MECANISMO PARA LA INTEGRACIÓN DE LA TECNOLOGÍA JAIN SLEE EN UN ENTORNO IMS PARA LA CREACIÓN Y PRESTACIÓN DE SERVICIOS DE VALOR AGREGADO

MONOGRAFÍA

María del Pilar Joaquí Chimachaná
Julián Orlando Peña Peña

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telemática
Línea de Investigación: Servicios Avanzados en Telecomunicaciones
Popayán, Septiembre de 2010

MECANISMO PARA LA INTEGRACIÓN DE LA TECNOLOGÍA JAIN SLEE EN UN ENTORNO IMS PARA LA CREACIÓN Y PRESTACIÓN DE SERVICIOS DE VALOR AGREGADO

María del Pilar Joaquí Chimachaná
Julián Orlando Peña Peña

Director: Javier Alexander Hurtado

Trabajo de Grado presentado como requisito para optar al título de Ingeniero en Electrónica y Telecomunicaciones

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telemática
Línea de Investigación: Servicios Avanzados en Telecomunicaciones

Popayán, Septiembre de 2010

A mi madre Eunice por el inmenso amor que siempre me ha dado, por ser mi amiga y consejera, por estar siempre junto a mí, por ser el pilar y el motor de cada una de mis metas, por ser una gran mujer, admirable por su tenacidad y convicción. A ella, a la mujer que con el pasar del tiempo su cabello se ha teñido de hermosos rayitos blancos que reflejan el esplendor de su alma. Le dedico este triunfo.

A mis adoradas hermanas Leyla y Maritza por quererme tanto, por apoyarme en cada uno de los caminos que tomo, por su ternura, por estar siempre cuando las necesito a pesar de la distancia, por ayudar a mi madre a educarme, por convertirse en dos madres más que permitieron que mi vida siempre estuviera llena de alegrías y felicidad, y principalmente por siempre creer en mí.

María del Pilar Joaquí Chimachaná

A mis padres Orlando y Doris, quienes durante toda mi vida solo se han preocupado por ofrecerme lo mejor. A ellos les debo todo lo que soy.

A mi hermana Viviana, quien es mi principal motivación para seguir adelante.

Julian Orlando Peña Peña.

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

CAPITULO 1. INTRODUCCION A JAIN SLEE

Este capítulo aborda una de las principales temáticas del presente trabajo, JAIN SLEE (Java API for Integrated/Intelligent Networks – Service Logic Execution Environment, API Java para Redes Integradas/Inteligentes – Entorno para la Ejecución de Lógicas de Servicio), el cual explica de forma clara y puntual cada uno de los aspectos más sobresalientes de la tecnología como lo son su arquitectura y funcionamiento general.

1.1 JAIN

Es un estándar que define un conjunto de API Java que permiten el desarrollo de aplicaciones portables y convergentes, que pueden prestarse de forma segura tanto en redes públicas conmutadas como en redes IP; esto gracias a que fue definido por la iniciativa JAIN, la cual está conformada por más de 80 compañías tales como: Ericsson, Siemens, Ulticom, Telcordia, Oracle, entre otras, interesadas en obtener una plataforma en donde los servicios puedan ser creados y desplegados rápidamente [1].

Así, JAIN especifica un gran número de API estandarizadas y libres de uso, incluyendo API para el control de llamadas, a nivel de protocolo para señalización y API de alto nivel para el desarrollo de aplicaciones que garanticen la compatibilidad entre soluciones de diferentes proveedores [2].

Muchas de estas API de hecho son frecuentemente implementadas en aplicaciones dentro de equipos como *softswitchs*, *gateways de Parlay/OSA*, etc. Entre las empresas que han implementado estas API están: Ulticom Inc, Telcordia Technologies, Nuera Communications, Iperia Inc, NexTone Commu-nications Inc, Tellabans Inc. entre otras [3].

1.2 SLEE

Es un entorno de ejecución de aplicaciones o servicios especializados que ofrece baja latencia, alta disponibilidad, alto throughput[1], funcionamiento asincrónico, orientación a eventos, tolerancia a fallas, capacidades de gestión, y demás requerimientos típicos del mundo de las telecomunicaciones [4] [5].

Tradicionalmente, los operadores de telecomunicaciones han utilizado SLEE para

1 Throughput es el volumen de información que viaja a través de un sistema en un intervalo de tiempo específico.

la prestación de sus servicios de valor agregado (llamada en espera, transferencia de llamadas, números gratuitos, llamadas por cobrar, etc.); sin embargo, estos típicamente son soluciones propietarias que no exponen interfaces estándares y por lo tanto hacen muy difícil combinar servicios implementados con herramientas que provienen de otros proveedores, haciendo muy complicada la interacción entre diferentes SLEE.

1.3 JAIN SLEE

Es la versión Java de un Entorno para la Ejecución de Lógicas de Servicio, estandarizado mediante el Java Community Process[2] y formalizado en su última versión disponible en la especificación JSR 240 [4] [5].

Debido a que es un estándar originado dentro de la iniciativa JAIN garantiza a los proveedores que certifiquen sus implementaciones la interacción de sus productos [6] [7], así como también favorece a los desarrolladores de soluciones de telecomunicaciones, quienes pueden crear productos que no están ligados a un proveedor, equipo o software en particular, sino que son portables entre diferentes plataformas.

En resumen, significa que los desarrollos son creados una sola vez y tienen la capacidad de ser puestos en funcionamiento sobre diferentes SLEE con muy pocas o ninguna modificación requerida.

1.4 Arquitectura General de JAIN SLEE

Figura 1. Arquitectura de JAIN SLEE.

2 Es un programa dedicado a la estandarización de especificaciones técnicas de tecnologías Java.

La figura 1 muestra la arquitectura general de JAIN SLEE.

A continuación están descritos los componentes más importantes en mayor detalle.

1.4.1 Agentes Externos al SLEE

Estos elementos son la fuente de información del SLEE con base en la cual funciona, es decir, son aquellos que generan cualquier tipo de ocurrencia que pueda ser conducida e interpretada por el SLEE como un Evento, con el fin de desencadenar una secuencia de procesos que correspondan a la información recibida. Estas fuentes pueden ser equipos de telecomunicaciones, tales como: centrales telefónicas, *softswitchs*, centralitas PBX y pasarelas; así mismo, protocolos de telecomunicaciones como: INAP, TCAP, SIP, TCP, UDP, Telnet, etc. Por otra parte cualquier software, como por ejemplo un motor de bases de datos, también puede generar eventos hacia el SLEE en forma de paquetes de algún protocolo preestablecido [8] [9].

1.4.2 Adaptadores de Recursos

Los RA (Resource Adaptors, Adaptadores de Recursos) son componentes que están integrados al SLEE y corresponden a las interfaces que éste tiene hacia la red. Su función es adecuar toda la información que manejan desde y hacia los agentes externos a un formato comprensible para el SLEE, en este caso objetos Java denominados eventos [8].

