Escuela Politécnica Superior

Grado en Ingeniería de Sistemas Audiovisuales

Trabajo Fin de Grado

“Infraestructura de señalización para proporcionar servicios audiovisuales en un emulador de redes”

AUTOR: Eva Mª Urbán Martínez

TUTOR: Ignacio Soto Campos

Leganés, Junio de 2017

Título: Infraestructura de señalización para proporcionar servicios audiovisuales en un emulador de redes.

Autor: Eva Mª Urbán Martínez.

Director: Ignacio Soto Campos.

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Trabajo Fin de Carrera el día __ de _____ del 2017 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de ___.

SECRETARIO VOCAL PRESIDENTE

AGRADECIMIENTOS

En primer lugar, a mi tutor Ignacio, gracias por tu infinita paciencia guiándome en este largo camino.

A Carmelo, por ser la persona más bonita con la que tengo la suerte de compartir mi vida. Gracias por recordarme que iba a poder con esto.

A mis padres, que son un ejemplo a seguir en todo lo que hacen. Gracias por aguantar mis momentos de pánico y por recordarme siempre lo que valgo.

A mis amigas, las de siempre, gracias por estar ahí a pesar de los kilómetros.

Y a toda la gente que he conocido en estos cinco años, gracias por compartir conmigo esta experiencia. Me llevo de vuelta a casa un pedacito de cada uno.

RESUMEN

La convergencia hacia un mundo "todo sobre IP" ha llevado al desarrollo de nuevas tecnologías que han transformado el mercado de las telecomunicaciones. Un ejemplo significativo es la arquitectura IMS (IP Multimedia Subsystem). Esta arquitectura facilita servicios muy demandados por los usuarios como son el tráfico de voz, vídeo y multimedia, todo ello mediante la conmutación de paquetes basada en el protocolo IP. IMS implementa un protocolo, SIP (Session Initiation Protocol), que permite señalizar y administrar las sesiones multimedia.

A la hora del desarrollo de aplicaciones multimedia es necesario validar su comportamiento. Una herramienta muy útil, cuyo uso se está extendiendo, son los emuladores de redes, que permiten, usando máquinas virtuales, emular una red y usar el mismo software que se usa en la red real. Además tienen funcionalidades tan completas como la interconexión de las redes virtuales a la red real. CORE (Common Open Research Emulator) es un ejemplo de emulador de redes muy versátil y con muchas funcionalidades.

En este proyecto se ha dotado al emulador de redes CORE de un sistema IMS para poder hacer experimentos en el emulador con servicios audiovisuales basados en señalización SIP. Este trabajo ha requerido el conocimiento y manejo de distintas tecnologías de redes, y de máquinas virtuales.

Las pruebas se han realizado con dos clientes IMS diferentes, uno que dispone de interfaz gráfica de usuario (Monster) y otro a través de scrips de SIPp. Además se ha capturado el tráfico de mensajes SIP para comprobar el funcionamiento del sistema, concluyendo que se han logrado los objetivos marcados, es decir, se consigue desarrollar una plataforma donde emular cualquier tipo de red y dotarla de un sistema IMS con el que se puede añadir señalización SIP. Esta plataforma es adaptable a cualquier red bajo estudio, de forma que se podrá utilizar en el estudio, desarrollo e implantación de servicios multimedia a través de redes IP.

Palabras clave: emulador, CORE, IMS, IP, multimedia, red, señalización, SIP, "todo sobre IP", virtual.

ABSTRACT

The convergence to an “all-IP” world has led to the development of new technologies that have transformed the telecommunication market. The IMS (IP Multimedia Subsystem) architecture is an important example. This architecture incorporates services demanded by users such as voice, video and multimedia traffic, all for packet switched networks based on the IP protocol. IMS implements a protocol, Session Initiation Protocol (SIP) that allows signalling and management of multimedia sessions.

To develop multimedia applications it is necessary to validate their behaviour. Network emulators are a useful tool which allow using virtual machines, emulate a network and use the same software that is used in the real network. In addition, they have functionalities like interconnection of virtual networks to the real network. CORE (Common Open Research Emulator) is an example of a very versatile network emulator with many functionalities.

In this project the CORE network emulator has been equipped with an IMS system to be able to offer audio-visual services based on SIP signalling. This work has required the knowledge and management of different network technologies, and of virtual machines.

The testing of the platform have been done with two different IMS clients, one that has a graphical user interface (Monster) and another through SIPp scripts. In addition, the traffic of SIP messages has been captured to check the operation of the system, concluding that the objectives have been achieved, is possible to develop a platform to emulate any type of network and equip it with an IMS system with SIP signalling. This platform is adaptable to any network under study for the development and implementation of multimedia services over IP networks.

Keywords: “all-IP”, emulator, CORE, IMS, IP, multimedia, network, signalling, SIP, virtual.

CONTENIDO

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

ACRÓNIMOS

3GPP 3rd Generation Partnership Project

AS Application Server *(Servidor de Aplicación)*

BSD Berkeley Software Distribution

CAMEL Customized Application for mobile-Network for Enhanced Logic

CDR Call Details Record

CMT Comisión del Mercado de las Telecomunicaciones

CNMC Comisión nacional de los mercados y a la competencia

CORE Common Open Research Emulator

CSCF Call State Control Function

DHCP Dynamic Host Configuration Protocol

DNS Domain Name System

E-CSCF Emergency Call State Control Function

ETSI European Telecommunications Standards Institute

FOKUS Fraunhofer institute Für Offene Kommunications Sisteme

GNS3 Graphical Network Simulator

GSM Global System for Mobile communications (*sistema estándar de telefonía)*

GSMA Global System for Mobile communications Asociation

GUI Graphical User Interface *(Interfaz Gráfica de Usuario)*

HSS Home Subscriber Server

HTTP Hypertext Transfer Protocol

IM-SSF IP Multimedia – Service Switching Function

IMS IP Multimedia Subsystem

I-CSCF Interrogating –Call Session Control Function

IETF Internet Engineering Task Force

IP Internet Protocol *(Protocolo de Internet)*

IP IMS IP Media Server *(Servidor de Media IP)*

MGC Media Gateway Controller

MGW Media Gateway

MRF Multimedia Resource Function

MRFP Multimedia Resource Function Controller

NAT Network Address Translation

NGN Next Generation Network (Redes de Siguiente *Generación)*

QoS Quality of Service *(Calidad de Servicio)*

RDP Remote Desktop Protocol

RTP Real-time Transport Protocol

S-CSCF Serving–Call Session Control Function

SDP Session Description Protocol

SGW Signalling Gateway Function

SIP Session Initiation Protocol

SIP-AS SIP – Application Server

SS7 Signalling System Nº. 7

TISPAN Telecommunication and Internet Converged Services and Protocols for Advanced Networks

P2P Peer-To-Peer

P-CSDF Proxy - Call Session Control Function

PUEL Personal Use and Evaluation License

Tcl/Tk Tool Command Language / Tool Kit

TIC Tecnologías de la Información y las Comunicaciones

TISPAN Telecommunication and Internet Converged Services and Protocols for Advanced Networks

OCA-SCS Open Service Access – Service Capability Server

OSE Open Source Edition

OSPF Open Shortest Path First *(Primer Camino Más Corto)*

RCS Servicios de Comunicación enriquecida

UE	User Equipment
UAC	User Agent Client
UAS	User Agent Server
URI	Universal Resource Identifier
VoIP	Voice over IP
VoLTE	Voice over Long Term Evolution

1. INTRODUCCIÓN Y OBJETIVOS

1.1. Contexto y motivación

El avance de Internet ha ido permitiendo desde hace varios años el uso de numerosos servicios exitosos, tales como el correo electrónico, el streaming de audio y vídeo, los intercambios de mensajes instantáneos o "chat" y otras múltiples aplicaciones.

Estos avances han llevado a los operadores de telecomunicaciones a reposicionar su actividad alrededor de las aplicaciones sobre IP (Internet Protocol), incluyendo telefonía y otros servicios audiovisuales que tradicionalmente se venían ofreciendo mediante otras redes. En este punto, el despliegue de servicios que proporcionen la infraestructura para la señalización que requieren este tipo de servicios se convierte en una cuestión estratégica para los operadores de telecomunicaciones. IMS (IP Multimedia Subsystem) es el mejor ejemplo de este tipo de arquitecturas, desarrollado y estandarizado por 3GPP [1] (3rd Generation Partnership Project) para redes móviles y por ETSI TISPAN [2] (European Telecommunications Standards Institute, Telecommunication and Internet Converged Services and Protocols for Advanced Networks) para redes fijas.

La incorporación de IMS en las redes fijas y móviles representa un cambio fundamental en las redes de telecomunicaciones de tipo voz. IMS es una arquitectura que pretende soportar tráfico multimedia (voz, vídeo, presencia…) mediante redes IP, y con independencia del medio de acceso: teléfonos móviles, fijos, computadores… en definitiva, para cualquier dispositivo IP de la red. Aunque los servicios que se pueden ofrecer sobre IMS son bastante innovadores, los fundamentos de dicha arquitectura son tan solo una nueva aplicación de una tecnología ya existente denominada SIP (Session Initiation Protocol). Este protocolo permite la señalización y administración de sesiones multimedia.

Debido a la importancia de las aplicaciones multimedia para los operadores de telecomunicación, son necesarias herramientas que permitan estudiar el despliegue de servicios IMS en redes de la forma más sencilla y realista posible. Los emuladores de redes en general, y CORE (Common Open Research Emulator) [3] en particular, son una buena opción para conseguir este fin. CORE permite emular múltiples escenarios de red en los que probar implementaciones de protocolos, además de la posibilidad de conectar estas redes virtuales a redes reales.

En este proyecto se quiere integrar IMS en CORE y así obtener un entorno muy flexible en el que probar soluciones basadas en IMS en entornos reales de red sin la complejidad de despliegues de equipos físicos reales.

1.2. Objetivos

El objetivo es dotar a un emulador de redes, CORE, de un sistema IMS para proporcionar servicios audiovisuales.

El alcance del proyecto recoge los siguientes puntos:

- Describir las características de IMS, así como las ventajas que ofrece.
- Describir las características de CORE, así como las ventajas que ofrece.
- Integrar una implementación de IMS con el CORE, de manera que sea posible acceder a servicios multimedia en los nodos virtuales del CORE.
- Desplegar en el CORE dos clientes IMS, uno con interfaz gráfica de usuario y otro a través de scripts de SIPp.
- Probar el funcionamiento de servicios multimedia en redes de prueba en el CORE usando la integración con la arquitectura IMS.

1.3. Estructura del documento

Para poder facilitar la comprensión de la memoria, se detalla, a continuación, la estructura de la misma:

- **Bloque 1: Introducción.**
  En este primer bloque se exponen las motivaciones del proyecto y se establecen los objetivos del mismo. Además, se detalla la estructura de la memoria.

- **Bloque 2: Estado del arte y alternativas de diseño.**
  Este segundo bloque está dedicado al estudio del estado del arte, donde se detallan las tecnologías utilizadas en el desarrollo del proyecto. Además se hace un análisis de las posibles alternativas de cada tecnología y del enfoque para el diseño de la solución del proyecto.

- **Bloque 3: Integración de IMS en el emulador CORE.**
  En este bloque se explican todos los procedimientos implementados para la realización práctica del proyecto. Incluye el diseño de la solución al problema propuesto, el despliegue realizado para dicha solución, así como la configuración de cada tecnología que interviene en la misma.

- **Bloque 4: Pruebas Realizadas.**
  En este bloque se describen y analizan las pruebas realizadas con dos clientes IMS, el Monster y los scripts de SIPp, de forma que se valida que el entorno desarrollado cumple con los objetivos.

- **Bloque 5: Conclusiones y futuras líneas de trabajo.**
  En este bloque se analiza la consecución de los objetivos y se describen las posibles líneas de trabajo futuro en relación con el proyecto.

- **Bloque 6: Planificación, presupuesto y entorno socio-económico.**
  En este apartado se recoge la planificación y el presupuesto de la realización del proyecto, además del impacto social y económico esperado de la aplicación del resultado del proyecto.

- **Bloque 7: Anexos.**
  - Anexo A: Configuración detallada de los distintos elementos del entorno.
    Configuración detalla empleada en la realización del proyecto.

  - Anexo B: Scripts de SIPp.
    Ficheros SIPp utilizados en las pruebas del proyecto.

  - Anexo C: Normativa y marco regulador.
    Descripción del marco regulador de las telecomunicaciones en España y análisis de la legislación aplicable de las tecnologías empleadas en el desarrollo del proyecto.

  - Anexo D: Extended Abstract.
    Resumen extendido del proyecto, en inglés.

2. ESTADO DEL ARTE Y ALTERNATIVAS DE DISEÑO

Se comienza analizando las tecnologías IMS y CORE utilizadas en el desarrollo del proyecto y se describen alternativas tecnológicas a las soluciones escogidas en el diseño del mismo.

2.1. IMS y SIP

2.1.1. Introducción

IMS [4] [5] es una arquitectura que permite integrar servicios multimedia sobre tecnología IP. Proporciona funcionalidades como la señalización para los servicios multimedia, la autenticación de usuarios y el control de acceso a servicios. Además, al estar basada en conmutación de paquetes permite, de forma opcional, aplicar QoS (Quality of Service).

Está basada en el uso de estándares definidos por el IETF (Internet Engineering Task Force) [6] como SIP y DIAMETER. SIP es un protocolo que permite la señalización y administración de sesiones, multimedia.

IMS fue diseñado para dar soporte a los servicios de multimedia IP a un número muy elevado de usuarios, por esta razón, la escalabilidad es una de sus características más importantes. Los servidores son distribuidos, de tal modo que la capacidad sea extensible, además, el protocolo SIP es fácil de depurar al tratarse de mensajes de texto. La arquitectura de IMS y el uso de interfaces abiertos, hace que la implementación de un nuevo servicio sea más fácil que en otro tipo de arquitecturas, lo que hace que estas aplicaciones sean escalables y flexibles.

2.1.2. IMS: Arquitectura y elementos principales

La forma más eficiente de comprender IMS es analizarlo de acuerdo a su modelo de capas. IMS es una arquitectura de “Capas” formado por:

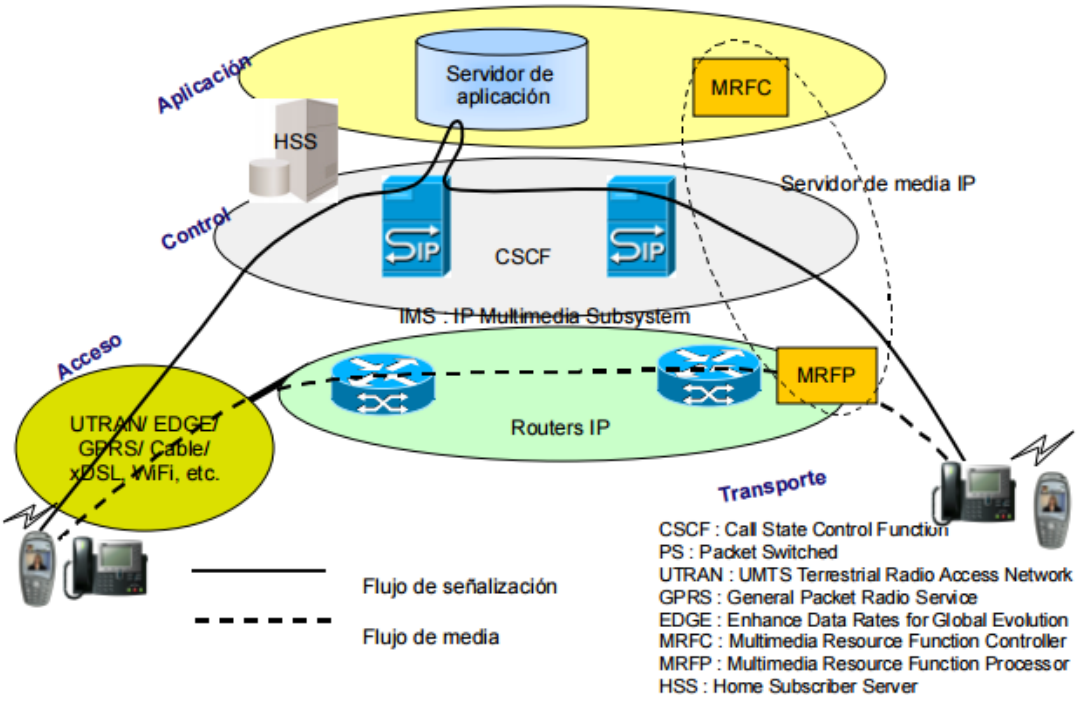


Ilustración 1. Arquitectura de IMS (tomada de [4])

- **La capa de *acceso*.**

Soporta cualquier tipo de acceso de alta velocidad, ya sea acceso móvil, de banda ancha Fija, redes de cable, WIFI, etc. IMS soporta la conversión de protocolos de forma que permite la intercomunicación con otras redes, en particular las basadas en conmutación de circuitos.

- **La capa de *transporte*.**

La capa de transporte está compuesta de los routers y hace referencia a todo el encaminado IP de los paquetes que circulan por la infraestructura. En esta capa se pueden integrar mecanismos de calidad de servicio (QoS). El terminal negocia sus capacidades y expresa sus exigencias durante la fase de establecimiento de sesión, mediante la señalización de control (SIP y SDP) y paralelamente se reservan los recursos necesarios en la red de acceso.

- **La capa de *control.***

La capa de control es en la que está centrada IMS. Consiste en controladores de sesión responsables del encaminamiento de la señalización entre usuarios y de la invocación de los servicios. A estos nodos se les denomina CSCF (Call State Control Function). Se explican en detalle más adelante.

- **La capa de Aplicación.**

Esta capa introduce los servidores de aplicación AS (Application Server) y MRF (Multimedia Resource Function), más conocidos como Servidores de Media IP o IP IMS (IP Media Server). Los servidores de aplicación proporcionan la lógica de los servicios implementados en IMS.

A continuación se estudian en profundidad los elementos principales de la arquitectura IMS que se pueden observar el la Ilustración 2.

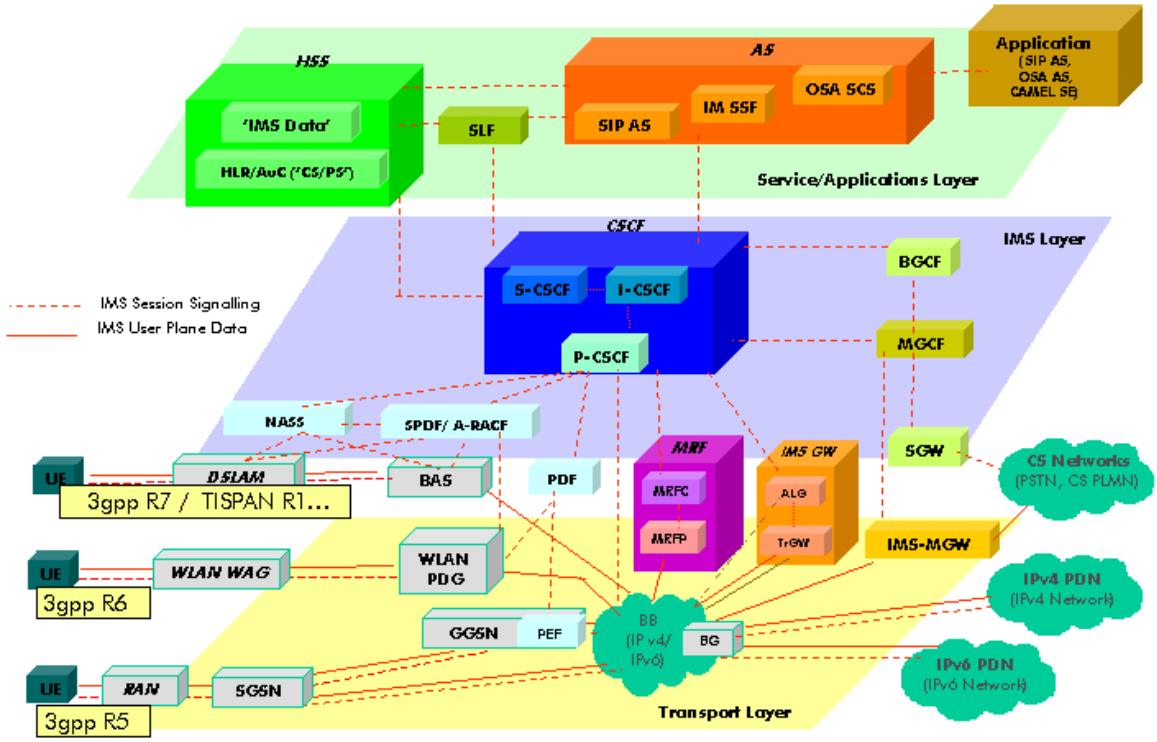


Ilustración 2. Elementos que componen IMS (tomada de [4])

I. CSCF o Función de Control de Sesión de Llamada.

Es el elemento principal dentro de la red IMS, la pieza clave para la señalización a través del protocolo SIP para establecer, modificar y terminar una sesión multimedia. Se puede pensar como un "servidor SIP". Este dispositivo desempeña tres funciones:

- *P-CSCF (Proxy - Call Session Control Function).* Función de control de sesión de llamada Proxy. Es el primer punto de contacto en el dominio IMS y actúa como entrada y salida de la misma.
  Sus funciones abarcan el enrutamiento de las peticiones SIP REGISTER, el encaminamiento de los métodos SIP emitidos por el terminal S-CSCF, el envío de las respuestas SIP al terminal, la generación de CDR (Call Details Record) y la compresión y descompresión de mensajes SIP.

- *I-CSCF (Interrogating –Call Session Control Function).* Función de control de sesión de llamada. Es el punto de contacto dentro de una red de operador para todas las sesiones destinadas a un usuario de ese operador.
  Su misión es asignar a cada usuario su correspondiente S-CSCF, la generación de CDRs y mantener la comunicación con el HSS (Home Subscriber Server) a través del protocolo DIAMETER, para acciones relacionadas con la autorización, autentificación y tarificación.

- *S-CSCF (Serving–Call Session Control Function).* Función de control de sesión de servidor. Asume el control de la sesión, manteniendo el estado de sesión con el fin de poder invocar servicios.
  Sus funciones son, principalmente, el control y mantenimiento de las sesiones de cada UE (User Equipment), la comunicación con el HSS para consultas del perfil de usuario y la consulta hacia el DNS (Domain Name System) para la resolución de direccionamiento y nombres.

CSCF ofrece también otras funciones, como *E-CSCF (Emergency CSCF*), función de servicio de emergencia que permite encaminar peticiones SIP realizadas como llamadas o servicios de emergencia.

II. HSS (Home Subscriber Server).

Es donde se encuentra la base de datos y perfiles de todos los usuarios IMS, es decir, es responsable de la autentificación y registro de usuarios, así como de los diferentes tipos de sesiones que estos pueden establecer, independientemente de la tecnología de acceso. Almacena la información de registro, las identidades del usuario, los parámetros de acceso y la información que permite la invocación de los servicios de usuario.

III. MGW (Media Gateway), MGC (Media Gateway Controller) y SGW (Signalling Gateway Function)

Son nodos para permitir la comunicación desde la red basada en IMS y la red telefónica basada en conmutación de circuitos: MGW en el plano de datos; y MGC junto con SGW en el plano de control.

IV. AS (Application Servers)

Los servidores de aplicación proporcionan la lógica de los servicios implementados en IMS. Generalmente dentro de la red existen múltiples servidores de aplicación, ya que cada uno suele implementar un solo servicio. Existen tres tipos de AS:

- *SIP-AS (SIP – Application Server).* Se comunica directamente con el S-CSCF asignado al usuario de una sesión para mantener el control SIP de la misma. Además, de manera

opcional, es capaz de comunicarse con el nodo destino, si son necesarios los datos que este almacena.

- *OSA-SCS (Open Service Access – Service Capability Server).* Este AS permite obtener una interfaz de comunicación hacia el entorno de aplicación OSA desde la red IMS. Se conoce como servidor de mediación, ya que los servidores OSA permiten acceder a los servicios de otra tecnología, sin implementarla en sí misma. Aunque en IMS la idea es implementar servicios en servidores SIP-AS, se usan servidores OSA-SCS para poder aprovechar en IMS servicios presentes en la plataforma OSA, normalmente medidas de seguridad.

- *IM*-SSF (IP Multimedia – Service Switching Function). Se trata de un servidor de mediación, que puede actuar como servidor de aplicación SIP que a la vez es capaz de comunicarse mediante el protocolo CAMEL (Customized Application for mobile-Network for Enhanced Logic) para utilizar los servicios de las redes GSM (Global System for Mobile communications), sistema estándar de telefonía móvil digital.

V. IP Media Server.

Un Servidor de Media IP o MRF interpreta la señalización SIP recibida del S-CSCF y controla el MRFP. Soporta servicios como conferencias multimedia o anuncios a usuarios. El MRFP (Multimedia Resource Function Controller) mezcla flujos de diferentes conexiones y procesa los flujos multimedia que recibe.

2.1.3. Open IMS Core

Open IMS Core [7] es una implementación de los principales componentes de la arquitectura IMS: servidores CSCF, el HSS y algunos programas complementarios (por ejemplo, para administración y gestión de la plataforma). Es un proyecto desarrollado por FOKUS (Fraunhofer institute Für Offene Kommunications Sisteme) de código abierto bajo licencia GPL2, que permite el desarrollo de servicios IMS. Open IMS Core proporciona un entorno de pruebas de la red IMS, facilitando el testeo y proporcionando una interfaz gráfica desde la que se lleva la gestión de usuarios y la configuración de la red.

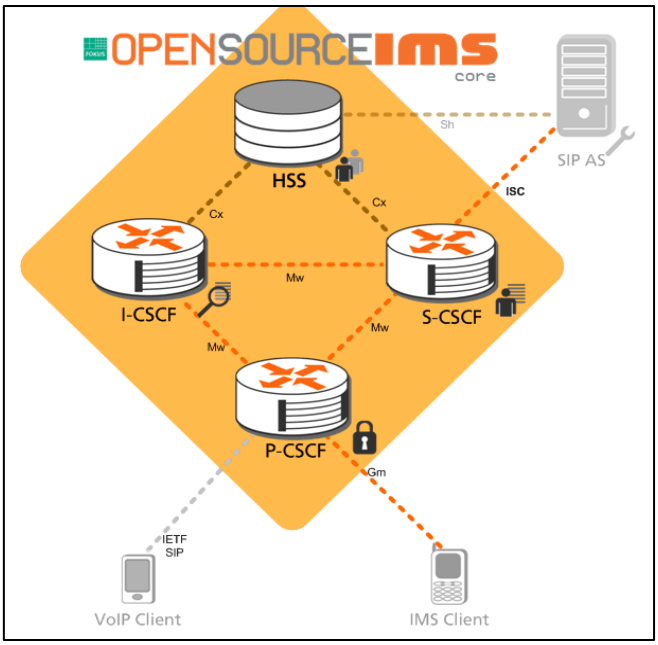


Ilustración 3. Componentes de la arquitectura Open IMS Core. (Tomada de [7])

En la página del proyecto se puede descargar una imagen de una máquina virtual lista para usar que incluye un cliente de IMS: Fokus. Este cliente permite llevar toda la gestión los usuarios que intervienen en el intercambio de multimedia desde la máquina virtual donde se aloja IMS, aunque otra opción válida hubiese sido instalar los servidores IMS en los nodos de la red emulada. Los detalles de esta implementación se describirán más adelante, en el apartado 3.

2.1.4. SIP

El Protocolo de Iniciación de Sesiones (SIP) [8] [9], es actualmente el protocolo de señalización más popular para el establecimiento de servicios multimedia en redes IP. Es utilizado en las aplicaciones más comunes de VoIP (Voice over IP), pero lo más interesante es su implantación en las redes 4G, ya que estas ya no tienen conmutación de circuitos, por lo que los servicios de telefonía en una red 4G, sin combinación con otras redes como 3G, se ofrecen mediante VoLTE (Voice over LTE) usando IMS y SIP. SIP trabaja junto con el protocolo RTP (Real-Time Transport Protocol) por el cual se transportan las comunicaciones de audio y vídeo.

SIP fue desarrollado por el IETF con la intención de ser un estándar para la iniciación, modificación y finalización de sesiones interactivas de usuario. Fue aceptado como el protocolo de señalización de 3GPP en el año 2000, convirtiéndose en un elemento permanente en la arquitectura IMS, de forma que la conmutación de paquetes fue sustituyendo poco a poco a la conmutación de circuitos, que utiliza el sistema de señalización SS7 (Signalling System Nº. 7).

Es un protocolo punto a punto, y como tal, requiere un núcleo de red sencillo y altamente escalable, con inteligencia distribuida en los extremos de la red, incluido en los

terminales. La característica más importante de SIP es que es un protocolo basado en texto, lo que posibilita una fácil implementación y depuración, haciéndolo flexible y extensible.

SDP (Session Description Protocol) es un protocolo integrado en SIP, que define un formato para describir los parámetros de iniciación de un flujo de tráfico multimedia, por ejemplo, permite describir el medio (vídeo, audio, etc.), el protocolo de transporte y el formato utilizado en la codificación. Mediante este protocolo cada extremo de la sesión describe sus capacidades multimedia y se define el tipo de sesión que se desea mantener. Esta información permite también negociar la QoS para el plano de datos, si es necesario.

