UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

ARQUITECTURA DE APLICACIONES PARA REDES CONVERGENTES

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELECTRICISTA

DIEGO EMILIO SAAVEDRA RENDIC

PROFESOR GUÍA:
JORGE SANDOVAL ARENAS

MIEMBROS DE LA COMISIÓN:
NÉSTOR BECERRA YOMA
ALBERTO CASTRO ROJAS

SANTIAGO DE CHILE
OCTUBRE 2008

RESUMEN DE LA MEMORIA
PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELECTRICISTA
POR: DIEGO SAAVEDRA RENDIC
FECHA: 13/10/2008
PROF. GUÍA: Sr. JORGE SANDOVAL A.

ARQUITECTURA DE APLICACIONES PARA REDES CONVERGENTES

Las redes de telecomunicaciones presentan elementos y funcionalidades comunes, las cuales en la actualidad tienden a converger hacia un componente común. Esta infraestructura colectiva permite proveer servicios de manera transversal a través de la red, sin importar la tecnología de acceso utilizada, ya sea telefonía móvil, telefonía fija o tradicional, redes de datos como Internet, u otras. En el caso de la telefonía móvil, la convergencia viene dada por la recomendación del grupo 3GPP (3rd Generation Partnership Group), la cual considera un núcleo de red basado en IP denominado IMS (IP Multimedia Subsystem).

El objetivo general del presente trabajo de título es el diseño e implementación de una plataforma para la provisión de aplicaciones multimedia en redes convergentes, basada en software de código abierto, que permita el desarrollo fácil de servicios de valor agregado y se integre a las redes desplegadas en la actualidad. Además, se busca comunicar e integrar la arquitectura con un núcleo IMS, estudiando la conexión a la plataforma mediante terminales de distinta tecnología de acceso.

En este trabajo se evalúan distintas alternativas de implementación para la plataforma. Las características de dichas opciones son comparadas a nivel de funcionalidades, eligiéndose OpenSER como el software a utilizar para desplegar la arquitectura. Variadas aplicaciones se implementan en el sistema, incluyendo la capacidad para realizar llamadas de voz tradicionales, el servicio de VoiceMail, un servidor de presencia, mensajería instantánea, Click-to-Dial, IPTV y el envío de SMS vía web. El siguiente paso consiste en integrar la arquitectura a un núcleo IMS, con tal de probar la interacción con las aplicaciones. Por último, se define y ejecuta un plan de pruebas, con el fin de certificar el funcionamiento de la plataforma de acuerdo a los estándares y garantizar su interoperabilidad con otros sistemas relevantes.

Se concluye que es posible desplegar una arquitectura básica de servicios basada en software de código libre, lo que permite reducir significativamente los costos de implementación de una plataforma de aplicaciones robusta y escalable. Se identifican también distintas interpretaciones de los protocolos, tanto en el núcleo de red como en los clientes, cobrando especial importancia el ajuste a los estándares para asegurar la compatibilidad entre plataformas y terminales de usuario. Por último, se presentan recomendaciones de trabajo a futuro basadas en los resultados de este proyecto.

A mis seres queridos.

"Cuando el objetivo te parezca difícil, no cambies de objetivo;
busca un nuevo camino para llegar a él."

Confucio

Índice

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

En este primer capítulo se introduce al lector en el tema del trabajo de título, detallando los antecedentes que motivan el desarrollo de la labor, los objetivos que se han planteado y el alcance del documento. Además, se presenta una descripción de la información contenida en este texto.

1.1 Motivación

En una red de telecomunicaciones, se ofrecen variados servicios y aplicaciones que son ejecutados en el terminal del usuario final. Se suele distinguir a las redes en función del acceso a éstas; por ejemplo, existen las redes de telefonía móvil, telefonía fija o tradicional (PSTN), voz sobre IP, redes de datos IP en general y muchas otras. Se puede pensar que dichas redes son totalmente independientes o con poca interrelación, pero actualmente la tendencia apunta a integrarlas en una sola red convergente, con un core (núcleo de red) común. Un ejemplo de este tipo de red convergente es la red de telefonía móvil 3G, que actualmente se encuentra en una etapa de desarrollo y puesta en marcha en diversos países, incluyendo Chile. En particular, el modelo desarrollado por el grupo 3GPP (3rd Generation Partnership Project) considera un core IMS (IP Multimedia Subsystem) basado en IP. La idea es alcanzar la ubicuidad de servicios y aplicaciones a través de la red, sin importar el tipo de acceso a ella, para lo cual se hace necesario una arquitectura de aplicaciones que se integre al core IMS y logre la transversalidad en la provisión de éstas.

Por otro lado, el desarrollo de aplicaciones en telecomunicaciones está supeditado al conocimiento de la red subyacente y sus protocolos, lo que excluye a muchos programadores que no tienen dichas nociones. Otro fin es poder extender la posibilidad de desarrollo de aplicaciones a tales personas, lo que es posible de obtener mediante un componente adecuado en la arquitectura que acerque y traduzca su lenguaje al de las telecomunicaciones.

Este proyecto se enmarca en el trabajo de la comunidad PlaySIP, grupo de investigación que se dedica, entre otros temas, al análisis y desarrollo de plataformas IMS y P2PSIP (aplicaciones peer-to-peer basadas en el protocolo SIP). Además de alumnos memoristas, el grupo está conformado por profesionales de diferentes empresas del rubro telecomunicaciones, lo que muestra el interés de la industria por estar a la vanguardia en el desarrollo de estos temas y permite la aplicación de los resultados de los proyectos en implementaciones comerciales.

1.2 Objetivos

A continuación se presentan los objetivos del presente trabajo de memoria.

1.2.1 General

Como objetivo general del proyecto se propone lo siguiente:

- Diseñar, implementar y probar una arquitectura de aplicaciones para redes convergentes, que se integre a las redes desplegadas actualmente y que permita un desarrollo fácil de servicios de valor agregado.

1.2.2 Específicos

Los objetivos específicos del trabajo consideran los siguientes aspectos:

- Comunicar e integrar la arquitectura con la capa de control de un core IMS.
- Analizar el funcionamiento de aplicaciones específicas sobre la plataforma, mediante un plan de pruebas (Test Plan).
- Estudiar la conexión a la arquitectura de aplicaciones, haciendo uso de terminales de distinta tecnología de acceso a la red.
- Aprovechar las potencialidades del protocolo SIP, tanto en el manejo de sesiones multimedia como en la provisión de servicios.
- Hacer uso de software y herramientas de código libre en la fase de implementación de la plataforma.

1.3 Alcance

Este trabajo está orientado principalmente a estudiantes, académicos y profesionales relacionados con el ámbito de las telecomunicaciones, dado su carácter técnico. La labor desarrollada también puede servir al público general interesado en conocer más sobre las redes convergentes y su desarrollo, a nivel introductorio.

1.4 Descripción de Contenidos

El Capítulo 1 destaca las motivaciones del presente proyecto y lo que se pretende lograr con la labor desarrollada, tanto generalmente como de forma específica, además de indicar el grupo objetivo al cual va dirigido el texto.

El Capítulo 2 corresponde a un marco teórico que introduce al lector a la problemática del proyecto desarrollado, entregándole las bases técnicas necesarias para comprender el tema y posterior implementación.

En el Capítulo 3 se detalla el desarrollo del trabajo en sí, presentando la metodología de trabajo y los resultados específicos obtenidos.

El Capítulo 4 representa la discusión de los resultados obtenidos en el Capítulo 3, planteando soluciones a los problemas que surgieron y recomendando desarrollos a futuro.

Por último, el Capítulo 5 reúne las conclusiones que se obtuvieron en el transcurso del trabajo. Se incluyen además las referencias, un glosario y los anexos respectivos.

Capítulo 2

Arquitecturas de Aplicaciones

El presente capítulo representa un marco teórico respecto a la evolución de las plataformas de servicios, mencionando algunas de ellas que se utilizan en la actualidad y detallando su implementación.

2.1 Evolución en la Provisión de Servicios

Las Redes Inteligentes [1] (INs, por sus siglas en inglés) surgieron a principio de los años 90 debido a la necesidad de desligar el desarrollo de servicios del desarrollo de los switches digitales, dadas sus limitaciones en ese aspecto. Así, el modelo de Red Inteligente separó físicamente las funciones de control de la llamada y las funciones de servicios de valor agregado, tal como muestra la siguiente figura:

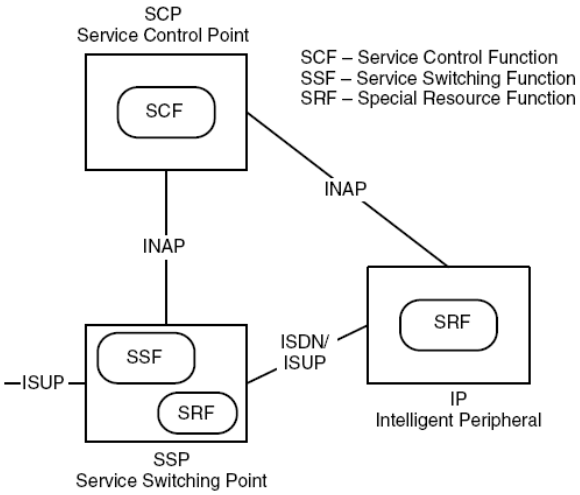

Figura 2-1 [1]: Red Inteligente

El SSP (Service Switching Point) es un switch digital con funcionalidades adicionales, SSF (Service Switching Function) y SRF (Special Resource Function), que le permiten hacer uso de los servicios de una plataforma de aplicaciones externa durante ciertos momentos del procesamiento de la llamada. Éste se comunica mediante el protocolo INAP con el SCP (Service Control Point), el cual a través de su funcionalidad SCF (Service Control Function) le indica al SSP cómo proceder en el establecimiento de la llamada. Cabe destacar que la SSF y la SRF pueden estar localizadas en una misma máquina, o bien esta última puede encontrarse físicamente separada en un ente llamado IP (Intelligent Peripheral), caso en el cual el SCP le indica al SSP que debe establecer un canal de comunicación temporal con el SRF mediante el protocolo ISDN o ISUP.

La Red Inteligente se desarrolló para proveer una manera uniforme de ofrecer traducción de números a través de las redes de los distintos operadores. Así, permite desplegar una gran cantidad de servicios, tales como los números 800 en redes fijas y el roaming en redes móviles. Actualmente se utiliza para proveer servicios de prepago en la mayoría de los teléfonos móviles.

Otro modelo similar que apareció junto con el modelo de Red Inteligente fue el modelo de Nodos de Servicio. La diferencia básica con el primero es la de juntar las tres funciones (SCF, SSF y SRF) en un solo elemento de red. Esta aproximación brinda mayor flexibilidad ya que no se utiliza el protocolo INAP ni un modelo de llamada estandarizado, pero su desventaja radica en la centralización de las funciones, lo que deriva en mayores recursos destinados al control y mantenimiento de la llamada.

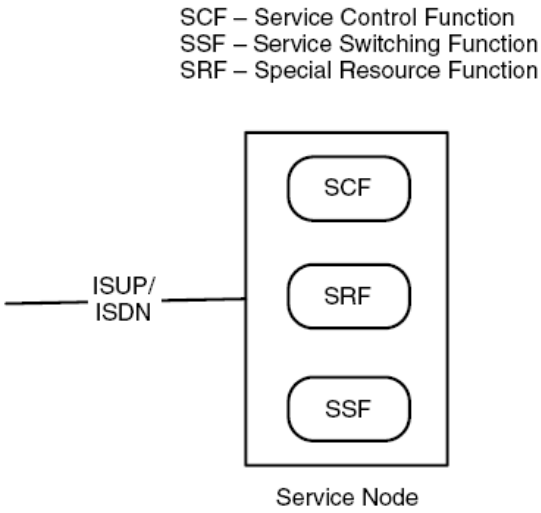

Figura 2-2 [1]: Nodos de Servicio

Estas estructuras planteadas en un principio se consideran arquitecturas de red monolíticas, en las cuales la transferencia de datos, el procesamiento de éstos y la interfaz de usuario (o red de acceso) están comprendidos dentro del mismo sistema. No obstante, representan un primer intento por separar las funciones de control y la conmutación de datos, acarreando el beneficio de desacoplar el desarrollo de nuevos servicios del progreso técnico de los switches, y asegurar que los servicios pudieran ser desarrollados por otras compañías aparte de los productores de estos elementos de red.

El siguiente paso fue el de plantear un modelo de capas, en el que se presentaban plataformas más especializadas y eficientes que en el caso anterior, mientras que se solucionaba el problema de poca escalabilidad de la estructura monolítica. Si bien esta solución representaba un avance, cada red con esta organización era independiente, y servicios cruzados entre redes se hacían difíciles de realizar y coordinar. Hoy en día, se plantea una estructura de convergencia entre dichas redes virtualmente separadas, capaz de ofrecer servicios sin importar el medio de acceso, utilizando un core (núcleo) común, basado en IP.

Un ejemplo de esta estructura la representa IMS (IP Multimedia Subsystem), arquitectura propuesta por el grupo 3GPP (3rd Generation Partnership Project), la cual se detallará a continuación.

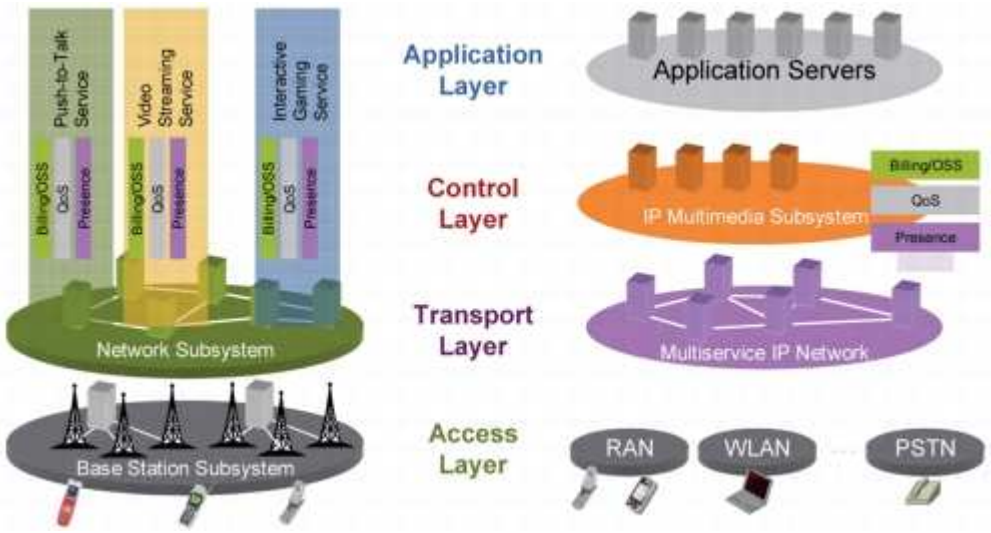

Figura 2-3 [2]: Evolución de Estructuras Monolíticas a Enablers por Capas

2.2 Arquitectura IMS 3GPP

3GPP [3] (Third Generation Partnership Project) es un acuerdo de colaboración en tecnología de telefonía móvil, establecido en Diciembre de 1998. Los miembros del grupo corresponden a ETSI (Europa), ARIB/TTC (Japón), CCSA (China), ATIS (América del Norte) y TTA (Corea del Sur).

El objetivo del 3GPP es desarrollar aplicaciones de telefonía móvil de tercera generación (3G), definiendo especificaciones y estándares bajo dicho propósito. Los sistemas 3GPP están basados en la evolución de los sistemas GSM, ahora comúnmente conocidos como sistemas UMTS. Cabe destacar que El 3GPP no debe ser confundido con 3GPP2, cuyos estándares y especificaciones están basados en la tecnología CDMA2000.

El roadmap de especificaciones referentes a telefonía móvil que el 3GPP desarrolla se detalla a continuación, poniendo énfasis en las más nuevas que dicen relación con las redes 3G.

Tabla 1 [3]: Releases (versiones) de estándares 3GPP

Versión	Año	Información
Release 99	2000	Primeras redes UMTS 3G, usando CDMA
Release 4	2001	Agregó red core basada en IP
Release 5	2002	Integra IMS, HSDPA
Release 6	2004	Integración con WLAN, agrega HSUPA, MBMS, mejoras a IMS (PoC y GAM)
Release 7	2007*	Reducción de latencia, QoS en VoIP, HSPA+, mejoras a la SIM, EDGE Evolution
Release 8*	2009*	E-UTRA, red full-IP (SAE)

* = Esperado

Cada “Release” incorpora variados documentos de especificaciones, incluyendo la última revisión de los estándares GSM. Los planes para el Release 7 están en desarrollo bajo el título Long Term Evolution (LTE); en la actualidad la mayoría de los operadores de redes móviles han

implementado redes Release 4 o bien Release 5 con HSDPA (High Speed Downlink Packet Access), planificando implementar IMS (IP Multimedia Subsystem) en un futuro cercano.

La introducción de IMS en las redes fijas y móviles [4] representa un cambio fundamental en las redes de telecomunicaciones de tipo voz. Las nuevas capacidades de las redes y de los terminales, el acercamiento entre la Internet y la voz, el contenido y la movilidad hacen aparecer nuevos modelos de redes, pero por sobre todo ofrecen un amplio potencial para el desarrollo de nuevos servicios. Con esta meta, IMS fue concebido para ofrecer a los usuarios la posibilidad de establecer sesiones multimedia usando todo tipo de acceso de alta velocidad y una red de conmutación de paquetes IP.

IMS provee una red IP multi-servicio, multi-acceso, segura y confiable:

- Multi-servicio: todo tipo de aplicaciones ofrecidas por un core (núcleo) de red, soportando diferentes niveles de calidad de servicio, podrán ser brindados al usuario.
- Multi-acceso: toda red de acceso de banda ancha, fija y móvil, podrá interconectarse a IMS.

IMS no es una única red sino diferentes redes que interoperan gracias a distintos acuerdos de roaming IMS fijo-fijo, fijo-móvil y móvil-móvil. Puede definirse como un enabler o catalizador que hace posible a los proveedores de servicios ofrecer:

- Servicios de comunicaciones en tiempo real, pseudo tiempo real y basadas en eventos, según una configuración cliente-servidor o entre entidades pares (Peer-to-Peer, P2P).
- La movilidad de servicios/movilidad del usuario (nomadismo).
- Varias sesiones y servicios simultáneamente sobre la misma conexión de red.
- Soporte para aplicaciones de redes ya desplegadas (Legacy).

2.2.1 Estructuración en Capas de la Arquitectura IMS

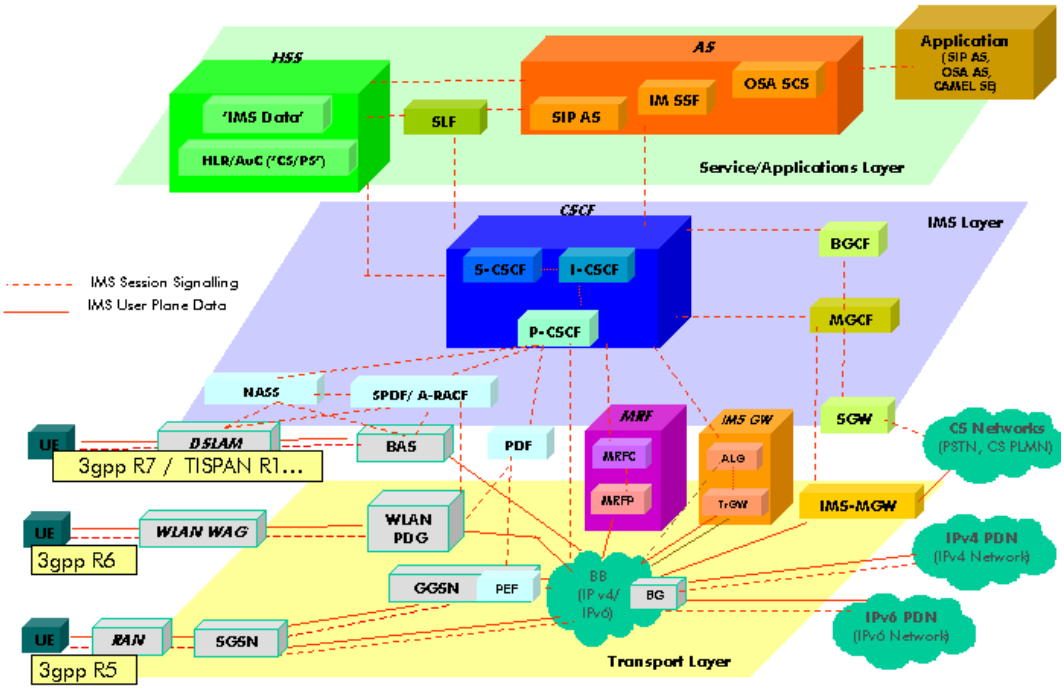

Figura 2-4 [5]: Arquitectura Global IMS

Como se vio en la sección de evolución de las arquitecturas de red, éstas están migrando hacia una estructura por capas, e IMS no se queda atrás en esta materia. Así, en la figura, cuatro capas importantes pueden ser identificadas:

- La **capa de acceso** puede representar todo acceso de alta velocidad tal como: UTRAN o CDMA2000 para redes móviles, xDSL, redes de cable, Wireless IP, WiFi, etc.
- La **capa de transporte** representa una red IP, la cual puede integrar distintos stacks de transmisión y mecanismos de calidad de servicio como MPLS. La capa de transporte esta compuesta de enrutadores o routers (edge routers para el acceso y core routers para el tránsito), conectados por una red de transmisión.
- La **capa de control** consiste en controladores de sesión responsables del encaminamiento de la señalización entre usuarios y de la invocación de los servicios. Estos nodos se denominan CSCF (Call State Control Function). IMS introduce entonces un ámbito de control de sesiones sobre el dominio de conmutación de paquetes.
- La **capa de aplicación** introduce las aplicaciones (servicios de valor agregado) propuestas a los usuarios. El operador, gracias a su capa de control, puede actuar como integrador de servicios ofrecidos por el mismo o bien por terceros. La capa de aplicación consiste en servidores de aplicación (Application Server, AS) y MRF (Multimedia Resource Function) que los proveedores llaman Servidores de Medios IP (IP Media Sever, IP MS).

La arquitectura por capas de IMS y su despliegue global se describen en las siguientes figuras:

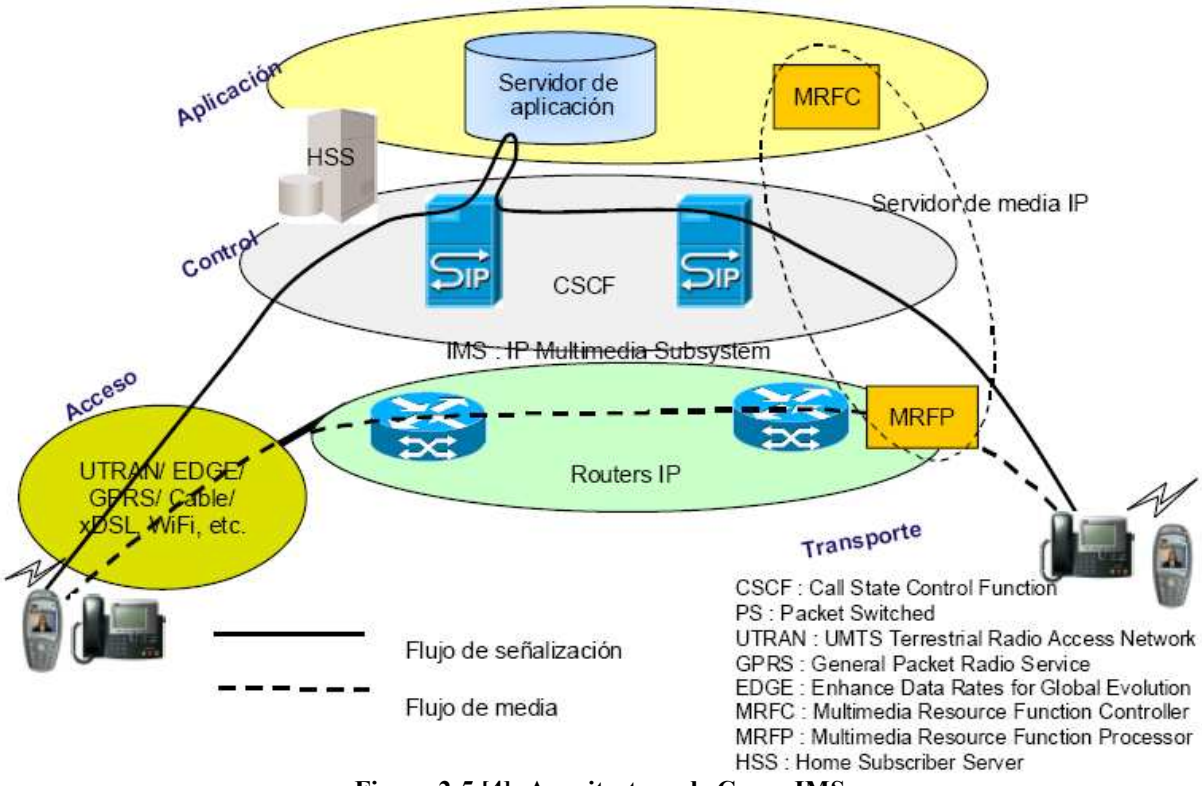

Figura 2-5 [4]: Arquitectura de Capas IMS

2.2.2 IMS y el Protocolo SIP

SIP [4] es un protocolo de señalización definido por la IETF (Internet Engineering Task Force) que permite el establecimiento, la liberación y la modificación de sesiones multimedia. SIP es usado en IMS como protocolo de señalización para el control de sesiones y el control de servicio. Así, reemplaza a la vez los protocolos ISUP (ISDN User Part) e INAP (Intelligent Network Application Part) del mundo de la telefonía, aportando el soporte de funciones multimedia.

SIP es adecuado para proveer servicios pues, ya que los elementos y protocolos subyacentes a SIP tienen mucho en común con HTTP. Es decir, diseñar e implementar nuevos servicios de voz u otros basados en SIP es tan fácil como crear páginas Web: las aplicaciones SIP son desarrolladas con métodos HTTP básicos, con los cuales los desarrolladores están familiarizados. Así, pueden crear aplicaciones orientadas a las telecomunicaciones de forma nativa y rápida, reduciendo el tiempo asociado a desplegar nuevas características en la red. Así, el operador se ve en la posibilidad de integrar transparentemente variadas aplicaciones, representando para los usuarios un servicio de comunicaciones en constante mejora, además de costos iniciales y recurrentes más bajos.

SIP se apoya sobre un modelo transaccional cliente/servidor, tal como HTTP, mientras que el direccionamiento utiliza el concepto de Uniform Resource Locator o URL SIP, parecido a una dirección e-mail. Cada participante en una red SIP es alcanzable por medio de una URL SIP.

Por otra parte, las solicitudes SIP son satisfechas por respuestas identificadas por un código numérico. Cabe destacar que SIP es un protocolo textual: un mensaje SIP está constituido de encabezamientos (headers), mientras que la mayor parte de los códigos de repuestas SIP provienen del protocolo HTTP. Por ejemplo, cuando el destinatario no se ha podido ubicar, un código de respuesta “404 Not Found” es enviado.

En un mensaje SIP común se pueden reconocer los siguientes componentes o campos fundamentales:

- METHOD: corresponde al tipo de mensaje que se está enviando (p. ej. REGISTER, INVITE, OPTIONS, BYE, etc.).
- Request-URI: indica el usuario al cual va destinado el mensaje.
- Via: contiene la dirección a la cual debe ser devuelta la respuesta a la petición, además de un parámetro branch que identifica la transacción.
- To: contiene un nombre y un URI SIP al cual está dirigido el mensaje.
- From: contiene un nombre y un URI SIP que indica el origen del mensaje, además de un parámetro tag agregado para propósitos de identificación en el cliente destino (p.ej. un softphone).
- Call-ID: identificador global formado por un número aleatorio y por el nombre del host o su dirección IP.
- Cseq: contiene un número entero y el nombre del método SIP. El número va incrementándose en una unidad por cada nueva dentro de un mismo diálogo.
- Contact: contiene un URI que sirve para indicar donde enviar peticiones futuras al cliente, a diferencia de Via que indica dónde enviar la respuesta a la petición en curso solamente.

Un ejemplo de mensaje SIP es el que sigue:

Tabla 2: Ejemplo Mensaje SIP

```
INVITE sip:Alice@172.16.231.186:21222 SIP/2.0
Max-Forwards: 69
Content-Length: 553
To: <sip:Alice@172.16.231.186:21222>;rinstance=b006705fab14c3e3
Contact: <sip:172.16.231.184:5060;fid=server_1>
Cseq: 1 INVITE
Via: SIP/2.0/UDP 172.16.231.184:5060;branch=z9hG4bK72725763-d2ff-4e0f-
9e24-823985c7595a
Content-Type: application/sdp
Call-Id: 172.16.231.184_6_159577508642757638
From:
<sip:Bob@172.16.231.184:14858>;rinstance=b8125ed1cea6a819;tag=ffbkzelc-e

v=0
o=- 8 2 IN IP4 172.16.231.184
s=CounterPath X-Lite 3.0
c=IN IP4 172.16.231.184
t=0 0
m=audio 24276 RTP/AVP 107 119 100 106 0 105 98 8 101
a=fmtp:101 0-15
a=rtpmap:107 BV32/16000
a=rtpmap:119 BV32-FEC/16000
a=rtpmap:100 SPEEX/16000
a=rtpmap:106 SPEEX-FEC/16000
a=rtpmap:105 SPEEX-FEC/8000
a=rtpmap:98 iLBC/8000
a=rtpmap:101 telephone-event/8000
a=sendrecv
m=video 27894 RTP/AVP 115 34
a=fmtp:115 QCIF=2 CIF=3 I=1 J=1 T=1 MaxBR=1960
a=fmtp:34 QCIF=2 CIF=3 MaxBR=1960
a=rtpmap:115 H263-1998/90000
a=rtpmap:34 H263/90000
a=sendrecv
```

Es importante destacar que la combinación de To, From y Call-ID definen completamente una relación peer-to-peer entre dos usuarios, es decir, un diálogo entre ellos, lo que permite sostener varias sesiones de forma simultánea. Los parámetros adicionales que se observan en el mensaje SIP de ejemplo dicen relación con el SDP (Session Description Protocol), sintaxis que se utiliza para negociar los parámetros básicos entre dos usuarios al transferirse algún tipo de medios. Según el estándar RFC 4566, un cuerpo SDP contiene fundamentalmente los atributos descritos a continuación [6], donde un asterisco (*) denota un campo opcional:

Descripción de la sesión:

- v= (versión del protocolo)
- o= (identificador de sesión y origen)
- s= (nombre de sesión)
- i=* (información de sesión)
- u=* (URI de descripción)
- e=* (dirección e-mail)
- p=* (número telefónico)

- c=* (información de conexión)
- b=* (información de ancho de banda)
- k=* (llave de encriptación)
- a=* (atributos de sesión)

Descripción de tiempo:

- t= (tiempo durante el cual la sesión está activa)
- r=* (cero o más tiempos de repetición)
- z=* (ajustes de zona horaria)

Descripción de medios, si están presentes:

- m= (tipo de medios y dirección de transporte)

Nota: se repiten los tipos i=*, c=*, b=*, k=* y a=*, pero ahora con información específica de los medios transferidos.