En pocas palabras puede decirse que los RA son los responsables de abstraer toda la red hacia el SLEE y hacia los desarrolladores [10], facilitando la manipulación del comportamiento de recursos de red para la creación de servicios de una forma mucho más rápida y eficiente.

Los RA son particulares para cada tipo de agente externo, por ejemplo, existen RA para el protocolo SIP (Session Initiation Protocol, Protocolo para el Inicio de Sesión), para el protocolo Diameter[3], etc. Además, existen RA más especializados, destinados por ejemplo a una variación del protocolo SIP o al manejo de un grupo seleccionado de mensajes del total de mensajes del protocolo Diameter.

3 Es un protocolo de red que sirve como base para crear extensiones para proporcionar servicios de autenticación, autorización y auditoría.

1.4.3 Componentes de Aplicación

Los componentes que implementan la lógica de los servicios y aplicaciones son los SBB (Service Building Blocks, Bloques para la Construcción de Servicios) ya que son ellos quienes tienen definidas las acciones a ejecutar para los diferentes eventos recibidos. Son ejecutados dentro de un contenedor el cual es encargado de controlar su ciclo de vida y de configurar su entorno de ejecución [5]. Así, un servicio puede estar compuesto de uno o más SBB, a la vez que un SBB puede reutilizarse en múltiples servicios, siendo esta una de las principales facilidades para la reutilización de código.

1.4.4 Framework

El framework[4] del SLEE engloba un conjunto de funcionalidades que son expuestas para que sean usadas tanto por los SBB como por otros componentes del SLEE.

A continuación están descritos los principales elementos del framework del SLEE.

1.4.4.1 Enrutador de Eventos

Debido a que dentro del SLEE existen muchos servicios, y que cada uno de ellos funciona sobre un conjunto determinado de tipos de eventos, es necesario que todos los eventos que llegan al SLEE a través de los RA sean encaminados sólo hacia los SBB interesados en procesarlos, para esto existe el Enrutador de Eventos [4]. Su lugar dentro de la arquitectura puede verse en la figura 2.

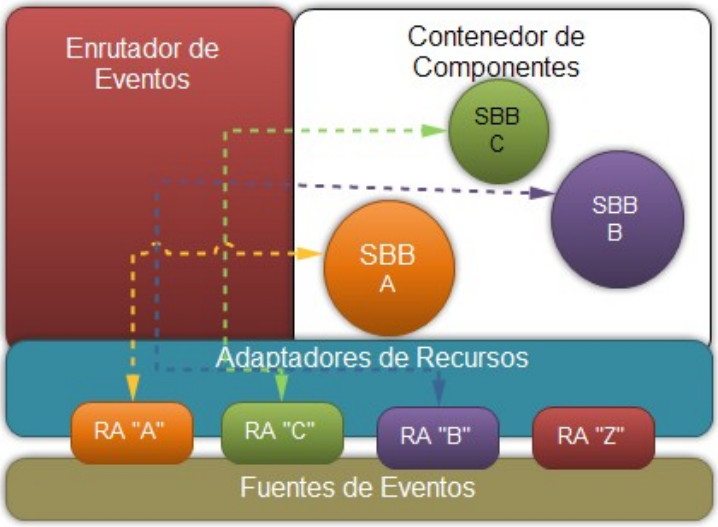

Figura 2. Enrutador de eventos.

4 Un framework es un marco de trabajo que facilita el desarrollo de software.

El Enrutador de Eventos es un componente fundamental del SLEE, y por ende su diseño e implementación deben ser guiados en todo momento pensando en el rendimiento, pues sus características lo hacen un candidato potencial para ser un cuello de botella que puede afectar el rendimiento de todo el SLEE [8]. Es por esto que en la especificación se incluye un modelo matemático riguroso destinado a guiar el diseño de estos Enrutadores para las implementaciones de JAIN SLEE.

1.4.4.2 Facilidades

Las Facilidades que ofrece el SLEE son funcionalidades típicas que tienden a ser usadas por gran parte de las aplicaciones y servicios, por lo cual los desarrolladores pueden enfocarse en la lógica de sus servicios y no en volver a implementar estas funcionalidades genéricas. Las siguientes son las facilidades incorporadas en la especificación 1.1 de JAIN SLEE [5] [8] [10]:

- **Facilidad de Temporización:** Ofrece temporizadores que pueden ser usados por los SBB para realizar tareas periódicas o que necesitan de conteos específicos de tiempo para el inicio de una acción determinada.

- **Facilidad de Alarmas:** Provee la funcionalidad de registrar alarmas dentro del SLEE con el objetivo de que sean monitoreadas por herramientas de gestión. Los RA, SBB y los Perfiles pueden registrar y quitar alarmas.

- **Facilidad de Trazado:** Brinda la funcionalidad de registrar trazas de información que puedan ser monitoreadas desde herramientas de gestión. Los RA, SBB, Perfiles y componentes internos del SLEE pueden hacer uso de esta facilidad.

- **Facilidad de Nombramiento de Contextos de Actividad:** Permite a los SBB dar nombres o alias a Contextos de actividad de modo que puedan ser usados desde otros SBB.

- **Facilidad de Perfiles:** Esta facilidad le permite a los SBB consultar y modificar datos almacenados en tablas de perfiles.

- **Facilidad de Búsqueda de Servicios:** Es usada solo por los RA con el propósito de consultar información sobre los tipos eventos que un servicio instalado en el SLEE puede recibir.

- **Facilidad de Búsqueda de Eventos:** Esta facilidad solo puede ser usada por los RA instalados en el SLEE y les permite consultar información sobre los tipos de eventos que están instalados.

1.4.5 Gestión

La especificación de JAIN SLEE ofrece capacidades de gestión a través de instrumentación JMX (Java Management Extensions, Extensiones de Gestión Java) de modo que clientes externos al SLEE pueden gestionarlo [6] [9].

Los componentes del SLEE que son gestionables son los siguientes:

- Todas las unidades Desplegables (Servicios, Perfiles, RA, Eventos, etc)
- La Facilidad de Alarmas.
- La Facilidad de Trazado.

1.4.6 Funcionamiento General de la Arquitectura

Con base en la figura 1 y las anteriores explicaciones, el funcionamiento general de la arquitectura de JAIN SLEE es como sigue:

Cuando se da una ocurrencia[5] en algún sistema externo conectado al SLEE que puede ser interpredata por este como un Evento, es enviado por su fuente hacia el RA correspondiente (por ejemplo, un mensaje SIP a través del adaptador de recursos SIP), posteriormente este es mapeado a un objeto Java y conducido al Enrutador de Eventos para finalmente llegar al SBB o SBB encargados de procesarlo según el tipo de Evento correspondiente; en este punto, los SBB ejecutan la lógica de aplicación definida por el desarrollador aprovechando o no las facilidades del SLEE. Una vez el Evento ha sido procesado, la respuesta o respuestas obtenidas pueden ser enviadas a destinos diferentes según la lógica, pudiendo ser éstos otros SBB o un SBB en particular, o la fuente de origen del Evento. El anterior comportamiento es típico en el SLEE, sin embargo este puede ser alterado a través de las capacidades de gestión.