En conclusión, las principales características de SIP son las siguientes:

- Inicia, termina, modifica y establece sesiones.
- Configura el uso RTP en el plano de datos.
- Utiliza mensajes de texto.
- Permite reserva de recursos mediante la información de SDP.
- El encaminamiento de sus mensajes es independiente del de los datos.
- Es un protocolo punto a punto.
- Permite el intercambio de información entre usuarios, con SDP, para establecer sus capacidades.

I. Arquitectura de SIP.

La entidad básica de SIP es el UA (User Agent). Un UA es un dispositivo final de usuario que se utiliza para establecer y liberar sesiones multimedia con otros UA. Se definen según sean clientes UAC (User Agent Client) o servidores UAS (User Agent Server), siendo capaces de actuar como uno u otro independientemente. Los UA no se conocen entre ellos, solo conocen la situación de un Proxy SIP. Por lo tanto, cualquier UA que quiera comenzar una sesión tendrá que comunicárselo al Proxy SIP. Además de encaminar los mensajes hacia su destino, el Proxy realiza las funciones de autentificación de usuarios, por lo que todo usuario UA tiene que registrarse en el Proxy para poder realizar una llamada. Estas funciones se dividen entre los CSCFs de IMS.

Otras entidades que pueden participar en el intercambio de mensajes SIP son el servidor *REGISTER (*La función de register en IMS la hace el S-CSCF*.)*, o registrador, que gestiona los registros de los usuarios en la red y mantiene la localización de los mismos, de tal forma que para que esos usuarios reciban mensajes pasarán por su correspondiente registrador, que los reencamina correctamente hacia el usuario y el servidor *REDIRECT* o redirector, que se encarga redirigir peticiones de clientes a otros servidores. También puede intervenir un B2BUA (UAS + UAC). El B2BUA es un servidor SIP que en lugar de reenviar señalización SIP recibe señalización SIP (función servidor UAS) y genera, a su vez, señalización SIP hacia el otro lado (función señalización UAC), es decir, genera dos diálogos SIP. En IMS los Application Servers pueden ser B2BUA

La identificación de los usuarios, servicios y nodos se realiza mediante URIs (Universal Resource Identifier). De esta forma los UA no tienen que manejar números de teléfono, sino

nombres con un formato similar al de un correo electrónico <sip:usuario@dominio; parámetros>, por ejemplo: sip: alice@open-ims.test.

Dentro de la red IMS cada usuario tiene asociadas dos identidades con las que se identificará, una identidad pública y otra privada. La identidad pública permite encaminar las peticiones SIP que se dirigen a este usuario y es registrada en la red IMS durante la fase de registro. Es la que usan otros usuarios para indicar con quién quieren comunicarse. La identidad privada es conocida por la red de forma que esta guarda una asociación entre la Identidad privada y la pública. La identidad privada es el identificador único con el que la red conoce al usuario, de forma que todas las funciones de registro, autentificación y acceso a la información de usuario se hacen a través de ella. Un usuario solo puede tener una identidad privada, pero varias públicas. Siguiendo el ejemplo:

- La identidad publica de usuario sería: sip: alice@open-ims.test
- La identidad privada de usuario sería: alice@open-ims.test

SIP tiene una arquitectura cliente-servidor con un esquema de comunicación similar a HTTP (Hypertext Transfer Protocol), donde el cliente es el encargado de iniciar la comunicación y el servidor es el encargado de responder a la petición.

II. Mensajes SIP

Existen dos tipos de mensajes SIP: las respuestas y las peticiones.

Los mensajes de respuesta SIP están formados por un código numérico más una componente textual. En la Tabla 1 se puede ver una lista de los tipos de respuestas en SIP.

TIPO DE MENSAJE	DESCRIPCIÓN	COMENTARIO
1XX	Respuesta informativa	La solicitud se ha recibido pero está en curso
2XX	Mensaje afirmativo	Mensaje recibido, entendido y aceptado
3XX	Redirección	Se indica que se redirija la petición original a otro servidor
4XX	Error en la petición del usuario	La solicitud no es soportada por el UA destino o servidor
5XX	Error en el servidor	El servidor no es capaz de resolver una petición que parece correcta.
6XX	Error global	La solicitud no puede ser procesada por ningún servidor.

Tabla 1. Tipos de respuestas SIP

Las peticiones o métodos SIP son también conocidos como mensajes de solicitud (Requests). Existen catorce tipos, algunos de los cuales son los siguientes:

- *Método INVITE*. Es el método que nos permite establecer una sesión entre dos UAs. Durante esta petición, la red encamina el mensaje desde el UA emisor hasta el receptor. La red utiliza la información de registro de los usuarios para saber la dirección de los mismos.
  SI el proceso se ha realizado adecuadamente, el UA final responde al INVITE con un mensaje 100 TRYING (aunque este mensaje también puede ser generado por un proxy SIP intermedio, para indicar que se está procesando la solicitud) y más tarde con un 180 RINGING.

- *Método BYE*. El mensaje BYE permite liberar una sesión en curso. Este método puede generarlo tanto el UA que generó la petición de sesión como el UA destino.
  Se contesta con el mensaje 200 OK si se termina la sesión de forma correcta.

- *Método REGISTER*. Con este método se indica la petición de registro en la red. En la cabecera del mensaje se envían los datos para la autentificación del usuario.
  Si el registro se ha realizado correctamente se contesta con un mensaje 200 OK.

- *Método CANCEL*. Se utiliza para terminar una llamada que esté en curso, pero no una que ya esté establecida, ya que en este caso se finalizaría con el método BYE.
  Se contesta con un 200 OK si la petición se ha procesado correctamente.

- *Método ACK.* Mediante este método es posible indicar que el proceso petición-respuesta se ha resuelto de manera satisfactoria. Se genera para confirmar que la respuesta a la petición se ha procesado correctamente.

- *Método SUBSCRIBE*. Utilizado por un cliente SIP para suscribirse a notificaciones (avisos de eventos).

- *Método INFO*. Se utiliza para proveer señalización adicional, no necesariamente relacionada con una llamada en proceso.

- *Método NOTIFY*. Es utilizado por el servidor para enviar a un usuario notificaciones de eventos que se han producido (por ejemplo, que un usuario se ha conectado). El usuario debe estar suscrito previamente a las notificaciones.

- *Método OPTIONS*. Es utilizado por el cliente SIP para consultar a otro cliente SIP o proxy sobre sus capacidades.

- *Método REFER*. Utilizado por un cliente SIP para solicitar que el destinario acceda a un recurso proporcionado en la solicitud.

En la Ilustración 4 se muestra un ejemplo de intercambio de mensajes SIP entre dos usuarios (Alice y Boris), a través de un B2BUA, que es un tipo de UA que recibe solicitudes y las procesa como si fuera un proxy. Alice inicia la sesión enviando un INVITE, que es contestado por Boris con los mensajes 100 Trying y un 180 Ringing. Si todo sucede correctamente Boris envía un 200 OK y el terminal de Alice contesta con un método ACK, momento en el cual se establece el intercambio de contenido multimedia a través de RTP. Alice decide terminar la sesión con un método BYE que es contestado por el terminal de Boris con un 200 OK.

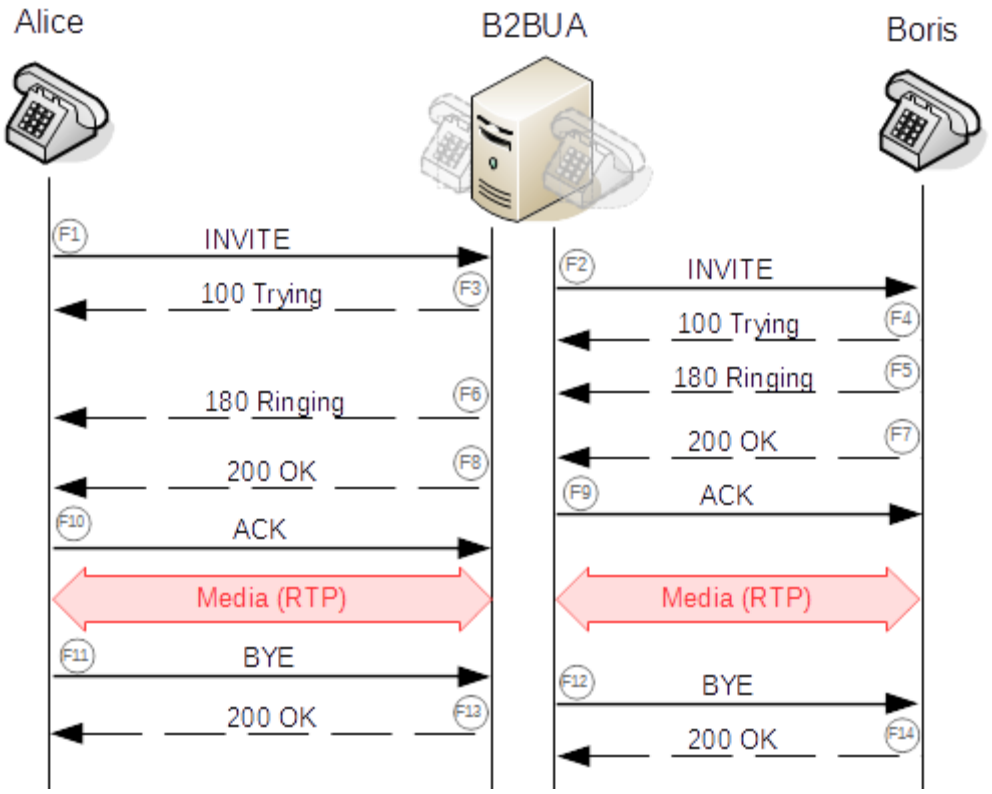


Ilustración 4. Ejemplo de sesión SIP (tomada de [10])

El **formato del mensaje SIP** [11] se compone de tres bloques, línea de inicio, cabecera y cuerpo del mensaje.

- La línea de inicio se conoce según el mensaje, como línea de petición (*request-line),* compuesta por el nombre del método, la URI del destinatario, y la versión del protocolo que se está empleando (hasta ahora siempre se señala SIP/2.0) o como línea de estado (*status-line),* para los mensajes de respuesta. El status-line comienza por la versión del protocolo y la respuesta SIP (indicador numérico y texto). Ejemplos de estas líneas serían:

  INVITE sip: alice@open-ims.test *SIP/2.0*, como línea de petición.

  SIP/2.0 200 OK, como línea de estado.

- La cabecera presenta el siguiente formato: <*header_name*>:<*header_value*>. Dónde el *name* indica el nombre de la cabecera y *value* el valor de la misma. Es posible que existan

varias cabeceras con el mismo nombre pero diferente valor. Dentro de la cabecera se indican los campos de:

- VIA: indica los servidores proxy por los que pasa una petición. En él se incluyen los campos *transport*, que indica el protocolo de transporte que se utiliza; *address*, que indica la dirección IP a la que se envía el mensaje; *port*, que hace referencia al puerto empleado para el envío; y *branch,* que identifica la transacción.

  via:SIP/2.0/<transport><address>:<port>;branch=<ID>

- To: es el campo en el que se indica la dirección SIP URI del receptor. Se utiliza en la fase de registro y establecimiento de sesión. Introduce los campos opcionales *name*, para indicar un nombre, y *tag*, para identificar los mensajes.

  To:"Name" <URI>;tag=<tag_value>

- From: indica la SIP URI del emisor, manteniendo el mismo formato que en el campo *To*.

- Call-ID: contiene una identificación única de la sesión. Incluye los campos *identifier,* un número hexadecimal identificador del mensaje, y *domain,* la dirección IP del servidor del dominio que genera el mensaje.

  Call-ID: <identifier>@<domain>

- Cseq: es el campo que permite conocer la petición a la que va vinculada una respuesta SIP. Contiene *seq_number* (número de secuencia) y *method* (*el nombre del método).*

  Cseq: <seq_number><method>

- Max-Forwards: indica el número máximo de saltos permitido para el mensaje SIP. Decrementa su valor en cada salto.

- Contact: Almacena las URIs en las cuales es posible localizar al UA. Presenta un *name*, la *URI* y un *optional_parameter* (parámetro opcional) con información adicional, generalmente relacionado con la validez temporal de la petición.

  Contact: "Name" <URI>=<optional_parameter>

- Record-Route: almacena la dirección del proxy SIP que inserta la cabecera mientras se hace el enrutamiento de la petición. De esta forma queda registrado por donde tienen que pasar las siguientes peticiones de la sesión.

- Content-type: solo se genera cuando los mensajes SIP contienen descriptores de sesión en el cuerpo del mensaje. Este campo indica el tipo de descriptor del cuerpo del mensaje. En IMS generalmente se utiliza SDP, indicándose como "application/sdp".

- Content-Length: al igual que *content-type*, solo se genera cuando los mensajes SIP contienen descriptores de sesión en el cuerpo del mensaje. Este campo indica el número de octetos del cuerpo del mensaje.

- El cuerpo del mensaje contiene la información que se intercambian los dos extremos en la comunicación. Gracias a la información de la línea de inicio y de la cabecera es posible la comunicación extremo a extremo. En IMS se usa SDP con un formato MIME, el cual permite que el cuerpo del mensaje conste de varias partes cada una con un formato distinto, lo cual vendrá especificado en la cabecera.

A continuación se puede observar un ejemplo de mensaje de SIP, donde se pueden observar los campos estudiados. En particular se trata de un mensaje de petición, en el que Alice invita a Bob a iniciar una conversación, y un mensaje de estado, en el que Bob acepa la solicitud de Alice.

```
INVITE sip: alice@open-ims.test SIP/2.0
Via: SIP/2.0/UDP host.open-ims.test
To: sip: bob@open-ims.test
From: sip: alice@open-ims.test
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Content-Type: application/sdp
Content-Length: 885

SIP/2.0 OK
Via: SIP/2.0/UDP host.open-ims.test
To: sip: alice@open-ims.test
From: sip: bob@open-ims.test
Call-ID: a84b4c76e66710
Content-Type: application/sdp
Content-Length: 780
```

Ilustración 5. Ejemplos de mensajes SIP.

III. Mensajes SDP

Como ya se ha mencionado, SDP es un protocolo destinado a la descripción de los parámetros que son necesarios para la notificación, el inicio y establecimiento de una sesión multimedia. La descripción de la sesión consiste en una serie de líneas de texto con la forma *<type>=<value>*, donde el campo *type* es un único carácter y *value* puede estar formados por dos parámetros. En la Tabla 2 se pueden observar los distintos campos *type*.

TIPO	DESCRIPCIÓN	OBLIGATORIO
V	Versión del protocolo	Sí
O	Identificador	Sí
S	Nombre de sesión	Sí
I	Información de la sesión	Sí
U	URI de la descripción	No
E	Dirección de correo	No
P	Número de teléfono	No
C	Información de conexión	No
B	Ancho de banda	No
T	Tiempo de sesión	Sí
R	Tiempo de repetición	No
Z	Tiempo de corrección	No
K	Clave de encriptación	No
A	Atributos	No
M	Descripción de media	Sí

Tabla 2. Campos de los mensajes SDP

A la hora de construir los mensajes, estos campos deben aparecer en el orden en el que se muestran en la Tabla 2. A continuación se analizan los campos más representativos:

- El primer campo, *"v="* muestra el número de versión que se usa del protocolo, en la actualidad siempre toma el valor de 0.

- El campo identificador *"o=",* indica quien origina la sesión. Está formado por varios campos a su vez, como el nombre de usuario en el host origen *<username>*, la sesión *<sess-id>*, el número de versión para la descripción de sesión en particular *<sess-version>* que irá decrementando cada vez que se realiza alguna modificación en los datos de la sesión, el tipo de red *<nettype>,* el tipo de dirección IP utilizada (IP4 o IP6) *<addrtype>,* y la dirección global de la máquina que ha iniciado la sesión *<unicast-address>.*

  o=<username> <sess-id> <sess-version> <nettype> <addrtype> <unicast-address>

- El campo *"s="* consiste en una cadena de caracteres que indican el nombre de sesión. Solo puede existir un campo de este tipo por cada sesión.

- El campo "c=" contiene los datos de conexión de la sesión, donde hay que enviar el tráfico una vez establecida la sesión. Es imprescindible que si existen varias descripciones cada una empezará un tipo de campo *"c="*. Sigue el formato:

  c=IN IP4 <unicast_IP-address>

- En ancho de banda se representa con el carácter, *"b="*. Es un campo opcional que indica el ancho de banda que puede usar la sesión. Contiene dos parámetros: *<bwtype>* que habitualmente usa los valores RR (para indicar el ancho de banda de algunos usuarios de la sesión) o RS (que indica el ancho de banda para los datos de los emisores activos de la sesión), y *<bandwidth>* que contiene el valor de bits por segundo.

  b=<bwtype>:<bandwidth>

- Los campos de tiempo, *"t="*, indican el comienzo y el final de la sesión. Pueden existir varios campos de tiempo, donde cada *"t="* adicional indica unidades de tiempo adicionales que la sesión permanece activa. Contiene dos campos, uno de *<start-time>*, que indica el inicio de la sesión y *<stop-time>* que indica el tiempo en el que acaba la sesión.

- El campo *"m="* indica los descriptores de media de la sesión, es decir, describe el tráfico en el plano de datos que se va a intercambiar en la sesión. Pueden existir varios en una misma sesión, si el tráfico tiene varios componentes, por ejemplo audio y vídeo. Contiene varios campos: <media> indica el tipo de datos, *<port>* indica el puerto de transporte donde los datos serán enviados, *<proto>* indica el protocolo empleado para el transporte, y *<fmt>* describe el formato del contenido (su codificación). Es decir:

  m=<media> <port> <proto> <fmt>

En la Ilustración 6 se puede observar un ejemplo de mensaje SIP que incluye todos los campos analizados anteriormente. En él, la usuaria Alice intenta establecer una conversación que incluye voz y vídeo.

```
v=0
o=Alice 3239874567 4447990887 IN IP4 10.0.0.1
s=Hablemos de aves
c=IN IP4 10.0.0.1
t=0 0
m=audio 20000 RTP/AVP 0
a=sendrecv
m=video 20002 RTP/AVP 31
a=sendrecv
```

Ilustración 6. Ejemplo de mensaje SDP

2.2. CORE

2.2.1. Introducción

CORE (Common Operator Open Research) es una herramienta para la emulación de redes en una o más máquinas virtuales. Consiste en una GUI (Graphical User Interface) de fácil manejo, donde se pueden generar diferentes topologías y configurar los equipos activos que forman la red. Las topologías también pueden ser implementadas usando lenguaje Python.

Utiliza técnicas de virtualización de sistemas operativos existentes para crear redes virtuales, donde las implementaciones de protocolos y aplicaciones usadas en sistemas reales se pueden ejecutar sin ser modificados. Estas redes emuladas pueden ser conectadas a redes y sistemas reales, lo que hace al emulador adecuado para realizar pruebas de plataformas, estudios de seguridad, evaluación de escenarios e investigación de protocolos.

CORE se puede descargar desde su página web oficial [3]. Es de código abierto, bajo licencia BSD (Berkeley Software Distribution). Además se ofrece la posibilidad de descargar la imagen de una máquina virtual con formato *vmdk* que contiene ya instalado el emulador. También se ofrece un manual [12] con consejos y otros datos útiles para el usuario.

Las ventajas que ofrece CORE y que han llevado a su elección frente a otros emuladores son las siguientes:

- Es de código abierto y se distribuye de forma gratuita.
- Fácil de usar (GUI intuitiva).
- Permite ejecutar código real sobre los equipos de la red.
- Ejecuta protocolos y aplicaciones sin modificarlos.
- Permite la conexión de redes reales en tiempo real.
- Eficiente y escalable.
- Es altamente personalizable.
- Fácil de instalar.

2.2.2. Arquitectura de CORE

CORE tiene una arquitectura modular, que permite la mezcla y combinación con diferentes componentes [13]. Por ejemplo, las redes emuladas de CORE se pueden comunicar con otros simuladores y emuladores, como ENAME (orientado a redes inalámbricas) y GNS3 (emulador gráfico de red que permite diseñar topologías, similar en objetivos a CORE). Se utiliza una API personalizada basada en sockets para comunicarse entre los servicios GUI y CORE, para así permitir la posibilidad de ejecutar diferentes componentes en diferentes máquinas físicas.

El emulador se puede ejecutar en plataformas FreeBSD y Linux, siendo esta última la más utilizada. En Linux, CORE utiliza nombres de red virtualizados para construir nodos virtuales y los integra en las redes utilizando un puente (bridge) Ethernet en Linux. Cada nodo de la topología creada ejecuta un sistema Linux virtualizado independiente de los demás, con acceso

a todas los programas del Linux base. Por lo tanto, en cada nodo, se puede ejecutar cualquier programa Linux como se haría en un equipo real con sistema operativo Linux.

Un demonio CORE gestiona las sesiones de emulación. Se controla a través de la GUI y utiliza módulos de Python, que pueden ser exportados directamente de scripts de Python.

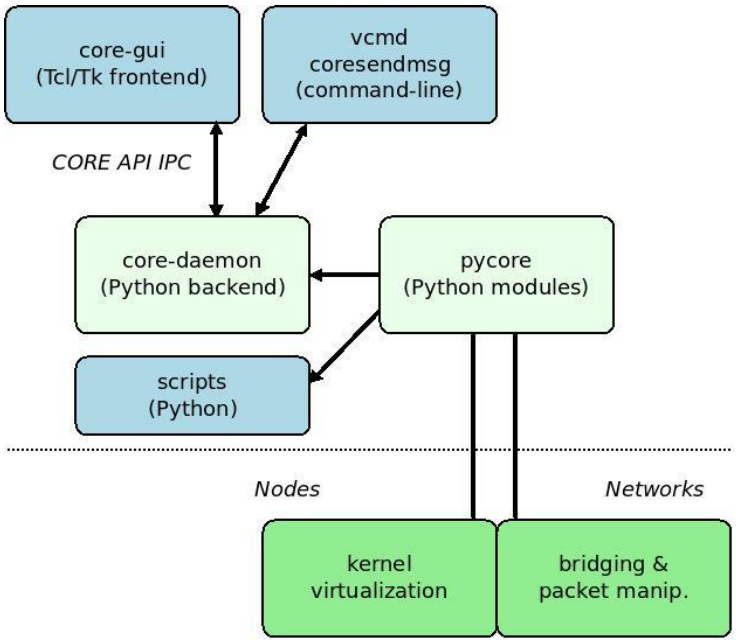


Ilustración 7. Arquitectura de CORE (tomada de [12])

2.2.3. Interfaz gráfica de usuario

La interfaz gráfica de usuario CORE es un programa Tcl / Tk (Tool Command Language / Tool Kit) [14] que presenta las siguientes características:

- Es modular, implementa la mayoría de los lenguajes.
- Es portable. Al ser un lenguaje de alto nivel se puede separar totalmente de la arquitectura del equipo.
- Es multiplataforma y permite su integración con otros lenguajes.
- Debido a que la API funciona a través de un socket TCP, la interfaz gráfica no necesita ejecutarse en la misma máquina que la emulación.

La interfaz gráfica de CORE tiene dos modos de trabajar:

- *Modo Edición*: En este modo se podrá crear la topología usando la barra de herramientas de la izquierda de la interfaz. La GUI proporciona un lienzo de dibujo, donde se colocan los nodos (router, PCs, hubs, etc…), que se podrán configurar a disposición del usuario.

- *Modo Ejecución*: Tras finalizar la edición, clicando el botón verde, se crea una instancia de la topología, construyéndose así la emulación. En este modo, el usuario

puede interactuar con los componentes de la topología. CORE ofrece una consola por cada nodo y Widgets personalizables para mostrar información gráfica de cada nodo, por ejemplo, se puede mostrar el encaminamiento OSPF (Open Shortest Path First) entre nodos usando líneas de colores variados.

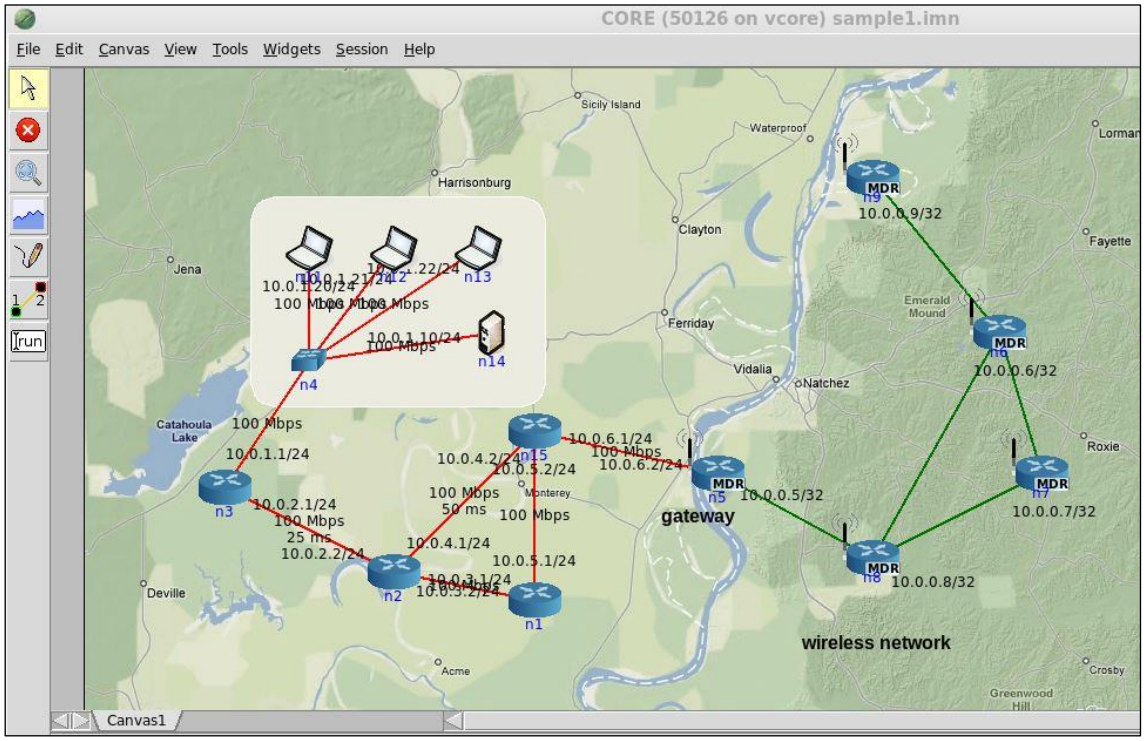


Ilustración 8. Captura interfaz gráfica CORE en el modo Ejecución.

2.3. Máquinas virtuales

2.3.1. Introducción

Una máquina virtual es un software capaz de emular en su interior otro sistema operativo, como si se tratase de un ordenador real [15]. En la actualidad son empleadas para probar otros sistemas operativos (distintos del que se tiene en la máquina física que ejecuta la máquina virtual), para usar aplicaciones disponibles en otros sistemas operativos, para ejecutar programas antiguos (que ya no pueden ejecutarse en la versión de sistema operativo de la máquina física), para probar una aplicación en distintos sistemas operativos, etc. Las máquinas virtuales se pueden clasificar en dos grandes categorías según su funcionalidad:

- *Máquinas virtuales de sistema*. Una máquina virtual de sistema es aquella que emula un ordenador completo, es decir, es un software que puede hacerse pasar por un ordenador real, de tal modo que se puede ejecutar otro sistema operativo en su interior. Tiene su propio disco duro, memoria, tarjeta gráfica y demás componentes de hardware, aunque todos ellos son virtuales. Son las máquinas que se van a utilizar en la realización de este proyecto.

- *Máquinas virtuales de proceso*. Una máquina virtual de proceso es menos ambiciosa que una de sistema. No emular un PC por completo, sino que ejecuta un proceso concreto, como una aplicación, en su entorno de ejecución.

Para usar una máquina virtual, lo primero que se necesita es instalar una aplicación en el ordenador anfitrión o host, capaz de crear o reproducir máquinas virtuales.

Como se ha mencionado anteriormente, tanto los desarrolladores de Open IMS Core como los del emulador de redes CORE ofrecen la descarga de una imagen de una máquina virtual con el software correspondiente ya instalado. Por lo tanto, en la realización de este proyecto se dispondrá de dos máquinas virtuales alojadas en la máquina anfitriona, cada una de las cuales estará dentro de una aplicación de máquinas virtuales diferente: la máquina virtual del CORE estará alojada en VirtualBox y la máquina de Open IMS Core se alojará en VMWare. Para más información sobre la configuración de estas máquinas virtuales, ir al apartado 3.1.Configuración del entorno.

2.3.2. VirtualBox

Oracle VM VirtualBox es un software de virtualización para arquitecturas x86 que fue desarrollado por la empresa alemana INNotek GmbH, pero que pasó a ser propiedad de Oracle Corporation. Por medio de esta aplicación es posible instalar sistemas operativos adicionales, conocidos como “sistemas invitados” dentro del sistema operativo anfitrión. Entre los sistemas operativos soportados de la máquina anfitrión, se encuentra GNU/Linux, Mac, Windows y Solaris. Dentro de estos sistemas es posible virtualizar los sistemas operativos, FreeBSD, GNU/Linux, OpenBSD, Windows, OS/2 Warp, Solaris, MS-DOS y muchos más.

