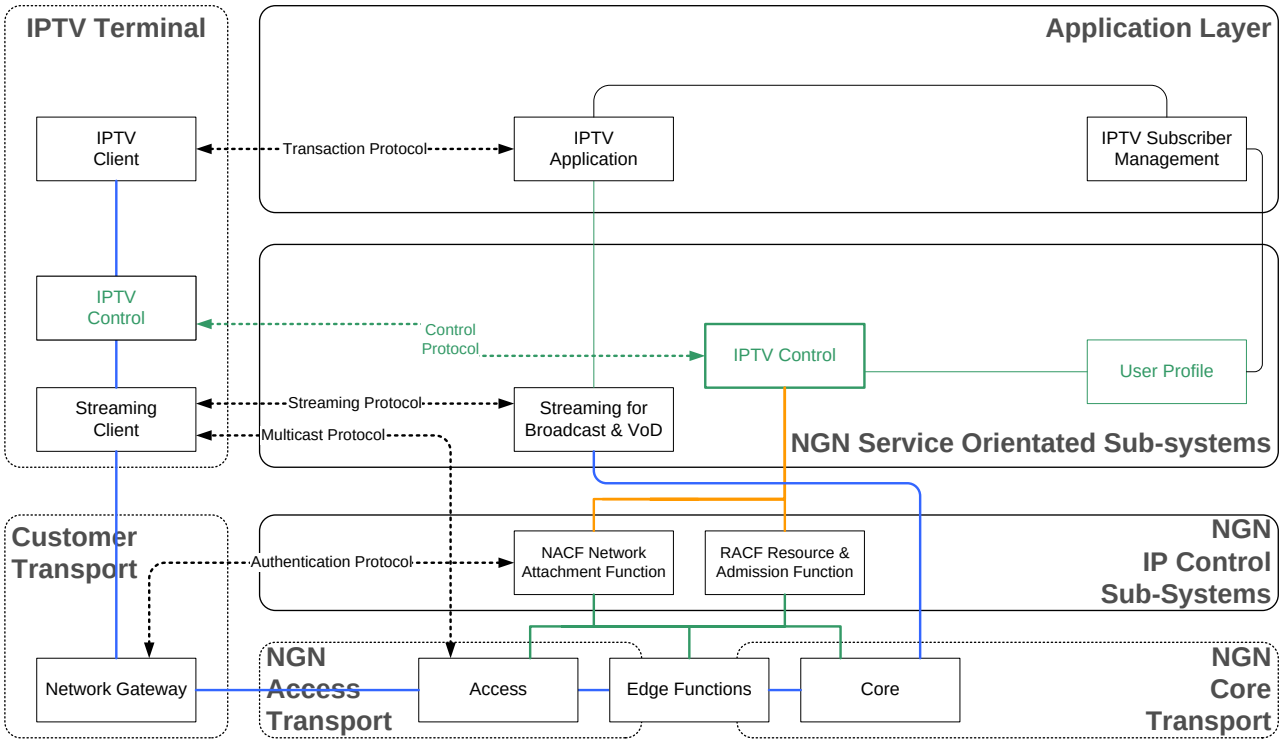


Figura 2: Arquitectura IPTV basada en NGN [12].

La diferencia que existe entre esta arquitectura y la evolución de ésta es la implementación de IMS. Dicho protocolo se utiliza para la autenticación, autorización y gestión de un cliente en la aplicación IPTV. En esta arquitectura el propio terminal IPTV realiza el control del usuario mediante su propio protocolo.

El hecho de la implementación de IMS en la arquitectura IPTV se debe al desarrollo de dicho protocolo que cada vez es más importante para la implementación de los servicios futuros de NGN. Aunque no se puede esperar que todos los servicios NGN en el futuro estén basados en IMS solamente.

2.1.3- Arquitectura basada en IMS

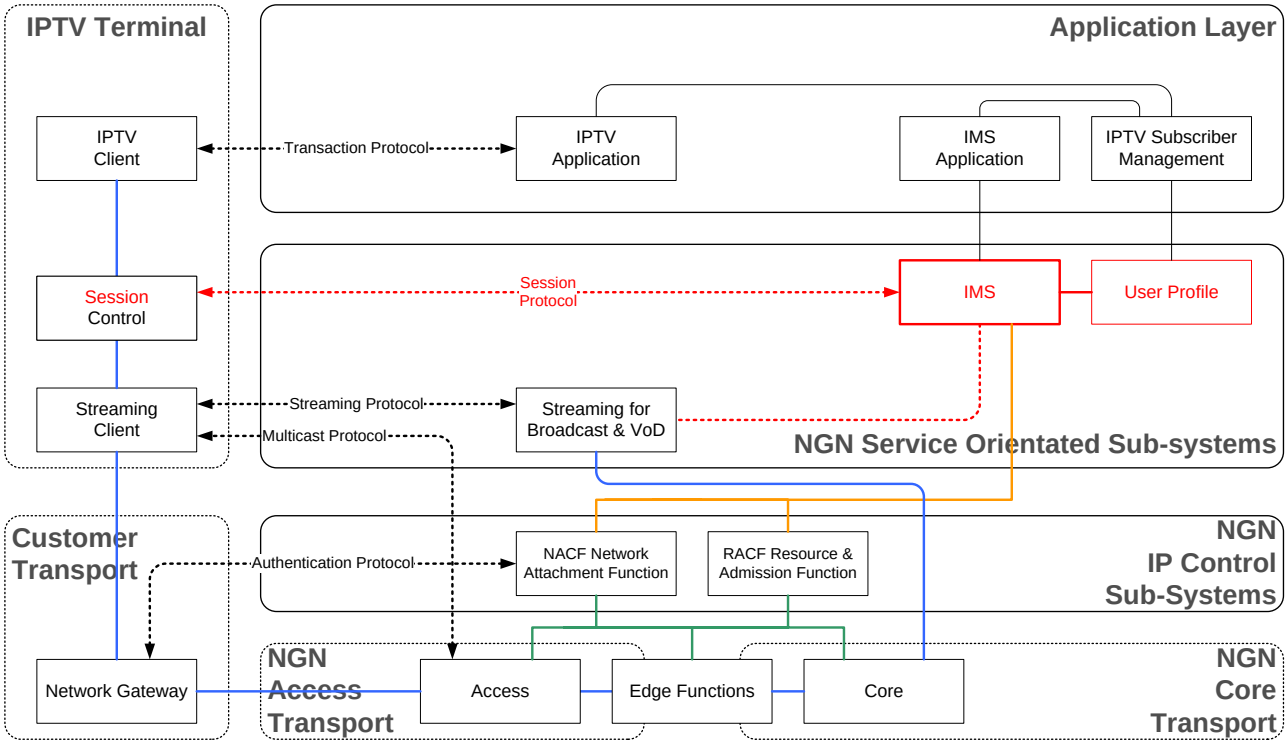


Figura 3: Arquitectura IPTV basada en IMS [12].

Esta arquitectura tiene numerosas ventajas. Entre ellas hay que destacar el apoyo a la movilidad, la interacción con los servicios permitidos de NGN, servicios de personalización y medios de comunicación. Además permite la integración de voz, vídeo, datos y servicios móviles como quadrule-play-services.

2.2- IMS

IMS [9] es un conjunto de especificaciones que comenzó con Release 5 3GPP de UMTS y siguió evolucionando en posteriores versiones.

IMS representa la convergencia a una arquitectura All-IP en 3G, donde se proporcionan servicios multimedia y se soporta telefonía a través de redes IP.

Se basa en el protocolo SIP para el establecimiento de sesiones. Dicho protocolo se explicará en el siguiente apartado.

Además implementa el plano de control para servicios IP multimedia. El plano de transporte esta implementado en GRPS o en cualquier otro tipo de acceso, por lo que los datos no atraviesan el CORE IMS.

Por otra parte, para el plano de servicios se combinan los elementos de ambos planos, el de control y el de transporte.

IMS posibilita un conjunto de servicios integrados, pero no define las aplicaciones o servicios que pueden ofertarse al usuario final, sino la infraestructura y los bloques que proporcionan el servicio.

Este protocolo tiene numerosas ventajas, entre las que hay que destacar:

- Proporciona QoS en sesiones multimedia. De este modo ofrece la posibilidad al operador de ofrecer distintos tipos de servicios a los usuarios.
- En cuanto a la seguridad, la lleva integrada en la arquitectura. Requiere autenticación con el cliente y define su propia arquitectura de servicio que es independiente de UMTS.
- Por parte del operador, éste tiene el control de la autenticación de los usuarios, autorización para acceder a los servicios y la tarificación. Además ofrece la posibilidad al operador de flexibilizar la forma de provisionar los servicios y controlarlos.

2.2.1- Arquitectura IMS

IMS tiene un conjunto de funcionalidades distintas. En este proyecto sólo vamos a comentar tres de ellas.

La funcionalidad de base de datos. El elemento principal es el Home Subscriber Server (HSS). Este elemento se encarga de almacenar la información del usuario. La información del perfil del abonado e información de subscripción, información de seguridad y/o autenticación del cliente y el estado del usuario. Utiliza el protocolo DIAMETER para comunicarse con el resto de elementos de la arquitectura.

En la siguiente figura sólo se muestra el elemento de control de la red IMS, CSCF (*Call Session Control Function*). Este elemento se encarga de procesar los mensajes de señalización para controlar la sesión multimedia de los usuarios.

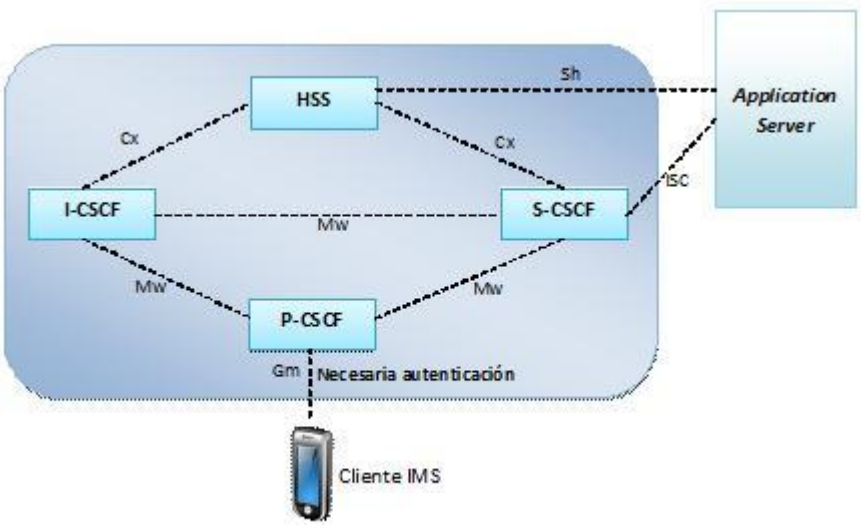


Figura 4: Arquitectura IMS.

Hay tres tipos de CSCF, los cuales provisionan diferentes funcionalidades:

- P-CSCF (Proxy CSCF): Es el primer servidor entre el terminal y la red IMS. Por él pasa toda la señalización SIP que los dispositivos se envían.
- I-CSCF (Interrogating CSCF): Es el punto de entrada a la red IMS desde redes externas. Selecciona y asigna un S-CSCF a un usuario con información del HSS. Para obtener el S-CSCF asignado y para la autorización inicial interactúa con el HSS a través de la interfaz DIAMETER Cx.
- S-CSCF (Serving CSCF): Proporciona el control de la sesión. Cada usuario es alojado dinámicamente en un S-CSCF durante el proceso de registro. Interactúa con el HSS a través de la interfaz DIAMETER Cx para obtener información de autenticación y el perfil del usuario. También interactúa con los *Application Server* (AS) para provisionar los servicios.

Por último mencionar los servidores de aplicaciones. Se encargan de proporcionar los servicios a los usuarios. No pertenecen al CORE IMS. Interactúan con el elemento S-CSCF para participar en el control de la sesión utilizando señalización SIP a través de la interfaz ISC.

También se comunica con el HSS por la interfaz DIAMETER Sh para descargar los perfiles de usuarios y cambiar el estado de los usuarios que se encuentran almacenados.

Opcionalmente pueden tener una interfaz con el usuario (Ut, XCAP/XML) para la configuración y gestión del servicio.

2.3- SIP

SIP [1] es un protocolo de señalización. Se utiliza para el establecimiento, finalización y modificación de sesiones multimedia y para el registro de clientes. También permite enviar y recibir mensajes y servicios de presencia. Por último ofrece movilidad, motivo por el que se va a utilizar este protocolo para la implementación del proyecto.

Sin embargo, este protocolo no permite la descripción de sesiones multimedia por lo que es necesario utilizar el protocolo SDP (*Session Description Protocol*). Tampoco participa de ningún modo en la transmisión de la información multimedia, por ello utiliza en conjunción los protocolos RTP (*Real-time Transport Protocol*), RTCP (RTP Control Protocol) y RTSP (*Real Time Streaming Protocol*).

En cuanto al direccionamiento, utiliza el concepto de URI (*Uniform Resource Identifier*) para identificar a los usuarios, servidores, proxies, etc.

El formato del URI es: *sip:[userinfo]hostport[parameters]*

- userinfo*:* almacena información del usuario, seguida de @.
- hostport*:* contiene el nombre del dominio o una dirección IP. Puede incluir el número del puerto.
- parameters: parámetros adicionales que se indican tras el carácter *;*.

2.3.1- Clientes y Servidores SIP

Utiliza el modelo de petición-respuesta similar al de HTTP o al de SMTP. A continuación se describen los distintos tipos de servidores y clientes SIP.

2.3.1.1- User Agents

Es un punto inicial o final de la sesión que actúa como cliente o servidor, dependiendo sí inicia o finaliza la sesión. En el *User Agent* (UA) se puede distinguir:

- Un UAC (*User Agent Client*) es una aplicación cliente que crea peticiones SIP.
- Un UAS (*User Agent Server*) es una aplicación que interactúa con el usuario cuando se recibe una petición SIP, y responde.

2.3.1.2- Back-To-Back User Agents (B2BUA)

Implementa por un lado un servidor y, por el otro, un cliente. Por tanto se comporta como un UAS y un UAC. Permite tener un control total sobre la transacción SIP.

2.3.1.3- Servidores SIP

No hay que confundir un servidor con un UAS. Un servidor es una entidad lógica, distinta al User Agent, que recibe una petición SIP de un User Agent y le responde.

2.3.1.3.1- Proxy

Se encarga de enrutar los mensajes SIP hasta el usuario destino. La respuesta sigue el mismo camino (a la inversa) que siguió el *request* original. Pero no sólo actúa como encaminador del paquete sino que tiene capacidad para interpretar, validar, redirigir, duplicar o eliminar estos mensajes.

Hay dos tipos principales de servidores proxy:

- Stateless proxies: actúan sin tener en cuenta el contexto de la transacción el que se encuadran los mensajes que procesan.
- Stateful proxies: almacenan la información sobre la transacción a la que pertenecen los mensajes y son capaces de actuar en consecuencia, como por ejemplo el P-CSCF y el S-CSCF. Ambos ejercen como servidores proxy con estado en el establecimiento, liberación y transferencia de sesiones.

2.3.1.3.2- Redirect

Recibe peticiones de clientes o servidores SIP, obtiene la asociación entre la sip-uri del usuario que se intenta contactar y sus direcciones de contacto actualizadas, para devolver estas direcciones de contacto al peticionario.

Estos servidores siempre utilizan mensajes de tipo 3xx (Redirección, utilizada para redirigir las peticiones a otra dirección).

2.3.1.3.3- Registrar

Gestiona la información de localización de usuarios en un determinado dominio. Los usuarios pueden modificar o borrar la asociación entre su sip-uri y las direcciones en las que desean ser contactados en un momento dado.

2.3.2- Mensajes SIP

Los mensajes SIP son legibles y tienen un formato único y genérico.

<start-line>
Cabeceras de SIP
Línea en blanco
<cuerpo-del-mensaje>

Lo único que difiere es la primera línea dependiendo del tipo de mensaje. Existen dos tipos de mensajes: requests y responses.
Para el caso de request el <start-line> tiene el siguiente formato:

<method><request-uri><SIP-version>

- method: propósito del request, por ejemplo: invite, register, refer, bye, ack, cancel…
- request-uri: URI del destinatario.
- SIP-version: versión del protocolo.

En cuanto a los mensajes responses se tiene:

<SIP-version><status-code><reason-phrase>

- status-code: código de tres dígitos que indica el resultado de comprender el request.
- Reason-phrase: explicación corta del status-code.

2.3.2.1- Principales métodos SIP

A continuación se muestra una lista con los métodos más utilizados en SIP [1]:

- INVITE: Esta petición se utiliza para el establecimiento de una sesión multimedia.
- ACK: Es una petición que no necesita respuesta. Es la respuesta final del INVITE, confirma que se ha recibido una respuesta final del mismo.
- BYE: Finaliza una sesión multimedia.
- CANCEL: Cuando todavía no se ha establecido la sesión, se puede enviar este mensaje indicando que se desea cancelar la sesión en curso.
- REGISTER: Registra a un usuario, informa de la localización actual del usuario.
- REFER: Se utiliza para que un UA pida a otro UA acceder a una URI o a un recurso. El ejemplo más común es la transferencia de sesiones entre usuarios.
- NOTIFY: Utilizado para notificar la aparición de nuevos eventos o de algún cambio que se haya producido. En el caso de este proyecto, informa del estado de la transferencia de la sesión.

2.3.3- Operaciones principales en SIP

2.3.3.1- Registro de un usuario SIP

El registro crea asociaciones en un servicio de localización existente en un dominio. Se tiene que enviar un *request* REGISTER a un servidor de tipo Registrar.

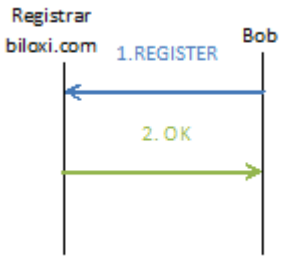


Figura 5: Registro de un usuario.

1. **REGISTER**

REGISTER sip:biloxi.com SIP/2.0

Via: SIP/2.0/UDP bobspc.biloxi.com:5060;branch=zahdsf54965fds5sd

Max-Forwards: 70

To: Bob <sip:bob@biloxi.com>

From: Bob <sip:bob@biloxi.com>;tag=45236

Call-ID: 465dsf589dsgf265879d

CSeq: 25 REGISTER

Contact: sip:bob@192.0.2.3

En el *request uri* aparece el dominio en el que se desea realizar el registro. La cabecera *contact* informa de en qué dirección se encuentra el usuario. Por lo que cada asociación enlaza una sip-uri pública con una o más direcciones de contacto.

2. **OK**

SIP/2.0 200 OK

Via: SIP/2.0/UDP bobspc.biloxi.com:5060;branch=zahdsf54965fds5sd;receive=192.0.2.3

To: Bob <sip:bob@biloxi.com>;tag=5236982

From: Bob <sip:bob@biloxi.com>;tag=45236

Call-ID: 465dsf589dsgf265879d

CSeq: 25 REGISTER

Contact: <sip:bob@192.0.2.3>

Expires: 7200

2.3.3.2- Establecimiento de una sesión

Para establecer una sesión se tiene que enviar un mensaje INVITE. Este *request* se encamina, a través de uno a más *proxies*, hasta alcanzar el destino.

Una vez que el mensaje llega a su destino, éste puede:

- Aceptar la sesión mediante una respuesta 2xx.
- Rechazar la sesión mediante una respuesta 3xx, 4xx, 5xx o 6xx.
- Enviar respuestas provisionales para informar al origen que está en progreso el establecimiento de la sesión mediante una respuesta 1xx.

Figura 6: Establecimiento de sesión.

1. INVITE

INVITE sip:bob@atlanta.com SIP/2.0

Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9dG5D568cdser

Max-Forwards: 70

To: Bob <sip:bob@atlanta.com>

From: Alice <sip:alice@atlanta.com>;**tag=123689**

Call-ID: s45e2fd59gf

CSeq: 235 INVITE

Contact: <sip:alice@pc33.atlanta.com >

6. OK

SIP/2.0 200 OK

Via: SIP/2.0/UDP atlanta.com;branch=z9dG5D3dgre697;received=192.0.2.2

Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9dG5D568cdser;received=192.0.2.1

To: Bob <sip:bob@atlanta.com>;**tag=452368**

From: Alice <sip:alice@pc33.atlanta.com>;**tag=123689**

Call-ID: s45e2fd59gf

CSeq: 235 INVITE

Contact: <sip:bob@192.0.2.3>

8. ACK

ACKsip:bob@192.0.2.3 SIP/2.0

Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9dG5D568cdser

Max-Forwards: 70

To: Bob <sip:bob@atlanta.com>;**tag=452368**

From: Alice <sip:alice@atlanta.com>;**tag=123689**

Call-ID: s45e2fd59gf

CSeq: 235 ACK

Una vez que se ha establecido la sesión, se debe mantener para los futuros mensajes *requests* los correspondientes *tags* del *From* y del *To* y el valor de la cabecera Call-ID. En el *request-uri* se suele poner la *uri* del campo contact. De este modo se consigue que los mensajes no atraviesen los diferentes proxies.