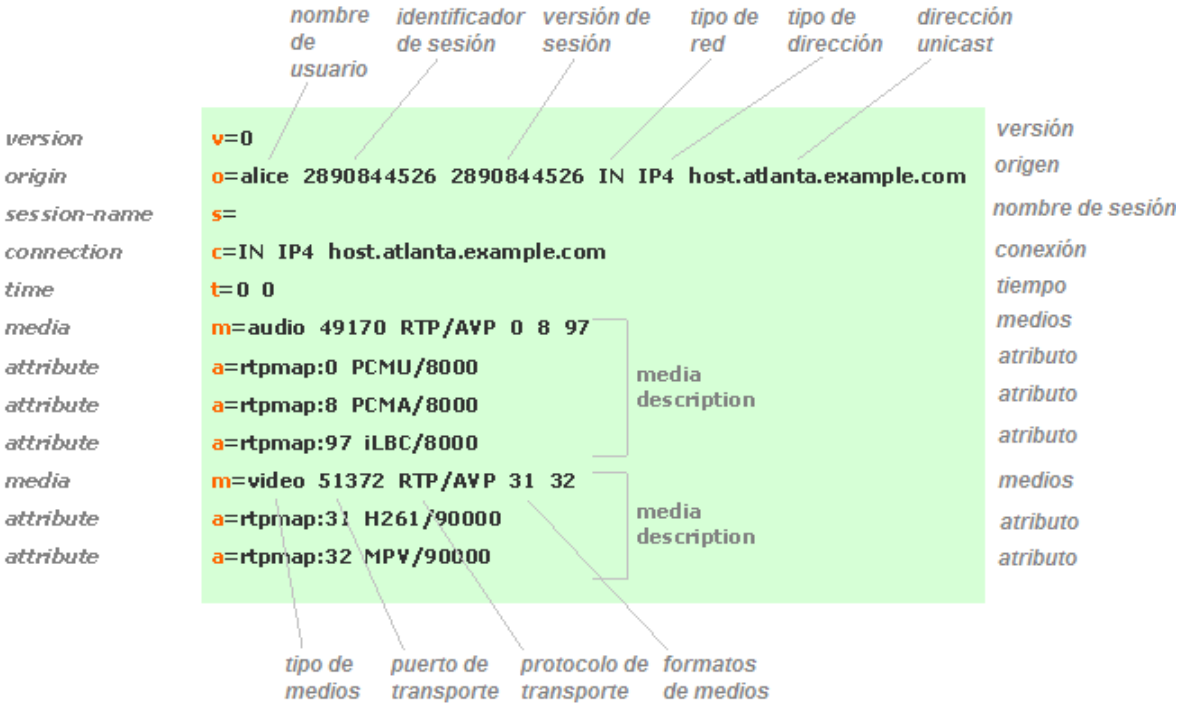

Figura 2-6 [6]: Ejemplo SDP

2.2.3 Entidades de Red IMS

Las entidades de red principales presentes en la red IMS [4] y sus funcionalidades se presentan a continuación:

Terminal IMS

Se trata de una aplicación sobre un equipo de usuario que emite y recibe solicitudes SIP. Se materializa a través de un software instalado sobre un PC, sobre un teléfono IP o sobre un terminal móvil GSM/UMTS (User Equipment o UE).

Home Subscriber Server (HSS)

La entidad Home Subscriber Server o HSS es la base principal de almacenamiento de los datos de los usuarios y de los servicios a los cuales se suscribieron. Los principales datos almacenados son las identidades del usuario, la información de registro y los parámetros de acceso, así como la información que permite la invocación de los servicios del usuario. La entidad HSS interactúa con las entidades de la red a través del protocolo DIAMETER.

Call State Control Function (CSCF)

El control de llamada iniciado por un terminal IMS tiene que ser asumido en la red nominal (red a la cual el usuario suscribe sus servicios IMS). Esto se debe a que el usuario puede suscribirse a una gran cantidad de servicios y algunos de ellos pueden no estar disponibles o pueden funcionar de manera diferente en una red visitada, entre otros problemas de interacción de servicios. Esto induce la definición de tres entidades: Proxy CSCF o P-CSCF, Interrogating CSCF o I-CSCF y Serving CSCF o S-CSCF.

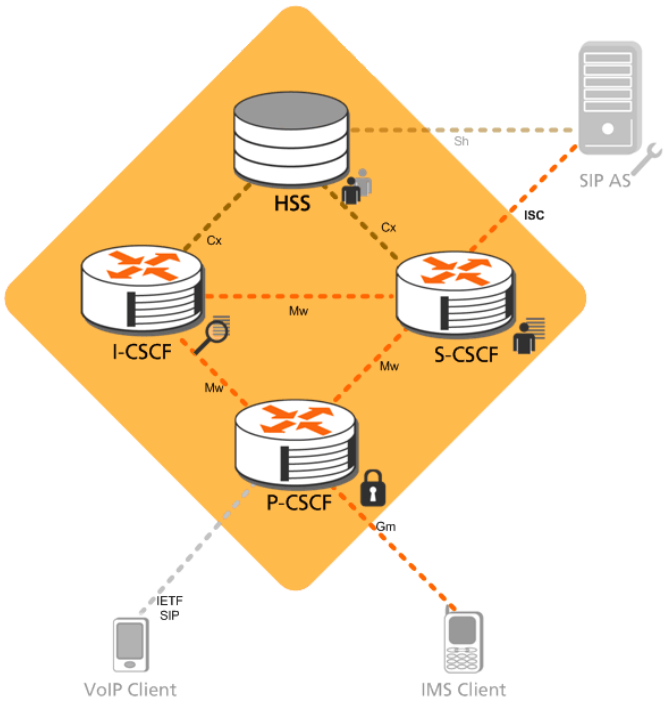

Figura 2-7 [7]: Entidades Básicas IMS

El P-CSCF es el primer punto de contacto en el dominio IMS. Su dirección es descubierta por un terminal móvil, por ejemplo, durante la activación de un contexto PDP para el intercambio de mensajes de señalización SIP, mediante una consulta a un servidor DNS.

El P-CSCF actúa como un Proxy Server SIP cuando encamina los mensajes SIP hacia el destinatario apropiado y como un User Agent SIP cuando termina la llamada (después de un error en el mensaje SIP recibido). Las funciones realizadas por la entidad P-CSCF abarcan:

- El encaminamiento del método SIP REGISTER emitido por el terminal a la entidad I-CSCF desde el nombre del dominio nominal
- El encaminamiento de los métodos SIP emitidos por el terminal al S-CSCF cuyo nombre ha sido obtenido en la respuesta del proceso de registrarse
- El envío de los métodos SIP o respuestas SIP al terminal
- La generación de Call Detailed Records o CDRs

- La compresión/descompresión de mensajes SIP

El I-CSCF es el punto de contacto dentro de una red para todas las sesiones destinadas a un usuario del respectivo operador. Pueden existir varias I-CSCF dentro de una red. Las funciones realizadas por la entidad I-CSCF incluyen:

- La asignación de un S-CSCF a un usuario que se registra
- El encaminamiento al S-CSCF de los métodos SIP recibidos desde otra red
- La obtención de la dirección del S-CSCF por parte del HSS
- La generación de CDRs

Por último, el S-CSCF asume el control de la sesión. Él mantiene un estado de sesión con el fin de poder invocar servicios. En una red de operadores, distintos S-CSCF pueden presentar diferentes funcionalidades. Las funciones realizadas por el S-CSCF durante una sesión incluyen:

- La emulación de la función Registrar ya que acepta los métodos SIP de registro y actualiza el HSS
- La emulación de la función Proxy Server ya que acepta los métodos SIP y los encamina
- La emulación de la función User Agent ya que puede terminar métodos SIP, por ejemplo cuando ejecuta servicios complementarios
- La interacción con servidores de aplicación después de haber analizado los criterios de contacto de los AS correspondientes (detallado más adelante)
- La generación de CDRs

Antes de poder utilizar los servicios del dominio IMS, tales como establecer una sesión multimedia o recibir un pedido de sesión, un usuario tiene que registrarse a la red. Esté el usuario en su red nominal o en una red visitada, este procedimiento involucra un P-CSCF.

Por otra parte, todos los mensajes de señalización emitidos por el terminal o destinados a él son relevados por el P-CSCF; el terminal nunca tiene el conocimiento de las direcciones de los demás CSCFs (I-CSCF y S-CSCF).

2.2.4 Arquitectura de Servicios IMS

La arquitectura de servicio IMS básica [4] está constituida por entidades servidores de aplicación, de servidores de medios IP y de S-CSCF equivalentes a servidores de llamadas.

El **servidor de aplicación SIP** o SIP AS ejecuta los servicios (p. ej. Push To Talk, Presencia, Prepago, Mensaje Instantáneo, etc.) y puede influenciar el desempeño de la sesión a pedido del servicio. El servidor de aplicación corresponde a la entidad SCF de la Red Inteligente.

El **servidor de medios IP** pone en obra la entidad funcional MRF. Él establece conferencias multimedia, difunde avisos de voz o multimedia y recolecta información del usuario. Se trata de la evolución de la entidad SRF de la Red Inteligente en el mundo multimedia.

El **servidor de llamadas SIP** o S-CSCF corresponde al punto desde el cual un servicio puede ser invocado. Dispone del perfil de servicio del usuario que le indica los servicios suscritos por éste y bajo cuáles condiciones invocar dichos servicios. Corresponde a la entidad SSF de la arquitectura de Red Inteligente.

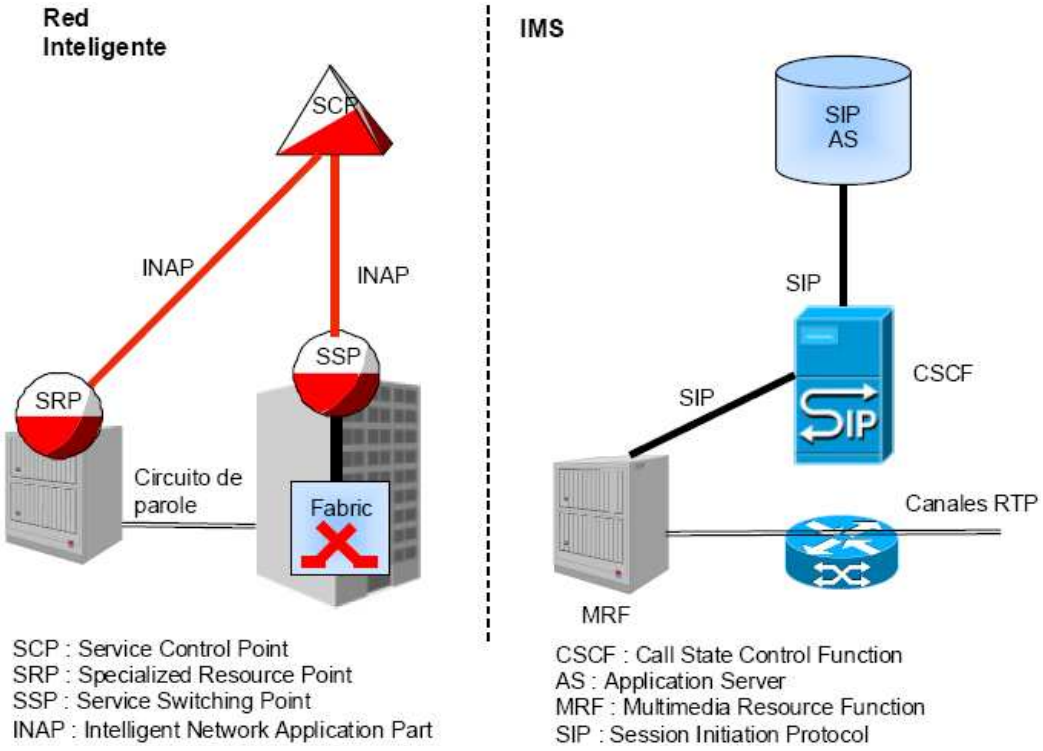

Figura 2-8 [4]: Red Inteligente vs. IMS

Cabe mencionar que el RFC 3976: Interworking SIP and Intelligent Network Applications trata el tema de integrar y soportar servicios provistos por la Red Inteligente en clientes basados en el protocolo SIP, cuando ocurre una llamada entre un host IP y un teléfono tradicional. La llamada es originada en un cliente SIP, pero los servicios para la llamada están provistos por datos y procedimientos residentes en la PSTN/IN. Para proveer dichos servicios de una manera transparente a los clientes SIP, el RFC detalla un mecanismo para realizar interworking entre SIP e INAP.

2.2.4.1 Entidades de la Arquitectura de Servicios IMS

La arquitectura de servicios IMS [4] consiste en un conjunto de AS interactuando con la red IMS (S-CSCF). En adición a los SIP AS mencionados en el punto anterior, los servidores de aplicación relevantes son:

- El punto de conmutación al servicio IM, IM-SSF (IP Multimedia Service Switching Function), el cual es un tipo particular de SIP AS que termina la señalización SIP sobre interfaz ISC por una parte y que tiene un papel de SSP RI/CAMEL por otra parte. Así. mediante el protocolo INAP/CAP es capaz de interactuar con los SCP RI/CSE CAMEL.
- El gateway OSA, OSA SCS (OSA Service Capability Server), que es un tipo específico de SIP AS que termina la señalización SIP sobre interfaz ISC y que interactúa con servidores de aplicación OSA usando el API OSA. Más adelante se detallará que significa OSA y en qué consiste su arquitectura particular.
- Un tipo especializado de servidor de aplicación SIP llamado gestor de interacción de servicios (Service Capability Interaction Manager, SCIM), que permite el manejo de las interacciones entre servidores de aplicación SIP.

Además de los AS, existe el mencionado servidor de medios, MRF. Establece conferencias multimedia, difunde anuncios de voz o multimedia y recaba información del usuario. Se trata de la evolución de la entidad SRF hacia el mundo multimedia. Las funciones de la entidad MRF son dos:

- La función MRF Processor o MRFP que procesa los medios a través del transporte RTP/UDP/IP.
- La función MRF Controller o MRFC que procesa la señalización.

Cabe destacar que la interfaz MR entre las entidades S-CSCF y MRFC es soportada por el protocolo SIP.

Todos los servidores de aplicación (IM-SSF y OSA SCS incluido) se comportan como servidores de aplicación SIP. Por otra parte, estos servidores de aplicación pueden interactuar con la entidad MRFC a través del S-CSCF para controlar las actividades de medios puestas en obra por la entidad MRFP.

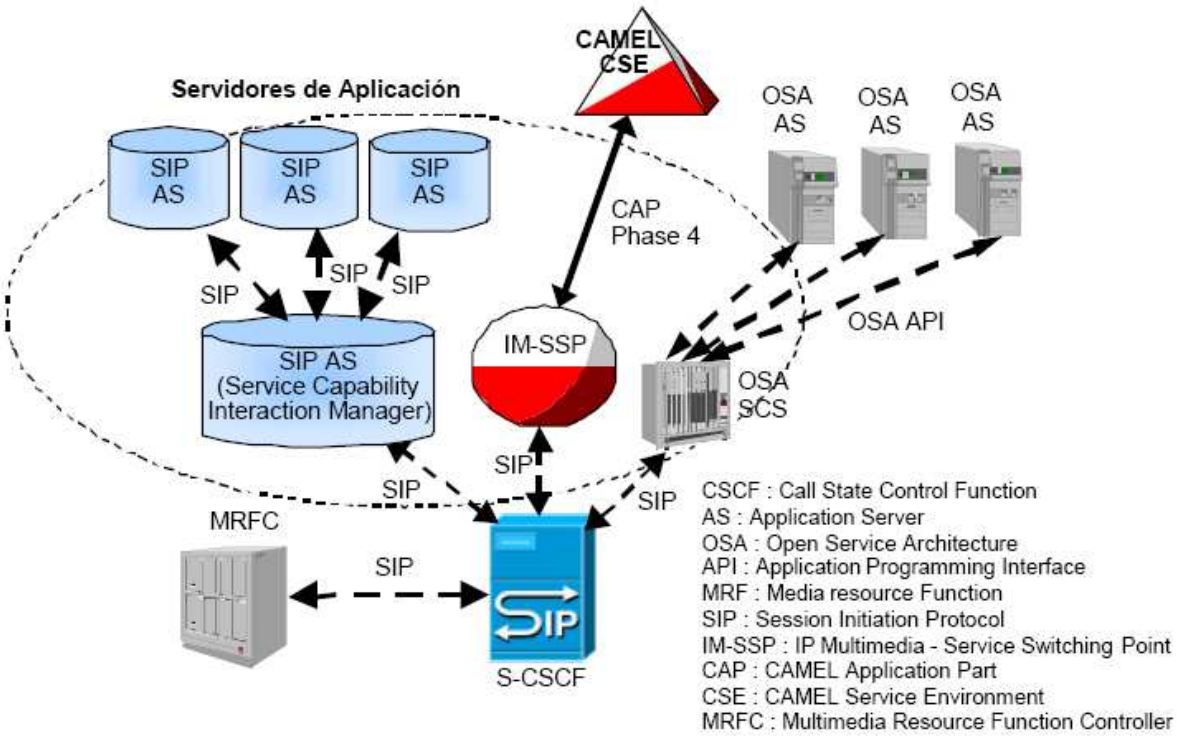

Figura 2-9 [4]: Arquitectura de Servicios IMS

2.2.4.2 Provisión de Servicios en IMS

En la arquitectura IMS [7], como se detalló anteriormente, los servicios son alojados y ejecutados en servidores de aplicación, por lo que se hace necesario tener un punto de referencia para poder enviar y recibir mensajes SIP entre estas entidades y los CSCFs. Esta interfaz se denomina ISC (IMS Service Control) y el protocolo seleccionado para su implementación corresponde a SIP. Los procedimientos de la interfaz ISC pueden corresponder al enrutamiento de un mensaje SIP hacia un servidor de aplicación, o bien al enrutamiento de una solicitud SIP iniciada por un servidor de aplicación, por ejemplo, en nombre de un usuario.

IMS provee los métodos necesarios para invocar y proveer servicios, lo que lleva a precisar tres pasos fundamentales:

1. Definir el posible servicio o conjunto de servicios.
2. Creación de información específica referente al usuario, cuando éste requiere una suscripción o la modifica, en formato de iFC (Initial Filter Criteria).
3. Encaminar el requerimiento inicial hacia un servidor de aplicación.

Los iFC son propios del perfil del usuario, y representan información específica de éste con respecto a la aplicación. Dichos datos son necesarios cuando la suscripción IMS del usuario contiene servicios de valor agregado a utilizar, o bien cuando un operador requiere utilizar servidores de aplicación como parte de su infraestructura IMS.

Un TP (Trigger Point) es un componente del iFC que es utilizado para decidir cómo es contactado un servidor de aplicación; es decir, a qué AS contactar y cómo se gatilla su intervención en la sesión. El TP contiene una o más instancias de SPT (Service Point Trigger), que comprenden los tipos condicionales que se detallan a continuación:

- Request-URI: corresponde a una identidad o dirección única del AS, por ejemplo, servidor@operador.com.
- SIP Method: indica el tipo de requerimiento, por ejemplo, mensajes del tipo INVITE.
- SIP Header: contiene información relativa al requerimiento, por lo que un SPT puede corresponder a la presencia o ausencia de dichos datos específicos, los cuales son interpretados como una expresión regular.
- Session Case: indica si el filtro debe ser utilizado por el S-CSCF que atiende a la llamada originada (Originating, del usuario que llama), terminada (Terminating, del usuario que es llamado), o terminada en un usuario no registrado en el core (Terminating Unregistered).
- Session Description: el SPT se define en base al contenido de cualquier campo SDP en el cuerpo del mensaje SIP, evaluado como una expresión regular.

Basado en lo anterior, el operador puede definir iFC para manejar a ciertos tipos de usuario. Por ejemplo, para aquellos usuarios que no se han registrado contra el core, se puede definir que cuando el SIP Method sea del tipo INVITE y el Session Case corresponda a Terminating Unregistered (lo que correspondería a una llamada hacia ese tipo de usuario), la llamada se desvíe hacia un buzón de voz localizado en una URI particular o simplemente se cancele la llamada.

Los iFC se codifican en lenguaje XML, como se muestra en el Anexo 1. Esta información se descarga del HSS hacia el S-CSCF cuando el usuario se registra o cuando el requerimiento va hacia un usuario no registrado. El S-CSCF evalúa los iFC de acuerdo a los siguientes pasos:

1. Chequear si al usuario se le está prohibido utilizar el servicio; si no, proceder.
2. Chequear si es un requerimiento iniciado por el usuario o terminado en éste, seleccionando los iFC para el caso correspondiente.
3. Chequear si el requerimiento coincide con los iFC registrados en el perfil del usuario de acuerdo a un orden estricto de prioridad. Es decir, si el requerimiento coincide con las reglas del iFC de más alta prioridad, se contacta al AS respectivo; si no coincide se sigue con el iFC de prioridad siguiente, y así sucesivamente hasta que se contacta al AS requerido.
4. Si el requerimiento no corresponde a ningún iFC y por lo tanto a ningún AS, enviar el requerimiento al siguiente nodo de acuerdo a las reglas de enrutamiento por defecto.

El AS recibe el requerimiento filtrado por la información establecida en los iFC, actuando como uno de los siguientes:

- Terminating UA: el AS actúa como UE, por ejemplo para proveer un servicio de Voice Mail.
- Redirect Server: servidor de redireccionamiento; el AS informa a quien originó el requerimiento sobre la nueva ubicación del usuario o sobre nuevos servicios, por ejemplo redireccionándolo a una página web específica.
- SIP Proxy: el AS procesa el requerimiento y lo envía de vuelta al S-CSCF, cambiando o agregando información en las cabeceras del mensaje SIP si es necesario.

Es preciso mencionar que el AS también puede actuar como Originating UA, siendo capaz de enviar requerimientos a los usuarios.

2.3 Otras Arquitecturas de Aplicaciones

En este apartado se detallarán algunas arquitecturas de servicios interesantes, las cuales resultan complementarias a IMS.

2.3.1 Parlay/OSA

Actualmente, los servicios a los cuales los usuarios acceden en una red dependen en demasía del operador, por lo cual se tienen usuarios cautivos y las aplicaciones nuevas tardan en aparecer. Los usuarios quieren que dichos servicios sean estables y soportados en cualquier red, teniendo la libertad de utilizarlos mediante cualquier tipo de acceso; los operadores quieren hacer negocio con ello, desarrollando nuevas tecnologías que incrementen sus ingresos o se reduzcan sus costos; y los desarrolladores de aplicaciones quieren poder vender sus productos, desplegándolos de una manera fácil y rápida. Así, se hacen necesarios cambios en los modelos de negocios que convengan los deseos de todas las partes y aumente los beneficios de cada una de ellas.

El ingreso de un nuevo proveedor de servicios estaba anteriormente supeditado al despliegue de una red, cuyo alto costo representaba un obstáculo mayor. En la actualidad, el cambio en los modelos de negocios mencionado permite que un nuevo jugador pueda ofrecer servicios a través de operadores establecidos. Así, aparecen nuevas fuentes de ingresos para el operador, que enriquece su oferta de servicios, incrementándose así el tráfico y abriéndose el negocio de las telecomunicaciones a otras entidades como lo son los nuevos proveedores de servicios mencionados.

Ahora bien, con tal de que lo anteriormente expuesto ocurra y se puedan ofrecer servicios transversales a través de redes convergentes como las actuales, debe existir un elemento técnico que comunique el mundo del operador con los distintos proveedores de aplicaciones. Es aquí donde aparece el concepto de la arquitectura Parlay/OSA [9].

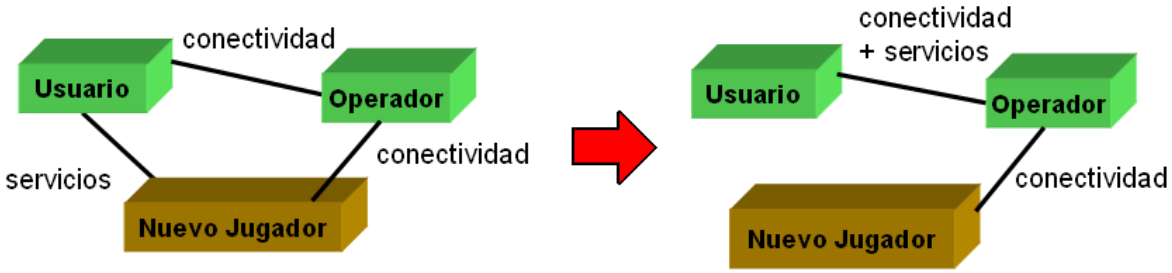

Figura 2-10 [10]: Modelo de Interacción Parlay/OSA

Parlay es un grupo de empresas ligadas a las telecomunicaciones y a las tecnologías de la información, que busca desarrollar APIs abiertas y ricas en funcionalidad que permitan el acercamiento de ambos mundos (Telco-TI). OSA, por su parte, acrónimo de Open Service Access, corresponde a la arquitectura para servicios móviles desarrollada por la 3GPP para la tercera generación de redes de telecomunicaciones móviles (3G-UMTS). Parlay/OSA es la API desarrollada por los grupos mencionados, que define un conjunto de interfaces abiertas y estandarizadas para permitir a terceros desarrollar aplicaciones que accedan a las funcionalidades de la red de los operadores. En realidad se trata de un conjunto de APIs, que buscan permitir la provisión de aplicaciones de localización, mensajería, presencia, control de la llamada y tarificación por contenido, entre otras, a los usuarios móviles.

El fin de Parlay, al utilizar APIs estandarizados y abiertos, es acortar el tiempo de puesta en servicio de las aplicaciones ("time to market"), aprovechando al máximo las capacidades de programación de los especialistas en Tecnologías de la Información para el desarrollo de aplicaciones asociadas al mundo de las Telecomunicaciones. De esta manera, servicios nuevos e innovadores desarrollados por terceros pueden ser desplegados, los cuales no dependen del acceso a la red, abriendo así la posibilidad de otorgar servicios multi-acceso. Parlay/OSA independiza el entorno de las aplicaciones de los protocolos de telecomunicaciones en la red subyacente, garantizando una comunicación transparente entre estos dos ámbitos que pueden parecer disímiles.

Hablando de Parlay en particular, la arquitectura considera tres entes lógicos claves: Client Application, Service Capability Servers (SCS) y Framework. A grandes rasgos, el primero proporciona la interfaz humano-máquina, el SCS traduce desde Parlay hacia el lenguaje de la red subyacente y viceversa, y el Framework se encarga del registro y autenticación de aplicaciones y servidores, control de fallas, y otras funciones administrativas.

Los SCS son entes funcionales que proporcionan interfaces Parlay/OSA a las aplicaciones, a través de abstracciones de las funcionalidades de la red accesibles mediante el API. A éstas últimas se les llama Service Capability Features (SCFs). Cada SCS presenta uno o más SCFs, que se definen en términos de la clase de interfaz y sus métodos.

El Framework, por su parte, permite acceder a los servicios de la red, aislando a su vez la infraestructura de la red de comunicaciones. Todas las aplicaciones que requieran la utilización de Parlay/OSA deben registrarse con el Framework, la cual está representada en la red mediante la Parlay/OSA Gateway. Actualmente existen variadas empresas que ofrecen dichas entidades de red, tales como Alcatel-Lucent y Ericsson.

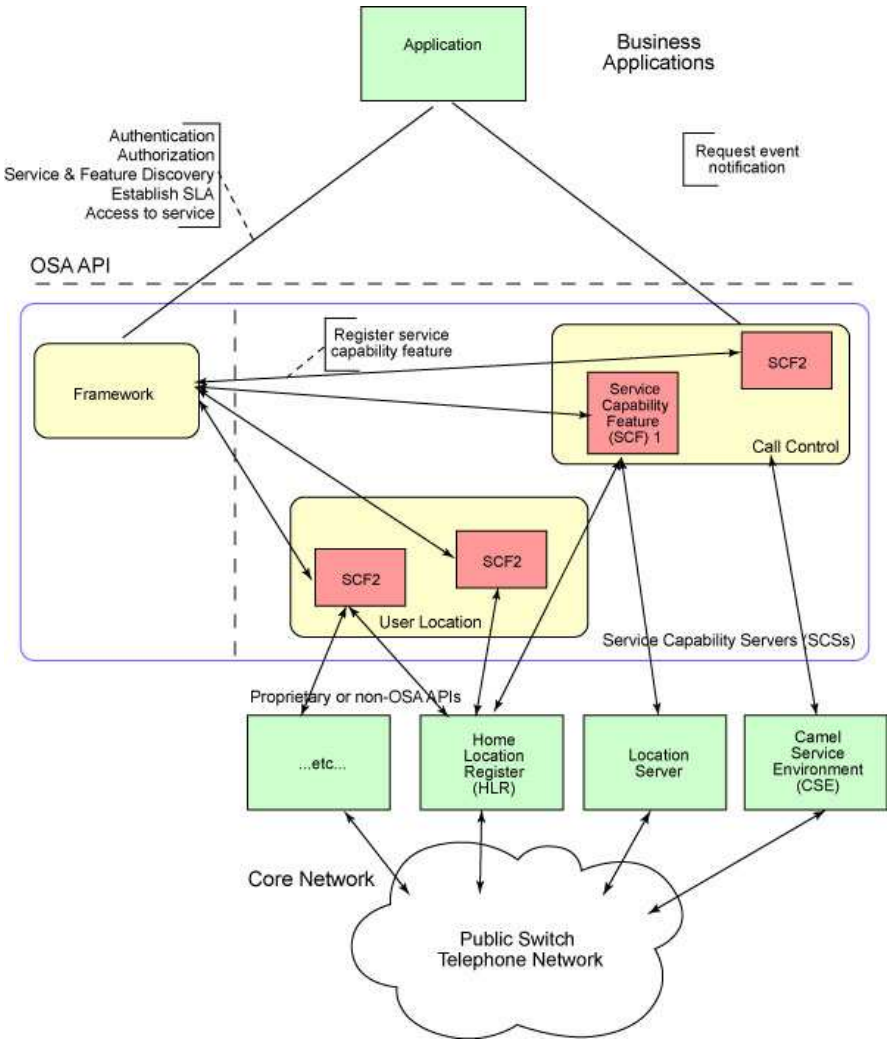

Figura 2-11 [11]: Arquitectura Detallada Parlay/OSA

En la actualidad, el grupo Parlay, en conjunto con la ETSI y la 3GPP, están desarrollando la estandarización de una serie de Web Services basados en la arquitectura Parlay/OSA y el lenguaje WSDL, bajo el nombre de Parlay X. La idea de estos estándares es presentar un conjunto de servicios Web que pueden ser utilizados para desarrollar aplicaciones de telecomunicaciones, partiendo de ellos como base. La especificación Parlay X 2.0 (versión actual) incluye las especificaciones para proveer servicios tales como mensajería corta, presencia, conferencia multimedia y administración de listas de contacto, entre otras. Más adelante se entrará en detalle sobre Parlay X.

2.3.2 OMA IMPS

OMA IMPS [12] no es en realidad una arquitectura de servicio, sino un enabler que presenta una estructura digna de destacar en este ámbito, como ejemplo de una plataforma desarrollada para la provisión de una aplicación específica.

La OMA (Open Mobile Alliance) es una organización creada en junio de 2002 que desarrolla estándares abiertos para la industria de la telefonía móvil. Su misión es proporcionar servicios interoperables que permitan trabajar a través de diversos países, operadoras y

dispositivos móviles (terminales de usuario), mientras que entre sus miembros se incluyen algunos de los más importantes fabricantes de dispositivos móviles, operadores de telefonía móvil y compañías de software. Las especificaciones OMA son agnósticas en cuanto al tipo de tecnología de red a utilizar en la conexión y el transporte de datos, es decir, la especificación de OMA para una funcionalidad concreta, es la misma para redes GSM, UMTS o CDMA2000.

IMPS (Instant Messaging and Presence Service) es un enabler desarrollado por la OMA, diseñado para intercambiar mensajes instantáneos e información de presencia no sólo entre dispositivos móviles, sino también entre dispositivos móviles y fijos. Sus orígenes se remontan a la iniciativa Wireless Village, la cual se incorporó luego a la OMA, finalizándose la versión 1.2 de IMPS. Actualmente IMPS se encuentra en la versión 1.3, la cual representa una mejora significativa, no siendo totalmente compatible con la versión 1.2 y anteriores.

IMPS incluye cuatro servicios principales:

- Presencia
- Mensajería instantánea
- Grupos
- Contenido compartido

La presencia es la tecnología clave para IMPS, e incluye la disponibilidad del dispositivo del cliente (mi teléfono está encendido/apagado, en una llamada, etc.), estado del usuario (disponible, no disponible, en una reunión, etc.), localización, capacidades del dispositivo del cliente (voz, texto, GPRS, multimedia, etc.) e información personal actualizable, tal como el humor (alegre, enojado, etc.) y actividades (deportes, cine, etc.).

Por otra parte, la mensajería instantánea es un concepto familiar para el mundo tanto móvil como fijo. Los clientes de chat en línea, tales como MSN Messenger y Google Talk, así como los SMS enviados entre celulares, son formas de mensajería instantánea. IMPS habilitará la mensajería instantánea móvil e interoperable, en conjunto con otras características innovadoras, para proveer una experiencia de usuario mejorada.