Inicialmente era ofrecida bajo una licencia de software que no era gratuito, pero hace unos años surgió VirtualBox OSE (Open Source Edition) bajo la licencia GPL2. Actualmente existe una versión Oracle VM VirtualBox, que es gratuita únicamente bajo uso personal o de evaluación, y está sujeta a la licencia PUEL (VirtualBox Personal Use and Evaluation License) y otra versión Open Source que es software libre sujeto a licencia GPL. Todas las versiones son descargables desde su página oficial [16].

Las **características** que ofrece VirtualBox son:

- Dispone de documentación técnica para el usuario muy completa.
- Fácil de manejar.
- Es multiplataforma.
- Ofrece la ejecución de máquinas virtuales de forma remota, por medio de RDP (Remote Desktop Protocol).
- Se pueden compartir archivos entre máquinas.
- Permite crear instancias para restaurar una máquina fácilmente.
- Soporte limitado para gráficos 3D.
- Permite utilizar aplicaciones virtualizadas como si se tratara de aplicaciones del sistema.
- Es compatible con máquinas virtuales de VMWare (al menos en teoría, en la práctica se ha comprobado que esto no siempre es fácil).
- Cuenta con herramienta de captura de vídeo.
- Soporta USB 2.0 y 3.0 instalando una extensión.
- Cifrado de unidades virtuales instalando una extensión.

2.3.3. VMWare

VMWare es una filial de EMC Corporation, que proporciona software de virtualización disponible para arquitecturas x86. VMWare tiene software de pago como VMWare Workstation, y otros gratuitos destinados a uso personal, como VMWare Server y VMWare Player, todos ellos descargables desde su página oficial [17].

VMWare funciona en sistemas operativos anfitrión Windows, Linux y Mac y puede crear multitud de sistemas operativos virtuales, al igual que en VirtualBox. Las versiones gratuitas de esta aplicación incluyen únicamente las funcionalidades básicas, mientras que las de pago incluyen todas las características, como la fácil creación de máquinas virtuales, la optimización de hardware, impresión sin drivers, la posibilidad de clonar máquinas y crear varias imágenes del sistema operativo.

La versión que interesa para el desarrollo de este proyecto, VMWare Workstation Player, tiene las siguientes **características**:

- Cuenta con numerosas herramientas y funciones para entornos empresariales.
- Permite compartir archivos fácilmente entre el host y el sistema virtualizado.
- Es compatible con lectores de tarjetas inteligentes.

- Soporta USB 3.0
- Permite crear instancias para restaurar el estado de una máquina virtual.
- Posibilidad de compartir máquinas virtuales.
- No requiere configuración adicional para la integración de la impresora o de una red.
- Gráficos 3D con OpenGL o DIrectX.

2.4. Alternativas de diseño

En este apartado se describen alternativas tecnológicas a las soluciones escogidas en este proyecto.

2.4.1. Alternativas a IMS

IMS nos proporciona una infraestructura para tener señalización para proporcionar servicios multimedia sobre redes IP. En este proyecto se opta por usar una implementación open source de la arquitectura IMS (Open IMS Core). Existen dos alternativas para obtener el mismo tipo de funcionalidad que típicamente se usan en redes IP, y que se podría haber adoptado en el proyecto: soluciones propietarias de señalización de sesiones, y el uso de implementaciones de servidores SIP (sin la arquitectura IMS).

Existen multitud de arquitecturas propietarias que sirven para implementar servicios audivisuales a través de internet. Algunas de ellas son:

- **Skype** [18] es un software propietario de Microsoft, que permite compartir texto, voz y vídeo sobre Internet entre sus usuarios. Es una aplicación de libre descarga accesible a través de varios dispositivos, en la que los usuarios no tienen que pagar por comunicarse entre sí. Skype también permite llamar a teléfonos convencionales, con un coste adicional. Utiliza un protocolo privado de telefonía VoIP y se distingue del resto de sofwares de este tipo porque, al menos en sus orígenes, utilizaba tecnologia P2P (Peer-To-Peer) en vez de cliente-servidor (aunque los detalles de funcionamiento no son públicos).

Ilustración 9. Ejemplo de dispositivos que soportan Skype (tomada de [18]).

- **Viber** [19] es una aplicación que permite la comunicación con cualquier punto del mundo desde un ordenador o un móvil sin ningún coste. A diferencia de Skype, en viber no se pueden realizar vídeollamadas, pero sí se puede compartir textos, audio,

fotos y realizar llamadas de voz. No se sabe qué sistema de señalización utiliza, ya que esta información no es pública.

Las demás alternativas, aunque sí usan SIP, no tienen el resto de componentes de la arquitectura IMS:

- **Solución de telefonía VoIP de Cisco** [20] [21] es un sistema de comunicaciones IP que permite el intercambio de servicios audivisuales a través de la red. Esta tecnología implementa SIP en la señalización de sus llamadas, permitiendo que las comunicaciones sean más seguras y eficaces. Incluye infrestructura de red, seguridad, movilidad, productos de administración de red, opciones flexibles de administración e implementación, etc. En la actualidad es empleada fudamentalmente por empresas (por ejemplo, en la Universidad Carlos III de Madrid) y requiere de la instalación de terminales para su correcto funcionamiento y pasarelas a la red telefónica conmutada.

- **Kamailio** [22] es un servidor SIP de código abierto. Permite aplicaciones de VoIP como son el tráfico de voz, vídeo y multimedia en tiempo real.

En resumen, aunque se han visto opciones que también hubieran podido ser interesantes para construir, en el proyecto, la infraestructura de señalización para servicios multimedia, se ha decidido utilizar IMS por ser esta una solución más completa, abierta (frente a sistemas de señalización propietarios), y que nos permite disponer de la misma infraestructura que los operadores utilizan en sus redes para ofrecer servicios audivisuales. Además, con Open IMS Core disponemos de una implementación de IMS de código abierto y uso gratuito, lo que nos da la máxima flexibilidad para alcanzar los objetivos del proyecto y posibles desarrollos futuros.

2.4.2. Alternativas a CORE

Una alternativa a CORE sería usar un simulador en lugar de un emulador. Pero un simulador no proporcionaría la misma funcionalidad: un simulador solo trata de reproducir el comportamiento del programa o protocolo a evaluar en la red, un emulador modela de forma precisa el hardware emulado y permite probar software real. Es decir, simular permite probar la lógica o el comportamiento de nuestra solución de red, emular permite probar la implementación de nuestra solución de red.

A continuación se muestra una lista de algunas alternativas y las razones por las que no han sido elegidas:

- **GNS3** (Graphical Network Simulator) [23] es un emulador gráfico de red de código abierto que permite diseñar topologías de red y realizar emulaciones sobre ellas. Es

una herramienta que permite la combinación de dispositivos virtuales y reales, utilizados para emular redes complejas. Sin embargo no ha sido elegido para la realización del proyecto ya que, a pesar de ser en algunas funcionalidades, en particular en equipos soportados, más potente que CORE, sus funcionalidades avanzadas, como la emulación de equipos CISCO, no interesaban en el desarrollo del proyecto, ya que los sistemas operativos necesarios para instalar en los equipos emulados son de pago. No habiendo una funcionalidad que hiciese GNS3 mejor para los objetivos del proyecto, se prefirió CORE a GNS3 porque CORE es usado en actividades docentes en el departamento donde se realiza este proyecto, por lo que la elección de CORE hace más aplicables en la práctica los resultados del mismo.

- **CiscoPacket Tracer** [24] es uno de los simuladores de redes más completos. Fue desarrollado por Cisco para realizar pruebas en sus routers, hubs y servidores. Es de fácil manejo y de cogido abierto, pero es un simulador (no un emulador) y, como se ha explicado antes, no sirve en la realización del proyecto (en el que se quiere evaluar implementaciones software desarrolladas para equipos reales y que puedan ser trasladadas a equipos reales).

- **Netsim** [25] es un simulador utilizado principalmente en investigaciones y en laboratorios de pruebas. También está desarrollado por Cisco, aunque soporta funciones más avanzadas que el simulador mencionado previamente. No se ha elegido ya que es un simulador, y además es de pago.

2.4.3. Alternativas de máquinas virtuales

Existen multitud de programas para crear máquinas virtuales. En la realización de este proyecto se han elegido VMWare y VirtualBox por varias razones, mencionadas en el apartado 2.3, algunas de las más destacables son:

- Son fáciles de manejar.
- Tienen una versión gratuita.
- Disponen de documentación técnica para el usuario muy completa.
- Permiten compartir archivos fácilmente entre el host y el sistema virtualizado.
- Permiten importar la imagen de una máquina ya existente.

Aun así, existen otros programas con funcionalidades similares. A continuación se muestra una lista de algunos de ellos junto a las razones por las que no se han seleccionado en el desarrollo de este proyecto:

- **Parallels** [26]: Es un software muy potente, que permite algunas de las características de los programas elegidos, como compartir archivos con el host y su facilidad de implementación. Su principal problema es que no tiene licencia gratuita.

- **QEMU** [27]: Es un software libre diseñado para sistemas operativos Linux que ejecuta el sistema directamente en el hardware del host. Es una herramienta muy potente pero no sirve para la realización del proyecto, ya que es para sistemas operativos Linux y el ordenador host utilizado implementa Windows.

- **Windows Virtual PC** [28]: es un programa de software libre que solo permite emular versiones anteriores de Windows. Por este motivo, ya que se necesita emular otros sistemas operativos, en particular Linux, no ha sido elegido para la realización de este proyecto.

2.5. Conclusiones del capítulo.

En este capítulo se ha ofrecido una visión de las principales tecnologías empleadas en el proyecto. En concreto se ha analizado la arquitectura IMS y el protocolo de señalización SIP, la implementación de IMS: Open IMS Core, el emulador de redes CORE, y los dos software de virtualización empleados, VirtualBox y VMWare.

Finalmente se han descrito algunas alternativas de diseño que se podrían emplear para dotar de señalización en servicios audiovisuales a un emulador de redes.

En los siguientes capítulos se describe el uso de estas tecnologías en el proyecto.

3. INTEGRACIÓN DE IMS EN EL EMULADOR CORE

Una vez analizadas las tecnologías que permiten llevar a cabo la implementación del sistema, se va a hacer un estudio de todos los pasos necesarios para integrar las tecnologías y desarrollar una plataforma para el despliegue de servicios audiovisuales basados en IMS en redes emuladas en CORE.

3.1. Configuración del entorno

Se comienza dando una visión global del entorno desarrollado. El objetivo final es crear una red emulada en CORE cuyos nodos se puedan comunicar con los elementos de la arquitectura IMS para acceder a servicios audiovisuales. Por lo tanto, hay que conectar los servidores de la arquitectura IMS a la red emulada dentro del CORE. Para ello se usa un ordenador físico anfitrión con sistema operativo Windows, y se despliegan en él dos máquinas virtuales (ambas con Linux): una donde se ejecutan los servidores de la arquitectura IMS, y otra donde se ejecuta el CORE y donde están, por lo tanto, las redes emuladas.

Para configurar fácilmente todas las máquinas (la física y las virtuales) del escenario se ha utilizado un router a modo de servidor DHCP. En la Ilustración 10 se puede observar el escenario completo incluyendo las direcciones IP.

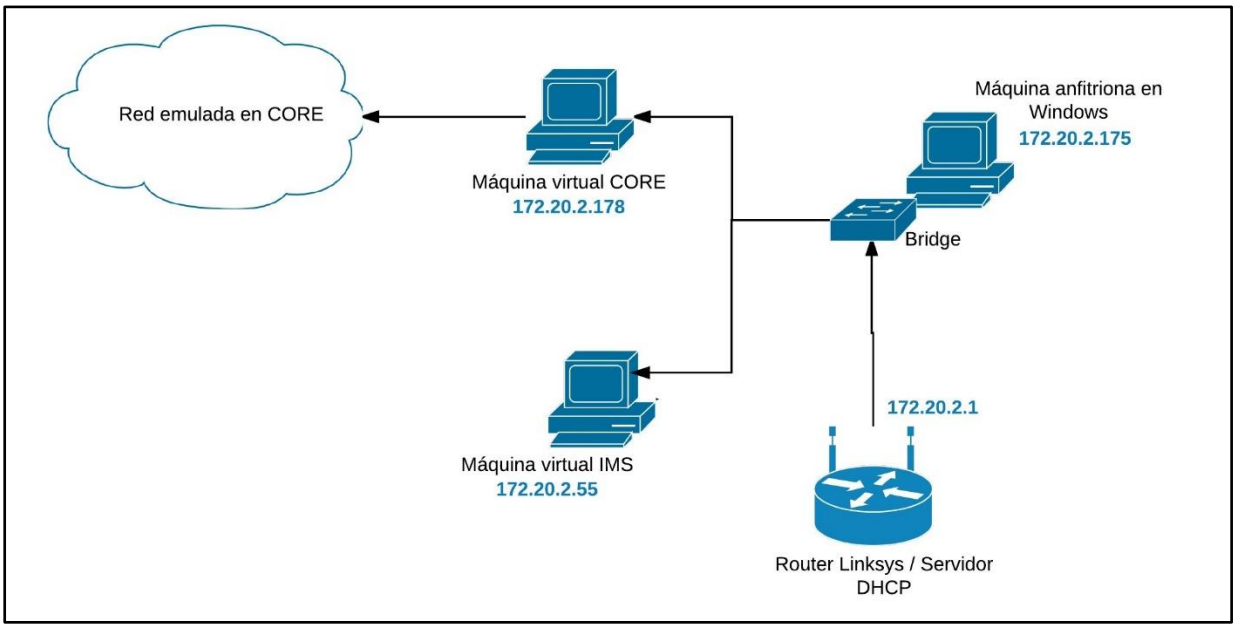


Ilustración 10. Escenario completo con direcciones IP.

3.1.1. Router

Para la realización del proyecto se ha utilizado un router inalámbrico que funciona como servidor DHCP (Dynamic Host Configuration Protocol) asignando una dirección IP a cada

dispositivo conectado a la red. Este paso no es completamente necesario pero facilita mucho la configuración de la máquina IMS que debe una dirección IP fija, tal y como se explica en el siguiente apartado 3.1.2 Open IMS Core (Una alternativa sería usar direccionamiento estático, pero usar DHCP es más cómodo porque así el equipo anfitrión esta siempre configurado para optener el direccionamiento por DHCP, tanto si se conecta a Internet como si se conecta a la red del escenario de pruebas). Además, aunque no se ha hecho, el router permitiría conectar otras redes al escenario, que también se comunicarían con el escenario emulado del CORE. Por ejemplo, se podría tener un servicio audiovisual entre una máquina dentro del CORE y una máquina física conectada al router (o en cualquier lugar de Internet siempre que el direccionamiento fuese el adecuado, el router se conectase a Internet y las rutas se configuraran adecuadamente).

Se ha usado un router inalámbrico de banda ancha Wireless – G [29]. Este router permite conectarse a la red de forma inalámbrica, compartiendo el acceso a Internet, archivos y datos de forma fácil y segura. Está configurado con tres subredes, siendo la primera dirección de cada subred la interfaz del router:

- La red WIFI, con el nombre core-testbed y unas direcciones asignadas 172.20.2.0/24. Es la subred que se utiliza en el desarrollo del proyecto.
- Una red cableada, que cubre las interfaces Ethernet 1-4 con las direcciones 172.20.0.0/24.
- Otra red cableada que cubre la interfaz Ethernet, con las direcciones 172.20.1.0/24.

En la realización del proyecto solo se ha utilizado la primera subred, la red WIFI, pero se podrían conectar otros nodos al resto de subredes. En este caso, para que las comunicaciones con el escenario fuesen posibles, sería necesario configurar una ruta en el router para indicar cómo llegar a las direcciones de la red emulada en CORE.

El protocolo DHCP, que asigna una dirección IP a cada dispositivo que se conecta a la red, está activado para todas las subredes, asignando una dirección IP según la subred correspondiente dentro del rango establecido. DHCP normalmente da la misma dirección IP a los equipos pero esto puede cambiar tras un largo periodo de tiempo, por lo que se va a forzar una entrada host en el router para asignar una IP fija a la MAC de cada interfaz. Se realiza este paso en los tres dispositivos unidos a la red (el ordenador físico y las dos máquinas virtuales). Además es necesario indicarle al router que anuncie por DHCP, en cada subred mencionada anteriormente, un servidor de DNS, que será el que está en la máquina virtual IMS. Esto es así porque se va a usar la máquina virtual IMS para configurar un servidor de DNS que se encargue de los nombres DNS que se usan en el escenario.

Los comandos concretos para la configuración del router están detallados en el primer apartado del ANEXO A: Configuración detallada de los distintos elementos del entorno.

3.1.2. Open IMS Core

La máquina virtual IMS aloja los distintos CSCF's, que son los que ya se ha mencionado en el apartado 2. A continuación se muestra una lista de los mismos, en la que también se indican los puertos por los que estos escuchan:

- P-CSCF o servidor de control de sesión de llamada Proxy. Se identifica con el nombre DNS pcscf.open-ims.test (manteniendo el nombre del dominio DNS predeterminado) y escucha peticiones en el puerto UDP 4060.

- I-CSCF cuyo nombre es icscf.open-ims.test y escucha peticiones en el puerto UDP 5060.

- S-CSCF o control sesión de servidor, cuyo nombre es scscf.open-ims.test y escucha peticiones en el puerto UDP 6060.

La máquina virtual IMS también aloja el servidor de DNS que se usa en todas las máquinas del escenario además de un sistema de gestión con una interfaz web, a través de la cual se lleva la gestión de usuarios.

I. Configuración de la máquina virtual.

En la página oficial del proyecto Open IMS [7] se puede acceder a la descarga del mismo de forma libre, bajo Licencia Pública General de GNU y en varios formatos. Existe la opción de descargar la imagen de una máquina virtual con el código instalado, tanto para Gentoo como para Ubuntu. Otra opción es descargarse el código de un servidor subversión (SVN), y proceder a su instalación siguiendo el manual accesible desde esa misma página.

Para la realización de este proyecto se optó por la primera opción, ya que era más fácil de implementar y cumplía los requisitos necesarios. Esta imagen está disponible en formato vmdk, el cual teóricamente es soportado por VirtualBox si se hace la conversión correcta. En un principio se optó por esta opción pero existían problemas de compatibilidad de gráficos, por lo tanto, y para eliminar errores de instalación derivados de la conversión del formato vmdk a otro soportado por VirtualBox, se decidió instalar la máquina en VMWare.

Tras descargar VMWare e importar la imagen de la máquina virtual, se procede a la configuración de la misma. En primer lugar se configura el modo de conexión a la red [30]. Existen cinco opciones:

- **Sin conexión de red:** No se configura ninguna conexión de red para la máquina virtual.

- **Modo Bridged o puente:** Este modo de conexión permite a la máquina virtual acceder a una red Ethernet externa. La máquina tiene que tener su propia

dirección IP en la red externa y otros equipos pueden acceder a ella a través de dicha IP. La dirección IP se asignará manualmente o mediante un servidor DHCP.

- **Modo NAT:** Este modo permite una conexión NAT (Network Address Translation), es decir, la máquina virtual y el propio sistema host comparten una sola IP que no es visible fuera de la propia red. La máquina virtual tendrá acceso a Internet y se podrá comunicar con el host usando una dirección privada y el NAT.

- **Modo red interna:** Este modo de red permite conectar varias máquinas virtuales entre ellas, creando una red privada. Las máquinas virtuales no pueden conectarse con el equipo anfitrión ni viceversa.

- **Modo Host-Only:** Host Only proporciona una conexión de red entre la máquina virtual y el sistema host, usando un adaptador de red virtual visible solo para el sistema operativo del host. La máquina solo podrá comunicarse con el host, ya que su red está dentro del propio host y es invisible e inaccesible para cualquier otro equipo.

El objetivo de configuración de la máquina virtual donde se aloja IMS es que esta no atienda solo a usuarios locales (usando la dirección de loopback), sino que se necesita acceder a ella desde otras máquinas, en concreto, desde la máquina virtual donde se ejecuta el CORE.

En la realización de este proyecto, puesto que se necesita conectar la máquina virtual, que contiene los servidores IMS, con otra donde se ejecuta el CORE, el modo NAT no sirve, ya que IMS tiene servidores que tienen que poder ser alcanzados iniciándose la comunicación desde el exterior (es decir, las comunicaciones no se inician solo desde la máquina virtual que contiene los servidores IMS, que es el tipo de comunicación que permitiría el NAT). Por lo tanto, el modo de conexión elegido es Bridged y la IP única será asignada por el servidor DHCP.

Los servidores IMS y el servidor de DNS de la máquina virtual IMS vienen configurados por defecto para escuchar en la dirección de loopback, es decir, solo es posible comunicarse con ellos desde la propia máquina virtual. Esta configuración no es válida en este proyecto, en el que se quiere que las peticiones a los servidores IMS vengan del exterior (de nodos en la red emulada). Por lo tanto hay que cambiar la configuración para que los servidores escuchen en la dirección IP asignada en el escenario a la máquina virtual con el Open IMS Core. Para ello hay que cambiar una serie de ficheros de configuración (tanto de los servidores IMS como del servidor de DNS) tal como se detalla en el ANEXO A. Además, hay que añadir registros al servidor de DNS para indicar la dirección IP para los nombres DNS de los servidores. Esta configuración se detalla en el apartado II. Máquina virtual donde se encuentra el Open IMS core del ANEXO A:

II. Creación y gestión de usuarios.

Desde de la máquina virtual donde se aloja IMS se puede acceder a un sistema de gestión con una interfaz web, a través de la cual se lleva la gestión de usuarios.

Los pasos a seguir en la creación de un nuevo usuario se detallan en el manual de instalación de Open IMS [31], en el "Paso 7: Configurar abonados". Para crear un nuevo usuario se accede al sistema de gestión del Open IMS Core y se crea una suscripción, una identidad privada, una identidad pública y se enlazan. A continuación se muestra un ejemplo para dar de alta al usuario Eva:

- En primer lugar se accede al apartado *"USER IDENTITIES"* del gestor y se muestra el siguiente menú desde donde se lleva la gestión de usuarios:

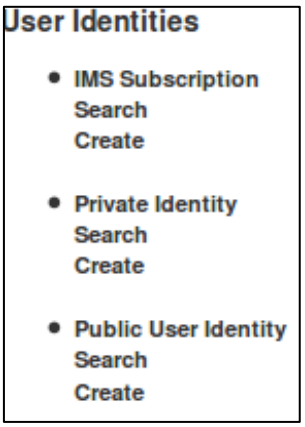


Ilustración 11. Menú del Interfaz web de gestión de usuarios en IMS.

- Se crea una subscripción, introduciendo los campos tal y como aparecen en la Ilustración 12.

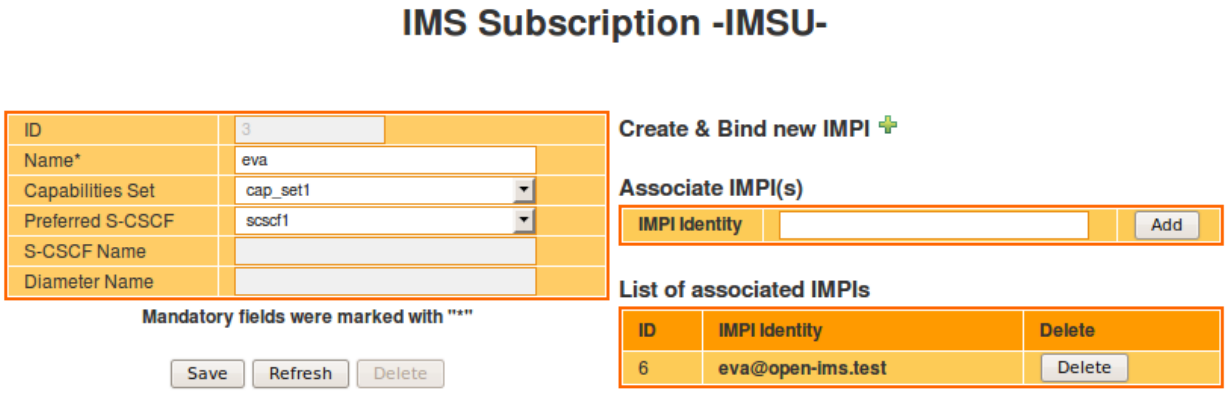


Ilustración 12. Creación de una suscripción en IMS.

- A través del enlace de la pantalla anterior *"Create & Bind new IMPI"* se crea una identidad privada. En este apartado se elige la identidad privada que tiene que tener el formato nombre@opne-ims.test y opcionalmente se puede introducir una clave secreta. Los campos han de quedar tal y como se muestran en la Ilustración 13.

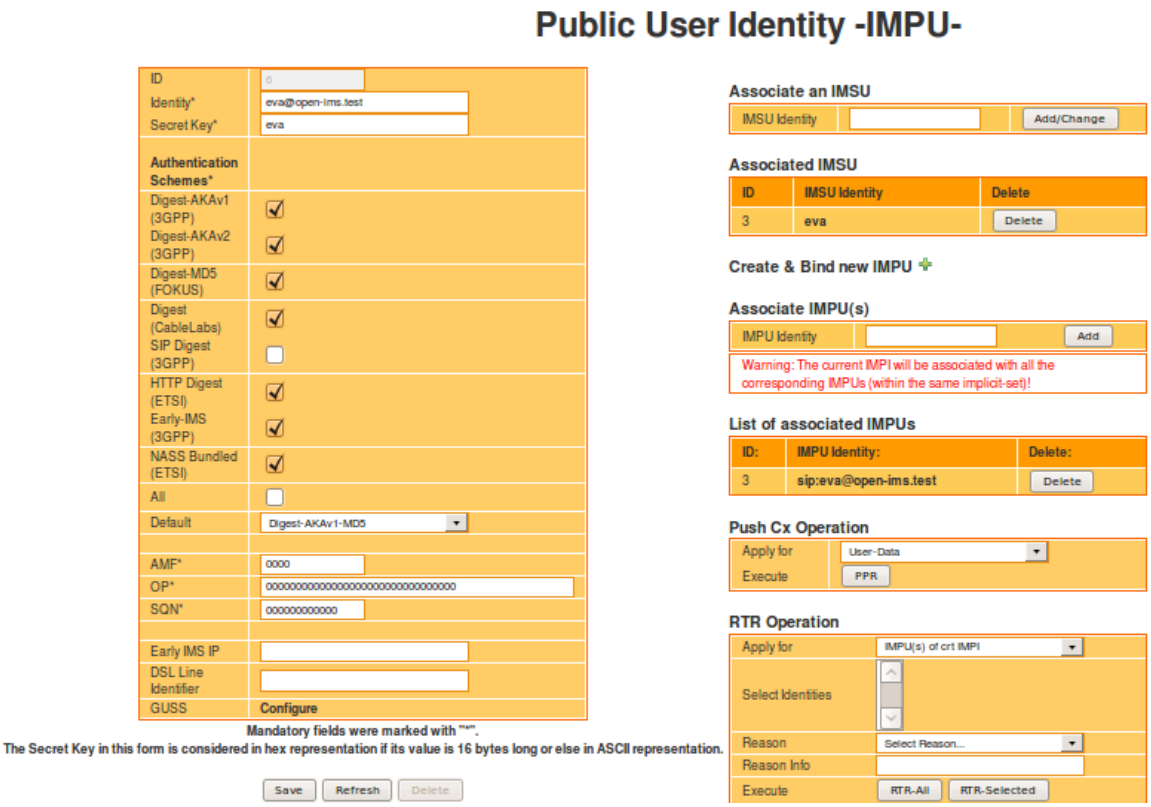


Ilustración 13. Creación de una identidad privada en IMS.

- Por último se da de alta la identidad pública con el formato sip:nombre@opne-ims.test y se enlaza con la identidad privada ya creada a través del submenú *"Associate IMPI(s) to IMPU".*

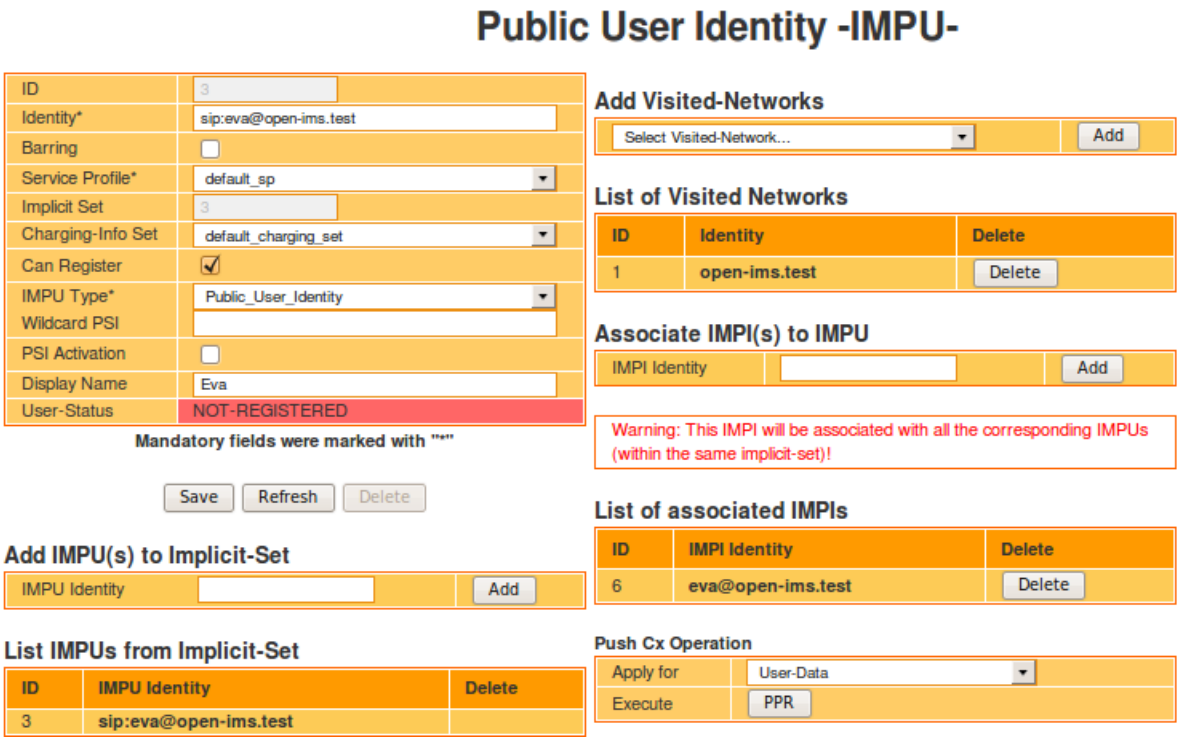


Ilustración 14.Creación de una identidad publica en IMS.

