TRABAJO DE FIN DE CARRERA

TÍTULO: Implementación de un servidor proxy SIP en JAVA

AUTOR: Alberto Guirao Villalonga

DIRECTOR: Toni Oller Arcas

FECHA: 7 de julio de 2006

Título: Implementación de un servidor proxy SIP en JAVA

Autor: Alberto Guirao Villalonga

Director: Toni Oller Arcas

Fecha: 7 de julio de 2006

Resumen

La Voz sobre IP (VoIP) es la tecnología capaz de transmitir conversaciones telefónicas a través de una red de conmutación de paquetes, en vez de la red de conmutación de circuitos tradicional.

Este documento explica las ventajas y los inconvenientes de la Voz sobre IP, así como sus protocolos más utilizados (H.323 y SIP).

Veremos con mayor extensión el protocolo SIP, ya que es el de mayor futuro y los diversos componentes de la arquitectura SIP. Además veremos como esta arquitectura SIP es usada en un marco móvil 3G como es IMS (IP Multimedia Subsystem).

En concreto nos centraremos en cómo se ha desarrollado un servidor SIP con funciones de registro, proxy y redirección, inspirándose en un proxy SIP ya existente. Este servidor cuenta además de una herramienta web para la gestión de los usuarios. Además veremos como el servidor proxy implementado se puede integrar dentro de la arquitectura IMS y que funciones desempeña.

Title: Implementación de un servidor proxy SIP en JAVA

Author: Alberto Guirao Villalonga

Director: Toni Oller Arcas

Date: July, 7th 2006

Overview

The Voice over IP (VoIP) is the technology to able to transmit telephone conversations through a network of commutation of packages, instead of the traditional network of commutation of circuits.

This document explains the advantages and the disadvantages of the Voice over IP, like the more used protocols (H.323 and SIP).

We will see with greater extension protocol SIP, since he is the one of greater future, and the diverse components of architecture SIP. In addition we will see as this architecture SIP is used in a mobile 3G world as it is IMS (IP Multimedia Subsystem).

In particular we will be centred in a SIP server with register, proxy and redirect functions, being inspired by already existing SIP proxy. This server counts in addition to a Web tool for the management of the users. In addition we will see like the implemented proxy server is possible to be integrated within the architecture IMS and functions that carries out.

ÍNDICE

ÍNDICE DE ILUSTRACIONES

INTRODUCCIÓN

En los últimos años Internet ha evolucionado ofreciendo nuevos servicios. Aún así actualmente coexisten dos grandes redes, una para telefonía y otra para datos. Por este motivo se creó VoIP (Voz sobre IP) con la idea de formar una única red de datos que sea capaz de ofrecer los servicios de telefonía con la misma calidad que tiene actualmente la red telefónica.

VoIP no tiene un estándar universal, sino que hay una serie de protocolos que cumplen con los requisitos de VoIP, entre ellos están H.323 y SIP. H.323 fue el primer protocolo que cumplió las normas de VoIP por lo que se popularizó inicialmente. Más tarde se creó SIP y hoy por hoy es el que tiene una mayor proyección.

SIP es un protocolo de señalización basado en el modelo de petición-respuesta, donde un cliente genera un mensaje de petición y el servidor le envía una respuesta a esa petición. Además este protocolo define una serie de elementos como los User Agents (UA) y servidores. Los primeros son las entidades básicas de la arquitectura SIP capaces de enviar y recibir mensajes. Las funciones principales de los servidores SIP son las funciones de proxy, registro y redirección.

En este proyecto se pretende crear un servidor SIP capaz de cumplir estas funciones básicas. Este servidor se ha desarrollado en Java y se ha tomado como referencia un servidor SIP ya existente llamado SER y desarrollado por iptel.org. Además para la administración del servidor proxy se ha creado una interfaz web que permite una fácil gestión.

El trabajo se ha organizado de la siguiente manera. En el primer capitulo se hace una introducción al mundo de la Voz sobre IP (VoIP) así como los protocolos de señalización (H.323 y SIP/SDP) y los protocolos del plano multimedia (RTP/RTCP) existentes para implementar VoIP.

En el capítulo dedicado al protocolo SIP (Session Initiation Protocol) describiremos el funcionamiento de SIP así como sus elementos básicos (User Agents y Proxy SIP) y viendo los intercambios de mensajes que estos elementos realizan. También se mostrará el funcionamiento del proxy SIP y el sentido de realizar un proxy SIP en Java.

Después veremos como en un entorno de móviles de 3G existe una especificación de 3GPP llamada IMS y que se basa en SIP para el desarrollo y despliegue de servicios.

Una vez desarrollado el servidor proxy, este se ha adaptado para que cumpla los requisitos definidos por IMS: autentificación (por ejemplo AKA), B2BUA y filter criteria.

Como conclusión, veremos como el servidor SIP que se ha desarrollado puede actuar como proxy SIP o como un elemento más dentro de la arquitectura IMS. En cada caso se verá como se comporta el proxy.

CAPÍTULO 1. VOZ SOBRE IP

1.1. Introducción a VoIP

La telefonía tal y como la conocemos se enfrenta a su propio futuro: la creciente popularización de servicios de llamadas a través de Internet puede suponer una auténtica revolución en las comunicaciones. La razón de este hecho es básicamente el ahorro que supone el mantenimiento, gestión y uso de una única red para transmisión tanto de voz como de datos. Ya hay alternativa a la telefonía tradicional, aunque aún existen ciertos problemas de regulación que podrían lastrar su desarrollo e implantación.

Las llamadas por la Red no son algo nuevo, aunque sí lo es su popularización. Básicamente, este tipo de servicio usa un sistema llamado Voz Sobre IP **[1]** (VoIP) para transmitir llamadas de voz a través de paquetes de datos utilizando la red de Internet.

La tecnología IP es ya utilizada (o se está implementando rápidamente) en las redes troncales de comunicaciones. Por ejemplo, Telefónica afirma que el 70% de las transmisiones de voz y datos ya se realiza a través de esta tecnología.

1.1.1. Ventajas de VoIP

Sus ventajas potenciales sobre la telefonía convencional son múltiples: es un servicio mucho más barato (a veces, incluso gratis), permite el nomadismo (es decir, el uso de un mismo número de teléfono independientemente de dónde se encuentre físicamente el usuario) y tiene capacidad multimedia. Además, la calidad del servicio es cada vez mejor, similar a la del teléfono convencional, y permite la interconexión.

Además este sistema permite una mayor optimización de los recursos, ya que la voz puede viajar comprimida y utiliza detectores de silencios para no enviar datos cuando los usuarios permanecen en silencio.

1.1.2. Inconvenientes de VoIP

El problema al que se enfrenta el desarrollo de la VoIP para usuarios en España es la definición del propio servicio, es decir, si se puede considerar 'telefonía' o no. La Comisión del Mercado de las Telecomunicaciones (CMT), de momento, lo ha diferenciado, y el Gobierno lo ha denominado *"servicio vocal nómada con capacidad multimedia"*.

Además las operadoras de telefonía están sujetas a una regulación que las obliga, entre otras cosas, a ofrecer servicios como el acceso a números de emergencia (112) y a garantizar unos niveles de calidad mínimos que son difíciles de asegurar utilizando la propia red de Internet como transporte.

1.2. Estándar VoIP

El estándar de Voz sobre IP fué definido en 1996 por la ITU (Unión Internacional de Telecomunicaciones). VoIP no tiene un protocolo definido como estándar, sino que define una serie de normas y elementos de red que han de tener toda red de VoIP.

1.3. H.323

El estándar H.323 **[2]** fue el primero en cumplir con la normativo VoIP. Especifica los componentes, protocolos y procedimientos que proveen los servicios de comunicación multimedia sobre redes de paquetes sin garantía de calidad de servicio, tanto para sesiones multipunto como punto a punto. La tecnología de red más común en la que se están implementando H.323 es IP (Internet Protocol). Además, H.323 también define la señalización necesaria para comunicaciones multimedia sobre redes IP. Para la capa multimedia utiliza los protocolos RTP/RTCP.

Los terminales y equipos H.323 soportan aplicaciones con requerimientos de tiempo real (voz y vídeo), así como aplicaciones de datos y combinaciones de ellas (videotelefonía ...etc). Los terminales H.323 pueden ser terminales explícitamente diseñados a este fin o pueden estar integrados en ordenadores en formato software.

El estándar H.323 incluye entre otras las siguientes recomendaciones:

- H.225.0: paquetización, sincronización y señalización.
- H.245: control del canal.
- G.711, G.722, G.723.1, G.728, G.729: codificación audio.
- Además también define recomendaciones sobre conferencias de datos en tiempo real, seguridad.....

H.323 define una serie de entidades en una red H.323 con una serie de funcionalidades:

- **Gatekeepers:** son entidades de control y señalización, siendo las entidades más complejas. Se encargan de gestionar el ancho de banda y a los usuarios.
- **Gateways:** los gateways (pasarelas) son los sistemas encargados de permitir que los equipos H.323 puedan operar con otras redes. Desarrollan la traducción de la señalización, información de control e

información de usuario, posibilitando así interoperabilidad entre redes, terminales y servicios, haciendo viable la integración de servicios aún con otras plataformas.

- **Terminales:** un terminal H.323 posibilita comunicaciones bidireccionales en tiempo real de voz, datos y vídeo. H.323 especifica los modos de operación requeridos para que los terminales de audio, vídeo y datos trabajen conjuntamente.

Hasta la fecha, el estándar H.323 ha evolucionado desde la primera versión H.323v1, hasta la última versión H323v4, mejorando la primera versión en cuestiones como seguridad, servicios suplementarios, identificación de llamadas, conexión rápida.....etc.

1.4. SIP

El Protocolo de Inicialización de Sesiones (SIP) **[3]** es un protocolo desarrollado por el IETF con la intención de ser el estándar para la iniciación, modificación y finalización de sesiones interactivas de usuario donde intervienen elementos multimedia como el video, voz, mensajería instantánea, juegos online y realidad virtual. En Noviembre del año 2000, SIP fue aceptado como el protocolo de señalización de 3GPP y elemento permanente de la arquitectura IMS (*IP Multimedia Subsystem*). SIP es uno de los protocolos de señalización para voz sobre IP, es usado simplemente para iniciar y terminar llamadas de voz y video. Todas las comunicaciones de voz/video van sobre RTP (*Real-time Transport Protocol*).

Un objetivo de SIP fue aportar un conjunto de las funciones de procesamiento de llamadas y capacidades presentes en la red pública conmutada de telefonía. Así, implementó funciones típicas que permite un teléfono común como son: llamar a un número, provocar que un teléfono suene al ser llamado o escuchar la señal de tono o de ocupado.

SIP también implementa muchas de las características del procesamiento de llamadas de SS7, aunque los dos protocolos son muy diferentes. SS7 es altamente centralizado, caracterizado por una compleja arquitectura central de red y unos terminales tontos (los tradicionales teléfonos de auricular).

SIP es un protocolo punto a punto. Como tal requiere un núcleo de red sencillo (y altamente escalable) con inteligencia distribuida en los extremos de la red, incluida en los terminales (ya sea mediante hardware o software). Muchas características de SIP son implementadas en los terminales en oposición a las tradicionales características de SS7, que son implementadas en la red.

Al ser un protocolo basado en texto posibilita una fácil implementación y depuración, y eso lo hace flexible y extensible. El sobreencabezamiento que implica usar un protocolo basado en texto no tiene mayor trascendencia, ya que SIP es un protocolo de señalización, y no es un protocolo para el intercambio de datos de usuario, donde si tendría consecuencias.

1.4.1. SDP

El SDP **[4]**, o Session Description Protocol, es un protocolo que define un formato
para describir los parámetros de inicialización de un streaming de media. Ha sido caracterizado por la IETF en el RFC 2327. SDP empezó como un componente de SAP (Session Announcement Protocol), pero se han encontrado otras utilidades, en concreto SDP se ha integrado dentro del protocolo SIP e incluso en un formato a parte, describiendo sesiones multicast.

En la descripción de un flujo se observa:

- El tipo de media (video, audio, etc.)
- El protocolo de transporte (RTP/UDP/IP, H.323, etc.)
- El formato de la media (H.261 video, MPEG video, etc.)

En una sesión IP multicast, se encuentra:

- Dirección multicast para la media
- Puerto de transporte para la media

Para una sesión IP unicast, se encuentra:

- Dirección remota para la media
- Puerto de transporte para la dirección de contacto

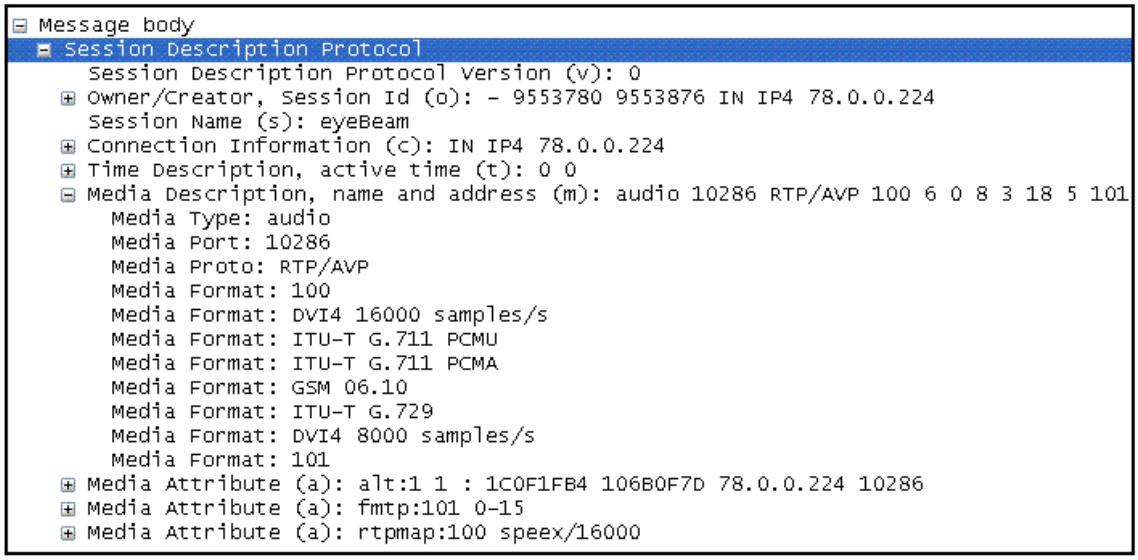

Ilustración 1. Ejemplo de paquete SDP capturado en el ethereal

1.5. Comparativa H.323 – SIP

SIP se caracteriza porque sus promotores tienen sus raíces en la comunidad IP y no en la industria de las telecomunicaciones. SIP ha sido estandarizado y dirigido principalmente por el IETF mientras que el protocolo de VoIP H.323 ha

sido tradicionalmente más asociado con la Unión Internacional de Telecomunicaciones. Sin embargo, las dos organizaciones han promocionado ambos protocolos del mismo modo.