En cuanto a los grupos, tanto los operadores y usuarios finales son capaces de crear y administrarlos. Los usuarios pueden invitar a sus amigos y familia a conversar en discusiones grupales, mientras que los operadores pueden crear grupos de interés común donde los usuarios finales pueden conocerse en línea.

Por último, el contenido compartido permite a los usuarios y operadores montar su propia área de almacenamiento donde pueden publicar imágenes, música y variados contenidos multimedia, compartiéndolos con otros individuos y grupos en una sesión de mensajería instantánea, por ejemplo.

IMPS es un sistema cliente-servidor, donde los clientes que se conectan con el servidor IMPS pueden ser terminales móviles, clientes fijos u otros servicios o aplicaciones. La arquitectura IMPS se presenta en la siguiente figura:

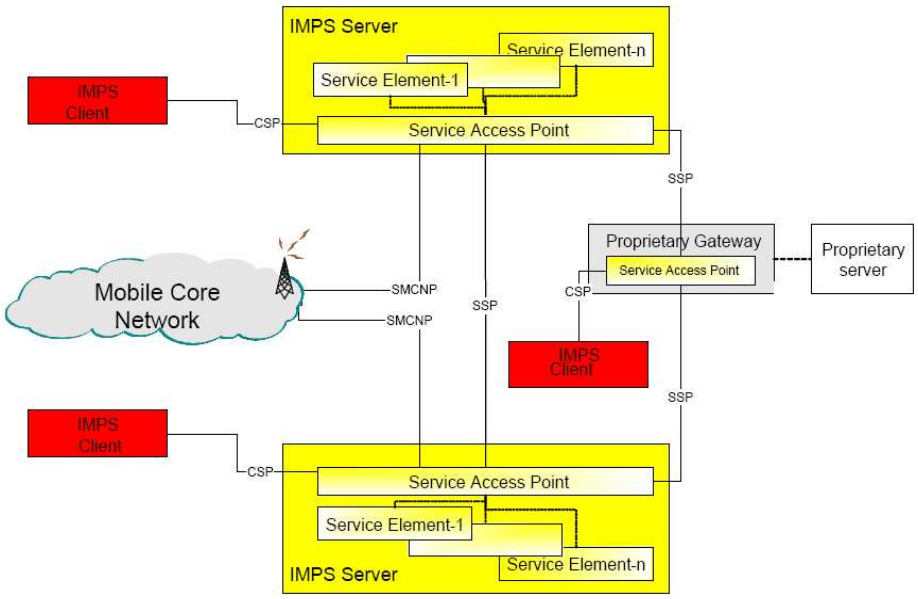

Figura 2-12 [12]: Arquitectura OMA IMPS

El servidor IMPS es el punto central en un sistema IMPS. Está compuesto de cuatro Application Service Elements que son accesibles a través del Service Access Point. Los Application Service Elements proveen la funcionalidad de gestión de la información específica de cada servicio, y corresponden a los siguientes:

- Presence Service Element (Presencia)
- Instant Messaging Service Element (Mensajería instantánea)
- Group Service Element (Grupos)
- Content Service Element (Contenido compartido)

El Service Access Point (SAP) sirve como interfaz entre el servidor IMPS y cualquier otra entidad de red, tal como los clientes IMPS, otros servidores IMPS, la red core y gateways propietarios hacia servidores no-IMPS. El Service Access Point cumple las funcionalidades de:

- Autentificación y autorización (Authentication and Authorization)
- Descubrimiento de servicios y acuerdos de servicio (Service Discovery, Service Agreement)
- Gestión de perfiles de usuario (User Profile Management)
- Relay de servicios (Service Relay)

Los clientes IMPS corresponden a terminales fijos o móviles, los cuales interoperan a través del Service Access Point utilizando CSP (Client Server Protocol).

Las interfaces y protocolos utilizados se pueden apreciar en la siguiente figura, en la cual se aprecia claramente la estructura de capas:

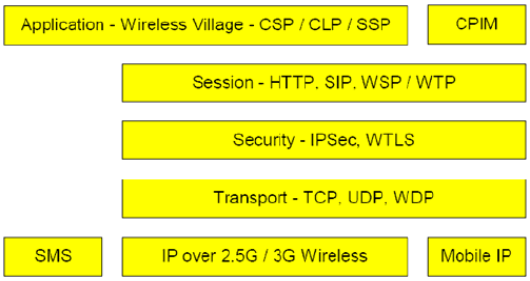

Figura 2-13 [12]: Protocolos OMA IMPS por Capas

A modo de ejemplo, se muestra a continuación el flujo de un mensaje instantáneo entre dos usuarios, el cliente A en una red IMPS y el cliente B en una red no-IMPS pero que interopera con la primera a través de un gateway propietario.

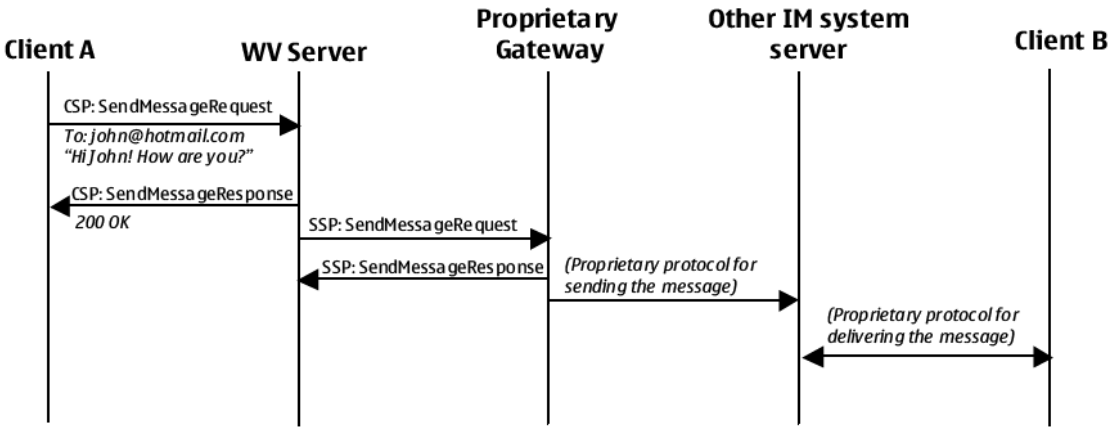

Figura 2-14 [13]: Flujo Sesión Mensajería OMA IMPS

2.4 Implementación de Servicios basados en SIP

En este apartado se ejemplificarán algunos de los innumerables servicios que se pueden proveer mediante una arquitectura basada en SIP, además de las opciones de implementación que existen.

2.4.1 Servicios Capaces de Ser Desplegados

Algunos ejemplos de aplicaciones que se pueden implementar en una plataforma basada en el protocolo SIP son los que siguen:

2.4.1.1 Servicios de Voz

La telefonía IP es uno de los servicios que se pueden proveer mediante el uso del protocolo SIP. Dicho protocolo hace las veces de señalización, mientras que la voz es transportada a través de un flujo de paquetes encapsulados en RTP.

Por otro lado, el servicio de Voicemail (correo de voz) es un servicio conocido que puede ser provisto mediante SIP. El servicio de Voicemail consiste en lo siguiente: un usuario llama a otro que se encuentra ocupado o no contesta, a lo cual el primero deja grabado un mensaje de voz en un repositorio específico del servicio. Luego, el segundo recibe una notificación de que tiene un mensaje a su disposición, por ejemplo a través de un SMS o directamente mediante un MWI (Message Waiting Indicator). De esta manera, el usuario puede recuperar el mensaje cuando esté desocupado o cuando él lo requiera, mediante una llamada de voz con señalización SIP como la descrita anteriormente.

2.4.1.2 Presencia y Mensajería Instantánea

La mensajería instantánea y la presencia son aplicaciones extremadamente populares en Internet. Realzando el actual servicio de SMS (Short Messaging System), estos servicios permiten la mensajería en tiempo real entre usuarios que están al tanto del estado de cada uno, lo que provee un mejor servicio y soporta nuevos tipos de escenarios de uso. Un usuario puede seleccionar varios modos de presencia (por ejemplo: ocupado, disponible, disponible para jugar, etc.) y compartir esta información con otros usuarios predefinidos.

SIP puede ser usado para mensajería instantánea y presencia mediante un método dedicado llamado MESSAGE, que puede transportar cualquier contenido MIME (como texto plano "text/plain" o una imagen "image/gif"), y que ha sido definido por la IETF para propósitos de mensajería. Cabe notar que SIP permite el uso de herramientas de presencia, aun cuando la gestión de dichas características, tales como la edición de la lista de contactos, se hace típicamente con herramientas basadas en HTTP o WAP. Con SIP, la mensajería instantánea o multimedia puede ser integrada transparentemente con comunicaciones conversacionales, creando el concepto de *comunicaciones ubicuas*: el uso transversal de variados medios de comunicación y servicios de mensajería, en tiempo real, dependiendo de la información de presencia de las partes.

Los siguientes requerimientos se deben satisfacer para utilizar presencia:

- La red debe ser capaz de identificar usuarios independientemente de su localización.
- Peticiones de suscripción deben poder ser remitidos al servidor que maneja a dicho usuario.
- El usuario debe ser capaz de indicar al servidor su información de presencia y opcionalmente su localización.
- La red debe poder remitir notificaciones a los suscriptores.
- La red debe ser escalable.
- La red debe ser capaz de entregar mensajes en tiempo real.

SIP provee todo lo anterior; por ejemplo remite peticiones INVITE u otras a servidores que manejan el usuario y el método REGISTER permite al usuario informar su localización y otra información al servidor. Más aún, la IETF ha desarrollado una serie de especificaciones bajo el alero del grupo SIMPLE (SIP for Instant Messaging and Presence Leverage Extensions) con el fin de definir una estructura basada en SIP que soporte los servicios de mensajería instantánea y presencia. Este protocolo hace uso del lenguaje XML/XCAP y está avalado también por la OMA en sus enablers relacionados con dichos servicios. En IMS, el HSS se transforma en un servidor

de presencia en sí, sino en una interfaz común para la información de presencia de un usuario, la cual queda disponible para las aplicaciones que quieran hacer uso de ella.

Un ejemplo de los servicios de mensajería instantánea y presencia sería el siguiente: el usuario A se registra para seguir la información de presencia del usuario B. Cuando A se conecta a la red, B recibe una notificación de ello mediante un mensaje instantáneo. Si ahora B quiere llamar a A, ocurre lo mismo que antes, pero la lógica de servicio de A chequea su disponibilidad. Encuentra que A está en reunión y acepta sólo mensajes instantáneos y e-mails, e informa de esto a B mediante un mensaje instantáneo. B envía un mensaje instantáneo a A. A contesta con otro mensaje instantáneo y se inicia una sesión de chat. Todo esto ocurre mientras el CSCF local está cobrando una pequeña tarifa por mensaje.

2.4.1.3 Otros Servicios

Ejemplos adicionales de servicios se presentan a continuación [14]:

Llamada Multimedia

Actualmente, cuando se llama a un usuario en un teléfono móvil GSM, el número del emisor es desplegado generalmente en la pantalla del teléfono del receptor. En vez de sólo mostrar el nombre, quien llama podría enviar su tarjeta de visita, una fotografía, un sonido personalizado, o cualquier otro elemento extra. Más aún, durante la llamada, los usuarios podrían intercambiar información tal como documentos, datos, imágenes o cualquier otro tipo de medios. En otras palabras, una llamada multimedia es el término para ampliar la definición de una llamada telefónica, agregándole otros tipos de medios.

Respuesta Automática

El usuario B es un fanático del fútbol y ofrece un servicio automático de entrega de resultados. Cuando otros usuarios le envían un mensaje SIP que incluye la palabra "fútbol", la respuesta es un mensaje HTML o ASCII generado automáticamente que incluye los últimos resultados. La palabra clave puede estar, por ejemplo, en la cabecera "Subject", o dentro de otra cabecera.

Podrían también configurarse reglas para que se enviara un SMS determinado si el usuario está en reunión, o presentar una voz de respuesta automática que alerte que el usuario está ocupado o hablando en el momento.

Otros servicios capaces de ser desplegados son los siguientes:

- Conferencia
- Push-to-Talk
- Click-to-Dial
- Un sinfín de otros servicios convergentes…

2.4.2 Flujos SIP de Aplicaciones

Una llamada SIP básica puede ser representada mediante el siguiente esquema:

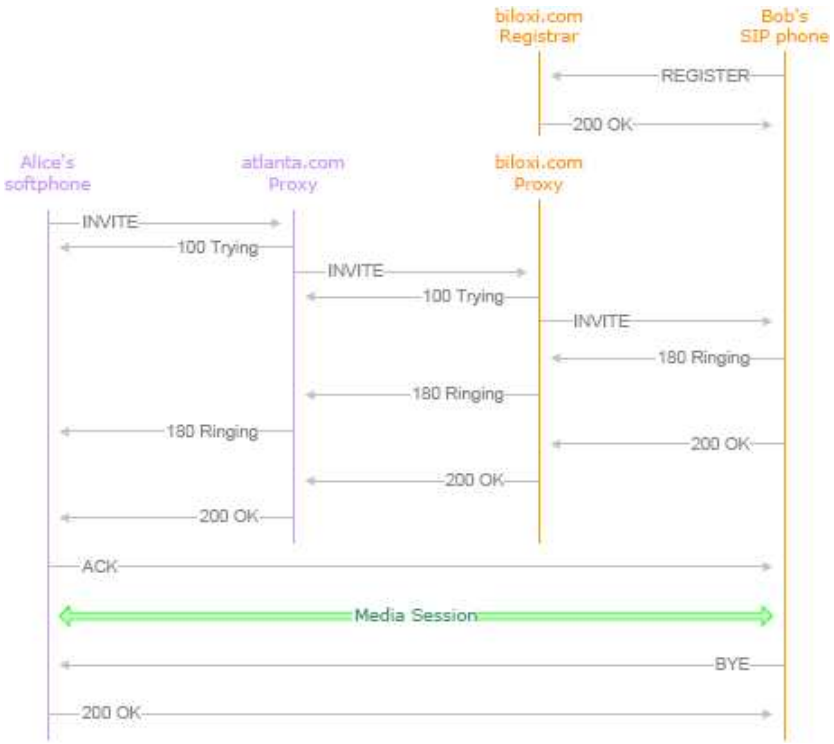

Figura 2-15 [15]: Flujo Sesión SIP Básica

Ahora bien, cada servicio tiene su flujo de mensajes propio, con ciertas características que lo diferencian de otros servicios. Algunos ejemplos se muestran a continuación:

Presencia

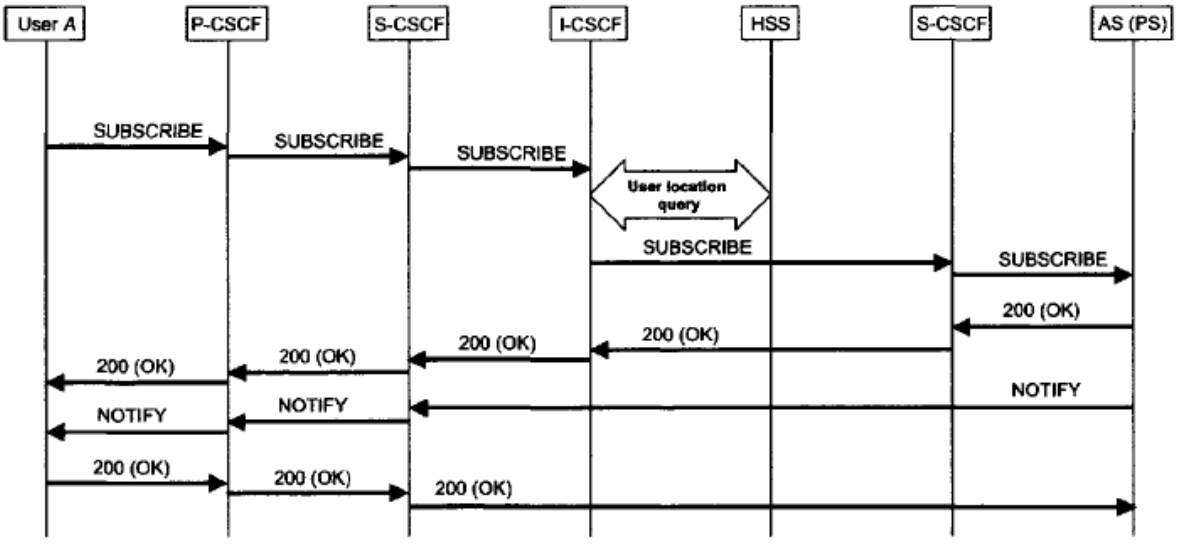

Figura 2-16 [8]: Suscripción Exitosa a Presencia

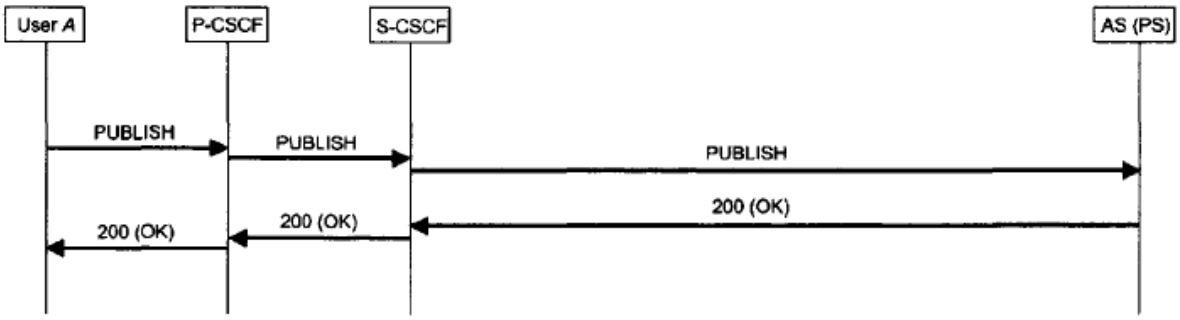

Figura 2-17 [8]: Publicación Exitosa de Presencia

Mensajería

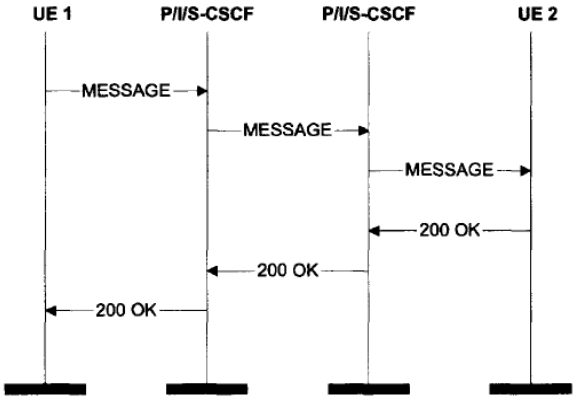

Figura 2-18 [8]: Flujo de Mensajería Inmediata

Conferencia

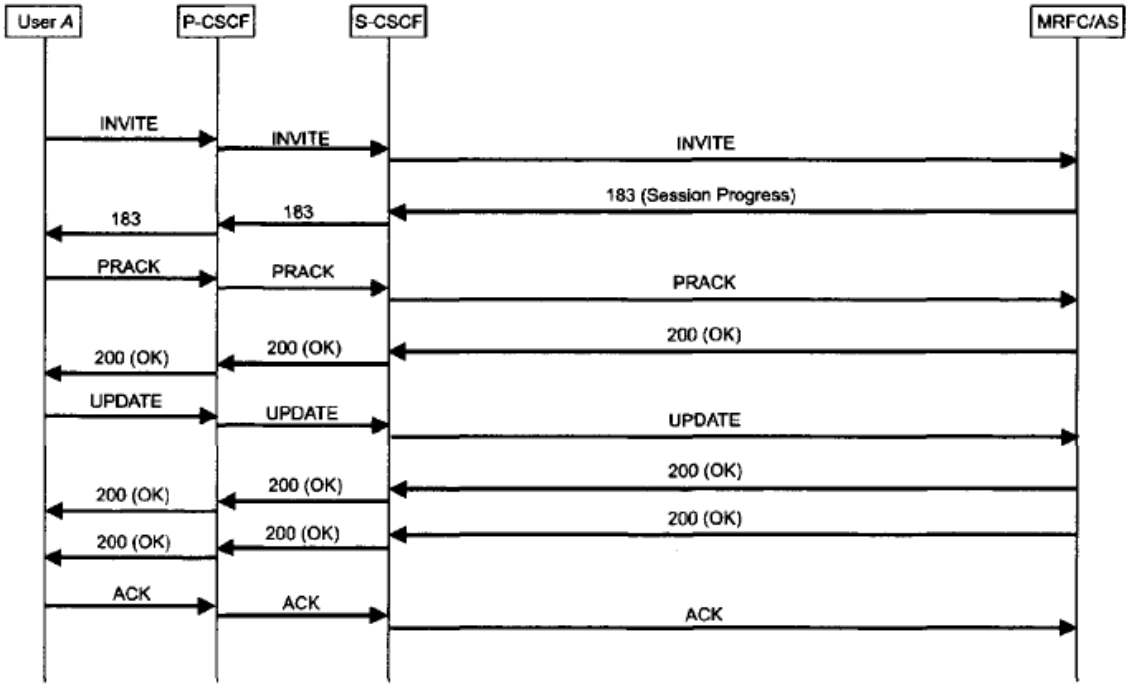

Figura 2-19 [8]: Creación de una Conferencia

2.4.3 Interfaces SIP

Existen variadas alternativas para implementar servicios basados en SIP [16], entre las cuales se pueden destacar las siguientes:

CPL

Call Processing Language (CPL) corresponde al primer API desarrollado para SIP. En estricto rigor, no es realmente un API, sino un lenguaje de scripting basado en XML para describir y controlar servicios de llamada. Está diseñado para ser implementado en servidores de red, o bien en servidores de UAs, y está pensado para ser simple, extensible, fácilmente editable mediante clientes gráficos e independiente del sistema operativo o del protocolo de señalización.

CPL está diseñado para la creación de servicios de usuario final: un intérprete de CPL es muy liviano y un servidor puede fácilmente analizar y validar un CPL, además de proteger contra comportamiento malicioso. Es adecuado para correr en un servidor donde a los usuarios no se les permite ejecutar programas arbitrarios, pues no tiene variables, loops ni capacidad para ejecutar programas externos. Tiene rutinas primitivas para tomar decisiones y llevar a cabo acciones basadas en las propiedades de las llamadas, tales como hora, quien llama, quien recibe la llamada, etc. Por último, el estándar se puede encontrar en el sitio web de la IETF [35] (RFC 3880).

SIP CGI

En Internet, la Common Gateway Interface (CGI) ha servido como un medio popular para programar Web Services. Los scripts CGI han sido el mecanismo inicial para hacer que los sitios web interactúen con bases de datos y otras aplicaciones. Debido a las semejanzas entre SIP y HTTP, CGI es un buen candidato para la creación de servicios en un ambiente SIP.

Como en HTTP, un script SIP CGI reside en el servidor y traspasa parámetros de mensaje mediante variables a un proceso separado. El proceso envía instrucciones de vuelta al servidor mediante su descriptor de archivos de salida estándar. SIP CGI es casi idéntico a HTTP CGI y es particularmente adecuado para servicios que contienen componentes Web sustanciales.

Un script CGI puede ser escrito en Perl, C, C++, Java u otros, haciéndolo accesible a una gran comunidad de desarrolladores. El estándar se puede encontrar en el sitio Web de la IETF [35] (RFC 3050).

SIP Servlets

Un servlet es una aplicación Java que corre en un servidor Web o un servidor de aplicaciones, la cual provee procesamiento en el lado del servidor, por ejemplo, para acceder a una base de datos. En otras palabras, es un reemplazo basado en Java para los scripts CGI, Active Server Pages (ASPs) y plug-ins propietarios escritos en C y C++. Los servlets son similares al concepto de los CGI, pero, en vez de usar un proceso separado, los mensajes son traspasados a una clase Java que corre en una máquina virtual (Java Virtual Machina, JVM) dentro del servidor.

Los SIP Servlets son similares a los HTTP Servlets; solamente complementan la interfaz para soportar funciones SIP. Además, como están escritos en Java, los Servlets son operables en distintos servidores y sistemas operativos. La especificación está siendo desarrollada bajo el programa Java Community Process(SM).

JAIN™ APIs

Las JAIN APIs están siendo especificadas como una extensión basada en comunidades para la plataforma Java. Al proveer un nuevo nivel de abstracción e interfaces Java asociadas para la creación de servicios a través de redes de conmutación de circuitos y de paquetes, JAIN está juntando protocolos IP y de la Red Inteligente para crear un mercado abierto.

El objetivo de la iniciativa JAIN es crear una cadena de valor abierta en la provisión de servicios orientados a las telecomunicaciones atendiendo la portabilidad de los servicios, la convergencia de las redes y el acceso a la red del proveedor del servicio.

- Portabilidad: las JAIN APIs adaptan interfaces propietarias para permitir aplicaciones verdaderamente portables.
- Convergencia: la tecnología JAIN permite que los servicios corran sobre cualquier arquitectura de red subyacente, sea ésta IP, ATM, TDM, inalámbrica, etc.
- Acceso del proveedor: las JAIN APIs especifican mecanismos para permitir el acceso directo a recursos y dispositivos de la red, con tal que se lleven a cabo acciones o funciones específicas en relación a los servicios.

Actualmente existen dos SIP APIs que han sido desarrolladas o están siéndolo, en el marco de la iniciativa JAIN:

- JAIN SIP: es un API de bajo nivel que se mapea directamente al RFC 2543 publicado por la IETF (actualmente RFC 3261). El desarrollo de JAIN SIP se encuentra finalizado y la especificación del API y el Reference Implementation and Technology Compatibility Kit (suite para realizar pruebas) pueden ser descargados libremente desde el sitio Web del programa Java Community Process.
- JAIN SIP Lite: es un API de alto nivel cuya meta es permitir a los desarrolladores crear aplicaciones basadas en SIP de forma rápida y sin necesidad de tener un conocimiento extensivo sobre el protocolo. Es decir, es un API ligero que encapsula al protocolo SIP.

Se podría nombrar también a los SIP Servlets como otra iniciativa asociada, pero fueron discutidos en un punto anterior.

Parlay

Discutido en secciones previas, el grupo Parlay fue formado en 1998 para especificar y promover APIs abiertas que integraran las aplicaciones de las tecnologías de la información con las capacidades del mundo de las telecomunicaciones. Los esfuerzos iniciales han sido enfocados en el control de las llamadas y la mensajería, si bien el foco primordial de Parlay es permitir a las aplicaciones acceder a las funcionalidades de las redes de telecomunicaciones de forma segura.

Las Parlay APIs consisten de dos categorías de interfaz:

- Interfaces de servicio: ofrecen a las aplicaciones acceso a determinada información y características de la red.
- Interfaces de framework: proveen las capacidades de soporte necesarias para que las interfaces de servicio sean seguras y manejables.

Las Parlay APIs están definidas en lenguaje UML (Unified Modeling Language). Las JAIN SPA (Service Provider APIs) definen un estándar de la industria en relación a las Parlay APIs que utiliza la tecnología Java. Además de las especificaciones Java API, las JAIN SPA

proveen implementaciones de referencia, suites de prueba y un programa completo de certificación.

Otra iniciativa relacionada con Parlay dice relación con Parlay X, lo cual se detallará más adelante en el apartado de Web Services.

2.4.4 P2PSIP (Peer-to-Peer SIP)

SIP está basado en un modelo cliente-servidor, donde cada entidad, dependiendo del contexto, se comporta como un cliente (UAC) o un servidor (UAS). Los modelos cliente-servidor tienen que superar un problema básico: la escalabilidad. Este problema, junto con el de la alta disponibilidad, puede ser resuelto utilizando técnicas Peer-to-Peer [17].

SIP es capaz de habilitar nuevas aplicaciones y es más efectivo en el despliegue de nuevos servicios, en comparación con aplicaciones Peer-to-Peer propietarias que están limitadas a su propósito original (p. ej. Skype). Sin embargo, las aplicaciones Peer-to-Peer han probado ser más escalables así como más resistentes a fallas y ataques.

Combinar SIP con técnicas Peer-to-Peer ofrece ventajas de ambos mundos, particularmente, supervivencia y escalabilidad. Hay dos maneras de combinar SIP y Peer-to-Peer:

a) Los UAs SIP pueden tener una alternativa a utilizar los DNS para sus búsquedas y traducciones de direcciones, utilizando una red distribuida de búsqueda (por ejemplo, un algoritmo de tablas con hash distribuido). Este mecanismo permite que los UA SIP funcionen sin Proxies SIP, pero su habilidad de alcanzar o ser alcanzados desde redes externas es limitada, además de realizar funciones a través de nodos no confiables. Así, se requiere modificar los UAs.
b) Los Proxies y Registrars SIP en una red overlay pueden tener la capacidad implícita de encontrar otros nodos y proveer las funciones necesarias a una gran cantidad de clientes SIP que dependen de aquellos. El acceso a DNS/ENUM y NAT traversal garantizan un servicio de mejor calidad que en el caso de UAs Peer-to-Peer no confiables (caso anterior). Este concepto no requiere cambios en los UAs existentes.

Una arquitectura que presente características de supervivencia puede ser implementada en la forma de un Proxy/Registrar P2P distribuido. Éste actúa como una red de súper-nodos, cuya arquitectura es escalable y resistente a fallas y ataques (pérdida de conectividad, ataques DoS, etc.). Nuevos nodos P2P se adhieren a la red instalando y ejecutando un software determinado, registrándose y pudiendo atender requerimientos SIP. Si el nodo falla, los requerimientos SIP son distribuidos automáticamente a nodos sobrevivientes, sin intervención manual alguna.

Por otro lado, la arquitectura puede ser escalada hasta administrar millones de suscriptores sin mermar su desempeño, simplemente agregando nodos a la red. A diferencia de P2P puro, el tiempo requerido para localizar un usuario y establecer una sesión interactiva es ínfimo (del orden de milisegundos).

La arquitectura está basada en URIs SIP, similares a una dirección de e-mail, que son buscadas y traducidas a través de requerimientos DNS estándar, siendo alcanzables globalmente dentro de la red e Internet, o bien pueden ser privadas.

En cuanto a los componentes de la arquitectura, se pueden nombrar los siguientes:

- Red overlay Peer-to-Peer
- DNS y ENUM
- Sistema de provisión
- Proxy/Registrar SIP
- Contabilidad mediante Call Data Record (CDR)
- NAT traversal utilizando "media relays" distribuidos
- User Agents (UAs)

El DNS y el sistema de provisión son los únicos componentes centralizados. El DNS garantiza la integridad y disponibilidad del nombre del dominio en Internet, mientras que el sistema de provisión se preocupa de que cualquier cambio sea hecho automáticamente, proporcionando además funciones de monitoreo. Todos los otros componentes están distribuidos en la red para proveer resiliencia y repartición de carga (flujo de datos).

2.5 Herramientas de Desarrollo

En esta sección se detallarán ciertas herramientas que permiten un desarrollo rápido y fácil de aplicaciones en un espectro amplio, es decir, no solamente basadas en SIP, además de técnicas adicionales para evaluar los servicios desarrollados.

2.5.1 SDP (Service Delivery Platform)

Para entregar los servicios mencionados anteriormente, se hace necesario un conjunto de componentes que permitan su desarrollo y despliegue. El término Plataforma de Entrega de Servicios [18] (Service Delivery Platform, SDP) dice relación con una arquitectura en telecomunicaciones pensada para permitir el desarrollo y despliegue rápido de nuevos servicios multimedia convergentes, desde servicios básicos en telefonía fija hasta servicios multimedia complejos que combinan audio y vídeo.

El auge de los sistemas de telecomunicaciones basados en IP frente a los sistemas actualmente desplegados (Legacy) ha incitado un cambio revolucionario en la orientación de la industria, desde sistemas propietarios hacia tecnologías abiertas, basadas en estándares. Así, se han abierto las puertas para que las compañías asociadas a las tecnologías de la información se acerquen al mundo de las telecomunicaciones, desarrollando SDPs alineados en la misma senda, así como soluciones de software aplicables a la arquitectura (tarificación convergente, manejo de la relación cliente/contenido, etc.).