1.5 Conceptos Fundamentales de la Especificación

La especificación de JAIN SLEE introduce una gran cantidad de conceptos, sin embargo, hay algunos que son esenciales y que son la base para el proceso de diseño de soluciones [5] [11].

1.5.1 Servicio

Un servicio[6] es una agrupación lógica y bien definida de SBB, que en conjunto

5 Puede ser por ejemplo levantar la bocina del teléfono para realizar una llamada, el deseo de enviar un mensaje de texto o un correo electrónico entre otros.

6 Esta definición se usará durante todo el trabajo, así mismo, se entenderá por aplicación al servicio o conjunto de

proveen un servicio más complejo que los sub-servicios que provee cada SBB por separado. Cada servicio tiene un SBB raíz, el cual cumple la función de inicialización del servicio, este SBB tiene la característica de que un ejemplar es obtenido automáticamente cuando el Enrutador de Eventos del SLEE recibe un Evento que aparece registrado como de interés para dicho Servicio.

Desde el punto de vista del SLEE, un servicio es un documento descriptor que apunta solamente al SBB raíz de dicho servicio.

1.5.2 SBB

La lógica de las aplicaciones y servicios desarrollados con JAIN SLEE es implementado en SBB. Debido a que los SBB funcionan con base en los eventos que envían y reciben, un SBB debe definir los tipos de eventos que puede manejar, de esta manera el Enrutador de Eventos no le enviará ningún Evento que el SBB no pueda recibir o no esté interesado en procesar. Para cada tipo de Evento declarado, el SBB debe implementar un método Java con la lógica encargada de procesar dicho tipo de evento. Son necesarios métodos diferentes para su envío y recepción.

Aunque los Eventos son el mecanismo primario de comunicación dentro del SLEE, la especificación de JAIN SLEE también brinda la posibilidad de realizar invocaciones sincrónicas entre SBB. Esta capacidad es utilizada solo entre SBB que pertenecen a un mismo servicio.

Cuando va a desarrollarse un servicio que consta de varios SBB, deben especificarse todas las relaciones padre a hijo dentro de la implementación mediante documentos descriptores, así como también las prioridades de acuerdo a las cuales un Evento debe ser entregado a dos o más SBB interesados en un mismo tipo de evento, de tal forma que el comportamiento del flujo de eventos e invocaciones entre los SBB queda unívocamente definido.

Para comprender más fácilmente las relaciones entre SBB, puede utilizarse una representación de sus relaciones mediante un grafo donde esten indicadas todas las relaciones padre a hijo existentes y las prioridades de entrega de eventos entre los SBB de un mismo o diferentes servicios, ya que un SBB puede pertenecer a servicios diferentes al mismo tiempo y muchos SBB pueden estar interesados en recibir un mismo tipo de evento.

servicios que prestan una funcionalidad con valor agregado hacia usuarios finales.

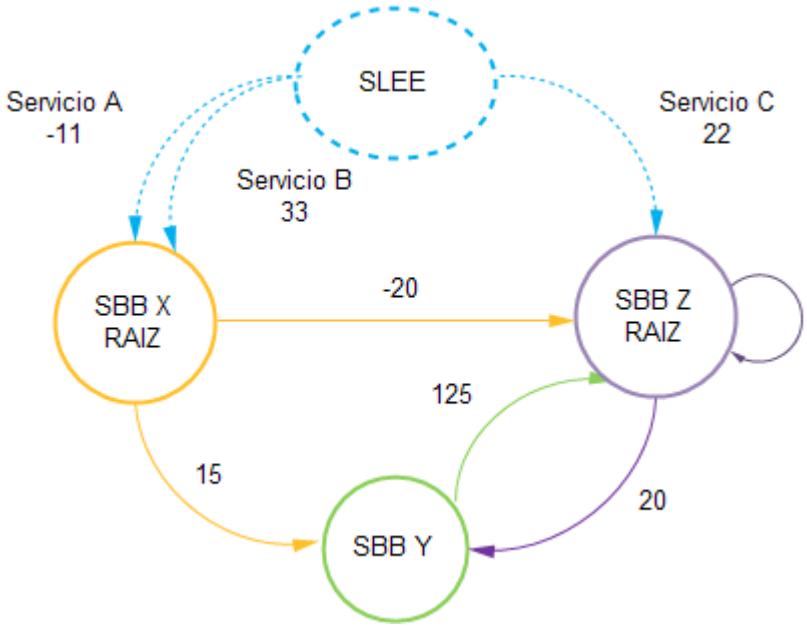

Figura 3. Grafo de relaciones entre SBB.

Como puede apreciarse en la figura 3, un SBB puede tener un SBB hijo de su mismo tipo, un SBB raíz de un servicio puede ser un SBB hijo de otro servicio y también muestra cómo el SLEE es el padre de todos los SBB raíz de todos los servicios.

La instancia de un SBB es denominada una Entidad SBB, no es un objeto java, es una entidad lógica que representa el estado persistente del SBB, ella es a su vez representada, cuando es necesario, por un Objeto Local de SBB, el cual finalmente si es un objeto Java común. Este es un mecanismo de protección ante fallas muy útil de JAIN SLEE, ya que de esta forma cuando ocurre un error grave y un Objeto Local de un SBB es perdido, simplemente se instancia otro y es cargado con toda la información que persistía en la Entidad SBB.

En el funcionamiento del SLEE, las relaciones entre SBB son representadas por los Arboles de Entidades de SBB, un ejemplo puede apreciarse en la figura 4.