3.1.3. CORE

En este paso se configura un escenario emulado de red que tenga funcionalidad para acceder a servicios audiovisuales. Para alcanzar este propósito es necesario que los nodos emulados puedan comunicarse con el exterior de la máquina CORE para acceder a los servidores IMS. Además será necesario instalar clientes IMS que puedan ser ejecutados desde los nodos emulados. Para tener una mejor visión global del escenario este se detalla en la Ilustración 15, incluyendo la red emulada en CORE.

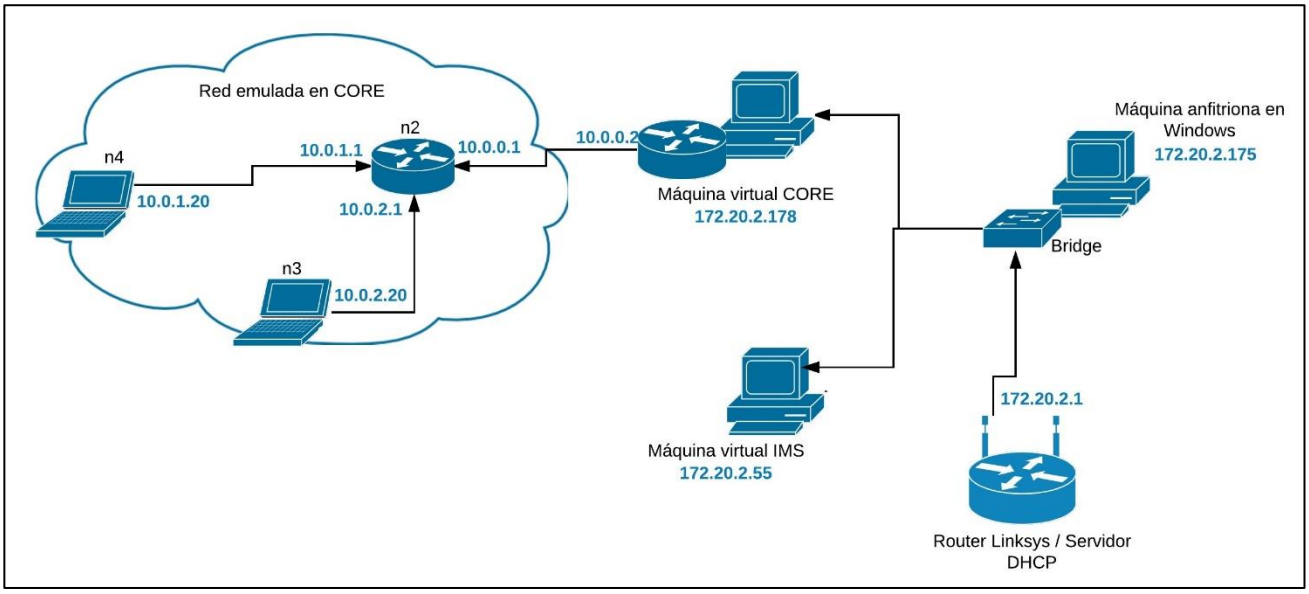


Ilustración 15. Escenario con la red emulada en CORE incluida.

I. Configuración de la máquina virtual

En primer lugar se analiza la configuración de la máquina virtual donde se aloja CORE.

CORE se puede descargar desde su página web oficial [3]. Es de código abierto, bajo licencia BSD, además se ofrece la posibilidad de descargar la imagen de una máquina virtual con formato *vdi* que contiene ya instalado el emulador. Para la realización de este proyecto se ha optado por esta opción, instalando la imagen virtual en VirtualBox.

Tras descargar VirtualBox e importar la imagen de la máquina virtual, se procede a la configuración de la misma. En primer lugar se configura el modo de conexión a la red. Los modos de conexión posibles en VirtualBox son los mismos que para VMWare y al igual que en el caso anterior, el modo de conexión utilizado es Bridged, con el objetivo de poder tanto recibir solicitudes de comunicación desde el exterior como generarlas hacia el exterior de la máquina virtual. La IP única será asignada por el servidor DHCP.

Se ha elegido usar el servidor de DNS alojado en la máquina virtual donde está el IMS como servidor de DNS para todo nuestro escenario, incluyendo la máquina virtual donde se aloja CORE. La configuración para que esto sea posible se detalla en el apartado Máquina Virtual donde se aloja CORE del ANEXO A:

II. Configuración del escenario emulado

La configuración del programa CORE debe permitir la comunicación de los nodos de la red emulada con el equipo donde se aloja el Open IMS Core.

En primer lugar se necesita conectar el host (máquina virtual) donde se aloja CORE a un nodo de la emulación CORE. Para ello, siguiendo el manual de CORE se crea un "RJ45 Tool" asociado a una interfaz "dummy". El nodo RJ45 incorpora al escenario emulado una interfaz de red de la máquina CORE, de modo que la máquina virtual que ejecuta el CORE pasa a tener una interfaz en el escenario emulado. Por lo tanto, activando el reenvío de tráfico en la máquina virtual que ejecuta el CORE, cualquier equipo que se pueda comunicar con esa máquina virtual, también podrá comunicarse con los nodos de la emulación. Los pasos a seguir son:

- Crear una interfaz dummy en la máquina virtual donde se aloja CORE.
- Crear un RJ45 conectado a un router en el escenario a simular y asociarle la interfaz dummy0.
- Ejecutar la emulación y configurar una dirección IP del enlace al que está unido RJ45.
- Activar el forwarding en la máquina virtual donde se aloja CORE

Estos pasos están detallados en el apartado IV. Escenario emulado en CORE del ANEXO A:

Las siguientes configuraciones para que los nodos de la emulación se comuniquen con las máquinas virtuales y con el ordenador host dependen del escenario creado en CORE. El escenario emulado en este proyecto es el mostrado en la Ilustración 16, donde la dummy descrita anteriormente se conecta a un router al que van conectados dos dispositivos, cada uno de los cuales servirá como un usuario IMS.

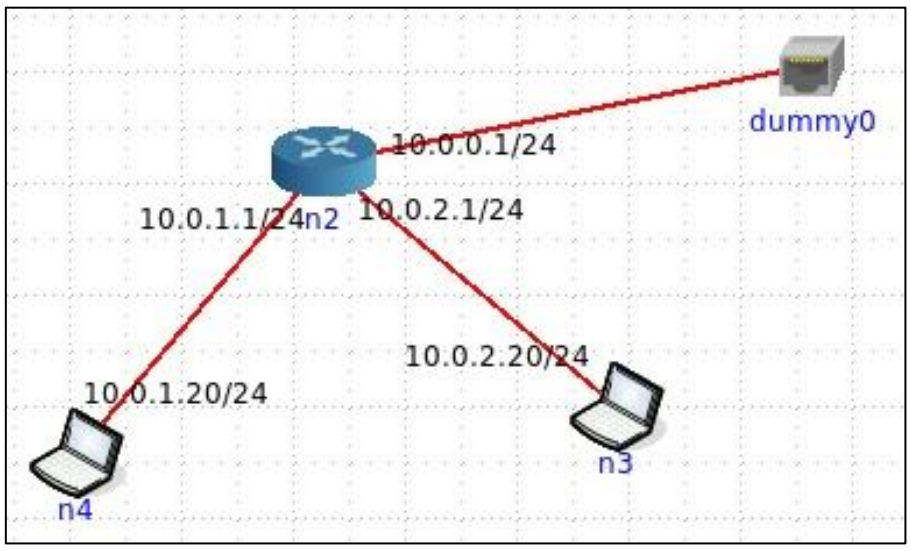


Ilustración 16. Escenario emulado en CORE.

Para que los nodos del escenario emulado se comuniquen con el router y el RJ45 del escenario emulado, es necesario configurar la tabla de rutas, es decir, se indicará que las direcciones del escenario emulado están accesibles vía un siguiente salto a la interfaz del router

al que está conectado el RJ45. También se tienen que añadir entradas a las tablas de rutas de la máquina virtual donde se aloja IMS y a la máquina host, para indicar la ruta que tienen que seguir los datos para llegar a los nodos de la red emulada en CORE. Los pasos a seguir se detallan en el apartado IV.Escenario emulado en CORE del ANEXO A:

De esta forma ya se podrá comunicar cualquier nodo del escenario emulado con todas las máquinas virtuales y la máquina host. Para comprobar que todo funciona correctamente se puede realizar una sencilla prueba que consiste en enviar un *"ping"* desde los nodos del escenario a las máquinas del escenario.

III. Instalación de clientes IMS.

Una vez que el escenario está configurado, con comunicación entre los nodos emulados en CORE y la máquina virtual donde se aloja IMS, es necesario probar que efectivamete se pueden usar los servidores IMS desde los nodos emulados para hacer uso de servicios multimedia. Para ello, se necesitan clientes IMS en los nodos emulados. Se han pribado dos tipos de clientes IMS:

- Cliente IMS con interfaz gráfica de usuario.

Esta opción consiste en descargarnos un Cliente IMS, como Monster. Este cliente dispone de una interfaz gráfica de usuario, desde la cual se puede interactuar con SIP, registrando a usuarios, realizando llamadas entre ellos, etc.

En la máquina virtual IMS descargada, tal y como se ha mencionado anteriormente, ya se dispone de un cliente IMS, en este caso es Foku-s Monster. Foku-s Monster es de código abierto y está disponible en varias versiones para distintos sistemas operativos. Por esta razón es el que se ha elegido para instalar también en la máquina CORE. La instalación [32] es distinta dependiendo del programa descargado. En este caso consistió simplemente en instalar una serie de paquetes y compilar el ejecutable que se ha descargado.

A la hora de ejecutarlo, es necesario que el Monster se abra en el nodo del escenario CORE que actúa como usuario, pero existe el problema de que los nodos emulados no tienen pantalla, solo tienen terminales de texto para interactuar con ellos. Por lo tanto, para ejecutar un programa con interfaz gráfica se tendrá que redireccionar la salida de la interfaz gráfica a la pantalla de la máquina virtual donde se ejecuta el CORE, para lo cual se emplea SSH. SSH es un protocolo que facilita las conexiones entre dos sistemas, de forma que se podrá acceder a cualquier máquina de la red que tenga instalado y activo dicho protocolo. En el caso bajo estudio lo que se ha hecho es, desde la máquina virtual que ejecuta el CORE, usar ssh para ejecutar monster en un nodo emulado del escenario CORE, redireccionando las X (la interfaz gráfica) a la máquina virtual donde se ejecuta el ssh (la máquina virtual con el CORE).

Los pasos para realizar estas tareas se detallan en el apartado V. Instalación de clientes IMS.

Ilustración 17. Interfaz de cliente IMS.

Una vez abierta el Monster en el nodo de la red emulada, es necesario registrar a uno de los usuarios que ya estaban dados de alta en IMS. Se accede al menú "*New*" y se procede a configurar un usuario, indicando, entre otros parámetros: el dominio, la identidad pública, la identidad privada, la clave secreta y el servidor pcscf. La configuración detallada se muestra en la Ilustración 18.

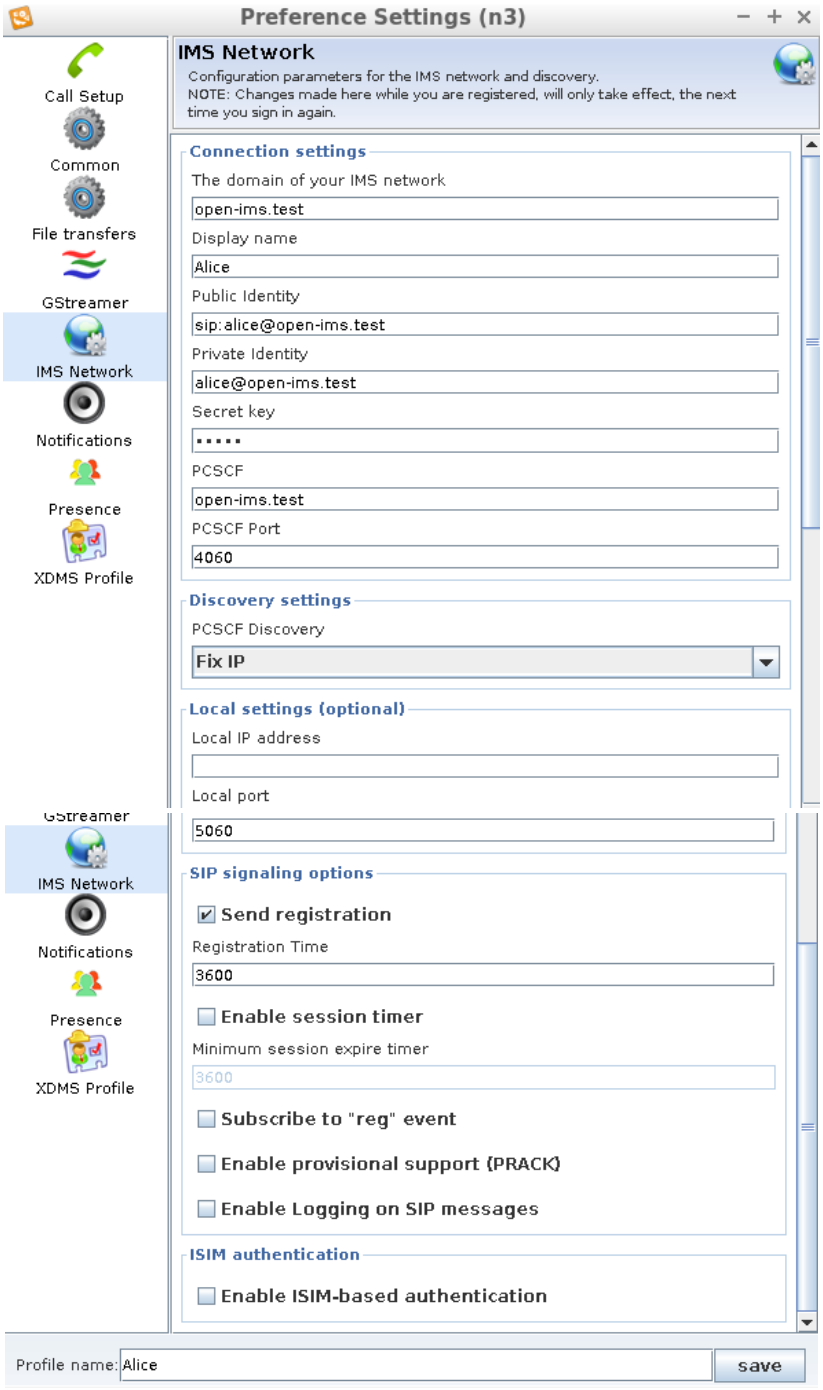


Ilustración 18. Configuración de un usuario en Monster.

También se puede indicar, en el apartado "GStreamer", los parámetros RTP para la llamada, es decir, el puerto RTP y los tipos de formatos de audio y vídeo que soporta, entre otras cosas.

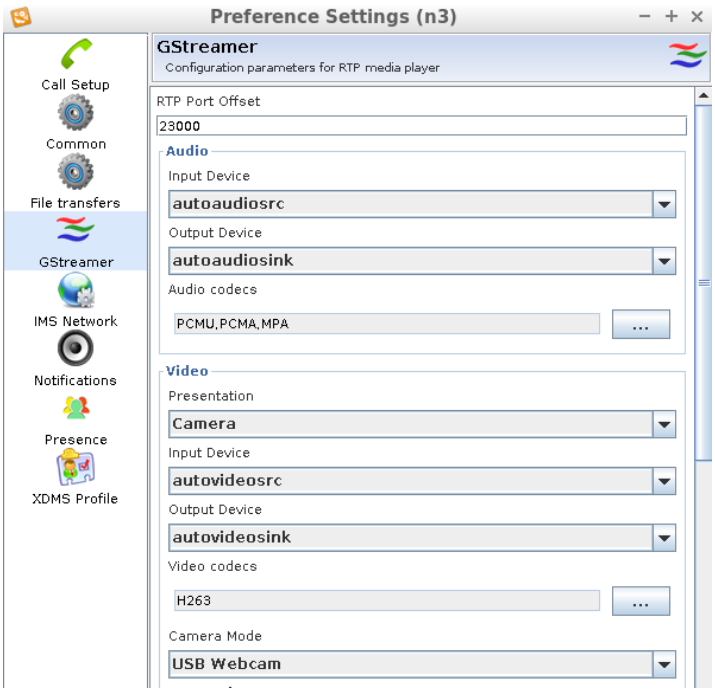


Ilustración 19. Ejemplo configuración GStreamer.

Por último, tras realizar todas las configuraciones correctamente ya se puede registrar el usuario. Una vez realizado el registro se accede a la interfaz desde donde se podrá interactuar con otros usuarios.

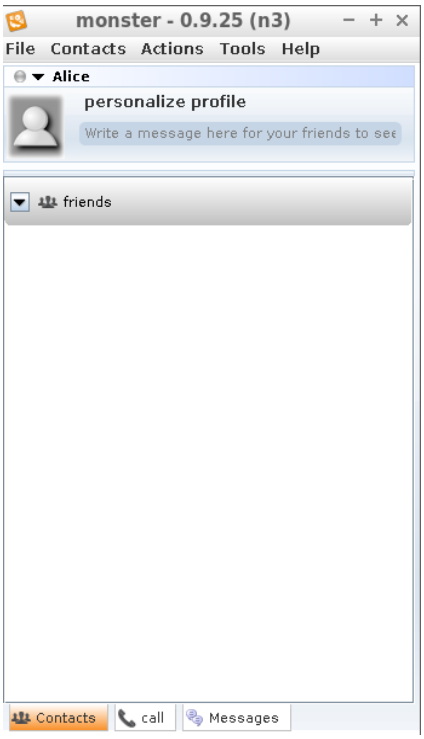


Ilustración 20. Interfaz Monster del usuario Alice.

- A través de scripts SIPp.

En esta opción se va a usar una herramienta, denominada SIPp, que permite de forma muy flexible y potente definir la generación de tráfico SIP. El primer paso es bajarse el código fuente de SIPp, que es una herramienta que permite generar tráfico SIP. Dicha herramienta lee archivos personalizables en los que se puede describir una llamada entre usuarios, o cualquier tipo de tráfico SIP. Además, SIPp realiza un resumen de la llamada al finalizarla donde se puede observar, entre otras cosas, un dibujo con los mensajes intercambiados.

SIPp está disponible como paquete en Ubuntu cuya descarga e instalación es mucho más sencilla, usando el gestor de paquetes apt-get. El problema es que la versión de SIPp en apt-get no está compilada con la opción de soporte de autenticación (openssl), y sin el soporte de autenticación, el cliente SIPp no puede autenticarse con los servidores IMS. Por eso necesitamos compilar e instalar desde el código fuente con la opción openssl.

SIPp es de licencia abierta, descargable desde su página web oficial [33] donde también se ofrece un manual de usuario. Los pasos a seguir para su instalación son muy sencillos y están detallados en el apartado V. Instalación de clientes IMS del ANEXO A.

Tras instalar SIPp, se debe cambiar el emulador de terminal de los nodos de la red emulada en CORE a un terminal abierto “xterm” y copiar los ficheros al directorio donde se ejecuta el SIPp (o poner la ruta absoluta donde está el fichero). Esta configuración está detallada en el apartado V. Instalación de clientes IMS del ANEXO A. En este punto ya se pueden ejecutar, en los nodos CORE que se desee, los scripts que definen el intercambio de tráfico SIP. Hay ejemplos de scripts para distintos tipos de comunicaciones en la página de SIPp. A continuación se muestra uno de ellos (un REGISTER), donde se puede observar en verde la línea de comandos que se utiliza para ejecutarlo. Todos los ficheros utilizados en este proyecto están detallados en el ANEXO B: Scripts de SIPp.

```
<!--
   sipp -sf register.xml open-ims.test -rsa 172.20.2.55:4060 -auth_uri open-ims.test -i 10.0.2.20 -p 5061 -m 1 -inf users.csv
-->
<scenario name="UAC Registering in IMS Core">
 <send retrans="500">
  <![CDATA[
     REGISTER sip:open-ims.test SIP/2.0 Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch] From: <sip:[field0]@open-ims.test>
     To: <sip:[field0]@open-ims.test> Call-ID: [call_id] CSeq: 1 REGISTER Max-Forwards: 20 Contact: <sip:[field0]@[local_ip]:
     [local_port]>;transport=[transport] Authorization: Digest username="[field1]@open-ims.test",realm="open-ims.test",uri="sip:open-
     ims.test",nonce="",response="",algorithm=MD5 Expires: 28800 Content-Length: 0
   ]]>
 </send>
 <recv response="401" auth="true"></recv>
 <send retrans="500">
  <![CDATA[
     REGISTER sip:open-ims.test SIP/2.0 Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch] From: <sip:[field0]@open-ims.test>
     To: <sip:[field0]@open-ims.test> Call-ID: [call_id] CSeq: 2 REGISTER Max-Forwards: 20 Contact: <sip:[field0]@[local_ip]:
     [local_port]>;transport=[transport] [field2] Expires: 28800 Content-Length: 0
   ]]>
 </send>
 <recv response="200"></recv>
 <pause milliseconds="2000"/>
</scenario>
```

Ilustración 21. XML de registro

Algunos de los parámetros que se han utilizado en la ejecución son:

- -rsa: Es la dirección IP donde se aloja IMS y el puerto pcscf.
- -i: Dirección IP del usuario que se va a registrar.
- -p: puerto local para usar en el envío de la señalización SIP. Puede ser cualquiera que esté libre, pero hay que tener en cuenta que el que se use en el registro hay que usarlo para toda la sesión SIP.
- -m: Indica el número de llamadas.
- -inf: Fichero con valores que se usarán en el script (por ejemplo, el nombre de un usuario a registrar en un procedimiento de registro).

3.2. Conclusiones del capítulo.

En este capítulo se ha explicado la forma de integrar la señalización SIP en el emulador CORE. Para ello se ha comenzado por una visión global de la configuración del entorno, donde se nombran todas las tecnologías empleadas y más tarde se ha procedido a explicar la configuración concreta del router y de las máquinas virtuales que alojan IMS y CORE. Además se ha descrito cómo dar de alta a un usuario IMS y dos opciones distintas para ejecutar clientes IMS en los nodos emulados.

En el siguiente capítulo se analizan las pruebas realizadas para comprobar el correcto funcionamiento de toda la infraestructura.

4. PRUEBAS REALIZADAS

A continuación se detallan las pruebas realizadas para comprobar el funcionamiento del sistema. Dichas pruebas se han realizado de dos formas distintas, cada una correspondiente a un cliente IMS. El entorno donde se realizan las pruebas es el descrito en el apartado anterior.

4.1. Pruebas con el cliente Monster

4.1.1. Registro de un usuario

Para comprobar el correcto funcionamiento del entorno se comienza por realizar un registro. Este proceso es una buena prueba de que todo está activo y ejecutándose, ya que utiliza todos los componentes.

Se ejecuta un cliente Monster en uno de los nodos del escenario y se procede al registro de uno de los usuarios que ya están dados de alta, en este caso el usuario Alice. Se lanza el Monster en el nodo n3 del escenario emulado en CORE (Ilustración 15) obteniendo la interfaz gráfica del programa (ver la izquierda de la Ilustración 22). Desde el programa ya se puede indicar que se quiere registrar al usuario Alice (la configuración se ha enseñado en el capítulo anterior). Si el proceso se realiza correctamente, Monster no indicará ningún mensaje de error y la pantalla cambiará para mostrar que el usuario está registrado (ver la derecha de la Ilustración 22).

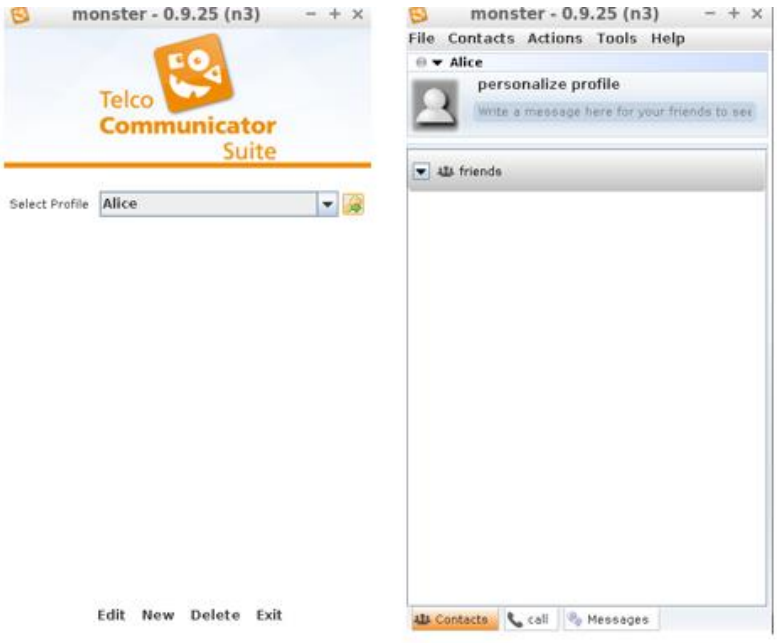


Ilustración 22. Iterfaz de Monster antes y después del registro de Alice.

Para comprobar el funcionamiento se captura, durante el proceso de registro, el tráfico de mensajes SIP que llegan a la IP donde está alojado IMS mediante Wireshark. Se puede observar la captura del mensaje de REGISTER en la Ilustración 23. En ella se pueden apreciar los

campos de la petición SIP: la línea de registro, donde se indica el método (REGISTER), la URI del destinatario, que en el caso de un mensaje REGISTER es el dominio del servidor donde se hace el registro (sip:open-imes.test) y la versión del protocolo (SIP/2.0); el campo "*Call-ID*" que contiene la identificación de la sesión; el campo *"From"* que indica la URI del emisor; el *"To"* que indica la URI del usuario que se está registrado (al tratarse de un REGISTER) y la *"via"* que indica los servidores proxy por los que pasara la petición, entre otros datos.

```
▽ Request-Line: REGISTER sip:open-ims.test SIP/2.0
    Method: REGISTER
    [Resent Packet: False]
▽ Message Header
    Call-ID: 5cf5d9b5aba185237cc6ef401d9e46af@10.0.2.20
  ▷ CSeq: 2 REGISTER
  ▷ From: "Alice" <sip:alice@open-ims.test>;tag=1001
  ▷ To: "Alice" <sip:alice@open-ims.test>
  ▷ Via: SIP/2.0/UDP 10.0.2.20:5060;branch=z9hG4bK79b380ced441cc914f571d97a9b9dffb373339
    Max-Forwards: 20
  ▷ [truncated] Authorization: Digest username="alice@open-ims.test",realm="open-ims.test",nonce="jtc5VH
    Expires: 3600
  ▷ Contact: "Alice" <sip:alice@10.0.2.20:5060>;+sip.instance=2de7ac7b-cdf8-49a5-a0c8-9dc22def8958
    User-Agent: monster Version: 0.9.25
    Content-Length: 0
```

Ilustración 23. Mensaje REGISTER del usuario Alice.

Un proceso de registro incluye el envío de un REGISTER, que se contesta con un 401 Unauthorized, este mensaje SIP incluye un reto que permite al cliente enviar de nuevo un REGISTER pero con información para realizar la autenticación. A este segundo REGISTER le sigue una respuesta OK, tal y cómo se muestra en la Ilustración 24, que confirma que el REGISTER se ha realizado con éxito. Uno de los detalles que se pueden observar es que se tiene el mismo "*Call-ID*" ya que es un mensaje correspondiente a la misma sesión. El resto de campos funcionan de forma similar al mensaje REGISTER.

```
▽ Status-Line: SIP/2.0 200 OK - SAR succesful and registrar saved
    Status-Code: 200
    [Resent Packet: False]
▽ Message Header
    Call-ID: 5cf5d9b5aba185237cc6ef401d9e46af@10.0.2.20
  ▷ CSeq: 2 REGISTER
  ▷ From: "Alice" <sip:alice@open-ims.test>;tag=1001
  ▷ To: "Alice" <sip:alice@open-ims.test>;tag=a4e5f2154cf3650649bd099eeba887d7-026c
  ▷ Via: SIP/2.0/UDP 10.0.2.20:5060;rport=5060;branch=z9hG4bK79b380ced441cc914f571d97a9b9dffb373339
    P-Associated-URI: <sip:alice@open-ims.test>
  ▷ Contact: <sip:alice@10.0.2.20:5060>;expires=3600;pub-gruu="sip:alice@open-ims.test;gr=2de7ac7b-cdf8-49a5-a0c8-9dc22def8958"
    Path: <sip:term@pcscf.open-ims.test:4060;lr>
    Service-Route: <sip:orig@scscf.open-ims.test:6060;lr>
    Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY, PUBLISH, MESSAGE, INFO
    P-Charging-Function-Addresses: ccf=pri_ccf_address
    Server: Sip EXpress router (2.1.0-dev1 OpenIMSCore (i386/linux))
    Content-Length: 0
```

Ilustración 24. Respuesta OK al REGISTER del usuario Alice.