La principal diferencia es que H.323 codifica los mensajes en un formato binario compacto adecuado para conexiones de gran ancho de banda, mientras que SIP guarda similitud a HTTP codificando los mensajes en formato ASCII, por lo que es legible pos humanos y sigue una estructura de petición-respuesta. H.323 es utilizado en videoconferencias mientras que SIP está más extendido para la telefonía IP.

1.6. Plano multimedia de VoIP

Tanto H.323 como SIP son protocolos de señalización para VoIP, pero necesitan un contenedor para transportar los datos multimedia en tiempo real. Este contenedor es el RTP/RTCP

1.6.1. RTP

El Protocolo de Transporte en tiempo Real (RTP) **[5]** es el encargado de la transmisión de los datos de las sesiones multimedia. Este protocolo está definido en el RFC 1889 y trabaja sobre el protocolo de transporte UDP. Al usar UDP, este protocolo tiene que aportar una información adicional a la aplicación para que la reproducción del flujo multimedia sea igual que en el origen.

1.6.2. RTCP

RTCP **[6]** realiza el control del flujo multimedia enviado por RTP. Por tanto RTCP aporta información sobre los participantes de la sesión y realiza una monitorización de la calidad de servicio.

Al realizar funciones de control, este protocolo trabaja sobre el protocolo de transporte TCP. Por este motivo los paquetes son bastante grandes, por lo que su envío queda reducido a un 5% de los paquetes enviados en una sesión RTP.

CAPÍTULO 2. IMS

2.1. Introducción a IMS

La falta de aplicaciones y servicios UMTS atractivos para el gran público y las tendencias de convergencia con Internet, motiva que se introduzca en 3GPP el concepto All-IP.

All-IP es una visión sobre el futuro de las redes de comunicaciones que ofrece modos de acceso que se integran de forma transparente en una capa de red basada en el protocolo de Internet IP. Además proporciona beneficios a los operadores como ahorro de costes de infraestructura, mejor escalabilidad, mayor flexibilidad y simplificación de la operación y mantenimiento.

IMS **[7]** (IP Multimedia Subsystem) representa la implantación de la arquitectura All-IP en 3G que permite al operador proporcionar a sus abonados una atractiva oferta de servicios multimedia como videoconferencia, voz sobre IP, “streaming”, mensajería instantánea, web, etc.

IMS es un sistema de control de sesión diseñado con tecnologías de Internet adaptadas al mundo móvil, que hace posible la provisión de servicios móviles multimedia sobre conmutación de paquetes (servicios IP multimedia).

SIP se encargaría de las funciones para el registro, establecimiento, liberación y mantenimiento de las sesiones IMS. Incluyendo funciones de enrutado de sesiones e identificación de usuarios y nodos, y también habilitando todo tipo de servicios suplementarios.

Características de IMS:

- La comunicación orientada a sesión de un usuario a otro(s) usuario(s), o de un usuario a un servicio.
- La comunicación en tiempo real o diferido.
- Las sesiones IP multimedia con un nivel adecuado de Calidad de Servicio para vídeo, audio y sonido, texto, imagen, etc.
- La identificación de usuarios, servicios y nodos mediante URIs (Universal Resource Identifier. Éstos ya no tienen que manejar números de teléfono sino nombres como el correo electrónico.

IMS no define las aplicaciones o servicios que pueden ofertarse al usuario final, sino la infraestructura y capacidades del servicio que los operadores o proveedores de servicio pueden emplear para construir sus propias aplicaciones y producir su oferta de servicios. En este sentido, IMS no impone límites, son la capacidad de la red de acceso y las características de los terminales las que fijan las restricciones.

Los servicios IMS pueden implementarse, por ejemplo, en una sola aplicación de usuario final que hace un uso coordinado y simultáneo de:

- Mensajería IP multimedia (instantánea y diferida)
- Servicios web
- Videoconferencia y llamadas de voz sobre IP
- Streaming
- Juegos en red
- Cualquier otro servicio de Internet basado en TCP/IP (por ejemplo los populares clientes de mensajería instantánea para PC).

2.2. Arquitectura IMS

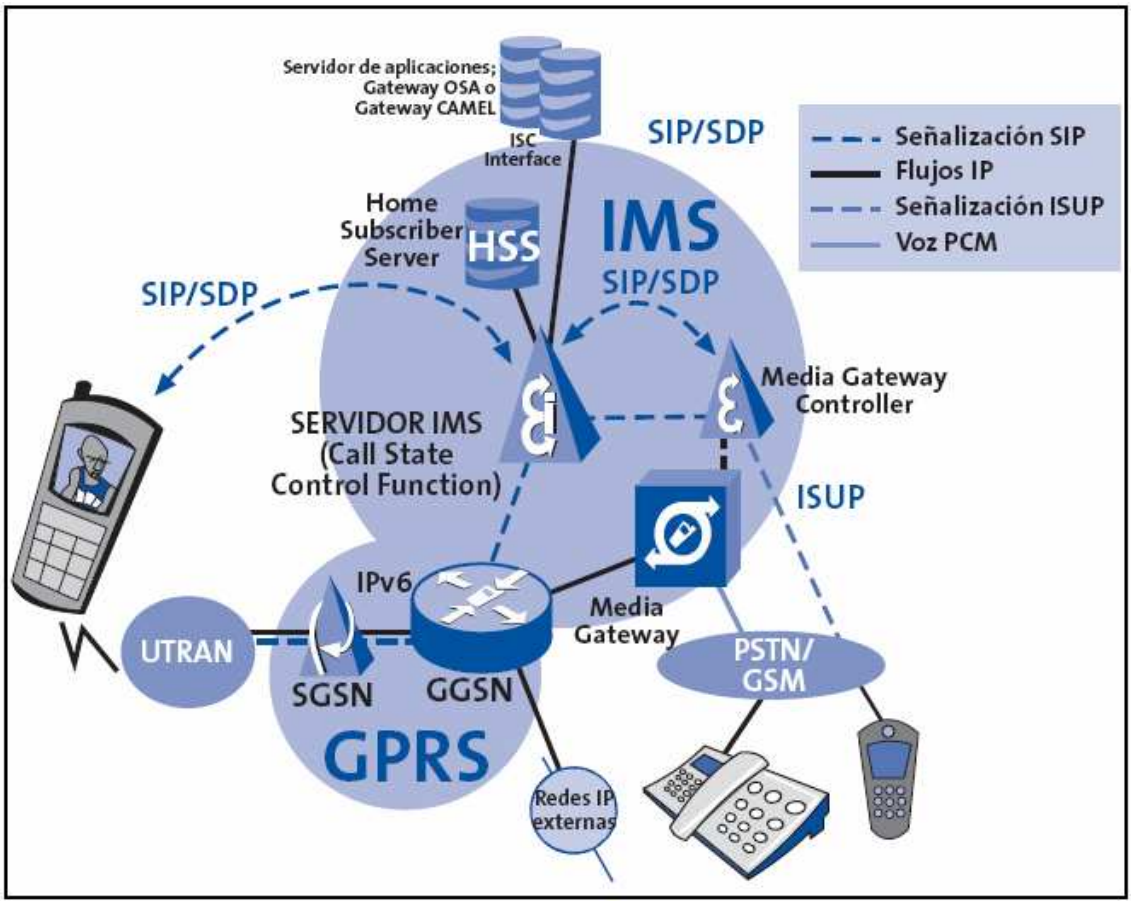

Ilustración 2. Esquema de la arquitectura IMS

La entidad funcional clave es el nodo CSCF (Call State Control Function), que es básicamente un servidor SIP con funciones de proxy. El CSCF ejecuta tres roles diferentes en la operativa de IMS: Proxy CSCF(P-CSCF), Serving CSCF(S-CSCF) y Interrogating CSCF(I-CSCF).

Además dentro de IMS hay otros nodos como son: Home Subscriber Server(HSS), Media Gateway Control Function (MGCF), IP Multimedia Media Gateway (IM-MGW) y los servidores de aplicación y las pasarelas.

En la ilustración 2 se ha omitido una serie de conjuntos de entidades funcionales por sencillez, pero que están definidas por 3GPP. Tal es el caso, por ejemplo, de la arquitectura para mensajería IMS, de las entidades funcionales del servicio de presencia, de los elementos para sesiones multiparticipante, de la arquitectura de interfuncionamiento con redes externas SIP IPv4 y de las entidades específicas de tarificación.

2.2.1. Proxy CSCF

El Proxy CSCF (P-CSCF) es el punto de entrada al subsistema IMS. Recibe directamente la señalización IMS desde el terminal, vía GPRS. Implementa las funciones de protección de señalización (seguridad) y el control de recursos del subsistema de transporte. En itinerancia (roaming) es el nodo en la red visitada que se encarga de enrutar la señalización de registro y sesión desde los terminales que se encuentran en situación de itinerancia hasta la red IMS nativa. Además, ejecuta las funciones comunes a los demás CSCF: el procesado y enrutado de señalización, la consulta del perfil de usuario en el HSS y la tarificación.

2.2.2. Serving CSCF

A cada usuario registrado en IMS se le asigna un S-CSCF, el cual se encarga de enrutar las sesiones destinadas o iniciadas por el usuario. También realiza el registro y autenticación del abonado IMS y la provisión de los servicios IMS (mediante el desvío de señalización a los servidores de aplicación). Asimismo aplica las políticas del operador de red y genera los registros de tarificación.

2.2.3. Interrogating CSCF

I-CSCF es nodo intermedio que da soporte a la operación IMS. El I-CSCF ayuda a otros nodos a determinar el siguiente salto de los mensajes SIP y a establecer un camino para la señalización. Durante el registro, el P-CSCF se ayuda del I-CSCF para determinar el S-CSCF que ha de servir a cada usuario. En situaciones de itinerancia y en sesiones interred, el I-CSCF es el punto de entrada conocido por la red IMS externa e indica el siguiente salto a realizar para la señalización. Opcionalmente, el I-CSCF efectúa funciones de ocultación de la topología de la red IMS ante redes externas, de forma que los elementos ajenos a IMS no puedan averiguar cómo se gestiona la señalización internamente (por ejemplo, el número, el nombre y la capacidad de los CSCF).

2.2.4. Home Subscriber Server

El HSS es una base de datos que almacena y gestiona el perfil del servicio IMS del abonado, almacena las claves de seguridad y genera vectores de autenticación, registra el estado de los abonados y almacena el nodo S-CSCF con el que el abonado se ha registrado, etc.

2.2.5. Media Gateway Control Function

La Media Gateway Control Function (MGCF) forma parte de la arquitectura de interfuncionamiento de IMS con las redes de circuitos. En concreto, implementa el plano de control del interworking, traduciendo la señalización IMS SIP/SDP a SS7, y viceversa. También se encarga de controlar la operación del IM-MGW.

2.2.6. IP Multimeda Media Gateway

El IP Multimedia Media Gateway (IM-MGW) implementa el plano de usuario de la arquitectura de interoperación de IMS con las redes de circuitos. En las redes TDM de circuitos se encarga de la transcodificación de flujos IMS sobre IP a datos de usuario.

2.2.7. Servidores de aplicación y pasarelas

3GPP define interfaces IMS entre el S-CSCF y el plano de servicios, de esta manera la señalización puede desviarse hacia el plano de servicio en base a una serie de criterios que se recogen en el perfil de abonado, que el HSS alberga y que el S-CSCF descarga durante el registro de cada abonado. Por tanto, el S-CSCF puede transferir la señalización de un registro o sesión hacia un servidor de aplicaciones SIP, o transferirla hacia una pasarela OSA o hacia una pasarela CAMEL, que traduce SIP en CAP

CAPÍTULO 3. ARQUITECTURA SIP

3.1. Mensajes SIP

El protocolo SIP es parecido al protocolo HTTP. Esto hace que SIP tenga una estructura cliente-servidor donde el cliente es el encargado de iniciar la comunicación y el servidor se encargada de responder a la petición. Además, al igual que ocurre con los mensajes HTTP, los mensajes de respuesta SIP tienen un código numérico asociado:

- **1xx :** Respuesta informativa
- **2xx :** Mensaje afirmativo
- **3xx :** Redirección
- **4xx :** Error en la petición del usuario
- **5xx :** Error en el servidor
- **6xx :** Error global

Los mensajes SIP están codificados en ASCII, por lo que son legibles por el ser humano. Este hecho implica que resulte más fácil trabajar con ellos.