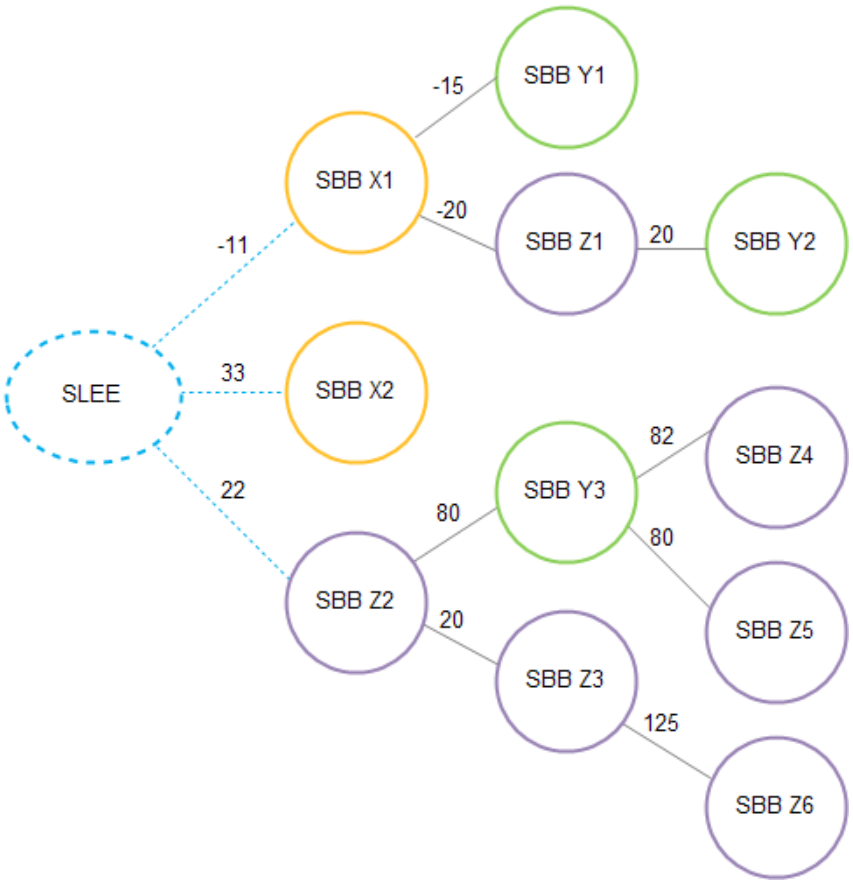

Figura 4. Árbol de Entidades de SBB.

De acuerdo a estos árboles el SLEE elimina las Entidades de SBB que ya no se necesiten, siempre eliminando primero a las entidades padre y luego a las entidades hijo.

1.5.3 Actividad

En la mayoría de los casos, muchos eventos están relacionados entre sí o pertenecen a una misma comunicación o sesión, por ejemplo, el establecimiento de una llamada telefónica, la cual es llevada a cabo en muchos pequeños pasos que son mapeados dentro del SLEE como eventos separados; debido a esto, fue creado el concepto de Actividad, por lo tanto ella representa un flujo de eventos relacionados.
Las actividades son representadas por Objetos de Actividad, es decir, este es la representación Java de una Actividad, el cual puede ser creado por las facilidades del SLEE o por entidades de RA; a su vez, este Objeto de Actividad es representado dentro del SLEE por un Contexto de Actividad, que a su vez es

representado y accede desde el SLEE haciendo uso de una Interfaz de Contexto de Actividad. De esta forma, cuando una Entidad de SBB debe manejar flujos relacionados de eventos, debe "suscribirse" al Contexto de Actividad correspondiente, el cual es accedido a través de una Interfaz de Contexto de Actividad representada por un Objeto de Interfaz de Contexto de Actividad.

Los contextos de actividad también sirven como un medio para la compartición de datos entre entidades de SBB que estén suscritas a un mismo Contexto de Actividad.

Las relaciones entre todos estos conceptos puede verse en la figura 5.

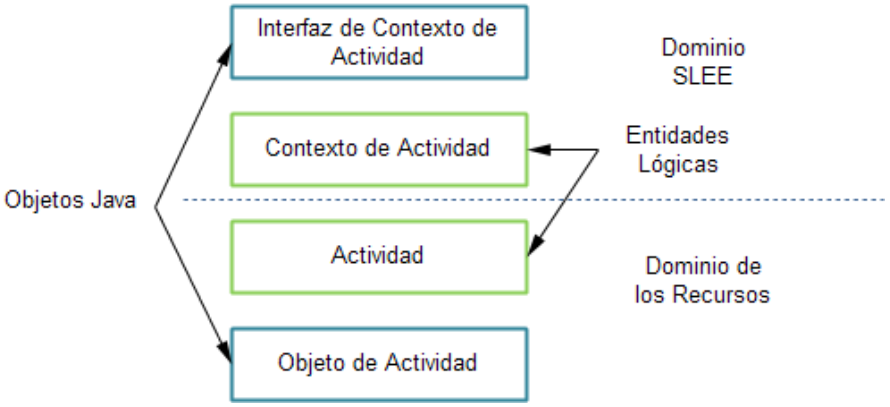

Figura 5. Actividad y Contexto de Actividad.

Otro aspecto importante acerca de la Interfaz de Contexto de Actividad es que puede ser de dos tipos, uno genérico proporcionado por el SLEE o uno personalizado proporcionado por el desarrollador, con el objetivo de definir la vista especifica que el SBB debe tener del Contexto de Actividad.

1.5.4 Perfiles

Los perfiles son un mecanismo altamente eficiente para el almacenamiento y el acceso a datos, estan enfocados hacia esquemas de datos poco complejos. Sin embargo, también pueden usarse bases de datos tradicionales mediante el uso de un RA apropiado, con la desventaja de un rendimiento menor por tratarse de un subsistema separado del SLEE, además que la complejidad del código necesario para acceder a estos datos seria mucho mayor, ya que la especificación ha definido una serie de mecanismos que facilitan el acceso a datos almacenados mediante perfiles, como puede ser observado en la figura 6.

Para la definición de perfiles existen los siguientes componentes:

- **Perfil:** Un perfil es un almacén de información comparable con un registro en una tabla de una base de datos. Todo Perfil corresponde a un Esquema.

- **Tabla de Perfiles:** Aquí son almacenados los perfiles que corresponden a un mismo Esquema.

- **Especificación de Perfil:** Es donde son especificados todos los parámetros necesarios para definir un Esquema de perfiles, así como también los mecanismos por medio de los cuales puede accederse a los perfiles por parte de los SBB u otras entidades.

Una característica importante de los perfiles es que puede haber muchas tablas de perfiles que correspondan a un mismo esquema.

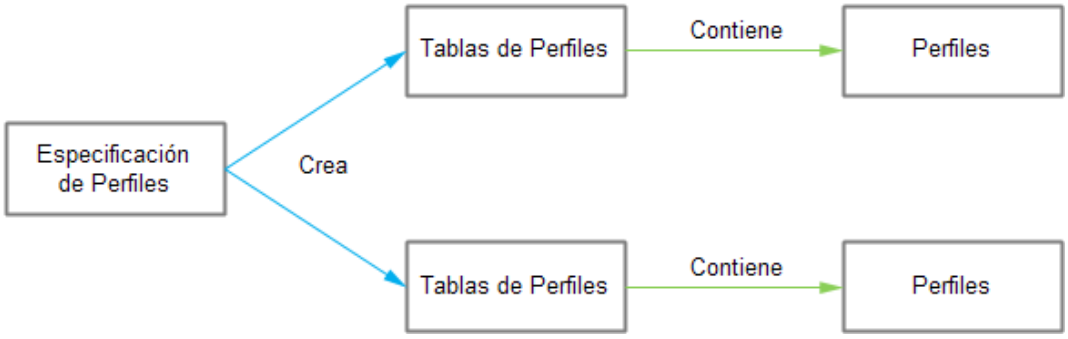

Figura 6. Perfiles.