Por último, se puede comprobar que el registro se ha realizado, volviendo a la máquina donde se aloja el Open IMS Core y observando en la interfaz web de IMS si el campo "*Reg.*

Status" de la identidad pública ha sido cambiado a "*Registered",* tal y como se muestra en la Ilustración 25.

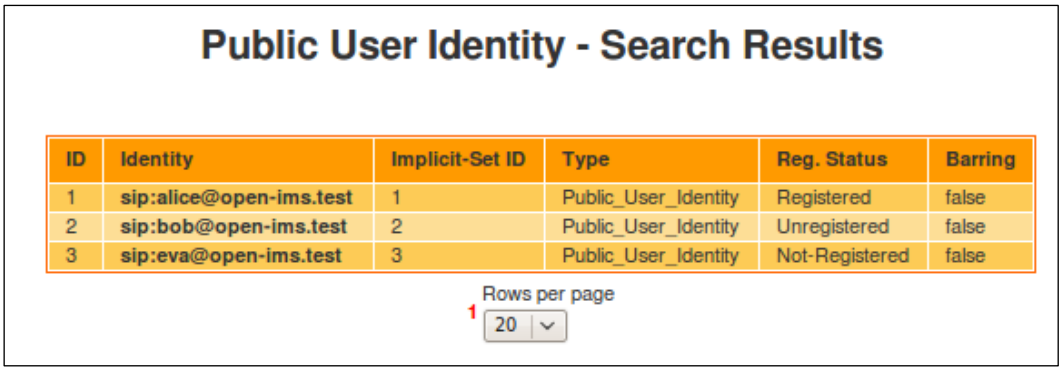


Ilustración 25. Interfaz web de IMS con las direcciones públicas de los usuarios y el estado de las mismas.

4.1.2. Llamada entre usuarios

La segunda prueba que se va a realizar es una llamada entre dos usuarios. Para ello se comienza lanzando un cliente Monster en cada uno de los nodos del escenario emulado en CORE y se registrará, de la misma forma que en el apartado anterior, a un usuario en cada cliente abierto, es decir, se registra en uno al usuario Alice y el otro al usuario Bob. Una vez registrado el cliente, se procede a añadir al otro usuario como contacto, con su dirección pública. Se realiza este paso en los dos usuarios.

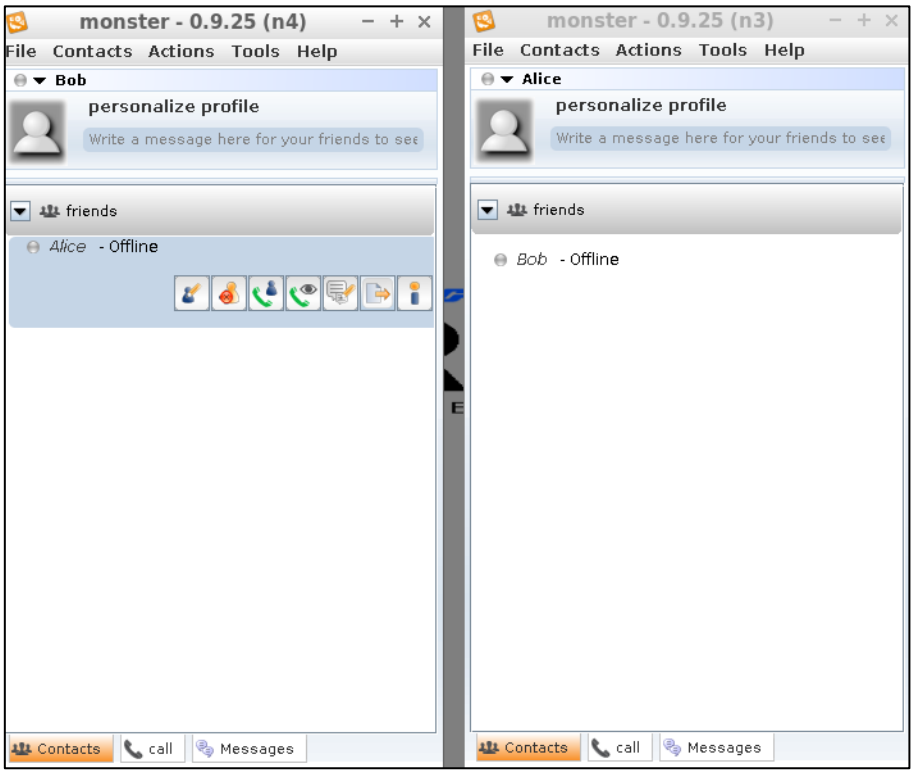


Ilustración 26.Dos clientes Mosnters cada uno con un usuario y con un amigo agregado.

En este caso Bob inicia un vídeollamada con Alice. Al iniciar el proceso aparecen dos nuevas ventanas[1] desde las que se lleva la gestión de la llamada entrante según cada usuario. Desde la pantalla del usuario Alice se puede observar quién es el que realiza la llamada entrante, y se puede aceptar y rechazar la llamada y desde la pantalla del usuario Bob se observa a quién se está realizando la llamada y se puede cancelar la misma.

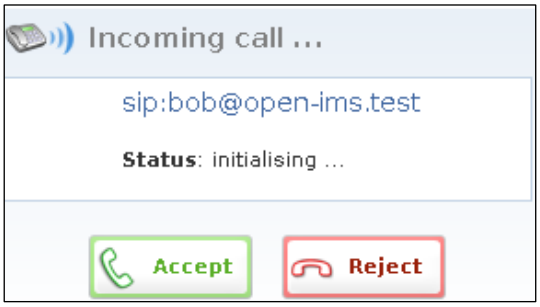


Ilustración 27. Interfaz de gestión de llamada entrante del usuario Alice.

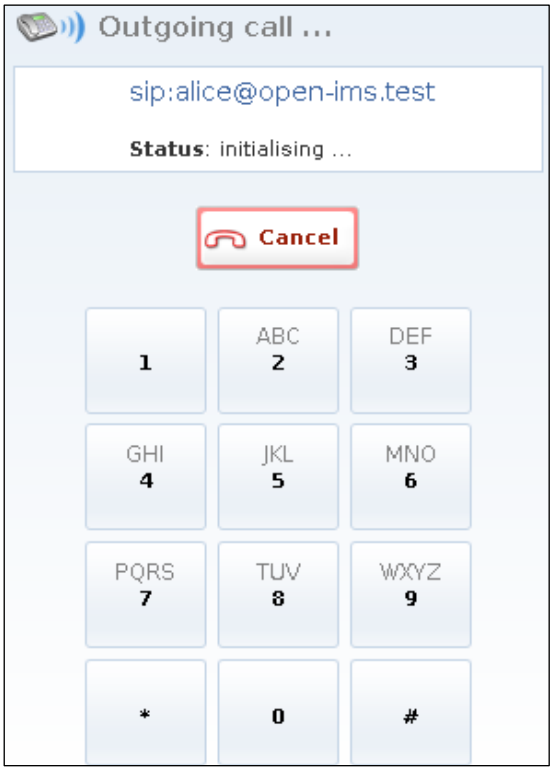


Ilustración 28. Interfaz de llamada realizada del usuario Bob.

Si Alice acepta la vídeollamada, se abren dos nuevas pantallas desde donde se realiza la llamada y se lleva la gestión de la misma.

[1] Dado que se tiene redirigida la salida gráfica de los nodos emulados hacia la pantalla de la máquina virtual que aloja el CORE, es ahí donde aparecen todas las ventanas de las interfaces gráficas de los programas ejecutados en los distintos nodos emulados.

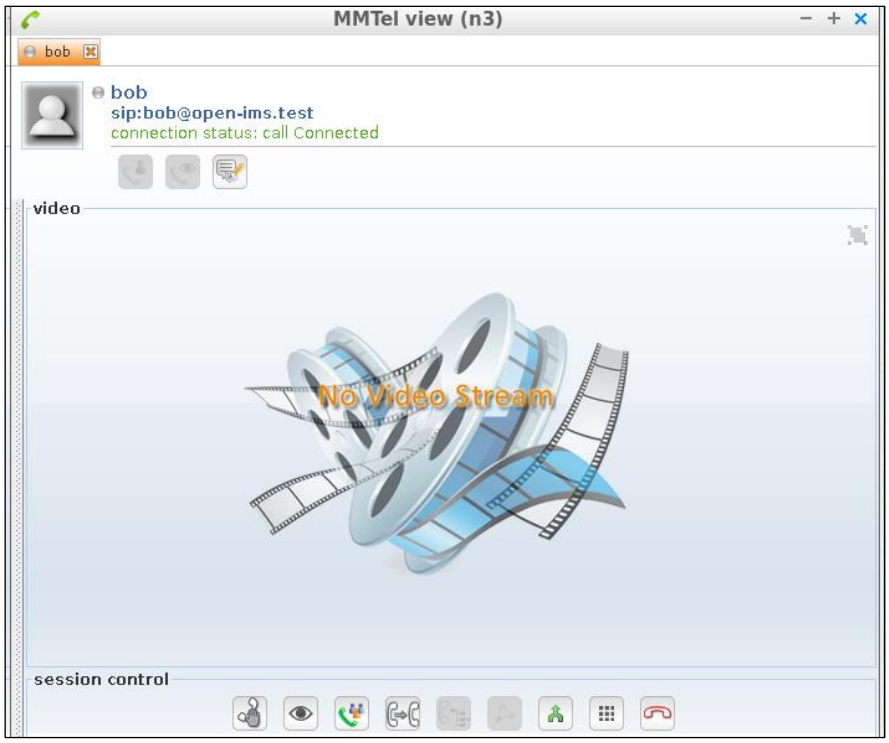


Ilustración 29. Interfaz de usuario de Bob en la vídeollamada.

Se puede capturar todo el tráfico de mensajes SIP con Wireshark, y analizar cada mensaje con todos sus campos. A continuación se muestra el mensaje INVITE enviado por Bob para comenzar la llamada. En este mensaje también se puede observar el cuerpo del mismo, es decir, la carga SDP. Los datos SDP describen la versión del protocolo (la 0), el identificador de la sesión, el tiempo de sesión y los descriptores en media, en este caso es un mensaje de audio que se trasmite por el puerto 23004 utilizando el protocolo RTP/AVP. Además se indica el formato que Bob soporta de audio y vídeo y más información de conexión, como por ejemplo la IP de origen.

```
▽ Request-Line: INVITE sip:alice@open-ims.test SIP/2.0
    Method: INVITE
    [Resent Packet: False]
▽ Message Header
    Call-ID: ee5856a610f07e16d3098e25ed7b85ab@10.0.1.20
  ▷ CSeq: 13 INVITE
  ▷ From: <sip:bob@open-ims.test>;tag=1017
  ▷ To: <sip:alice@open-ims.test>
  ▷ Via: SIP/2.0/UDP 10.0.1.20:5060;branch=z9hG4bK6a4020862364ee8f3cad8d860d641a88363233
    Max-Forwards: 20
    Route: <sip:orig@scscf.open-ims.test:6060;lr>
    Content-Type: application/sdp
  ▽ Contact: "Bob" <sip:bob@10.0.1.20:5060>
    ▷ Contact Binding: "Bob" <sip:bob@10.0.1.20:5060>
    User-Agent: monster Version: 0.9.25
    Content-Length: 239
▽ Message Body
```

Ilustración 30. Mensaje INVITE.

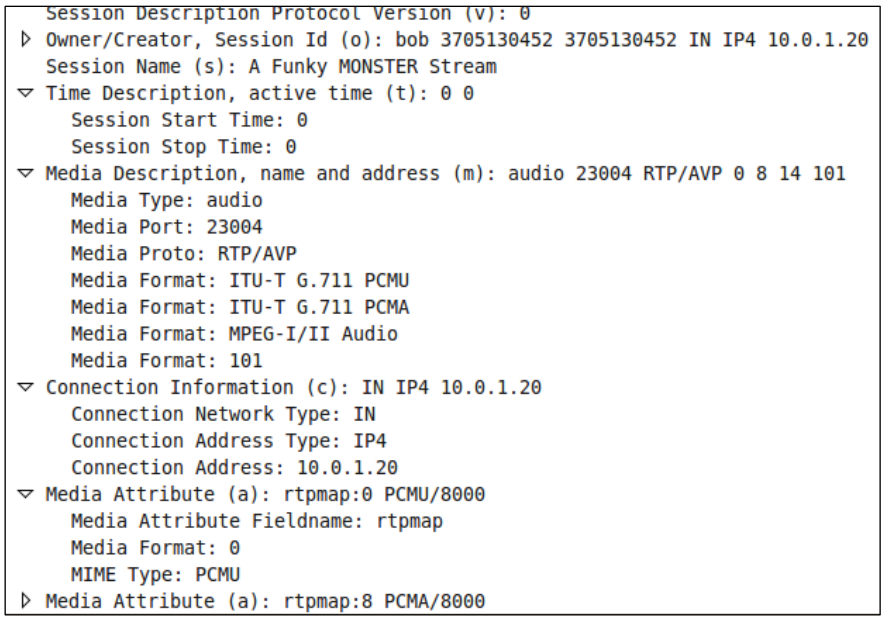


Ilustración 31. Carga SDP del mensaje INVITE.

Se decide colgar la llamada desde el programa del usuario Bob. El resultado esperado será que el UA de Bob envíe al de Alice un mensaje de BYE. Y, efectivamente, con Wireshark se captura dicho mensaje, que se muestra en la Ilustración 32.

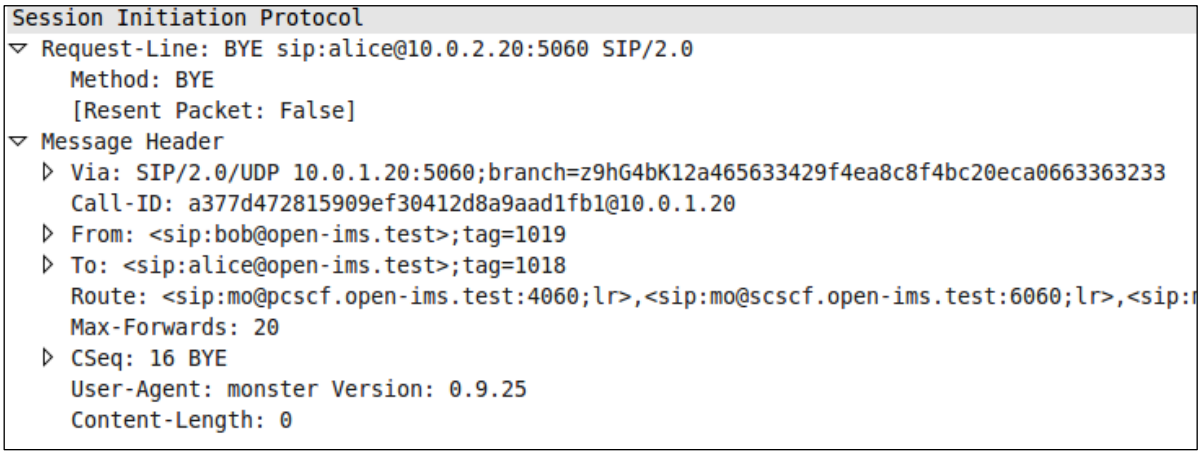


Ilustración 32. Mensaje BYE.

Además, Wireshark contiene una herramienta accesible desde el menú *Statistics -> VoIP Calls* que te muestra un diagrama con los mensajes intercambiados en la llamada. El diagrama de esta llamada es el que se muestra en la Ilustración 33. Se puede apreciar cómo el usuario Bob situado en el nodo n4 del escenario CORE (IP: 10.0.1.20) inicia la vídeollamada, el usuario Alice (registrado en 10.0.2.20) acepta la misma, y esta se ejecuta hasta que Bob inicia el proceso de terminarla. También se observan como todos los mensajes de señalización SIP pasan por la máquina que contiene los servidores de IMS, con dirección IP 172.20.55 (Ilustración 15).

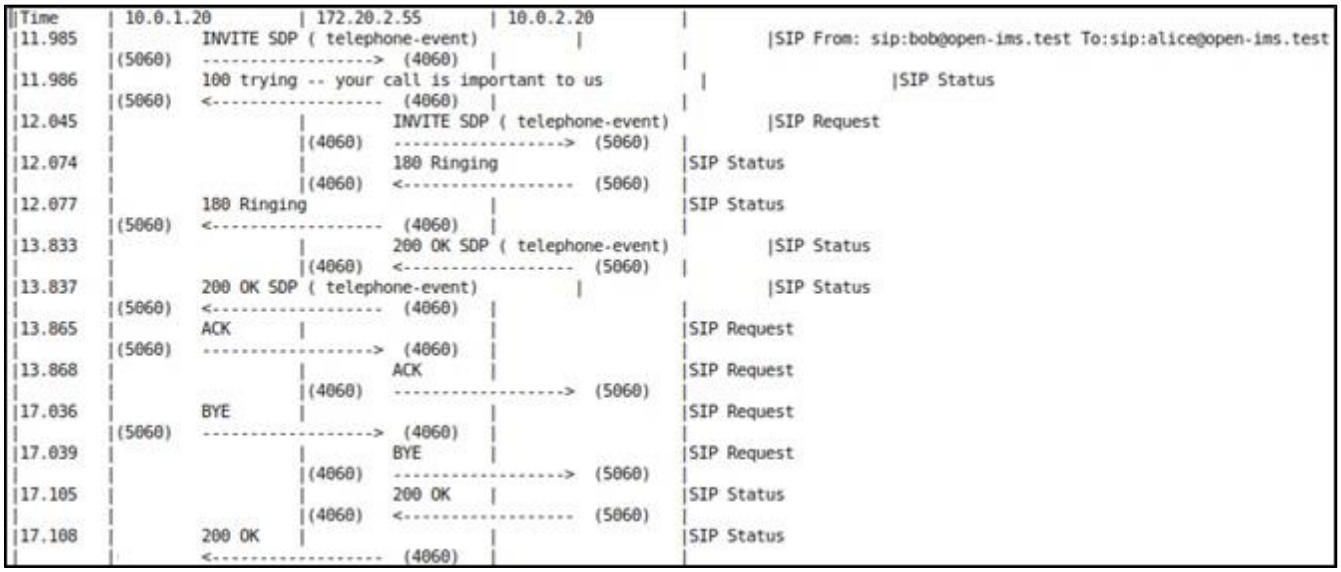


Ilustración 33.Intercambio de mensajes SIP de la llamada.

4.1.3. Llamada cancelada

En esta última prueba con el cliente Monster, Bob intenta realizar una llamada a Alice pero esta la rechaza antes de empezarla.

El intercambio de mensajes capturados con Wireshark es el mostrado en la Ilustración 34. El mensaje para terminar la sesión es un 406 Busy, ya que la cancelación de la misma se produce antes de que empiece el intercambio de vídeo.

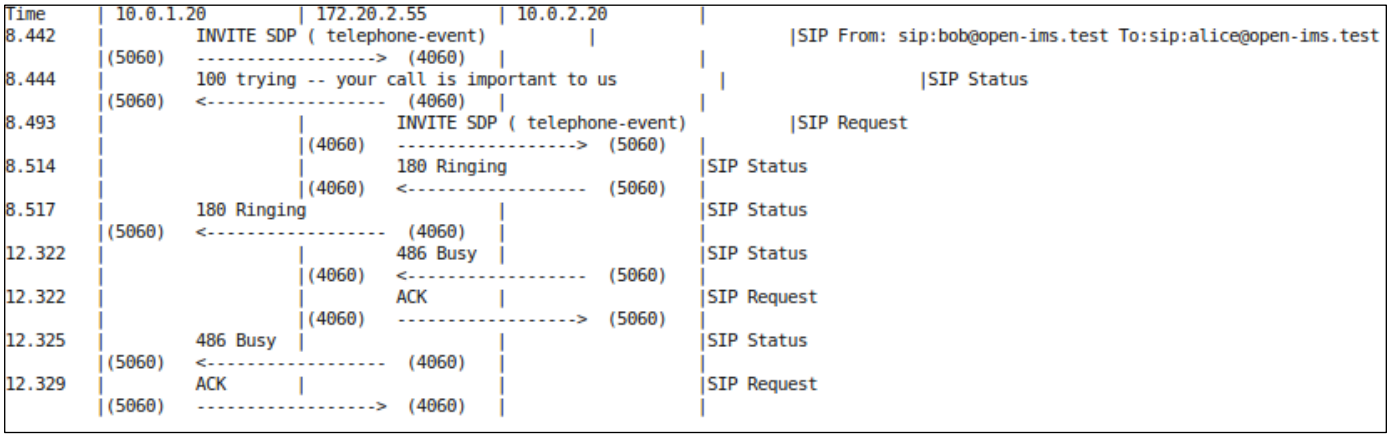


Ilustración 34. Intercambio de mensajes de una llamada cancelada.

4.2. Pruebas con scripts de SIPp

4.2.1. Registro y des-registro de un usuario

En este caso también se comienzan las pruebas con el registro de un usuario. Para ello, una vez realizada la configuración del apartado V. Instalación de clientes IMS del ANEXO A, se procede al registro del primer usuario contenido en el archivo users.csv (ver Ilustración 35) en un nodo del escenario emulado en CORE (el n4 en esta prueba). Para ello, se ejecuta el script SIPp *register* en un terminal del nodo (n4) donde queremos hacer el registro. Todos los ficheros utilizados están detallados en el ANEXO B: Scripts de SIPp.

	A	B	C	D	E	F	G	H
1	SEQUENTIAL;;							
2	#Usuarios a registrar en core IMS;;							
3	alice;alice;[authentication username=alice@open-ims.test password=alice]							
4								

Ilustración 35.Contenido del archivo users.csv

El terminal desde dónde se ejecuta el script muestra información acerca de la evolución de la ejecución del mismo. El resultado es el esperado, tal y como se muestra en la Ilustración 36: se ha enviado un REGISTER que ha sido contestado por un mensaje 401 Unauthorized y más tarde otro REGISTER que ha sido aceptado con un mensaje 200 (OK). También se pueden apreciar otros datos, como el tiempo de la llamada o el puerto utilizado.

```
------------------------------ Scenario Screen -------- [1-9]: Change Screen --
 Call-rate(length)   Port   Total-time  Total-calls  Remote-host
 10.0(0 ms)/1.000s   5061       2.42 s            1  172.20.2.55:5060(UDP)

 Call limit reached (-m 1), 0.000 s period  0 ms scheduler resolution
 0 calls (limit 60)                      Peak was 1 calls, after 0 s
 0 Running, 2 Paused, 0 Woken up
 0 dead call msg (discarded)             0 out-of-call msg (discarded)
 1 open sockets

                              Messages  Retrans   Timeout   Unexpected-Msg
   REGISTER ---------->         1         0         0
        401 <----------         1         0         0         0
   REGISTER ---------->         1         0         0
        200 <----------         1         0         0         0
      Pause [   2000ms]         1                             0
------------------------------ Test Terminated --------------------------------
```

Ilustración 36. Petición de Registro a través de SIPp.

Al terminar el intercambio de mensajes SIP definido en el script, se muestra un resumen de las estadísticas de toda la interacción.

```
---------------------------- Statistics Screen ------- [1-9]: Change Screen --
 Start Time              | 2017-06-06   17:42:54:545    1496763774.545657

 Last Reset Time         | 2017-06-06   17:42:56:969    1496763776.969737

 Current Time            | 2017-06-06   17:42:56:970    1496763776.970779

-------------------------+---------------------------+--------------------------
 Counter Name            | Periodic value            | Cumulative value
-------------------------+---------------------------+--------------------------
 Elapsed Time            | 00:00:00:001              | 00:00:02:425
 Call Rate               |    0.000 cps              |    0.412 cps
-------------------------+---------------------------+--------------------------
 Incoming call created   |        0                  |        0
 OutGoing call created   |        0                  |        1
 Total Call created      |                           |        1
 Current Call            |        0                  |
-------------------------+---------------------------+--------------------------
 Successful call         |        0                  |        1
 Failed call             |        0                  |        0
-------------------------+---------------------------+--------------------------
 Call Length             | 00:00:00:000              | 00:00:02:320
------------------------------ Test Terminated --------------------------------
```

Ilustración 37. Resumen del Registro.

Se puede comprobar que el registro se ha realizado con éxito volviendo a la máquina donde se aloja el Open IMS Core y observando en la interfaz web de IMS si el campo "*Reg. Status*" de la identidad pública ha sido cambiado a "*Registered",* tal y como se muestra en la Ilustración 38.

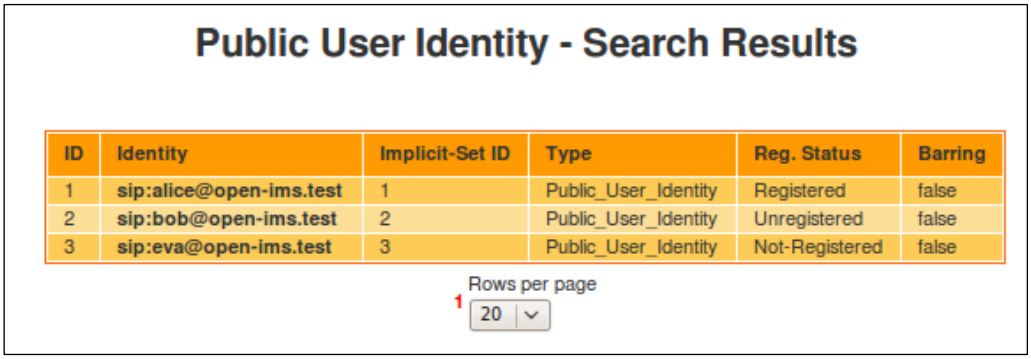


Ilustración 38. Interfaz web de IMS con las direcciones públicas de los usuarios y el estado de las mismas.

De la misma forma que se ha realizado el registro, se procede a des-registrar a un usuario, lo único que cambia es el script que se utiliza, en este caso se necesita el unregister. El intercambio de mensajes es el que se muestra en la Ilustración 39.

```
------------------------------ Scenario Screen -------- [1-9]: Change Screen --
Call-rate(length)   Port   Total-time  Total-calls  Remote-host
10.0(0 ms)/1.000s   5061       2.23 s            1  172.20.2.55:5060(UDP)

Call limit reached (-m 1), 0.223 s period  1 ms scheduler resolution
0 calls (limit 60)                  Peak was 1 calls, after 0 s
0 Running, 2 Paused, 2 Woken up
0 dead call msg (discarded)         0 out-of-call msg (discarded)
1 open sockets

                                 Messages  Retrans   Timeout   Unexpected-Msg
  REGISTER ---------->           1         0         0
       401 <----------           1         0         0         0
  REGISTER ---------->           1         0         0
       200 <----------           1         0         0         0
     Pause [   2000ms]           1                             0
------------------------------ Test Terminated --------------------------------
```

Ilustración 39. Intercambio de mensajes para des-registrar.

Tras este proceso el usuario debe aparecer como *"Not-register"* en la interfaz web de IMS, y comprobamos (ver Ilustración 41) que así es.

Public User Identity - Search Results

ID	Identity	Implicit-Set ID	Type	Reg. Status	Barring
1	sip:alice@open-ims.test	1	Public_User_Identity	Not-Registered	false
2	sip:bob@open-ims.test	2	Public_User_Identity	Not-Registered	false
3	sip:eva@open-ims.test	3	Public_User_Identity	Not-Registered	false

Ilustración 40. Interfaz web de IMS donde se muestra el usuario des-registrado.

4.2.2. Llamada entre usuarios

Para realizar una llamada entre dos usuarios se tiene que registrar a cada uno en un nodo de la red emulada en CORE (siguiendo el procedimiento que acabamos de describir, Alice en n4 y Bob en n3). Una vez que los dos usuarios están registrados, se comprueba si el registro se ha realizado con éxito en la máquina IMS, tal y como se muestra en la Ilustración 41.