9. BYE

BYE sip:alice@pc33.atlanta.com SIP/2.0

Via: SIP/2.0/UDP 192.0.2.3:branch=z9dG5fjrio425d

From: Bob <sip:bob@atlanta.com>;**tag=452368**

To: Alice <sip:alice@atlanta.com>;**tag=123689**

Call-ID: s45e2fd59gf

CSeq: 452 BYE

2.4- Movilidad SIP

Debido a la gran cantidad de dispositivos que tienen los usuarios es necesario aportar movilidad a las sesiones establecidas, independientemente de la tecnología subyacente. Se pretende ofrecer algo más que *hand-off* entre las estaciones bases y las subredes, así como compensar la falta de despliegue de IP en el terminal móvil [5]. El protocolo de señalización que se utiliza es SIP. Esto se debe a que, como se mencionó anteriormente, este protocolo permite establecer sesiones multimedia entre dos o más participantes.

Existen cuatro tipos de movilidad: movilidad de terminal, movilidad de sesión, movilidad personal y movilidad de servicio. Este proyecto se centra en la implementación de la movilidad de sesión, por lo que los restantes tipos de movilidad se mencionaran con menor detalle.

2.4.1- Movilidad de Terminal

Hace posible que el terminal se cambie de subred, misma tecnología o distinta, sin que le afecte a las sesiones activas. Por ejemplo, un cliente tiene establecida una sesión desde su ordenador con otro usuario. Dicho cliente está conectado mediante una conexión Ethernet. En cierto momento de la sesión, el cliente se cambia a una conexión Wi-Fi (por cualquier motivo) y la sesión activa se sigue manteniendo.

2.4.2- Movilidad Personal

Permite a un usuario estar localizado en diferentes terminales mediante la misma dirección lógica. Por ejemplo, un usuario quiere ser contactado mediante un teléfono fijo, un PC y un teléfono móvil. Este usuario puede utilizar estos dispositivos al mismo tiempo o alternando entre ellos. Además, puede ser contactado por cualquier dispositivo usando el mismo nombre gracias a SIP forking proxies.

2.4.3- Movilidad de Servicio

Ofrece la posibilidad a los usuarios de mantener el acceso a sus servicios mientras se mueven o cambian de dispositivo o de red.

Para ello, SIP ofrece un mecanismo de sincronización a través de servidores. De este modo cada cierto tiempo el usuario se vuelve a registrar o, cuando cambie la dirección de red, informa de propiedades del dispositivo gracias a la cabecera *contact*.

2.4.4- Movilidad de Sesión

Se denomina movilidad de sesión a la transferencia de una sesión multimedia activa entre dos dispositivos diferentes a otro terminal sin que la comunicación se vea afectada.

Para poder transferir la sesión es necesario un marco en el que el terminal pueda descubrir otros dispositivos disponibles y de este modo incluirlos en una sesión activa. Una vez que conozca los dispositivos disponibles será capaz de transferir una sesión a otro terminal.

Para poder llevar a cabo la movilidad de sesión, debe de cumplirse una serie de condiciones. Una de ellas es que cuando se produce dicha transferencia, en el terminal remoto no puede aparecer como una nueva sesión y que la interrupción de los datos sea mínima.

Además el terminal que inicia la movilidad debe de tener funcionalidades mejoradas para poder soportar dicha movilidad.

Por último hay que tener en cuenta las diferencias entre los dispositivos móviles, como puede ser la resolución de la pantalla o el ancho de banda.

Hay diversas propuestas de cómo realizar la movilidad de sesiones en SIP. Una de las más conocidas es la presentada en [7] que introduce dos formas de movilidad de sesión: Mobile Node Control (MNC) y Session Handoff (SH).

2.4.4.1- MNC

El Mobile Node (MN) actúa como un tercer interlocutor encargado de controlar la transferencia de sesión. Éste se encargará de establecer la sesión con cada uno de los diferentes dispositivos y a continuación actualizar la sesión con el nodo remoto. El gran inconveniente de este modo es que el MN debe de estar activo para poder mantener las sesiones.

Además se puede transferir la sesión no sólo a un único dispositivo, sino a varios. A continuación se muestra el caso de transferir la sesión a un dispositivo pero se explica también el otro caso.

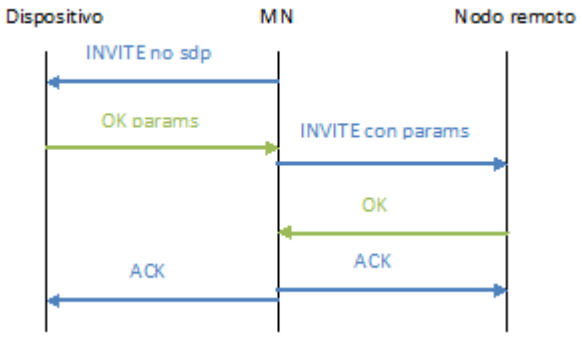


Figura 7: Modo MNC para la transferencia a un dispositivo.

Primero se establece una sesión entre el MN y el nodo remoto. A continuación el MN inicia la movilidad de sesión, para ello informa al dispositivo de que se va a establecer una sesión, pero el mensaje SDP está vacío. Será en la respuesta del dispositivo donde incluya el mensaje SDP indicando el puerto donde va a estar escuchando o los codecs que se van a utilizar. Una vez que el MN tiene esta información, actualiza la sesión con el nodo remoto enviando un INVITE con los parámetros de la respuesta.

Para el caso de múltiples dispositivos se sigue el mismo procedimiento que para uno solo. Ahora el MN primero deberá iniciar la sesión con los distintos dispositivos antes de actualizarla con el nodo remoto.

2.4.4.2- SH

En este modo se dispone del mismo escenario que en el caso anterior; un dispositivo para transferir, un MN y un nodo remoto. Pero ahora el MN no actúa como intermediario entre los dos extremos sino como un dispositivo más de la comunicación.

Para realizar el modo SH es necesario el método REFER, con este método se invita al dispositivo a una sesión con el nodo remoto.

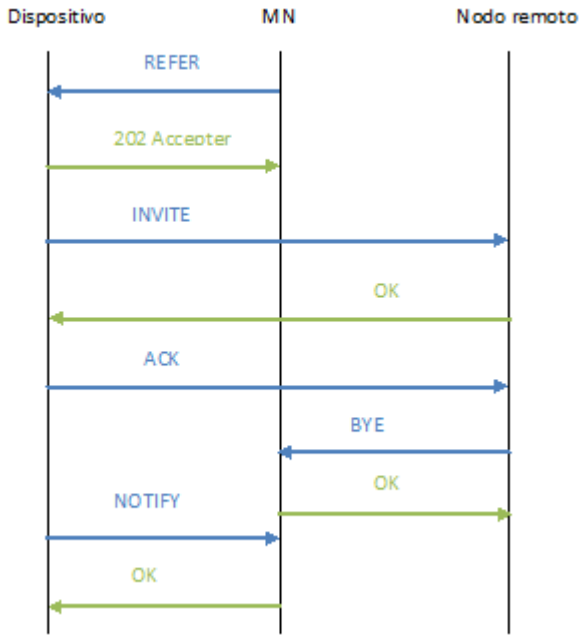


Figura 8: Modo SH para transferir una sesión a un dispositivo.

Como se observa en la figura anterior, una vez que se ha establecido la sesión entre el nodo remoto y el dispositivo, se libera la sesión que había entre el nodo remoto y el MN. Además el dispositivo debe de notificar al MN que ha establecido una sesión con el nodo remoto.

Para este caso resulta más complicado realizar la movilidad de sesión a varios dispositivos, con lo que actualmente sólo existe la posibilidad de transferir una sesión a un único dispositivo.

Capítulo 3

Solución a la movilidad de sesión basada en SIP

El objetivo de este proyecto es, como se mencionó en el capítulo 1, permitir la movilidad de sesiones multimedia mediante los modos pull y push.

A continuación se muestran los elementos que componen la solución:

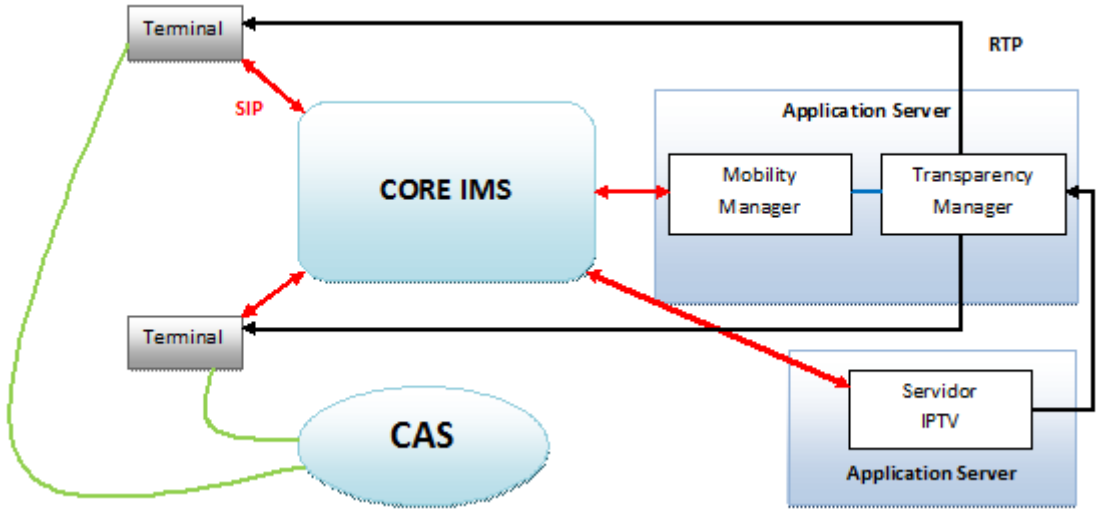


Figura 9: Elementos de la solución.

En el escenario se tienen una serie de terminales, que serán uno de los extremos de la comunicación. El otro extremo de la comunicación es el Servidor IPTV.

Los terminales de la comunicación tienen que tener la capacidad de movilidad, ya que serán los encargados de iniciar la transferencia de sesión. Sin embargo, el Servidor IPTV no tiene que soportar la movilidad.

Sólo falta por mencionar tres elementos. Por un lado está el CORE IMS, que se explicó en el capítulo anterior las características de sus elementos. Por otro lado se tiene el CAS, Context-Awareness System, y otro *Application Server* (AS).

Para poder transferir la sesión, el último AS mencionado está compuesto además por el Mobility Manager y el Transparency Manager. El conjunto de estos dos elementos se denomina Service Continuity Agent. Será este servidor el que realice el servicio de movilidad de sesiones multimedia, de tal forma que resulte dicha movilidad transparente para el extremo de la sesión sin movilidad (Servidor IPTV). El Mobility Manager se encarga de procesar la señalización SIP que se envían los dos extremos de la comunicación, mientras que el Transparency Manager se centra en la gestión del flujo de datos de los usuarios.

Por último mencionar el CAS, Context-Awareness System. Este elemento se encarga de interactuar con los terminales con movilidad para almacenar o consultar información de los usuarios registrados o de las sesiones establecidas.

3.1- Movilidad de sesión

Con este proyecto se pretende poder transferir una sesión multimedia establecida entre un dispositivo móvil y el Servidor IPTV a otro terminal móvil, sin finalizar dicha sesión o estableciendo una nueva.

Para realizar esta movilidad se utilizan dos modos diferentes, modo pull y modo push.

- **Modo pull:** El usuario tiene establecida una comunicación con un Servidor IPTV. Este usuario decide transferir la comunicación a otro de sus dispositivos; será este dispositivo el que inicie la transferencia por medio de un mensaje INVITE con la cabecera *replace*. Gracias al AS, el Servidor IPTV no se entera del cambio de terminales.
- **Modo push:** En este caso, será el terminal donde está establecida la sesión con el Servidor IPTV, el que inicie la transferencia de la comunicación por medio de un mensaje REFER. En la cabecera *refer-to* de dicho mensaje se indica el dispositivo al que se desea transferir la sesión. Como ocurría en el caso pull el Servidor IPTV permanece ajeno a la movilidad de sesión gracias al Application Server.

Para los dos tipos de movilidad es necesario que los terminales interactúen con el CAS. En el primer caso es necesario conocer la información de la sesión establecida, concretamente la cabecera *call-ID* del INVITE que inició la sesión y los tags procedentes de las cabeceras *from* y *to* del ACK, ya que son estos tres datos los que se ponen en la cabecera *replace* del nuevo INVITE.

Para el caso del push, es necesario tener la información de la sip-uri al que se quiere realizar la movilidad. Esta sip-uri es la que tiene que figurar en la cabecera *refer-to* del mensaje REFER.

3.2- Application Server

Este proyecto se basa en la implementación, por parte del cliente, de la movilidad de sesión. La parte del AS ha sido implementado en otro proyecto, sin embargo es necesario explicar su función para entender los elementos que intervienen en la comunicación.

En el proyecto se va a utilizar un Servidor IPTV como aplicación. Para poder realizar la movilidad es necesario tener otro *Application Server* distinto, al que se llamará por IPTV Continuity Agent. Gracias al IPTV Continuity Agent, el Servidor IPTV no tiene constancia del cambio de terminales móviles durante la sesión multimedia.

En la siguiente imagen se muestran los distintos bloques del IPTV Continuity Agent y sus interfaces de comunicación.

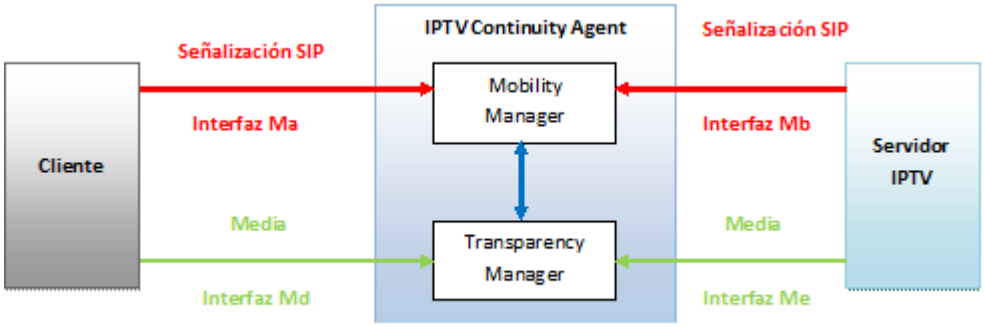


Figura 10: Componentes del IPTV Continuity Agent.

3.2.1- Mobility Manager

El Mobility Manager se encarga de gestionar la señalización SIP y de este modo proporcionar la movilidad de sesión. Actúa como un Back-To-Back-UserAgent de SIP, para aportar la transparencia de sesiones multimedia al Servidor IPTV.

Además controla al Transparency Manager en el intercambio de los datos.

3.2.2- Transparency Manager

Con la ayuda del Mobility Manager se gestiona el flujo de datos entre el terminal y el Servidor IPTV, de tal manera que la movilidad sea transparente para el extremo del IPTV.

3.3- Señalización concreta para los distintos tipos de transferencia de sesión

A continuación se explica detalladamente los mensajes de señalización que se deben enviar para realizar el modo pull y el modo push.

3.3.1- Modo pull

Para explicar la señalización se utiliza el siguiente escenario: dos terminales móviles, un Servidor IPTV, un IPTV Continuity Agent y el CAS. El CAS almacena la información del usuario o de la sesión. El Servidor IPTV actúa como el elemento fijo de la sesión; serán los dos dispositivos los que transfieran la sesión. El IPTV Continuity Agent permite que la transferencia de sesión multimedia sea transparente al Servidor IPTV.

El terminal 1 inicia una sesión con un Servidor IPTV. Una vez establecida la sesión, el terminal 2 transfiere dicha sesión a su terminal obteniendo la información necesaria del CAS.

A continuación se muestran los mensajes SIP necesarios para este tipo de movilidad.

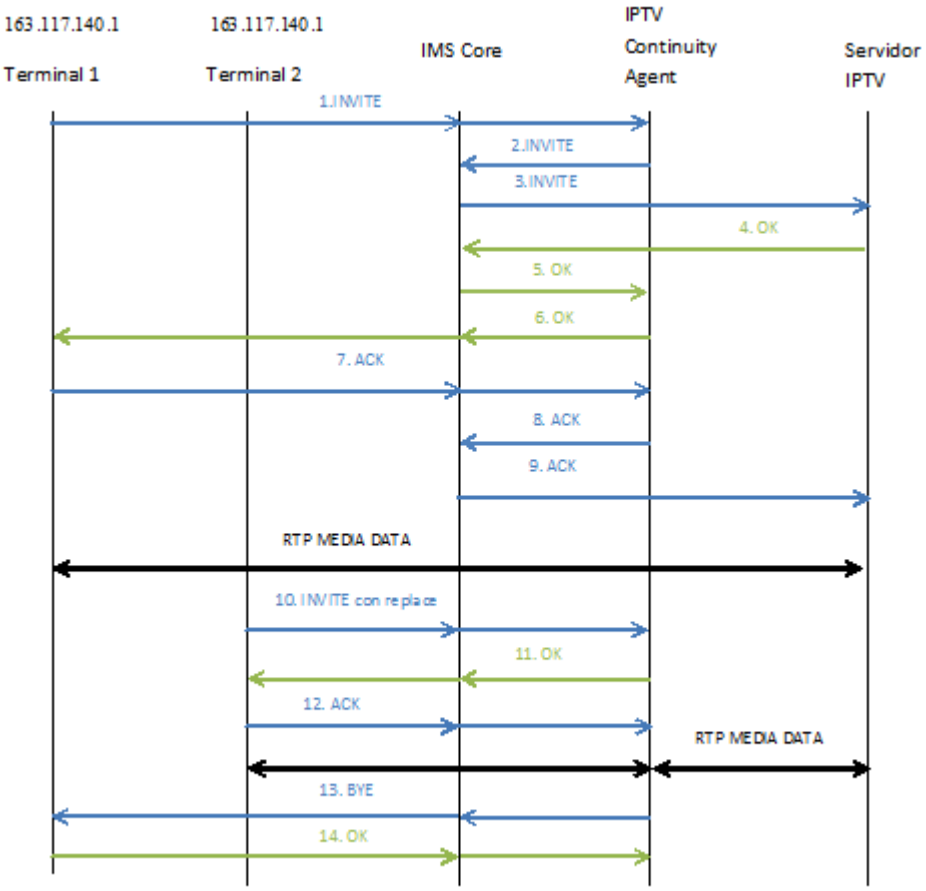


Figura 11: Señalización para el modo pull.

Primero se establece la sesión entre el terminal 1 y el Servidor IPTV mediante la secuencia de mensajes correspondientes.

1. INVITE

INVITE sip:iptv_platform@open-ims.test SIP/2.0

Via: SIP/2.0/UDP 163.117.140.1:5060;branch=f478gvdfgg45df

Max-Forwards: 70

To: <sip:iptv_platform@open-ims.test>

From: <sip:terminal1@open-ims.test>;**tag=452359**

Call-ID: 548sdf21gr89erg3

Contract: <sip:terminal1@163.117.140.1>

CSeq: 4558 INVITE

4. **OK**

SIP/2.0 200 OK

Via: SIP/2.0/UDP 1163.117.140.1:5060;branch=f478gvdfgg45df

Max-Forwards: 70

To: <sip:iptv_platform@open-ims.test>; **tag=74523**

From: <sip:terminal1@open-ims.test>;**tag=452359**

Call-ID: 548sdf21gr89erg3

Contract: <sip:mobilitymanager@163.117.140.36>

CSeq: 4558 INVITE

Una vez que la sesión ha sido iniciada, el terminal 2 realiza la movilidad de sesión enviando un mensaje INVITE pero añadiendo una cabecera nueva, *Replace*, que contiene la información de la sesión. La información necesaria es el Call-ID de la sesión que se quiere transferir y los tags de la cabecera From y To. El dispositivo 2 obtiene todos estos datos del CAS.