1.5.5 Transacciones

Uno de los principales mecanismos que tiene el SLEE para ofrecer tolerancia a fallas son las transacciones. Una Transacción puede verse como una unidad de trabajo que engloba operaciones individuales consecutivas que al final ejecutan una tarea determinada, de tal modo que sí durante la ejecución de una de las operaciones miembro de la Transacción ocurre un error, toda la Transacción es abortada, los cambios efectuados hasta ese momento son revertidos, y ésta es considerada como no realizada, lo que asegura que tareas importantes nunca queden inconclusas o se pierdan. De este modo, cuando una Transacción termina exitosamente, es garantizado que todas las acciones derivadas de la ejecución de las operaciones miembro fueron efectivamente realizadas.

1.5.6 Contenedor Persistente Gestionado

Un CMP (Container Managed Persistence, Contenedor Persistente Gestionado) es un campo virtual que pertenece a una Entidad SBB y que sirve para almacenar información, como por ejemplo datos arbitrarios de servicio, interfaces hacia SBB hijo, etc.

Los campos CMP de una Entidad SBB deben ser definidos por el desarrollador.

1.6 Consideraciones para el Uso de JAIN SLEE

1.6.1 Ventajas

Dentro de las ventajas que caracterizan a la especificación JAIN SLEE están [4][6] [9]:

- **Estándar:** la especificación JAIN SLEE es un estándar de uso libre, por ello es de esperar que en el mercado pueda encuentrarse una amplia oferta de productos en un futuro próximo.

- **Independencia de la red:** facilita la integración de cualquier red al tener como base del desarrollo de aplicaciones, un modelo de componentes capaz de interactuar con cualquier protocolo, API o recurso de red, ello permite que muchos operadores puedan seguir obteniendo beneficios de sus anteriores infraestructuras a la vez que estas evolucionan.

- **Fiabilidad y robustez:** debido a que basa su funcionamiento en un modelo de transacciones, el sistema de tolerancia a fallas obtenido es muy eficiente.

- **Reutilización:** dado que los SBB son componentes reutilizables, mucho tiempo de desarrollo en nuevos servicios es ahorrado, a la vez, los costos disminuyen de manera importante.

- **Portabilidad de las aplicaciones:** debido al uso del lenguaje de programación Java y a la rigurosidad de la especificación JAIN SLEE, el desarrollo de aplicaciones y servicios es realizada una sola vez y se puede migrar a diferentes plataformas con ninguno o muy pocos cambios requeridos.

- **Fácil de operar y mantener:** el uso de JMX posibilita capacidades de gestión y monitoreo simples y potentes desde una gran variedad de

interfaces clientes, ya sean locales o distribuidas.

- **Creación de servicios convergentes:** gracias a la posibilidad de conexión del SLEE a través de RA con redes heredadas, redes modernas y en especial Internet, puede lograrse desarrollar servicios que combinan múltiples tecnologías de red y que son ofrecidos a una mayor cantidad de usuarios.

- **Orientado a eventos:** esto lo hace propicio para aplicaciones de telecomunicaciones, sin tener que realizar complejos estudios y adaptaciones a otros modelos de programación.

- **Fuerte manejo de invocaciones asincrónicas:** permitido por el comportamiento de los SBB, posibilita el manejo de múltiples eventos de forma independiente y eficiente.

- **Facilidades:** debido a que existen facilidades provistas por el SLEE, las aplicaciones o servicios no deben preocuparse por implementar funcionalidades genéricas.

1.6.2 Desventajas

Dentro de las desventajas que caracterizan a JAIN SLEE están [8]:

- **Una empinada curva de aprendizaje:** a pesar de que el lenguaje de programación es Java, este posee extensiones que corresponden al SLEE exclusivamente y que incorporan nuevos conceptos que probablemente son de difícil compresión inclusive para desarrolladores con experiencia.

- **Pocas implementaciones:** a la fecha, existen muy pocas implementaciones disponibles, donde solo una de ellas es libre y algunas de las propietarias solo están parcialmente basadas en la especificación y por ende no están certificadas.

1.7 Usos de JAIN SLEE

Desde su diseño, el estándar JAIN SLEE fue destinado para servir como entorno de ejecución para aplicaciones y servicios de telecomunicaciones en general. Sin embargo, estas aplicaciones de JAIN SLEE son muy similares entre sí en cuanto a sus arquitecturas y propósitos, por lo que pueden ser agrupados como sigue [12] [13] [14] [15]:

1.7.1 JAIN SLEE como Plataforma de Servicios

Este es el uso típico esperado para el estándar, actúa como entorno de ejecución para servicios de valor agregado a la vez que sirve como un ente integrador de diferentes tecnologías de red.

1.7.2 JAIN SLEE como Entorno de Ejecución para SDP

Una implementación de JAIN SLEE puede usarse dentro del núcleo de una SDP (Service Delivery Plaform, Plataforma para la Entrega de Servicios), específicamente como el SLEE, un ejemplo de ello es la SDP de Motorola, GAMA, la cual tiene en su interior a Rhino, la implementación de JAIN SLEE de la compañía Opencloud.

1.7.3 JAIN SLEE como Entorno Abierto de Ejecución para Terceros

Un tema critico para los operadores es como exponer la red hacia terceros. JAIN SLEE puede usarse como un entorno abierto de ejecución que aprovecha las características de la red a través de un RA conectado a una pasarela de Parlay/OSA, de modo que las capacidades de la red hacia redes inseguras sean expuestas de una manera confiable.

1.7.4 JAIN SLEE en Aplicaciones Especializadas

Debido sobre todo a las ventajas en cuanto al rendimiento y tolerancia a fallas, JAIN SLEE está detrás de implementaciones de productos especializados que son parte de otros sistemas más grandes. Un ejemplo es una implementación de un CSCF (Call Session Control Function, Función de Control de Sesiones de Llamada) de una red IMS (IP Multimedia Subsystem, Subsistema Multimedia IP) como un servicio desarrollado sobre JAIN SLEE. Este servidor es un componente fundamental de la arquitectura IMS y generalmente constituye un cuello de botella debido a su rol central, por lo que JAIN SLEE es una excelente alternativa para ese uso. De hecho existen algunas iniciativas que tienen como meta implementar todos los servidores de la capa de aplicación de la arquitectura de IMS como servicios JAIN SLEE.

CAPITULO 2. RELACIONES FUNCIONALES ENTRE APLICACIONES O SERVICIOS JAIN SLEE Y UN ENTORNO IMS

Dado que las aplicaciones y servicios en un entorno IMS pueden alojarse en un servidor de aplicaciones SIP, a continuación son descritas las interfaces que esta entidad puede manejar según las especificaciones definidas por el 3GPP, ellas son: TS 23218 [16], TS 23002 [17], TS 23228 [18], TS 24623 [19], TS 32260 [20], de las cuales algunas son utilizadas en el mecanismo de integración entre un AS (Application Server, Servidor de Aplicaciones) que implementa el estándar JAIN SLEE y el entorno IMS descrito posteriormente.