Public User Identity - Search Results

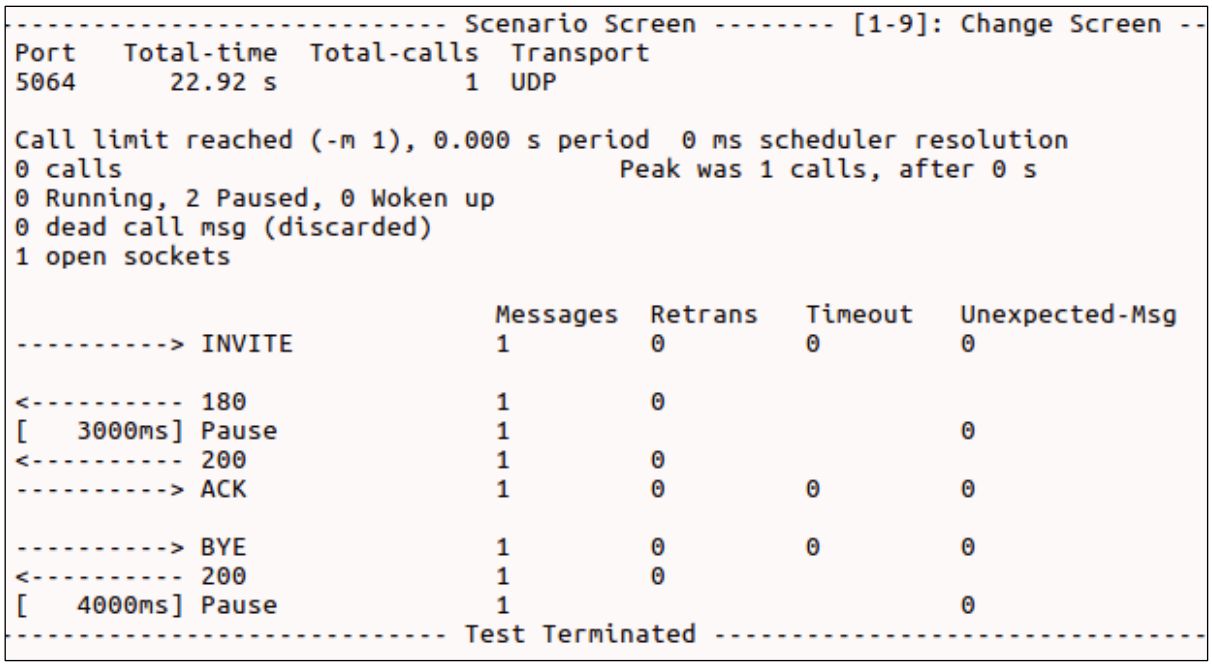
ID	Identity	Implicit-Set ID	Type	Reg. Status	Barring
1	sip:alice@open-ims.test	1	Public_User_Identity	Registered	false
2	sip:bob@open-ims.test	2	Public_User_Identity	Registered	false
3	sip:eva@open-ims.test	3	Public_User_Identity	Not-Registered	false

Ilustración 41. Interfaz web de IMS donde se muestran los usuarios registrados.

Uno de los usuarios actuará como cliente (Alice), y otro como servidor (Bob). Para que Alice actúe como cliente hay que ejecutar, en un terminal del nodo n4, el script de *uac*. Este script (la evolución de su ejecución se puede ver en la Ilustración 42) envía un INVITE, se queda esperando para enviar un ACK y, tras un cierto tiempo, es el mismo que envía un BYE para terminar la llamada. Para que Bob actúe como servidor, hay que ejecutar, en un terminal del nodo n3, el script uas. Este script (la evolución de su ejecución se puede ver en la Ilustración 43 se queda esperando por un INVITE, al que contesta con un 180 Ringing. Nótese que Alice recibe antes del 180, un 100 Trying (que bob no genera), pero ese mensaje lo genera el p-cscf de IMS para indicar que está cursando la llamada. Una vez el usuario Bob "contesta", se envía desde el UAS el 200 OK, y luego se espera por el ACK de forma que se completa el establecimiento de la sesión. Tras esto, el UAS se queda esperando por el BYE al que contesta con el 200 OK. Si la llamada se realiza con éxito, en el terminal aparecerá un resumen de la misma (por ejemplo, el resumen en el terminal de Bob puede verse en la Ilustración 44).

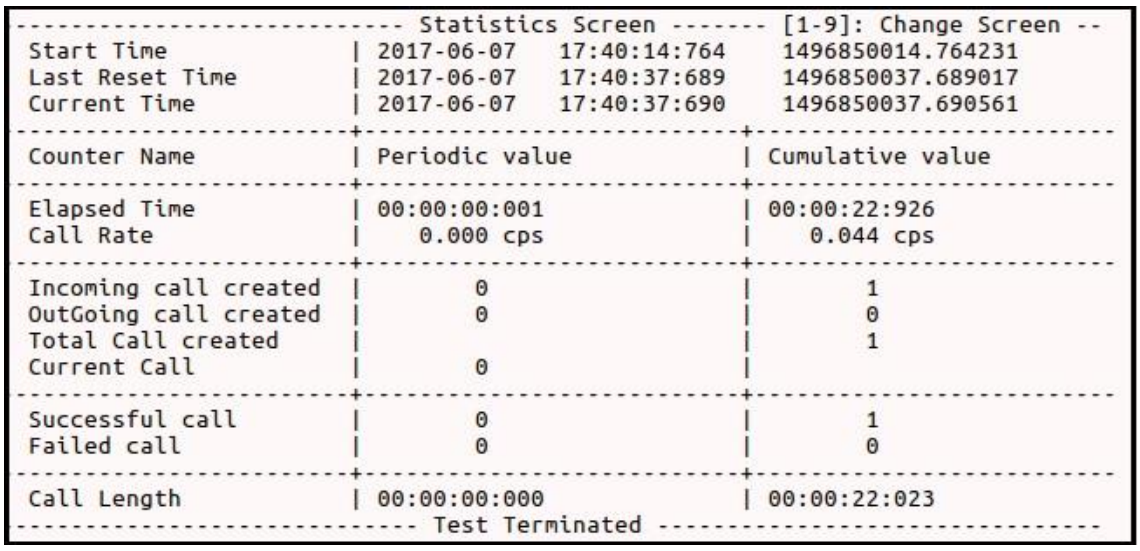
```
------------------------------ Scenario Screen -------- [1-9]: Change Screen --
  Call-rate(length)   Port   Total-time  Total-calls  Remote-host
  10.0(0 ms)/1.000s   5063      21.59 s            1  172.20.2.55:5060(UDP)

  Call limit reached (-m 1), 0.000 s period  0 ms scheduler resolution
  0 calls (limit 510)                  Peak was 1 calls, after 0 s
  0 Running, 2 Paused, 0 Woken up
  0 dead call msg (discarded)          0 out-of-call msg (discarded)
  1 open sockets

                                Messages  Retrans   Timeout   Unexpected-Msg
      INVITE ---------->        1         0
         100 <----------        1         0         0         0
         180 <----------        1         0         0         0
         200 <----------        1         0         0         0
         ACK ---------->        1         0
              [ NOP ]
       Pause [    15.0s]        1                             0
         BYE ---------->        1         0
         200 <----------        1         0         0         0

       Pause [   2000ms]        1                             0
------------------------------ Test Terminated --------------------------------
```

Ilustración 42. Datos de la llamada del terminal de Alice.



```
---------------------------- Scenario Screen -------- [1-9]: Change Screen --
Port   Total-time  Total-calls  Transport
5064      22.92 s            1  UDP

Call limit reached (-m 1), 0.000 s period  0 ms scheduler resolution
0 calls                                  Peak was 1 calls, after 0 s
0 Running, 2 Paused, 0 Woken up
0 dead call msg (discarded)
1 open sockets

                                 Messages  Retrans   Timeout   Unexpected-Msg
----------> INVITE               1         0         0         0

<---------- 180                  1         0
[   3000ms] Pause                1                             0
<---------- 200                  1         0
----------> ACK                  1         0         0         0

----------> BYE                  1         0         0         0
<---------- 200                  1         0
[   4000ms] Pause                1                             0
---------------------------- Test Terminated --------------------------------
```

Ilustración 43. Datos de la llamada del terminal de Bob.



```
---------------------------- Statistics Screen ------- [1-9]: Change Screen --
 Start Time              | 2017-06-07   17:40:14:764    1496850014.764231
 Last Reset Time         | 2017-06-07   17:40:37:689    1496850037.689017
 Current Time            | 2017-06-07   17:40:37:690    1496850037.690561
-------------------------+-----------------------------+--------------------------
 Counter Name            | Periodic value              | Cumulative value
-------------------------+-----------------------------+--------------------------
 Elapsed Time            | 00:00:00:001                | 00:00:22:926
 Call Rate               |    0.000 cps                |    0.044 cps
-------------------------+-----------------------------+--------------------------
 Incoming call created   |        0                    |        1
 OutGoing call created   |        0                    |        0
 Total Call created      |                             |        1
 Current Call            |        0                    |
-------------------------+-----------------------------+--------------------------
 Successful call         |        0                    |        1
 Failed call             |        0                    |        0
-------------------------+-----------------------------+--------------------------
 Call Length             | 00:00:00:000                | 00:00:22:023
---------------------------- Test Terminated --------------------------------
```

Ilustración 44. Resumen de la llamada.

Si se capturan los mensajes SIP intercambiados con Wireshark y se saca el dibujo de la llamada se pueden confirmar todo el flujo de la llamada, que es el esperado.

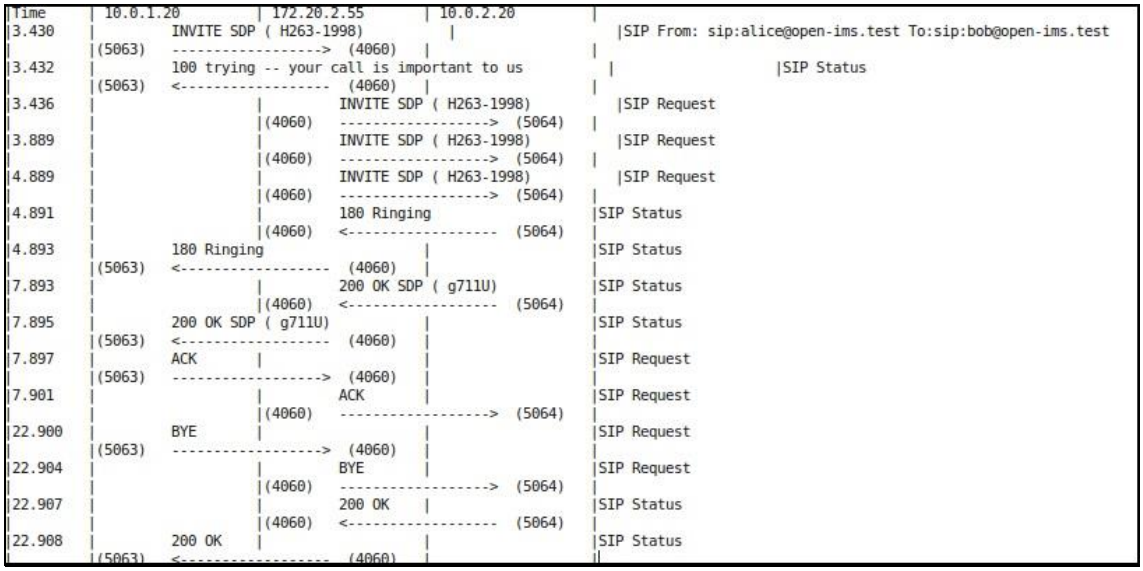


Ilustración 45. Dibujo de la llamada obtenido con Wireshark.

4.3. Conclusiones del capítulo.

En este capítulo se han mostrado las pruebas realizadas para comprobar el correcto funcionamiento del entorno desarrollado en el proyecto. Estas pruebas se han realizado con dos clientes IMS, el Monster y los scripts de SIPp, de forma que se valida que el entorno desarrollado cumple con los objetivos planteados. En concreto, hemos verificado que es posible crear distintos diálogos SIP desde nodos en escenarios de red emulados en CORE, usando una arquitectura IMS desplegada en otra máquina virtual.

Este capítulo sirve de base al siguiente, que nos ofrece las conclusiones generales del proyecto.

5. CONCLUSIONES Y FUTURAS LÍNEAS DE TRABAJO

En este capítulo se va a proceder al análisis de las conclusiones del trabajo realizado, así como analizar algunas líneas de trabajo que se podrían afrontar en el futuro.

5.1. Conclusiones

La conclusión principal es que se ha conseguido el objetivo del proyecto, es decir, dotar al emulador de redes CORE con un sistema IMS para proporcionar servicios audiovisuales.

Se ha conseguido desarrollar, por lo tanto, una plataforma donde se puede crear cualquier tipo de red emulada, y donde se dispone de un sistema IMS que proporciona todo lo necesario para la gestión de sesiones para servicios multimedia. Esta plataforma es adaptable a cualquier red bajo estudio, de forma que se puede utilizar para el desarrollo e implantación de servicios multimedia a través de redes IP.

Las tecnologías usadas, como CORE y Open IMS han sido corazón del proyecto. Ambas tecnologías son de fácil acceso y su instalación es sencilla gracias a los distintos manuales que ofrecen. De todas formas, ha habido que superar una serie de retos:

- Configurar Open IMS para atender peticiones procedentes de máquinas distintas de donde se ejecutan los servidores IMS.

- Configurar un servidor DNS para atender a todas las máquinas (virtuales dentro del escenario de red emulado, virtuales, o reales) del escenario.

- Gestionar el direccionamiento IP del escenario, tanto de máquinas virtuales dentro del escenario de red emulado, de máquinas virtuales y de máquinas reales. Como parte de esta tarea se configuró un servidor DHCP para servir direcciones de forma más sencilla y el servidor de DNS del escenario.

- Configurar la comunicación IP de las distintas máquinas virtuales y de éstas con las máquinas virtuales en la emulación.

- Instalación y uso de clientes IMS en las máquinas virtuales del escenario de red emulado.

Se ha validado que la plataforma funciona adecuadamente para probar servicios audiovisuales con dos herramientas: un agente de usuario IMS típico, como el que usarían usuarios reales (Monster), y una herramienta de experimentación con señalización SIP (SIPp) que permite de forma muy flexible definir intercambios de señalización SIP y comprobar que se producen correctamente. Con ambas herramientas se ha podido registrar y desregistrar usuarios, y establecer sesiones. Además, se ha capturado el tráfico intercambiado entre los agentes de usuario, y con los servidores IMS, mostrando lo potente que es la plataforma desarrollada como entorno para la experimentación y prueba de servicios audiovisuales sobre redes IP.

5.2. Futuras líneas de trabajo

A continuación se analizan posibles trabajos futuros en línea con este proyecto:

- Automatizar la configuración del entorno en IMS y en CORE en el que se desarrolla el proyecto.

- Conectar una maquina física al escenario, de forma que un extremo de la comunicación esté en el escenario de red emulado y el otro en una máquina en una red real que podrá visualizar vídeo en las llamadas.

- Implantar un servidor de vídeo al que se le puedan pedir servicios usando IMS.

- Incorporar otros tipos de protocolos se señalización alternativos o complementarios a SIP que permitiesen hacer estudios comparativos, como RTSP o XMPP.

6. PLANIFICACIÓN, PRESUPUESTO Y ENTORNO SOCIO-ECONÓMICO

6.1. Planificación

6.1.1. Fases de desarrollo

La duración de la realización del proyecto ha sido de alrededor de 6 meses. A continuación se resumen las fases en las que se ha dividido y el tiempo empleado en cada una de ellas por los dos ingenieros participantes en el proyecto:

1. **Planteamiento del proyecto.**
   En esta fase se expuso la idea del proyecto en general, concretando las necesidades y los objetivos generales del mismo.
   - Duración: 21 horas.

2. **Estudio de las tecnologías utilizadas.**
   Esta fase consistió en un estudio de las posibles tecnologías a emplear para una mejor compresión de los objetivos del proyecto, además de posibles alternativas a las tecnologías expuestas en el planteamiento del proyecto.
   - Duración: 20 horas.

3. **Diseño inicial del sistema.**
   Fase en la que se concretó el diseño y la arquitectura de los componentes que forman el sistema, así como la interacción entre ellos.
   - Duración: 21 horas.

4. **Instalación de los recursos utilizados.**
   Instalación y configuración de las herramientas utilizadas: red creada por el router donde residen VirtualBox con CORE y VMWare con IMS. Esta fase llevó más tiempo del planeado, debido a las múltiples complicaciones que surgieron en la instalación de Open IMS.
   - Duración: 74 horas.

5. **Configuración del entorno de implementación.**
   Se realizó a la par que la última fase de la instalación y fue el corazón del proyecto, llevando también una alta proporción del tiempo total del mismo. En ella se configuraron las máquinas virtuales y la red, para un correcto funcionamiento de todo el conjunto. Además se configuró el escenario donde se realizan las pruebas.
   - Duración: 92 horas.

6. **Pruebas y conclusiones.**
   En esta última fase se realizaron pruebas para comprobar el correcto funcionamiento del sistema. También aquí se extrajeron las conclusiones del proyecto.
   - Duración: 107 horas.

7. **Redacción de la memoria.**
   Redacción de la documentación de todo el proyecto. Se llevó a cabo a lo largo de casi todo el desarrollo del proyecto, comenzando tras la fase del diseño del sistema.
   - Duración: 100 horas.

Durante todo el desarrollo del proyecto ha existido otra fase correspondiente con el seguimiento del trabajo, que consistía en reuniones con el tutor para resolver fallos y dudas, así como para verificar los avances en el desarrollo del proyecto.

6.1.2. Diagrama de fases

En el diagrama de Gant que se muestra a continuación se representa la duración de cada fase del proyecto en días.

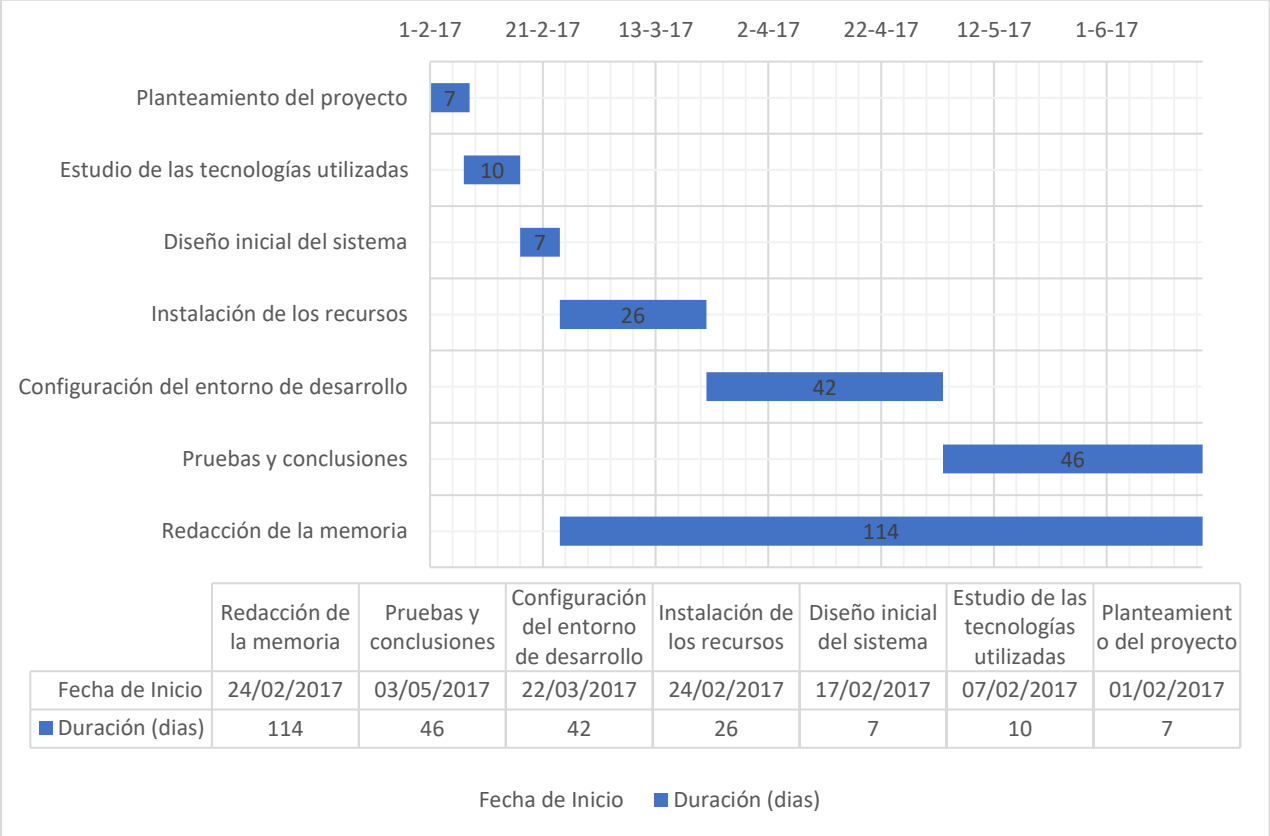


Ilustración 46. Diagrama de fases del proyecto.

6.1.3. Diagrama Pert

A continuación se muestra el diagrama pert del proyecto, donde las etapas tienen la identificación del número de fase indicado del apartado 6.1.1. Los números indicados en rojo son los tiempos (en días) early y en verde se indican los tiempos last, siendo ambos iguales en todas las fases ya que dichas fases no se han realizado con holgura, es decir, el retraso en alguna de las actividades supondría el retraso de todo el proyecto.

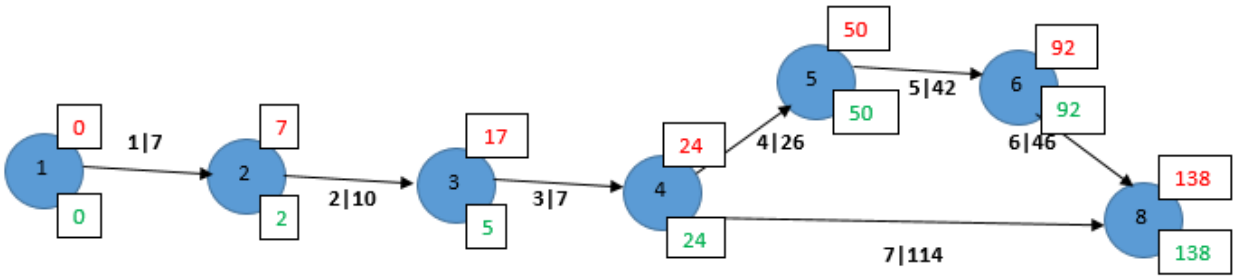


Ilustración 47. Diagrama Pert del proyecto.

6.2. Presupuesto

6.2.1. Recursos empleados

- **Recursos humanos:**
  - 1 Ingeniero Senior.
  - 1 Ingeniero Junior.

- **Recursos materiales:**
  - 1 PC portátil de gama media.
  - 1 Router inalámbrico: Wireless – G, de Linksys.

- **Licencias de software:**
  - Microsoft Office 2013.
  - Conexión a Internet durante 5 meses.
  - VirtualBox versión 5.1.14.
  - VMWare Workstation 12 player.
  - VCore 4.6.
  - OpenSourceIMSCore V.01 2009.
  - SIPP 3.3.
  - FOKU's Monster
  - Whireshark 2.2.6.

6.2.2. Presupuesto detallado

1.- Autor: Eva Mª Urbán Martínez

2.- Departamento: Ingeniería Telemática

3.- Descripción del Proyecto:

- Titulo: Infraestructura de señalización para proporcionar servicios audiovisuales en un emulador de redes.

- Duración (meses): 6 meses

- Tasa de costes Indirectos: 20%

4.- Presupuesto total del Proyecto: 10637,27 €

5.- Desglose presupuestario:

A continuación se realiza un análisis de los costes asociados a la realización del proyecto. Estos costes se encuentran desglosados en mano de obra, material y licencias de software.

Los primeros costes analizados son los costes asociados a la mano de obra. En primer lugar se describe un desglose de las actividades realizadas por las dos personas involucradas en la realización del proyecto, es decir, un ingeniero senior y otro junior. Tras analizar la asociación de las horas empleadas, se muestra en forma de tabla los costes asociados a la mano de obra.

TAREAS	DEDICACIÓN INGENIERO JUNIOR [HORAS]*	DEDICACIÓN INGENIERO SENIOR [HORAS]
PLANTEAMIENTO DEL PROYECTO	14	7
ESTUDIO DE LAS TECNOLOGÍAS UTILIZADAS	20	0
DISEÑO INICIAL DEL SISTEMA	14	7
INSTALACIÓN DE LOS RECURSOS UTILIZADOS	52	22
CONFIGURACIÓN DEL ENTORNO DE IMPLEMENTACIÓN	84	8
PRUEBAS Y CONCLUSIONES	92	15
REDACCIÓN DE LA MEMORIA.	80	20
TOTAL	**356**	**79**

Tabla 3. Desglose del tiempo empleado en cada tarea.

*Las horas se han calculado en base a una dedicación de dos horas por jornada.

TIPO DE MANO DE OBRA	CANTIDAD [HORAS]	COSTE UNITARIO [€/H]	COSTE TOTAL [€]
INGENIERO SENIOR	79	30	2370
INGENIERO JUNIOR	356	18	6408
		Total	8778

Tabla 4. Costes personal.

A continuación se muestran en forma de tabla los costes asociados a la adquisición del material necesario para la elaboración del proyecto.

CONCEPTO	CANTIDAD	COSTE [€]	DEDICACIÓN [MESES]	PERIODO DE DEPRECIACIÓN [MESES]	COSTE IMPUTABLE[€]
PC PORTÁTIL DE GAMA MEDIA	1	590	5	72	40,97
ROUTER INALÁMBRICO	1	60	5	72	4,17
				Total	45,14

Tabla 5. Costes materiales.

Se muestra en forma de tabla los costes asociados a la única licencia de software que ha sido empleada y que no es de libre adquisición

CONCEPTO	CANTIDAD [MESES]	COSTE UNITARIO [€]	COSTE TOTAL [€]
MICROSOFT OFFICE	5	8,25	41,25
		Total	41,25

Tabla 6. Costes licencias de software.

Por último, se muestra en forma de tabla los costes totales de la realización del proyecto.

CONCEPTO	COSTE TOTAL [€]
PERSONAL	8778
MATERIAL	45,14
LICENCIA DE SOFTWARE	41,25
COSTES INDIRECTOS (20%)	**1772,88**
TOTAL (SIN IVA)	**10637,27**

Tabla 7. Costes totales

6.3. Entorno Socio-económico

El avance de Internet y la convergencia hacia un mundo "todo sobre IP" está transformando el sector de las telecomunicaciones. El despliegue de IMS en las redes de telecomunicaciones permite a los operadores integrar las llamadas fijas y móviles con servicios multimedia, como son la televisión o el vídeo bajo demanda, todo ello mediante redes IP.

En el mercado español, la mayoría de las operadoras ofrecen paquetes de servicios que buscan atraer a usuarios, cada vez más exigentes en cuanto al coste. Anteriormente, dichos servicios se ofrecían con infraestructuras diferentes para cada uno o incluso con proveedores distintos, sin embargo, hoy en día el operador puede ofrecer a sus clientes el acceso a los servicios multimedia a través de una infraestructura común basada en redes IP. Todo ello permite a las operadoras solventar las demandas de los usuarios con unos precios muy competitivos ya que se reducen mucho los costes en el despliegue de la red.

Debido a los múltiples beneficios de la implantación de IMS, a partir de 2010, tras el congreso de GSMA (Global System for Mobile communications Asociation), se impulsa el despliegue de IMS en operadoras de telefonía móvil y fija. Es Ericsson la empresa encargada de implementar la primera red comercial basada en IMS, que se instaló en el 2005 en Telefónica España. A partir de este momento, las grandes compañías empiezan a implantar estas redes, combinando la conmutación de circuitos con la conmutación de paquetes. Cada vez más operadores ofrecen el servicio VoIP, según datos de la CNMC (Comisión nacional de los mercados y a la competencia) [34], a finales de 2014 el número de líneas fijas que usaban VoIP superó la cifra de 4 millones, siendo estas un 20,8% del total.

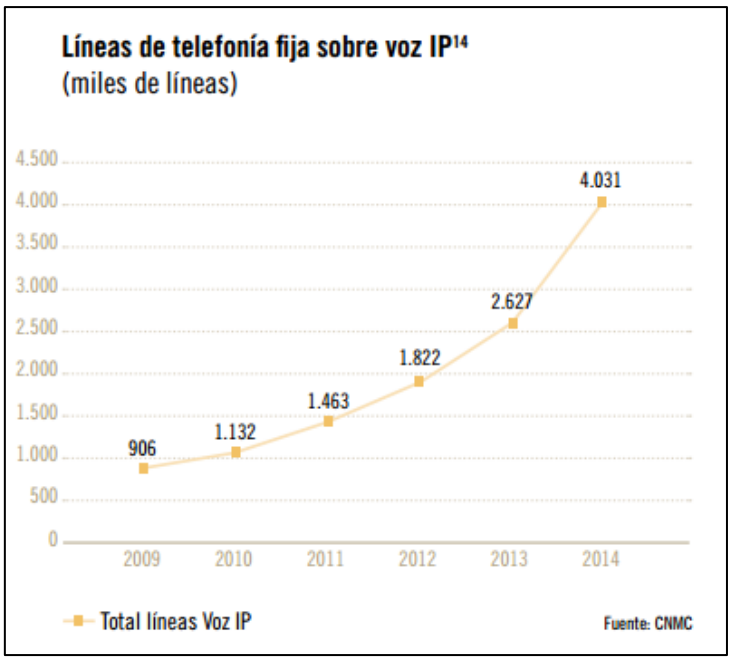


Ilustración 48. Líneas de telefonía fija sobre IP. (Tomada de [34])

En la actualidad la implantación de IMS está en pleno desarrollo, se está avanzando hacia un futuro en el que todas las comunicaciones se basarán en redes IP.

En cuanto a SIP sólo, sin IMS por detrás, se sabe que existen multitud de aplicaciones que aplican dicho protocolo o algún otro protocolo propietario basado en SIP (por ejemplo y 3CXPhone), pero es muy difícil definirlo en cifras, ya que estos datos suelen ser privados.