Los mensajes SIP están estructurados de la siguiente forma:

- **Request-Line:** Contiene la URL del destino de la solicitud.
- **To:** Indica el receptor de la petición.
- **From:** Indica quien ha iniciado la petición.
- **Via:** Indica la ruta tomada por la petición.
- **Call-ID:** Identifica a todos los registros de un cliente determinado.
- **CSeq:** Contiene la petición del mensaje de solicitud y el número de secuencia.
- **Contact:** URL de contacto para comunicaciones adicionales.

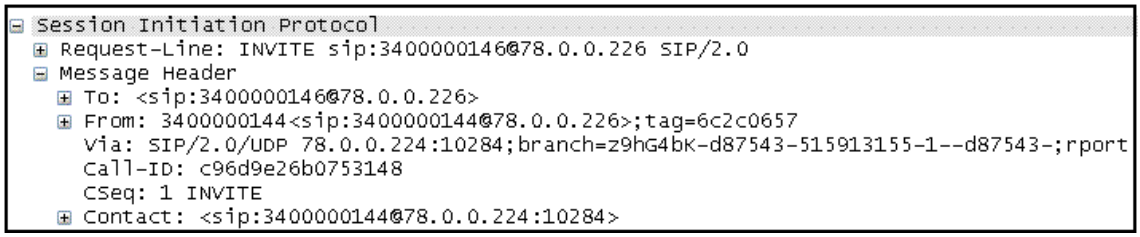

Ilustración 3. Ejemplo de mensaje SIP capturado con el Ethereal

3.1.1. User Agent

El User Agent (UA) es la entidad básica de la arquitectura SIP capaz de enviar y recibir mensajes. Los UA se definen según sean clientes (User Agent Client) o servidores (User Agent Server). Un cliente SIP (teléfono IP, softphones, etc) es capaz de actuar como cliente o como servidor, dependiendo del caso.

Ilustración 4. Cliente SIP X-Lite

El cliente SIP utilizado en este proyecto ha sido el softphone X-Lite debido a que es fácil de usar y es de libre distribución. X-Lite esta desarrollado por la compañía CounterPath. Esta compañía también desarrolla softphones comerciales de mayor valor añadido que el X-Lite.

3.1.1.1. Ejemplo de comunicación entre Users Agents

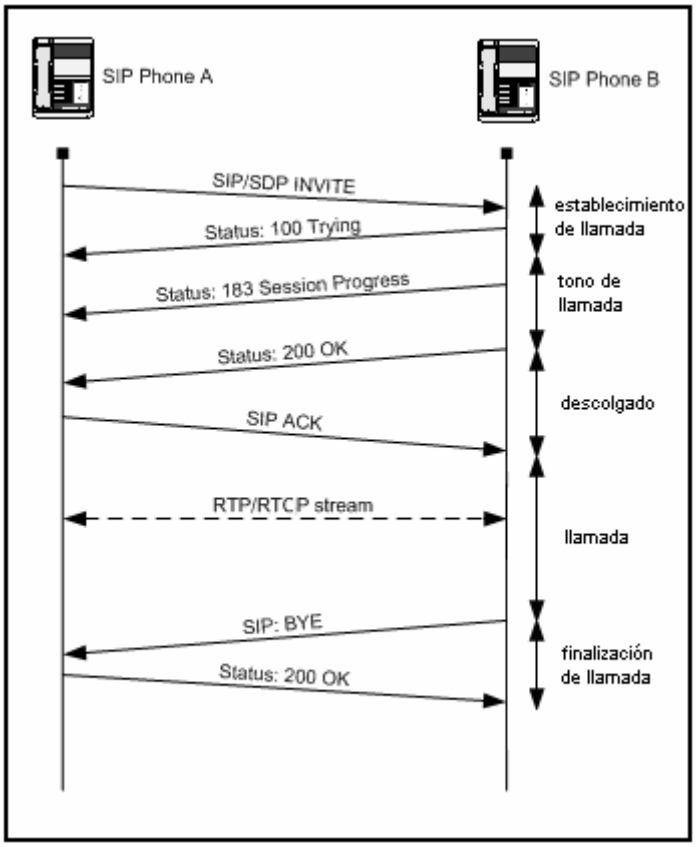

Ilustración 5. Mensajes SIP intercambiados en una comunicación

En la ilustración 5 el teléfono A actúa como cliente y el teléfono B como servidor. En una comunicación sin errores los mensajes intercambiados serian los siguientes:

- **SIP/SDP INVITE:** El cliente hace una petición de establecimiento de llamada al servidor.
- **100 Trying:** El servidor indica que ha recibido correctamente el INVITE y que va a intentar establecer la conexión
- **183 Session Progress:** La conexión se ha establecido y ahora se espera la intervención del usuario final para que descuelgue el teléfono. Este mensaje se puede sustituir por el mensaje 180 Ringing. En el cliente se escucharían los tonos de llamada mientras que en el servidor el teléfono sonaría para llamar la atención del usuario final.
- **200 OK:** El usuario final ha descolgado el teléfono y el servidor indica al cliente este hecho.
- **SIP ACK:** El cliente confirma el descuelgue y a partir de aquí empieza la comunicación entre los usuarios a través de la sesión RTP.
- **SIP BYE:** Se envía este paquete cuando un usuario cuelga el teléfono, entonces el otro teléfono lo tiene que confirmar con un 200 OK. Cuando se recibe la confirmación del otro terminal se da por finalizada la llamada.

3.1.2. Proxy SIP

En el ejemplo anterior el usuario que realiza la llamada tiene que conocer la situación del otro terminal para poder establecer la conexión, sino la conoce el mensaje SIP nunca podrá llegar al destino y no se podrá realizar la comunicación. En los escenarios reales este hecho no se da, los Users Agents no se conocen entre ellos, solo conocen la situación de un proxy SIP y por tanto cuando quieran llamar a un usuario tendrán que comunicárselo a este proxy.

Además de encaminar los mensajes hacia su destino, el proxy realiza las funciones de autentificación de usuarios. Para ello, los usuarios tienen que registrarse en el proxy SIP para poder realizar una llamada.

3.1.2.1. Ejemplo de comunicación entre Users Agents con proxy

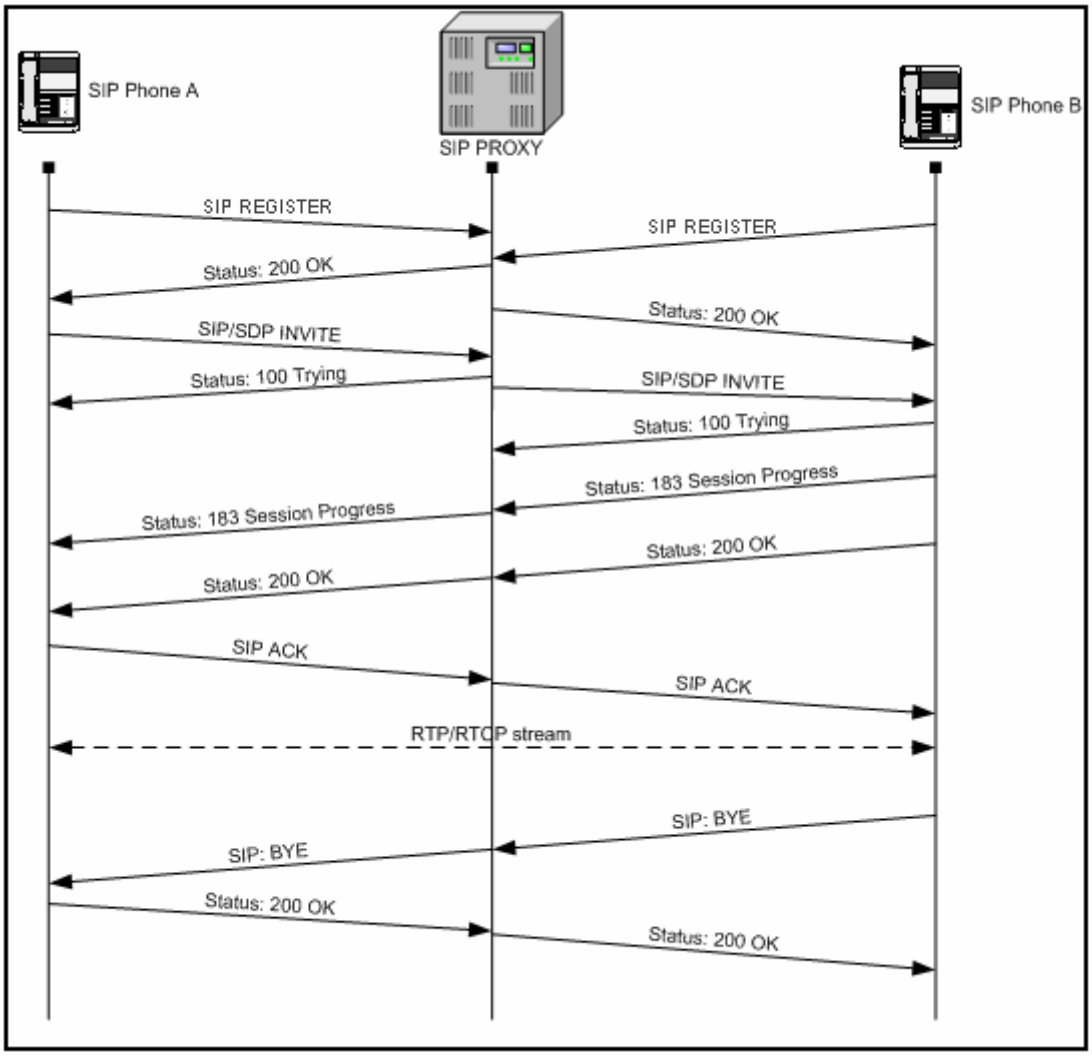

Ilustración 6. Mensajes SIP intercambiados en una comunicación con proxy

El esquema que se muestra en la ilustración 6 es el mismo mostrado en la ilustración 5. La diferencia es que ahora el teléfono A no tiene que preocuparse en localizar el teléfono B, ya que de eso se encargará el proxy SIP.

También hay que destacar que para realizar la llamada con éxito los dos terminales se tienen que haber registrado previamente en el proxy como usuarios, para ello utilizan el mensaje “SIP REGISTER”. Si el registro se finaliza con éxito por parte del proxy (es decir, el usuario tiene permiso para utilizar el proxy), este le devolverá un 200 OK. A partir de entonces el usuario ya podrá realizar o recibir llamadas.

Cada mensaje SIP tendrá como destino el proxy, y este se encargará de localizar al usuario final y reenviarle dicho mensaje. Por contra, la sesión RTP/RTCP si que se realiza entre los usuarios finales, por lo que estos tipos de paquetes no pasarán por el proxy. Así se evita sobrecargar al proxy de trabajo ya que en una llamada normal este tipo de paquetes serán los de mayor número.

3.2. Proxy comercial: SER - OPENSER

Uno de los proxys SIP más utilizados es el SIP Express Router (SER) **[8]**. Este proxy esta desarrollado por la compañía alemana iptel.org y es de libre distribución. Funciona bajo un sistema operativo linux, soporta 150 llamadas por segundo y esta programado en lenguaje C.

Este proxy esta programado de forma modular. Los módulos se cargan dentro del núcleo del SER en el momento de arrancar por medio de un archivo de texto de configuración llamado “ser.cfg”. Esta propiedad hace que el SER sea configurable por el usuario cargando los módulos que le interesan e incluso crear sus propios módulos siguiendo unas pautas marcadas por iptel.org.

Entre los módulos desarrollados por iptel.org cabe destacar:

- **Auth y auth_db:** contienen las funciones necesarias para la autentificación con la base de datos.
- **Enum:** (Electronic NUMber/tElephone NUmber Mapping). Implementa las funciones necesarias para transformar un número de teléfono internacional en una consulta a una base de datos ENUM de un servidor DNS.
- **Msilo:** este módulo proporciona la opción de poder enviar mensajes a los usuarios del SER que se encuentren offline. Cuando un usuario offline se conecta, se le envían todos los mensajes almacenados.
- **Nathelper:** este módulo incorpora la ayuda necesaria para poder atravesar los NAT traversals.
- **Pa:** este módulo implemente un servidor con estado de presencia. Es decir, el servidor avisa cuando el estado de algún usuario cambia

- **Permissions:** este módulo es usado para determinar si una llamada tiene permiso para establecerse.
- **Pdt:** este módulo traduce los códigos numéricos a dominios y los actualiza respecto al R-URI.
- **Registrar:** el módulo contiene el proceso lógico para los mensajes *REGISTER*.
- **Sms:** este módulo hace posible la comunicación entre SIP y la red de GSM. La comunicación es posible mediante SIP a SMS y viceversa.
- **Transaction:** implementa la lógica necesaria para gestionar las transacciones SIP.

Además ofrece la posibilidad de gestionar los usuarios de la base de datos del SER a través de la consola gracias al comando “serctl”. Esta herramienta contiene las opciones de crear, modificar o eliminar usuarios autorizados a utilizar el SER o a mostrar que usuarios se encuentran online, entre otras opciones.

Aparte del SER desarrollado por iptel.org, existe un proyecto paralelo llamado OPENSER **[9]**. Este proyecto esta dirigido por 3 de los programadores del SER y consiste en desarrollar nuevos módulos para este servidor proxy. Para un desarrollo más rápido se cuenta con la aportación desinteresada de la comunidad SIP. Si se crean nuevos módulos interesantes, estos se acaban integrando en el SER.