2.1 Entorno IMS completo para un AS SIP

Como está descrito en el Anexo A, existe una variedad de entidades IMS comunicadas de forma directa con el AS con el propósito de poder intercambiar información [21], ya sean parámetros como parte de la lógica para la ejecución de los servicios o señalización.

2.1.1 Interfaz ISC

El punto de referencia ISC (IMS Service Control, Interfaz de Control de Servicio IMS) está entre un S-CSCF (servidor encargado del control de sesiones SIP) y un AS, como es mostrado en la figura 7. Esta interfaz utiliza el protocolo SIP modificado por el 3GPP para IMS.

Los procedimientos sobre esta interfaz pueden dividirse en dos categorías principales, la primera corresponde al encaminamiento de solicitudes de inicio de sesión hacia un AS y la segunda es referida a las solicitudes de inicio de sesión inicializadas por el mismo AS, las cuales pueden ser realizadas en nombre de un usuario o en nombre de un servicio.

Después de llevar a cabo el registro de un usuario dentro de la red IMS, un S-CSCF es encargado de descargar los perfiles de servicios asociados a dicho usuario desde el HSS (Home Subscriber Server, Servidor Local de Datos de Abonado) correspondiente, el cual contiene la información que indica cuando ciertos tipos de mensajes SIP enviados por este usuario deben ser re-enviados hacia un AS a través de esta interfaz.

Figura 7.Interfaz de Control de Servicio IMS.

Los mensajes de solicitudes SIP que pueden intercambiarse entre un S-CSCF y un AS son [22] [26]:

- **SIP MESSAGE:** es usado para transferir mensajes instantáneos durante la sesión.

- **SIP INVITE:** es usado para dar inicio a una sesión, generando una transacción[7] SIP sobre la cual se lleva a cabo la negociación de los parámetros de la sesión.

- **SIP OPTIONS:** es usada para determinar qué métodos SIP soporta un Agente de Usuario SIP[8].

- **SIP UPDATE:** puede ser usado para modificar parámetros asociados con un mensaje SIP INVITE que fue enviado previamente.

- **SIP NOTIFY:** es usado para informar a un Agente de Usuario SIP sobre la ocurrencia de un evento específico que es de su interés.

- **SIP SUBSCRIBE:** es empleado por un Agente de Usuario SIP para suscribirse a los cambios de estado de otra entidad.

- **SIP INFO:** es encargado de transportar información relacionada a la sesión.

- **SIP ACK:** es usado para completar una transacción iniciada por el envío de un SIP INVITE.

- **SIP CANCEL:** su función consiste en detener el efecto de una solicitud enviada previamente.

- **SIP BYE:** Tiene como propósito dar por terminada una sesión SIP previamente establecida.

- **SIP REFER:** con su uso puede lograrse que mensajes que llegan a un Agente de Usuario SIP puedan ser reenviados de forma automática hacia

7 Transacción SIP es una sucesión lógica de mensajes SIP que es iniciada con una solicitud.
8 Agente de Usuario SIP es una entidad lógica que actúa como punto final en una red, y capaz de crear y recibir mensajes SIP.

un destino establecido por este mensaje.

- **SIP REGISTER:** es encargado de llevar a cabo el registro de un Agente de Usuario SIP en el entorno IMS.

- **SIP PUBLISH:** es usado por un Agente de Usuario SIP que desea publicar su información de estado.

2.1.2 Interfaz Sh

Esta interfaz es el punto de referencia ubicado entre un AS y un HSS, y hace la función de "puente" que le permite a un AS hacer consultas, actualizaciones y suscripciones de notificación sobre datos asociados a identidades almacenadas en un HSS mediante una aplicación especializada del protocolo Diameter.

Los procedimientos que pueden realizarse sobre la interfaz Sh pueden ser clasificados de la siguiente manera [22]:

- **Procedimientos de manipulación de datos:** consisten en la descarga de datos desde un HSS hacia un AS y la actualización de datos en el HSS por parte de un AS.

- **Procedimientos de suscripción y notificación:** permiten a un AS suscribirse a los cambios que sufran ciertos datos de una identidad específica dentro de un HSS, para que este último le envíe notificaciones cuando un cambio sobre dichos datos sea llevado a cabo.

La figura 8 y la tabla 1 resumen los mensajes disponibles para la interfaz Sh, denominados también como comandos.

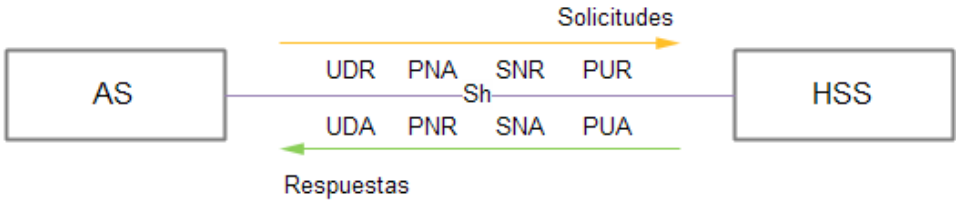

Figura 8. Mensajes de Diameter para la Interfaz Sh.

Pareja de Mensajes	Propósito	Abreviación	Fuente	Destino
User Data Request/Answer	Solicitar y entregar datos de una identidad específica.	UDR UDA	AS HSS	HSS AS
Profile Update Request/Answer	Actualizar datos en el HSS.	PUR PUA	AS HSS	HSS AS
Subscribe Notification Request/Answer	Realizar subscripciones o cancelaciones de subscripciones a cambios en los datos de una identidad.	SNR SNA	AS HSS	HSS AS
Push Notification Request/Answer	Enviar notificaciones de cambios sobre datos a un AS.	PNR PNA	HSS AS	AS HSS

Tabla 1. Comandos Diameter para la Interfaz Sh.

Los datos accesibles a través de la interfaz Sh están definidos en la especificación 3GPP TS 29.328 [23], siendo algunos de ellos: datos arbitrarios de servicio, la identidad pública IMS, el estado de un usuario IMS, el nombre del S-CSCF correspondiente a la identidad, los criterios de filtro inicial, la información de localización, etc., sobre los cuales pueden ejecutarse los comandos. Sin embargo, no todas las operaciones pueden ser realizadas sobre estos datos debido a algunas restricciones impuestas en las especificaciones de IMS.

2.1.3 Interfaz Dh

Esta interfaz está entre un SLF (Subscriber Location Function, Función de Localización de Suscriptor) y un AS como puede observarse en la figura 9. Es usada para obtener la dirección del HSS que almacena la información de un perfil de usuario en particular y/o información necesaria para sus servicios. Esta interfaz solo está presente en las redes IMS que cuentan con más de un HSS. Es importante destacar que sobre ella se trabaja de forma muy similar a como es hecho sobre la interfaz Sh.