A menudo, las SDPs presentan un directorio unificado de clientes que provee una arquitectura de información para todas las bases de datos de clientes existentes en la red convergente, incluyendo las de empresas, consumidores, clientes fijos, clientes móviles, etc. Las SDPs son empleadas para aplicaciones orientadas tanto a empresas como a consumidores comunes, siendo su uso en el primer caso centrado principalmente en la integración de capacidades de telecomunicaciones y de tecnologías de la información.

Los siguientes son los componentes básicos de una SDP:

- Ambiente de Creación de Servicios (Service Creation Environment, SCE)
- Ambiente de Control de Servicios (Service Control Environment)
- Ambiente de Dirección y Ejecución de Servicios
- Abstracciones para control de medios, presencia/localización, integración, y otras capacidades de comunicaciones de menor nivel

El punto de entrada es el SCE, usado por el desarrollador para crear software, scripts y otros recursos que implementen los servicios a desplegar. El propósito del SCE es facilitar la creación rápida de nuevos servicios de comunicaciones, es decir, lo mencionado anteriormente como una característica fundamental de las SDPs. Por ejemplo, el auge de SCEs basados en J2EE (Java) y SIP ha acelerado la adopción de SDPs específicos que integran esta característica. Así, programadores familiarizados con el lenguaje Java pueden desarrollar aplicaciones de telecomunicaciones usando J2EE, SIP y servicios web basados, por ejemplo, en Parlay.

Otro aspecto de las SDPs es que deben estar al tanto del punto de presencia del usuario, esto es, el lugar de acceso del cliente a los servicios convergentes donde sus preferencias y derechos son evaluados en tiempo real, asegurando que la información entregada al usuario corresponda a su contexto y predilección. La utilización de estándares como SIP y derivaciones de éste como SIMPLE en la arquitectura SDP empleada, facilitan el uso de esta característica y se vuelven un factor crítico en la implementación.

En cuanto a la integración, una solución SDP basada en estándares debe minimizar la necesidad de integración en tres áreas fundamentales:

1. Hacia los componentes de la red core subyacente
2. Entre aplicaciones de soporte tales como tarificación, activación de servicio y relación con los clientes
3. Servicios y aplicaciones de terceros

La implementación de SOA (Service-Oriented Architecture, Arquitectura Orientada a Servicios) en una solución de este tipo se esfuerza en minimizar las necesidades de integración utilizando servicios web e interfaces basadas en estándares. Las SOAs pueden ser usadas como una tecnología de integración de aplicaciones dentro de un SDP pero son mejor aprovechadas cuando son utilizadas en funciones de menor desempeño, tales como conexiones entre aplicaciones transaccionales (OSS, BSS) y la SDP.

2.5.2 IDE (Integrated Development Environment)

Un entorno de desarrollo integrado [19] (Integrated Development Environment, IDE) es un programa compuesto por un conjunto de herramientas mediante las cuales un programador puede desarrollar una aplicación específica. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI. Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes.

Los IDEs proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como Java, C, C++, C#, Visual Basic, etc. En algunos lenguajes, un IDE

puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación y probar las aplicaciones en forma interactiva, sin necesidad de trabajo orientado a archivos de texto. El lenguaje Visual Basic, por ejemplo, puede ser usado dentro de las aplicaciones de Microsoft Office, lo que hace posible escribir sentencias Visual Basic en forma de macros para Microsoft Word.

Es posible que un mismo IDE pueda funcionar con varios lenguajes de programación. Este es el caso de Eclipse, que mediante plug-ins se le puede añadir soporte de lenguajes adicionales. De hecho, es uno de los IDE más utilizados, junto con NetBeans, JBuilder, JCreator, JDeveloper, C++Builder, Turbo C y Turbo C++.

Componentes

- Un editor de texto.
- Un compilador.
- Un intérprete.
- Herramientas de automatización.
- Un depurador.
- Posibilidad de ofrecer un sistema de control de versiones.
- Factibilidad para ayudar en la construcción de GUIs.

2.5.3 Web Services

Un servicio Web [20] (Web Service) es una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios Web para intercambiar datos en redes de computadores, tales como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos. Las organizaciones OASIS y W3C son los comités responsables de la arquitectura y reglamentación de los servicios Web. Para mejorar la interoperabilidad entre distintas implementaciones de servicios Web se ha creado el organismo WS-I, encargado de desarrollar diversos perfiles para definir de manera más exhaustiva estos estándares.

Estándares empleados

- Web Services Protocol Stack: así se denomina al conjunto de servicios y protocolos de los servicios Web.
- XML (Extensible Markup Language): es el formato estándar para los datos que se vayan a intercambiar.
- SOAP (Simple Object Access Protocol) o XML-RPC (XML Remote Producer Call): protocolos sobre los que se establece el intercambio.
- Otros protocolos: los datos en XML también pueden enviarse de una aplicación a otra mediante protocolos normales como HTTP (Hypertext Transfer Protocol), FTP (File Transfer Protocol), o SMTP (Simple Mail Transfer Protocol).
- WSDL (Web Services Description Languages): es el lenguaje de la interfaz pública para los servicios Web. Es una descripción basada en XML de los requisitos funcionales necesarios para establecer una comunicación con los servicios Web.

- UDDI (Universal Description, Discovery and Integration): protocolo para publicar la información de los servicios Web. Permite a las aplicaciones comprobar qué servicios web están disponibles.
- WS-Security (Web Service Security): protocolo de seguridad aceptado como estándar por OASIS (Organization for the Advancement of Structured Information Standards). Garantiza la autenticación de los actores y la confidencialidad de los mensajes enviados.

Ventajas de los servicios Web

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.
- Los servicios Web fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.
- Al apoyarse en HTTP, los servicios Web pueden aprovecharse de los sistemas de seguridad como firewalls sin necesidad de cambiar las reglas de filtrado.
- Permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados.
- Permiten la interoperabilidad entre plataformas de distintos fabricantes por medio de protocolos estándar.

Inconvenientes de los servicios Web

- Para realizar transacciones no pueden compararse en su grado de desarrollo con los estándares abiertos de computación distribuida como CORBA.
- Su rendimiento es bajo si se compara con otros modelos de computación distribuida. Es uno de los inconvenientes derivados de adoptar un formato basado en texto, y se debe a que entre los objetivos de XML no se encuentra la brevedad ni la eficacia de procesamiento.
- Al apoyarse en HTTP, pueden esquivar medidas de seguridad basadas en firewall cuyas reglas tratan de bloquear o auditar la comunicación entre programas a ambos lados de la barrera.
- Existe poca información de servicios Web para algunos lenguajes de programación.
- Problemas con protocolos seguros como HTTPS.

Razones para crear servicios Web

La principal razón para usar servicios Web es que se basan en HTTP sobre TCP (Transmission Control Protocol) en el puerto 80. Dado que las organizaciones protegen sus redes mediante firewalls, que filtran y bloquean gran parte del tráfico de Internet, cierran casi todos los puertos TCP salvo el 80, que es, precisamente, el que usan los navegadores. Los servicios Web se enrutan por este puerto, por la simple razón de que no resultan bloqueados.

Otra razón es que, antes de que existiera SOAP, no había buenas interfaces para acceder a las funcionalidades de otros computadores en red. Las que había eran ad-hoc y poco conocidas, tales como algunas APIs privadas.

Una tercera razón por la cual los servicios Web son muy prácticos es que pueden aportar gran independencia entre la aplicación que usa el servicio Web y el propio servicio. De esta forma, los cambios a lo largo del tiempo en uno no deben afectar al otro. Esta flexibilidad será cada vez más importante, dado que la tendencia a construir grandes aplicaciones a partir de componentes distribuidos más pequeños es cada día más usada.

2.5.3.1 Parlay X

Parlay X [1] es un conjunto de Web Services que permiten el desarrollo de aplicaciones orientadas a las telecomunicaciones. Su alto nivel de abstracción y el uso de tecnologías derivadas del mundo de las Tecnologías de la Información, como son los Web Services, permiten que el desarrollo de estas aplicaciones quede abierto también a los desarrolladores de TI, quienes no son necesariamente expertos en redes ni en los protocolos utilizados en telecomunicaciones.

Parlay X no depende directamente de Parlay; puede invocar funciones a través del API Parlay pero también puede manejar interfaces directamente con las plataformas de servicios o los elementos de red subyacentes, tal como muestra la siguiente figura:

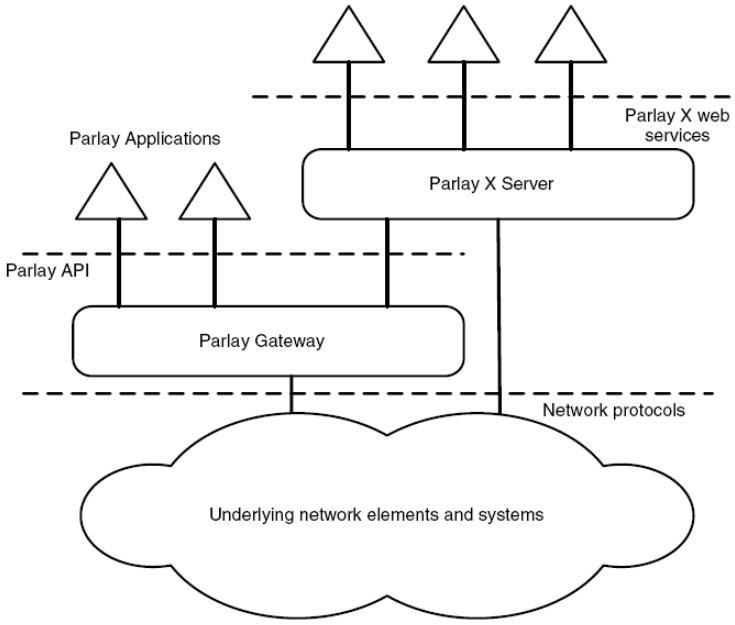

Figura 2-20 [1]: Arquitectura Parlay X

Las aplicaciones que hacen uso de las APIs Parlay X pueden ser escritas en una variedad de lenguajes de programación, dado que están basados en Web Services, utilizando así interfaces definidas en XML y transmitidas mediante el protocolo SOAP.

Los estándares actuales definen 14 APIs de Web Services Parlay X, los cuales se enuncian a continuación:

- Common (define objetos y servicios comunes a múltiples APIs)
- Third Party Call Control (control de la llamada por terceras partes)
- Call Notification (notificación de llamada)
- Short Message (mensajería corta)
- Multimedia Message (mensajería multimedia)
- Payment (tarificación)
- Account Management (gestión de cuentas)
- Terminal Status (estatus del terminal)
- Terminal Location (localización del terminal)

- Call Handling (manejo de la llamada)
- Audio Call (llamada de voz)
- Multimedia conference (conferencia multimedia)
- Address List Management (gestión de lista de direcciones)
- Presence (presencia)

2.5.4 Test Plan

Como técnica a utilizar en el trabajo, se puede nombrar el desarrollo y ejecución de un Test Plan. Un Test Plan es una aproximación sistemática a probar el desempeño de un sistema, tal como una máquina o un software. El plan contiene típicamente una descripción detallada del flujo de los procesos que lo componen. El propósito de un Test Plan es especificar quién hace qué, cuándo lo hace y por qué lo hace, en las etapas de diseño, construcción, ejecución y análisis de un test. El Test Plan también describe el ambiente de pruebas y los recursos que se requieren para llevarlo a cabo. Más aun, facilita las comunicaciones entre el equipo de pruebas, entre estos últimos y el equipo desarrollador, y entre el equipo de pruebas y los directivos.

El Test Plan es un documento operacional que es la base para las pruebas, describiendo estrategias y casos de pruebas. Aunque su naturaleza es operacional, presenta también aspectos administrativos, por lo que se puede usar tanto para nuevos desarrollos como para mantenimiento. Cabe destacar que se trata de un documento en constante evolución.

Un Test Plan debería ser construido a partir de una plantilla estandarizada. Hay numerosas fuentes que contienen dichas plantillas; en el caso de software existe el estándar IEEE 829-1998, “A Standard for Testing Application Software” (Perry) y “The Handbook of MIS Application Software Testing” (Mosley). Es también aceptable combinar partes de dichos documentos para construir un estándar híbrido para Test Plans. Así, las partes fundamentales de un Test Plan son las siguientes:

- Alcance de las pruebas
- Cronograma
- Documentos a entregar
- Detalle de las pruebas a realizar
- Criterios de aprobación
- Riesgos y contingencias

Capítulo 3

Desarrollo

El presente capítulo muestra el trabajo realizado en cuanto a metodología, implementación de la plataforma y resultados obtenidos de los servicios desplegados.

3.1 Evaluación de Alternativas de Implementación

Para empezar, se debió seleccionar entre diferentes alternativas un software correspondiente a un AS que manejara los requerimientos y respuestas SIP necesarios para proveer los servicios. Con este fin, tres programas candidato se compararon en cuanto a funcionalidades, lo que se detalla a continuación, para luego elegir fundamentadamente cuál de ellos utilizar.

3.1.1 Características de las Diferentes Alternativas

OpenSER [22]

- Servidor SIP (RFC3261) robusto y de alto desempeño: servidor de registro (Registrar), localización (Location), redirección (Redirect) y Proxy.
- Código breve, funcionalidad puede ser ampliada mediante módulos (existe repositorio con más de 70 módulos).
- Interfaz de módulos "Plug&Play": habilidad de agregar nuevas extensiones sin modificar el core, asegurando una gran estabilidad de los componentes del core.
- Procesamiento del SIP Proxy puede ser "stateless" (sin estados) o "stateful" (con estados)
- Soporte para UDP/TCP/TLS (capas de transporte), IPv4 e IPv6 (con bloqueo de IPs por blacklists), DNS (SRV, NAPTR, ENUM), CPL (RFC3880), dominios múltiples, balance de carga, enrutamiento por mínimo costo.
- Lenguaje de scripting para los archivos de configuración: forma poderosa y flexible de desplegar servicios SIP ajustados a requerimientos particulares.
- Interfaz de mantenimiento vía archivo FIFO y Sockets UNIX, soporte para SNMP y XMLRPC.
- Autentificación, Autorización y Contabilidad (AAA) vía base de datos (mySQL, PostgreSQL, UnixODBC, archivos de texto), RADIUS y DIAMETER.
- Soporte de NAT traversal, para tráfico SIP y RTP.
- Interfaz para programación en PERL, interfaz para Java SIP Servlets: aplicaciones y extensiones embebidas, extensión de servicios VoIP e integración con Web Services.
- Gateway para SMS o XMPP, interconexión directa con gateways PSTN.
- Escalabilidad: capacidad de agregar servidores OpenSER adicionales.

- Redundancia directa, capacidad de operar en plataformas VoIP distribuidas.
- Puede ejecutarse en sistemas embebidos, con recursos limitados.
- Con balance de carga y en modo "stateless", puede manejar hasta 5000 peticiones de establecimiento de llamada por segundo; en sistemas con alta memoria, puede servir a más de 300000 suscriptores.

Ubiquity [23]

Plataformas Soportadas		
Sistema Operativo	• Solaris 8, 9, 10; Windows Server 2003 Enterprise Edition; Windows 2000 Advanced Server; Redhat Enterprise Linux 4; SuSE Linux ES 8, 9	
Java	• JRE 1.4.2, 1.5	
Base de Datos	• IBM DB2 UDB v8	• Oracle 9i, 10g
	• MySQL 5.0 Database Server – Community Edition (soportado en UDE Developer Edition 1.2 y posterior solamente)	
Hardware	• Servidores comerciales de HP, IBM, Intel, Sun, y otros	
Infraestructura de Red e Interoperabilidad		
Media Server	• Cantata SnowShore IP Media Server, Convedia CMS-1000/6000, HP OpenCall Media Platform	
J2EE Application Server	• Cualquier J2EE Application Server que soporte RMI o SOAP (JBoss App Server, BEA WLS, IBM Websphere)	
Session Border Controller	• Netrake, Newport Networks	
Media Gateways/PSTN	• Cisco 5350, proveedores de terminaciones PSTN Tier 1 vía conexión SIP directa	
3GPP IMS	• RFC 3588 Base Diameter Stack; interfaces Service Control (ISC), Ro, Rf, Sh y APIs; generador de ICID, AVP, CDR, CTF, CDF; Registration Data Reader	
Ambiente de Ejecución de Servicios		
SIP A/S Service Host	• JAIN SIP Servlet Container que cumpla la norma JSR 116	
SIP SOA Application Framework	• Extensión opcional para el contenedor JSR 116; framework de desarrollo y despliegue de aplicaciones basado en SOA	
External Application Environment	• Interfaces RMI y SOAP a plataformas de aplicación externas; interoperabilidad con Microsoft Connected Services Framework	
Desempeño y Escalabilidad		
Escalabilidad	• Escalamiento horizontal mediante agregación de más servidores o canales por cluster • Escalamiento vertical mediante agregación de más CPUs y memoria por servidor	
Desempeño	• Desempeño (llamadas por segundo) se incrementa linealmente con servidores/canales adicionales; miles de llamadas por segundo por cluster	
Clustering y Alta Disponibilidad		
Balance de Carga	• Balance de carga a través de múltiples SIP Service Hosts o Canales	

	en un cluster vía Service Director
Alta Disponibilidad	• Disponibilidad del servicio del 99.999%
Continuidad del Servicio	• Protección de fallas e intercambio de servicio para asegurar que las sesiones SIP se mantengan abiertas si un elemento de servicio falla • Actualizaciones "en caliente"
Desarrollo de Aplicaciones y Creación de Servicios	
Ubiquity Developer Studio	• Plug-ins para ID Eclipse que aceleran el desarrollo de servicios en el SIP A/S
Ubiquity Developer Edition	• Versión del SIP A/S para laptop o PC
Componentes de Servicio SIP SOA	• IVR, Conferencing, Media Server Control, SIP Dialog Control • Parlay X 2.0 Open Web Services: Third Party Call Control, Call Notification, Call Handling, Audio Call, Multimedia Conference
Lenguajes de Desarrollo de Servicios	• Java (SIP Servlets), Java (SOA Service Components), BPEL (Service Components), Web Services (SOAP/WSDL) vía API Parlay X 2.0
JSR 116 SIP Servlet API	• Framework de programación basada en estándares
Plantillas y Scripts ANT	• Integración con la herramienta de construcción de software ANT (estándar de la industria)
IMS APIs	• Diameter, interfaz Sh, IMS Charging Identifiers (ICIDs), Registration Data Reader, Attribute-Value Pairs (AVP), interfaz Ro, interfaz Rf, CDRs
Eventlet API	• Implementar triggers no-SIP (RMI, SOAP, JNDI) para aplicaciones SIP
Microsoft Connected Services Framework	• CSF Gateway — expone OWS como Microsoft Well Enabled Services (WES) para incorporar control de llamada en tiempo real a aplicaciones CSF
Microsoft .NET	• SIP Objects.NET de Inova IT—habilita acceso a Open Web Services de Visual Studio a través de una API simple, intuitiva, orientada a .NET y herramientas visuales basadas en Windows Workflow Foundation • SIP Servlets.NET de Inova IT— provee acceso al API de SIP Servlets JSR116 API y habilita a un desarrollador de .NET para construir aplicaciones SIP Servlet utilizando el lenguaje .NET de su gusto (por ejemplo C#) y Visual Studio
Operación, Administración, Mantenimiento y Provisión	
Gestión y Provisión	• Ubiquity Element Manager GUI • Ubiquity Managemenet Server • Instalaciones Silenciosas

Tarificación	• 3GPP IMS Diameter, Ro, Rf, ICID, generación de CDR, CTF, CDF • Logging Extraction Service
Gestión de Red	• API de sistema de mantenimiento basada en JMX • SNMPv2c • Conector que cumple la norma JSR 160
Seguridad	• TLS, SIP DIGEST • Mecanismos SIP estándar para autenticación salto a salto • Contraseñas cifradas para gestión y administración de clusters
Estándares SIP Soportados	
SIP RFCs	• 2246, 2327, 2543, 2782, 2806, 2976, 3261, 3262, 3263, 3264, 3265, 3266, 3311, 3312, 3323, 3326, 3420, 3428, 3455, 3515, 3556, 3588, 3665, 3725, 3856, 3863, 3880, 3856, 3891, 3911, 3966, 4028

GlassFish [24]

La comunidad GlassFish implementa un AS de código libre basado en Java EE 5. GlassFish es un AS robusto, comercial, de calidad de producción y compatible, cuyo desarrollo, despliegue y redistribución es libre. Algunas de las razones para utilizar GlassFish se detallan a continuación:

- GlassFish es una implementación de Java EE 5 de código libre y basada en una comunidad de desarrollo (Java EE 5 es una gran mejora sobre versiones anteriores de J2EE).
- GlassFish es de calidad de producción, de buen desempeño y es escalable.
- GlassFish se ejecuta sobre SJS AS 9.0, Java EE 5 SDK, y NetBeans 5.5.
- GlassFish ya ha sido adoptado por desarrolladores y está captando la atención de los analistas.
- GlassFish es utilizado por otros grupos, tales como JEUS 6 Preview de TMaxSoft.
- GlassFish presenta variadas características, incluyendo Java DB, Java Blueprints y ejemplos AJAX.
- GlassFish se integra a su framework y aplicación favoritos.
- GlassFish tiene una extensa documentación, además de soporte para herramientas de migración.
- GlassFish tiene una consola de administración y soporte de seguridad de calidad de producción.
- GlassFish está a la vanguardia de la adopción de JPA (Java Persistence API).
- GlassFish presenta un stack de Web Services de calidad y en constante mejora.
- GlassFish a través de Grizzly entrega desempeño y flexibilidad de primer nivel (Web Tier).
- GlassFish tiene soporte para desarrolladores, servicios de software, entrenamiento, clases, foros, blogs y más.
- GlassFish soporta AJAX y scripting.
- GlassFish tiene soporte para plataformas y herramientas SOA, incluyendo JBI y BPEL.

- GlassFish es utilizado directamente para la implementación de referencia oficial de Java EE 5.
- GlassFish es el código base para el Sun Java System Application Server.
- GlassFish incorporará las caracterísiticas disponibles hoy en SJS AS 8.x.
- Muchos componentes se encuentran el repositorio Maven, y pronto habrán más.

3.1.2 Click-to-Dial en las Diferentes Alternativas

Para ejemplificar aun más las diferencias entre las alternativas de implementación, se presenta a continuación de forma general la manera de proveer el servicio de Click-to-Dial en cada una de ellas.

OpenSER + WeSIP [25]

El usuario de la aplicación utiliza una página web o un servicio web para introducir la dirección del usuario que desea llamar, así como los detalles (nombre de usuario, contraseña, dominio) de la cuenta SIP que será usada para efectuar la llamada. Con toda esta información, la aplicación inicia una sesión 3PCC (Third Party Call Control, RFC 3725) que primero contacta a quien llama. Cuando quien llama contesta, se contacta separadamente a quien se desea llamar. Cuando éste contesta, se conectan ambos enlaces usando técnicas estándar de 3PCC. El cliente (usuario de la página web o cliente de servicio web) interroga sobre el estado de ambos enlaces usando llamadas HTTP o SOAP repetidamente.

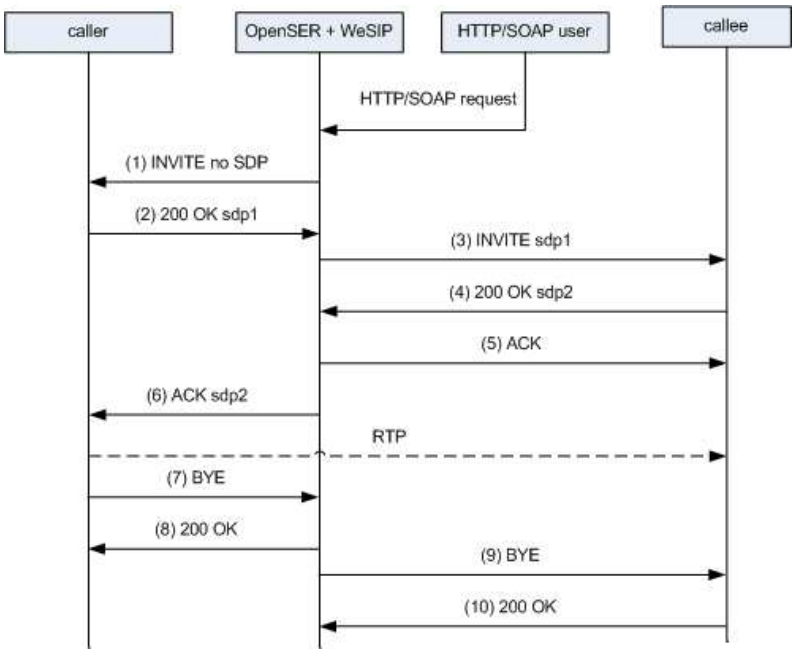

Figura 3-1 [25]: Flujo C2D WeSIP

La aplicación Click-to-Dial sigue el estándar 3pcc RFC 3725 para control de llamadas de terceros. El modelo elegido del RFC es el "Flow 1" debido a su simplicidad. El diagrama muestra los eventos de señalización más importantes:

En la página web de WeSIP se entrega un código que implementa la aplicación Click-to-Dial usando dos tipos de interfaz. Una de ellas es una página web en AJAX que utiliza DWR (Direct Web Remoting). La segunda es un servicio web implementado con AXIS. El código y los archivos necesarios se encuentran en el siguiente link:
http://www.wesip.eu/mediawiki/index.php/Special:UserDownload?action=download&download=click2call-08_11_06.sar

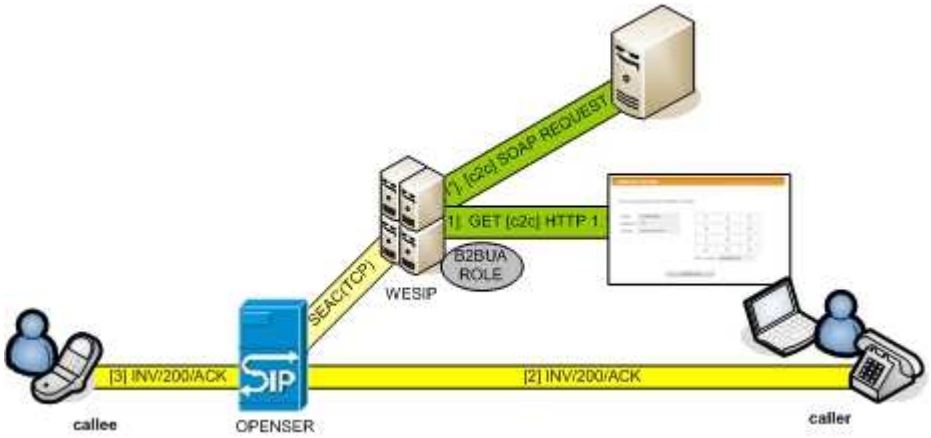

Figura 3-2 [25]: Arquitectura C2D WeSIP

Ubiquity [23]

Esta plataforma hace uso de los llamados Open Web Services (OWS), que proveen una interfaz externa estándar al SIP AS y a los Service Components que residen en él. Los OWS están basados en la especificación Parlay X, lo que permite a aplicaciones externas hacer uso de servicios de telecomunicaciones disponibles en el SIP AS mediante una interfaz abierta, basada en SOAP. En este caso, también se hace uso de un contenedor Java (JEE), llamado JBoss, que se comunica con el contenedor de SIP Servlets del AS.

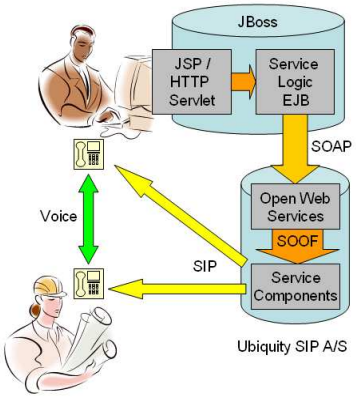

Figura 3-3 [23]: Arquitectura C2D Ubiquity

El usuario de la parte superior utiliza cualquier dispositivo compatible con Web para acceder a su libreta de direcciones centralizada. La capa de presentación de la libreta de direcciones está representada mediante JSP y CSS residente en Tomcat, que en sí pertenecen a JBoss. El JSP invocado por el usuario invocará a su vez a un EJB residente en la plataforma JBoss. Este último será responsable de proveer una lista de contactos contenidos en la libreta de direcciones del usuario. Adicionalmente, permite que el JSP pueda iniciar una llamada a un

usuario en la libreta, invocando un servicio Web residente en el SIP AS, que a su vez invoca los métodos apropiados en los Service Components vía SOOF. Estos componentes son responsables de generar la señalización apropiada para llevar a cabo la llamada.

Cada contacto será mostrado de una manera definida por el JSP. Al hacer click sobre una dirección de contacto, el EJB de la lista de contactos invocará a los servicios residentes en el SIP AS para establecer una llamada entre el contacto y la dirección del usuario local (especificada también en el formulario Web). Del punto de vista del usuario, hacer click en su página causará que su teléfono suene, y al contestar estará comunicado con el contacto en el cual hizo click.

Un aspecto importante de esta aplicación es que no necesita demasiadas líneas de código para integrar estos componentes. El código y los archivos necesarios se encuentran en el siguiente link:
https://developer.ubiquitysoftware.com/support/tutorials/web-services-tutorials/resolveuid/cc419ba139979ccab1c93a192c0f746c

Java GlassFish + SailFin [26]

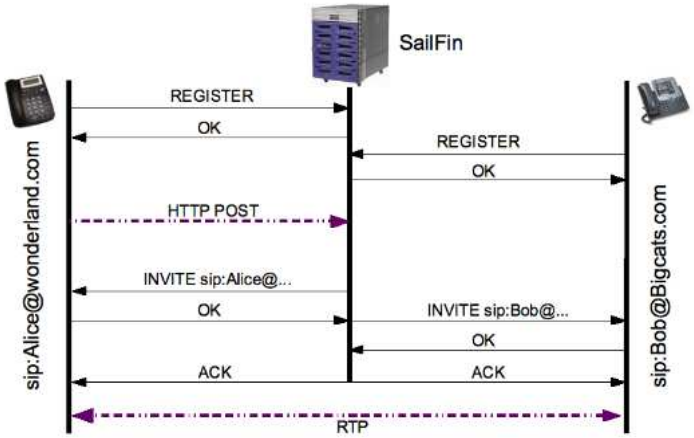

Figura 3-4 [26]: Flujo C2D GlassFish

En este caso, GlassFish utiliza un contenedor de SIP Servlets llamado SailFin. Los pasos que sigue la aplicación son los siguientes:

- Alice y Bob se conectan cada uno a la aplicación web
- Alice y Bob registran cada uno un Softphone SIP
- Alice hace click en el enlace “Call” para llamar a Bob
- El teléfono de Alice suena
- Cuando Alice contesta, suena el teléfono de Bob
- Cuando Bob contesta, se lleva a cabo la conexión
- Cuando uno de ellos cuelga, el otro también es desconectado

Un esquema de los mensajes SIP intercambiados se presenta a continuación:

Para manejar estos mensajes, se utilizan dos SIP Servlets y un HTTP Servlet, cuyas características se presentan a continuación:

Registrar SIP Servlet

Este SIP Servlet maneja el registro de los softphones. Cuando Alice o Bob inician su softphone, éste registra su dirección con un Proxy SIP. En este caso, se graba la información de contacto de los softphones usando la API Java Persistence.

PlaceCall HTTP Servlet

Este HTTP Servlet recibe información HTTP cuando Alice hace click en el enlace "Call" en su navegador. Utilizando esta información, envía el INVITE inicial a Alice y almacena la dirección de Bob.

Call SIP Servlet

Este SIP Servlet maneja la respuesta OK al INVITE enviado por el Servlet anterior. Una vez que un mensaje OK de Alice es recibido, el Servlet envía un INVITE a Bob, especificando las preferencias de medios de Alice. Cuando se recibe un OK de Bob, el Servlet envía un ACK a ambas partes. También maneja los mensajes BYE y la disolución de la llamada.

El código y los archivos necesarios se encuentran en el siguiente link:
http://wiki.glassfish.java.net/attach/SipClickToDialExample2/ClickToDial.zip

3.1.3 Elección de Software

Analizando los puntos anteriores, se puede ver que es factible desplegar una arquitectura simple para proveer servicios, tales como Click-to-Dial, mediante las tres opciones de implementación estudiadas. Sin embargo, hay dos de ellas que presentan serios obstáculos en relación a los objetivos del proyecto, lo cual se analizará a continuación.