Una gran desventaja del SER es, que al estar programado en lenguaje C, el programador ha de trabajar con unas grandes estructuras de datos. Como ejemplo, en la ilustración 7 vemos como es la estructura de datos en C de un mensaje SIP. Como se puede observar, es muy costoso para el programador gestionar toda esa estructura para un solo mensaje SIP.

```
struct sip_msg {
    unsigned int id;               /* message id, unique/process*/
    struct msg_start first_line;   /* Message first line */
    struct via_body* via1;         /* The first via */
    struct via_body* via2;         /* The second via */
    struct hdr_field* headers;     /* All the parsed headers*/
    struct hdr_field* last_header; /* Pointer to the last parsed header*/
    int parsed_flag;               /* Already parsed header field types */

    /* Via, To, CSeq, Call-Id, From, end of header*/
    /* first occurrence of it; subsequent occurrences
     * saved in 'headers'
     */

    struct hdr_field* h_via1;
    struct hdr_field* h_via2;
    struct hdr_field* callid;
    struct hdr_field* to;
    struct hdr_field* cseq;
    struct hdr_field* from;
    struct hdr_field* contact;
    struct hdr_field* maxforwards;
    struct hdr_field* route;
    struct hdr_field* record_route;
    struct hdr_field* content_type;
    struct hdr_field* content_length;
    struct hdr_field* authorization;
    struct hdr_field* expires;
    struct hdr_field* proxy_auth;
    struct hdr_field* www_auth;
    struct hdr_field* supported;
    struct hdr_field* require;
    struct hdr_field* proxy_require;
    struct hdr_field* unsupported;
    struct hdr_field* allow;
    struct hdr_field* event;

    char* eoh;        /* pointer to the end of header (if found) or null */
    char* unparsed;   /* here we stopped parsing*/

    struct ip_addr src_ip;
    struct ip_addr dst_ip;

    char* orig;       /* original message copy */
    char* buf;        /* scratch pad, holds a modified message,
                       *  via, etc. point into it
       */
    unsigned int len; /* message len (orig) */

    /* modifications */
    str new_uri;              /* changed first line uri*/
    int parsed_uri_ok;        /* 1 if parsed_uri is valid, 0 if not */
    struct sip_uri parsed_uri; /* speed-up > keep here the parsed uri*/

    struct lump* add_rm;        /* used for all the forwarded
                                 * requests */
    struct lump* repl_add_rm;   /* used for all the forwarded replies */
    struct lump_rpl *reply_lump; /* only for locally generated replies !!!*/

    char add_to_branch_s[MAX_BRANCH_PARAM_LEN];
    int add_to_branch_len;

    /* index to TM hash table; stored in core to avoid unnecessary calcs */
    unsigned int  hash_index;

    /* allows to set various flags on the message; may be used for
     * simple inter-module communication or remembering processing state
     * reached
     */
    flag_t flags;
};
```

Ilustración 7. Estructura del SER para un mensaje SIP

CAPÍTULO 4. PROYECTO JSER

4.1. Motivaciones del proyecto

Aunque el SER ofrece la posibilidad de implementar nuevos módulos, la forma de programarlos no es nada amigable. Hay que seguir unas rígidas estructuras de datos que provocan que el desarrollo de nuevos módulos sea una tarea costosa.

Por eso, se ha decidido desarrollar un nuevo proxy SIP que siga la filosofía del SER pero que sea más amigable desde el punto de vista del programador. Este proxy se ha desarrollado en java (lenguaje orientado a objetos, muy extendido, multiplataforma e independiente de la arquitectura) y se le ha dado el nombre de JSER (Java SER).

Como ejemplo, en la figura 7 veíamos como tratar un mensaje con el SER suponía utilizar una estructura muy grande, por el contrario para el JSER, tenemos una clase llamada *UserRequest*, que pasando por parámetro el mensaje SIP que hemos recibido obtenemos los valores más importantes del mismo.

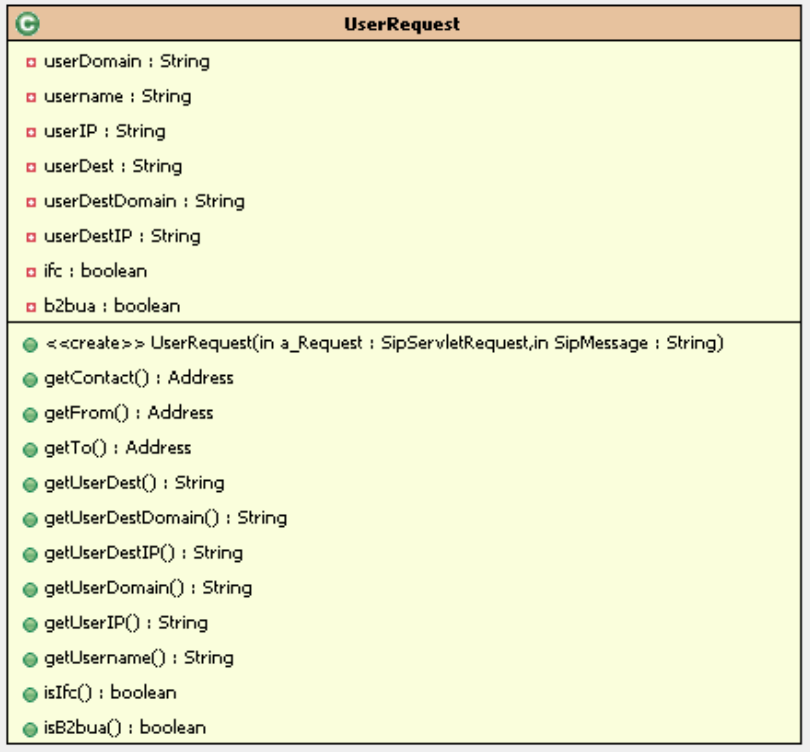

Ilustración 8. Clase del JSER para un mensaje SIP

4.1. Alcance del proyecto

En este proyecto se pretende implementar un servidor SIP que cumpla con las funciones de proxy, registro y redirección definidas en el RFC 3261. Además se pretende adaptar este servidor proxy para que actúe como un elemento del S-CSCF dentro de la arquitectura IMS. Este servidor SIP lo hemos denominado JSER (Java SER), ya que está inspirado en el proxy SIP de iptel.org llamado *SIP Express Router* (SER).

Para cumplir estos requisitos, el trabajo se ha separado en diferentes módulos:

- **Núcleo:** Es el encargado de cumplir con las funciones de registro, proxy y redirección.
- **Módulo de administración:** Cumple con la función de gestionar los usuarios que pueden acceder al servidor SIP.
- **Extensión de autenticación:** Es la encargada de obtener los vectores de autentificación de la base de datos HSS (dentro de una arquitectura IMS).
- **Extensión S-CSCF/IMS:** Encargada de adaptar el servidor proxy para que funcione dentro de la arquitectura IMS como parte del elemento S-CSCF.

4.2. Arquitectura JSER

Un usuario tiene dos vías de acceso para comunicarse con el JSER:

- Mediante la **interfaz web**. Para poder acceder por esta vía es necesario un cliente HTTP. El acceso al JSER se limita a la gestión de la base de datos interna del programa.
- A través de un **cliente SIP**. Por esta vía el usuario se comunica con el núcleo del JSER. Este núcleo será el encargado de comunicar al usuario con los otros módulos del sistema.

Además de la interacción de los usuarios con el JSER, este servidor se puede comunicar con otros nodos de la arquitectura IMS:

- **módulo de autenticación:** Se comunica con la base de datos HSS para obtener los vectores de autentificación del usuario mediante el protocolo Diameter. Este módulo solo está activo si se requiere una autenticación de los usuarios del JSER.
- **Módulo B2BUA:** Se comunica con otros los otros dos nodos (Filter Criteria y Call Control) del S-CSCF. Este módulo solo tiene sentido si el JSER funciona dentro de una arquitectura IMS.

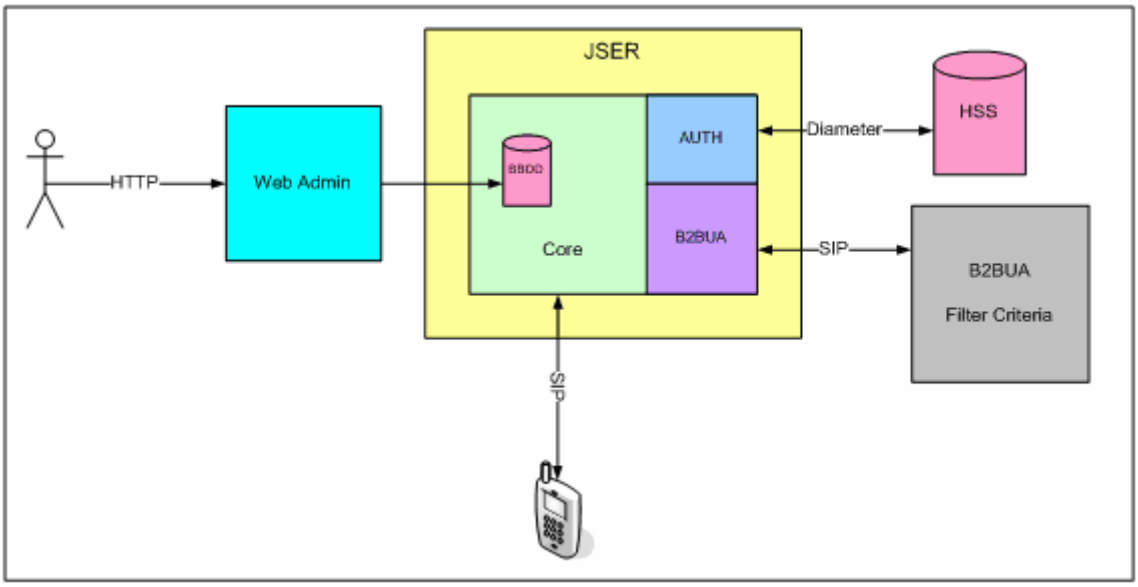

Ilustración 9.Módulos de la arquitectura JSER

4.3. Núcleo del JSER

El núcleo del JSER a su vez se puede dividir en diversos módulos:

- **Proxy/Redirect:** Gestiona la lógica de los mensajes INVITE recibidos por el JSER.
- **Register:** Gestiona la lógica de los mensajes REGISTER recibidos por el JSER.
- **Contenedor SIP:** Permite implementar las lógicas de Proxy, Redirect y Register.
- **Data Access Object (DAO):** Contiene la lógica para el acceso a la base de datos interna del JSER.

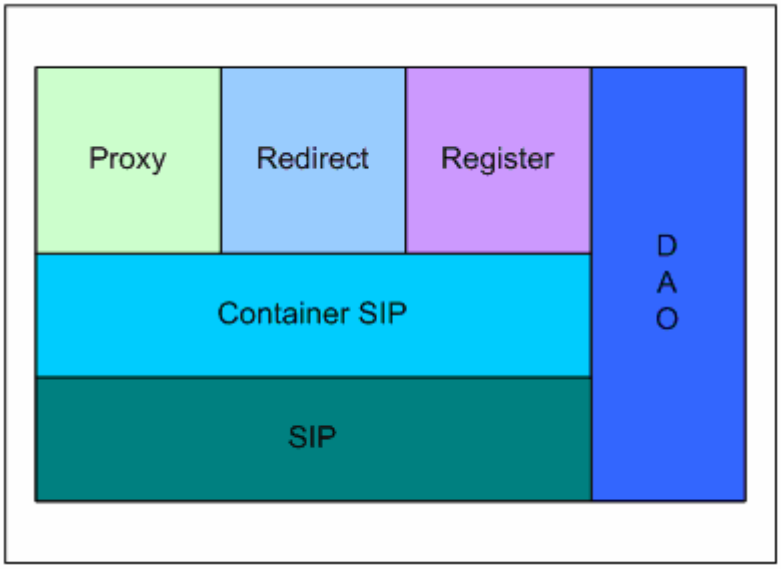

Ilustración 10. Diagrama del núcleo del JSER

4.3.1. Contenedor SIP

Tomcat es una implementación de referencia de J2EE que hace las funciones de servidor de aplicaciones. Tomcat trabaja con páginas JSP (Java Servlet Page) y servlets, por eso también se define como un contenedor de servlets con entorno JSP.

WeSIP **[10]** esta basado en Tomcat y es el nombre comercial del servidor de aplicaciones convergentes de VozTelecom.

WeSIP está diseñado para que pueda actuar como UAC, UAS, B2BUA o proxy dentro de cualquier entorno SIP, permitiendo así contener una amplia gamma de aplicaciones. Por ejemplo, VoIP, compartir archivos, etc. Utilizar al mismo tiempo un HTTP Servlet permite una convergencia entre aplicaciones SIP y HTTP.

En este proyecto usaremos WeSIP como contenedor para sipservlets para poder desarrollar el JSER utilizando las ventajas de los sipservlets.

4.3.1.1. SipServlet

Un servlet es una tecnología de Java que se ejecuta en un servidor Web y permite compilar páginas Web para que sean mostradas a los clientes. Cabe destacar que compila las páginas en tiempo real, cosa que permite recoger información del usuario y crear una página Web concreta para el cliente o cambiar la información de una página frecuentemente.

Al igual que los servlets, sipservlet tiene un funcionamiento orientado a eventos, de manera que se puede definir su comportamiento en función del mensaje SIP recibido. Pero a diferencia de los servlets, sipservlet puede recibir y generar tanto peticiones como respuestas.

4.3.2. Lógica de registro

La lógica de registro es la encargada de autorizar a los usuarios para que puedan usar el servidor proxy. Los mensajes SIP ligados a esta lógica son los REGISTER.