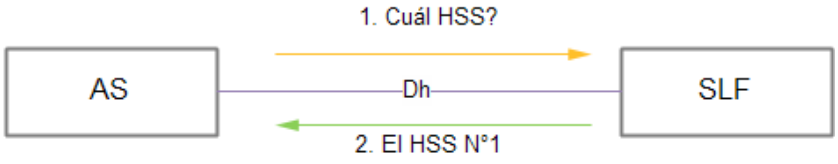

Figura 9. SLF y un AS.

2.1.4 Interfaz Ma

Este punto de referencia está ubicado entre un I-CSCF (Interrogating-CSCF, CSCF Interrogador) y un AS, sus usos son:

- Iniciar una sesión en nombre de un usuario o de una Identidad de Servicio Pública en el caso de que el AS no tenga conocimiento del S-CSCF asignado a ese usuario o a esa Identidad de Servicio Pública, tal como es mostrado en la figura 10.

- Enviar solicitudes SIP destinadas a una Identidad de Servicio Pública de un servicio de forma directa hacia un AS que lo hospeda, tal como es mostrado en la figura 11.

El protocolo usado sobre esta interfaz es SIP, pueden usarse extensiones pero no es recomendado [18].

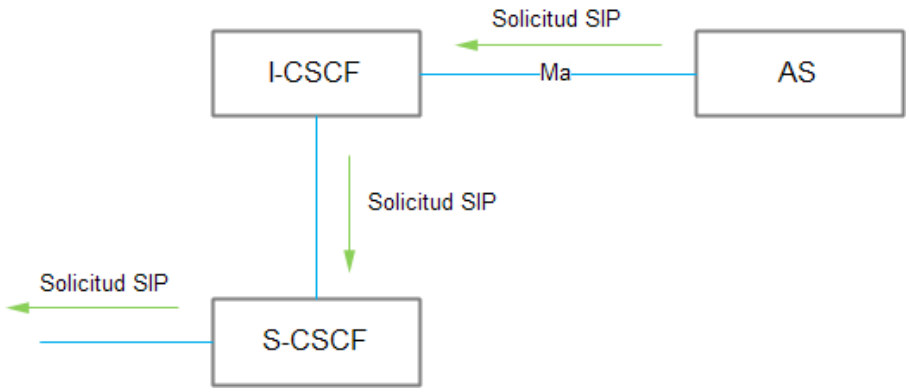

Figura 10. El AS inicia una sesión desconociendo el S-CSCF.

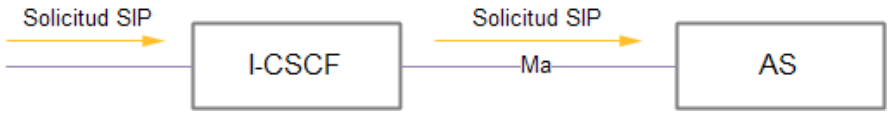

Figura 11. I-CSCF enviando solicitudes SIP al AS.

2.1.5 Interfaz Cr

Un AS también puede interactuar con un MRFC (Media Resource Function Controller, Función de Control de Recursos Multimedia) a través de la interfaz Cr

con el fin de ejercer control sobre un MRFP (Media Resource Function Processor, Función de Procesamiento de Recursos Multimedia) y así poder realizar control multimedia relacionado a la asignación de recursos necesarios para las llamadas, como puede verse en la figura 12. El protocolo usado sobre esta interfaz es SIP [24].

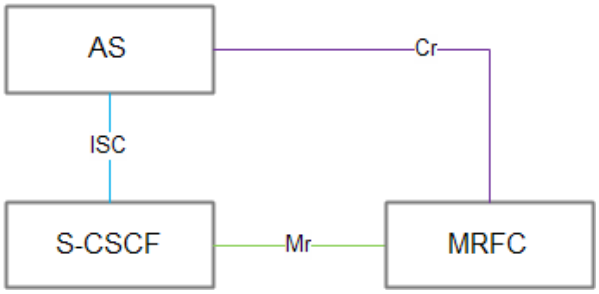

Figura 12. Punto de referencia entre un AS y un MRFC.

2.1.6 Interfaz Mr'

Esta interfaz existe también entre un MRFC y un AS, tiene como fin el intercambio de mensajes de control de sesión sin pasar a través del S-CSCF, ver figura 13. El protocolo usado sobre este punto de referencia está basado en SIP.

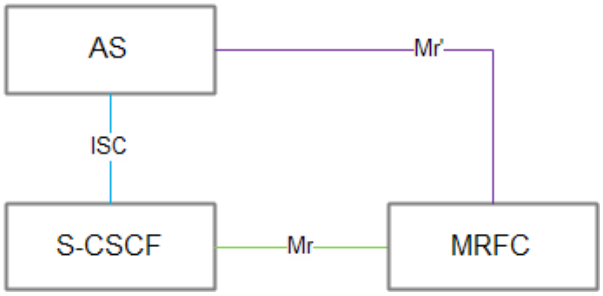

Figura 13. Unión entre un AS y un MRFC.

2.1.7 Interfaz Ut

Este punto de referencia tiene lugar entre un terminal de usuario y un AS, permitiendo a los usuarios gestionar de forma segura sus servicios de red relacionados con la información alojada en un AS, ver figura 14. Los usuarios pueden utilizar esta interfaz para personalizar sus servicios. Algunos ejemplos de servicios que utilizan esta interfaz son los servicios de presencia y los servicios de *Push to Talk* sobre celular.

Los protocolos que pueden ser utilizados sobre esta interfaz son: XCAP (XML Configuration Access Protocol, Protocolo de Acceso a Configuración XML), y HTTP (Hypertext Transfer Protocol, Protocolo de Transferencia de Hipertexto).

Figura 14. Punto de referencia entre un AS y el terminal de usuario.

2.1.8 Interfaz Rc

Esta interfaz está entre un MRB (Media Resource Broker, Mediador de Recursos Multimedia) y un AS como es mostrado en la figura 15, permitiéndole a este último solicitar recursos multimedia que necesiten ser asignados a una llamada. La interfaz utiliza el protocolo SIP, pero los detalles concretos no han sido especificados todavía [16].

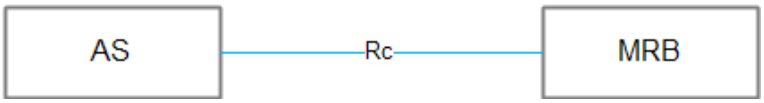

Figura 15. Punto de referencia Rc entre un AS y un MRB.

2.1.9 Interfaz Rf

Este punto de referencia está entre un AS y un CCF (Charging Collection Function, Función de Recolección de Información de Cobro) ver figura 16. Es usado con el fin de enviar información de cobro *offline*, modo que consiste básicamente en coleccionar información de la sesión sin alterar en tiempo real los sistemas de cobro [18] [25].