10. **INVITE**

INVITE sip:iptv_platform@open-ims.test SIP/2.0

Via: SIP/2.0/UDP 163.117.140.2:5060;branch=d7f5d25fes

Max-Forwards: 70

To: <sip:iptv_platform@open-ims.test>

From: <sip:terminal2@open-ims.test>;tag=120348

Call-ID: f415gfths4ds4

Replace: 548sdf21gr89erg3;from-tag=452359;to-tag=tag=74523

Contact: <sip:terminal2@163.117.140.2>

CSeq: 895 INVITE

Al recibir el mensaje número 12 (12. ACK) la sesión ya ha sido transferida al segundo terminal. El siguiente paso es finalizar la sesión con el primer terminal.

Hay que tener presente que este proyecto sólo se encarga de la parte de señalización entre el terminal y el IMS CORE. La parte del IPTV Continuity Agent y la del Servidor IPTV han sido implementadas en otro proyecto.

3.3.2- Modo push

Se continúa con el mismo escenario anterior, pero ahora es el terminal 1 él que inicia la transferencia de sesión una vez que la sesión se ha establecido con el Servidor IPTV.

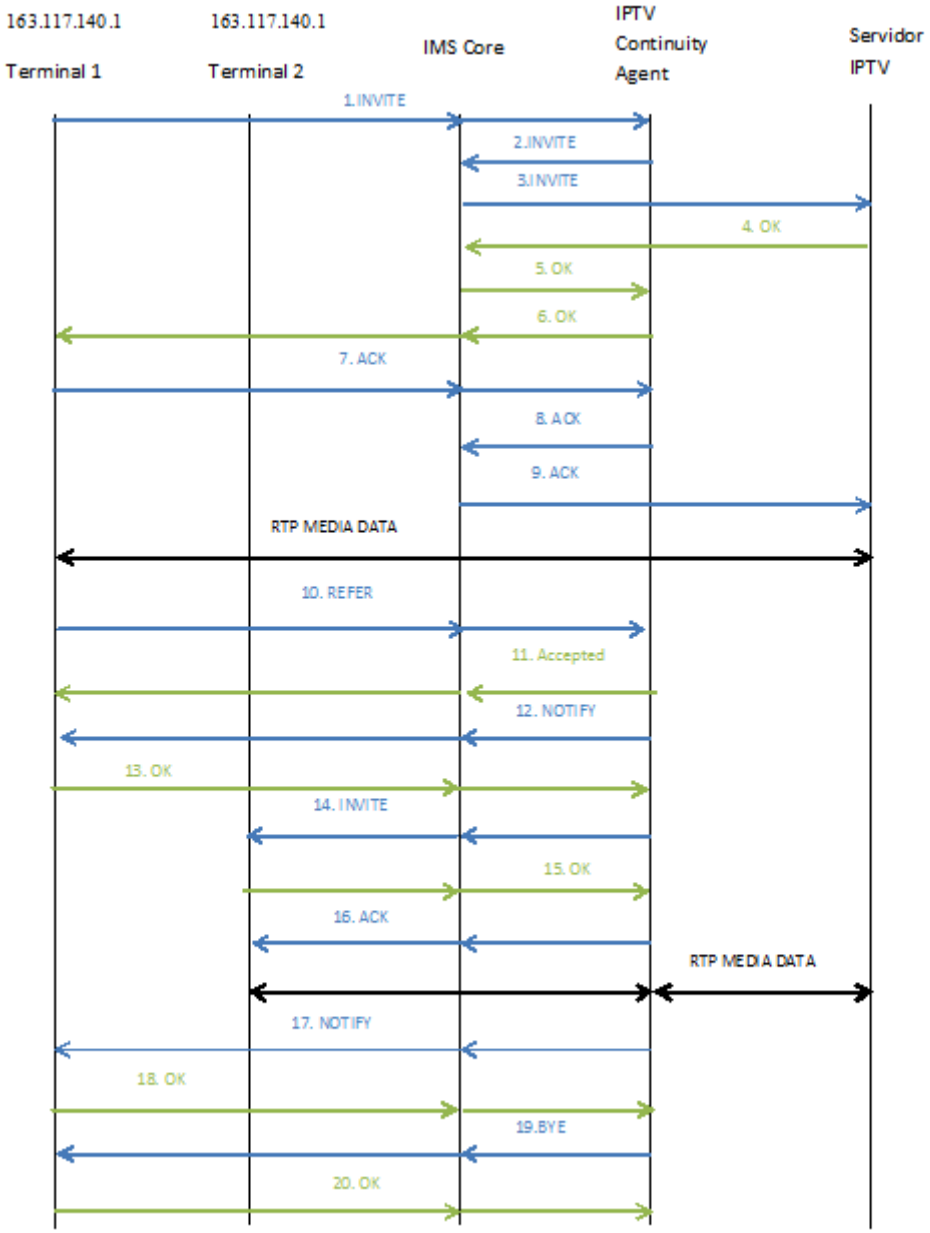


Figura 12: Señalización para el modo push.

10. REFER

REFER sip:iptv_continuity_agent@163.117.140.131 SIP/2.0

Via: SIP/2.0/UDP 163.117.140.2:5060;branch=d7f5d25fes

Max-Forwards: 70

To: <sip:iptv_platform@open-ims.test>

From: <sip:terminal1@open-ims.test>;tag=59863

Call-ID: easd452df1dgt45et

Contract: <sip:terminal1@163.117.140.1>

CSeq: 52 REFER

Refer-To: <sip:terminal2@open-ims.test>

Referred-By: <sip:terminal1@open-ims.test>

Con el REFER se indica a quién se desea transferir la sesión mediante la cabecera *refer-to* y de quién procede el mensaje, *referred-by*. Para conseguir esta información es necesario hacer una consulta al CAS. Será el CAS el que informe de los posibles dispositivos existentes para transferir la sesión.

Seguidamente, el IPTV Continuity Agent envía una notificación al terminal 1 para corroborar que le ha llegado el REFER (8 NOTIFY).

Posteriormente será el IPTV Continuity Agent el que envíe el INVITE al terminal 2 para poder establecer la sesión multimedia.

Una vez que se ha establecido la sesión entre el Servidor IPTV y el terminal 2, el IPTV Continuity Agent notifica al terminal 1 que toda ha salido satisfactoriamente (13 NOTIFY). De esta forma, sólo queda finalizar la sesión con el terminal 1.

Como ocurre para el modo pull, el Servidor IPTV no tiene conocimiento del cambio de dispositivo gracias a la colaboración del IPTV Continuity Agent.

Sin embargo, puede ocurrir que un usuario esté registrado con la misma sip-uri en distintos dispositivos, por lo que al enviar el REFER, el IPTV Continuity Agent remitirá el INVITE a todos los dispositivos que se encuentren registrados con la sip-uri de la cabecera refer-to.

Para evitar realizar forking se utilizan las GRUUs (Global Routable User Agent URIs).

3.3.2.1- GRUUs

Se utilizan los GRUUs para identificar de manera única a un usuario registrado. Es decir, en el caso de que un abonado esté registrado en varios dispositivos con la misma sip-uri, poder diferenciarlos. De este modo si se toma una llamada, sólo se recibirá en el dispositivo con el GRUU indicado y no en todos los terminales.

Un GRUU contiene dos propiedades:

- Se dirige a una instancia específica del User Agent.
- Debe ser posible que un GRUU sea invocado desde cualquier lugar de Internet.

Hay dos tipos diferentes de GRUUs:

- GRUU pública. Se añade a la sip-uri unos parámetros grs, que contiene la representación de la instancia del User Agent. Por ejemplo: *sip:alice@example.com;gr=f45d2gf3sdf85*
- GRUU temporal. Se utiliza para proteger la privacidad del usuario. El GRUU tendrá el parámetro gr ya sea con valor o no. Por ejemplo: *sip:ds632fda953@example.com;gr*

Una manera simple de obtener un GRUU es a través de la petición REGISTER. El usuario añade a la cabecera *contact* la capacidad *+sip.instance,* que identifica de manera única una instancia User Agent. Por ejemplo:

Contact: <sip:alice@163.117.140.226>;

+sip.instance=<45ds35f78g-456s-sdgd-dfgs56fg>

El servidor registrar procesa la petición y dispone de dos GRUUs en la respuesta del registro. Uno de ellos es un GRUU temporal y el otro es público. El primero se devuelve en el parámetro de la cabecera contact *temp-gruu* y el segundo en *pub-gruu*.

Contact: sip:alice@163.117.140.226

;pub-gruu="sip:alice@example.com;gr=45ds35f78g-456s-sdgd-dfgs56fg";

temp-gruu="sip:fetgvdghrths@example.com;gr";

+sip.instance="45ds35f78g-456s-sdgd-dfgs56fg"

expires=3600

Volviendo al problema del envío del REFER, añadiendo en la cabecera refer-to, el GRUU asociado al terminal se consigue enviar el INVITE a un solo terminal.

Capítulo 4

Librería Doubango y aplicación IMSDroid

Este capítulo explica brevemente la aplicación que se ha utilizado para la realización del proyecto y la librería que usa. Se expondrá la estructura que sigue tanto la aplicación como la librería y se describirá el código más relevante.

4.1- Doubango

Doubango realiza proyectos de código abierto para redes 3GPP IMS/LTE. En sus proyectos, efectúa tanto la parte del cliente como la del servidor. Sus características se pueden ver en [17].

Actualmente tienen numerosos proyectos, entre ellos destaca IMSDroid. Una aplicación Android que actúa como un cliente SIP/IMS.

4.1.1- Introducción a la librería nativa de IMSDROID

IMSDroid sólo tiene una librería nativa, **tinyWRAP.so**, escrita en C++ y procedente de doubango. Los archivos se generan mediante JNISWIG. La arquitectura es la siguiente:

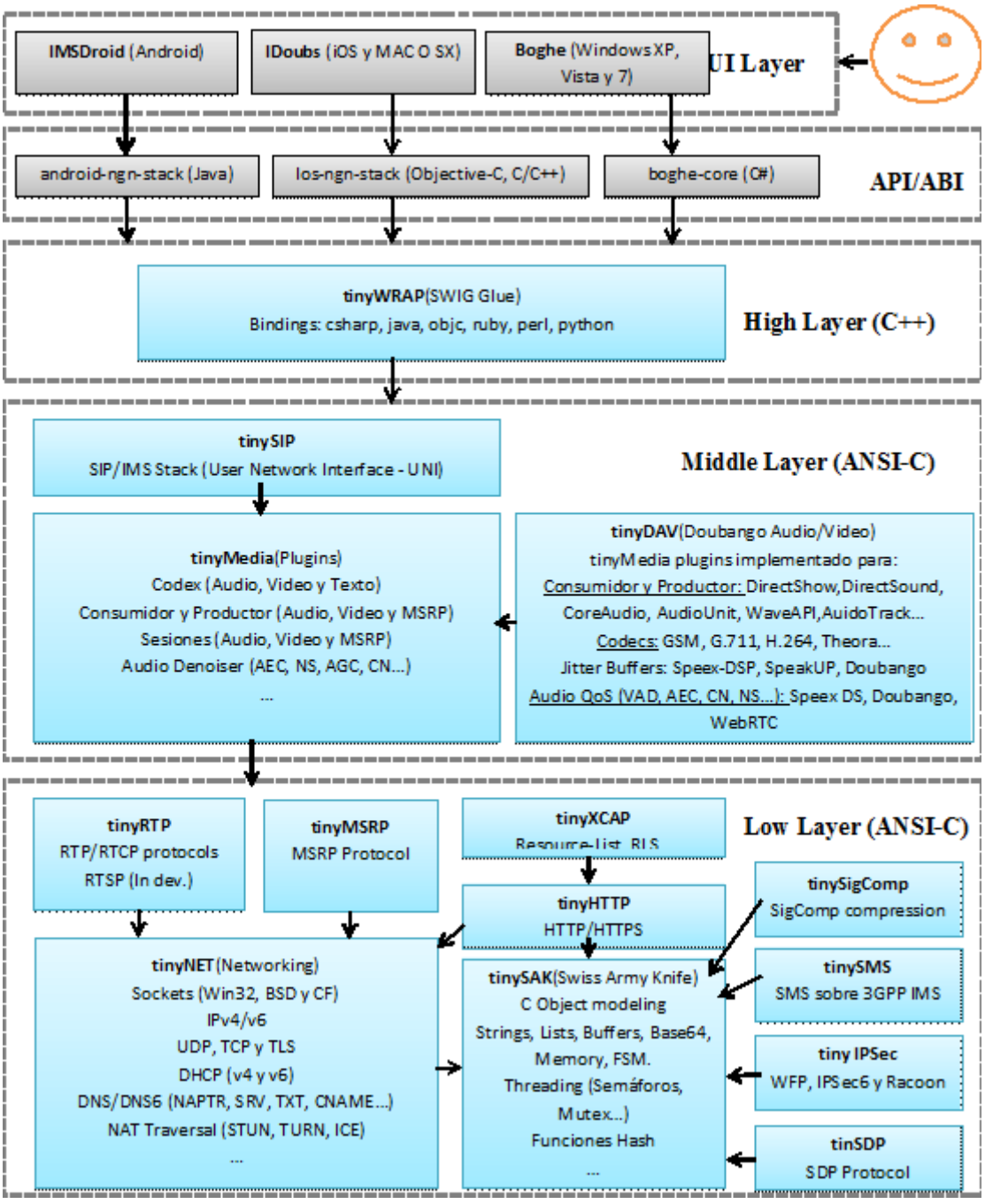


Figura 13: Arquitectura IMS de Doubango.

Además de contener todas las librerías de las que depende tinyWRAP, tiene otros dos directorios: *android-proyects* y *bindings*.

El primero de ellos contiene el fichero de configuración, *root.mk*, para compilar todos los proyectos y poder generar los archivos.

El otro directorio está compuesto por una serie de carpetas que hacen referencia a distintos lenguajes de programación. Aparte tiene una carpeta común, *_common*, a todos los lenguajes de programación y un fichero para generar el código que se ha modificado. En esta carpeta se encuentran las clases que posteriormente se incluirán en el API *android-ngn-stack*.

Entre todas las clases del directorio *_common* hay que destacar dos de ellas:

- SipMessage: En esta clase se encuentran los métodos necesarios para poder coger el valor de las cabeceras del mensaje SIP y del mensaje SDP.

- SipSession: Es sin duda la más importante por las clases que contiene.

  - SipSession: Se encarga de añadir o eliminar capacidades, añadir cabeceras al mensaje SIP, cambiar las cabeceras from y to, entre otras más funcionalidades.

  - InviteSession: Los métodos de esta clase realizan las funciones de aceptar o rechazar una comunicación, entre otras funciones.

  - CallSession: Realiza el establecimiento de una sesión del tipo que sea, audio o audio/video. Sus métodos más importantes son:

    - bool callAudio (const char* remoteUriString, ActionConfig* config=tsk_null): Establece una sesión audio, mandando un INVITE a la sip-uri que se le pasa como parámetro.

    - bool callAudioVideo (const char* remoteUriString, ActionConfig* config=tsk_null): Este método realiza la misma función que el anterior pero para una llamada de audio/video.

    - bool transfer (const char* referToUriString, ActionConfig* config=tsk_null): Es este método el que manda un mensaje REFER modificando el valor de la cabecera refer-to, valor del parámetro referToUriString.

4.1.2- TinySIP

Resulta interesante hablar de este directorio para entender un poco mejor el funcionamiento de la aplicación IMSDroid. Esta carpeta es la que se encarga de todas las funcionalidades del protocolo SIP.

TinySIP se estructura en una serie de carpetas y archivos, los más significativo se encuentran dentro de las carpetas *include* y *src*. En la carpeta *include* se hayan las librerías de los archivos, es decir, los ficheros con extensión .h. Es en el directorio *src* donde están las clases, es decir, los ficheros en C++.

Por su parte, la carpeta *src* contiene una serie de directorios y archivos entre los que hay que destacar los siguientes:

- El directorio api: Contiene una serie de archivos que definen la estructura y los tipos de eventos para cada acción. Además tiene un conjunto de

métodos capaces de enviar mensajes SIP. Un ejemplo de fichero es el *tsip_api_invite*. Para este caso existen los siguientes tipos de eventos: tsip_i_new_call, tsip_i_request, tsip_ao_request, tsip_o_ect_trying, tsip_m_early_media… Entre los métodos de la clase destaca:

- int tsip_invite_event_signal (tsip_invite_event_type_t type, tsip_ssession_handle_t* ss, short status_code, const char *phrase, const struct tsip_message_s* sipmessage): Este método crea una señal con el tipo de evento que se le pasa como parámetro.

- int tsip_api_invite_send_invite (const tsip_ssession_handle_t* ss, tmedia_type_t type,…): Este método crea la acción correspondiente al mensaje INVITE y el diálogo conveniente para mandarla posteriormente a la sesión.

- int tsip_api_invite_send_ect (const tsip_ssession_handle_t* ss, tmedia_type_t type,…): Este método se encarga de crear la acción correspondiente al mensaje REFER y enviarla a la sesión.

- El directorio dialogs: Esta carpeta contiene todos los tipos de diálogos (invite, info, options, registration,…) y los métodos necesarios capaces de enviar cualquier mensaje SIP, dependiendo del estado en que se encuentre la sesión. Por ejemplo enviar 100 TRYING después de enviar el INVITE.

- El directorio headers: Se encarga de crear la estructura de todas las cabeceras del mensaje SIP y los métodos precisos para crearlas.

- El archivo tsip_action: Define el conjunto de acciones del protocolo SIP, entre ellas destacan: tsip_atype_register (envía una petición SIP REGISTER), tsip_atype_invite (envía una petición SIP INVITE o reINVITE), tsip_atype_hold (pone una sesión en estado de espera), tsip_atype_ect (transfiere una sesión), tsip_atype_accept (acepta una sesión o un mensaje)… Además contiene una serie de métodos capaces de crear dichas acciones o de modificarlas entre otras cosas.

- El archivo tsip_event: Define los eventos del protocolo SIP y la estructura de éstos. Algunos de estos eventos son: tsip_event_invite, tsip_event_register, tsip_event_dialog, tsip_event_message… Además concreta los códigos de SIP a partir del 700 hasta el 900. Por otra parte, también detalla métodos capaces de inicializar los eventos o crear una señal asociada a ese evento.

- El archivo tsip_message: Define la estructura de un mensaje SIP con su línea de petición o de respuesta y sus cabeceras. Contiene un conjunto de métodos capaces de añadir una cabecera al mensaje o coger el valor de dicha cabecera, crear un nuevo mensaje SIP…

- El archivo tsip_session: Define la estructura de una sesión SIP; la lista de capacidades, la lista de cabeceras, la pila, la sip-uri del origen y del destino, la estructura media… También contiene los métodos necesarios para gestionar y manipular la sesión, es decir, para poder crearla, modificarla…

4.2- Aplicación IMSDroid

Como se ha mencionado anteriormente esta aplicación es un cliente SIP/IMS, que puede realizar llamadas tanto de audio como de audio/video utilizando el protocolo SIP. También puede enviar mensajes de texto, entre otras funcionalidades.

IMSDroid comenzó con una versión beta que sólo soportaba un codec de video y unos pocos codecs de audio. Poco a poco se han ido introduciendo nuevas capacidades a la aplicación hasta conseguir las características que se encuentran en [16].

El código de la aplicación IMSDroid está distribuido de la siguiente manera:

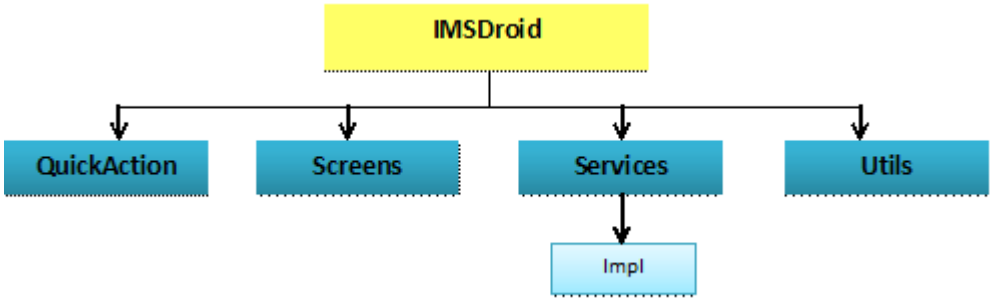


Figura 14: Esquema de la aplicación IMSDroid.

- Directorio IMSDroid: Además de contener cuatro paquetes, tiene una serie de ficheros que definen el motor de la aplicación.

- Directorio QuickAction: Se encarga de todo lo relacionado con los iconos que se muestran en las actividades. Poder añadir una imagen, texto o incluso aportarle una función pulsado el icono. Pero no sólo se centra en la manipulación del icono, sino en el conjunto de ellos y su apariencia en la pantalla del dispositivo.