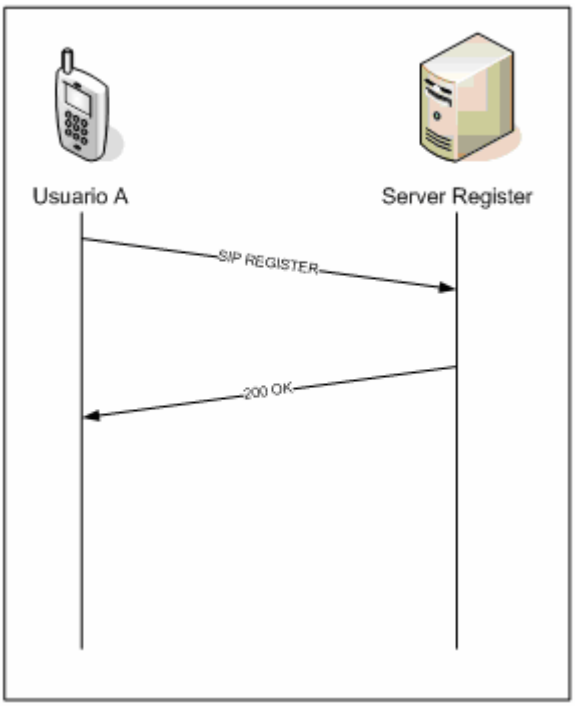

Ilustración 11. Lógica de registro

En la lógica de registro caben destacar dos conceptos diferentes:

- **Registrar:** Es el proceso al que se somete un usuario para darle de alta en el servidor proxy.
- **Autenticar:** Es el proceso para verificar que el usuario es realmente quien dice ser.

Por tanto, el registro es obligatorio para todos los usuarios, pero la autenticación puede variar ya que hay diversos métodos como Digest o AKA para realizar la autenticación, e incluso se puede registrar un usuario sin autenticarlo si así se decide.

4.3.2.1. Registro sin autenticación

La lógica de registro funciona de la siguiente manera:

- Cuando llega un nuevo mensaje REGISTER se comprueba que el usuario que realiza la petición esté en la tabla Subscriber de la base de datos del servidor proxy.
- Si el usuario no existe se le manda un mensaje de error 401 Unauthorized, dando por finalizada la comunicación.
- En el caso que el usuario exista, se le incluye en la tabla Location de la base de datos. Esta tabla contiene a los usuarios que se encuentran conectados e incluye la información necesaria para localizarlos.
- Una vez dado de alta en la tabla Location, se envía al usuario un mensaje de confirmación 200 OK. Desde ese momento el usuario podrá iniciar y recibir llamadas.

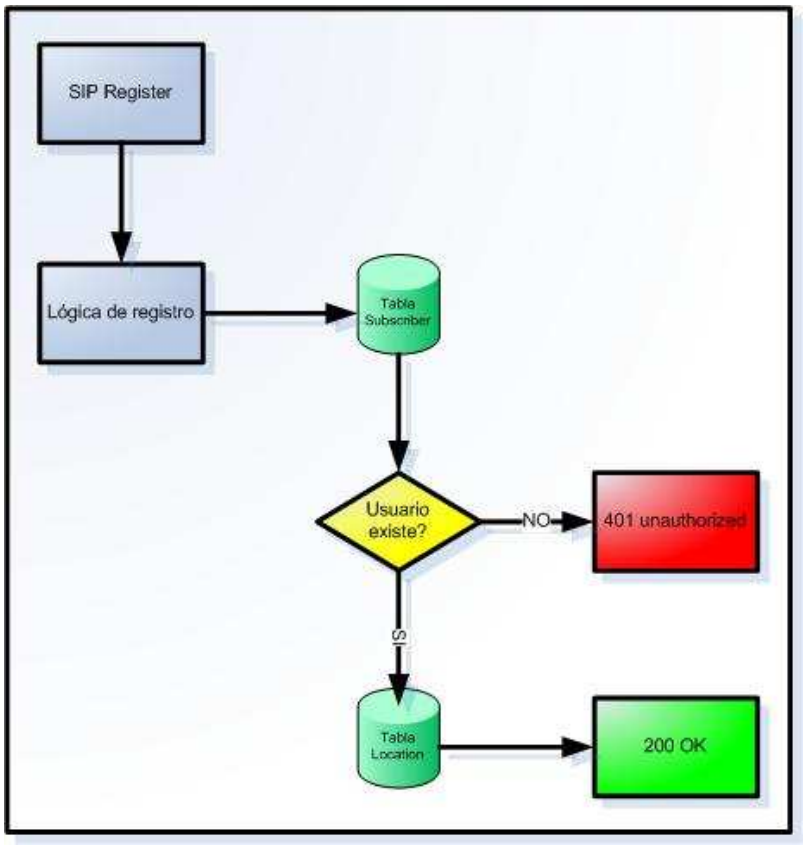

Ilustración 12. Diagrama de la lógica de registro sin autenticación

4.3.2.2. Registro con autenticación usando AKA

Utilizando el método de autenticación AKA **[11]**, el registro de un usuario será de la siguiente forma:

- **1:** Un usuario envía un mensaje de REGISTER hacia el servidor proxy. Después el servidor pide los vectores de autenticación relacionados con este usuario a la base de datos HSS.
- **2:** De entre los vectores de autenticación obtenidos, el servidor proxy selecciona uno.
- **3:** El servidor envía un mensaje de error 401 Unauthorized. Dentro del mensaje se le incluye el método que tiene que utilizar para autenticarse (En este caso el Digest) y los parámetros que tiene que usar en los cálculos (RAND y AUTN).
- **4:** El usuario realiza los cálculos correspondientes con los parámetros que le han dado más su password.
- **5:** El usuario vuelve a enviar un nuevo mensaje de REGISTER, pero en este caso se adjunta la información de autenticación a la que llamaremos RES.
- **6:** Por su parte, el servidor realiza los mismos cálculos que el cliente ya que también conoce la contraseña de ese usuario, este cálculo lo llamaremos XRES. Después comprueba de que RES sea equivalente a XRES.
- **7:** Si es equivalente, lo notifica a la base de datos HSS y al usuario mediante un mensaje de confirmación 200 OK.

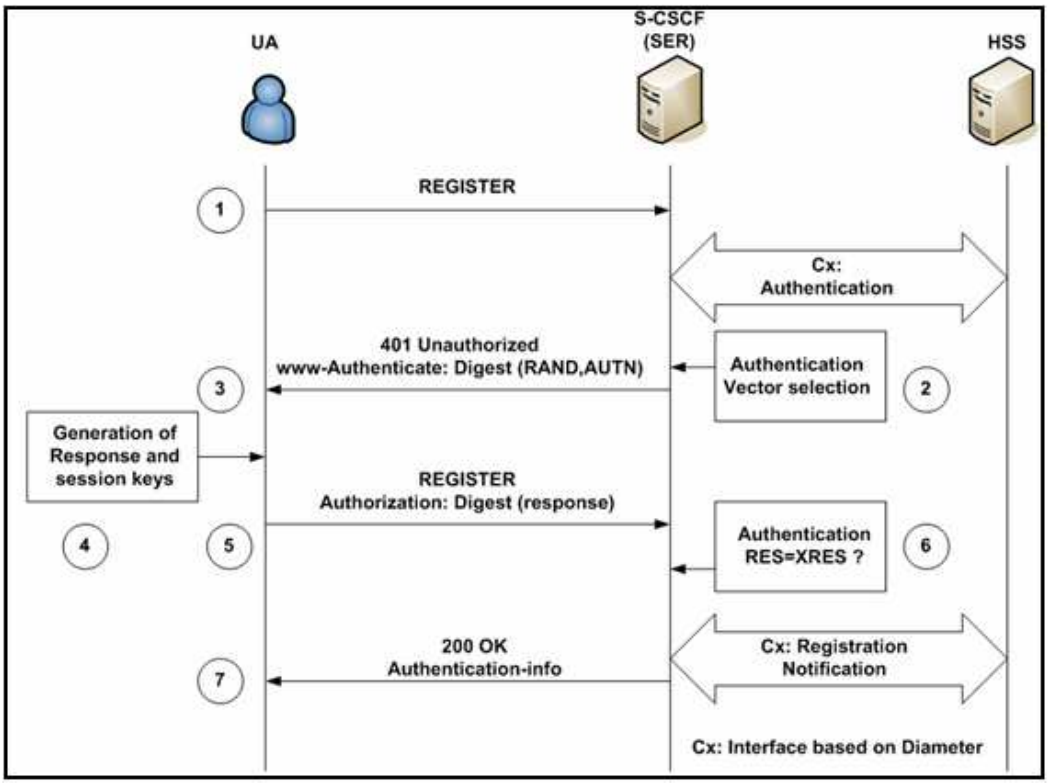

Ilustración 13. Lógica de registro con autenticación AKA

4.3.3. Lógica de proxy/redirect

La lógica de proxy y reenvio es la encargada de reenviar hacia el destino todos los mensajes que lleguen al servidor.

Los nodos de la arquitectura SIP solo sabrán la localización del servidor proxy, así que cualquier mensaje que quieran enviar lo tendrán que hacer pasando por este proxy. Cuando el mensaje llegue al proxy, este será el encargado de localizar en su base de datos al nodo destino y reenviarle el mensaje, o informar al nodo origen que no existe el destino al que intenta llegar.

Esta lógica afecta a todo tipo de mensajes SIP a excepción de los SUBSCRIBE, ya que estos últimos tienen como destino el propio servidor proxy.

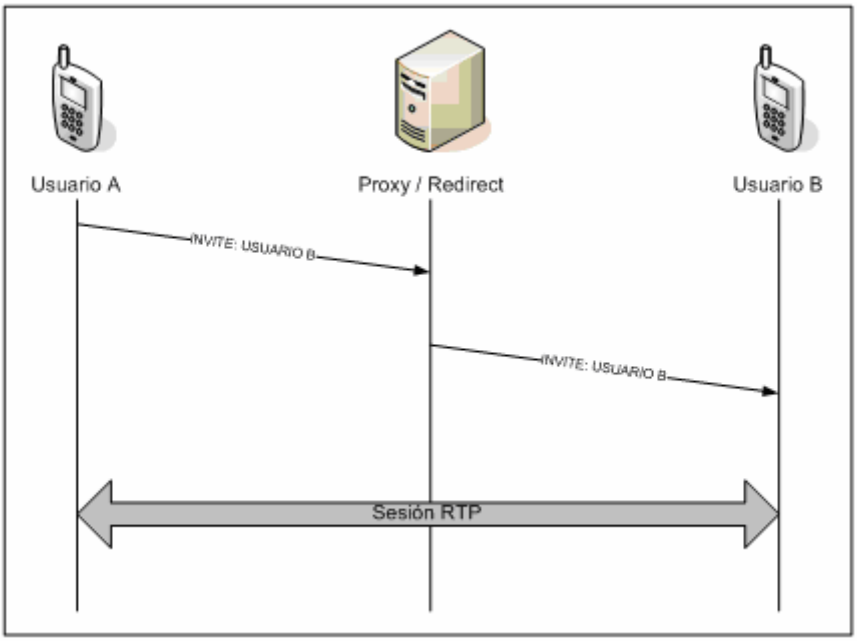

Ilustración 14. Lógica de proxy y reenvio

En la ilustración 14 vemos como el usuario A quiere iniciar una conversación de voz con l usuario B. Para realizar esta comunicación envia un mensaje INVITE con destino al usuario B pero con dirección del proxy, ya que es el único nodo que conoce de la red. El proxy es el encargado de buscar la dirección del usuario B y reenviarle el mensaje. Una vez el usuario B recibe el mensaje, ya se puede realizar el flujo multimedia entre los usuarios A y B para que se puedan comunicar. Este flujo RTP no pasa por el proxy, sino que se realiza directamente entre los usuario A y B. Esto es devido a que el proxy solo gestiona los mensajes SIP, y así se consigue no saturar al proxy haciendole reenviar todo el flujo multimedia hacia sus destinos.

4.3.4. Data Access Object

Data Access Object (DAO) hace referencia al patrón de diseño para el acceso a la base de datos del servidor. En este caso, y siguiendo el patrón DAO se ha usado hibernate.

La base de datos interna del JSER consta de 2 de dos tablas que son usadas para el correcto funcionamiento del servidor, estas dos tablas son:

- **Subscriber:** Contiene los datos y las claves de los usuarios que tienen permiso para utilizar el proxy.
- **Location:** Contiene la lista de los usuarios que están activos en el proxy, así como sus URIs para poder contactar con ellos.

A parte de las tablas para el funcionamiento del JSER, se ha creado una tabla auxiliar llamada **webusers**. Esta tabla tiene como objetivo guardar los usuarios que tengan acceso a la herramienta web JSER Admin que se explicara en el apartado 4.4.