6.3.1. Plan de explotación del proyecto

En el este punto en el que se encuentra el sector, son necesarias herramientas que permitan estudiar el despliegue de servicios IMS en las redes. Aquí es donde aparece este proyecto, que pretende ser una herramienta de ayuda en el estudio de soluciones de servicios audiovisuales en redes en entornos de investigación. De esta forma se busca ayudar en el desarrollo e implantación de dichas redes, de forma que se puedan mejorar los servicios que las operadoras ofrecerán a los usuarios en el futuro.

Por lo tanto, no se trata de un proyecto cuya explotación esté orientada a la venta y el beneficio económico, sino que se busca un impacto social positivo a medio plazo, en la mejora de la oferta de los servicios multimedia a través de redes IP.

7. BIBLIOGRAFÍA

[1] «3GPP Home Page.,» [Página Web]. URL: http://www.3gpp.org/. [Último acceso: 30 04 2017].

[2] «Portal ETSI-TISPAN,» [Página Web]. URL: https://portal.etsi.org/tispan/tispan_tor.asp. [Último acceso: 30 05 2017].

[3] «Common Open Research Emulator (CORE),» [Página web]. URL: https://www.nrl.navy.mil/itd/ncs/products/core [Último acceso: 11 04 2017].

[4] M. Poikselkä y G. Mayer, THE IMS: IP Multimedia Concepts and Services, 3rd Edition, ISBN: 978-0-470-72196-4. John Wiley & Sons. Marzo 17, 2009.

[5] S. Znaty, J.L. Dauphin, R. Geldwerth, «IP Multimedia Subsystem: Principios y Arquitectura,» [Libro online]. URL: http://efort.com/media_pdf/IMS_ESP.pdf [Último acceso: 10 04 2017].

[6] «The Internet Engineering Task Force (IETF),» [Página Web]. URL: http://www.ietf.org/. [Último acceso: 06 06 2017].

[7] «Open Source IMS CORE,» [Página web]. URL: http://www.openimscore.org/. [Último acceso: 06 05 2017].

[8] W. E. Castellanos. y G. Hernández, «Multimedia en Redes Móviles: SIP e IMS,» REVISTA ESPEC. EN SIST. INFOR. Y ELECT. DE TELEC vol.1, nº.2, 2007 ISSN-1909-258X.

[9] S. R. Rodera, «TRABAJO FIN DE CARRERA: Diseño e implementación de un Punto Neutro para VoIP,» Universidad Politénica Cataluña, Septiembre 2005 URL: http://upcommons.upc.edu/bitstream/handle/2099.1/3540/40477-2.pdf?sequence=2. [Último acceso: 11 04 2017].

[10] «Session Initiation Protocol,» [Wikipedia]. URL: https://es.wikipedia.org/wiki/Session_Initiation_Protocol [Último acceso: 10 04 2017]

[11] B. M. González, «TRABAJO FIN DE CARRERA: Estudio de la tecnología IMS y diseño de una solución de telefonía multimedia. ,» Universidad Politénica de Madrid, Septiembre 2012 URL: http://oa.upm.es/14111/1/PFC_BARBARA_MORATA_GONZALEZ.pdf [Último acceso: 10 04 2017].

[12] «CORE Manual,» URL: https://downloads.pf.itd.nrl.navy.mil/docs/core/core-html/ [Último acceso: 11 04 2017].

[13] J. Ahrenholz, «Comparison of CORE network emulation platforms,» [Artículo académico] Military Communications Conference, MILCOM, 2010. URL: http://ieeexplore.ieee.org/abstract/document/5680218/.

[14] J. L. Alvarez, R. Cores, J. L. Rivas, J. Rodeiro «TCL-TK, una nueva generación de interfaces visuales para usuario,» Facultad de Ciencias del Mar, Universidad de Vigo. [Artículo web] URL: http://www.quadernsdigitals.net/datos/hemeroteca/r_11/nr_183/a_2409/2409.htm [Último acceso: 14 04 2017].

[15] I. Ramírez, «Máquinas virtuales: qué son, cómo funcionan y cómo utilizarlas.,» Julio 2016. [Artículo web]. URL: https://www.xataka.com/especiales/maquinas-virtuales-que-son-como-funcionan-y-como-utilizarlas [Último acceso: 06 05 2017].

[16] Oracle., «VirtualBox,» [Página web]. URL: https://www.virtualbox.org/. [Último acceso: 06 05 2017].

[17] «VMWare,» [Página web]. URL: http://www.vmware.com/. [Último acceso: 06 05 2017].

[18] «Skype Home,» [Página web]. URL: https://www.skype.com/es/. [Último acceso: 18 05 2017].

[19] «Viber Home,» [Página web]. URL: https://www.viber.com/es/. [Último acceso: 18 05 2017].

[20] «Cisco, IP Phones.,» [Página web]. URL: http://www.cisco.com/c/en/us/products/collaboration-endpoints/ip-phones/index.html. [Último acceso: 18 05 2017].

[21] G. Calvo, «PROYECTO FIN DE CARRERA (Universidadde cataluña): Instalación de telefonía Cisco e integración y configuración de Asterisk dentro de la estructura telefónica de Labco.,» Marzo 2012. URL: https://upcommons.upc.edu/bitstream/handle/2099.1/14887/82193.pdf [Último acceso: 18 05 2017].

[22] «Kamailio - The Open Source SIP Server,» [Página web]. URL: https://www.kamailio.org/w/. [Último acceso: 31 05 2017].

[23] «GNS3 Home,» [Página web]. URL: https://www.gns3.com/. [Último acceso: 17 05 2017].

[24] «Cisco Networking Academy,» [Página web]. URL: https://www.netacad.com/es/about-networking-academy/packet-tracer/. [Último acceso: 17 05 2017].

[25] «NetSim,» [Página web]. URL: http://tetcos.com/. [Último acceso: 17 05 2017].

[26] «Parallels Home,» [Página web]. URL: http://www.parallels.com/es/. [Último acceso: 17 05 2017].

[27] «QUEMU Home,» [Página web]. URL: http://wiki.qemu.org/Main_Page. [Último acceso: 05 17 2017].

[28] «Microsoft Windows Virtual PC,» [Página web]. URL: https://www.microsoft.com/es-es/download/details.aspx?id=3702. [Último acceso: 17 05 2017].

[29] «LINKSYS,» [Página web]. URL: http://www.linksys.com/cr/support-product?pid=01t80000003KXPxAAO. [Último acceso: 17 05 2017].

[30] V. M. G. Pozuelo, «Tipos de conexiones de red en sofware de virtualización.,» [Artículo web]. URL: http://www.ticarte.com/sites/su/users/7/file/tipos-de-redes-en-virtualbox-y-vmware.pdf. [Último acceso: 17 05 2017].

[31] «Open Source IMS Core: Documentación / Guía de Instalación.,» [En línea]. Available: http://www.openimscore.org/documentation/guia-de-instalacion/. [Último acceso: 17 05 2017].

[32] H. Nemati, «Personal Website os Hani Nemati. FOkU's Monster client on Ubuntu 12.04,» [Página web]. URL: https://sites.google.com/site/haninemati/developing-ims/fokus-monster-client. [Último acceso: 17 05 2017].

[33] «SIPp Home,» [Página web]. URL: http://sipp.sourceforge.net/ [Último acceso: 17 05 2017].

[34] CNMC, «Informe Económico Sectorial de las Telecomunicaciones y el Audiovisual,» 2015. URL: https://data.cnmc.es/datagraph/files/Informe_Economico_Telecomunicaciones_CNMC.pdf [Último acceso: 02 06 2017].

ANEXO A: Configuración detallada de los distintos elementos del entorno.

I. Router

Los pasos utilizados en la configuración del router son:

- Se accede al router a través de Telnet que se trata de un protocolo de red que nos permite acceder a otra máquina para manejarla remotamente. Para lo cual se ejecuta en el terminal los siguientes comandos:
  >> telnet 172.20.2.1

- Un vez que se ha accedido a la configuración del router se pasa a modificar el archivo */etc/config/dhcp* de forma que se le indica al router que anuncie por DHCP un servidor de DNS, cambiando la línea *list dhcp_option* por la dirección IP de la máquina virtual IMS. Este paso se ha de realizar para cada subred, tal y como se muestra en la Ilustración 49.

```
config dhcp lan
        option interface        lan
        option start            50
        option limit            150
        option leasetime        12h
        list dhcp_option        '6,172.20.2.55'

config dhcp wan
        option interface        wan
        option start            50
        option limit            150
        option leasetime        12h
        list dhcp_option        '6,172.20.2.55'

config dhcp wifin

config dhcp wifin
        option interface        wifin
        option start            50
        option limit            150
        list dhcp_option        '6,172.20.2.55'
```

Ilustración 49. Configuración del router para las subredes.

- En este mismo archivo se asigna una dirección IP fija a cada dirección MAC de los dispositivos conectados a la red, tal y como se muestra en la Ilustración 50.

```
config host
        option name     IMSCore
        option mac      00:0c:29:73:ab:22
        option ip       172.20.2.55


config host
        option name     VCore
        option mac      08:00:27:10:43:0f
        option ip       172.20.2.178

config host
        option name     Host
        option mac      18:4F:32:99:44:D9
        option ip       172.20.2.175
```

Ilustración 50. Configuración router para los diferentes dispositivos.

- Así mismo, la configuración de las interfaces (en particular la de la inalámbrica que es la que se usa) se encuentra detallada de los archivos /etc/config/wireless y /etc/config/network (se muestran en la Ilustración 51 y la Ilustración 52 respectivamente).

```
#### VLAN configuration
config wifi-device  wl0
        option type     broadcom
        option channel  11
        option hwmode   11bg
        option txpower  10
        option disabled 0

config wifi-iface
        option device   wl0
        option network  wifin
        option mode     ap
        option ssid     core-testbed
        option encryption none
```

Ilustración 51. Archivo /etc/config/wireless del router.

```
#### VLAN configuration
config switch eth0
        option enable   1

config switch_vlan eth0_0
        option device   "eth0"
        option vlan     0
        option ports    "0 1 2 3 5"

config switch_vlan eth0_1
        option device   "eth0"
        option vlan     1
        option ports    "4 5"

#### Loopback configuration
config interface loopback
        option ifname   "lo"
        option proto    static
        option ipaddr   127.0.0.1
        option netmask  255.0.0.0


#### LAN configuration
config interface lan
        option type     bridge
        option ifname   "eth0.0"
        option proto    static
        option ipaddr   172.20.0.1
        option netmask  255.255.255.0


#### WAN configuration
config interface        wan
        option ifname   "eth0.1"
        option proto    static
        option ipaddr   172.20.1.1
        option netmask  255.255.255.0

config interface wifin
        option ifname   "wl0"
        option proto    static
        option ipaddr   172.20.2.1
        option netmask  255.255.255.0
```

Ilustración 52. Archivo /etc/config/network del router

II. Máquina virtual donde se encuentra el Open IMS core

La máquina virtual con el OpenIMS core se ha de configurar para responder a las peticiones que reciban los servidores IMS y el DNS.

Para configurar las llamadas a los servidores IMS se cambia la dirección que viene configurada por defecto (127.0.0.1) por la dirección IP asignada a la máquina virtual, en nuestro caso 172.20.2.55, en la siguiente lista de ficheros:

/opt/OpenIMSCore/etc/hss/hss.properties
/opt/OpenIMSCore/etc/hss/DiameterPeerHSS.xml
/opt/OpenIMSCore/etc/pcscf.cfg
/opt/OpenIMSCore/etc/icscf.cfg

/opt/OpenIMSCore/etc/scscf.cfg
/opt/OpenIMSCore/etc/icscf.xml
/opt/OpenIMSCore/etc/scscf.xml

Se procede de la misma forma, cambiando las IP del siguiente fichero, para configurar el DNS de la máquina virtual que responderá a las peticiones para resolver nombres de los servidores IMS:

/etc/bind/db.open-ims.test

```
$ORIGIN open-ims.test.
$TTL 1W
@                       1D IN SOA       localhost. root.localhost. (
                                        2006101001      ; serial
                                        3H              ; refresh
                                        15M             ; retry
                                        1W              ; expiry
                                        1D )            ; minimum

                        1D IN NS        ns
ns                      1D IN A         172.20.2.55

pcscf                   1D IN A         172.20.2.55

open-ims.test.          1D IN A         172.20.2.55
icscf                   1D IN A         172.20.2.55

_sip                    1D SRV 0 0 5060 icscf
_sip._udp               1D SRV 0 0 5060 icscf
_sip._tcp               1D SRV 0 0 5060 icscf

open-ims.test.          1D IN NAPTR 10 50 "s" "SIP+D2U" ""      _sip._udp
open-ims.test.          1D IN NAPTR 20 50 "s" "SIP+D2T" ""      _sip._tcp


scscf                   1D IN A         172.20.2.55

hss                     1D IN A         172.20.2.55

ue                      1D IN A         172.20.2.55

presence                1D IN A         172.20.2.55

opense                  1D IN A         172.20.2.55

sipsee                  1D IN A         172.20.2.55

anubis                  1D IN A         172.20.2.55

xdms                    1D IN A         172.20.2.55
xmldb                   1D IN A         172.20.2.55

openpe                  1D IN A         172.20.2.55
interceptor             1D IN A         172.20.2.55
```

Ilustración 53. Fichero/etc/bind/db.open-ims.test

Además, en el archivo /etc/bind/named.conf.options se ha de cambiar la línea “listen-on” para indicar al DNS que escuche a todos (any). Esto permitirá que el DNS sea consultado desde fuera de la máquina virtual, cosa que harán los clientes IMS. Queda entonces el archivo con la siguiente configuración:

```
options {
        directory "/var/cache/bind";

        // If there is a firewall between you and nameservers you want
        // to talk to, you may need to fix the firewall to allow multiple
        // ports to talk.  See http://www.kb.cert.org/vuls/id/800113

        // If your ISP provided one or more IP addresses for stable
        // nameservers, you probably want to use them as forwarders.
        // Uncomment the following block, and insert the addresses replacing
        // the all-0's placeholder.

         forwarders {
                10.147.9.3;
                195.37.77.164;
        };

        auth-nxdomain no;    # conform to RFC1035
        #listen-on { 127.0.0.1; };
        listen-on { any; };

        listen-on-v6 { ::; };
};
```

Ilustración 54. Configuración fichero named.conf.options

A continuación, se debe reiniciar los dos servidores, el DNS y todo los de IMS:

/etc/init.d/bind9 restart
/opt/OpenIMSCore/bin/kill.sh
/opt/OpenIMSCore/bin/start.sh

Por último se comprueba que el archivo /etc/resolv.conf haya sido configurado por el DHCP (del router) correctamente para indicar el servidor de DNS a usar por la máquina. El archivo tiene que tener la configuración indicada en la Ilustración 55Ilustración 55.

```
# Generated by NetworkManager
domain open-ims.test
search open-ims.test
nameserver 172.20.2.55
```

Ilustración 55. Configuración archivo resolv.conf

III. Máquina Virtual donde se aloja CORE

En la configuración de CORE también es necesario comprobar si la configuración del archivo /etc/resolv.conf es correcta. El archivo tiene que tener la misma configuración que en IMS, tal y como se ha mostrado en la Ilustración 55 ya que el servidor DNS alojado en la máquina virtual de IMS también sirve como servidor DNS a CORE.

IV. Escenario emulado en CORE

Se configura el escenario para habilitar la comunicación entre los nodos emulados dentro del CORE (Ilustración 15) con equipos fuera de la emulación. Los pasos a seguir para crear un RJ45 Tool y enlazarlo a una dirección IP por la que se accede al escenario emulado en CORE son:

- Crear un interfaz dummy en la máquina virtual donde se aloja CORE, de la siguiente forma:
  >> sudo modprobe dummy numdummies=1

- Crear un RJ45 conectado a un router en el escenario a simular y asociarle la interfaz dummy0.

- Ejecutar la emulación y configurar una dirección IP de la siguiente forma:
  >> sudo brtl show
  (Para mostrar todos los bridges del escenario, siendo el que interesa el asignado a la interfaz dummy0).
  >> sudo ip addr add 10.0.0.2/24 dev b.XXXXX.XXXXX
  (Siendo 10.0.0.2/24 la interfaz asignada al RJ45 y b.XXXXX.XXXXX, el identificador del puente de la dummy creada).

Tras realizar los pasos anteriores, donde se creaba una dummy y se la asignaba una IP, se procede a la configuración de la tabla de rutas, de manera que se indica que las direcciones del escenario emulado están accesibles vía un siguiente salto a la interfaz del router al que está conectado el RJ45. Los pasos a seguir son los siguientes:

- En la máquina virtual donde se aloja CORE, cuya dirección IP es 172.20.2.178.
  >> sudo ip route add 10.0.0.0/16 via 10.0.0.1
  (Indica que las direcciones del escenario CORE están accesibles vía un siguiente salto, 10.0.0.1, que es la interfaz del router a la que se conecta el RJ45).
  >> sudo sysctl –w net.ipv4_forward=1
  (Para activar el forwarding en la máquina virtual donde se aloja CORE).

- En el nodo n2 del escenario emulado, correspondiente con el router, se añade también un siguiente salto:
  >> sudo ip route add 172.20.2.0/24 via 10.0.0.2
  (Indica que todas las direcciones de la red local donde se alojan las máquinas virtuales están accesibles a través del siguiente salto 10.0.0.2).
  Si se muestra la tabla de rutas del router (con el comando *route –n*), el resultado obtenido es el que se muestra en la Ilustración 56.

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        0.0.0.0         255.255.255.0   U     0      0        0 eth0
10.0.1.0        0.0.0.0         255.255.255.0   U     0      0        0 eth1
10.0.2.0        0.0.0.0         255.255.255.0   U     0      0        0 eth2
172.20.2.0      10.0.0.2        255.255.255.0   UG    0      0        0 eth0
```

Ilustración 56. Tabla rutas del router de la red emulada.

También se tienen que añadir entradas a las tablas de rutas de la máquina virtual donde se aloja IMS y a la máquina host. En la máquina IMS se realiza de forma similar a los pasos anteriores:

```
>> sudo ip route add 10.0.0.0/16 via 172.20.2.178
```

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.20.2.0      0.0.0.0         255.255.255.0   U     1      0        0 eth5
10.0.0.0        172.20.2.178    255.255.0.0     UG    0      0        0 eth5
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 eth5
0.0.0.0         172.20.2.1      0.0.0.0         UG    0      0        0 eth5
```

Ilustración 57. Tabla de rutas de la máquina donde se aloja IMS

En la máquina host física, al ser un sistema Windows, se realiza de forma diferente. Se tiene que acceder a Símbolo de Sistema como administradores e introducir el siguiente comando:

```
>> route ADD 10.0.0.0 MASK 255.255.0.0 172.20.2.178
```

V. Instalación de clientes IMS

i. Interfaz gráfica de usuario

Para la instalación del Monster se necesita instalar una serie de paquetes y compilar el ejecutable que se ha descargado. Para instalar los paquetes se ejecutan las siguientes líneas en el terminal:

```
>>apt-get install libexosip2-dev
>>apt-get install libgtk2.0-dev
>>apt-get install libxml2-dev
>>apt-get install libcurl4-openssl-dev
>>apt-get install libgstreamer0.10-0
>>apt-get install libgstreamer-plugins-base0.10-dev
>>apt-get install gstreamer0.10-plugins-base
>>apt-get install gstreamer0.10-plugins-good
>>apt-get install gstreamer0.10-plugins-bad
>>apt-get install gstreamer0.10-plugins-ugly
>>apt-get install gstreamer0.10-ffmpeg
```

```
>>apt-get install libavcodec-extra-53
>>apt-get install libavcodec-unstripped-51
>>apt-get install libvlc-dev
>>apt-get install vlc
```

Para arrancar el cliente a través de SSH se ejecuta la siguiente línea de comandos en un terminal de la máquina virtual donde se ejecuta el CORE:

```
>> ssh –X 10.0.2.20 /home/core/Desktop/mosnter-0.9.25/monster
```

Donde *10.0.2.10* es el nodo CORE donde se quiere que se lance el Monster y */home/core/Desktop/mosnter-0.9.25/monster* es la ruta donde está el ejecutable. Con la opción –X se redirecciona la salida gráfica al servidor de ventanas X donde se ejecuta el SSH.

ii. Scripts SIPp

Los pasos a seguir para la instalación de SIPp son muy sencillos. En primer lugar hay que descargarse el paquete y proceder a su extracción. Para lo cual se ejecuta en el terminal, en el directorio donde se encuentra la descarga:

```
>> tar -xvzf sipp-xxx.tar.gz
```

En segundo lugar, se accede a la carpeta donde se ha descomprimido SIPp y se compila de la siguiente forma, (se muestra la forma de compilarlo con todas las opciones aunque solo se necesita --with-openssl):

```
>> ./configure --with-sctp --with-pcap --with-openssl
>> make
```

Por último se debe cambiar el emulador de terminal de los nodos de la red emulada en CORE al “xterm”. Para ello se comprueba en primer lugar cual es el emulador funcionando con al siguiente línea de comandos:

```
>> echo $TERM
```

Si no es el que se necesita se cambia de la siguiente forma:

```
>> export TERM=xterm
```

ANEXO B: Scripts de SIPp

A continuación se muestran los scripts de SIPp [33] utilizados en el proyecto (que están basado en los ejemplos distribuidos con la herramienta SIPp). La primera línea de cada fichero, marcada en azul, es la línea que se utilizada para llamar a los scripts.

I. Fichero register

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">

<!--
 sipp -sf register.xml open-ims.test -rsa 172.20.2.55:4060 -auth_uri open-ims.test -i 10.0.1.2 -p
5062 -m 1 -inf users.csv
-->
<scenario name="UAC Registering in IMS Core">

    <!—Se envía el mensaje REGISTER de la siguiente forma-->
  <send retrans="500">
   <![CDATA[

     REGISTER sip:open-ims.test SIP/2.0
     Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
    <!—El parámetro [field0] corresponde al primer parámetro de los separados con punto y
coma del fichero users.csv-->
     From: <sip:[field0]@open-ims.test>
     To: <sip:[field0]@open-ims.test>
     Call-ID: [call_id]
     CSeq: 1 REGISTER
     Max-Forwards: 20
     Contact: <sip:[field0]@[local_ip]:[local_port]>;transport=[transport]
     Authorization: Digest username="[field1]@open-ims.test",realm="open-
ims.test",uri="sip:open-ims.test",nonce="",response="",algorithm=MD5
     Expires: 28800
     Content-Length: 0

    ]]>
  </send>

<!— Espera recibir un mensaje 401 Unathorized-->
  <recv response="401" auth="true" >
  </recv>

<!— Se vuelve a enviar el mensaje REGISTER-->
  <send retrans="500">
   <![CDATA[

     REGISTER sip:open-ims.test SIP/2.0
```

```
    Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
    From: <sip:[field0]@open-ims.test>
    To: <sip:[field0]@open-ims.test>
    Call-ID: [call_id]
    CSeq: 2 REGISTER
    Max-Forwards: 20
    Contact: <sip:[field0]@[local_ip]:[local_port]>;transport=[transport]
<!— Field2 es el tercer campo de los separados por punto y coma del fichero csv, e incluye la información necesaria para hacer la autentificación del usuario -->
    [field2]
    Expires: 28800
    Content-Length: 0

   ]]>
 </send>

<!— Espera recibir un mensaje 200-->
 <recv response="200" >
 </recv>

<!— Espera 2000 milisegundos y termina el escenario-->
 <pause milliseconds="2000" />

</scenario>
```

II. Fichero unregister

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">

<!--
 sipp -sf unregister.xml open-ims.test -rsa 172.20.2.55:4060 -auth_uri open-ims.test -i 10.0.1.2 -p 5062 -m 1 -inf users.csv
-->
<scenario name="UAC Registering in IMS Core">

<!— Se envía un mensaje REGISTER-->
 <send retrans="500">
  <![CDATA[

    REGISTER sip:open-ims.test SIP/2.0
    Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
    From: <sip:[field0]@open-ims.test>
    To: <sip:[field0]@open-ims.test>
    Call-ID: [call_id]
    CSeq: 1 REGISTER
    Max-Forwards: 20
    Contact: <sip:[field0]@[local_ip]:[local_port]>;transport=[transport]
    Authorization: Digest username="[field1]@open-ims.test",realm="open-ims.test",uri="sip:open-ims.test",nonce="",response="",algorithm=MD5
```

```
    Expires: 0
    Content-Length: 0

  ]]>
 </send>

<!— Espera recibir un mensaje 401-->
 <recv response="401" auth="true" >
 </recv>

<!— Se envía un mensaje REGISTER-->
 <send retrans="500">
  <![CDATA[

    REGISTER sip:open-ims.test SIP/2.0
    Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
    From: <sip:[field0]@open-ims.test>
    To: <sip:[field0]@open-ims.test>
    Call-ID: [call_id]
    CSeq: 2 REGISTER
    Max-Forwards: 20
    Contact: <sip:[field0]@[local_ip]:[local_port]>;transport=[transport]
    [field2]
<!— Aquí está la diferencia con el script de registro. En el caso de querer desregistrar  aun
usuario se pone el tiempo de duración del registro a 0-->
    Expires: 0
    Content-Length: 0

  ]]>
 </send>

<!— Espera recibir un mensaje 200 OK-->
 <recv response="200">
 </recv>

<!— Espera 2000 milisegundos y termina el escenario-->
 <pause milliseconds="2000" />

</scenario>
```

III. Fichero uas

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">

<!--
 sipp -sf uas.xml -i 10.0.2.20 -p 5063 -m 1 -inf callee.csv
-->
<scenario name="Basic UAS">
```

```
 <!—Se guardan las rutas -->
 <recv request="INVITE" crlf="true" rrs="true">
  <action>
   <!—Se coge sesión Id y version -->
   <ereg regexp="o=([[:alnum:]]*) ([[:alnum:]]*) ([[:alnum:]]*)" search_in="body"
assign_to="1,2,8,9"/>
   <!-- Hacemos un log para las variables que no usamos -->
   <log message="[$1][$2]"/>
  </action>
 </recv>

<!— Se envía un mensaje 180 Ringing-->
 <send>
  <![CDATA[

   SIP/2.0 180 Ringing
   [last_Via:]
   [last_From:]
   [last_To:];tag=[call_number]
   [last_Call-ID:]
   [last_CSeq:]
   [last_Record-Route:]
   Contact: <sip:[field0]@[local_ip]:[local_port];transport=[transport]>
   Content-Length: 0

  ]]>
 </send>

<!— Se espera un tiempo y se envía un mensaje 200 OK-->
 <pause milliseconds="3000" />
 <send>
  <![CDATA[

   SIP/2.0 200 OK
   [last_Via:]
   [last_From:]
   [last_To:];tag=[call_number]
   [last_Call-ID:]
   [last_CSeq:]
   [last_Record-Route:]
   Contact: <sip:[field0]@[local_ip]:[local_port];transport=[transport]>
   Content-Type: application/sdp
   Content-Length: [len]

<!— Carga SDP que hemos aceptado en la comunicación y define cómo van a ser los datos en
el plano de usuario. Aquí se usa el session ip y version number que se han cogido de la carga
SDP recibida en el INVITE-->
   v=0
   o=[field0] [$8] [$9] IN IP[local_ip_type] [local_ip]
   s=-
   c=IN IP[media_ip_type] [media_ip]
   t=0 0
```

```
    m=audio [media_port] RTP/AVP 0
    a=rtpmap:0 PCMU/8000

   ]]>
  </send>

<!— Se espera recibir un ACK-->
  <recv request="ACK"
       crlf="true">
  </recv>

<!— Para ser consistentes con el uac, indicamos que estamos en sesión.-->
  <nop>
    <action>
      <exec command="echo 'Sesión'" />
    </action>
  </nop>


<!— Cuando recibimos un mensaje BYE se envía un 200 OK-->
  <recv request="BYE">
  </recv>

  <send>
    <![CDATA[

      SIP/2.0 200 OK
      [last_Via:]
      [last_From:]
      [last_To:]
      [last_Call-ID:]
      [last_CSeq:]
      Contact: <sip:[field0]@[local_ip]:[local_port];transport=[transport]>
      Content-Length: 0

    ]]>
  </send>

</scenario>
```

IV. Fichero uac

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">

<!--
 sipp -sf uac.xml open-ims.test -s bob -rsa 172.20.2.55:4060 -i 10.0.1.2 -p 5062 -m 1 -inf
caller.csv
-->
<scenario name="Basic UAC">
```

```
<!—Se envía un INVITE correctamente -->
<send>
 <![CDATA[

   INVITE sip:[service]@open-ims.test SIP/2.0
   Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
   From: [field0] <sip:[field0]@open-ims.test>;tag=[call_number]
   To: [service] <sip:[service]@open-ims.test>
   Call-ID: [call_id]
   CSeq: 1 INVITE
   Contact: <sip:[field0]@[local_ip]:[local_port];transport=[transport]>
   Max-Forwards: 70
   Subject: IMS Call
   Route: <sip:orig@scscf.open-ims.test:6060;lr>
   Content-Type: application/sdp
   Content-Length: [len]

   v=0
   o=[field0] 53655765[pid][call_number] 2353687637 IN IP[local_ip_type] [local_ip]
   s=-
   c=IN IP[media_ip_type] [media_ip]
   t=0 0
   m=audio [auto_media_port+1] RTP/AVP 0
   a=rtpmap:0 PCMU/8000
   m=video [auto_media_port+3] RTP/AVP 96
   a=rtpmap:96 H263-1998/90000

 ]]>
</send>

<!—Se espera revivir una respuesta 100 -->
<recv response="100"
    optional="true">
</recv>

<!—Se espera revivir un mensaje 180 RINGING -->
<recv response="180" optional="true">
</recv>

<!—Se guardan las rutas -->
<recv response="200" rrs="true">
</recv>

<!—Se envía un mensaje ACK -->
<send>
 <![CDATA[

   ACK [next_url] SIP/2.0
   Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
   From: [field0] <sip:[field0]@open-ims.test>;tag=[call_number]
   To: [service] <sip:[service]@open-ims.test>[peer_tag_param]
```

```
    Call-ID: [call_id]
    CSeq: 1 ACK
    Contact: <sip:[field0]@[local_ip]:[local_port];transport=[transport]>
    Max-Forwards: 70
    Subject: IMS Call
    [routes]
    Content-Length: 0

  ]]>
 </send>

 <nop>
  <action>
   <exec command="echo 'Sesión'" />
  </action>
 </nop>

 <!—Se espera 15000 milisegundos -->
 <pause milliseconds="15000"/>

 <!—Se envía un mensaje BYE par finalizar al llamada-->
 <!-- <send retrans="500"> -->
 <send>
  <![CDATA[

    BYE [next_url] SIP/2.0
    Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
    From: [field0] <sip:[field0]@open-ims.test>;tag=[call_number]
    To: [service] <sip:[service]@open-ims.test>[peer_tag_param]
    Call-ID: [call_id]
    CSeq: 2 BYE
    Contact: <sip:[field0]@[local_ip]:[local_port];transport=[transport]>
    Max-Forwards: 70
    Subject: Performance Test
    [routes]
    Content-Length: 0

  ]]>
 </send>

 <!—Se espera recibir un mensaje 200 OK -->
 <recv response="200" crlf="true">
 </recv>

 <!—Se hace una pausa y se termina la llamada -->
 <pause milliseconds="2000" />

</scenario>
```

V. Fichero users.csv

SEQUENTIAL;;
#Usuarios a registrar en core IMS;;
alice;alice;[authentication username=alice@open-ims.test password=alice]

VI. Fichero caller

SEQUENTIAL
alice

VII. Fichero callee

SEQUENTIAL
bob

ANEXO C: Normativa y marco regulador

I. Marco regulador de las telecomunicaciones en España

El maco regulador de las telecomunicaciones es España está controlado por la Comisión del Mercado de las Telecomunicaciones (CMT). Este organismo público se encarga de garantizar el correcto funcionamiento de los mercados de comunicaciones a través del establecimiento y la supervisión del cumplimiento de las obligaciones de los operadores y del fomento de la competencia en los mercados de servicios audiovisuales, además de intervenir en la resolución de conflictos entre dichas operadoras. Hoy en día, dicha comisión está integrada dentro de la Comisión Nacional de Los Mercados y la Competencia (CNMC).