- Directorio Screens: Es el más importante de la aplicación y del que más código se ha modificado o incluso añadido nuevos ficheros. Hay un fichero por cada actividad de la aplicación. Estos ficheros realizan todas las funcionalidades de cada actividad. Hay que destacar dos actividades:

    - ScreenAV: Es bastante importante porque contiene las funciones que se ejecutan al realizar, recibir o finalizar una llamada de audio o audio/video, además de mostrar en la pantalla del móvil diferentes diseños dependiendo en el estado en que se encuentre la llamada.

Mencionar el siguiente método:

- public static boolean makeCall (String remoteUri, NgnMediaType mediaType): Primero comprueba si es válida la sip-uri del destino para poder crearse una NgnAVSession y llamar al método de dicha clase capaz de realizar una llamada.

- ScreenHome: Esta clase se encarga del menú de inicio de la aplicación y de gestionar los iconos que se deben mostrar, dependiendo de si el usuario está registrado o no.

- Directorio Services: Sólo contiene un fichero que controla la actividad que se está ejecutando en el terminal. Permite iniciar, parar, destruir o mostrar la actividad. También lleva una lista del orden en que se ejecutan las actividades para poder realizar la función de retorno a la actividad anterior.

- Directorio Utils: Contiene cuatro ficheros, los cuales simplemente se encargan de definir el formato de la fecha, manipular los botones agregándoles imágenes o texto, el mantenimiento de la relación de aspecto, etc.

Es interesante explicar la librería que utiliza la aplicación android-ngn-stack, porque es ésta la que usa la librería doubango para mandar y gestionar los mensajes SIP. Al igual que la aplicación IMSDroid, esta librería está distribuida por una serie de directorios, los cuales se muestran en la siguiente figura.

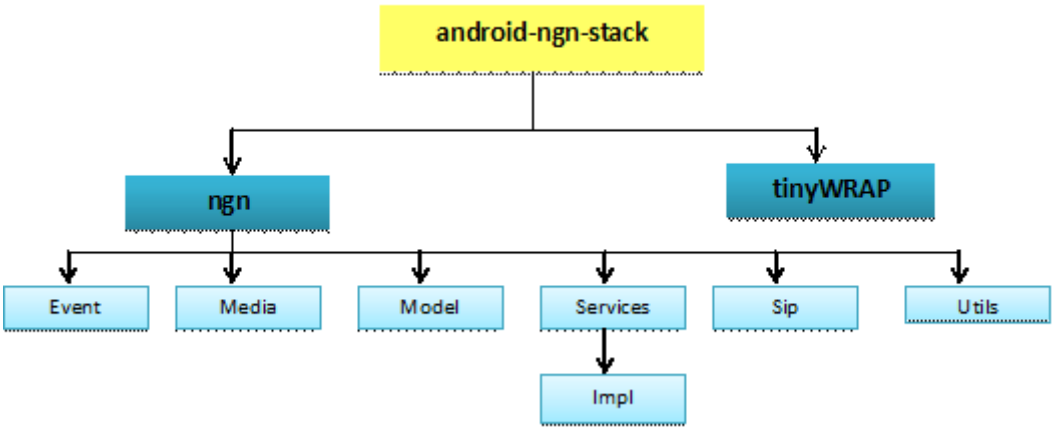


Figura 15: Esquema de la librería android-ngn-stack.

- Directorio ngn: A parte de contener otros directorios, tiene tres clases.

  - NgnEngine: Es el principal punto de acceso a todos los servicios (SIP, XCAP, MSRP,…). Desde cualquier parte del código se puede obtener una instancia a esta clase mediante el método *getInstance()*.

  - NgnApplicaction: Es un objeto global que define la aplicación. Se debe extender esta clase en la propia aplicación Android.

  - NgnNativeService: Se ejecuta en segundo plano.

- Directorio event: Se encarga de todo lo relacionado con los eventos de SIP. Para cada tipo de acción existen dos ficheros que los relaciona, uno para definir todos los tipos de eventos relacionados con la acción y el otro para el argumento de dicha acción. Las acciones que contiene el paquete son: invite, registration, subscription, publication, stack, msrp, messaging y media plugin.

- Directorio media: Este paquete lleva a cabo todas las funciones media, como por ejemplo los tipos media que soporta la aplicación. También gestiona la cámara móvil y los proxies, tanto del consumidor como del productor.

- Directorio model: Define un modelo para el historial de los mensajes, para los contactos de la agenda, etc.

- Directorio services: Gestiona las funciones de un servicio como el sonido, el almacenamiento, la configuración, etc. Contiene una serie de interfaces que será el directorio *impl* el que defina las clases correspondientes.

    - Directorio impl: Este directorio tiene una de las clases más importantes de android-ngn-stack, NgnSipService. En esta clase se encuentra el método para poder comprobar si el registro es válido o no, dependiendo de la configuración del terminal. Además es capaz de enviar los distintos tipos de eventos. Por otro lado está compuesto de una clase estática, MySipCallback. Es en ella donde se reciben los diferentes eventos y dependiendo del tipo de evento realiza una acción u otra, por ejemplo:

        - public int OnInviteEvent (InviteEvent e): Este método se llama automáticamente cuando se recibe un evento invite y dependiendo del tipo de evento envía un mensaje u otro. Por ejemplo, para el tipo tsip_ao_request manda un mensaje RINGING si el código del mensaje es 180 y, en caso contrario, envía un SIP_RESPONSE.

- Directorio sip: Es capaz de realizar cualquier función del protocolo SIP. Sus ficheros se encargan de definir una llamada de audio y audio/video, una sesión SIP o MSRP para compartir archivos, crear un stack SIP/IMS, modificar las sip-uris del mensaje, etc. Contiene una serie de clases entre las que hay que mencionar:

    - NgnAVSession: Es esta clase la que se encarga de llamar a los métodos de la clase CallSession para poder enviar un mensaje INVITE, entre otras funciones.

    - NgnRegistrationSession: En esta clase se encuentran los métodos necesarios para enviar una petición REGISTER o UNREGISTER.

    - NgnSipSession: Define una sesión SIP. Contiene una serie de

métodos para gestionar la sesión, como por ejemplo: finalizar la sesión, conseguir el *stack* SIP asociado,… También se puede añadir capacidades al mensaje u obtener los valores de las cabeceras del mensaje.

- Directorio utils: Realiza funciones similares a las del directorio de la aplicación IMSDroid con su mismo nombre. Añade funciones relacionadas con la sip-uri o atributos para referirse a la configuración del terminal.

- Directorio tinyWRAP: Este paquete no se puede modificar, es generado automáticamente al modificar el código de la carpeta _common de la librería doubango.

Capítulo 5

Implementación de la movilidad de sesión con el protocolo SIP en la aplicación IMSDroid

En este capítulo se explica cómo se ha implementado, en la aplicación IMSDroid, la capacidad de transferir una sesión multimedia a través de los dos tipos de movilidad de sesión que se han descrito en el capítulo tercero.

Antes de empezar implementar dichos modos de movilidad de sesión, se realizó el CAS y con ello la comunicación entre el terminal y el CAS.

Una vez que la parte del CAS estaba realizada se comienza a implementar los distintos tipos de transferencia de sesión. Se empieza por el modo pull, ya que este modo es el que presenta menor señalización y por ello es más fácil implementarlo.

Una vez que el modo pull funciona correctamente, se implementó el modo push en la aplicación. No sin antes modificar parte del código para que las sip-uris soportasen las GRUUs.

Para acabar con el proyecto se modifica la interfaz gráfica de la aplicación dándole un aspecto de cliente IPTV.

Antes de describir cómo se ha llevado a cabo este proyecto se ha solucionado un pequeño error que tenía la aplicación IMSDroid.

5.1- Error al colgar una sesión multimedia entrante

Aunque la aplicación es muy completa y posee bastantes funcionalidades, tiene un pequeño error que se ha solucionado durante el desarrollo del proyecto.

El problema se produce cuando un terminal recibe una nueva sesión y quiere terminarla, no se finaliza dicha sesión. La sesión multimedia se acaba para este dispositivo pero para el otro terminal, el cual inició la sesión, sigue establecida.

En primer lugar, antes de solucionar el problema se debe averiguar por qué ocurre dicho error en la aplicación. Para averiguarlo se observaron los paquetes que se envían al finalizar una sesión multimedia, ya sea entrante o saliente del terminal. El problema reside en la cabecera *route* de la petición BYE. Dicha cabecera representa la lista de proxies a través de los cuales se envía la solicitud.

Esta lista, al realizar la petición al CORE de IMS, debe de seguir siempre el siguiente orden:

- Primero debe pasar por el proxy P-CSCF.
- A continuación la petición debe pasar por el S-CSCF.

Sin embargo, para el problema descrito al principio de este apartado no se realiza en este orden, sino que en orden inverso. Para resolver este inconveniente, se procede a examinar el código.

Primero se estudió el código de la aplicación, pero no se encontró ningún error, por lo que se procedió a examinar el código de la librería doubango. Tras su examen se comprobó que el problema radicaba en la propia librería, exactamente en la manera en que se crea la cabecera *route*.

La librería crea la cabecera *route* a partir de otra, *record-route*. El motivo es que esta cabecera guarda la lista de proxies por los que pasa la petición. Sin embargo se debe de tener en cuenta las dos situaciones en las que se puede encontrar el dispositivo.

La primera de ellas es que el terminal realice una nueva sesión multimedia. En este caso, la librería coge la cabecera *record-route* del mensaje de aceptación de la sesión (200 OK) para crear la cabecera *route* de los posteriores mensajes SIP. El campo *record-route* del mensaje es el siguiente:

Figura 16: Cabecera Record-Route del mensaje 200 OK.

Observando la figura anterior y teniendo en cuenta el orden que debe tener la cabecera *route* es obligatorio copiar la lista de proxies en orden inverso. Para realizarlo, la librería utiliza el método *tsip_dialog_update* de la clase tinySIP→src→dialogs→tsip_dialog.c. Este método utiliza la función *tsk_list_push_front_data (list, data),* que recorre los elementos de la lista y los inserta en data en orden inverso.

La otra situación es que el dispositivo reciba una sesión multimedia. Para este caso se coge la cabecera *record-route* del mensaje INVITE que se recibe, el cual tiene la siguiente estructura:

```
Record-Route: <sip:mt@pcscf.open-ims.test:4060;lr>
Record-Route: <sip:mt@scscf.open-ims.test:6060;lr>
```

Figura 17: Cabecera Record-Route del mensaje INVITE.

Como se observa, no se debe invertir el orden de la lista de los proxies, este es el fallo, porque la librería sí invierte el orden de la cabecera *record-route*. De modo que el mensaje de finalización que se manda tiene el siguiente esquema:

```
Route: <sip:mt@scscf.open-ims.test:6060;lr>
Route: <sip:mt@pcscf.open-ims.test:4060;lr>
```

Figura 18: Cabecera Route del mensaje BYE.

Al mandar el mensaje BYE con esta ruta, el CORE IMS rechaza la petición y envía una respuesta con código 400 indicando que no es posible seguir la ruta indicada porque la ruta es incorrecta.

Para solucionar el problema simplemente hay que guardar la lista en el mismo orden, por lo que es necesario encontrar la parte del código errónea. Se examina la clase tsip_dialog.c y se observa que en el método *tsip_dialog_update2* se utiliza la misma función que para el método *tsip_dialog_update*, por lo que ordena la lista en orden inverso. Para arreglarlo solamente hay que cambiar esa función por otra que ordene la lista en el mismo orden, es el caso de la función *tsk_list_push_front_data (list, data)*.

5.2- CAS

Para poder realizar la movilidad de sesión es necesario guardar una serie de datos en una base de datos. Para ello es necesaria la comunicación entre el teléfono móvil y la base de datos. Esta comunicación se realiza gracias a un servidor Apache, el cual actúa como intermediario entre el dispositivo y dicha base de datos.

Dependiendo del modo de movilidad de sesión que se esté realizando son necesarios diferentes datos por lo que es necesario crear varias tablas.

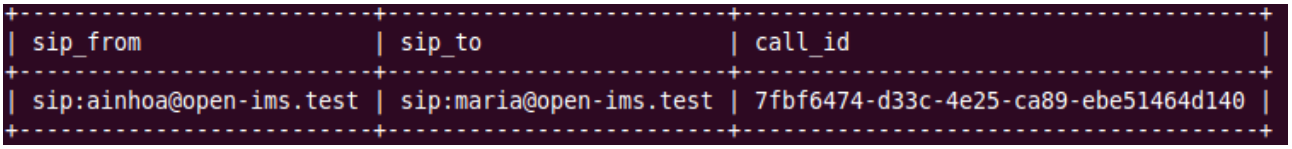
Por mayor comodidad, el código del servidor Apache se ha realizado en php. El motivo es la cantidad de información que hay por Internet además de tener nociones de php.

La comunicación entre el terminal y el CAS se realiza mediante peticiones HTTP utilizando el método POST. Se usa este método porque es el idóneo para transferir datos del cliente al servidor. Además el CAS también puede enviar datos al cliente como respuesta a su petición.

Por otra parte, cuando el dispositivo necesita enviar ciertos datos al CAS o, el CAS responder con una serie de información al cliente, se utiliza el lenguaje XML.

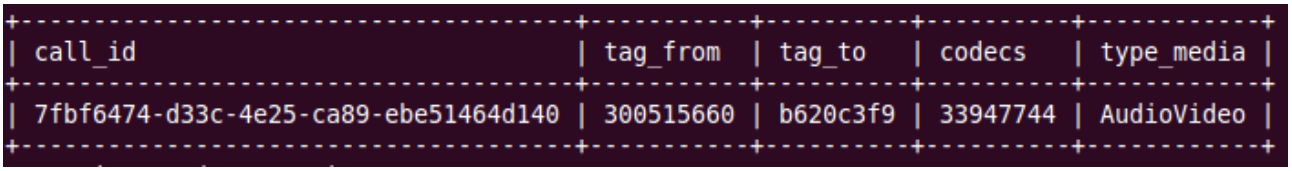
5.2.1- Modo Pull

Para el modo pull se han creado dos tablas, dichas tablas se muestran a continuación:



sip_from	sip_to	call_id
sip:ainhoa@open-ims.test	sip:maria@open-ims.test	7fbf6474-d33c-4e25-ca89-ebe51464d140

Figura 19: Tabla user_sip.



call_id	tag_from	tag_to	codecs	type_media
7fbf6474-d33c-4e25-ca89-ebe51464d140	300515660	b620c3f9	33947744	AudioVideo

Figura 20: Tabla information.

La tabla user_sip almacena información de los participantes de la sesión. Tiene los siguientes campos:

- sip_from: Es de tipo carácter y hace referencia al origen de la sesión.
- sip_to: Es de tipo carácter y hace referencia al destino de la sesión.
- call_id: Es de tipo carácter e identifica una sesión.

La segunda tabla guarda información de la sesión. Estas dos tablas están relacionadas por el campo call_id, es único por tanto. No pueden existir dos sesiones con el mismo identificador de sesión. La tabla information tiene los siguientes campos:

- call_id: Es de tipo carácter e identifica una sesión
- tag_from: Es de tipo carácter y hace referencia al tag de la sip-uri del origen.
- tag_to: Es de tipo carácter y hace referencia al tag de la sip-uri del destino.
- codec: Es de tipo carácter y hace referencia a los codec del SDP.
- type_media: Es de tipo carácter y hace referencia al tipo de sesión establecida. Este campo se utiliza para crear una nueva sesión del tipo que se le diga.

Para este modo de movilidad de sesión son necesarios cuatro tipos de comunicaciones.

1. Al establecer una sesión es imprescindible guardar en las dos tablas anteriores la información de la sesión.

   Dicha información se obtiene mediante el mensaje de aceptación (200 OK) que se recibe al enviar el mensaje INVITE y con la configuración del propio terminal. Después de enviar el ACK, se manda una petición HTTP

utilizando el método POST y usando el lenguaje XML para los datos, para guardar dichos datos en las tablas de la base de datos.

2. Al finalizar una sesión es obligatorio eliminar la información asociada a esta sesión. Para ello se realiza una petición cuando el estado de la sesión es *terminated*. Para eliminarla de la base de datos es preciso mandar al CAS el identificador de llamada. El CAS recibe el call-id y busca en las tablas con cual se corresponde para borrar los registros asociados a éste.

3. Al encontrarse en la pantalla del modo pull es necesario realizar una consulta al CAS para que éste sea capaz de responderle con todas las sesiones establecidas en ese momento. Para este caso el cliente no envía ningún tipo de datos al CAS, pues es el CAS el que envía los datos al teléfono móvil.

4. Por último, es preciso efectuar una petición al CAS para que el dispositivo reciba la información de la sesión y así poder realizar este tipo de movilidad de sesión. El terminal envía al CAS el call-id de la sesión requerida y el CAS le responde con los tags y los codec correspondientes.

5.2.2- Modo Push

Para este modo sólo se ha creado una tabla, ya que simplemente es necesario almacenar la información de registro del usuario.

sip_uri	gr	display
sip:ainhoa@open-ims.test	81x6ky4d-o2xd-p5z7-p5z7-8paioc0xsycw	samsung galaxy s2
sip:tania@open-ims.test	eldglzgu-3j5a-3j5a-3j5a-n16i0szwgzpd	samsung ace

Figura 21: Tabla register.

Esta tabla contiene los siguientes campos:

- sip_uri: Es de tipo carácter y hace referencia a la sip-uri con la que se realizó el registro.

- gr: Es de tipo carácter y hace referencia a la GRUU de la sip-uri. Es única para cada registro.

- display: Es de tipo carácter y hace referencia al campo display name de la configuración de identidad de la aplicación. Este valor no se puede repetir.

A continuación se explica los tipos de comunicación que existen entre el terminal y el CAS:

1. Una vez que el registro ha sido satisfactorio, es imprescindible almacenar la información mencionada anteriormente. Al recibir el mensaje de registro, el terminal envía los datos al CAS para que éste los guarde en la base de datos.

2. Al efectuar el des registro del usuario han de eliminarse los datos del usuario de la base de datos. Para ello, el dispositivo envía al CAS su GRUUS asociada. El CAS recibe la GRUUS y busca en la base de datos cuál de todos debe de suprimir.

3. En la pantalla de Transfer Sessions se realiza una petición al CAS para que éste responda con todos los usuarios que están registrados. A continuación, el propio terminal filtra la información de su registro gracias a su GRUUS asociada. De este modo, en la pantalla del teléfono móvil aparece una lista con todos los usuarios registrados excepto el del propio dispositivo.

4. Antes de que un usuario se registre, es necesario hacer una comprobación del campo display de la tabla para que no se repita. Para ello, hay que realizar una petición al CAS con todos los usuarios que están registrados. Esta petición es la misma que para el caso anterior.

5.3- Modo Pull

Como se ha mencionado anteriormente, para realizar este tipo de movilidad de sesión solamente hay que añadir la cabecera *replace* al mensaje INVITE y, mantener los mismos codecs del SDP.

Para poder crear la cabecera *replace* es necesario conseguir el call-id de la sesión establecida y los tags de la cabecera from y to. También, es preciso conocer los codecs para crear el mismo mensaje SDP y poder mantener la misma sesión multimedia.

Obtener el call-id del mensaje INVITE no es demasiado costoso ya que hay un método que consigue dicho valor, *getSipHeaderValue (String header)*. Este método pertenece a la clase SipMessage. Sin embargo, no hay ningún método para obtener los tags de la cabecera from y to, en la librería sólo se consigue la sip-uri. Para ello es necesario modificar el código de la librería doubango.

En la clase SipMessage.cxx de la librería doubango se crean dos nuevos métodos, *getSipFromTag()* y *getSipToTag()*, los cuales devuelven los tags correspondientes a las cabeceras from y to, respectivamente.

Además es necesario almacenar el tipo de sesión media, pues a la hora de crear una nueva sesión multimedia hay que pasarle como parámetro el tipo que se desea establecer. Por lo que también se guardará este valor en la base de datos. Conseguir el

tipo de media no tiene mucha complicación, ya que la librería android-ngn-stack tiene el método *getMediaType()* en la clase NgnInviteSession que obtiene dicho valor.

Una vez que se sabe cómo conseguir estos datos, se procede a almacenarlos en la base de datos únicamente cuando se ha iniciado la sesión. Para ello se crean una serie de atributos en la clase NgnSipService que se corresponden con cada uno de dichos datos. Será al recibir el mensaje OK del INVITE del que se obtengan los datos necesitados y se guarden en sus correspondientes atributos. Para poder conseguir estos datos desde cualquier clase es necesario crear los métodos *get* de cada uno. Solamente queda ingresar estos datos en la base de datos. Se envía una petición al CAS con todos los datos que se quieren almacenar. Esta petición se realiza en la clase AVSession, cuando la sesión se encuentra en el estado *incall*. Para el caso del tipo media no es necesario crear un atributo porque su valor se obtiene con la llamada al método *getMediaType()* en el momento de realizar la petición al CAS.