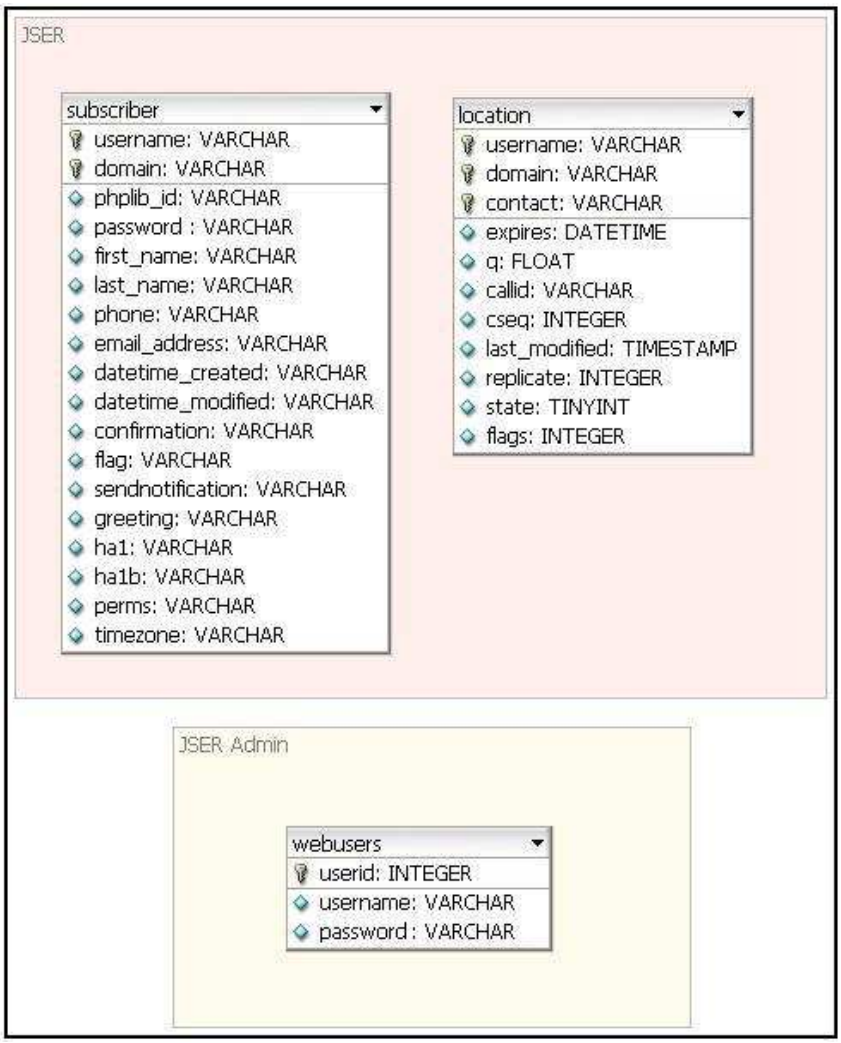

Ilustración 15. Estructura de las tablas usadas en el proyecto

4.3.4.1. Hibernate

La comunicación entre la base de datos MySQL y JSER se realiza mediante Hibernate **[12]**. Hibernate es un servicio de persistencia entre objetos/relaciones y consultas para Java. Hibernate es una capa de persistencia objeto/relacional y un generador de sentencias sql. Te permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos. De una manera rápida y optimizada se puede generar bases de datos en cualquiera de los entornos soportados : Oracle, DB2, MySql, etc..

A nivel de aplicación, la base de datos serán objetos Java que podrán ser leídos y modificados.

La transformación de los campos de la base de datos a objetos Java se realiza mediante ficheros XML donde son mapeados cada campo, es decir, en estos ficheros se indica para cada campo de la base de datos su equivalencia en Java. La configuración de hibernate se realiza mediante un fichero XML donde se indica la configuración de la base de datos y la localización de los archivos de mapeado de la base de datos.

La conexión entre Java y la base de datos se realiza mediante un JDBC, que es una colección de interfaces Java y métodos de gestión de manejadores de conexión hacia cada modelo específico de base de datos. Por tanto hay un JDBC para cada tipo de base de datos. Hibernate da soporte para cualquier tipo de base de datos sin tener que modificar el código programado, solo se tendría que asociar el conector JDBC apropiado.

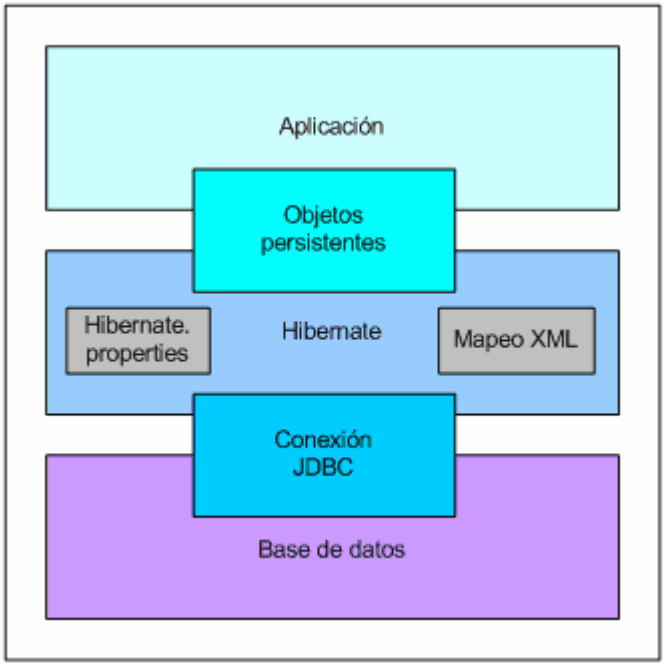

Ilustración 16. Estructura de Hibernate

4.4. Administración web

El motivo de esta administración vía web es ofrecer una interfaz atractiva y fácil de utilizar que para poder gestionar los usuarios con acceso al servidor proxy. A parte, al ser una interficie web, permite una gestión remota.

La programación de esta herramienta se ha realizado con Struts (ver el siguiente apartado 4.4.1.) y para la comunicación entre la base de datos y la web, se ha aprovechado las conexiones de hibernate realizadas para la comunicación entre el JSER y la base de datos.

Ilustración 17. Menú principal de JSER Admin

La interficie web ofrece las siguientes posibilidades:

- **Registrarse:** Antes de empezar a utilizar la web, se comprueba que el usuario tiene permisos para administrar la base de datos, para ello se le pide un nombre de usuario y una contraseña.
- **Buscar usuarios de JSER:** Permite buscar usuarios del JSER por nombre de usuario, dominio o por identificador de usuario permitiendo modificar o borrar estos usuarios.
- **Listar usuarios de JSER:** Muestra todos los usuarios del JSER y además permite modificar o borrarlos.

- **Añadir usuarios de JSER:** Permite dar de alta nuevos usuarios que puedan acceder al proxy JSER.
- **Borrar usuarios de JSER:** Permite borrar a usuarios que tienen permiso para acceder al JSER.
- **Modificar usuarios de JSER:** Permite gestionar los usuarios del JSER para poder modificar sus valores.
- **Mostrar usuarios online de JSER:** Muestra los usuarios de JSER que se encuentran activos en ese instante.
- **Añadir web users:** Esta opción ofrece la posibilidad de añadir usuarios que puedan acceder a esta herramienta web.
- **Listar usuarios de la web:** Muestra todos los usuarios que tienen la posibilidad de acceder a la herramienta web y además permite modificar o borrar este tipo de usuarios.
- **Borrar usuarios de la web:** Permite poder borrar a usuarios que tienen permiso para acceder a la herramienta web.
- **Modificar usuarios de la web:** Permite gestionar los usuarios de la web para poder modificar sus valores
- **Logout:** Sale de la herramienta de gestión.

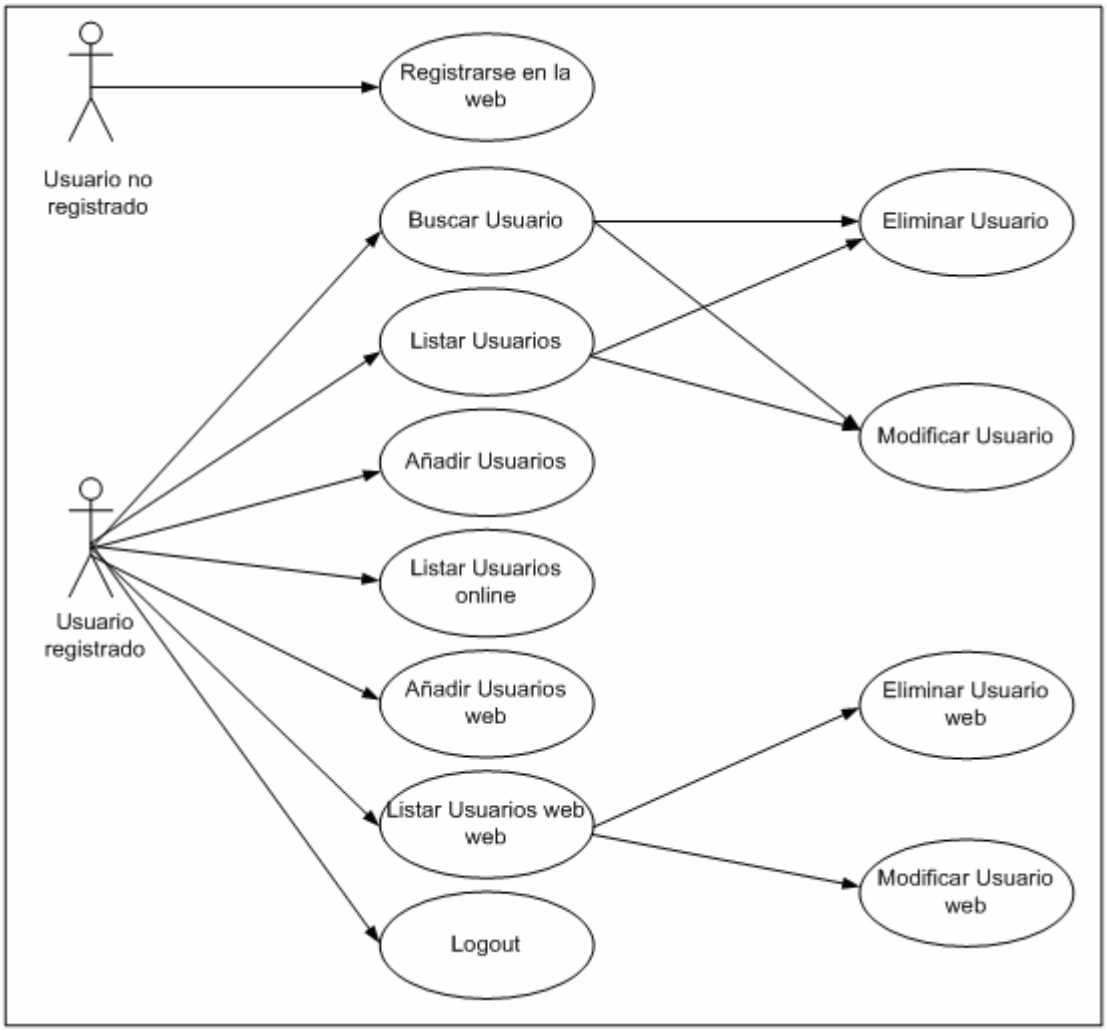

Ilustración 18. Diagrama de casos de uso de la web

4.4.1. Struts

Struts **[13]** es un framework para aplicaciones web java que implementa el modelo MVC (Modelo Vista Controlador).

El paradigma MVC consiste en dividir las aplicaciones en tres partes:

- Controlador
- Modelo
- Vistas

El **controlador** es el encargado de redirigir o asignar una aplicación (un modelo) a cada petición; el controlador debe poseer de algún modo, un "mapa" de correspondencias entre peticiones y respuestas (aplicaciones o modelo) que se les asignan.

El **modelo** seria la aplicación que responde a una petición, es la lógica de negocio a fin de cuentas. Una vez realizadas las operaciones necesarias el flujo vuelve al controlador y este devuelve los resultados a una **vista** asignada.

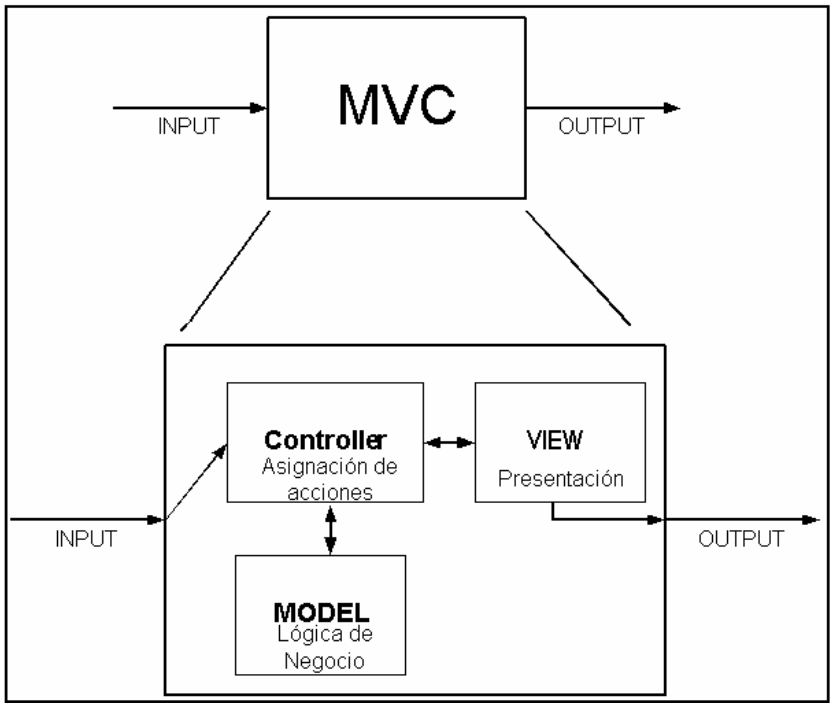

Ilustración 19. Esquema del modelo MVC

Con este sistema obtenemos una separación total entre lógica de negocio y presentación. A esto se le pueden aplicar opciones como el multilenguaje, distintos diseños de presentación,.. etc sin alterar la lógica de negocio.

La separación de capas como presentación, lógica de negocio, acceso a datos es fundamental para el desarrollo de arquitecturas consistentes, reutilizables y más fácilmente mantenible, lo que al final resulta en un ahorro de tiempo en desarrollo en posteriores proyectos.

4.5. JSER dentro de IMS

Dentro de la arquitectura IMS, el JSER actúa como parte del S-CSCF (Ver apartado ...). Este elemento está compuesto por dos subsistemas:

- **JSER:** Encargado de registrar a los usuarios y de enrutar los mensajes hacia su destino y desarrollado en este proyecto.
- **B2BUA:** Entidad lógica capaz de recibir y procesar mensajes INVITE, MESSAGE y SUBSCRIBE. Determina como estos mensajes deben ser contestados y es capaz de crear nuevas conexiones. A diferencia del proxy SIP, el B2BUA mantiene el estado completo de las llamadas y es capaz de redirigir flujos de media. El B2BUA ha sido desarrollado en el TFC de *Javi Castillo* titulado **Desarrollo de B2BUA – FilterCriteria para S-CSCF en IMS**. A su vez, el B2BUA se puede dividir en el Filtro IFC y en el Call Control.