Para empezar, la alternativa de Ubiquity es de tipo propietario, es decir, su código no está abierto para su modificación y/o agregación de características por parte de terceros. Esto representa un claro impedimento ya que se hace difícil desarrollar soluciones que se adapten a requerimientos específicos o que requieran de ligeras modificaciones al software para poder ser provistas.

Por otro lado, la opción de GlassFish, si bien representa una alternativa de código libre y con alto respaldo basado en comunidades, no presenta una conexión intuitiva con los sistemas de AAA y tarificación, lo que constituye una desventaja potente del punto de vista del operador. Además, su orientación hacia Java lo hace poco flexible frente a otras tecnologías y opciones de implementación, siendo incierta su interconexión con otros sistemas desplegados.

Así, finalmente, la opción será utilizar OpenSER como SIP Server junto con WeSIP como B2BUA y Application Server, ya que presentan la combinación óptima de flexibilidad, escalabilidad e interoperabilidad, además de ser fácilmente modificables y adaptables a requerimientos específicos (en el caso de OpenSER).

3.2 Implementación

En esta sección se detallará la instalación de las aplicaciones y los resultados que se obtuvieron durante el despliegue de los servicios.

3.2.1 Topología Utilizada

La topología de la red que se pensó inicialmente para los trabajos se presenta en la siguiente figura:

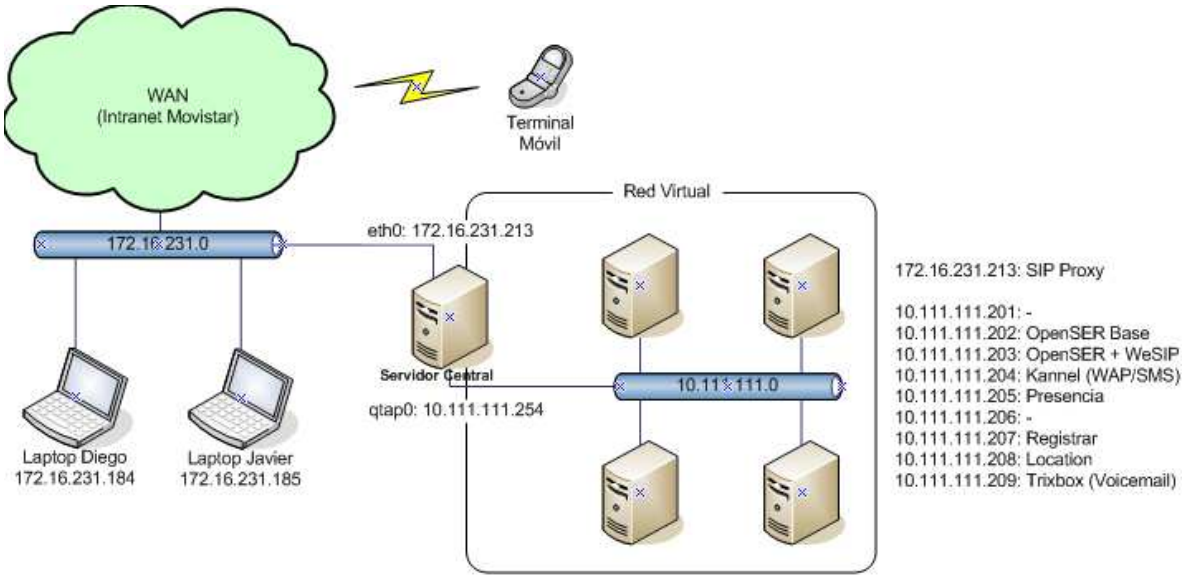

Figura 3-5: Topología Inicial de la Plataforma

Como se puede apreciar, la arquitectura consiste en varios servidores virtuales corriendo en una máquina host. La decisión de virtualizar dice relación con la escasez de equipos disponibles.

Durante el desarrollo del trabajo, se fueron efectuando modificaciones que favorecieron la implementación de la plataforma. Por ejemplo, el servidor de presencia se trasladó hacia el host central. Como en dicha máquina también se ejecutaba el core IMS, se optó por diferenciar los servidores por puertos; el servidor de presencia pasó a tener el puerto 5065 en la máquina host, mientras que el core implementado utilizó los puertos 4060 (P-CSCF), 5060 (I-CSCF) y 6060 (S-CSCF). Además, los servidores Registrar y Location distribuidos se colocaron también en el host, corriendo en la misma configuración que el servidor de presencia. La forma de configurar OpenSER se detallará a continuación.

3.2.2 Instalación y Configuración Básica

Se realizó la instalación [27] de OpenSER 1.2.2 sin soporte para TLS, en un sistema Linux con la distribución Ubuntu 7.10 “Gutsy Gibbon”. Algunos paquetes deben ser instalados o actualizados a su última versión antes de proceder con la compilación e instalación del programa. Estos paquetes son los siguientes:

- gcc o icc
- bison o yacc
- flex
- GNU make, install, tar y gzip

- mysqlclient y zlib (lib y devel) para el soporte de bases de datos MySQL (opcionalmente se puede instalar PostgreSQL u ODBC)
- Otras librerías opcionales para soporte de Jabber, CPL y Presence Agent

En este caso particular se tuvo que instalar bison, flex, openssl, mysql (cliente y devel) y zlib (devel).

El archivo que contiene los binarios de OpenSER se debe descomprimir usando `tar`, y luego se ingresa a la carpeta:

Tabla 3: Instalación OpenSER (1)

```
tar -xzvf openser-1.2.2-notls_src.tar.gz
cd openser-1.2.2-notls
```

Luego se compila e instala el programa junto a todos sus módulos como root o super user (su), usando `prefix=/` si se quiere instalar en el directorio principal del sistema de archivos, o sin usar `prefix` si se desea colocar en `/usr/local`. En el primer caso se ejecutan las siguientes instrucciones:

Tabla 4: Instalación OpenSER (2)

```
sudo make prefix=/ all
sudo make prefix=/ install
```

El problema es que ni el módulo de MySQL ni el de presencia se compilan ni instalan por defecto, por lo que se debe ejecutar lo siguiente:

Tabla 5: Instalación OpenSER (3)

```
sudo make prefix=/ modules=modules/mysql modules
sudo make prefix=/ modules=modules/presence modules
sudo make prefix=/ install
```

Las cinco instrucciones anteriores pueden reemplazarse por las dos siguientes, que incluyen los módulos de MySQL y presencia en la compilación e instalación:

Tabla 6: Instalación OpenSER (4)

```
sudo make prefix=/ all include_modules="mysql presence"
sudo make prefix=/ install include_modules="mysql presence"
```

Así, los módulos `mysql.so` y `presence.so` estarán instalados en la carpeta `/lib/openser/modules`.

El siguiente paso es configurar MySQL, empezando por establecer una contraseña para la cuenta root:

Tabla 7: Instalación OpenSER (5)

```
mysql -u root
update mysql.user set Password = PASSWORD('password') where User =
'root';
flush privileges;
exit;
```

Luego se debe crear la base de datos de OpenSER mediante el script `openser_mysql.sh` ubicado en `/sbin`:

Tabla 8: Instalación OpenSER (6)

```
cd /sbin
openser_mysql.sh create
```

Durante la creación, se pedirá un nombre de usuario, contraseña, y el dominio SIP ("domain/realm") al cual servirá el servidor. En este caso se utilizó como dominio la IP del PC en el cual se instaló el software, ya que las pruebas se realizarían en la red local. Aun así, se pueden agregar dominios manualmente en otra ocasión.

El siguiente paso es abrir el archivo de configuración `/etc/openser/openser.cfg` con un editor de texto, para editarlo antes de su primera ejecución, reemplazando "`openser.org`" por el dominio especificado anteriormente. Este archivo de configuración permite el registro de usuarios contra el servidor SIP sin autenticar; si se desea realizar autenticación de los usuarios al momento del registro, se debe quitar el símbolo de comentario (#) a las siguientes líneas:

Tabla 9: Instalación OpenSER (7)

```
- loadmodule "/usr/lib/openser/modules/mysql.so"

- loadmodule "/usr/lib/openser/modules/auth.so"

- loadmodule "/usr/lib/openser/modules/auth_db.so"

- modparam("usrloc", "db_mode", 2)

- modparam("auth", "calculate_ha1", yes)

- modparam("auth_db", "password_column", "password")

- if (!www_authorize("dominio_o_ip_del_host", "subscriber")) {
- www_challenge("dominio_o_ip_del_host", "0");
- break;
- };
```

Nótese que en las últimas líneas se debe ingresar nuevamente el dominio al que servirá el servidor SIP o en su defecto la IP del host, si se ejecuta para servir un entorno local. Además, se debe comentar la siguiente línea (agregando # al principio):

Tabla 10: Instalación OpenSER (8)

```
modparam("usrloc", "db_mode", 0)
```

Cabe destacar que todo este proceso de generación de configuraciones se puede automatizar utilizando una herramienta disponible en http://www.sipwise.com/wizard/, la cual genera un archivo de configuración completo de acuerdo a las características que se elijan.

A continuación, se debe editar el archivo `/etc/openser/openserctlrc`, quitando el símbolo de comentario a todas las líneas excepto a la que dice `OSER_FIFO="FIFO"`, para que funcione el soporte para MySQL. Además, para no estar introduciendo la contraseña cada vez que se ejecuta `openserctl`, se puede agregar la línea `DBRWPW=openserrw`. Cabe destacar que se

están usando las contraseñas por defecto, las cuales deben cambiarse lo antes posible para evitar riesgos.

Después de este paso, se puede ejecutar OpenSER mediante el comando `openserctl start`, o directamente mediante el comando `openser`. Cabe mencionar que éste se ejecuta de acuerdo a la configuración establecida en `openser.cfg`; si se requiere utilizar otro archivo de configuración, por ejemplo `config.cfg`, se debe ejecutar el comando `openser -f config.cfg`. Para agregar un par de usuarios a la base de datos mediante MySQL y revisar la configuración FIFO, se pueden ejecutar los siguientes comandos:

Tabla 11: Instalación OpenSER (9)

```
openserctl add user1 user1pw user1@localhost
openserctl add user2 user2pw user2@localhost
openserctl moni
```

Así, en la base de datos se tienen tres usuarios: admin, user1 y user2. A estas alturas se puede utilizar un cliente SIP para efectuar un registro contra OpenSER usando alguno de estos tres usuarios, para poder comenzar a utilizar servicios basados en SIP. Por ejemplo, utilizando el programa X-Lite, se puede establecer la siguiente configuración básica para el registro:

Tabla 12: Instalación OpenSER (10)

```
Username: user1
Password: user1pw
Authorization user name: user1 (se puede dejar en blanco)
Domain/Realm: dominio.com
```

La configuración básica de OpenSER con autenticación se adjunta en el Anexo 2. Cabe destacar que se deben reemplazar los parámetros `dominio.com`, `ip_registrar`, `puerto_registrar` e `ip_location` por los valores correspondientes.

3.2.3 Aplicaciones

En esta sección se detallará el despliegue de los diferentes servicios basados en SIP que corresponden a la implementación de la plataforma de aplicaciones.

3.2.3.1 Servicios de Voz

La configuración básica de OpenSER vista en la sección 3.2.2 permite manejar sesiones de telefonía IP, por defecto. El servidor SIP actúa como Proxy, redirigiendo hacia el usuario destino el mensaje INVITE que llega desde el usuario origen, mientras que la voz viaja como un flujo RTP Peer-to-Peer.

Por otro lado, para proveer el servicio de Voicemail, se recurre al software Asterisk. Específicamente, se utiliza un software llamado trixbox CE (Community Edition) [28], el cual además de Asterisk incluye una serie de herramientas complementarias, en adición a una interfaz gráfica basada en Web accesible desde cualquier navegador. La principal característica de trixbox CE es que corresponde a un software de código abierto; existen a su vez versiones propietarias de características más completas.

Para proveer el servicio, es necesario hacer ciertas modificaciones al archivo de configuración básico de OpenSER detallado en el anexo 2 [29]. Si solamente se desea proveer el servicio de Voicemail, se pueden omitir o comentar las líneas del archivo de configuración que correspondan al manejo de otros tipos de requerimiento. Esto resultaría en agregar las siguientes líneas al archivo mencionado, en los lugares adecuados y reemplazando `ip_trixbox` por el valor correspondiente:

Tabla 13: Líneas Adicionales, Configuración Voicemail

```
route[3] {
        # ----------------------------------------------------------------
        # INVITE Message Handler
        # ----------------------------------------------------------------
…

        # Redireccionar hacia Voicemail

        # Comprobamos si no esta marcado el flag 10, para reenviar a
Voicemail si da error

        if(!isflagset(10)) {
                t_on_failure("1");
        };
# Servidor Voicemail
        if(uri=~"sip:\*98@.*") {
                route(4);
        }
# Casilla Voicemail
        if(uri=~"sip:\*981@.*") {

                # Si ya tiene la cabecera se la quito

                if(is_present_hf("X-VM")) {
                        remove_hf("X-VM");
                }
                append_hf("X-VM: $fU\r\n");
                route(4);
        }


…
}
```

```
# --------------------------
# Redireccion hacia Voicemail
# --------------------------

failure_route[1] {
        if(!t_was_cancelled()) {
                if (t_check_status("(486)|(408)")) {
                        rewritehostport("ip_trixbox:5060");
                        append_branch();
                        # Activamos el flag 10 para evitar bucles
                                setflag(10);
                        t_relay();
                        exit;
                }
        }
}
```

Cabe destacar que la lógica implementada captura los errores de comunicación, reconociendo éstos mediante un flag particular del mensaje y redirigiéndolos hacia el servidor de Voicemail. Así, es posible soportar desvíos de los siguientes tipos: CFNA (Call Forwarding No Answer), CFNR (Call Forwarding on No Reply) y CFB (Call Forwarding Busy); en este caso la lógica no posibilita la redirección del tipo CFU (Call Forwarding Unconditional), pues no se trata de un error de la llamada sino un desvío predeterminado para todas las llamadas.

3.2.3.2 Presencia y Mensajería Instantánea

La configuración básica de OpenSER vista en la sección 3.2.2 permite manejar sesiones de mensajería instantánea, por defecto, ya que actúa como Proxy redirigiendo los mensajes MESSAGE hacia el destino requerido.

Por otra parte, para proveer el servicio de presencia, es necesario realizar modificaciones al archivo de configuración básico de OpenSER del anexo 2 [30]. Cabe destacar que si sólo se desea proveer el servicio de presencia, las líneas del archivo de configuración correspondientes al manejo de registro, mensajes INVITE y otros pueden ser comentadas u omitidas. Esto deriva en la adición de las siguientes líneas al archivo de configuración mencionado, en las posiciones respectivas y reemplazando `ip_db` e `ip_servidor` por los correspondientes valores:

Tabla 14: Líneas Adicionales, Configuración Presencia

```
loadmodule "presence.so"
```

```
# -- presence params --
modparam("presence", "db_url", "mysql://openser:openserrw@ip_db/openser")
modparam("presence", "force_active", 1)
modparam("presence", "max_expires", 3600)
modparam("presence", "server_address", "sip:ip_servidor:5065")
```

```
route{
        …

        # presence handling
        if(method=="PUBLISH"){
                route(2);
        }

        if(method=="SUBSCRIBE"){
                route(2);
        }

        …
}
```

```
# presence handling route
route[2]
{
        # absorb retransmissions
        if (! t_newtran())
        {
                sl_reply_error();
                exit;
        };

        if(is_method("PUBLISH"))
```

```
        {
                handle_publish();
                t_release();
        } else if( is_method("SUBSCRIBE")) {
                handle_subscribe();
                t_release();
        };

        exit;
}
```

3.2.3.3 Otros Servicios

Aplicaciones de distinta naturaleza a las presentadas anteriormente fueron provistas de la siguiente manera:

C2D (Click-to-Dial)

Se optó por utilizar el programa UCT B2BUA [31], el cual envía los mensajes INVITE a dos usuarios registrados contra un Proxy SIP o contra un core IMS, para luego contactar a ambos en una sesión P2P. El programa se instala mediante el paquete Debian `uctb2bua`, el cual a su vez instala los paquetes adicionales necesarios (fundamentalmente `libexosip`). La forma más fácil de ejecutar el programa es vía Web, como muestra la figura 3-7.

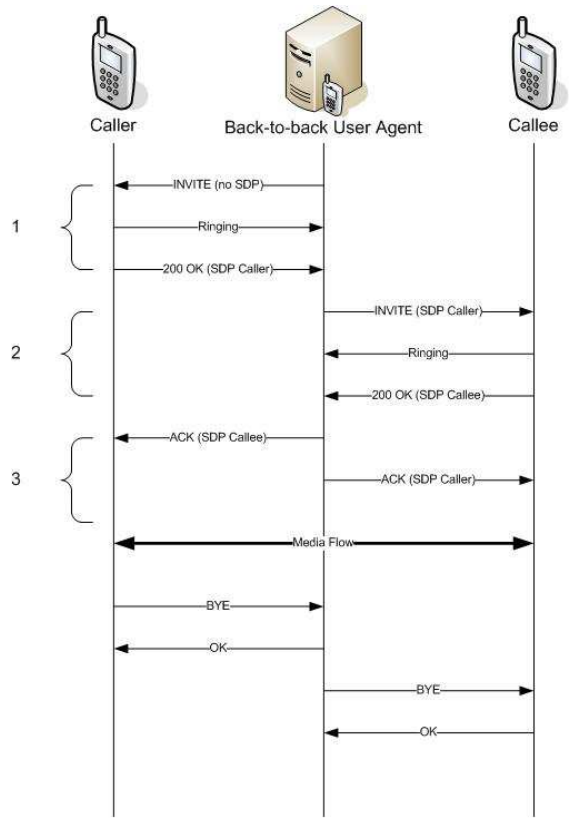

Figura 3-6[31]: Flujo Llamada UCT B2BUA

Figura 3-7: Interfaz Web UCT B2BUA

Gateway SMS

Para proveer el servicio de Gateway SMS se optó por el software Kannel [32], alternativa flexible y modificable a gusto del usuario. Nuevamente se optó por instalar el paquete Debian respectivo (`kannel`), modificando luego a discreción los archivos de configuración `kannel.conf` y `smskannel.conf` para incluir los datos de la red y el SMSC (Short Message Service Center, centro de mensajería corta) a utilizar. Los comandos para ejecutar el programa son los siguientes:

Tabla 15: Comandos Kannel

`sudo ./gw/bearerbox /home/user/smskannel.conf` `sudo ./gw/smsbox /home/user/smskannel.conf`
`sudo ./test/fakesmsc -i 1 -m 10 "100 200 text nop"`

En el Anexo 3 se presenta una configuración básica del archivo `smskannel.conf` con conexión a un SMSC mediante el protocolo SMPP v3.4, en la cual debe introducirse valores válidos paras las IPs y puertos respectivos (host y SMSC). Por otro lado, si no se dispone de una cuenta en un SMSC, se puede implementar un Fake SMSC mediante el último comando mostrado, el cual envía 10 mensajes de prueba con el comando `nop`.

Para probar el servicio se optó por implementar una página Web basada en PHP que extrajera la información de presencia de un usuario mediante MySQL y ofreciera la opción de enviar dichos datos vía SMS. El código se detalla en el Anexo 4. Cabe destacar que el archivo considera un `include` para obtener ciertos parámetros como el `$db_user` y `$db_passwd`. Más adelante se mostrará la apariencia de dicha Web.

Parlay X

Se optó por implementar la versión de ejemplo de JDeveloper de Oracle [33], proveyéndose un servicio de presencia basado en Parlay X Web Services. Para instalar y ejecutar la demo, se siguió paso a paso la guía de instalación que se detalla en el archivo http://www.oracle.com/technology/products/ocms/pdf/parlayxjdeveloper.pdf.

IPTV

El paquete Debian `uctiptv` [34] instala el programa y los archivos necesarios. Lo que sigue es aprovisionar el AS en el core IMS mediante la interfaz gráfica del HSS (http://localhost:8080, username hssAdmin, password hss), siguiendo los pasos que se muestran a continuación:

- Agregar un Application Server en el puerto 7070.

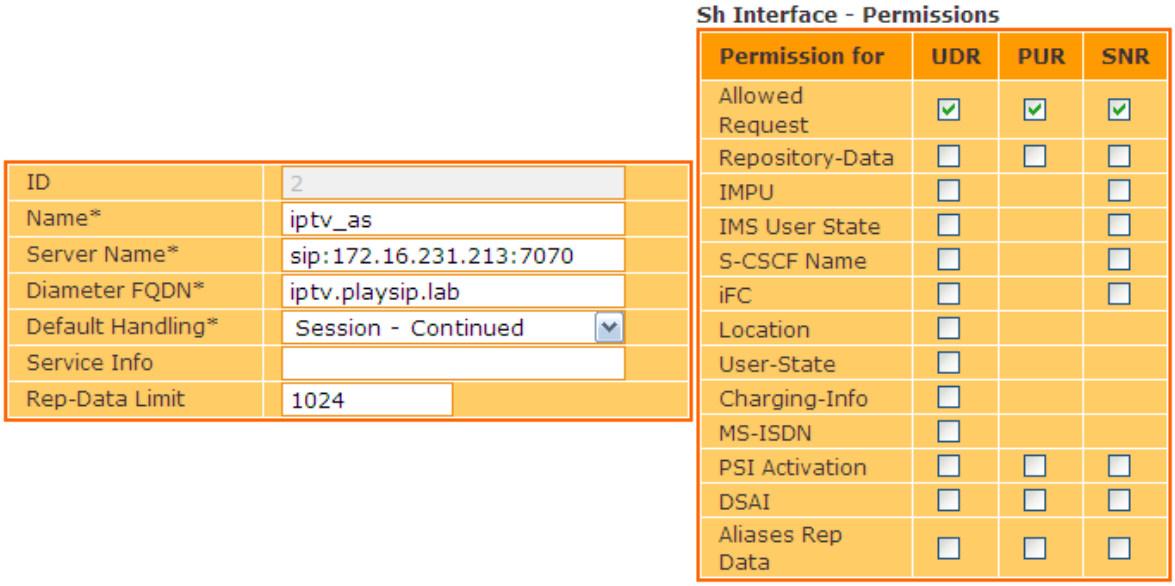

Figura 3-8: Instalación UCT IPTV - AS

- Agregar un Trigger Point y los SPT correspondientes; en este caso se contacta al servidor mediante un INVITE a channelX@iptv.playsip.lab, por lo

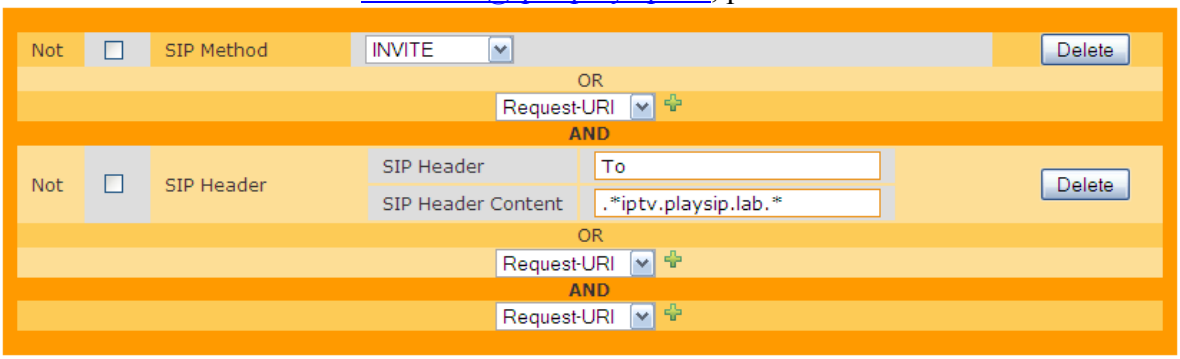

Figura 3-9: Instalación UCT IPTV - TP

- Asociar el Application Server y el respectivo Trigger Point con un iFC.

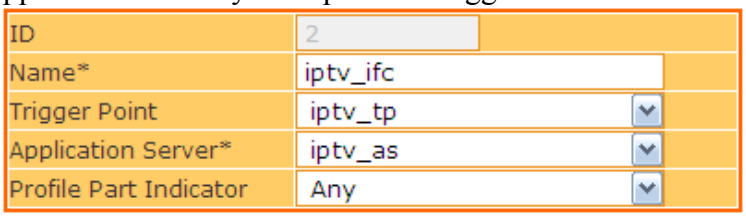

Figura 3-10: Instalación UCT IPTV - iFC

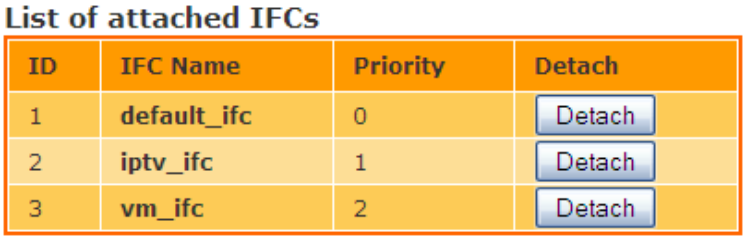

Figura 3-11: Instalación UCT IPTV – Service Profile

Para tener el un canal transmitiendo el archivo `channel1.mpg` a 100 kb/s en channel1@iptv.playsip.lab, el servidor se ejecuta mediante el siguiente comando:

Tabla 16: Comando UCT IPTV

```
uctiptv 100000 1 channel1.mpg
```

3.2.4 Interacción con Core IMS

Se considera la conexión de los servicios contra un OpenIMSCore desplegado en la máquina host. Para que los servicios desplegados interactúen con el core IMS, los AS respectivos deben ser aprovisionados en el HSS mediante la interfaz gráfica, de la manera explicada anteriormente para el caso de IPTV. Los pasos entonces son los siguientes:

- Agregar un Application Server.
- Agregar un Trigger Point y los SPT correspondientes.
- Asociar el Application Server y el respectivo Trigger Point con un iFC.
- Agregar el iFC al perfil de servicio deseado, con la respectiva prioridad.

3.3 Test Plan

En esta sección se detalla el plan de pruebas definido para probar la arquitectura desplegada. Posteriormente, se listan los resultados arrojados al aplicar dicho procedimiento a la plataforma, además de capturas interesantes que serán discutidas en el siguiente capítulo.

3.3.1 Definición

Los clientes a utilizar durante las pruebas corresponden al cliente SIP X-Lite 3.0, el cliente IMS UCT 1.0.11 (1.0.12 en algunas pruebas) y el cliente SIP incluido en un teléfono móvil Nokia E61i. Previamente a las pruebas de servicios, los clientes deben estar registrados contra un servidor SIP o contra un core IMS; es por esta razón que se incluyen pruebas básicas de registro.

Las pruebas podrán resultar exitosas (OK), no exitosas (NOK) o no serán aplicables (NA). Las pruebas que no se realicen entrarán en la categoría anterior. Como cada prueba puede presentar matices o situaciones particulares, se entregarán observaciones ad-hoc.

El plan de pruebas se define de acuerdo a la siguiente tabla:

Tabla 17: Plan de Pruebas

ID	Prueba
1	Registro SIP: X-Lite
2	Registro SIP: Nokia
3	Registro IMS: UCT
4	Registro IMS: Nokia
5	Sesión Voz SIP/SIP: X-Lite/X-Lite
6	Sesión Voz SIP/SIP: X-Lite/Nokia
7	Sesión Voz SIP/IMS: X-Lite/UCT
8	Sesión Voz SIP/IMS: Nokia/UCT
9	Sesión Voz IMS/IMS: UCT/UCT
10	Sesión Voz IMS/IMS: Nokia/UCT
11	Sesión Voicemail SIP: X-Lite
12	Sesión Voicemail SIP: Nokia
13	Sesión Voicemail IMS: UCT
14	Sesión Voicemail IMS: Nokia
15	Sesión Mensajería Instantánea SIP/SIP: X-Lite/X-Lite
16	Sesión Mensajería Instantánea SIP/IMS: X-Lite/UCT
17	Sesión Mensajería Instantánea IMS/IMS: UCT/UCT
18	Presencia OpenSER SIP: X-Lite
19	Presencia OpenSER IMS: UCT
20	Sesión Presencia OpenSER SIP/SIP: X-Lite/X-Lite
21	Sesión Presencia OpenSER SIP/IMS: X-Lite/UCT
22	Sesión Presencia OpenSER IMS/IMS: UCT/UCT
23	Sesión Presencia Parlay X: Oracle/Web
24	Sesión Click-to-Dial SIP/SIP: X-Lite/X-Lite
25	Sesión Click-to-Dial SIP/SIP: X-Lite/Nokia
26	Sesión Click-to-Dial SIP/IMS: X-Lite/UCT
27	Sesión Click-to-Dial SIP/IMS: X-Lite/Nokia
28	Sesión Click-to-Dial IMS/IMS: UCT/UCT
29	Sesión Click-to-Dial IMS/IMS: Nokia/UCT
30	Sesión IPTV SIP: X-Lite
31	Sesión IPTV SIP: Nokia
32	Sesión IPTV IMS: UCT
33	Sesión IPTV IMS: Nokia
34	Envío SMS: Web (PC)
35	Envío SMS: Web (Nokia)

3.3.2 Resultados

A continuación se detallarán los resultados obtenidos en la aplicación del Test Plan, a través de una tabla resumen para cada tipo de servicio provisto, además de incluir las observaciones surgidas en cada una de las pruebas. Estas últimas se discutirán en el siguiente capítulo.

3.3.2.1 Registro

Las pruebas de registro arrojaron los siguientes resultados:

Tabla 18: Resultados Test Plan, Registro

ID	Prueba	Resultado
1	Registro SIP: X-Lite	OK
2	Registro SIP: Nokia	OK
3	Registro IMS: UCT	OK
4	Registro IMS: Nokia	OK

A continuación se presentan los flujos obtenidos para el registro SIP e IMS en el caso del cliente Nokia. En el primer caso se trata de un registro sin autenticación, mientras que contra el core IMS se utiliza el algoritmo de autenticación AKAv1-MD5.