Hay que tener en cuenta que se deberá eliminar la información de la sesión cuando ésta finalice. Por ello, cuando la sesión esté en estado *terminated* se envía otra petición al CAS, pero en este caso se adjunta el call-id de la sesión que se desea eliminar.

Una vez que se ha conseguido almacenar y eliminar los datos en la base de datos, es necesario crear una nueva actividad en la aplicación. Esta actividad deberá aparecer en el menú principal exclusivamente cuando el usuario esté registrado. Para ello se añade un nuevo ítem en el menú de la clase ScreenHome.java con el siguiente icono,  . Este ítem se ha colocado en quinta posición, como se muestra en la siguiente figura.

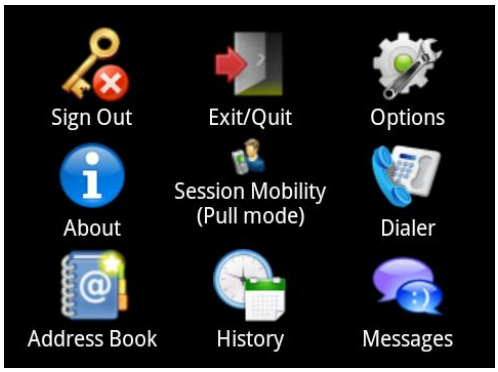


Figura 22: Menú de inicio con el nuevo icono.

No hay que olvidar, que al crear una nueva clase en la aplicación se debe de incluir esta actividad en el archivo AndroidManifest.xml. Si no se incluye en el fichero, la actividad no será reconocida por la aplicación.

A continuación se procede a realizar la nueva clase que se denomina ScreenPullMode.java. Esta clase simplemente consiste en una lista con todas las sesiones establecidas en ese momento.

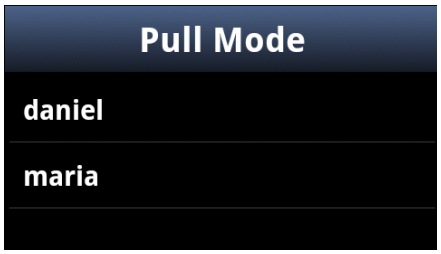


Figura 23: Actividad Modo Pull.

Como se puede observar en la figura anterior se muestra una lista con una serie de elementos. Estos elementos se distinguen unos de otros por medio de un nombre, el cual hace referencia al destino de la sesión. Es la sip-uri de la cabecera to eliminando el dominio donde el usuario está registrado. Los datos se consiguen realizando una petición al CAS para que éste los obtenga de la base de datos. Con la ayuda del método getItems de la propia clase, se añaden a la lista los datos y de la forma que se desee.

En el caso de haber un número elevado de sesiones activas, en la pantalla del teléfono móvil no se podrán ver todas. Para verlas, la lista tiene una barra de desplazamiento vertical.

Para realizar el modo pull sólo hay que pulsar sobre el elemento de la lista al que se quiera realizar la movilidad. Si por alguna razón no se ha podido efectuar este modo, se muestra en la pantalla la siguiente frase "*Error al realizar la petición.*".

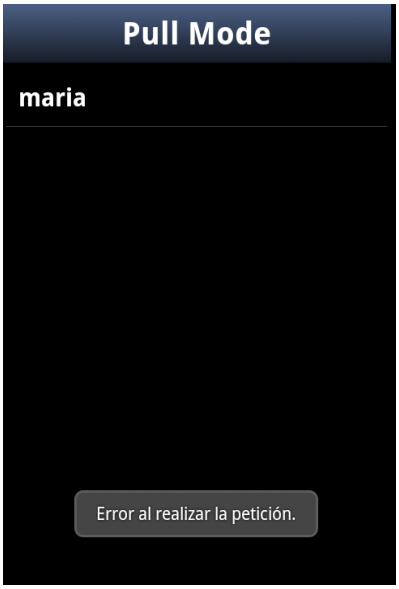


Figura 24: Erros al realizar la petición.

Para efectuar este modo de movilidad se han seguido los mismos pasos que para realizar una llamada telefónica, pero modificando algunas cosas. Se ha añadido la cabecera *replace* al mensaje INVITE. Además, se ha incluido para los dos casos, mandar un mensaje INVITE con y sin la cabecera replace, la GRUU de la sip-uri del origen a la cabecera from del mensaje. Este cambio se explicará con más detenimiento después en este capítulo.

Antes de efectuar este modo de movilidad de sesión es preciso conocer el call-id de la sesión, el tag de la sip-uri del origen y del destino, el entero que indica los tipos de codecs que se necesita para el protocolo SDP y el tipo media de la sesión. Esto se consigue mediante una petición al CAS, enviando el call-id de la sesión requerida al pulsar sobre el elemento de la lista correspondiente.

Una vez obtenidos todos los datos necesarios, se crea el string correspondiente al campo replace del mensaje INVITE y, se realiza una llamada al método *public static boolean makePull (String fromUri, String toUri, String replace, NgnMediaType mediaType, int codec, String gr)* de la clase ScreenAV pasándole como parámetros la sip-uri origen y destino, la cabecera replace, el tipo de media, los codecs y la GRUU del origen. Este método primero comprueba que las sip-uris son válidas. Si es así, procede a crearse una sesión media nueva con el tipo correspondiente y después, vuelve a llamar a otro método para poder realizar el modo pull, método *public boolean makePull* (*String fromUri, String toUri, String replace, NgnMediaType mediaType, int codec, String gr*) de la clase NngAVSession de la librería android-ngn-stack.

Es este método el que, dependiendo del tipo media de la sesión multimedia, envía un mensaje INVITE u otro, lo único que cambia es el SDP del mensaje. En el mensaje INVITE se añade la cabecera replace. También se modifica los codecs del mensaje SDP para que sean los mismos que los del primer mensaje.

Las siguientes figuras muestran un ejemplo del modo pull. La primera de ellas se corresponde con el mensaje OK que se obtiene para poder ver el call-id de la sesión y los tags. La otra sólo muestra la cabecera replace del nuevo INVITE para comprobar los datos anteriores.

```
To: <sip:daniel@open-ims.test>;tag=97499020
Via: SIP/2.0/UDP 10.0.2.15:33726;received=163.117.140.226;branch=z9hG4bK1557049883;rport=34336
Record-Route: <sip:mo@scscf.open-ims.test:6060;lr>,<sip:mo@pcscf.open-ims.test:4060;lr>
CSeq: 427185293 INVITE
Call-ID: bab8ef45-4343-4775-98f9-440247442620
From: <sip:tania@open-ims.test;gr=joig3gd4-ew1i-ew1i-orbm-bje1tob58ykm>;tag=1122253153
P-Asserted-Identity: <sip:daniel@open-ims.test>
Contact: <sip:163.117.140.131:5060;transport=udp>
Content-Type: application/sdp
Content-Length: 256
```

Figura 25: Mensaje OK como respuesta al INVITE.

```
Replaces: bab8ef45-4343-4775-98f9-440247442620;from-tag=1122253153;to-tag=97499020
```

Figura 26: Cabecera Replace del nuevo INVITE.

5.4- Modo Push

La aplicación IMSDroid no tiene la capacidad de incluir en la sip-uri la GRUU, por lo que antes de realizar este modo de movilidad de sesión es necesario aportar a la sip-uri del registro del usuario una GRUU que lo asocie. De esta forma es posible identificar usuarios con la misma sip-uri y evitar realizar forking, como ya se ha mencionado en el capítulo tres.

5.4.1- Globally Routable User Agent Uris (GRUUs)

Previamente a otras acciones, es necesario conocer el formato que debe tener la GRUU. Para ello se observan los paquetes procedentes del Monster (Multimedia Open InterNet Services and TelecommunicationEnviRonment), el cual es un marco extensible de plug-and-play desarrollado por Fraunhofer FOKUS. La GRUU tiene la siguiente estructura: *XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX,* donde X toma cualquier valor alfanumérico.

Una vez conocido el formato de la GRUU se procede a su implementación en la aplicación. Se necesita un nuevo atributo en la clase NgnRegistrationSession.java que haga referencia a la globally routable user agent uri del usuario, denominado gr. Este atributo es privado y final. Es privado para que el resto de las clases no puedan acceder a él, por este motivo se ha creado un método, *getGr()* para devolver el valor del atributo. Y es final para que no pueda ser sobrescrito o redefinido, es decir, se trata de una constante.

Al crear un nuevo registro de sesión (*public NgnRegistrationSession (NgnSipStack sipStack)*), se inicializa el atributo gr y posteriormente se añade a la cabecera contact del mensaje mediante el siguiente método *addCaps ("+sip.instance", gr)*. Este método procede de la clase NgnSipSession y simplemente se encarga de añadir capacidades a la cabecera contact del mensaje. El valor del atributo gr se crea utilizando un método que genera una cadena alfanumérica aleatoria de la longitud indicada, *String getCadenaAlfanumAleatroria (int longitud).*

A continuación se procede a añadir en las cabeceras from, refer-to y referred-by la GRUU del usuario. Para las dos primeras cabeceras resulta fácil su implementación, sin embargo para el referred-by se ha modificado parte del código de la librería doubango.

Para la cabecera from simplemente se añade la GRUU a la sip-uri del origen. Antes de realizar el establecimiento de cualquier sesión, en la clase NgnAVSession se modifica esta cabecera por medio del método *setFromUri*, por lo que resulta sencillo concatenar los dos string, sip-uri y gr, y pasarlo como parámetro del método de la siguiente forma *setFromUri (fromUri+";gr="+gr).*

Esta cabecera se ha modificado en los métodos *makeCall*, *makePull* y *referCall*. El primero y el segundo de ellos se han cambiado con el fin de que, al enviar cualquier INVITE, la cabecera from del paquete tenga la GRUU y se conozca de quién procede el

mensaje. Para el *referCall*, es el mismo motivo que los anteriores, pero en este caso para el mensaje REFER.

Para el caso de refer-to, resulta sencillo su implementación porque el método que envía el REFER, *transfer (String refer-to)*, recibe como parámetro la sip-uri de dicha cabecera. Por lo que se procede de la misma manera que para la cabecera from, concatenando los dos string.

Sin embargo para la última cabecera, referred-by, es necesario poder modificarla como ocurre para el caso de la cabecera from. De este modo se procedería de la misma forma que en los casos anteriores, concatenando strings.

Antes de realizar la modificación en la librería doubango, el método *transfer* sólo necesita un parámetro que hace referencia a la cabecera refer-to del mensaje REFER. Para crear la cabecera referred-by del mensaje simplemente coge el atributo IMPU de la configuración de la identidad, por lo que no hay la opción de modificar dicha cabecera. De esta forma, es preciso cambiar el código de la librería para poder añadir otro parámetro a este método que permita modificar el valor de la cabecera referred-by. Para ello se modifican las siguientes partes del código:

- En el directorio tinySIP:
  - En la carpeta dialogs se modifica el archivo tsip_dialog_invite.ect.c. Concretamente el método *send_REFER* añadiendo un nuevo parámetro que hace referencia al valor de la cabecera referred-by.
  - En la carpeta action se modifica la estructura de tsip_action_t, especialmente la estructura ect, añadiendo un nuevo campo from dónde se almacenará el valor correspondiente a la cabecera referred-by.
  - En la carpeta api se modifican los archivos tsip_api_invite.h y tsip_api_invite.c el método *tsip_api_invite_send_ect*, para añadir un nuevo parámetro que hace referencia al nuevo campo creado en la estructura tsip_action_t.
- En el directorio bindings/_common:
  - Sólo se modifican los archivos SipSession.cxx y SipSession.h para añadir un parámetro que hace referencia al referred-by del mensaje REFER.

La siguiente figura muestra un ejemplo de un mensaje REFER, en el que las cabeceras from, refer-to y referred-by contienen la GRUU de cada sip-uri.

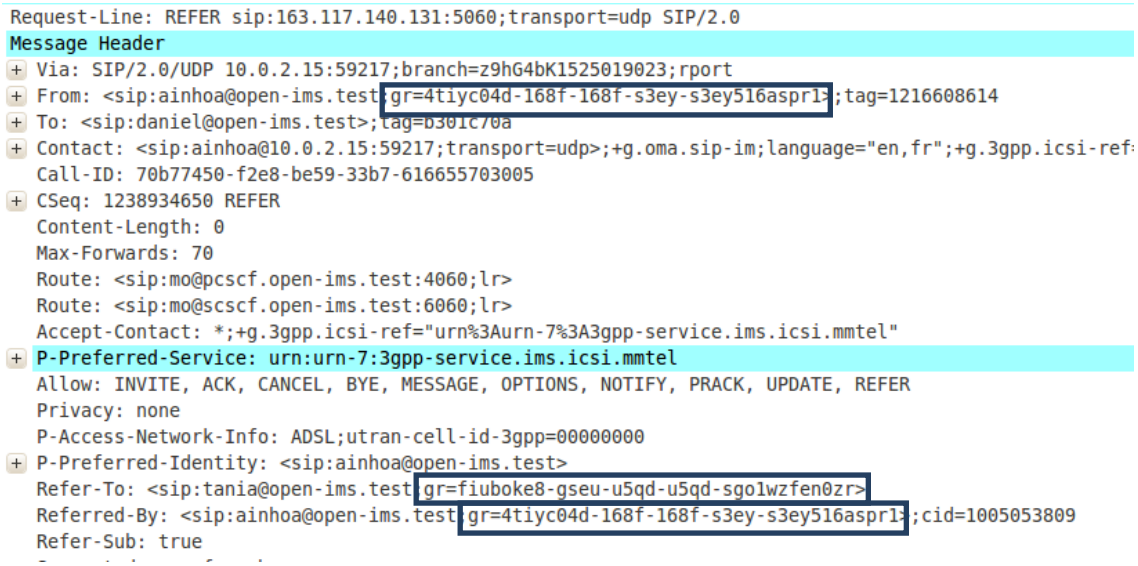


Figura 27: Mensaje REFER.

5.4.2- Implementación del envío del REFER en la aplicación

Como se ha explicado en el capítulo tres, el modo push se realiza enviando un mensaje REFER al terminal que se desee transferir la sesión. Este terminal aceptará o no la transferencia de la sesión multimedia.

La pantalla para este modo de movilidad de sesión sigue el mismo formato que para la del modo pull, pero añadiendo un nuevo botón. Esta pantalla consiste en una lista con todos los usuarios registrados, con una barra de desplazamiento vertical y un botón para volver a mostrar el seguimiento de la sesión multimedia. Sin embargo, para este caso no se ha creado una actividad nueva, sino que se ha añadido en la clase AVSession el código necesario para mostrar la pantalla en el terminal.

Para que aparezca una lista con los usuarios registrados es necesario que previamente se hayan almacenado en la base de datos. En clase ScreenHome, al recibir un evento de registro del tipo *REGISTRATION_OK* se inserta en la base de datos la información del usuario mediante una petición al CAS.

Del mismo modo, es preciso eliminar la información del usuario cuando se desregistra, por lo que al recibir un evento de registro pero en este caso del tipo *UNREGISTRATION_OK*, se suprime su información realizando una petición al CAS con la GRUU del usuario que se desea borrar.

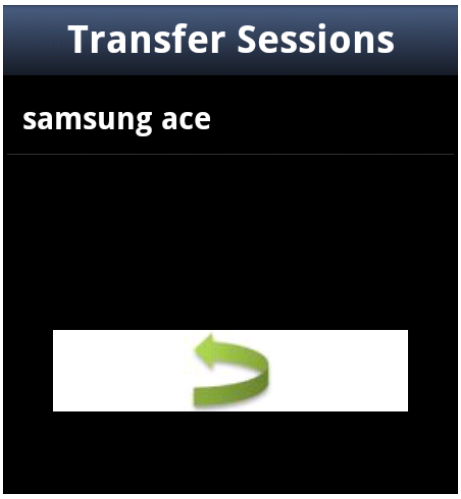


Figura 28: Pantalla para transferir una sesión.

En este caso se tiene una lista con un elemento cuyo nombre hace referencia al campo de display name, perteneciente a la configuración de identidad del usuario. Los datos se consiguen mediante una petición al CAS para que éste los obtenga de la base de datos. Es el método *getItems* de la clase AVSession el que devuelve estos datos como una lista de array para mostrarlos en la pantalla.

Es al pulsar sobre un elemento de la lista, cuando se pide al CAS la sip-uri y la GRUU asociada a ese display name. Para que el CAS sólo nos devuelva un usuario es necesario que el valor del display name sea único por cada registro del terminal.

Se ha modificado parte del código para que antes de enviar un REGISTER se consulte en la base de datos si el display name del dispositivo está siendo usado por otro terminal o por lo contrario es diferente a todos los que hay en la base de datos. Si existe un usuario registrado con el mismo display name que se desea registrar, la aplicación no deja enviar el mensaje REGISTER al CORE IMS y muestra el siguiente mensaje "*Nombre de usuario repetido. Introduzca otro display name*". Esta parte del código se ha cambiado en el método register de la clase NgnSipService de la librería android-ngn-stack.

Antes de explicar la implementación del modo push, es interesante comentar cómo se llega a esta pantalla en la aplicación. Para realizar la transferencia de sesión es obligatorio tener establecido una sesión. Por ello la pantalla del modo push se realiza en la clase ScreenAV.

En la clase AVSession hay un método que crea un menú con distintas opciones, *public boolean createOptionsMenu (Menu menu)*, dependiendo del estado de la sesión (incoming, inprogress, incall, terminated, terminating). Para el estado incall se añade un nuevo icono que corresponde con la transferencia de sesión, pero sólo cuando se trata de una sesión audio/video. Al pulsar este icono,  , se da paso a la pantalla de la última figura mostrada. El menú de opciones que se obtiene en la pantalla cuando se ha establecido una sesión de audio/video es el siguiente:

Figura 29: Menú en la pantalla cuando se ha establecido una sesión de audio/video.

Cuando se pulsa el icono anterior, automáticamente se llama al método *loadListRefer()*, que se encarga de mostrar en la pantalla del terminal la lista con los usuarios registrados y el botón de volver.

Para realizar el modo push simplemente hay que pulsar sobre el elemento de la lista al que se desee transferir la sesión. Si por alguna razón hay algún tipo de error que impida enviar el refer, en la pantalla se muestra un mensaje, como ocurre en el modo pull, pero en este caso el mensaje es *"Error al realizar el push."*.

Si no se produce ningún error, se siguen los siguientes pasos para enviar el REFER:

1. Cuando se pulsa un elemento de la lista, se realiza una petición al CAS por medio del display name para obtener la sip-uri y el gr asociada con el elemento seleccionado y, de este modo, poder crear el valor de la cabecera refer-to. Una vez que se tiene el valor se llama al método *referCall* de la clase AVSession. Este método recibe como parámetro el valor del refer-to.

2. El método *referCall* comprueba si hay una NgnAVSession creada. En su caso llama a un método de su clase, *referCall*, que recibe como parámetros la sip-uri del refer-to, la sip-uri y la GRUU del usuario registrado en ese terminal para poder crear el valor de la cabecera referred-by. En caso contrario no realiza ninguna operación.

3. Es en la clase NgnAVSession el método referCall quien llama al método *transfer* de la clase CallSession, éste es el que se encarga de enviar el REFER. El método transfer sólo recibe como parámetros las cabeceras refer-to y referred-by del mensaje. Antes de enviarlo se comprueba si el terminal está conectado. En el caso de que no lo estuviera se rechaza la transferencia de sesión.

Como se explicó en el capítulo 3, al enviar el REFER, automáticamente se envía un INVITE al terminal que se quiere realizar la transferencia de sesión. Esto conlleva que en dicho terminal aparezca la siguiente imagen:

Figura 30: Pantalla cuando se recibe el INVITE.

No se observa el nombre del usuario que realizó la movilidad de sesión, sino del usuario que se encuentra en el otro extremo de la sesión. Será este terminal el que elija si desea continuar con la sesión multimedia o no.

5.5- Cambio de la interfaz

Como se ha mencionado al principio de este capítulo, se ha modificado la interfaz gráfica de la aplicación para darle otra imagen.

A continuación se enumeran los distintos cambios realizados:

1. El menú de inicio de la aplicación, cuando el usuario está registrado, se ha modificado. Se han eliminado algunos iconos del menú y se ha cambiado un icono.

Figura 31: Menú principal.