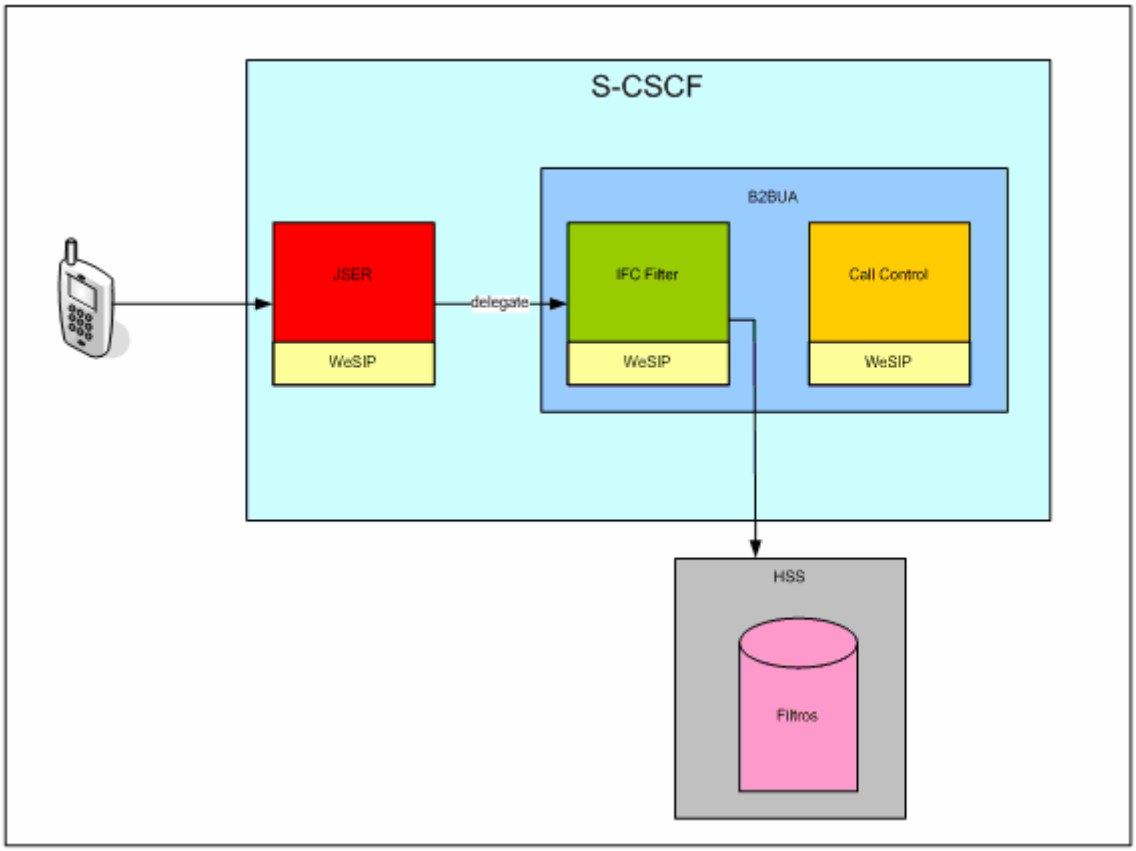

Ilustración 20. Estructura del S-CSCF

4.5.1. Filtro IFC

El Filtro IFC es el encargado de determinar si el usuario que intenta llamar tiene permisos para realizar la llamada. Para ello aplica un filtrado a través de un fichero XML. Cada usuario tiene su propio filtro y se guardan en la base de datos HSS. Los mensajes que pasan por el filtro son INVITE, MESSAGE y SUBSCRIBER, ya que son los que inician diálogo.

El filtro IFC funciona sobre WeSIP utilizando como contenedor de SipServlets. Esto es debido a que el Filtro tiene que acceder a los parámetros de los mensajes y por tanto necesita las funcionalidades que ofrece los SipServlets.

4.5.2. Call Control

El Call Control actúa de proxy para los flujos RTP. Es decir, es capaz de crear, monitorizar, controlar, manipular o cerrar sesiones RTP facilitando así, la comunicación entre dos o más participantes y otros elementos de la red.

Para programar el Call Control se utiliza la implementación JCC de la empresa VozTelecom, denominada jaicc así como el jaiccSipServlet encargado de interpretar los mensajes SIP. Esta implementación está también recogida dentro del servidor WeSIP.

4.5.2.1. Java Call Control

Java Call Control (JCC) **[14]** es una interfaz en la que se pretende abstraer cualquier protocolo de señalización. En esta interfaz se definen todos los procesos propios de un protocolo de señalización como serian el crear o cerrar sesiones o realizar cambios en dichas sesiones.

Dentro del JCC se definen cuatro objetos que son necesarios para cualquier implementación:

- Provider: Es el encargado de crear y monitorizar las llamadas.
- Address: Terminal lógico (en SIP la URL del terminal)
- Call: Representa una llamada pudiendo mantener dos o más terminales unidos.
- Connection: Relaciona un Call con una Address. Es utilizado para saber en detalle el estado de cada participante en la llamada.

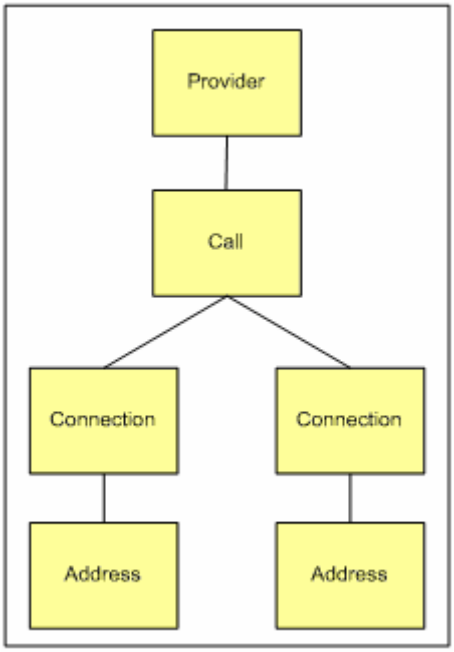

Ilustración 21. Relación básica entre los objetos de JCC

4.5.3. Escenario planteado: Servicio de anuncios para llamadas telefónicas

Para comprobar el funcionamiento del S-CSCF se decidió crear un servicio de valor añadido que consiste en insertar publicidad a las llamadas telefónicas. El usuario que realiza la llamada primero se le redireccionará a un servidor de media el cual tendrá las cuñas publicitarias. Una vez haya escuchado el anuncio, se le redirigirá directamente al usuario que intenta llamar.

Con este sistema publicitario las operadoras tendrían unos ingresos extra y así podrían ofrecer tarifas más atractivas a los usuarios.

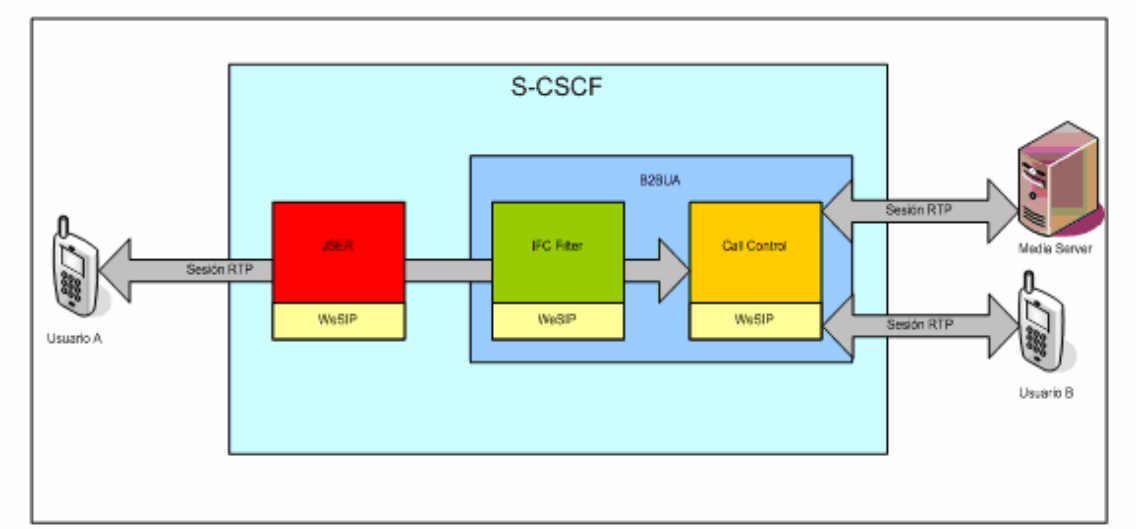

Ilustración 22. Escenario planteado

Como se ve en la ilustración 22, el Call Control será el encargado de conmutar entre el servidor de Media donde estarán las cuñas publicitarias y el usuario final de la llamada.

4.5.4. Lógica de registro en IMS

La lógica de registro dentro del sistema IMS es la que se puede observar en la ilustración 23. La diferencia con la lógica mostrada en el apartado 4.3.2.2. radica que los mensajes tienen que pasar por el P-CSCF y el I-CSCF. En nuestro escenario estos dos elementos no han sido implementados así que el usuario accede directamente al S-CSCF.

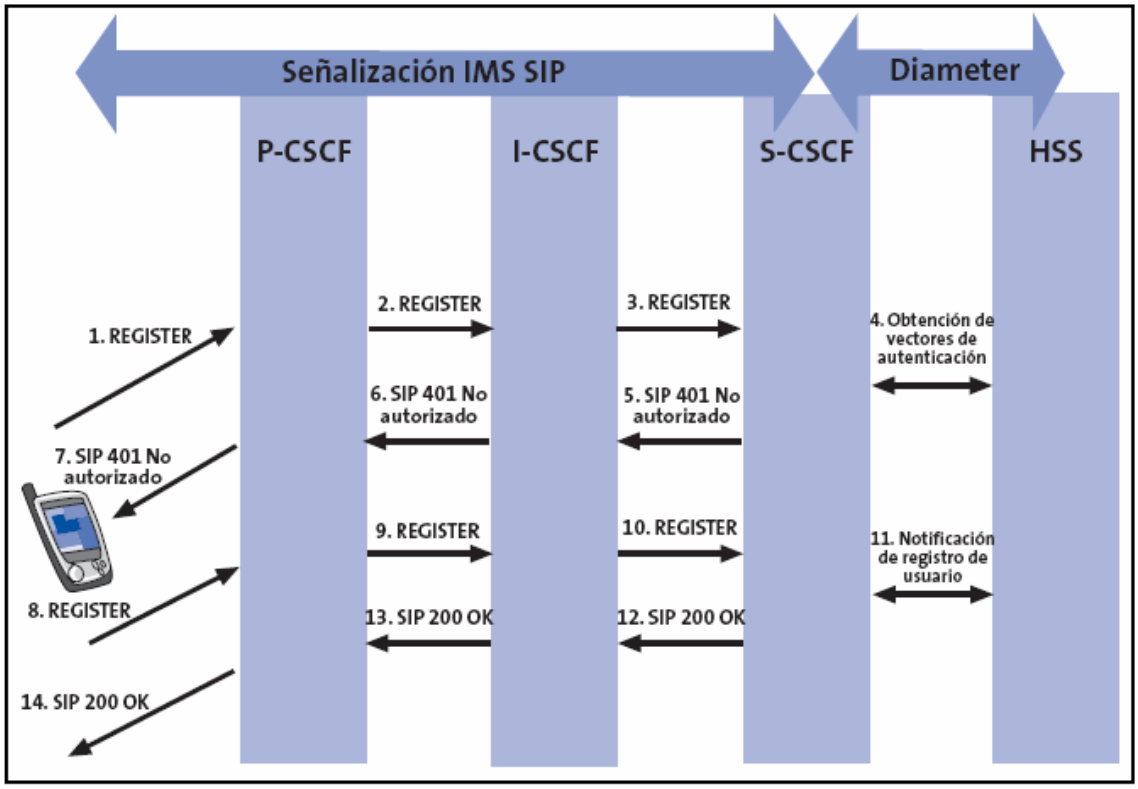

Ilustración 23. Lógica de registro en el sistema IMS

4.5.5. Encaminamiento de mensajes dentro de IMS

En la ilustración 24 se muestra la lógica de intercambio de mensajes seguida para IMS. Para simplificar el gráfico solo se han mostrado los mensajes invite y las sesiones RTP.

En este esquema el único nodo que sabe la localización de todos los otros nodos es el proxy JSER y por tanto todos los mensajes tendrán que pasar por él. El inconveniente es que todos los demás nodos tienen que estar registrados en el proxy, incluyendo el Filtro IFC, el Call Control y el Media Server.