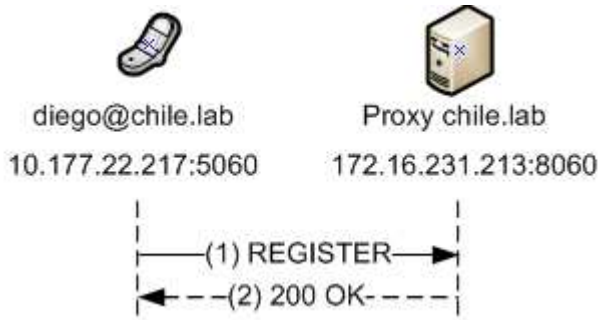

Figura 3-12: Registro SIP Nokia sin autenticación

Tabla 19: Mensaje (1) REGISTER - Registro SIP Nokia sin autenticación

```
REGISTER sip:172.16.231.213:8060 SIP/2.0
Route: <sip:172.16.231.213:8060;lr>
Via: SIP/2.0/UDP
10.177.22.217:5060;branch=z9hG4bKgsen0knbs1hc6cc70pe1elm;rport
From: <sip:diego@chile.lab>;tag=ec870kiviphc67on0pe1
To: <sip:diego@chile.lab>
Contact: <sip:diego@10.177.22.217>;expires=3600
CSeq: 1202 REGISTER
Call-ID: VEhMkIqNoIch9wps5VWOtDuYHuwUEC
Supported: sec-agree
User-Agent: Nokia RM-227 1.0633.22.06
Max-Forwards: 70
Content-Length: 0
```

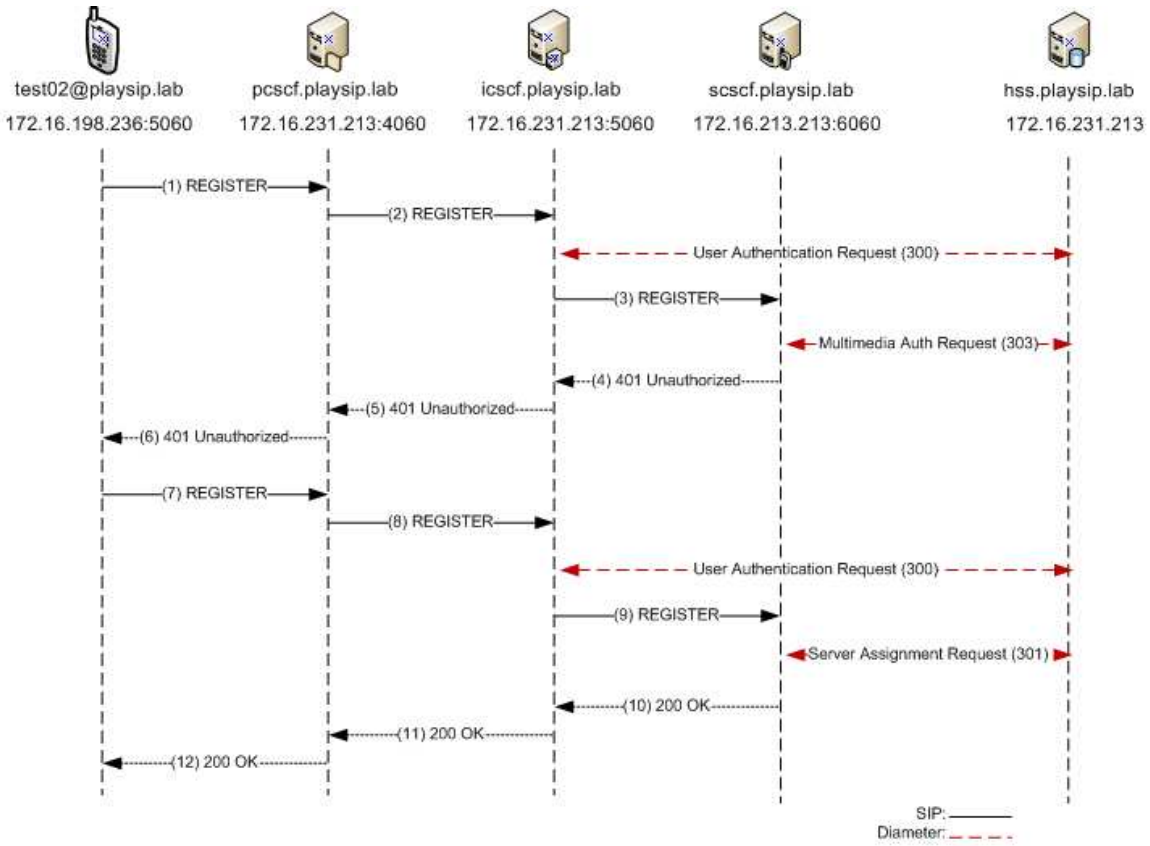

Figura 3-13: Registro IMS Nokia con autenticación Digest-AKAv1

Tabla 20: Mensaje (7) REGISTER - Registro IMS Nokia con autenticación Digest-AKAv1

```
REGISTER sip:icscf.playsip.lab SIP/2.0
Route: <sip:172.16.231.213:4060;lr>
Via: SIP/2.0/UDP 172.16.231.213:4060;branch=z9hG4bK45f4.fe12b963.0
Via: SIP/2.0/UDP
172.16.198.236:5060;branch=z9hG4bKquq18h8k8k6l4vlv4riunhr;rport=5060
From: <sip:test02@playsip.lab>;tag=4mu2dpavm5hc6c9n06hk
To: <sip:test02@playsip.lab>
Contact: <sip:test02@172.16.198.236>;expires=3600
CSeq: 1311 REGISTER
Call-ID: KTwCVjtsoIc62M6zjUShUQ-9LGsTLv
Supported: sec-agree
User-Agent: Nokia RM-227 1.0633.22.06
Max-Forwards: 16
Content-Length: 0
Authorization: Digest
realm="playsip.lab",nonce="6404672e1861d0af9eff7209ad926bfe",algorithm=MD5,use
rname="test02@playsip.lab",uri="sip:icscf.playsip.lab",response="b044980422ed3
fdfe37464946f798596", integrity-protected="no"
Path: <sip:term@pcscf.playsip.lab:4060;lr>
Require: path
P-Charging-Vector: icid-value="P-CSCFabcd484d4ee4000003d3";icid-generated-
at=172.16.231.213;orig-ioi="playsip.lab"
P-Visited-Network-ID: playsip.lab
```

3.3.2.2 Servicios de Voz

El resumen de resultados obtenidos para los servicios de voz se presenta en la siguiente tabla:

Tabla 21: Resultados Test Plan, Servicios de Voz

ID	Prueba	Resultado
5	Sesión Voz SIP/SIP: X-Lite/X-Lite	OK
6	Sesión Voz SIP/SIP: X-Lite/Nokia	OK
7	Sesión Voz SIP/IMS: X-Lite/UCT	OK
8	Sesión Voz SIP/IMS: Nokia/UCT	NA
9	Sesión Voz IMS/IMS: UCT/UCT	OK
10	Sesión Voz IMS/IMS: Nokia/UCT	NA
11	Sesión Voicemail SIP: X-Lite	OK
12	Sesión Voicemail SIP: Nokia	NOK
13	Sesión Voicemail IMS: UCT	NOK
14	Sesión Voicemail IMS: Nokia	NA

En el caso de los servicios de voz, las llamadas entre clientes del mismo tipo (SIP/SIP o IMS/IMS) resultaron satisfactorias, así como la comunicación entre un cliente SIP y un usuario registrado contra el core IMS. En cuanto al servicio de Voicemail, se presentaron ciertos problemas al acceder a éste desde un cliente IMS, los cuales serán discutidos en el siguiente capítulo. Adicionalmente, las pruebas de Voicemail para el cliente Nokia registrado contra OpenSER tampoco resultaron satisfactorias.

A continuación se presenta el flujo obtenido para una llamada de voz entre dos usuarios IMS pertenecientes al mismo dominio, además del flujo hacia y desde el servidor de Voicemail al rescatar un mensaje, utilizando un cliente SIP X-Lite. Se presentan también los mensajes INVITE iniciales para cada caso.

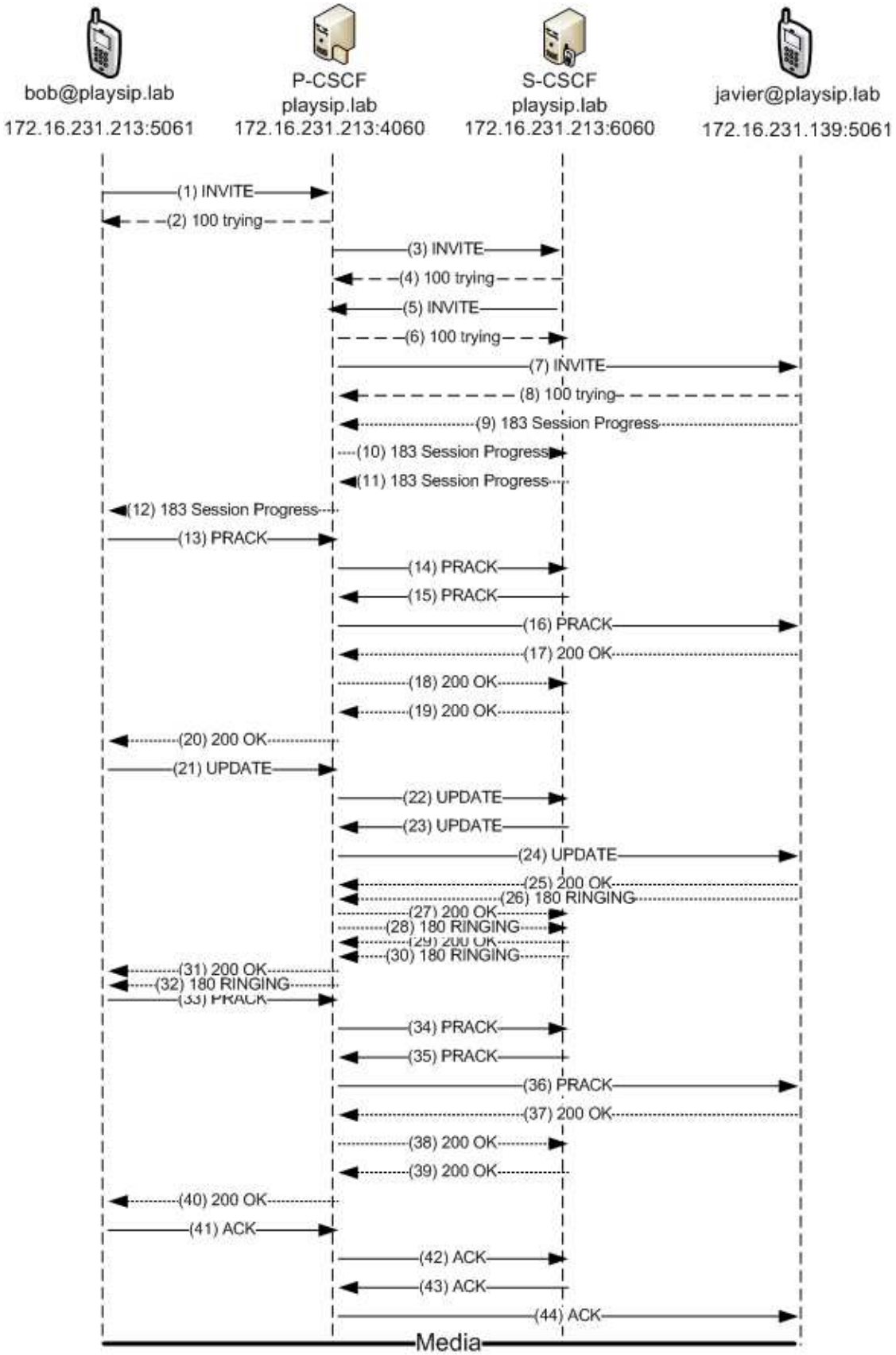

Figura 3-14: Llamada IMS en el Mismo Dominio

Tabla 22: Mensaje (1) INVITE - Llamada IMS en el Mismo Dominio

```
INVITE sip:javier@playsip.lab SIP/2.0
Via: SIP/2.0/UDP 172.16.231.213:5061;rport;branch=z9hG4bK1087001947
Route: <sip:orig@scscf.playsip.lab:6060;lr>
From: "Bob" <sip:bob@playsip.lab>;tag=865285700
To: <sip:javier@playsip.lab>
Call-ID: 269933587
CSeq: 20 INVITE
Contact: <sip:bob@172.16.231.213:5061>
Content-Type: application/sdp
Allow: INVITE, ACK, CANCEL, BYE, PRACK, UPDATE, REFER, MESSAGE
Max-Forwards: 70
User-Agent: UCT IMS Client
Subject: IMS Call
Expires: 120
P-Preferred-Identity: "Bob" <sip:bob@playsip.lab>
P-Preferred-Service: urn:xxx:3gpp-service.ims.icsi.mmtel
Privacy: none
P-Access-Network-Info: IEEE-802.11a
Require: precondition
Require: sec-agree
Proxy-Require: sec-agree
Supported: 100rel
Content-Length:   441

v=0
o=- 0 0 IN IP4 172.16.231.213
s=IMS Call
c=IN IP4 172.16.231.213
t=0 0
m=audio 26213 RTP/AVP 3 0 101
b=AS:64
a=rtpmap:3 GSM/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-11
a=curr:qos local none
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos mandatory remote sendrecv
m=message 21039 TCP/MSRP
a=accept-types:text/plain
a=path:msrp://172.16.231.213:21039/3a72007d16b9;tcp
```

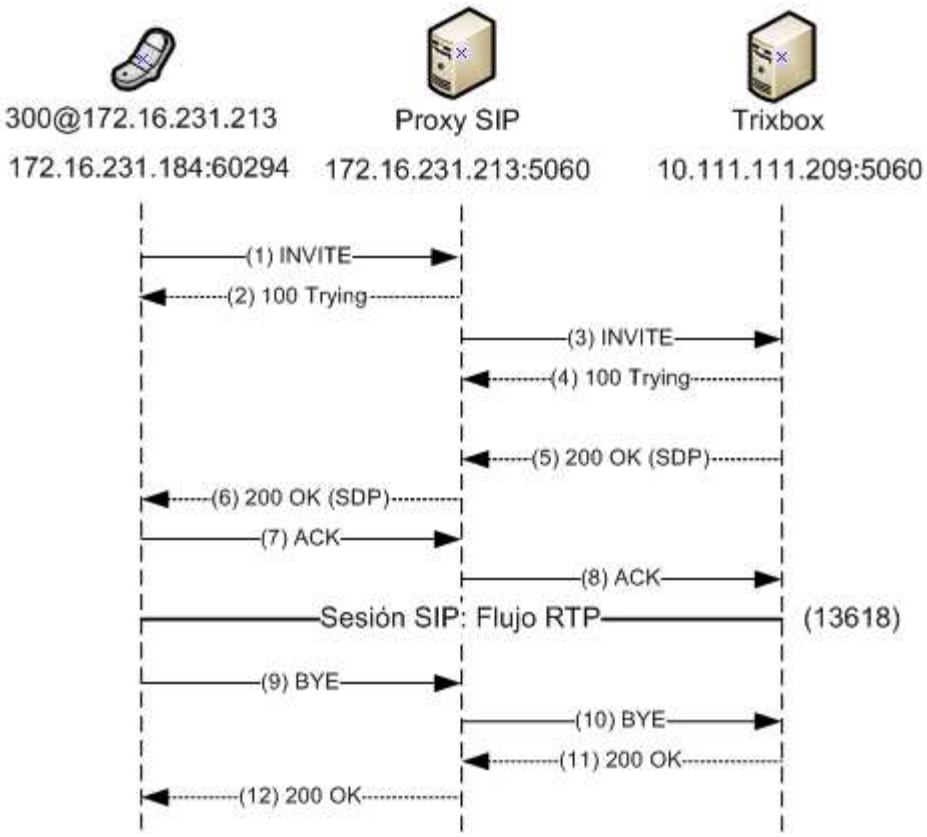

Figura 3-15: Llamada SIP a Voicemail

Tabla 23: Mensaje (1) INVITE - Llamada SIP a Voicemail

```
INVITE sip:*98@10.111.111.209:5060 SIP/2.0
Record-Route: <sip:172.16.231.213;lr=on;ftag=7008163a>
Via: SIP/2.0/UDP 172.16.231.213;branch=z9hG4bKf126.c08f42f.0
Via: SIP/2.0/UDP 172.16.231.184:60294;branch=z9hG4bK-d87543-f20c42211816f868-
1--d87543-
Max-Forwards: 69
Contact: <sip:300@172.16.231.184:60294>
To: "*98"<sip:*98@172.16.231.213>
From: "Juanito Mena"<sip:300@172.16.231.213>;tag=7008163a
Call-ID: YzNkMzgyMjM4ZDU5ZjNmOThkN2YyYzdmYTVhYTU1YmM.
CSeq: 2 INVITE
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY, MESSAGE, SUBSCRIBE,
INFO
Content-Type: application/sdp
Proxy-Authorization: Digest
username="300",realm="oser.test",nonce="47b44974eecf0a9569af9853050b0249be8b72
5e",uri="sip:*98@172.16.231.213",response="b39a747ea977d772ee059d8540b21801",a
lgorithm=MD5
User-Agent: X-Lite release 1011s stamp 41150
Content-Length: 524

v=0
o=- 6 2 IN IP4 172.16.231.184
s=CounterPath X-Lite 3.0
c=IN IP4 172.16.231.184
t=0 0
m=audio 9092 RTP/AVP 107 119 100 106 0 105 98 8 101
a=alt:1 3 : ECO4BFoN gNa0cLEd 172.16.231.184 9092
```

```
a=alt:2 2 : 5CsWRXnN Un/99Quw 192.168.102.1 9092
a=alt:3 1 : Zsfuf0/j PyTY0Zio 192.168.154.1 9092
a=fmtp:101 0-15
a=rtpmap:107 BV32/16000
a=rtpmap:119 BV32-FEC/16000
a=rtpmap:100 SPEEX/16000
a=rtpmap:106 SPEEX-FEC/16000
a=rtpmap:105 SPEEX-FEC/8000
a=rtpmap:98 iLBC/8000
a=rtpmap:101 telephone-event/8000
a=sendrecv
```

3.3.2.3 Presencia y Mensajería Instantánea

La que sigue corresponde a la tabla resumen de resultados para los servicios de presencia y mensajería instantánea.

Tabla 24: Resultados Test Plan, Servicios de Presencia y Mensajería Instantánea

ID	Prueba	Resultado
15	Sesión Mensajería Instantánea SIP/SIP: X-Lite/X-Lite	OK
16	Sesión Mensajería Instantánea SIP/IMS: X-Lite/UCT	NA
17	Sesión Mensajería Instantánea IMS/IMS: UCT/UCT	OK
18	Presencia OpenSER SIP: X-Lite	OK
19	Presencia OpenSER IMS: UCT	OK
20	Sesión Presencia OpenSER SIP/SIP: X-Lite/X-Lite	OK
21	Sesión Presencia OpenSER SIP/IMS: X-Lite/UCT	NA
22	Sesión Presencia OpenSER IMS/IMS: UCT/UCT	OK
23	Sesión Presencia Parlay X: Oracle/Web	OK

Los flujos obtenidos y mensajes de ejemplo para la mensajería instantánea se muestran a continuación:

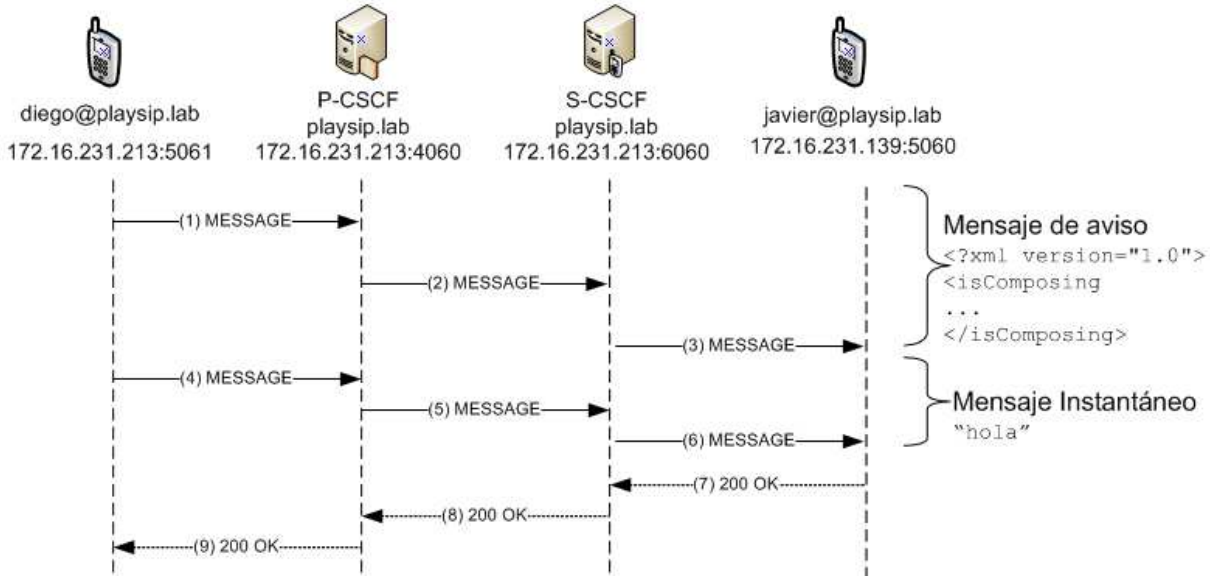

Figura 3-16: Mensajería Instantánea

Tabla 25: Mensaje (1) MESSAGE - Mensajería Instantánea

```
MESSAGE sip:javier@playsip.lab SIP/2.0
Via: SIP/2.0/UDP 172.16.231.213:6060;branch=z9hG4bK530e.7e9b6bf.0
Via: SIP/2.0/UDP 172.16.231.213:4060;branch=z9hG4bK530e.c5561b41.0
Via: SIP/2.0/UDP 172.16.231.213:5061;rport=5061;branch=z9hG4bK10683234
From: "Diego" <sip:diego@playsip.lab>;tag=2053192809
To: <sip:javier@playsip.lab>
Call-ID: 1757419275
CSeq: 20 MESSAGE
Content-Type: application/im-iscomposing+xml
Max-Forwards: 15
User-Agent: UCT IMS Client
P-Access-Network-Info: IEEE-802.11b
Content-Length:   300
P-Asserted-Identity: "Diego" <sip:diego@playsip.lab>
P-Charging-Vector: icid-value="P-CSCFabcd48b16e44000000f1";icid-
generated-at=172.16.231.213;orig-ioi="playsip.lab"

<?xml version="1.0" encoding="UTF-8"?>
<isComposing xmlns="urn:ietf:params:xml:ns:im-iscomposing"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:ietf:params:xml:ns:im-composing iscomposing.xsd">
<state>active</state>
<contenttype>text/plain</contenttype>
</isComposing>
```

Tabla 26: Mensaje (4) MESSAGE - Mensajería Instantánea

```
MESSAGE sip:javier@playsip.lab SIP/2.0
Via: SIP/2.0/UDP 172.16.231.213:5061;rport;branch=z9hG4bK1375888091
Route: <sip:orig@scscf.playsip.lab:6060;lr>
From: "Diego" <sip:diego@playsip.lab>;tag=1178541260
To: <sip:javier@playsip.lab>
Call-ID: 1005268703
CSeq: 20 MESSAGE
Content-Type: text/plain
Max-Forwards: 70
User-Agent: UCT IMS Client
P-Preferred-Identity: "Diego" <sip:diego@playsip.lab>
P-Access-Network-Info: IEEE-802.11b
Content-Length:     4

hola
```

El primer mensaje corresponde a una alerta hacia el usuario destino, del tipo “el usuario origen está escribiendo un mensaje”, para que el primero sepa que luego obtendrá texto del segundo. Como se puede apreciar, dicha alerta se encuentra codificado en lenguaje XML.

Luego, el mensaje en sí es enviado en formato de texto plano.

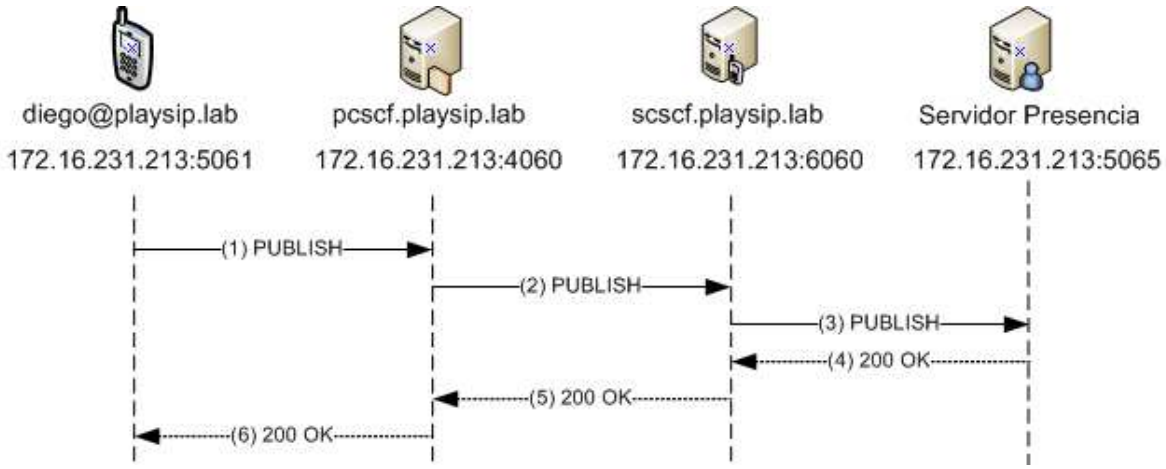

Figura 3-17: Publicación de Presencia

Tabla 27: Mensaje (1) PUBLISH - Publicación de Presencia

```
PUBLISH sip:diego@playsip.lab SIP/2.0
Via: SIP/2.0/UDP 172.16.231.213:5061;rport;branch=z9hG4bK1509944107
Route: <sip:orig@scscf.playsip.lab:6060;lr>
From: <sip:diego@playsip.lab>;tag=1024062830
To: <sip:diego@playsip.lab>
Call-ID: 411691920
CSeq: 21 PUBLISH
Content-Type: application/pidf+xml
Max-Forwards: 70
User-Agent: UCT IMS Client
Content-Disposition: render;handling=required
Expires: 3600
Event: presence
SIP-If-Match: a.1219466694.14640.9
Content-Length:   285

<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
xmlns:im="urn:ietf:params:xml:ns:pidf:im"
entity="sip:diego@playsip.lab">
[09]<tuple id="UCTIMSClient">
[09][09]<status>
[09][09][09]<basic>open</basic>
[09][09]</status>
[09][09]<note>Away</note>
[09]</tuple>
</presence>
```

La información de presencia de un usuario corresponde también a datos codificados en lenguaje XML. El mensaje PUBLISH es recibido por el servidor de presencia, el cual almacena la información del usuario en una base de datos. En el caso de OpenSER y el módulo de presencia implementado, la base de datos corresponde a MySQL, a la cual se accede mediante consultas SQL estándar.

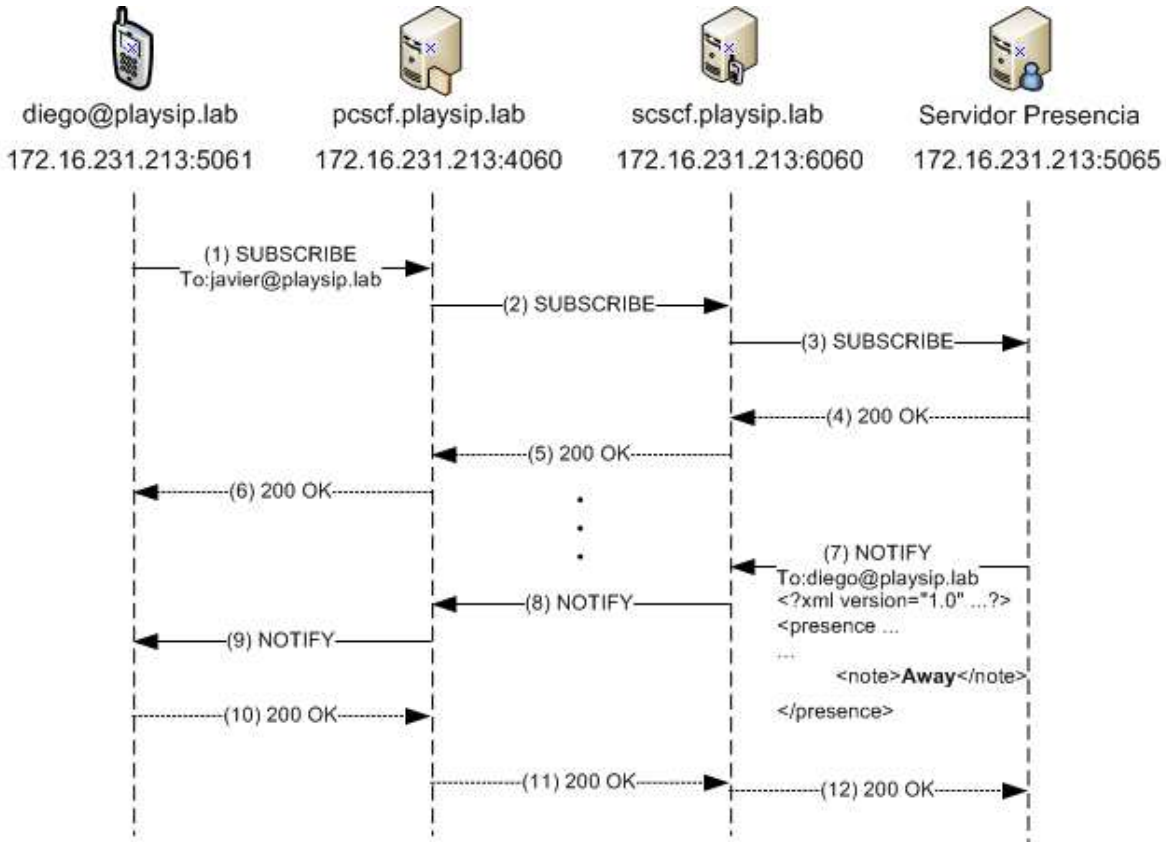

Figura 3-18: Suscripción a Presencia

Tabla 28: Mensaje (1) SUBSCRIBE - Suscripción a Presencia

```
SUBSCRIBE sip:javier@playsip.lab SIP/2.0
Via: SIP/2.0/UDP 172.16.231.213:5061;rport;branch=z9hG4bK1949360157
Route: <sip:orig@scscf.playsip.lab:6060;lr>
From: <sip:diego@playsip.lab>;tag=1646018427
To: <sip:javier@playsip.lab>
Call-ID: 1576209949
CSeq: 20 SUBSCRIBE
Contact: <sip:diego@172.16.231.213:5061>
Max-Forwards: 70
User-Agent: UCT IMS Client
Expires: 600
Event: presence
Content-Length: 0
```

El usuario puede suscribirse contra el servidor de presencia mediante un mensaje SUBSCRIBE (pasando a ser un "watcher") para recibir información actualizada sobre la presencia de otro usuario. Así, cada vez que este último renueve su información de presencia, el servidor informará al watcher del cambio y actualizará el estado de acuerdo a ello. Esto se realiza a través de un mensaje NOTIFY, como se muestra en el siguiente flujo.

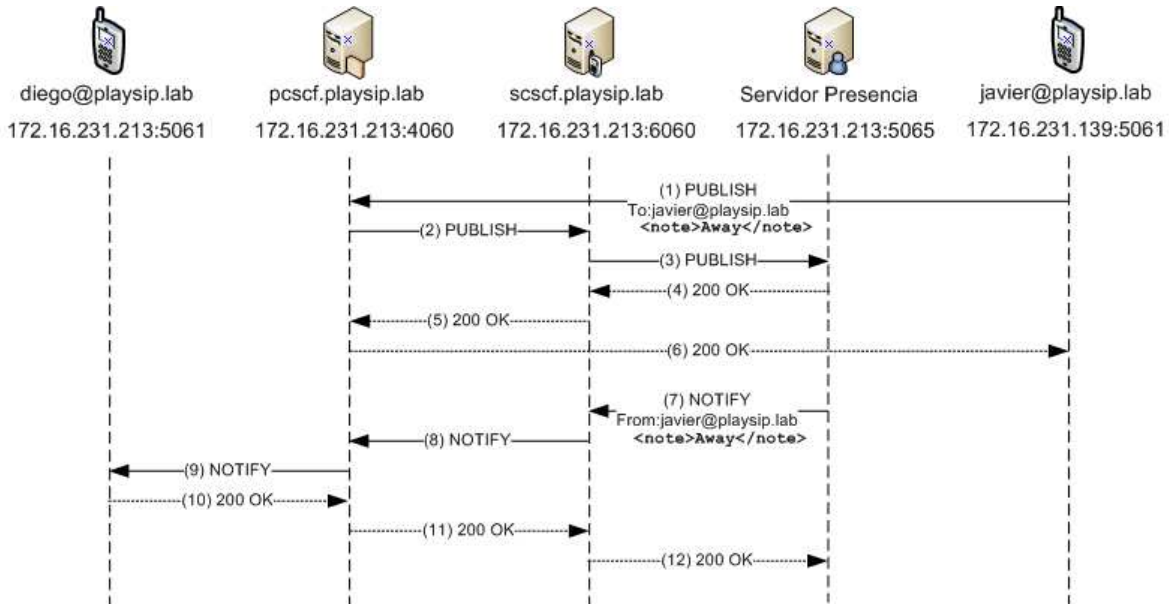

Figura 3-19: Sesión con Presencia

Tabla 29: Mensaje (7) NOTIFY - Sesión con Presencia

```
NOTIFY sip:diego@172.16.231.213:5061 SIP/2.0
Via: SIP/2.0/UDP 172.16.231.213:5065;branch=z9hG4bK841d.4b352347.0
To: sip:diego@playsip.lab;tag=1646018427
From: sip:javier@playsip.lab;tag=10.14638.1219587034.9
CSeq: 3 NOTIFY
Call-ID: 1576209949
Route: <sip:mo@scscf.playsip.lab:6060;lr>,
       <sip:mo@pcscf.playsip.lab:4060;lr>
Content-Length: 275
User-Agent: OpenSER (1.2.2-notls (i386/linux))
Max-Forwards: 70
Event: presence
Contact: <sip:172.16.231.213:5065>
Subscription-State: active;expires=383
Content-Type: application/pidf+xml

<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
xmlns:im="urn:ietf:params:xml:ns:pidf:im"
entity="sip:javier@playsip.lab">
[09]<tuple id="UCTIMSClient">
[09][09]<status>
[09][09][09]<basic>open</basic>
[09][09]</status>
[09][09]<note>Away</note>
[09]</tuple>
</presence>
```

El servicio de presencia mediante Web Service Parlay X funciona de la misma manera que el caso anterior, con la diferencia que en este caso se utiliza un cliente propietario de Oracle junto con un cliente basado en Web. Los flujos son los mismos; cambia un poco el formato XML de la información de presencia, lo que será discutido en el próximo capítulo.