Como se puede observar, se han eliminado los iconos del historial de las llamadas, la agenda de los contactos guardados y la función de enviar mensajes.

El icono de llamada,  , se ha cambiado por  , igual que el texto.

2. El segundo cambio consiste en eliminar el marcador del número de teléfono por otro que simplemente sea un cuadro de texto dónde se introduce la sip-uri destino.

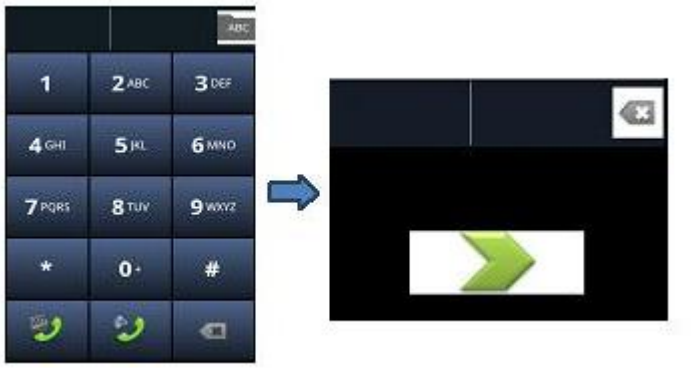


Figura 32: Pantalla para establecer una sesión.

Además se ha eliminado la opción de elegir entre realizar una llamada de audio o de audio/video. Ahora sólo se puede realizar sesiones de audio/video, pero se sigue pudiendo recibir llamadas de audio.

3. Otro cambio significativo es a la hora de realizar o recibir una llamada. Se ha modificado un poco el aspecto, eliminando sobre todo las imágenes de teléfono que aparecían y, cualquier texto que diera a atender que se trataba de una llamada telefónica.

Figura 33: Pantalla para establecer una llamada de audio.

En esta pantalla se ha eliminado la imagen avatar, procedente del usuario del otro extremo, en el caso de que esté guardado en memoria. Además se ha sustituido el botón de colgar por otro que simplemente tiene una X, para indicar que pulsando dicho botón se finaliza la sesión.

Figura 34: Pantalla al recibir una llamada.

Aquí se realizan las mismas modificaciones que en el apartado anterior. Además se elimina el título de arriba y se añade el texto *¿Desea aceptar esta sesión?*

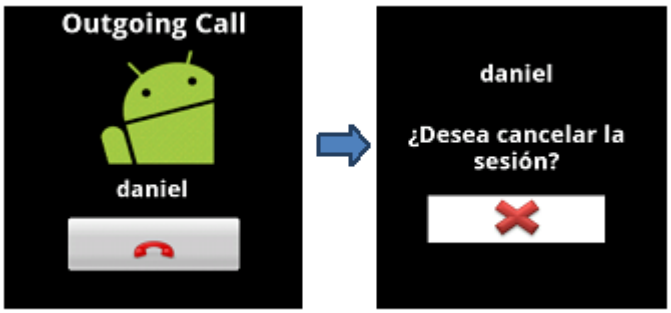


Figura 35: Pantalla para rechazar una sesión que todavía no se ha establecido.

En este caso se cambia la frase por otra que índica la opción que se puede realizar, *¿Desea cancelar la sesión?*

4. El último cambio de la interfaz se ha realizado en la pantalla cuando se efectúa una sesión de audio/video. Se ha eliminado una pantalla que aparece abajo a la derecha, porque esta pantalla es para mostrar el vídeo del propio terminal. Además se ha modificado el menú de opciones, quedando muy simple sólo con dos funciones. La primera función es terminar la sesión y la otra es transferir la sesión a otro terminal.

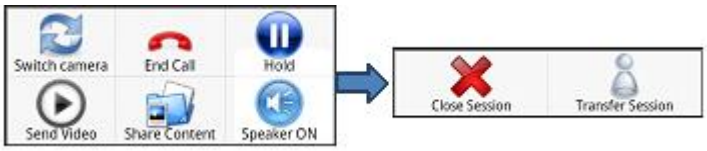


Figura 36: Menú de opciones.

Capítulo 6

Evaluación de la implementación

En este capítulo se describen las pruebas realizadas durante la implementación del proyecto y la evaluación de los resultados obtenidos.

Este proyecto ha utilizado la versión 2.0.453. Esta es la primera versión beta que soporta los codecs del vídeo. Actualmente IMSDroid se encuentra por la versión 2.0.504. Esta versión es capaz de realizar llamadas sipML5 o fijar en forma de vídeo cuando se llama en Chrome, además de soportar más tipos de codecs.

Además en la última versión se ha solucionado el problema que tenía la versión 2.0.453, cuando se colgaba una sesión multimedia entrante.

6.1- Entorno de pruebas.

A continuación se muestra el escenario de pruebas que se ha utilizado durante la implementación del proyecto.

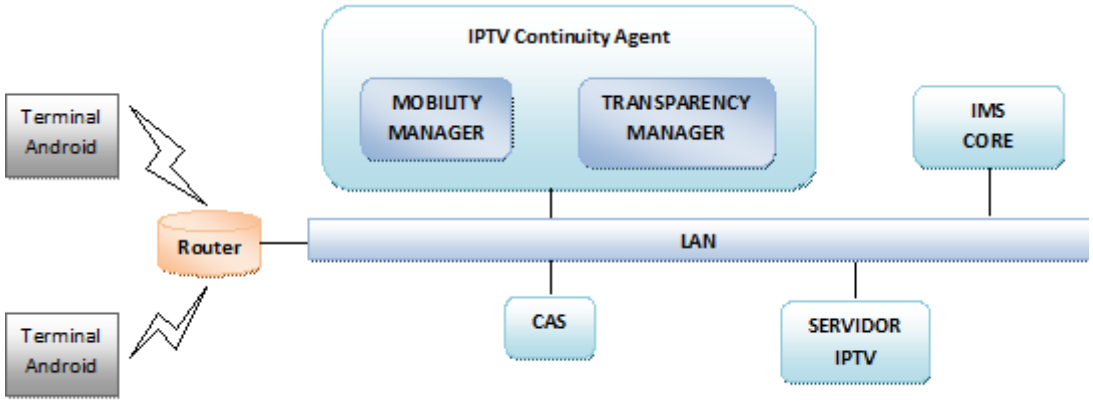


Figura 37: Escenario de pruebas.

Como plataforma IMS se ha utilizado la aplicación de código abierto Fokus Open IMS Core [27]. El IMS Core se ha desarrollado en una máquina virtual Linux Ubuntu 9.04. Se han creado tres usuarios en esta plataforma, uno de ellos representa al servidor IPTV y los otros dos a los terminales Android.

El CAS es una aplicación java que se ejecuta en otra máquina virtual Linux Ubuntu 10.4, la cual interactúa con la base de datos MySQL 5.1.6. Es en la base de datos MySQL donde se guarda el contexto de la información. El CAS envía/recibe peticiones HTTP al/desde el terminal Android para que se produzca la transferencia de sesiones multimedia.

Servidor IPTV se está ejecutando en la misma máquina virtual donde se encuentra el CAS. Este servidor no tiene la capacidad de movilidad y simplemente almacena un vídeo multimedia.

Mobility Manager y el Transparency Manager se ejecuta dentro del IMS Core. Esta aplicación ha sido desarrollada en otro proyecto, por lo que en este proyecto sólo se ha utilizado su funcionamiento no se ha modificado ninguna característica.

Todos estos elementos están conectados a la LAN por medio de una conexión Ethernet.

También se ha utilizado dos móviles Android, un Samsung Galaxy S2 con versión de Android 4.0.4 y un HTC Wildfire S con versión 4.0.4. Estos terminales tienen la capacidad de movilidad SIP. Los dos dispositivos están conectados por Wi-Fi a un Linksys WRT 54GL Wi-Fi router.

Uno de estos dos terminales móviles establece una sesión multimedia con el Servidor IPTV. A continuación se procede a transferir la sesión activa al otro dispositivo Android

mediante los dos modos de movilidad de sesión. Es posible realizar la transferencia de dicha sesión con la ayuda del CAS.

A continuación se muestra dos figuras donde se observa el escenario de pruebas en el laboratorio y la transferencia de sesión entre los dos dispositivos.

Figura 38: Modo Pull: Establecimiento de la sesión.

Figura 39: Modo Pull: Transferencia de sesión.

En estas dos figuras se muestra el modo pull. Primero la sesión multimedia se establece con el terminal Samsung Galaxy S2 donde se muestra el vídeo almacenado en el Servidor IPTV y a continuación dicho vídeo es transferido al otro terminal. En la pantalla del PC se tiene el CAS mostrando las tres tablas de dicha base de datos.

6.1.1- CAS

Una vez que se tiene la estructura del CAS y los archivos necesarios para el intercambio de información entre el terminal y CAS, se procede a comprobar en el Wireshark (analizador de tráfico) si se realiza correctamente la comunicación entre el dispositivo y el CAS.

Se capturan los distintos mensajes y se comprueba la información que se ha enviado. La información se envía mediante el protocolo XML porque resulta más cómodo tratar los datos.

Después de realizar una serie de pruebas y comprobar que todo funciona correctamente, se realiza el primer modo de movilidad de sesión.

6.1.2- Modo Pull

Para probar este modo no se utilizó de primeras el Servidor IPTV, sino que se empezó realizando una llamada de audio entre dos usuarios SIP.

El escenario consiste en dos usuarios SIP con movilidad y otro usuario SIP sin movilidad. Uno de los usuarios SIP con movilidad establece una llamada con el cliente SIP sin movilidad. Es el otro usuario con movilidad el que inicia desde otro terminal la transferencia de sesión enviando el mensaje INVITE con la cabecera replace.

Nuevamente con el analizador de tráfico se comprueba que todas las cabeceras del mensaje son correctas. Además visualmente se demuestra cómo la sesión se transfiere al otro terminal.

Cuando se cambia la interfaz de la aplicación, también se cambia el escenario de pruebas. Es en este momento cuando se cambia el usuario SIP sin movilidad por el Servidor IPTV y se realizan llamadas de audio/vídeo, es decir, el escenario de la figura anterior.

Para este caso se procede de la misma manera, analizando los mensajes enviados por el Wireshark y visualmente con los teléfonos móviles.

6.1.3- Modo Push

Para el modo push se siguen los mismos pasos que para el pull. Primero se utiliza un usuario SIP sin movilidad y posteriormente se sustituye por el servicio IPTV.

Igualmente se comprueba visualmente que se transfiere correctamente la sesión y además se analiza el mensaje REFER que se envía con la ayuda del Wireshark.

6.2- Evaluación y resultados.

Para evaluar la implementación del proyecto se mide el tiempo que se tarda en establecer la nueva sesión que se desea transferir. Para el modo pull se calcula el tiempo desde que se envía el primer mensaje INVITE que lleva la cabecera replace y se recibe el primer paquete de datos.

En cuanto al otro modo de movilidad se calcula el tiempo desde que se envía el mensaje REFER hasta que se recibe el primer paquete de datos.

De los dos modos de movilidad de sesión, el modo push tendrá que tener un tiempo superior debido a que necesita más señalización que el modo pull.

Debido a la dificultad de incluir en el propio terminal el código necesario para calcular este tiempo, se ha recurrido a capturar los mensajes con la ayuda del Wireshark. Con el

Wireshark se capturará una serie elevada de paquetes para poder realizar la media de dichos tiempos y obtener así una medida aceptable.

6.2.1- Modo Pull

Como se ha mencionado anteriormente se va a capturar un número suficiente de paquetes para poder realizar la medida. Sin embargo existe un problema, al tratarse de terminales móviles se necesita de una aplicación para poder capturar los paquetes en el terminal. Esta aplicación se llama *shark*. Dicha aplicación actúa como el analizador de tráfico Wireshark. Para poder utilizar la aplicación es necesario tener permisos de administrador, teniendo que modificar el sistema operativo del dispositivo para conseguir el control total sobre él (*rootear*).

Una vez que el móvil Samsung Galaxy S2 ha sido rooteado, se le ha cambiado la versión Android del terminal por la última del mercado. El móvil ha pasado de tener la versión 2.3.6 a la versión 4.0.4 de Android.

Para realizar la medida se han realizado los siguientes pasos:

1. Un terminal cualquiera establece la sesión con el Servidor IPTV.
2. El Samsung Galaxy S2 es el que transfiere la sesión enviando el mensaje INVITE con la cabecera replace y del mismo modo recibiendo los datos del Servidor IPTV.

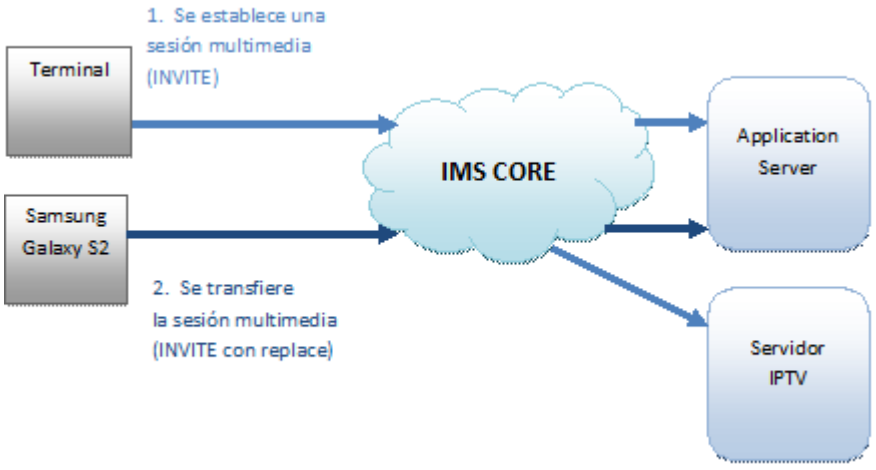


Figura 40: Esquema de pruebas para el modo pull.

Se realizan treinta medidas y se obtiene *113.4487 mseg*.

Como se puede observar el resultado obtenido es bastante bajo, inapreciable para la vista del usuario.

6.2.1- Modo Push

Para poder realizar esta medida es preciso modificar una pequeña parte del código para que al recibir una sesión, el terminal automáticamente la acepte. Con esto se consigue eliminar el pequeño retardo que se produce hasta que el usuario acepte la sesión del terminal.

Para este caso es necesaria la utilización de dos terminales móviles para realizar la medida. Por lo que aparte del Samsung Galaxy S2 se ha utilizado el HTC Wildfire S. Éste último también ha sido rooteado y cambiado a la versión 4.0.4 de Android.

Al tener dos terminales distintos existe el problema de sincronización, es decir, cada dispositivo Android tiene una hora diferente y resulta bastante complicado conseguir que los dos móviles tengan la misma hora. Con lo que para resolver dicha dificultad se envía un paquete UDP en modo broadcast para que sea recibido por los dos terminales y tener un tiempo de referencia.

El paquete se envía con una pequeña aplicación, la cual envía como datos IMSDROID. Dicho paquete es capturado por la aplicación shark de cada móvil, se observa el tiempo en el que se recibe el mensaje y se toma como tiempo de referencia para los siguientes paquetes. De este modo se sincronizan los dos dispositivos y se puede medir el tiempo de transferencia.

Para la realización de esta medida siempre se procede de la siguiente manera:

1. Primero se envía el paquete UDP que será recibido por los dos terminales.
2. El Samsung Galaxy S2 establece conexión con el Servidor IPTV y a continuación manda el REFER dirigido al HTC Wildfire S.
3. El terminal Samsung Galaxy S2 inicia la transferencia de sesión.
4. Es el terminal HTC Wildfire S el que recibirá los datos procedentes del Servidor IPTV.

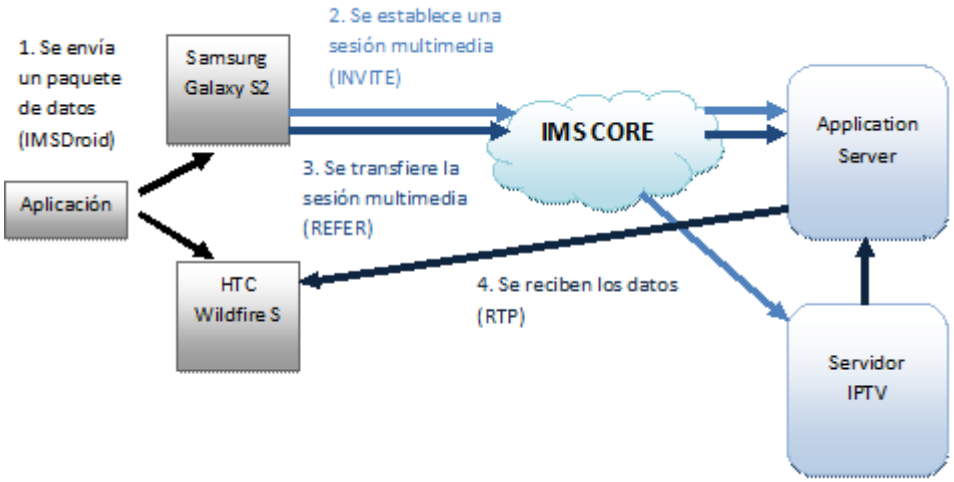


Figura 41: Esquema de pruebas para el modo push.

Para este caso también se toman treinta medidas y se obtiene *1.1293 seg*.

Los datos obtenidos para el modo push son mayores que para el modo pull, como debe de ser. Sin embargo el valor obtenido es demasiado alto. Hay una gran diferencia entre un modo y otro.

Además realizando las pruebas se comprueba que la transferencia de sesión no se realiza inmediatamente y es percibida por los usuarios.

Debido a la elevada medida que se obtiene para este modo, se ha realizado la misma prueba pero utilizando SIPp (genera tráfico SIP) en vez de la aplicación IMSDroid. Los valores obtenidos son más bajos que los obtenidos con los dispositivos móviles. El valor obtenido usando SIPp ha sido 73 ms con 30 repeticiones.

Una explicación a este resultado que parece bastante probable, es que el terminal tiene programado algún tipo de retardo a la hora de aceptar una sesión entrante. A ésta explicación se ha llegado después de observar en las trazas que el terminal al recibir un INVITE envía varios RINGINGs antes de contestar con un 200 OK.

Capítulo 7

Conclusiones

Se ha logrado implementar la movilidad de sesión en la aplicación IMSDroid tanto para el modo pull como para el push. Para conseguirlo, ha sido necesario modificar el código de la aplicación así como la librería android-ngn-stack y tinyWRAP.so.

El proyecto se ha desarrollado teniendo en cuenta que un cliente registrado en el CORE IMS, pueda establecer una sesión con un Servidor IPTV. Por lo que se ha cambiado la interfaz de la aplicación. Ahora no se puede realizar llamadas de audio, sólo de audio/vídeo. Aunque sí se pueden recibir llamadas de audio de otros dispositivos.

Para evitar realizar forking se han implementado las GRUUs a la aplicación móvil, de este modo cuando un usuario se registra en el CORE IMS se le asocia una GRUU por cada dirección de contacto. De este modo un usuario puede estar registrado en dos o más terminales distintos y sólo querer transferir la sesión a un determinado dispositivo. Gracias a estar integrado como un AS en IMS, la movilidad de sesiones estaría disponible para otros servicios provistos por la plataforma IMS aparte del IPTV.

Se ha probado los dos modos de movilidad de sesión y se ha conseguido que funcionen correctamente en cualquier dispositivo móvil que disponga del sistema operativo Android, independientemente de la versión de éste.

Por otro lado se ha conseguido un valor aceptable en la medida del modo pull, hecho que no se ha logrado para el modo push. Para este caso obtenemos un valor demasiado alto a la hora de transferir la sesión.

Para el modo push se obtiene un elevado tiempo de transferencia de sesión. Como se ha mencionado en el capítulo anterior, parece probable que el terminal tenga programado algún tipo de retardo a la hora de aceptar una sesión entrante. Ya que al evaluar este modo de movilidad con SIPp se obtiene una medida más razonable.

Esto conlleva a seguir trabajando en este inconveniente y ver si de algún modo es posible reducir bastante el tiempo de transferencia de sesión para el modo push.

Otros posibles trabajos futuros relacionados con este proyecto serían:

- Por un lado continuar implementando otros tipos de movilidad como por ejemplo la movilidad de terminal, para que al cambiar de 3G a WIFI, se siga manteniendo la sesión.
- Por otra parte sería interesante que los datos de la sesión se enviaran de manera multicast, es decir, a todos aquellos usuarios que de alguna manera están subscritos a un mismo servicio. Con ello se consigue un servicio como el de la televisión IP. Siendo necesario modificar primeramente la librería doubango.
- Finalmente se debería portar la funcionalidad que se ha implementado en este proyecto en la última versión de la aplicación IMSDroid. De este modo se podría disfrutar de las nuevas mejoras que se han ido incorporando.