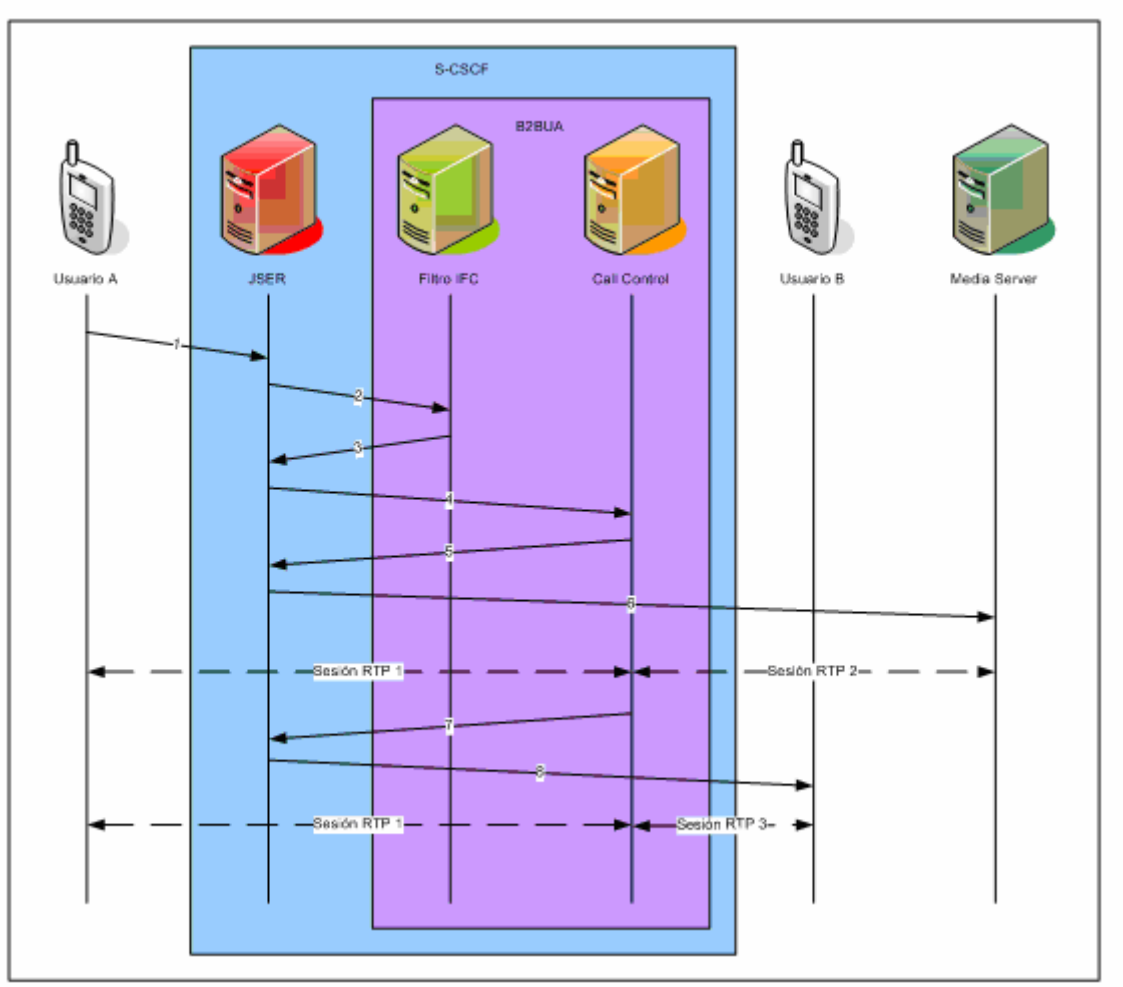

Ilustración 24. Mensajes invite intercambiados en el escenario S-CSCF

Mensaje 1: En este escenario, una vez que el usuario A y B se han registrado, el usuario A decide llamar al usuario B. Para ello el usuario A envía un invite hacia el único nodo que conoce del sistema, el proxy JSER.

Mensaje 2: El proxy, al recibir la petición de llamada y comprobar que el usuario al que intenta llamar esta disponible, delega en el filtro la tarea de decidir si el usuario A tiene permiso para realizar la llamada.

Mensaje 3: Una vez comprobado los permisos del usuario A aplicando el filtro del IFC asociado a este usuario, el IFC decide que puede establecer la llamada a través del Call Control. Para Indicar esto, el filtro reenvía el invite anterior indicando que el destino es el Call control.

Mensaje 4: El proxy al recibir la respuesta del IFC sabe que tiene que establecer una conexión entre el usuario A y el Call Control, así que reenvía el invite anterior hacia el Call Control. Después de este invite se establecería la conexión RTP 1 entre el usuario A y el Call Control.

Mensaje 5: El Call Control realiza un invite hacia el Media Server. Como el invite lo ha realizado el Call Control, el JSER sabe que no tiene que pasar por el filtro IFC así que reenvía el invite hacia su destino.

Mensaje 6: El Media Server recibe el invite y se establece la conexión RTP 2. El Call Control al tener las dos sesiones RTP abiertas conectará los dos flujos. De esta manera se conectará el usuario A con el Media Server donde estarán las cuñas publicitarias.

Mensaje 7: Cuando ha finalizado la cuña publicitaria, la conexión RTP 2 se cierra, pero no así la sesión RTP 1, por lo que el usuario A sigue con la conexión abierta. Así que el Call Control crea un invite para llamar al usuario B (el verdadero destino de la llamada).

Mensaje 8: El proxy al recibir el invite del Call Control reenvía el mensaje hacia su destino, por lo que es recibido por el usuario B y se crea la sesión RTP 3 entre el Call Control y el usuario B. Como en el caso anterior, El Call Control conectará este flujo RTP con el que sigue abierto de la sesión RTP 1, por lo que el usuario A y B se podrán comunicar.

CAPÍTULO 5. CONCLUSIONES

5.1. Impacto medio ambiental

La filosofía de IMS es que la separación de los elementos como el I-CSCF, P-CSCF o S-CSCF es siempre una separación lógica y no física. Esto significa que el estándar IMS no impone utilizar un servidor por cada elemento por lo que se podría implementar un único servidor que hiciera todas las funciones mencionadas en este trabajo.

Desde el punto de vista medio ambiental, este tema es crucial, porque el derroche energético y de materiales que representaría utilizar un servidor para cada elemento sería muy elevado.

Aplicando esta filosofía al proyecto, los tres elementos utilizados del S-CSCF: proxy JSER, Filtro IFC y Call Control, pueden funcionar en una misma máquina siempre y cuando se le asignen puertos diferentes a cada elemento.

Otro punto de vista medio ambiental interesante son los softphones (software que permite realizar las funciones de un teléfono SIP). Con estos softphones no es necesario tener teléfonos físicos, con el consiguiente ahorro de material y energético debido a su fabricación.

5.2. Conclusiones sobre los objetivos

JSER cumple con las funciones básicas que ha de tener todo proxy SIP que son las de register y las de proxy/reenvío de mensajes. Aunque en un principio, como todo proyecto, se aspiraba a más objetivos como implementar algún mecanismo de autentificación para el register o crear algún servicio de valor añadido en el proxy como un servidor de presencia.

Comparando el SER (servidor proxy SIP comercial) con el JSER, gana el primero en eficiencia, debido a que el propio lenguaje de programación C aporta una respuesta más rápida que un programa en Java, ya que este último es un lenguaje interpretado. Además el SER ha sido programado por un equipo de programadores profesionales de la empresa iptel.org por lo que está más optimizado que el JSER.

Aún así JSER aporta unas ventajas que no tiene el SER: Es más sencillo e intuitivo de programar, es independiente del sistema operativo y del hardware (gracias al lenguaje Java) y aporta una herramienta de gestión web (JSER Admin) que mejora a la existente en el SER llamada “serctl” y basada en la línea de comandos.

5.3. Líneas futuras

El proyecto del JSER se puede seguir mejorando creciendo horizontalmente, como por ejemplo implementar alguno de los módulos existentes para el SER o creando módulos nuevos. Si no se quisiera avanzar por esta vía, se podría seguir diseñando algún otro módulo de la arquitectura IMS.

Otra vía es crecer verticalmente creando nuevas aplicaciones con las infraestructuras creadas.

5.4. Conclusiones personales

Personalmente ya había participado en proyectos de grande envergadura gracias al bloque optativo de telemática titulado “Intensificación en Sistemas Telemáticos” donde éramos cinco componentes y nos teníamos que coordinar el tiempo para llegar a los objetivos. Este hecho hacia que cada uno se acabara especializando en un solo tema y se confiara ciegamente en el trabajo de los otros compañeros perdiendo a veces la visión global del proyecto.

En cambio, este hecho no ha pasado en este proyecto, ya que al realizarlo en solitario he tenido que hacer frente a todos los puntos complicados del proyecto. He trabajado con diversas tecnologías como Struts para la creación de páginas web, hibernate para el acceso a base de datos y los SipServlets para el desarrollo del JSER.

Además de las tecnologías que he aprendido y usado en este proyecto, he trabajado para integrarlas con el protocolo SIP y he tenido una visión global de las tareas del JSER dentro de una gran estructura como lo es el IMS.

CAPÍTULO 6. BIBLIOGRAFÍA

[1] VoIP [en línea]: *Recursos VoIP- Voz sobre IP: Telefonía IP*. [Última consulta: 7 de julio de 2006] Disponible en: <http://www.recursosvoip.com/>

[2] Especificaciones para H.323 [en línea]: *H.323 Standards* [Última Consulta: 7 de julio de 2006] Disponible en: <http://www.packetizer.com/voip/h323/standards.html>

[3] Especificaciones para SIP, RFC 3261 [en línea]: *Session Initiation Protocol* [Última Consulta: 7 de julio de 2006] Disponible en: <http://www.faqs.org/rfcs/rfc3261.html>

[4] Especificaciones para SDP, RFC 2327 [en línea]: *SDP: Session Description Protocol*. [Última Consulta: 7 de julio de 2006] Disponible en: <http://www.ietf.org/rfc/rfc2327.txt?number=2327>

[5] Especificaciones para RTP, RFC 3550 [en línea]: *RTP: A Transport Protocol for Real-Time Applications*. [Última Consulta: 7 de julio de 2006] Disponible en: <http://www.faqs.org/rfcs/rfc3550.html>

[6] Especificaciones para RTCP, RFC 3550 [en línea]: *RTP Profile for Audio and Video Conferences with Minimal Control*. [Última Consulta: 7 de julio de 2006] Disponible en: <http://www.faqs.org/rfcs/rfc3550.html>

[7] información sobre IMS [en línea]: *Evolución al domino IMS.* [Última Consulta: 7 de julio de 2006] Disponible en: <http://www.telefonica.es/sociedaddelainformacion/pdf/publicaciones/movilidad/capitulo_12.pdf>

[8] información sobre SER [en línea]: *SIP Express Router – Developers Guide.* [Última Consulta: 7 de julio de 2006] Disponible en: <http://www.iptel.org/ser/doc/serdev/serdev.pdf>

[9] información sobre OPENSER [en línea]: *OpenSER Project.* [Última Consulta: 7 de julio de 2006] Disponible en: <http://www.openser.org/>

[10] información sobre WeSIP [en línea]: *WeSIP.* [Última Consulta: 7 de julio de 2006] Disponible en: <http://ssl.voztele.com:28080/wesip.htm>

[11] información sobre AKA [en línea]: *Security Analysis and Enhancements of 3GPP Authentication and Key Agreement*

Protocol. Última Consulta: 7 de julio de 2006] Disponible en: <http://www.fang.ece.ufl.edu/mypaper/tw05zhang.pdf>

[12] información sobre Hibernate [en línea]: *Hibernate*. Última Consulta: 7 de julio de 2006] Disponible en: <http://www.hibernate.org/>

[13] información sobre struts [en línea]: *Struts Framework.* Última Consulta: 7 de julio de 2006] Disponible en: <http://struts.apache.org/>

[14] información sobre JCC [en línea]: *Java Call Control.* Última Consulta: 7 de julio de 2006] Disponible en: <http://www.argreenhouse.com/papers/rjain/chapter_jccpaper13.pdf>

ANEXOS

A. Acrónimos

H.323: Recomendación del ITU-T (International Telecommunication Union), que define los protocolos para proveer sesiones de comunicación audiovisual en cualquier paquete de la red.

HTML: El HTML, acrónimo de Hypertext Markup Language (lenguaje de etiquetaje de hipertexto), es un lenguaje de marcas diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas Web.

IETF: Internet Engineering Task Force, en castellano Grupo de Trabajo en Ingeniería de Internet. Es una organización internacional abierta de normalización, que tiene como objetivos el contribuir a la ingeniería de Internet, actuando en diversas áreas, tales como transporte, encaminamiento, seguridad. Fue creada en EE.UU. en 1986.

ITU: La Unión Internacional de Telecomunicaciones. Es el organismo especializado de las Naciones Unidas encargado de regular las telecomunicaciones, a nivel internacional, entre las distintas administraciones y empresas operadoras.

JCC: Siglas de Java Call Control. JCC es una API de Java para crear, monitorizar, controlar, manipular y colgar sesione en un entorno convergente entre PSTN y conmutación de paquetes (Internet).

JDBC: acrónimo de Java Database Connectivity, un API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java independientemente del sistema de operación donde se ejecute o de la base de datos a la cual se accede utilizando el dialecto SQL del modelo de base de datos que se utilice.

RTCP: Siglas RTP Control Protocol (Protocolo de control para RTP). La función primaria es proporcionar información al origen de la calidad de servicio de la distribución de información.

RTP: Siglas de Real-time Transport Protocol (Protocolo de Transporte de tiempo Real). Es un protocolo de nivel de transporte utilizado para la transmisión de información en tiempo real como por ejemplo audio y video en una video-conferencia.

SDP: Siglas de Session Description Protocol (Protocolo de Descripción de Sesión). Está diseñado para describir las sesiones multimedia con el bjetivo de anunciar sesiones, invitación de sesión y otras formas de inicio de una sesión de multimedia.

SIP: Siglas de Session Initiation Protocol (Protocolo de Inicio de Sesión). Protocolo de señalización que se utiliza para iniciar sesiones multimedia interactivas entre usuarios de redes IP.

SS7: Sistema de señalización por canal común nº 7. Este sistema es el utilizado por la red telefónica para la señalización de las llamadas.

UA: User Agent, es la entidad básica de la arquitectura SIP capaz de enviar y recibir mensajes.

VoIP: Siglas de Voz sobre IP. Definido en 1996 por la ITU. Define un sistema de enrutamiento de conversaciones de voz mediante paquetes basados en IP por la red de Internet.

WeSip: WeSIP es un contenedor convergente de HTTP y SIP Servlets, desarrollado por VozTelecom.