La Agenda Digital para España, marca los pasos a seguir en las Tecnologías de la Información y las Comunicaciones (TIC) para el cumplimiento de la Agenda Digital Europea en 2020. Dentro de esta agenda se encuentra el Plan de telecomunicaciones y redes ultrarrápidas. Este plan contempla medidas para potenciar el despliegue de redes, renovando las redes ya existentes y abaratando los costes de la extensión de las mismas. Es decir, se pretende financiar la implantación de nuevas tecnologías (como las redes IMS) para favorecer el desarrollo tecnológico del país. Estas pautas también se recogen en la Ley General de Telecomunicaciones, dónde además se menciona la importancia del estudio y desarrollo de estas nueva tecnologías, que se han de impulsar con inversión en proyectos de I+D.

II. Legislación aplicable sobre las tecnologías empleadas

En cuanto a los diferentes softwares utilizados en el proyecto, la regulación gira en torno a los conceptos de software libre y software propietario. El software propietario se basa, principalmente, en que el acceso a su código fuente no es libre, es decir, se encuentra en disposición de su desarrollador, quien es el que decide quién puede disponer de él y con qué fin. En el otro extremo se encuentra el software libre, el cual puede ser copiado, modificado y utilizado libremente con cualquier fin. Normalmente el software libre es concebido como gratuito, debido a su término inglés "*free software*", pero es una concepción errónea, ya que este término se refiere a su carácter libre, no gratuito. De hecho existen multitud de empresas basadas en la comercialización de software libre, de manera que obtienen ingresos por su desarrollo y comercialización.

Algunas de las características que presenta el **software libre** son:

- Libertad de copia, modificación y estudio.
- Libertad de uso con cualquier fin.
- Libertad de distribución.
- Mayor seguridad y fiabilidad.
- Existencia de aplicaciones para varios sistemas operativos.
- Menor precio.
- Menor compatibilidad con el hardware.

- Inexistencia de garantías por parte del autor.

Las características que presenta el **software propietario** son:

- Facilidad de adquisición.
- Mayor compatibilidad con el hardware.
- Dependencia del autor en cuanto al uso.
- Imposibilidad de modificación y distribución.
- Mayor coste.

La ley española de propiedad intelectual es la encargada de regular la protección de los derechos del autor de los creadores de programas de ordenador. La protección incluye al programa en sí, su documentación y las obras derivadas del mismo. Además contempla los derechos de explotación, según los cuales la ley prohíbe la reproducción, adaptación o cualquier transformación y la distribución del programa de ordenador a todo aquél que no sea el titular de estos derechos.

Los softwares libres también cuentan con medidas de protección, como son la gran variedad de licencias existentes. Las licencias regulan la distribución, la creación y la copia de los programas de ordenador, haciendo que se cumplan las libertades que caracterizan al software libre.

Entre las licencias con las que se ha trabajado en este proyecto están la licencia GPL (utilizada por VirtualBox) y en la GNU GPL (utilizada por OpenIMS), siento esta licencia una de las más restrictivas dentro del grupo GPL. En dicha licencia el autor del programa conserva intactos sus derechos de copyright, pero sí se permite la distribución del programa, bajo la condición de que todas las subversiones se encuentran dentro de los términos de la licencia. Otra de las licencias mencionadas es la licencia BSD (utilizada por CORE) que tiene unas características similares a la GNU GPL, es decir, el autor mantiene el copyright de manera que se atribuye la autoría de todos los trabajos derivados del programa, pero se permite la libre distribución del mismo.

ANEXO D: Extended Abstract

I. Introduction

The convergence to an "all-IP" world has led to the development of new technologies that have transformed the telecommunication market. The IMS (IP Multimedia Subsystem) architecture is an important example. This architecture incorporates services demanded by users such as voice, video and multimedia traffic, all for packet switched networks based on the IP protocol. IMS implements a protocol, Session Initiation Protocol (SIP) that allows signalling and management of multimedia sessions.

To develop multimedia applications it is necessary to validate their behaviour. Network emulators are a useful tool which allow using virtual machines, emulate a network and use the same software that is used in the real network. In addition, they have functionalities like interconnection of virtual networks to the real network. CORE (Common Open Research Emulator) is an example of a very versatile network emulator with many functionalities.

In this project the CORE network emulator has been equipped with an IMS system to be able to offer audio-visual services based on SIP signalling. This work has required the knowledge and management of different network technologies, and of virtual machines.

The testing of the platform have been done with two different IMS clients, one that has a graphical user interface (Monster) and another through SIPp scripts. In addition, the traffic of SIP messages has been captured to check the operation of the system, concluding that the objectives have been achieved, is possible to develop a platform to emulate any type of network and equip it with an IMS system with SIP signalling. This platform is adaptable to any network under study for the development and implementation of multimedia services over IP networks.

II. State of art

i. IMS and SIP

IMS is an architecture that allows the integration of multimedia services over IP technology. It provides functionalities such as signalling for multimedia services, user authentication and access control services. In addition you can apply QoS (Quality of Service). It is based on the use of standards defined by the IETF (Internet Engineering Task Force) as SIP and DIAMETER.

The Session Initiation Protocol (SIP) is currently the most popular signalling protocol for establishing IP communications that need multimedia services. It is used in the most common applications of VoIP (Voice over IP), but the most interesting use it is the implementation in 4G networks, since these do not have circuit switching, so the telephone services in a 4G network, without combined with other networks such as 3G, are offered by VoLTE (Voice over LTE) using

IMS and SIP. SIP works in conjunction with the RTP protocol transporting audio and video communications.

SDP (Session Description Protocol) is an integrated protocol in SIP, which defines a format to describe the initiation parameters of a multimedia traffic flow, for example, it allows to describe the medium (video, audio, etc.), the transport protocol and the format used in the encoding. Through this protocol each end of the session describes its capabilities. This information also allows you to negotiate the QoS for the data plane, if necessary.

I. IMS architecture.

IMS is an architecture of "Layers" formed by:

- **The access layer**: It supports any type of high-speed access, whether mobile access, fixed broadband, cable networks, WIFI, etc. IMS supports the conversion of protocols in a way that allows the intercommunication with other networks, in particular those based on circuit switching.

- **The transport layer:** The transport layer is composed of the routers and refers to the entire IP routing of the packets that circulate through the infrastructure. In this layer, quality of service (QoS) mechanisms can be integrated. The terminal negotiates its capabilities and expresses its requirements during the session establishment phase, through control signalling (SIP and SDP).

- **The control layer:** The control layer is the center of IMS. It consists of session controllers responsible for the routing of signalling between users and the invocation of services. These nodes are called CSCF (Call State Control Function).

- **The Application layer:** This layer introduces Application Server and MRF (Multimedia Resource Function) application servers, better known as IP Media Servers (IMS). Application servers provide the logic of services implemented in IMS.

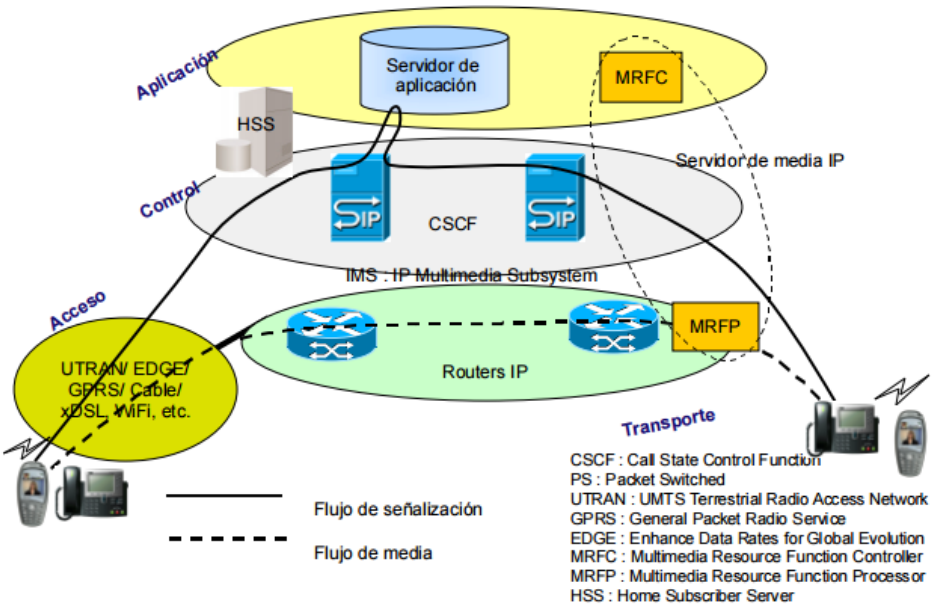


Illustration 58. Network Architecture and IMS Services (taken from [4])

In the Illustration 59 you can see the elements of the IMS architecture, some of which are studied below.

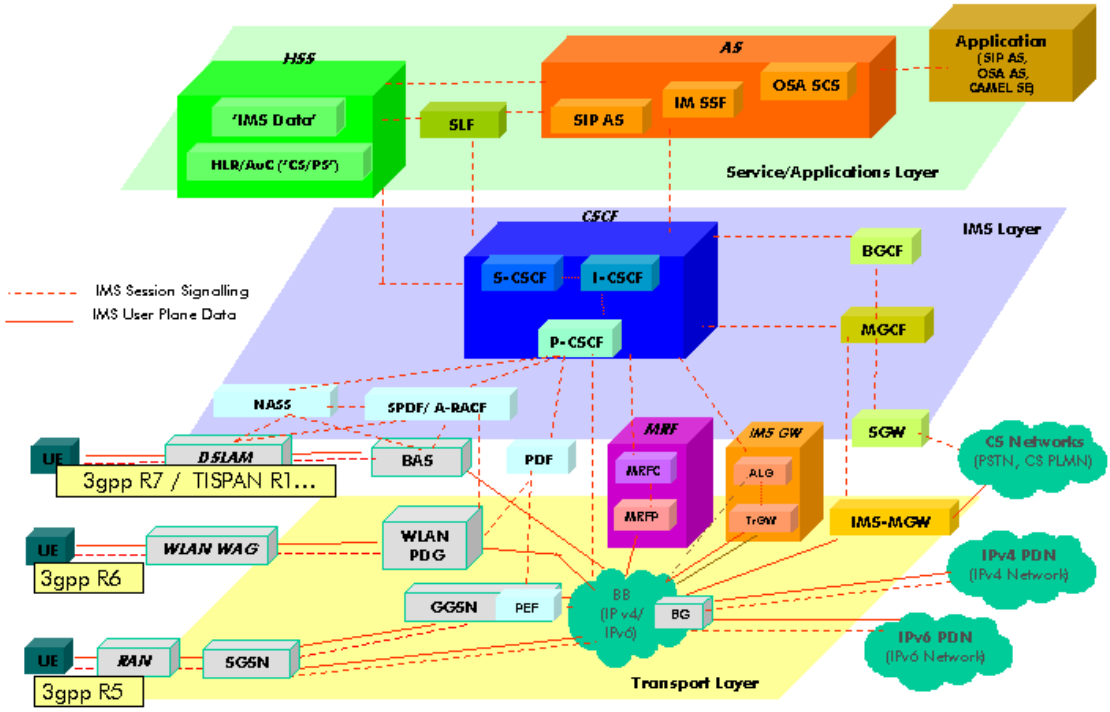


Illustration 59 the elements of the IMS architecture (taken from [4])

- **CSCF or Call Session Control Function:** It is the main element within the IMS network, the key piece for signalling through the SIP protocol to establish, modify and terminate a multimedia session. It can be thought of as a "SIP Server". This device performs three functions:
    - P-CSCF (Proxy - Call Session Control Function).
    - I-CSCF (Interrogating -Call Session Control Function).
    - S-CSCF (Serving-Call Session Control Function).

- **HSS (Home Subscriber Server):** It is where the database and profiles of all IMS users are located. It is responsible for the authentication and registration of users, as well as the different types of sessions that they can establish, regardless of the access technology.

- **MGW (Media Gateway), MGC (Media Gateway Controller) and SGW (Signalling Gateway Function):** They are nodes to allow communication from the IMS-based network and the circuit-switched telephone network: MGW in the data plane; and MGC together with SGW in the control plane.

- **AS (Application Servers)**: Application servers provide the logic of services implemented in IMS. Generally within the network there are multiple application servers, since each one usually implements a single service. There are three types of AS:
    - SIP-AS (SIP - Application Server).
    - OSA-SCS (Open Service Access – Service Capability Server).
    - IM-SSF (IP Multimedia – Service Switching Function).

II. SIP architecture.

The basic entity of SIP is the UA (User Agent). An UA is an end user device that is used to establish and free multimedia sessions with other UAs. They are defined according to whether they are UAC (User Agent Client) or UAS (User Agent Server) servers, being able to act as one or the other independently. The UA is not known among them, they only know the situation of a SIP Proxy. Therefore, any UA that wants to start a session will have to communicate it to the SIP Proxy. In addition to forwarding the messages to their destination, the Proxy performs user authentication functions, so every UA user has to register with the Proxy in order to make a call. These functions are divided into the CSCFs.

Other entities that may participate in the exchange of SIP messages are the REGISTER server, which manages the users records in the network and the location of them, so that those users receive messages they will pass through their corresponding register, which correctly redirects them to the user and the REDIRECT server, which is responsible for redirecting client requests to other servers. In addition, it can intervene a B2BUA (UAS + UAC). The B2BUA is a SIP server that instead of forwarding SIP signalling, it receives SIP signalling (UAS server function) and in turn generates SIP signalling to the other side (UAC signalling function), that is, it generates two SIP dialogs.

The identification of users, services and nodes is done through Universal Resource Identifier (URIs). Each user has associated two identities to identify themselves, a public and a private identity. The public identity allows to route the SIP requests that are addressed to this user and is registered in the IMS network during the registration phase. It is used by other users to indicate who they want to communicate. The private identity is known by the network so that it keeps an association between the Private and Public Identity. The private identity is the only identifier by which the network knows the user, so that all functions of registration, authentication and access to user information are made through it.

There are two types of SIP messages: responses and requests. SIP response messages consist of a numeric code plus a textual component and SIP requests are also known as request messages. There are fourteen types, some of which are as follows:

- **INVITE**: This method allows us to establish a session between two UAs. During this request, the network routes the message from the sending UA to the receiver. The network uses the registration information of the users to know the address of the same. If the process has been done properly, the final UA responds to INVITE with a message TRYING 100 and later with a 180 RINGING.

- **BYE:** The BYE message allows us to end a session in progress. This method can be generated by both the UA that generated the session request and the target UA. The 200 OK message is answered if the session is terminated correctly.

- **REGISTER:** This method indicates the registration request on the network. In the header of the message, the data is sent for user authentication. If the registration was successful, a 200 OK message is answered.

- **CANCEL**: It is used to end a call that is in progress, but not one that is already established, since in this case it would end with the BYE method. If the registration was successful, a 200 OK message is answered.

- **ACK:** By means of this method it is possible to indicate that the request-response process has been satisfactorily resolved. It is generated to confirm that the response to the request has been processed correctly.

The format of the SIP message consists of three blocks, start line, header and message body.

- The **start line** is known as request-line, for the request messages, or as a status line, for the response messages. For example: *INVITE sip: alice@open-ims.test SIP / 2.0*, as a request line and *SIP / 2.0 200 OK*, as status line.

- The **header** has the following format: <header_name>: <header_value>. Where *header_name* indicates the name of the header and *header_value* the value of the header. There may be several headers with the same name but different value.

- The **body** of the message contains the information that the ends exchanged in the communication. Thanks to the information of the start line and the header, end-to-end communication is possible. In IMS SDP is used with a MIME format, which allows the body of the message to consist of several parts each with a different format.

ii. CORE

CORE (Common Operator Open Research) is a tool for network emulation in one or more virtual machines. It consists of a GUI (Graphical User Interface) of easy handling, where you can generate different topologies and configure the active equipment that form the network. Topologies can also be implemented using Python language.

It uses virtualization techniques from existing operating systems to create virtual networks, where protocol implementations and applications used in real systems can be run without modification. These emulated networks can be connected to real networks and systems, which makes the emulator suitable for platform testing, security studies, scenario evaluation and protocol research.

CORE can be downloaded from its official website. It is open source, under BSD license (Berkeley Software Distribution). It also offers the possibility of downloading the image of a virtual machine with vmdk format that already contains the emulator installed.

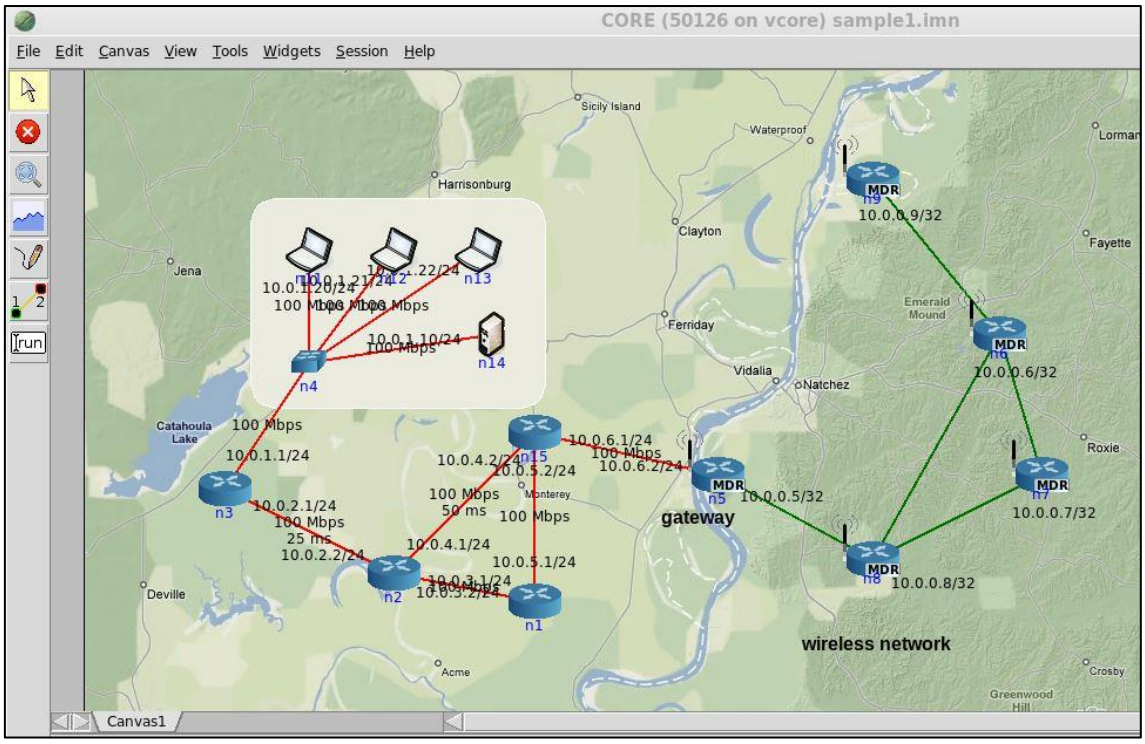


Illustration 60. Capture of CORE's GUI

iii. Virtual machines

A virtual machine is a software capable of emulating another operating system, as if it were a real computer. They are currently used to test other operating systems (other than the physical machine running the virtual machine), to use applications available on other operating systems, to run old programs (which can no longer run on the operating system of the physical machine), to test an application on different operating systems, etc.

In the realization of this project we use two virtual machines hosted in the host machine, each of which will be inside a different application: CORE virtual machine will be hosted in VirtualBox and the IMS machine will be hosted in VMWare.

III. Implementation and testing

The final objective of the project is to create a network emulated in CORE whose nodes can communicate with the elements of the IMS architecture to access audio-visual services. Therefore, you need to connect the IMS architecture servers to the emulated network within CORE. To do this we used a physical host computer with a Windows operating system and two virtual machines (both with Linux) are deployed on it: one where the IMS architecture servers are executed, the other where CORE is executed and where it is both emulated networks.

To configure all the machines (physical and virtual) of the scenario, a router has been used as DHCP server. In Illustration 61 we can see the complete scenario including the IP addresses.

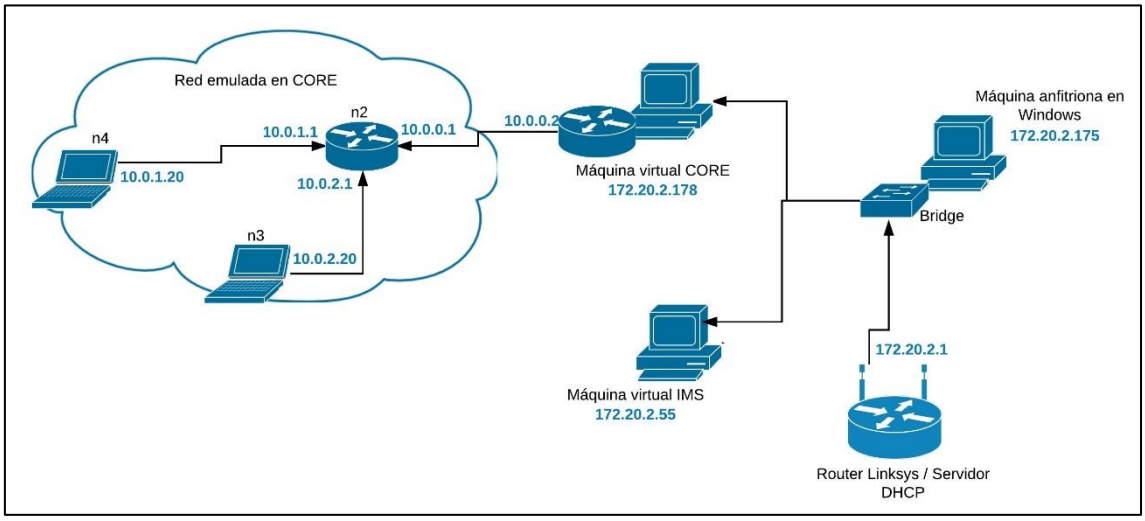


Illustration 61.Full scenario with IP addresses.

For the realization of the project we use a wireless router that works as DHCP server (Dynamic Host Configuration Protocol) assigning an IP address to each device connected to the network.

The IMS virtual machine hosts the DNS server that is used in all the machines in the scenario. In addition, within this virtual machine is housed a management system with a web

interface, through which user management is carried out. The goal of configuring the virtual machine where IMS is hosted is that it does not only serve local users (using the loopback address, that is, the virtual network interface), but also needs access to it from other virtual machines , specifically, from CORE.

In CORE we configure an emulated network scenario that has functionality to access audio-visual services. To achieve this purpose it is necessary for the emulated nodes to communicate with the exterior of the CORE machine to access the IMS servers. It will also be necessary to install IMS clients that can be executed from the emulated nodes. There are two options of IMS client:

- **Through a user interface**: This option consists of downloading an IMS Client, such as Monster. This client has a graphical user interface, from which you can interact with SIP, registering users, making calls between them, etc.

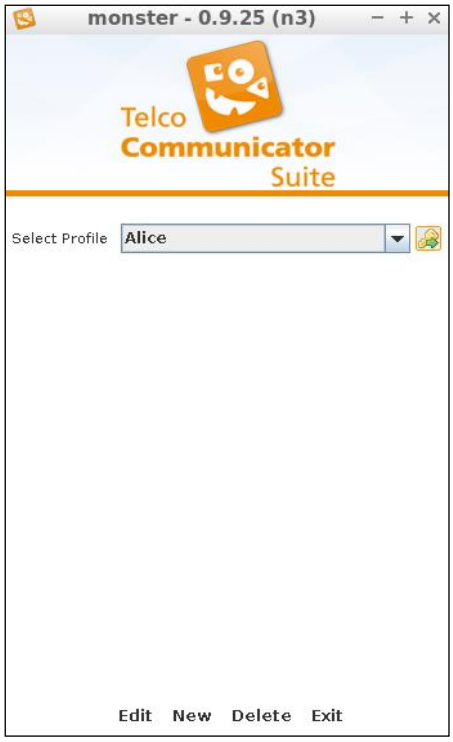


Illustration 62. Client IMS whit with graphical user interface

- **Through SIPp scripts**: In this option the first step is to download the SIPp source code, which is a tool that allows you to generate SIP traffic. This tool reads customizable files in which you can describe a call between users, or any type of SIP traffic. In addition, SIPp summarizes the call at the end of the call where you can see, among other things, a drawing with the messages exchanged.

The **tests** have been performed in two different ways, each corresponding to a method for executing the IMS client.. A registration and a call have been made with each IMS client and

it has been proven that both get the required results. In addition, all the SIP message traffic has been captured with Wireshark where the details of the message can be observed.

IV. Conclusions and future work

i. Conclusions

The main conclusion obtained is that the objective of the project has been achieved, that is to provide the CORE network emulator with an IMS system to provide audio-visual services.

It has been possible to develop a platform where any kind of emulated network can be created, and where an IMS system is available that provides everything necessary for the management of sessions for multimedia services. This platform is adaptable to any network under study, so that it can be used for the development and implementation of multimedia services over IP networks.

The technologies used, such as CORE and Open IMS have been the heart of the project. Both technologies are easily accessible and easy to install thanks to the manuals they offer. Anyway, we overcome a number of challenges:

- Configure Open IMS to handle requests from machines other than where IMS servers are running.

- Configure a DNS server to serve all machines (virtual within the emulated network scenario, virtual, or real) of the scenario.

- Manage the IP addressing of the scenario, both of virtual machines within the emulated network scenario, virtual machines and real machines. As part of this task a DHCP server was configured to serve addresses more easily and the DNS server of the scenario.

- Configure the IP communication of the different virtual machines and of these with the virtual machines in the emulation.

- Installation and use of IMS clients in the virtual machines of the emulated network scenario.

The platform works properly to test audiovisual services with two tools: a typical IMS user agent, such as that used by real users (Monster), and an experimentation tool with SIP signaling (SIPp) that allows very flexible Define SIP signaling exchanges and verify that they occur correctly. Both tools have been able to register and deregister users, and set up sessions. In addition, traffic exchanged between user agents and IMS servers has been captured, showing

how powerful the platform is developed as an environment for experimentation and testing of audiovisual services over IP networks.

ii. Future work

The possible future work in line with this project is:

- Automate the configuration of the environment in IMS and CORE in which the project is developed.
- Provide an audiovisual server with a physical machine, so that one end of the communication is in the emulated network scenario and the other a machine in a real network that can visualize video in the calls.
- Deploy a video server to which services can be ordered using IMS.
- Incorporate other types of protocols that are alternative or complementary to SIP signalling that would allow comparative studies, such as RTSP or XMPP.