Apéndice A

Planificación del proyecto

A.1- División en actividades y tareas del proyecto.

El proyecto se puede dividir en un conjunto de actividades con sus respectivas tareas. A continuación se muestran las actividades y tareas del proyecto:

a) Documentación y estudio del estado del arte del proyecto.

   1. Búsqueda de bibliografía: Consiste en recopilar toda la información posible sobre el protocolo SIP y las diferentes movilidades de dicho protocolo. A dicha tarea se le ha asignado una duración de 7 jornadas.

   2. Estudio de la bibliografía: El objetivo de esta tarea es analizar la información obtenida, para conseguir los conocimientos precisos de modo que se pueda realizar el proyecto correctamente. La duración del estudio de la bibliografía es de 9 jornadas.

b) Familiarización con la aplicación IMSDroid.

   1. Estudio de la aplicación IMSDroid: Este trabajo se centra en el estudio del código de la aplicación para obtener un buen manejo de la misma. A dicha tarea se le ha asignado una duración de 6 jornadas.

   2. Simulación de la aplicación IMSDroid: Se pretende conocer todas las capacidades y funcionalidades de la aplicación. Este trabajo se realiza durante 3 jornadas.

c) Familiarización con la librería android-ngn-stack.

   1. Estudio de la librería android-ngn-stack: El objetivo es conocer el código de la librería para poder modificar dicho código cuando sea necesario. Se necesitan 6 jornadas para realizar esta tarea.

d) Familiarización con la librería doubango.

    1. Estudio de la librería doubango: Esta tarea se encarga de analizar cada directorio de la librería para adquirir soltura con el código de la misma y para conocer dónde y cómo se envían los mensajes del protocolo SIP. Esta librería contiene bastantes carpetas, por tanto un número elevado de ficheros, por lo que revisarlos todos ha retrasado 10 jornadas el tiempo dedicado.

    2. Realización del archivo de configuración y construcción de la librería doubango: Se pretende conocer la forma de construir la librería al cambiar parte de código, necesario para la implementación del proyecto. La duración de esta tarea es de 4 jornadas.

e) Desarrollo de la implementación.

    1. Programación para el modo pull: Se implementa el modo pull de la movilidad de sesión y se prueba el funcionamiento de este tipo de movilidad de sesión. Al ser el primer modo que se implementa requiere un tiempo mayor, aunque sea el modo más asequible de realizar. Por ello se ha necesitado una duración de 20 jornadas.

    2. Programación para el modo push: El objetivo de esta tarea es implementar el otro tipo de movilidad de sesión y probar el funcionamiento. La duración de esta tarea es de 16 jornadas.

    3. Cambio de la interfaz: Con esta tarea se pretende cambiar el aspecto de la aplicación. Esta operación sólo ha necesitado 3 jornadas.

f) Evaluación de la implementación.

    1. Realización de las pruebas de la implementación: Se pretende probar la implementación en función de las pruebas que se hayan decidido realizar. Han sido necesarias 4 jornadas.

g) Documentación y memoria del proyecto.

    1. Planificación de la redacción de la memoria: En esta tarea se planifica el índice de la memoria y el contenido de cada uno de ellos. Se emplean 2 jornadas para la realización de estos trabajos.

    2. Redacción de la memoria: El objetivo es redactar la memoria del proyecto. La duración de esta fase es de 28 jornadas.

    3. Revisión y corrección de la memoria: Una vez que se ha terminado de escribir la memoria, es necesario realizar una revisión y corregir los fallos. Esta tarea dura 7 jornadas.

A.2- Cuadro resumen de la planificación temporal del proyecto

Tarea	Descripción de la tarea	Duración
A	**Documentación y estudio del estado del arte del proyecto**	**16 jornadas**
A.1	Búsqueda de bibliografía	7 jornadas
A.2	Estudio de la bibliografía	9 jornadas
B	**Familiarización con la aplicación IMSDroid**	**9 jornadas**
B.1	Estudio de la aplicación IMSDroid	6 jornadas
B.2	Simulación de la aplicación IMSDroid	3 jornadas
C	**Familiarización con la librería android-ngn-stack**	**6 jornadas**
C.1	Estudio de la librería android-ngn-stack	6 jornadas
D	**Familiarización con la librería doubango**	**14 jornadas**
D.1	Estudio de la librería doubango	10 jornadas
D.2	Realización del archivo de configuración y construcción de la librería doubango	4 jornadas
E	**Desarrollo de la implementación**	**39 jornadas**
E.1	Programación para el modo pull	20 jornadas
E.2	Programación para el modo push	16 jornadas
E.3	Cambio de la interfaz	3 jornadas
F	**Evaluación de la implementación**	**4 jornadas**
F.1	Realización de las pruebas de la implementación	4 jornadas
G	**Documentación y memoria del proyecto**	**37 jornadas**
G.1	Planificación de la redacción de la memoria	2 jornadas
G.2	Redacción de la memoria	28 jornadas
G.3	Revisión y corrección de la memoria	7 jornadas
	TOTAL	**125 jornadas**

Tabla 1: Planificación temporal del proyecto

La duración total del proyecto es de 125 jornadas de 8 horas de trabajo según la planificación. Si se consideran 22 jornadas de trabajo al mes, se obtiene una duración total de 5 meses y 15 jornadas de trabajo.

Apéndice B

Presupuesto

En este apéndice se desarrolla el presupuesto asociado al proyecto con la colaboración de dos ingenieros senior.

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.- Autor: Ainhoa Sesmero Fernández

2.- Departamento: Telemática

3.- Descripción del Proyecto:

- Titulo	**Implementación de movilidad de sesión con el protocolo SIP**
- Duración (meses)	**8**
Tasa de costes Indirectos:	**20%**

4.- Presupuesto total del Proyecto (valores en Euros):

Euros	33.200

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a título informativo)	Categoría	Dedicación (hombres mes) [a)]	Coste hombre mes	Coste (Euro)
Sesmero Fernández, Ainhoa		Ingeniero	8	2.694,33	21.554,64
Sandonís Consuegra, Víctor		Ingeniero Senior	0,9	4.289,54	3.860,59
Calderón Pastor, María		Ingeniero Senior	0,5	4.289,54	2.144,77
					0,00
					0,00
				Total	27.560,00

[a)] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)

Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d)]
Ordenador de desarrollo y pruebas	800,00	100	8	60	106,67
Dos terminales Android	500,00	100	2	60	16,67
		100		60	0,00
		100		60	0,00
		100		60	0,00
					0,00
				Total	123,33

[d)] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} x C x D$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado

B = periodo de depreciación (60 meses)

C = coste del equipo (sin IVA)

D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
	Total	0,00

OTROS COSTES DIRECTOS DEL PROYECTO[e)]

Descripción	Empresa	Costes imputable
	Total	0,00

[e)] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas, otros,…

6.- Resumen de costes

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	27.560
Amortización	123
Subcontratación de tareas	0
Costes de funcionamiento	0
Costes Indirectos	5.537
Total	33.220

Apéndice C

Manual del Instalación

C.1- Eclipse, SDK y NDK

En primer lugar se descarga el paquete eclipse de Internet para Linux, por ejemplo se elige el paquete clásico. A continuación, se descomprime el paquete y se crea un directorio llamado *eclipse*. En dicho directorio se encuentran todos los ficheros del programa incluyendo el ejecutable. El ejecutable también se llama *eclipse*.

El siguiente paso es instalar el SDK. Para ello primero se descarga el paquete desde la web de Android. Hay diferentes paquetes dependiendo del sistema operativo del ordenador y el número de bits. Este paquete se desempaqueta en un directorio llamado android-sdk.linux_x86. No hay que olvidar la ubicación del directorio del SDK en el sistema porque habrá que hacer referencia más adelante.

Posteriormente se abre el programa Eclipse y se selecciona **Help → Install New Software** para instalar el plugin ADT. En la pantalla que aparece se hace clic en **Add**, situado la esquina superior derecha. Aparece el siguiente cuadro de diálogo.

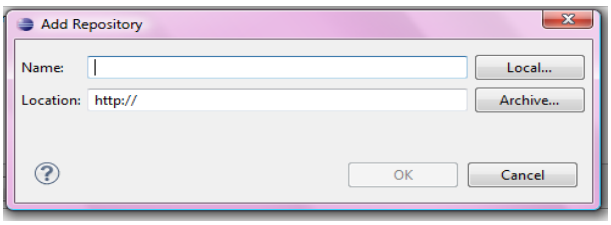


Figura 42: Cuadro de diálogo.

En name se introduce por ejemplo *ADT plugin* y, en location *https://dl-ssl.google.com/android/eclipse/*. Y después hacer clic en **Ok**.

Si existe algún problema para obtener el plugin, cambia en location *https* por *http* (https se prefiere por razones de seguridad).

En la ventana anterior aparece el paquete Developer Tools, hay que marcarlo y pulsar **Next**. En la siguiente ventana, aparece una lista de las herramientas de uso para ser

descargada pulsando **Next**. A continuación leemos y, se aceptan los acuerdos de licencia y se pulsa **Finish**. Cuando se finalice la instalación es necesario reiniciar Eclipse.

Seguidamente se procede a configurar el plugin ADT, especificando la ubicación del directorio de Android SDK.

Se selecciona **Help** → **Preferences**. En el cuadro de diálogo que aparece se elige **Android** desde el panel de la izquierda. Después, se pulsa **Browser** para localizar el directorio de Android SDK.

Antes de instalar Android NDK, hay que instalar las plataformas y los paquetes del SDK. Para ello se selecciona **Window** → **Android SDK Manager** o, se pulsa en la barra de herramientas el  icono . Aparece la siguiente ventana.

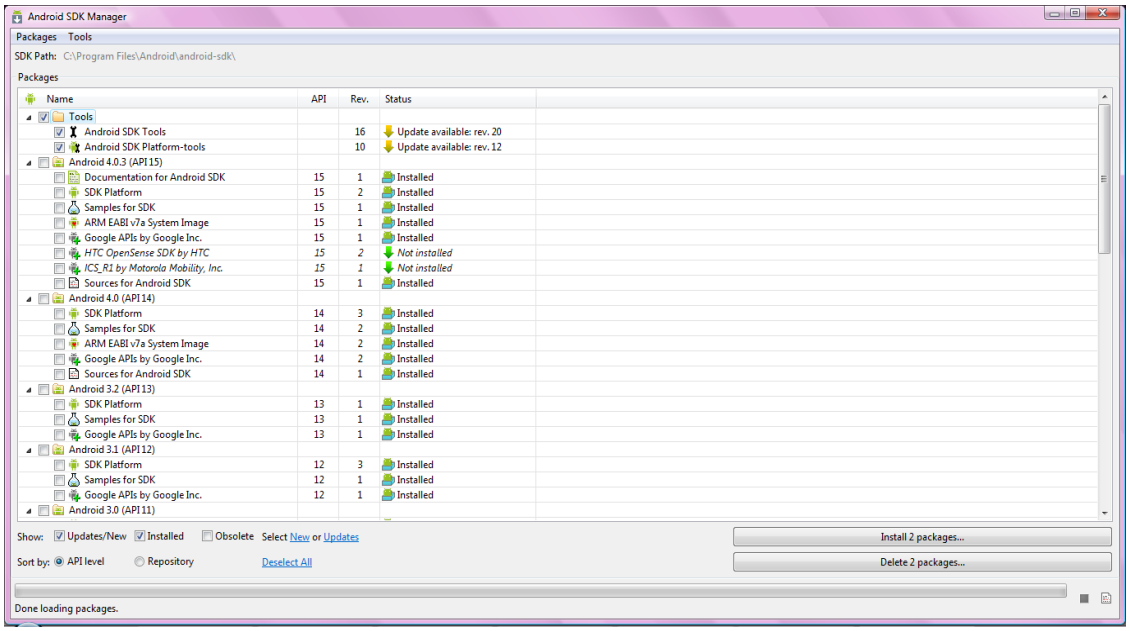


Figura 43: Android SDK Manager.

Automáticamente se seleccionan los paquetes que no se encuentran instalados. Sólo hay que pulsar **Install**.

Una vez que está instalado Eclipse y Android SDK se debe de editar el fichero `~/.bashrc` y añadir al final del documento las siguientes líneas:

```
export PATH=${PATH}: ../android-sdk-linux/tools
export PATH=${PATH}: ../eclipse
```

Para finalizar, es necesario instalar Android NDK, porque comprende un conjunto de herramientas que permiten poner en práctica las partes de la aplicación utilizando código nativo. Primero se descarga el paquete correspondiente dependiendo de la plataforma. Se descomprime el paquete en un directorio denominado *android_ndk_<version>* y ya está listo para comenzar a trabajar.

C.2- Apache, MySQL y PHP

En primer lugar, se recomienda actualizar los repositorios para tener nuestro sistema al día. Para ello se abre un nuevo terminal y se escriben las siguientes dos líneas:

```
1. ~$ sudo apt-get update
2. ~$ sudo apt-get upgrade
```

Después se instala la base de datos MySQL con la siguiente instrucción:

```
~$ sudo apt-get install mysql-server-4.1
```

Se cambia la contraseña porque por defecto se instala en blanco.

```
~$ sudo /usr/bin/mysqladmin –u root password pass
```

A continuación se instala el servidor Apache y el manejador de páginas dinámicas PHP5.

```
~$ sudo apt-get install apache2
~$ sudo apt-get install php5
```

Finalmente se instalan los archivos necesarios para que la base de datos tenga soporte con php5 y apache2.

```
~$ sudo apt-get install libapache2-mod-auth-mysql
~$ sudo apt-get install php5-mysql
```

Una vez que esta todo instalado se reinicia el servidor.

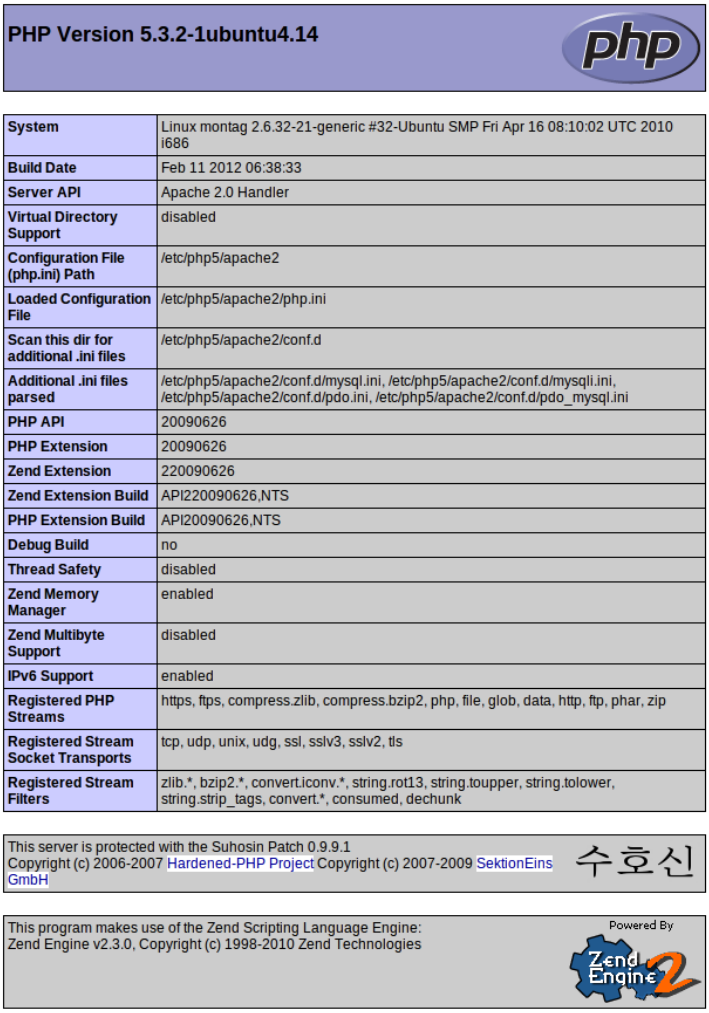
```
~$ sudo /ect/init.d/apache2 restart
```

Para comprobar que todo funciona correctamente se crea un fichero de prueba con el siguiente código.

```
~$ sudo gedit /var/www/test.php

        <?
        phpinfo();
        ?>
```

Se abre un navegador y se escribe la siguiente url: *http://localhost/test.php*, y debe aparecer la siguiente página web.



PHP Version 5.3.2-1ubuntu4.14

System	Linux montag 2.6.32-21-generic #32-Ubuntu SMP Fri Apr 16 08:10:02 UTC 2010 i686
Build Date	Feb 11 2012 06:38:33
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
Additional .ini files parsed	/etc/php5/apache2/conf.d/mysql.ini, /etc/php5/apache2/conf.d/mysqli.ini, /etc/php5/apache2/conf.d/pdo.ini, /etc/php5/apache2/conf.d/pdo_mysql.ini
PHP API	20090626
PHP Extension	20090626
Zend Extension	220090626
Zend Extension Build	API220090626,NTS
PHP Extension Build	API20090626,NTS
Debug Build	no
Thread Safety	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
Registered PHP Streams	https, ftps, compress.zlib, compress.bzip2, php, file, glob, data, http, ftp, phar, zip
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, sslv3, sslv2, tls
Registered Stream Filters	zlib.*, bzip2.*, convert.iconv.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk

Figura 44: Web de información del servidor.

Apéndice D

Manual de la librería Doubango

D.1- Descarga del código

Antes de comenzar, hay que descargarse el código de la librería, por lo que se abre un terminal y se escribe

```
svn checkout http://doubango.googlecode.com/svn/branches/2.0/doubango
doubango-read-only
```

Una vez que el código de la librería se ha descargado se procede a modificar algunos ficheros para que se pueda desarrollar la transferencia de sesiones. Los ficheros que se han modificado se han mencionado en el capítulo 4 de la memoria.

Otra manera más cómoda de obtener el código de la librería es por medio del CD entregado con el código del proyecto.

D.2- Ajuste del fichero root.mk

A continuación es necesario ajustar el fichero root.mk. En él se indicará la ubicación de Android SDK y NDK, entre otros. Hay que seguir los siguientes pasos:

1. Ir a **$DOUBANGO_HOME/android-projects** y abrir el fichero **root.mk**.
2. Modificar la variable **$ANDROID_NDK_ROOT** por la ubicación del directorio raíz del NDK.
3. Modificar la variable **$ANDROID_SDK_ROOT** por la ubicación del directorio raíz del SDK.
4. Cambiar la variable **$ANDROID_PLATFORM** al directorio raíz de la plataforma android que se prefiera (por ejemplo android-5).

5. Modificar la variable **$ANDROID_GCC_VER** con la versión gcc que se tenga instalada.

6. La variable **$ANDROID_HOST** depende del sistema operativo, si es Windows XP/Vista/7 tendrá como valor **Windows**, si es Unix **Linux-86** y si es MAC OS X, **Darwin-x86**.

7. Abrir un nuevo terminal y añadir el directorio raíz binario al **$PATH** del sistema.

```
exportPATH=$ANDROID_NDK_ROOT/toolchains/arm-linux-
androideabi-4.4.3/prebuilt/linux-x86/bin:$PATH
```

8. Modificar la variable **$CFLAGS** con el valor que se prefiera, por ejemplo:

```
export CFLAGS="-Os -DDEBUG_LEVEL=DEBUG_LEVEL_ERROR"
```

D.3- Compilación y construcción de la librería

Para crear la librería libtinyWRAP.so deben de seguirse los siguientes pasos:

1. En el caso de que se haya modificado el código que se encuentra en la carpeta **$DOUBANGO_HOME/bindings/_common**, debe de generarse el código nuevo para todos los tipos de lenguajes de programación. Para ello es preciso ir al siguiente directorio:

   ```
   cd $DOUBANGO_HOME/bindings
   ```

   Después hay que construir todos los ficheros para los distintos lenguajes de programación.

   ```
   sh autogen.sh
   ```

   No olvidar de añadir los ficheros cambiados a *android-ngn-stack/src/org/doubango/tinyWRAP*.

   Al generar los nuevos ficheros es posible que el archivo **tinyWRAPJNI** que se encuentra en **$DOUBANGO_HOME/bindings/java/android** se haya modificado, por lo que hay que añadirlo al directorio *android-ngn-stack/src/org/doubango/tinyWRAP*.

2. Ir al directorio android-projects:

   ```
   cd $DOUBANGO_HOME/android-projects
   ```

3. Construir todos los proyectos:

   ```
   ../bindings/java/android/buildAll.sh
   ```

4. Se genera un archivo en **$DOUBANGO_HOME/android-projects/output** llamado **libtinyWRAP_armv5te.so**. Hay que cambiar

el nombre del fichero por **libtinyWRAP.so**. y, mover dicho fichero a **android-ngn-stack/libs/armeabi**.