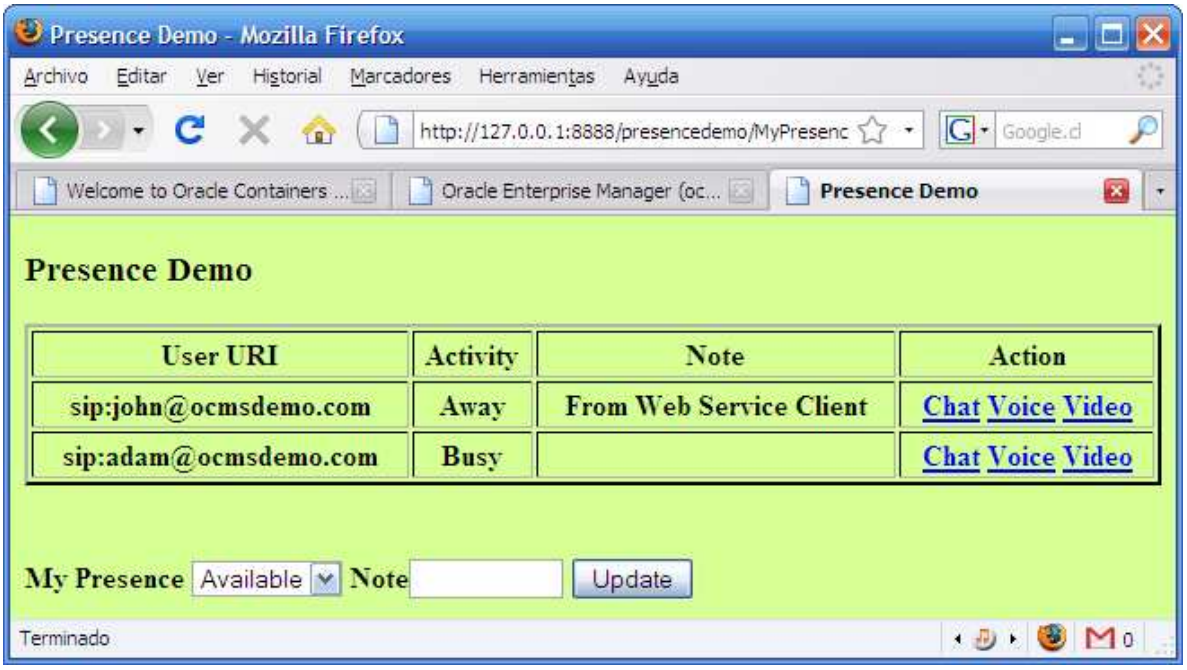

Figura 3-20: Presencia Parlay X Web

Figura 3-21: Presencia Parlay X Oracle Communicator

3.3.2.4 Otros Servicios

A continuación se muestra la tabla resumen de resultados para los servicios adicionales desplegados.

Tabla 30: Resultados Test Plan, Otros Servicios

ID	Prueba	Resultado
24	Sesión Click-to-Dial SIP/SIP: X-Lite/X-Lite	NOK
25	Sesión Click-to-Dial SIP/SIP: X-Lite/Nokia	NA
26	Sesión Click-to-Dial SIP/IMS: X-Lite/UCT	NA
27	Sesión Click-to-Dial SIP/IMS: X-Lite/Nokia	NA
28	Sesión Click-to-Dial IMS/IMS: UCT/UCT	NOK
29	Sesión Click-to-Dial IMS/IMS: Nokia/UCT	NA
30	Sesión IPTV SIP: X-Lite	NOK
31	Sesión IPTV SIP: Nokia	NOK
32	Sesión IPTV IMS: UCT	OK
33	Sesión IPTV IMS: Nokia	NA
34	Envío SMS: Web (PC)	OK
35	Envío SMS: Web (Nokia)	OK

A continuación se presenta el flujo obtenido para una sesión C2D entre dos usuarios SIP, incluyendo el flujo de medios teórico que no pudo ser establecido, lo cual se discutirá en el siguiente capítulo.

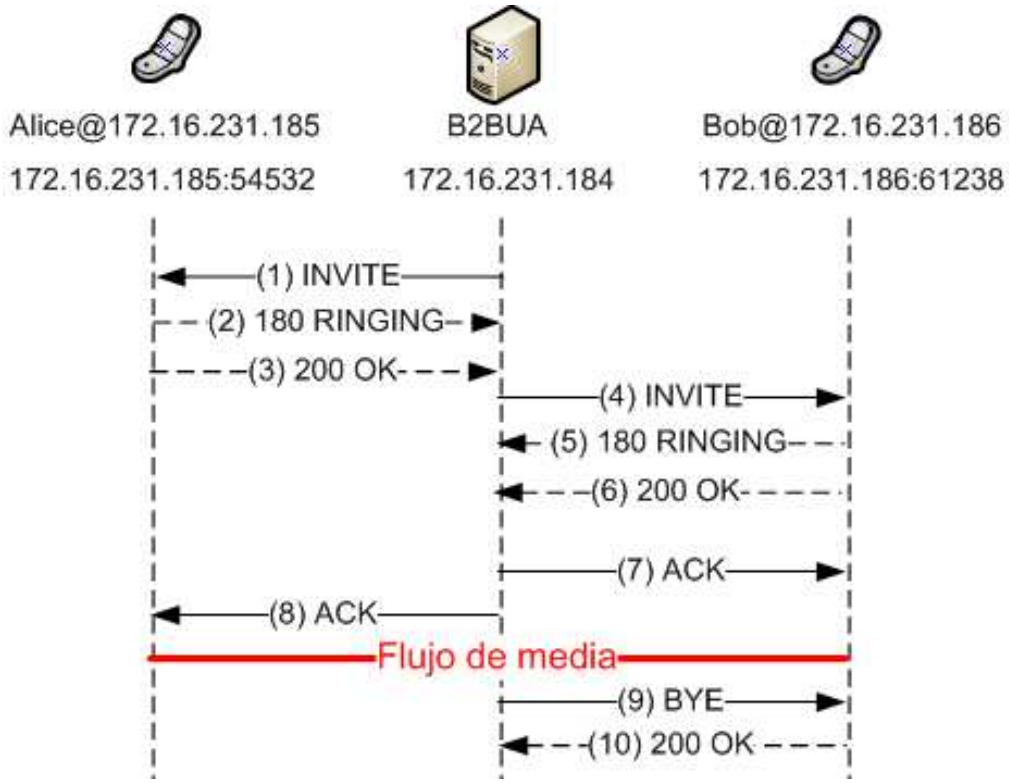

Figura 3-22: Click-to-Dial SIP

Tabla 31: Mensaje (7) ACK – Click-to-Dial SIP

```
ACK sip:Bob@172.16.231.186:61238 SIP/2.0
Max-Forwards: 69
Content-Length: 546
To:
<sip:Bob@172.16.231.186:61238>;rinstance=de475c3d49dcc195;tag=a759236d
Cseq: 1 ACK
Via: SIP/2.0/UDP 172.16.231.184:5060;branch=z9hG4bKffefca38-faee-4746-
9278-7d42e8c5b11c
Content-Type: application/sdp
From:
<sip:Alice@172.16.231.185:54532>;rinstance=d8aeff9fa6125dc6;tag=ffcq248r-
8
Call-Id: 172.16.231.184_2_6860402202160814491

v=0
o=- 2 2 IN IP4 172.16.231.186
s=CounterPath X-Lite 3.0
c=IN IP4 172.16.231.186
t=0 0
m=audio 5734 RTP/AVP 107 119 100 106 0 105 98 8 101
a=fmtp:101 0-15
a=rtpmap:107 BV32/16000
a=rtpmap:119 BV32-FEC/16000
a=rtpmap:100 SPEEX/16000
a=rtpmap:106 SPEEX-FEC/16000
a=rtpmap:105 SPEEX-FEC/8000
a=rtpmap:98 iLBC/8000
a=rtpmap:101 telephone-event/8000
a=sendrecv
m=video 41818 RTP/AVP 115 34
a=fmtp:115 QCIF=1 I=1 J=1 T=1 MaxBR=1960
a=fmtp:34 QCIF=1 CIF=1 MaxBR=1960
a=rtpmap:115 H263-1998/90000
a=rtpmap:34 H263/90000
a=sendrecv
```

En el caso de IPTV, se muestra el flujo de datos para un cliente registrado contra el core IMS y el mensaje INVITE respectivo.

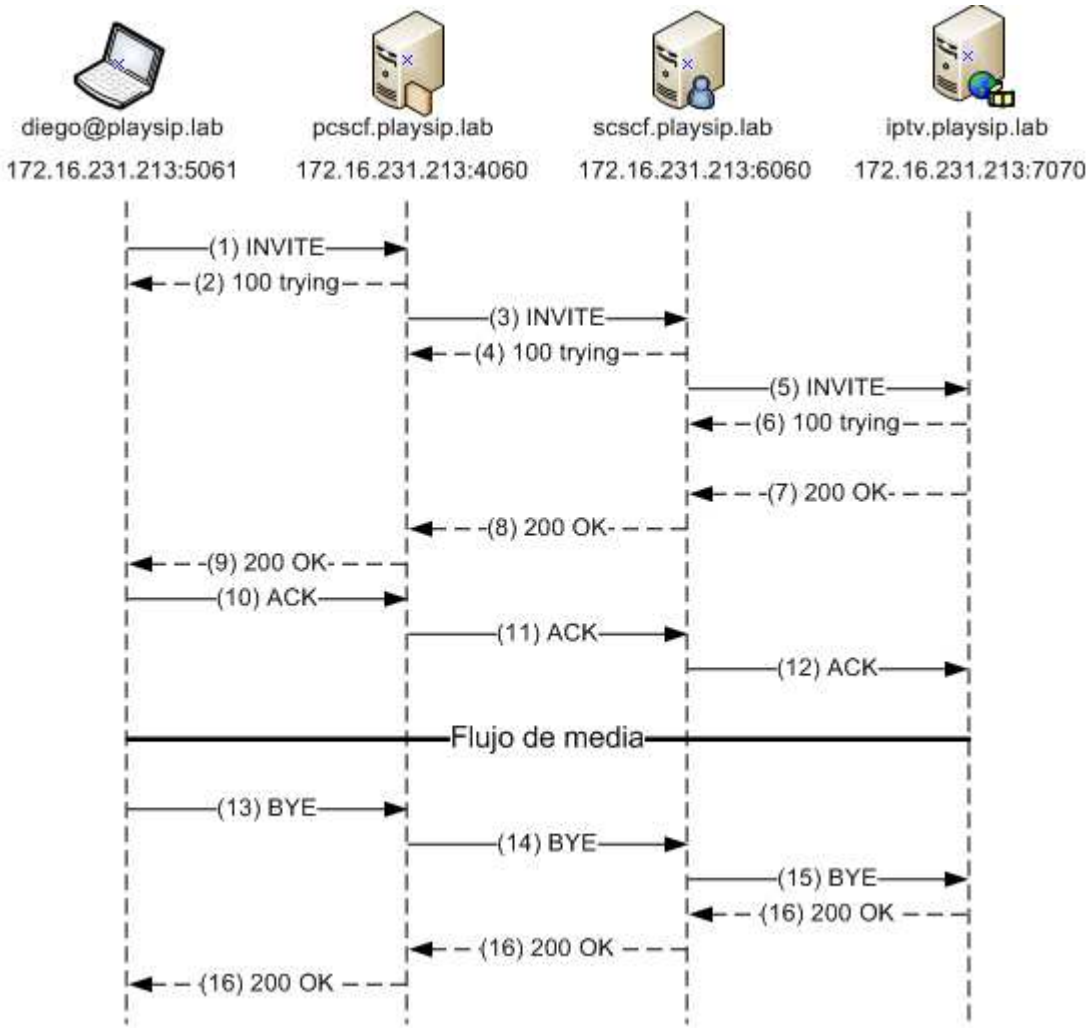

Figura 3-23: Sesión IPTV IMS

Tabla 32: Mensaje (1) INVITE - Sesión IPTV IMS

```
INVITE sip:channel1@iptv.playsip.lab SIP/2.0
Via: SIP/2.0/UDP 172.16.231.213:5061;rport;branch=z9hG4bK893075488
Route: <sip:orig@scscf.playsip.lab:6060;lr>
From: "Diego" <sip:diego@playsip.lab>;tag=866150752
To: <sip:channel1@iptv.playsip.lab>
Call-ID: 1377316301
CSeq: 20 INVITE
Contact: <sip:diego@172.16.231.213:5061>
Content-Type: application/sdp
Allow: INVITE, ACK, CANCEL, BYE, PRACK, UPDATE, REFER, MESSAGE
Max-Forwards: 70
User-Agent: UCT IMS Client
Subject: IMS Call
Expires: 120
P-Preferred-Identity: "Diego" <sip:diego@playsip.lab>
P-Preferred-Service: urn:xxx:3gpp-service.ims.icsi.mmtel
Privacy: none
```

```
P-Access-Network-Info: IEEE-802.11b
Require: sec-agree
Proxy-Require: sec-agree
Supported: 100rel
Content-Length:   524

v=0
o=- 0 0 IN IP4 172.16.231.213
s=IMS Call
c=IN IP4 172.16.231.213
t=0 0
m=audio 17537 RTP/AVP 3 0 101
b=AS:64
a=rtpmap:3 GSM/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-11
a=curr:qos local none
a=curr:qos remote none
a=des:qos none local sendrecv
a=des:qos none remote sendrecv
m=video 17145 RTP/AVP 96
b=AS:128
a=curr:qos local none
a=curr:qos remote none
a=des:qos none local sendrecv
a=des:qos none remote sendrecv
a=rtpmap:96 H263-1998
a=fmtp:96 profile-level-id=0
```

Por último, con respecto al servicio Web de información de presencia y envío vía SMS presenta la siguiente interfaz en un PC:

Figura 3-24: Envío SMS Web

Al encaminar exitosamente el SMS, el siguiente mensaje es mostrado:

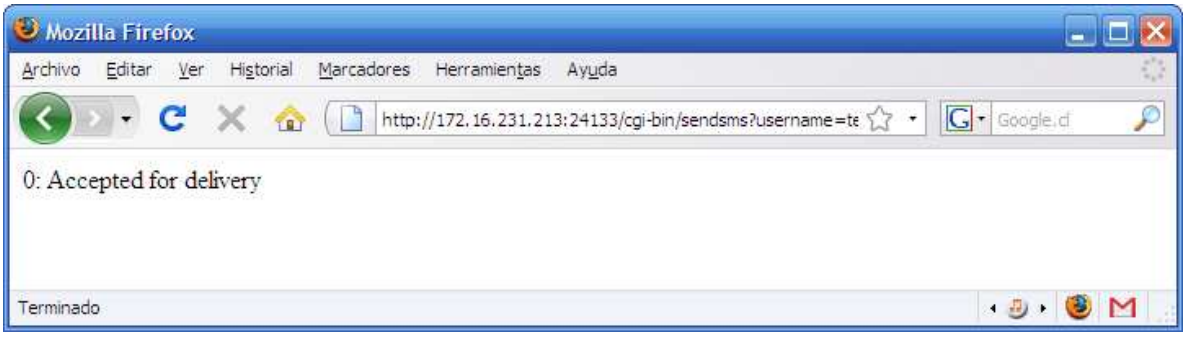

Figura 3-25: Confirmación envío SMS Web

En el caso del teléfono Nokia, la página Web también puede ser desplegada, a lo cual el envío de SMS se hace transparente para el usuario.

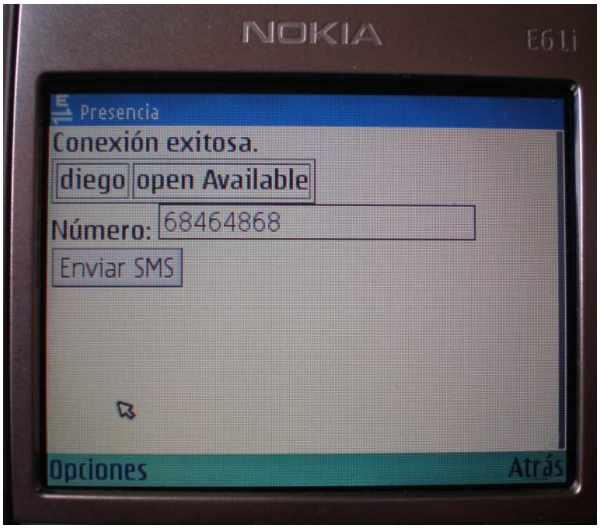

Figura 3-26: Envío SMS Nokia

El mensaje con la información de presencia recibido tiene el siguiente aspecto:

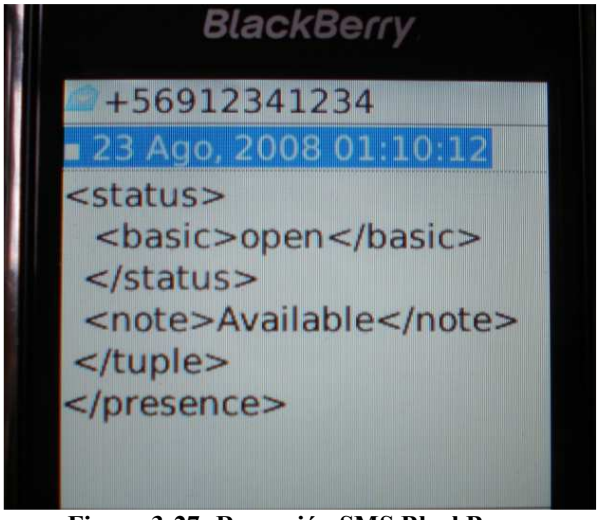

Figura 3-27: Recepción SMS BlackBerry

Capítulo 4

Discusión de Resultados

En este capítulo se examinan los resultados del capítulo anterior, mencionando y haciendo críticas tanto de lo que resultó exitoso como de lo que no, planteando así el desarrollo del tema a futuro.

4.1 Resultados Test Plan

La siguiente tabla resume la compatibilidad obtenida para cada servicio en los distintos tipos de cliente probados, diferenciando entre servicios SIP puros y SIP IMS:

Tabla 33: Resumen compatibilidad de servicios en clientes

SIP	X-Lite	Nokia
Registro	OK	OK
Llamada Voz (SIP/SIP)	OK	OK
VoiceMail	OK	NA
Mensajería Instantánea	OK	NA
Presencia	OK	NA
Click-to-Dial	NOK	NA
IPTV	NOK	NA
SMS Web	NA	OK

IMS	UCT	Nokia
Registro	OK	OK
Llamada Voz (IMS/IMS)	OK	NA
VoiceMail	NOK	NA
Mensajería Instantánea	OK	NA
Presencia	OK	NA
Click-to-Dial	NOK	NA
IPTV	OK	NA
SMS Web	NA	NA

A continuación se comentarán y discutirán los resultados específicos del plan de pruebas aplicado, descrito en el capítulo anterior.

4.1.1 Registro

Se logró registrar los diferentes clientes tanto contra el servidor SIP OpenSER como contra el core IMS. Como se puede ver en los flujos y capturas de la sección 3.3.2, se logró registrar un usuario utilizando el cliente SIP nativo del teléfono Nokia, primero sin autenticación contra el Proxy/Registrar SIP y luego utilizando autenticación AKAv1 contra el core IMS. Esto

representa una clara oportunidad para desarrollar servicios compatibles con clientes ya desplegados en el mercado, como es el caso del cliente SIP nativo d dicho teléfono móvil.

Cabe destacar que al intentar registrarse contra el core en otra oportunidad, el registro contra el core IMS no llegó a realizarse. Esto ocurrió dado que al momento de autenticar el nombre de usuario era cambiado durante el flujo del mensaje. A continuación se detalla el REGISTER que incluye las credenciales:

Tabla 34: Nombre de usuario cambia al autenticar, Nokia IMS

```
REGISTER sip:icscf.playsip.lab SIP/2.0
Route: <sip:172.16.231.213:4060;lr>
Via: SIP/2.0/UDP
10.177.24.115:5060;branch=z9hG4bKk9i12goh8da92vlve199o4r;rport
From: <sip:md5@playsip.lab>;tag=sh570kh8clhc6g4o0ljd
To: <sip:md5@playsip.lab>
Contact: <sip:md5@10.177.24.115>;expires=3600
CSeq: 1206 REGISTER
Call-ID: X8ZMkH52oIc3dwlCS-5I4Bir1BALuS
Supported: sec-agree
User-Agent: Nokia RM-227 1.0633.22.06
Max-Forwards: 70
Authorization: Digest
realm="playsip.lab",nonce="lYO1Y3QRDuDLK6CyFexMXj89Z4lLQgAARtsoHO7vqqQ=",algor
ithm=AKAv1-
MD5,username="730020900067705@ims.mnc002.mcc730.3gppnetwork.org",uri="sip:icsc
f.playsip.lab",response="5738bc9b461b9a8131451eb61d12f611"
Content-Length: 0
```

Así, el nombre de usuario es totalmente distinto al original `md5`, por lo que el HSS no lo reconoce como válido y le prohibe el registro mediante un mensaje `403 Forbidden - HSS User Unknown`. Se sospechó en un principio que el problema se debía a un cambio en los equipos del core de paquetes de la operadora, específicamente el GGSN, cuyo proveedor difiere del original, pero notando el rango de la IP asignada al teléfono móvil se puede notar que el registro detallado en la sección 3.3.2.1 se realizó a través de la red WiFi y no vía la red GSM. Por otro lado, el cambio de nombre de usuario se debe a lo propuesto en la especificación 3GPP TR 33.978, que establece que la identidad pública del usuario IMS se construye a partir de los datos almacenados en la tarjeta SIM del teléfono móvil, de la siguiente manera:

[IMSI]@ims.mnc[MNC].mcc[MCC].3gppnetwork.org

Así, el IMSI del usuario acompaña al MNC (Mobile Network Code), que en el caso de la red de Movistar Chile es el 02 (se agregan ceros a la izquierda hasta completar tres dígitos), y también al MCC (Mobile Country Code), que en el caso de Chile es el 730. Se puede decir entonces que son detalles a tener en cuenta el desplegar servicios en una operadora real, y que deben ser analizados en profundidad para poder realizar exitosamente el registro del cliente SIP Nokia vía la red GSM.

4.1.2 Servicios de Voz

En cuanto a los servicios de voz, tanto la plataforma SIP como el core IMS proveen casi transparentemente el servicio de telefonía IP, pero en este caso se desarrollaron las pruebas en un

mismo segmento de red. En un despliegue real, se comunican usuarios de redes distintas que muchas veces hacen uso de NAT, apareciendo entonces el problema del *NAT traversal* (el paso del flujo de datos a través de NATs) para el flujo de medios entre ellos. Si bien la señalización SIP puede ser enrutada debidamente, los medios encapsulados en el protocolo de transporte RTP son Peer-to-Peer, por lo que necesitan un método para ser enrutados adecuadamente.

Para evitar dicho inconveniente, una opción es utilizar Proxies para los medios RTP (RTP Proxies), que redirijan y enluten los medios entre los usuarios. OpenSER ofrece dicha alternativa mediante el módulo `nathhelper` y la utilización de `rtpproxy`, o en su defecto la utilización del módulo `mediaproxy`. Otra alternativa es utilizar servidores STUN o un SBC (Service Border Controller), que modifican tanto el flujo de señalización como el de medios para que no ocurran problemas con los NATs.

Por otro lado, un problema recurrente al intentar comunicar dos clientes IMS consistió en la caída sin razón aparente del S-CSCF del core. Si se intentaba llamar nuevamente bajo las mismas condiciones, la llamada podía ser cursada. Este hecho no hace más que acentuar el carácter de servicio en constante desarrollo de la implementación del core (OpenIMSCore), en la cual pueden surgir *bugs* que se van solucionando a medida que la comunidad los informa a los desarrolladores. Así, una comunicación fluida con los autores del software, y más aún, un aporte personal al desarrollo del tema pueden ayudar a remediar los problemas que brotan y contribuir a una constante mejora del programa.

En cuanto al servicio de Voicemail, si bien se implementó sin problemas en la plataforma SIP, se presentaron problemas al intentar desplegar el servicio en el core IMS. Se realizaron las configuraciones respectivas y aprovisionamiento adecuado mediante la interfaz gráfica del HSS, pero al intentar contactar al servidor, el S-CSCF no encontraba al AS en la dirección especificada y dejaba de funcionar. Debido a esto, se debía ejecutar nuevamente el script correspondiente al S-CSCF para recobrar su funcionalidad. Esto se considera como un error relacionado con el punto anterior, a lo cual se recomienda indagar a fondo en las razones por las cuales el S-CSCF cesa su tarea de forma errática.

4.1.3 Presencia y Mensajería Instantánea

En el caso del servicio de presencia basado en Web Services Parlay X, el cliente a utilizar es el llamado Oracle Communicator, pues el ejemplo implementado considera aportes propietarios que obligan a ocupar dicho software. Este punto es importante, ya que las plataformas propietarias cierran la puerta al desarrollo de nuevos o mejores servicios. Si bien en este caso se ofrece una plataforma de desarrollo mediante JDeveloper, el hecho de tener que utilizar un cliente propietario sin posibilidad de agregarle funcionalidades o características hace que se dificulte su adopción masiva.

El principal punto a discutir en este caso corresponde a las diferencias entre los mensajes XML que transportan la información de presencia, de acuerdo al cliente utilizado. A continuación se presenta un ejemplo de información de presencia para cada uno de los clientes considerados (UCT IMS Client, X-Lite y Oracle Communicator en conjunto con el Web Service Parlay X):

Tabla 35: XML Presencia UCT

```
<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
xmlns:im="urn:ietf:params:xml:ns:pidf:im"
entity="sip:diego@playsip.lab">
[09]<tuple id="UCTIMSClient">
[09][09]<status>
[09][09][09]<basic>open</basic>
[09][09]</status>
[09][09]<note>Away</note>
[09]</tuple>
</presence>
```

Tabla 36: XML Presencia X-Lite

```
<?xml version='1.0' encoding='UTF-8'?>
<presence xmlns='urn:ietf:params:xml:ns:pidf'
xmlns:dm='urn:ietf:params:xml:ns:pidf:data-model'
xmlns:rpid='urn:ietf:params:xml:ns:pidf:rpid'
xmlns:c='urn:ietf:params:xml:ns:pidf:cipid'
entity='sip:300@172.16.231.213'>
     <tuple id='te13d9a46'>
          <status>
               <basic>open</basic>
          </status>
     </tuple>
     <dm:person id='pa662dc2b'>
          <rpid:activities><rpid:away/></rpid:activities>
          <dm:note>Away</dm:note>
     </dm:person>
</presence>
```

Tabla 37: XML Presencia Web Service Parlay X

```
<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
entity="sip:adam@ocmsdemo.com">
        <tuple id="A4bc1a1f9-eb62-40b2-bb4b-194139ed3d01">
                <status>
                        <basic>open</basic>
                </status>
                <deviceID xmlns="urn:ietf:params:xml:ns:pidf:data-
model">mac:001a7348c8a9</deviceID>
                        <note>Oracle Communicator</note>
        </tuple>
        <device xmlns="urn:ietf:params:xml:ns:pidf:data-model"
id="C4bc1a1f9-eb62-40b2-bb4b-194139ed3d01">
                <deviceID>mac:001a7348c8a9</deviceID>
        </device>
        <person xmlns="urn:ietf:params:xml:ns:pidf:data-model"
id="B4bc1a1f9-eb62-40b2-bb4b-194139ed3d01"><activities
xmlns="urn:ietf:params:xml:ns:pidf:rpid"/>
                <note/>
        </person>
</presence>
```

Como se puede apreciar, las diferencias saltan a la vista. Cada cliente implementa su propio "sabor" de la información de presencia, lo que a la larga produce problemas de compatibilidad entre clientes. Si bien el formato de la información está normado (IETF RFC

3863), los clientes hacen su propia interpretación del estándar y presentan la información de manera distinta, cayendo en la incompatibilidad citada anteriormente.

4.1.4 Otros Servicios

El servicio de C2D (Click-to-Dial) no pudo ser implementado de forma correcta, ni con OpenSER+WeSIP, ni con Mobicents, ni con UCT B2BUA. Si bien con estos dos últimos se lograba contactar a las dos partes, al momento en que ambas reciben sus respectivos ACK y empieza el flujo de datos, la comunicación se cae y no llega a existir un intercambio de paquetes de voz encapsulados en RTP, generándose un BYE hacia uno de los involucrados y terminándose la sesión. Hasta el momento no se ha podido dar con el origen de este problema, por lo que se deberá indagar más profundamente en futuros trabajos.

En el caso de IPTV, para IMS la implementación es totalmente intuitiva y transparente, mientras que para un usuario SIP se hace necesario tener en cuenta el soporte de los codecs involucrados en la provisión del stream de video. Éstos se negocian previamente mediante el protocolo SDP: el INVITE inicial incluye los codecs que soporta el cliente, el servidor responde con un OK con los cuales él puede enviar los medios, llegándose a un acuerdo final del codec a utilizar mediante el ACK que envía finalmente el usuario.

Tabla 38: Codecs IPTV

```
ACK sip:channel1@172.16.231.213:7070 SIP/2.0
Via: SIP/2.0/UDP 172.16.231.213:6060;branch=0
Via: SIP/2.0/UDP 172.16.231.213:4060;branch=0
Via: SIP/2.0/UDP 172.16.231.213:5061;rport=5061;branch=z9hG4bK1721522400
From: "Diego" <sip:diego@playsip.lab>;tag=866150752
To: <sip:channel1@iptv.playsip.lab>;tag=254462604
Call-ID: 1377316301
CSeq: 20 ACK
Contact: <sip:diego@172.16.231.213:5061>
Content-Type: application/sdp
Max-Forwards: 15
User-Agent: UCT IMS Client
Content-Length:   303
P-Asserted-Identity: <sip:diego@playsip.lab>

v=0
o=- 0 0 IN IP4 172.16.231.213
s=IMS_Call
c=IN IP4 172.16.231.213
t=0 0
m=audio 0 RTP/AVP
m=video 17145 RTP/AVP 96
b=AS:128
a=curr:qos local sendrecv
a=curr:qos remote none
a=des:qos none local sendrecv
a=des:qos none remote sendrecv
a=rtpmap:96 H263-1998
a=fmtp:96 profile-level-id=0
```

Por ejemplo, en este caso se indica que no habrá audio durante el stream, y que el video será transmitido en el formato H.263+ (H.263-1998). Si bien el cliente X-Lite manifiesta soporte

para dicho codec, las pruebas realizadas con éste no fueron satisfactorias en cuanto a video, lo que saca a la luz una incompatiblidad surgida de una compatibilidad aparente, la cual se debe tener en cuenta.

Por último, el envío de SMS vía Web y conexión a un SMSC presenta varios puntos interesantes a considerar. Primero que todo, la capacidad de hacer transparente al usuario el envío de información específica (en este caso, información de presencia), permite que se puedan proveer servicios de valor agregado utilizando dicha característica. Además, la capacidad de poder realizarlo sobre un cliente desplegado masivamente como lo es un PC o un teléfono móvil, permite que los servicios puedan llegar a más usuarios y enriquecer su experiencia de uso. Por último, se abre la puerta a proveedores o terceros para que puedan proveer sus propios contenidos, llegando a acuerdos monetarios con el respectivo operador de la red.

4.2 Orquestación de Servicios

La orquestación o combinación de servicios se hace difícil cuando los clientes no son totalmente compatibles, como es el caso que se dio durante el trabajo desarrollado. Aún así, existe la posibilidad de modificar los clientes o desarrollar aplicaciones específicas sobre APIs o suites de desarrollo para plataformas específicas. Por ejemplo, el teléfono móvil Nokia utilizado presenta un sistema operativo Symbian S60 con un cliente SIP nativo pero algo limitado; lo interesante es que existe un API con el cual se pueden desarrollar aplicaciones para dicha plataforma en diversos lenguajes. Por ejemplo, para el lenguaje C o C++ existe la suite Carbide C/C++ que permite programar aplicaciones para S60 y probarlas en un emulador ad-hoc previamente a su despliegue en un teléfono móvil real.