5. Hay que construir de nuevo todos los proyectos pero habilitando el parámetro NEON:

```
../bindings/java/android/gpl.sh
```

6. Al ejecutar la instrucción anterior se genera un archivo en **$DOUBANGO_HOME/android-projects/output** denominado **libtinyWRAP_armv7te.so**, se cambia el nombre del fichero por **libtinyWRAP.so**, y se mueve este fichero a **android-ngn-stack/libs/armeabi-v7a**.

Es necesario construir todos los proyectos habilitando el parámetro NEON y deshabilitando, porque el parámetro NEON depende de la versión de android del dispositivo. Para versiones inferiores o iguales a 2.3.4 es necesario que el parámetro NEON esté deshabilitado. Mientras que para versiones superiores es necesario que se encuentre habilitado, porque en caso contrario los nuevos métodos no se podrán ejecutar en estos dispositivos y, la aplicación no funcionará.

Apéndice E

Manual de la aplicación IMSDroid

Antes que nada es necesario tener el código de la aplicación por si se desea realizar algún cambio. Dicho código se puede obtener de dos maneras. La primera de ellas mediante el CD que se ha adjuntado con este proyecto o descargándolo mediante el siguiente comando:

```
svn checkout http://imsdroid.googlecode.com/svn/branches/2.0
nombre_directorio
```

Una vez se ha descargado el código es preciso realizar los cambios que se han explicado en el capítulo 4 para que se pueda realizar la transferencia de sesión.

E.1- Configuración

Para poder registrar un dispositivo y de este modo utilizar la aplicación es necesario configurar la identidad del terminal y los ajustes de red. Para ello es preciso ir a la pantalla de Options y de ahí a Identity o a Network.

E.1.1- Identidad

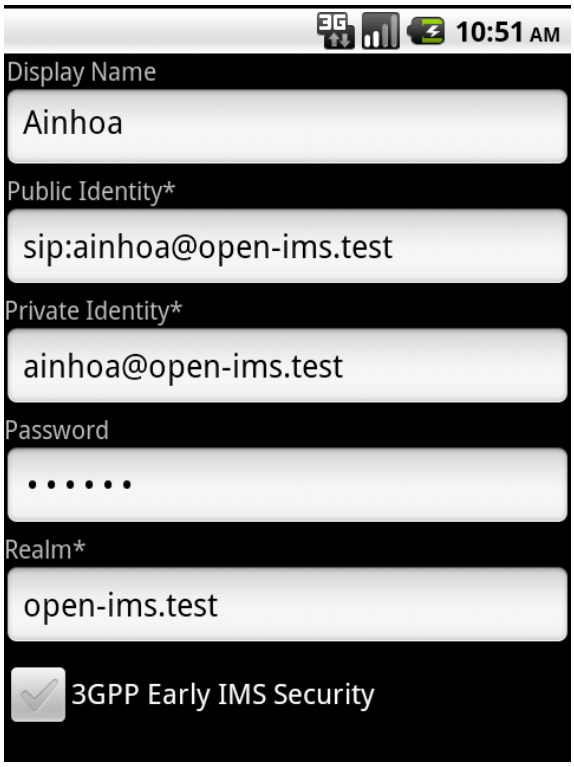


Figura 45: Pantalla de identidad.

En la pantalla de identidad se encuentran las siguientes opciones:

- Display Name: El nickname.
- Public Identity: El identificador público que se utilizar para recibir y realizar llamadas.
- Private Identity: El identificador único asignado a cada usuario.
- Password: La contraseña.
- Realm: El dominio de autenticación.
- 3GPP Early IMS Security: Si no se utiliza un servidor IMS se puede usar esta opción para desactivar algunos procedimientos pesados de autenticación de IMS.

E.1.2- Ajuste de red

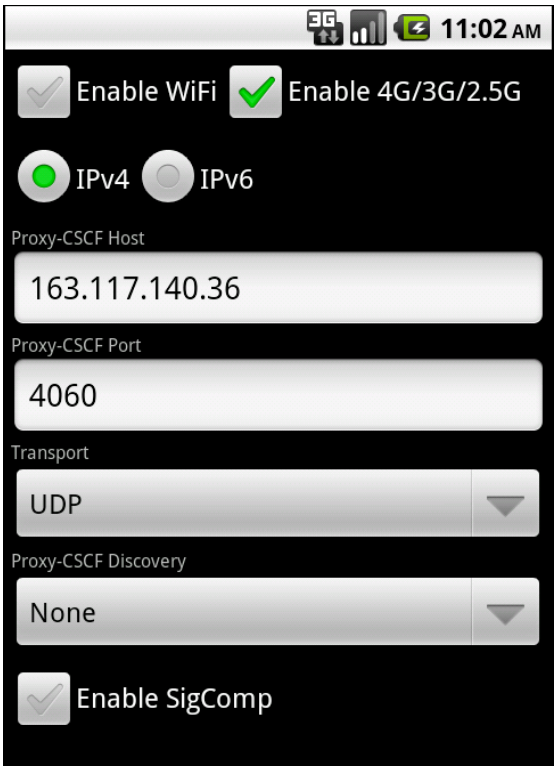


Figura 46: Pantalla de ajuste de red.

Como ocurría con la pantalla de configuración de la identidad, en esta pantalla también existen distintas opciones.

- Enable Wifi: Para habilitar el Wifi.
- Enable 4G/3G/2,5G: Para habilitar 4G (LTE), 3G(UMTS) o 2,5G(EDGE). Esta opción debe de estar habilitada cuando se utiliza el emulador.
- IPv4 o IPv6: Define la versión IP del proxy-CSCF. Si no está seguro, marcar IPv4.
- Proxy-CSCF Host: Es la dirección IP del servidor SIP o el proxy de salida.
- Proxy-CSCF Port: Es el puerto asociado al proxy-CSCF.
- Transport: El protocolo de transporte. Hay dos opciones, UDP y TCP.
- Proxy-CSCF Discovery: Debería estar en None.
- Enable SigComp: Si está habilitado permite la comprensión de señalización.

Apéndice F

Manual de MySQL

F.1- Conectarse y desconectarse del servidor

Es necesario conectarse al servidor mediante un nombre de usuario y una contraseña. Se debe especificar el nombre de host cuando el servidor se está ejecutando en otro ordenador diferente a donde está establecida la conexión. Se abre un terminal nuevo y se escribe:

```
~$ mysql –h host –u user –p

~$ Enter password: *****
```

user representa el nombre de usuario y *host* el nombre del ordenador donde se está ejecutando el servidor. Si todo ha ido correctamente aparecerá el prompt *mysql>*. Para conectarse a la base de datos de este proyecto hay que escribir lo siguiente:

```
~$ mysql –u root –p

~$ Enter password: sesmer88
```

Para desconectarse del servidor simplemente basta con la instrucción *exit*.

F.2- Crear y utilizar una base de datos

Se pueden tener varias bases de datos para diferentes casos. Para conocer las bases de datos que existen actualmente se utiliza la siguiente instrucción:

```
mysql> SHOW DATABASES;
+----------+
| Database |
+----------+
| mysql    |
| test     |
| tmp      |
+----------+
```

Figura 47: Mostrar las bases de datos disponibles.

Para poder acceder a una base de datos es precisa la siguiente sentencia:

```
mysql> USE test
Database changed
```

Figura 48: Sentencia para acceder a una base de datos.

Otro comando importante es aquel que permite crear una base de datos.

```
mysql> USE test
Database changed
```

Figura 49: Sentencia para crear una base de datos.

F.3- Crear y utilizar las tablas

Como ocurre para las bases de datos, también es posible ver todas las tablas de una base de datos.

```
mysql> SHOW TABLES;
Empty set (0.00 sec)
```

Figura 50: Sentencia para mostrar las tablas.

Si se desea crear una tabla nueva se usa la sentencia *CREATE TABLE*, especificando la estructura de dicha tabla.

```
mysql> create table nombre_tabla (nombre_atributo
  tipo_atributo, nombre_atributo tipo_atributo);
```

Un ejemplo de creación de una tabla es el que se muestra a continuación:

```
mysql> CREATE TABLE pet (name VARCHAR(20), owner VARCHAR(20),
    -> species VARCHAR(20), sex CHAR(1), birth DATE, death DATE);
```

Figura 51: Ejemplo de creación de una tabla.

Las tres sentencias que más se utilizan para gestionar las tablas son:

1. Insertar un nuevo elemento en la tabla.

   ```
   mysql>insert into nombre_tabla values(valor_atributo,
   valor_atributo, …);
   ```

2. Seleccionar un elemento de la tabla.

   ```
   mysql> select * from nombre_tabla;
   ```

   Con esta sentencia se muestran todos los elementos de la tabla, si se desea mostrar sólo un elemento hay que añadir a la sentencia anterior la palabra *WHERE* con el nombre del atributo seguido del valor del atributo que se desee filtrar.

   ```
   mysql> select * from nombre_tabla where
   nombre_atributo=valor_atributo;
   ```

3. Eliminar un elemento de la tabla.

   ```
   mysql>delete from nombre_tabla;
   ```

   Esta instrucción elimina todos los registros de la tabla. También se puede utilizar la palabra *WHERE* como en el caso anterior, para elegir qué registro se desea eliminar de la tabla.

Apéndice G

Código del CAS

G.1- Ficheros de comunicación

A parte de los distintos archivos existentes para cada comunicación con el CAS, se ha creado una librería que contiene una función [23]. Esta función es capaz de convertir la respuesta de una petición de la base de datos al formato del lenguaje XML. La librería se muestra a continuación:

```
<?php
//Esta función se encarga de convertir a un documento XML la respuesta de la base de datos
//Tiene como parámetros
//@param $resultado, la respuesta de la base de datos
//@param $nombreDoc, nombre del documento, por defecto toma el valor de information
//@param $nombreItem, nombre de cada ítem del documento, por defecto toma el valor de session_number
function mysql_XML ($resultado, $nombreDoc='information', nombreItem='session_number') {
$campo = array();
    //Llenamos el array de nombres de campos
for ($i=0; $i<mysql_num_fields ($resultado); $i++)
        $campo[$i] = mysql_field_name ($resultado, $i);
//Creamos el documento XML
$dom = new DOMDocument ('1.0', 'UTF-8');
//Creamos el primer hijo del documento, del que colgaran otros hijos con la información de la base de datos
$doc = $dom->appendChild ($dom->createElement ($nombreDoc));
//Recorremos el resultado
for ($i=0; $i<mysql_num_rows ($resultado); $i++) {
//Creamos el ítem
$nodo = $doc->appendChild ($dom->createElement ($nombreItem));
//Agregamos los campos que corresponden
for ($b=0; $b<count ($campo); $b++) {
$campoTexto = $nodo->appendChild ($dom->createElement ($campo[$b]));
```

```
                $campoTexto->appendChild ($dom->createTextNode (mysql_result ($resultado, $i,
$b)));
}
    }
    //Retomamos el archivo XML como cadena de texto
$dom->formatOutput = true;
return $dom->saveXML();
}
?>
```

G.1.1- Modo pull

G.2.1.1- Establecimiento de la sesión

```
<?php
//Configuración de la base de datos
//host-> dirección IP del servidor, localhost
//user->nombre del usuario, root
//password->contraseña del usuario, sesmer88
$db = mysql_connect (host,user,password);
mysql_select_db ("imsdroid",$db);
//Se almacenan los datos que envía el cliente
$postdata = $HTTP_RAW_POST_DATA;
//Los datos que se reciben se convierte en un string xml
$xmlstring = <<<XML
$postdata
XML;
//Interpreta un string xml en un objeto
$xml = simplexml_load_string ($xmlstring);
//Se cogen todos los datos que manda el cliente uno a uno
$from = $xml->from;
$from_tag = $xml->from_tag;
$to = $xml->to;
$to_tag = $xml->to_tag;
$call_id = $xml->call_id;
$codec = $xml-> codec;
$type_media = $xml->type_media;
//Se inserta en las tablas
$sql_user_sip = mysql_query ("insert into user_sip (sip_from,sip_to,call_id) values
('{$from}','{$to}','{$call_id}')",$db);
$sql_information = mysql_query ("insert into information
(call_id,tag_from,tag_to,codecs,type_media) values
('{$call_id}','{$from_tag}','{$to_tag}','{$codec}','{$type_media}')",$db);
?>
```

G.1.1.2- Finalización de la sesión

```
<?php
//Configuración de la base de datos
//host-> dirección IP del servidor, localhost
//user->nombre del usuario, root
//password->contraseña del usuario, sesmer88
$db = mysql_connect (host,user,password);
mysql_select_db ("imsdroid",$db);
//Se almacenan los datos que envía el cliente
$postdata = $HTTP_RAW_POST_DATA;
//Se convierte a un string xml
$xmlstring = <<<XML
$postdata
XML;
//Interpreta un string xml en un objeto
$xml = simplexml_load_string ($xmlstring);
//Se coge el call-id que manda el cliente
$call_id = $xml->call_id;
//Se elimina las tablas
$sql_user_sip = mysql_query ("delete from user_sip where call_id = '{$call_id}';",$db);
$sql_information = mysql_query ("delete from information where call_id = '{$call_id}';",$db);
?>
```

G.1.1.3- Petición de todas las sesiones establecidas

```
<?php
//Se incluye la librería
include "lib.php";
//Configuración de la base de datos
//host-> dirección IP del servidor, localhost
//user->nombre del usuario, root
//password->contraseña del usuario, sesmer88
$db = mysql_connect (host,user,password);
mysql_select_db ("imsdroid",$db);
//Se cogen todos los registros de la tabla user_sip
$sql = mysql_query ("select * from user_sip;");
//Se devuelven todos estos registros mediante el lenguaje XML
echo mysql_XML ($sql);
?>
```

G.1.1.4- Información de la sesión

```
<?php
//Se incluye la librería
include "lib.php";
```

```
//Configuración de la base de datos
//host-> dirección IP del servidor, localhost
//user->nombre del usuario, root
//password->contraseña del usuario, sesmer88
$db = mysql_connect (host,user,password);
mysql_select_db ("imsdroid",$db);
//Se almacenan los datos que envía el cliente
$postdata = $HTTP_RAW_POST_DATA;
//Se convierten los datos en un string xml
$xmlstring = <<<XML
$postdata
XML;
//Se interpreta un string xml en un objeto
$xml = simplexml_load_string ($xmlstring);
//Se almacena el call-id que envía el cliente
$call_id = $xml->call_id;
//Se pide a la base de datos toda la información de ese call-id
$sql = mysql_query ("select * from information where call_id = '{$call_id}';");
//Se envía la información al cliente mediante el lenguaje XML
echo mysql_XML ($sql);
?>
```

G.1.2- Modo Push

G.1.2.1- Registro de un usuario

```
<?php
//configuración de la base de datos
//host-> dirección IP del servidor, localhost
//user->nombre del usuario, root
//password->contraseña del usuario, sesmer88
$db = mysql_connect (host,user,password);
mysql_select_db ("imsdroid",$db);
//Se almacenan los datos que envía el cliente
$postdata = $HTTP_RAW_POST_DATA;
//Se convierten los datos en un string xml
$xmlstring = <<<XML
$postdata
XML;
//Se interpreta el string xml en un objeto
$xml = simplexml_load_string ($xmlstring);
//Se almacena la información que envía el cliente
$sip_uri = $xml->sip_uri;
$gr = $xml->gr;
$display = $xml->display;
//Se inserta en la tabla dicha información
$sql_user_sip = mysql_query ("insert into register (sip_uri,gr, display) values
('{$sip_uri}','{$gr}','{$ display})",$db);
```

```
?>
```

G.1.2.2- Desregistro de un usuario

```
<?php
//Configuración de la base de datos
//host-> dirección IP del servidor, localhost
//user->nombre del usuario, root
//password->contraseña del usuario, sesmer88
$db = mysql_connect (host,user,password);
mysql_select_db ("imsdroid",$db);
//Se almacenan los datos que envía el cliente
$postdata = $HTTP_RAW_POST_DATA;
//Se convierte a un string xml
$xmlstring = <<<XML
$postdata
XML;
//Se interpreta en un objeto
$xml = simplexml_load_string ($xmlstring);
//Se almacenan los datos del cliente
$gr = $xml->gr;
//Se borra el registro con ese gr
$sql= mysql_query ("delete from register where gr = '{$gr}';",$db);
?>
```

G.1.2.3- Petición de todos los registros

```
<?php
//Se incluye la librería
include "lib.php";
//Configuración de la base de datos
//host-> dirección IP del servidor, localhost
//user->nombre del usuario, root
//password->contraseña del usuario, sesmer88
$db = mysql_connect (host,user,password);
mysql_select_db ("imsdroid",$db);
//Se cogen todos los registros
$sql = mysql_query ("select * from register;");
//Devuelve la petición utilizando el lenguaje XML
echo mysql_XML ($sql);
?>
```

Bibliografía

[1] SIP: Understanding the Session Initiation Protocol, 3rd Edition, Alan B. Johnston, Artech House 2009.

[2] SIP Home page: http://www.cs.columbia.edu/sip.

[3] The IMS: IP Multimedia Concepts and Services in the Mobile Domain, 3rd Edition, Mikka Poikselka, Georg Mayer, Hisham Khartabil, Aki Niemi, 2004.

[4] 3GPP, IP Multimedia call control based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3, TS 24.229 v9.8.0 Release 9, 3rd Generation Partnership Proyect (3GPP), June 2011.

[5] Henning Schulzrinne and Elin Wedlund, "Application-Layer Mobility Using SIP", Mobile Computing and Communication Review, 1 (2):47-57, 2000.

[6] Iván Vidal, Antonio de la Oliva, Jaime Gracia-Reinoso and Ignacio Soto. 2011. TRIM: An architecture for transparent IMS-based mobility. Comput. Netw 55, 7 May
2011.

[7] R. Schacham, H. Schulzrinne, S. Thaholsri and W. Kellerer, "Session Initiation Protocol (SIP) Session Mobility", RFC 5631 (Informational), October 2009.

[8] R. Sparks, Tekelec and A. Johnston, "Session Initiation Protocol (SIP) Call Control-Transfer", RFC 5589 (Proposed Best Current Practice), June 2009.

[9] 3GPP, IP Multimedia Subsystem (IMS) service continuity; Stage 2, TS 23.237 v11.1.0 Release 11, 3rd Generation Partnership Proyect (3GPP), June 2011.

[10] P. Kyzivat, "Registration Event Package Extension for Session Initiation Protocol (SIP) Globally Routable User Agent URIs (GRUUs)", RFC 5628 (Proposed Standard), October 2009.

[11] J. Rosenberg, "Obtaining and Using Globally Routable User Agent URIs (GRUUs) in the Session Initiation Protocol (SIP)", RFC 5627 (Proposed Standard), October 2009.

[12] FOCUS GROUP ON IPTV, 3rd FG IPTV meeting: Mountain View, USA, 22-26 January 2007, "Working Document: IPTV Architecture".

[13] R. Sparks, "The Session Initiation Protocol (SIP) Refer Method", RFC 3515 (Proposed Standard), April 2003.

[14] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Primaria, "SIP: Session Initiation Protocol", RFC 3261 (Proposed Standard), June 2002.

[15] R. Mahy, B. Biggs, R. Dean, "The Session Initiation Protocol (SIP) "Replaces" Header", RFC 3891 (Proposed Standard), September 2004.

[16]SIP: http://datatracker.ietf.org/wg/sip/charter/.

[17] IMSDroid: http://code.google.com/p/imsdroid/.

[18] Doubango: http://www.doubango.org/.

[19] Descarga del eclipse: http://www.eclipse.org/downloads/?osType=linux.

[20] Tutorial de Android: http://developer.android.com.

[21] Tutorial de instalación de Apache+MySql+Php5: http://www.ubuntu-es.org/node/91538.

[22] Tutorial de MySql: http://dev.mysql.com/doc/refman/5.0/es/tutorial.html.

[23] Tutorial de XML: http://www.w3schools.com/xml/.

[24] Tutorial de PHP: http://www.w3schools.com/php/.

[25] Petición POST en java: http://blogdavidrodriguez.piensaennaranja.com/2009/03/27/lanzar-una-peticion-post-en-java/.

[26] Convertir la respuesta de la base de datos a xml utilizando php: http://www.cristalab.com/tutoriales/crear-un-archivo-xml-con-un-resultado-de-mysql-en-php-c68404l/.

[27] Open IMS Core, Fraunhofer Institute for Open Communication System: http://www.openimscore.org.