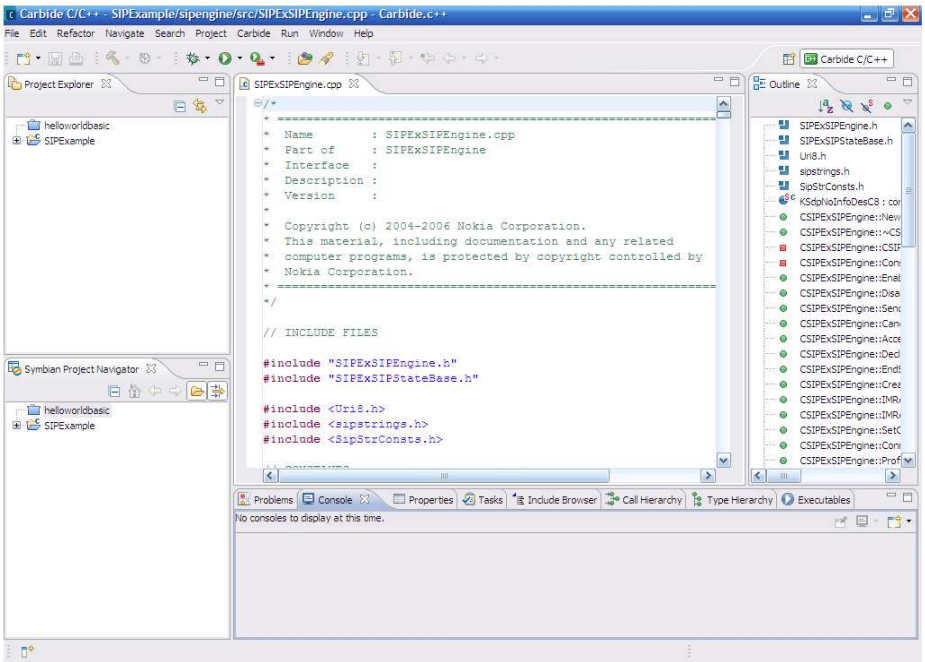

Figura 4-1: Carbide C/C++

Por otro lado, para que la orquestación de servicios se pueda llevar a cabo, los servidores de aplicaciones SIP deben soportar las extensiones al protocolo que introduce IMS, tales como los P-Headers. Es importante entonces que el desarrollo tanto de las aplicaciones como de los

servidores que las proveen evolucione en una misma dirección, guiado por estándares y normas que eviten las incompatibilidades.

4.3 Integración con Plataformas Existentes

Como se mencionaba anteriormente, el hecho de que algunos teléfonos móviles traigan incorporado un cliente SIP abre la puerta para proveer aplicaciones ricas en contenido provistas por plataformas basadas en dicho protocolo, como IMS. Así también, el hecho que se puedan desarrollar aplicaciones nuevas sobre dichos terminales y con soporte para protocolos multimedia como SIP deriva en que la integración con las plataformas ya establecidas sea más transparente. Cabe mencionar nuevamente que la estandarización es clave en este aspecto, ya que la interfaces entre plataformas se encuentran normadas por los organismos internacionales de estandarización.

Capítulo 5

Conclusiones

IMS representa beneficios tanto para los operadores y propietarios de las redes como para los fabricantes de equipos, los proveedores o desarrolladores de servicios, y los usuarios finales. La arquitectura definida y sus interfaces estandarizadas permiten que proveer nuevos y mejores servicios de valor agregado en la red se haga de forma fácil y rápida, enriqueciendo así la experiencia del usuario final. El operador puede desarrollar servicios propios o bien incluir servidores de aplicaciones de terceros, mientras cumplan con las normas IMS para asegurar la compatibilidad.

Por otro lado, IMS permite que las aplicaciones se ofrezcan sin importar el tipo de acceso o terminal de usuario, lo que se traduce en una transversalidad en la provisión de servicios que beneficia al usuario. Así, éste puede tener acceso a sus propios servicios desde cualquier tipo de terminal, en el momento que él desee. Además, la información del usuario es accesible por las diferentes aplicaciones de manera centralizada; una identidad única a través de la red permite que la ubicuidad de servicios pueda ser real.

El protocolo SIP proporciona un método de señalización eficiente y preciso, facilitando la provisión de servicios multimedia y permitiendo sostener varias sesiones multimedia simultáneas. Aun así, es necesario proveer los medios para que el flujo de datos entre usuarios sea transparente y fluido en el tránsito por los nodos que componen la red.

Un aspecto a destacar es la importancia del cliente de software que utiliza el usuario al ejecutar una aplicación. Los servicios desplegados y capaces de serlo dependen fuertemente de las capacidades del terminal y el software ejecutado sobre éste. Así, cobra relevancia el desarrollo de clientes que soporten de la mejor manera posible los servicios provistos y permitan que la interacción con la plataforma de servicios se haga de la manera más transparente posible.

Existen terminales compatibles con IMS ya en circulación, los cuales permitirán una evolución más rápida en el uso de aplicaciones relacionadas con las nuevas capacidades de la red. Además, las APIs y otras plataformas de desarrollo permiten que los clientes puedan actualizarse y obtener nuevas características conducentes a la compatibilidad con las nuevas redes.

Como se mencionó en la discusión, la conformidad con las especificaciones de los organismos de estandarización es clave para poder proveer servicios compatibles entre distintos tipos de terminales, ya sea por el tipo de acceso que representan o por las interfaces que utilizan para comunicarse dentro de la red. La provisión de aplicaciones y comunicación entre terminales, como ya se detalló, se ve complicada por diferencias entre los clientes de software, pero los servicios estandarizados permiten el desarrollo rápido de aplicaciones personalizadas que aumentan la compatibilidad entre dichas entidades de usuario. Así, es recomendable elegir estándares abiertos, pues los protocolos propietarios retrasan el ingreso de nuevas tecnologías y aumentan su costo de implementación, principalmente en cuanto a integración.

Por otro lado, es perfectamente posible desplegar una arquitectura básica de aplicaciones para redes convergentes basada en software de código libre. Si se requieren funcionalidades adicionales o mayor robustez, deben tomarse ciertas consideraciones. Generalmente los programas de código abierto presentan ciertos problemas que se van solucionando a medida que van ocurriendo, liberando actualizaciones al software periódicamente, por lo que se debe tener cuidado para mantenerse al tanto de dichos cambios.

En el caso particular del trabajo desarrollado, la plataforma implementada permite proveer servicios básicos de telecomunicaciones y ciertas aplicaciones avanzadas, a clientes SIP e IMS, con ciertas limitaciones que pueden ser mejoradas en el futuro. Así, representa una primera aproximación a una plataforma de servicios para redes convergentes, cuyos problemas de compatibilidad pueden ser analizados y mejorados por quien se interese.

El siguiente paso puede consistir en transformar la arquitectura desplegada en una plataforma portable, con tal de proveer los servicios de una manera rápida en redes que no tienen dichas funcionalidades en la actualidad, o actuar como un laboratorio para probar las aplicaciones en un core IMS ya desplegado. Se recomienda también trabajar más a fondo en la combinación y orquestación de servicios ya desplegados en la plataforma, con tal de proveer servicios avanzados que brinden una mejor experiencia al usuario. Por último, también se puede trabajar en relación al registro en el core IMS de teléfonos móviles con stack SIP, vía la red GSM, punto clave para masificar el uso de las aplicaciones en terminales que se encuentran actualmente en circulación.

Referencias

[1] BATES, J. et al. 2006. Converged Multimedia Networks. Inglaterra, John Wiley & Sons. 348p.

[2] CISCO. Supporting the IP Multimedia Subsystem for Mobile, Wireline, and Cable Providers [en línea]. http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns549/net_implementation_white_paper0900aecd80395cb0_ns609_Networking_Solutions_White_Paper.html [Consulta: septiembre 2007]

[3] WIKIPEDIA. 3GPP [en línea]. http://en.wikipedia.org/wiki/3GPP [Consulta: octubre 2007]

[4] ZNATY, S. et al. IP Multimedia Subsystem: Principios y Arquitectura. EFORT [en línea]. http://www.efort.com/media_pdf/IMS_ESP.pdf [Consulta: noviembre 2007]

[5] PLAYSIP. Comunidad PlaySIP [en línea]. http://www.playsip.org/ [Consulta: agosto 2007]

[6] TECH INVITE. ABNF Grammar for SDP – Session Description Protocol (RFC 4566) [en línea]. http://www.tech-invite.com/Ti-sdp-abnf.html [Consulta: octubre 2007]

[7] FRAUNHOFER FOKUS. The Open IMS Core Project [en línea]. http://www.openimscore.org/ [Consulta: enero 2008]

[8] POIKSELKÄ, M. et al. 2004. The IMS: IP Multimedia Concepts and Services in the Mobile Domain. Inglaterra, John Wiley & Sons. 419p.

[9] UNMEHOPA, M. et al. 2006. Parlay/OSA: From Standards to Reality. Inglaterra, John Wiley & Sons.

[10] CAICEDO, O. Parlay/OSA [en línea]. http://wapcolombia.unicauca.edu.co/documentos/osaparlay.ppt [Consulta: noviembre 2007]

[11] IBM. Business Apps via Telco Gateway, Part 1: Introduction to the Parlay Architecture [en línea]. http://www.ibm.com/developerworks/library/wi-telco/index.html [Consulta: noviembre 2007]

[12] OMA. 2005. WV-040 System Architecture Model, Instant Messaging and Presence Service [IMPS] v1.2.

[13] MÄKELÄINEN, S. y MICHAEL, M. P. 2005. OMA IMPS (Previously Wireless Village) [en línea]. http://martinpeter.michael.googlepages.com/OMAIMPS.pdf [Consulta: enero 2008]

[14] NOKIA. SIP Applications and Services [en línea]. http://www.forum.nokia.com/document/Forum_Nokia_Technical_Library/contents/FNTL/SIP_applications_and_services.htm [Consulta: octubre 2007]

[15] TECH INVITE. RFC 3261's Main Example Revisited [en línea]. http://www.tech-invite.com/Ti-sip-CF3261.html [Consulta: octubre 2007]

[16] SIPCENTER. Programming SIP [en línea]. http://www.sipcenter.com/sip.nsf/html/Programming+SIP [Consulta: octubre 2007]

[17] SIPCENTER. Addressing survivability and scalability of SIP networks by using Peer-to-Peer protocols [en línea]. http://www.sipcenter.com/sip.nsf/html/AG+P2P+SIP [Consulta: octubre 2007]

[18] WIKIPEDIA. Service Delivery Platform [en línea]. http://en.wikipedia.org/wiki/Service_Delivery_Platform [Consulta: enero 2008]

[19] WIKIPEDIA. Entorno de Desarrollo Integrado [en línea]. http://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado [Consulta: enero 2008]

[20] WIKIPEDIA. Servicio Web [en línea]. http://es.wikipedia.org/wiki/Servicio_Web [Consulta: enero 2008]

[21] WIKIPEDIA. Test Plan [en línea]. http://en.wikipedia.org/wiki/Test_plan [Consulta: enero 2008]

[22] OPENSER. OpenSER Project [en línea]. http://www.openser.org/ [Consulta: septiembre 2007]

[23] UBIQUITY. Ubiquity SIP Application Server [en línea]. http://www.ubiquity.net/ [Consulta: septiembre 2007]

[24] GLASSFISH. Glassfish – Open Source Application Server [en línea]. https://glassfish.dev.java.net/ [Consulta: septiembre 2007]

[25] WESIP. WeSIP – The OpenSER Converged Application Server [en línea]. http://www.wesip.com/mediawiki/index.php/Main_Page [Consulta: septiembre 2007]

[26] SAILFIN. SailFin Project [en línea]. https://sailfin.dev.java.net/ [Consulta: septiembre 2007]

[27] NATAS. A Beginners Guide to OpenSER [en línea]. http://www.oldskoolphreak.com/tfiles/voip/beginners_openser.txt [Consulta: octubre 2007]

[28] TRIXBOX. trixbox CE – The Open Platform for Business Telephony [en línea]. http://www.trixbox.org/ [Consulta: enero 2008]

[29] IBARRA, S. 2007. HOWTO: OpenSER + Asterisk como Voicemail Server [en línea]. http://www.saghul.net/blog/2007/08/22/howto-openser-asterisk-como-voicemail-server/ [Consulta: enero 2008]

[30] IBARRA, S. 2007. HOWTO: Presencia SIMPLE con OpenSER [en línea]. http://www.saghul.net/blog/2007/09/15/howto-presencia-simple-con-openser/ [Consulta: enero 2008]

[31] UCT. How to Set Up a Back-to-Back User Agent in Your IMS Network [en línea]. http://uctimsclient.berlios.de/back-2-back_user_agent_howto.html [Consulta: febrero 2008]

[32] KANNEL. Kannel: Open Source WAP and SMS Gateway [en línea]. http://www.kannel.org/ [Consulta: marzo 2008]

[33] ORACLE. Oracle JDeveloper [en línea]. http://www.oracle.com/technology/products/jdev/index.html [Consulta: marzo 2008]

[34] UCT. How to Run the UCT IPtv Streaming Server [en línea]. http://uctimsclient.berlios.de/uctiptv_howto.html [Consulta: febrero 2008]

[35] IETF RFC Page [en línea]. http://www.ietf.org/rfc.html [Consulta: octubre 2007]

[36] MIRANDA, J. 2008. Construcción de Laboratorios Docentes para Arquitectura IMS. Memoria para optar al título de Ingeniero Civil Electricista. Santiago, Chile. Facultad de Ciencias Físicas y Matemáticas, Universidad de Chile.

Acrónimos

3GPP	Third Generation Partnership Project
3PCC	Third Party Call Control
AAA	Authentication, Authorization and Accounting
AKA	Authentication and Key Agreement
API	Application Programming Interface
AS	Application Server
ASCII	American Standard Code for Information Interchange
B2BUA	Back-to-Back User Agent
BSS	Business Support System
C2D	Click-to-Dial
CAMEL	Customized Applications for Mobile Network Enhanced Logic
CAP	CAMEL Application Part
CDMA	Code Division Multiple Access
CDR	Charging Data Record
CFB	Call Forwarding Busy
CFNA	Call Forwarding No Answer
CFNR	Call Forwarding on No Reply
CFU	Call Forwarding Unconditional
CGI	Common Gateway Interface
CPL	Call Processing Language
CSCF	Call Session Control Function
CSE	CAMEL Service Environment
DNS	Domain Name System
DoS	Denial of Service
EDGE	Enhanced Data Rate for GSM Evolution
ENUM	Electronic Number Mapping
GGSN	Gateway GPRS Support Node
GPRS	Global System for Mobil communications
GUI	Graphical User Interface
HSDPA	High Speed Downlink Packet Access
HSS	Home Subscriber Server
HTTP	Hipertext Transfer Protocol
I-CSCF	Interrogating Call Session Control Function
IDE	Integrated Development Environment
IETF	Internet Engineering Task Force
iFC	Initial Filter Criteria
IMPS	Instant Messaging and Presence Service
IMS	IP Multimedia Subsystem
IMSI	International Mobile Subscriber Identity
IN	Intelligent Network
INAP	Intelligent Network Application Part
IP	Internet Protocol
ISDN	Integrated Services Digital Network
ISUP	ISDN User Part

LTE	Long Term Evolution
MCC	Mobile Country Code
MIME	Multimedia Internet Message Extensions
MNC	Mobile Network Code
MPLS	MultiProtocol Label Switching
MRF	Multimedia Resource Function
MRFC	Multimedia Resource Function Controller
MRFP	Multimedia Resource Function Processor
OMA	Open Mobile Alliance
OSA	Open Service Access
OSS	Operational Support System
P2P	Peer-to-Peer
P-CSCF	Proxy Call Session Control Function
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RADIUS	Remote Authentication Dial-In User Service
RFC	Request For Comments
RTP	Real Time Protocol
SCE	Service Creation Environment
SCIM	Service Capability Interaction Manager
SDP	Session Description Protocol
SDP	Service Delivery Platform
SIM	Subscriber Identity Module
SIMPLE	SIP for Instant Messaging and Presence Leverage Extensions
SIP	Session Initiation Protocol
SMPP	Short Message Peer-to-Peer Protocol
SMS	Short Message Service
SMSC	Short Message Service Center
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SPT	Service Point Trigger
SQL	Structured Query Language
SRF	Special Resource Function
SSF	Service Switching Function
SSP	Service Switching Point
TCP	Transfer Control Protocol
TLS	Transport Layer Security
TP	Trigger Point
UA	User Agent
UAC	User Agent Client
UAS	User Agent Server
UDP	User Datagram Protocol
UE	User Entity
UML	Unified Modeling Language
UMTS	Universal Mobile Telecommunications System
URI	Uniform Resource Identifier
UTRAN	UMTS Terrestrial Radio Access Network
VoIP	Voicer over IP
WAP	Wireless Application Protocol

WLAN	Wireless Local Area Network
WSDL	Web Services Description Language
XCAP	XML Configuration Access Protocol
xDSL	Digital Subscriber Line
XML	eXtensible Markup Language

Anexos

En este apartado se incluyen los anexos respectivos a los cuales se hace referencia en la descripción del trabajo.

Anexo 1: Ejemplo XML iFC

Tabla 39: Ejemplo XML iFC

```
<?xml version="1.0" encoding="UTF-8"?>
<IMSSubscription>
      <PrivateID>alice@playsip.lab</PrivateID>
      <ServiceProfile>
            <PublicIdentity>
                  <Identity>sip:alice@playsip.lab</Identity>
                  <Extension>
                        <IdentityType>0</IdentityType>
                  </Extension>
            </PublicIdentity>
            <InitialFilterCriteria>
                  <Priority>0</Priority>
                  <TriggerPoint>
                        <ConditionTypeCNF>1</ConditionTypeCNF>
                        <SPT>
                              <ConditionNegated>0</ConditionNegated>
                              <Group>0</Group>
                              <Method>PUBLISH</Method>
                              <Extension></Extension>
                        </SPT>
                        <SPT>
                              <ConditionNegated>0</ConditionNegated>
                              <Group>0</Group>
                              <Method>SUBSCRIBE</Method>
                              <Extension></Extension>
                        </SPT>
                        <SPT>
                              <ConditionNegated>0</ConditionNegated>
                              <Group>1</Group>
                              <SIPHeader>
                                    <Header>Event</Header>
                                    <Content>.*presence.*</Content>
                              </SIPHeader>
                              <Extension></Extension>
                        </SPT>
                  </TriggerPoint>
                  <ApplicationServer>
                        <ServerName>sip:172.16.231.213:5065</ServerName>
                        <DefaultHandling>0</DefaultHandling>
                  </ApplicationServer>
            </InitialFilterCriteria>
      </ServiceProfile>
</IMSSubscription>

<?xml version="1.0" encoding="UTF-8"?>
```

```
<IMSSubscription>
      <PrivateID>alice@playsip.lab</PrivateID>
      <ServiceProfile>
            <PublicIdentity>
                  <Identity>sip:alice@playsip.lab</Identity>
                  <Extension>
                        <IdentityType>0</IdentityType>
                  </Extension>
            </PublicIdentity>
            <InitialFilterCriteria>
                  <Priority>0</Priority>
                  <TriggerPoint>
                        <ConditionTypeCNF>1</ConditionTypeCNF>
                        <SPT>
                              <ConditionNegated>0</ConditionNegated>
                              <Group>0</Group>
                              <Method>PUBLISH</Method>
                              <Extension></Extension>
                        </SPT>
                        <SPT>
                              <ConditionNegated>0</ConditionNegated>
                              <Group>0</Group>
                              <Method>SUBSCRIBE</Method>
                              <Extension></Extension>
                        </SPT>
                        <SPT>
                              <ConditionNegated>0</ConditionNegated>
                              <Group>1</Group>
                              <SIPHeader>
                                    <Header>Event</Header>
                                    <Content>.*presence.*</Content>
                              </SIPHeader>
                              <Extension></Extension>
                        </SPT>
                  </TriggerPoint>
                  <ApplicationServer>
                        <ServerName>sip:172.16.231.213:5065</ServerName>
                        <DefaultHandling>0</DefaultHandling>
                  </ApplicationServer>
            </InitialFilterCriteria>
            <InitialFilterCriteria>
                  <Priority>1</Priority>
                  <TriggerPoint>
                        <ConditionTypeCNF>1</ConditionTypeCNF>
                        <SPT>
                              <ConditionNegated>0</ConditionNegated>
                              <Group>0</Group>
                              <Method>INVITE</Method>
                              <Extension></Extension>
                        </SPT>
                        <SPT>
                              <ConditionNegated>0</ConditionNegated>
                              <Group>1</Group>
                              <SIPHeader>
                                    <Header>To</Header>

      <Content>.*iptv.playsip.lab.*</Content>
                              </SIPHeader>
                              <Extension></Extension>
```

```
                        </SPT>
                    </TriggerPoint>
                    <ApplicationServer>
                        <ServerName>sip:172.16.231.213:7070</ServerName>
                        <DefaultHandling>0</DefaultHandling>
                    </ApplicationServer>
                </InitialFilterCriteria>
        </ServiceProfile>
</IMSSubscription>
```

Anexo 2: Configuración Básica OpenSER

Tabla 40: Configuración Básica OpenSER

```
#
# $Id: openser.cfg basic $
#

# ----------- global configuration parameters ------------------------

debug=9            # debug level (cmd line: -ddddddddddd)
fork=yes
log_stderror=yes    # (cmd line: -E)
children=4
# Uncomment these lines to enter debugging mode
#fork=no
#log_stderror=yes
#

port=5060

# uncomment the following lines for TLS support
#disable_tls = 0
#listen = tls:your_IP:5061
#tls_verify_server = 1
#tls_verify_client = 1
#tls_require_client_certificate = 0
#tls_method = TLSv1
#tls_certificate = "/usr/local/etc/openser/tls/user/user-cert.pem"
#tls_private_key = "/usr/local/etc/openser/tls/user/user-privkey.pem"
#tls_ca_list = "/usr/local/etc/openser/tls/user/user-calist.pem"

# ------------------ module loading ----------------------------------

#set module path
mpath="/usr/local/lib/openser/modules/"

# Uncomment this if you want to use SQL database
loadmodule "mysql.so"

loadmodule "sl.so"
loadmodule "tm.so"
loadmodule "rr.so"
loadmodule "maxfwd.so"
loadmodule "usrloc.so"
loadmodule "registrar.so"
loadmodule "textops.so"
```

```
loadmodule "mi_fifo.so"

# Uncomment this if you want digest authentication
# mysql.so must be loaded !
loadmodule "auth.so"
loadmodule "auth_db.so"
loadmodule "uri_db.so"
# ----------------- setting module-specific parameters ---------------

# -- mi_fifo params --

modparam("mi_fifo", "fifo_name", "/tmp/openser_fifo")

# -- usrloc params --

modparam("usrloc", "db_url",
"mysql://openser:openserrw@ip_location/openser")
#modparam("usrloc", "db_mode",   0)

# Uncomment this if you want to use SQL database
# for persistent storage and comment the previous line
modparam("usrloc", "db_mode", 3)

# -- auth params --
# Uncomment if you are using auth module
#
modparam("auth_db", "calculate_ha1", yes)
#
# If you set "calculate_ha1" parameter to yes (which true in this
config),
# uncomment also the following parameter)
#
modparam("auth_db", "password_column", "password")
modparam("auth_db", "db_url",
"mysql://openser:openserrw@ip_location/openser")

# --uri params --
modparam("uri_db", "db_url",
"mysql://openser:openserrw@ip_location/openser")

# -- rr params --
# add value to ;lr param to make some broken UAs happy
modparam("rr", "enable_full_lr", 1)

# -------------------------  request routing logic -------------------

# main routing logic

route{

        # initial sanity checks -- messages with
        # max_forwards==0, or excessively long requests
        if (!mf_process_maxfwd_header("10")) {
                sl_send_reply("483","Too Many Hops");
                exit;
        };

        if (msg:len >=  2048 ) {
                sl_send_reply("513", "Message too big");
```

```
                exit;
        };

        # we record-route all messages -- to make sure that
        # subsequent messages will go through our proxy; that's
        # particularly good if upstream and downstream entities
        # use different transport protocol
        if (!method=="REGISTER")
                record_route();

        # subsequent messages withing a dialog should take the
        # path determined by record-routing
        if (loose_route()) {
                # mark routing logic in request
                append_hf("P-hint: rr-enforced\r\n");
                route(1);
        };

        if (!uri==myself) {
                # mark routing logic in request
                append_hf("P-hint: outbound\r\n");
                # if you have some interdomain connections via TLS
                #if(uri=~"@tls_domain1.net") {
                #       t_relay("tls:domain1.net");
                #       exit;
                #} else if(uri=~"@tls_domain2.net") {
                #       t_relay("tls:domain2.net");
                #       exit;
                #}
                route(1);
        };

        # if the request is for other domain use UsrLoc
        # (in case, it does not work, use the following command
        # with proper names and addresses in it)
        if (uri==myself) {

                if (method=="ACK") {
                     route(1);
                     exit;
                } if (method=="INVITE") {
                     route(3);
                     exit;
                } if (method=="REGISTER") {
                     route(2);
                     exit;
                };

                lookup("aliases");
                if (!uri==myself) {
                        append_hf("P-hint: outbound alias\r\n");
                        route(1);
                };

                # native SIP destinations are handled using our USRLOC DB
                if (!lookup("location")) {
                        sl_send_reply("404", "Not Found");
                        exit;
                };
```

```
                append_hf("P-hint: usrloc applied\r\n");
        };

        route(1);
}


route[1] {
        # send it out now; use stateful forwarding as it works reliably
        # even for UDP2TCP
        if (!t_relay()) {
                sl_reply_error();
        };
        exit;
}

# route[2] will route REGISTER messages towards the registrar
# No database connectivity is required here, as the registrar then sends
the messages to the location server
route[2] {
       #rewriting the host allows the registrar to know that the message
is destined for it
       rewritehost("ip_registrar");
       if(!t_relay("udp:ip_registrar:puerto_registrar")) {
              sl_reply_error();
       };
#       t_relay();
        exit;
}

route[3] {
        # -----------------------------------------------------------------
        # INVITE Message Handler
        # -----------------------------------------------------------------
        if (!proxy_authorize("dominio.com","subscriber")) {
                proxy_challenge("dominio.com","0");
                exit;
        } else if (!check_from()) {
                sl_send_reply("403", "Use From=ID");
                exit;
        };
        consume_credentials();
        lookup("aliases");
        if (uri!=myself) {
                route(1);
                exit;
        };
        if (!lookup("location")) {
                sl_send_reply("404", "User Not Found");
                exit;
        };
        route(1);
}
```

Anexo 3: Configuración Básica Gateway SMS Kannel

Tabla 41: Configuración Básica Gateway SMS Kannel

```
#------------------------------------------
# CORE
#
# There is only one core group and it sets all basic settings
# of the bearerbox (and system). You should take extra notes on
# configuration variables like 'store-file' (or 'store-dir'),
# 'admin-allow-ip' and 'access.log'

group = core
admin-port = 13000
smsbox-port = 13001
admin-password = bar
#status-password = foo
admin-deny-ip = "*.*.*.*"
admin-allow-ip = "ip_1;ip_2;ip_3"
log-file = "/home/user/kannel/kannel.log"
log-level = 0
box-deny-ip = "*.*.*.*"
box-allow-ip = "ip_1;ip_2;ip_3"
#unified-prefix = "+358,00358,0;+,00"
access-log = "/home/user/kannel/access.log"
store-file = "/home/user/kannel/kannel.store"
#ssl-server-cert-file = "cert.pem"
#ssl-server-key-file = "key.pem"
#ssl-certkey-file = "mycertandprivkeyfile.pem"
dlr-storage = mysql

#------------------------------------------
# SMSC CONNECTIONS
#
# SMSC connections are created in bearerbox and they handle SMSC specific
# protocol and message relying. You need these to actually receive and
send
# messages to handset, but can use GSM modems as virtual SMSCs

# This is a fake smsc connection, _only_ used to test the system and
services.
# It really cannot relay messages to actual handsets!

#group = smsc
#smsc = fake
#smsc-id = FAKE
#port = 10000
#connect-allow-ip = *.*.*.*

group = smsc
smsc = smpp
host = ip_smsc
port = puerto_smsc
receive-port = puerto_recepcion_smsc
smsc-username = "user"
smsc-password = "pass"
system-type = "system_type"
address-range = ""

#------------------------------------------
# SMSBOX SETUP
#
```

```
# Smsbox(es) do higher-level SMS handling after they have been received
from
# SMS centers by bearerbox, or before they are given to bearerbox for
delivery

group = smsbox
bearerbox-host = 127.0.0.1
sendsms-port = 13013
global-sender = 12341234
#sendsms-chars = "0123456789 +-"
log-file = "/home/user/kannel/smsbox.log"
log-level = 0
access-log = "/home/user/kannel/access.log"

#-----------------------------------------------
# SEND-SMS USERS
#
# These users are used when Kannel smsbox sendsms interface is used to
# send PUSH sms messages, i.e. calling URL like
# http://kannel.machine:13013/cgi-
bin/sendsms?username=tester&password=foobar...

group = sendsms-user
username = tester
password = foobar
#user-deny-ip = ""
#user-allow-ip = ""

#-----------------------------------------------
# SERVICES
#
# These are 'responses' to sms PULL messages, i.e. messages arriving from
# handsets. The response is based on message content. Only one sms-
service is
# applied, using the first one to match.

group = sms-service
keyword = nop
text = "You asked nothing and I did it!"

# There should be always a 'default' service. This service is used when
no
# other 'sms-service' is applied.

group = sms-service
keyword = default
text = "No service specified"

#-----------------------------------------------
# MySQL DLR
#

group = mysql-connection
id = mydlr
host = ip_host
username = kannel
password = kannel
database = dlr
max-connections = 1
```

```
group = dlr-db
id = mydlr
table = dlr
field-smsc = smsc
field-timestamp = ts
field-destination = destination
field-source = source
field-service = service
field-url = url
field-mask = mask
field-status = status
field-boxc-id = boxc
```

Anexo 4: Código PHP Web Presencia/Envío SMS

Tabla 42: Código PHP Web Presencia/Envío SMS

```
<html>
<head>
      <title>Presencia</title>
</head>
<body>

<?php
// Conexion, seleccion de base de datos
include ("../../Info/openser.php");
include ("../../Info/kannel.php");
$enlace = mysql_connect('172.16.231.213:3306', $db_user, $db_passwd) or
die('No pudo conectarse : ' . mysql_error());
echo "Conexi&oacute;n exitosa.\n";
mysql_select_db('openser') or die('No pudo seleccionarse la BD.');

// Realizar una consulta SQL
$consulta  = 'SELECT username,body FROM presentity';
$resultado = mysql_query($consulta) or die('La consulta fall&oacute;: ' .
mysql_error());

// Impresion de resultados en HTML
echo "<table border=1>\n";
while ($linea = mysql_fetch_array($resultado, MYSQL_ASSOC)) {
    echo "\t<tr>\n";
    foreach ($linea as $valor_col) {
        echo "\t\t<td>$valor_col</td>\n";
    }
    echo "\t</tr>\n";
}
echo "</table>\n";

// Liberar conjunto de resultados
mysql_free_result($resultado);

// Cerrar la conexion
mysql_close($enlace);
?>

<!--Enviar SMS-->
```

```
<FORM ACTION="http://172.16.231.213:24133/cgi-bin/sendsms" METHOD=GET>
<?php
echo "<INPUT TYPE=\"hidden\" NAME=\"username\" VALUE=\"".$sms_user."\">
<INPUT TYPE=\"hidden\" NAME=\"password\" VALUE=\"".$sms_passwd."\">\n";
?>
N&uacute;mero:
<INPUT TYPE="text" NAME="to">
<?php
echo "<INPUT TYPE=\"hidden\" NAME=\"text\"
VALUE=\"".strstr($valor_col,"<status>")."\">\n";
?>
<INPUT TYPE="submit" VALUE="Enviar SMS">
</FORM>

</body>
</html>
```