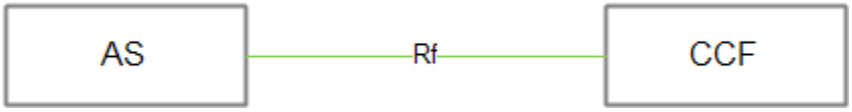

Figura 16. Punto de referencia Rf entre un AS y un CCF.

2.1.10 Interfaz Ro

Este punto de referencia está entre un AS y un ECF (Event Collection Function, Función de Recolección de Información de Eventos), ver figura 17, es usado con el fin de poder enviar información de cobro *online*, modo que consiste básicamente en llevar la información en tiempo real hacia los sistemas de cobro y de monitoreo de los servicios con el fin de tener presente si el usuario puede consumir o no el servicio según el saldo que exista en su cuenta [18] [25].

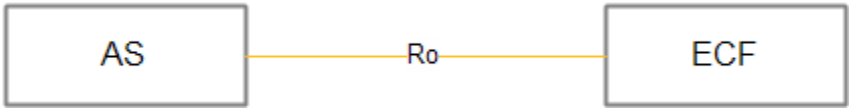

Figura 17. Punto de referencia Ro entre un AS y un ECF.

2.2 Interfaces JAIN SLEE para la Integración con IMS

Según las anteriores descripciones, un AS que este en una red IMS debería poder usar varias de las interfaces anteriormente descritas de modo que las capacidades de dicha red sean aprovechadas adecuadamente, para lo cual debería utilizar los protocolos mencionados.

Es aquí donde JAIN SLEE muestra uno de sus principales beneficios, el cual consiste en la independencia de protocolos gracias a su esquema basado en RA para las comunicaciones con entidades externas, en este caso entidades IMS. Entonces, para que un servidor JAIN SLEE en un entorno IMS pueda utilizar estas interfaces solo necesita contar con los RA correspondientes. Sin embargo, el uso de los protocolos no es el típico en la mayoría de los casos y el uso de extensiones es común, por lo que los servicios que usen estas interfaces deben conocer las particularidades de cada caso y aplicarlas de tal modo que no existan conflictos a bajo nivel.

Es por esto, que la integración de un servicio existente previamente desarrollado en JAIN SLEE en una red IMS no es transparente y requiere de una adecuación en la lógica de comunicaciones de dicho servicio.

2.2.1 Adaptador de Recurso SIP

Este RA proporciona una interfaz responsable de enviar y recibir mensajes SIP entre Agentes de Usuario SIP y el SLEE. Para ello, realiza la función de convertir los mensajes SIP recibidos en eventos para que puedan ser procesados por los servicios, y convertir eventos SIP en mensajes SIP para enviarlos nuevamente

hacia la red.

Este RA debe ser compatible con el RFC 3261 [26], con el fin de asegurar la compatibilidad con clientes SIP ordinarios, y adicionalmente con los RFC 3455 [27], RFC 3325 [28], RFC 3313 [29], RFC 3327 [30], RFC 3608 [31] que definen extensiones para IMS, de manera que pueda asegurarse compatibilidad con clientes IMS.

El uso de este RA por parte de los servicios desplegados en el SLEE asegura que estos puedan utilizar todas las interfaces IMS que funcionan con el protocolo SIP. Sin embargo, para poder usar el RA, los servicios deben conocer a fondo el protocolo SIP y sus extensiones para IMS debido a que el RA provee acceso a bajo nivel, es decir, a través del RA puede tenerse acceso a cabeceras SIP, desde las más básicas hasta las menos usuales.

2.2.2 Adaptador de Recurso Diameter

Este RA captura los mensajes del protocolo Diameter en cualquiera de sus aplicaciones y los convierte en un evento que es entregado en una Actividad, por lo tanto, este RA puede ser usado por cualquier servicio alojado en el SLEE que desee usar las interfaces que emplea este protocolo, siempre y cuando sea utilizado de la manera descrita en la aplicación Diameter definida para dicha interfaz y sean utilizados los correspondientes AVP(Attribute Value Pair, Parejas Atributo-Valor) de manera rigurosa, lo cual hace que la complejidad del servicio sea incrementada drásticamente dado que el protocolo Diameter no es un protocolo simple [32].

2.3 Modos de Operación de un Servidor de Aplicaciones con un S-CSCF

Para poder definir mejor la conexión entre un AS y un entorno IMS, la especificación 3GPP 23218 [16] ha establecido los diferentes modos de operación entre un AS y su entidad principal, el S-CSCF. Estos corresponden a cinco modos básicos de operación para procesar solicitudes SIP.

Por lo tanto, los servicios alojados en un AS pueden ser construidos usando combinaciones entre estos modos.

Los modos de operación son los siguientes:

2.3.1 Servidor de Aplicaciones actuando como Agente de Usuario Cliente

En este modo de operación el mensaje SIP de entrada es dirigido por el S-CSCF

al AS, el cual a través del RA SIP actúa como un Agente de Usuario Cliente, ver figura 18.

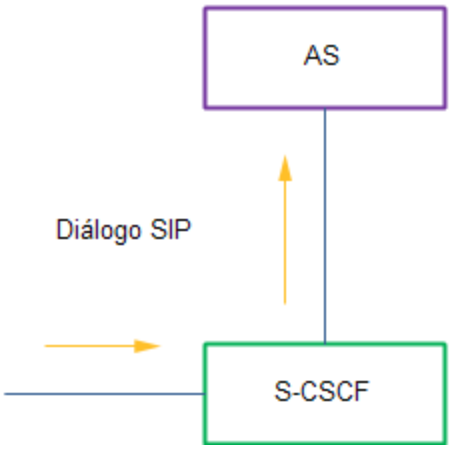

Figura 18. AS como Agente de Usuario Cliente.

2.3.2 Servidor de Aplicaciones actuando como Agente de Usuario Servidor

En este modo de operación el AS actúa como el cliente que inicia la llamada, generando solicitudes SIP que son enviadas a través del S-CSCF, que a su vez las envía hacia su destino, ver figura 19.

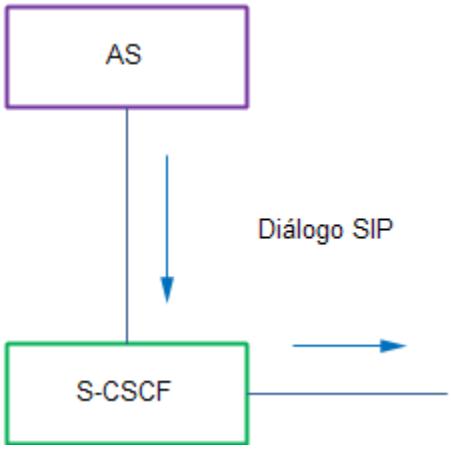

Figura 19. AS como Agente de Usuario Servidor.

2.3.3 Servidor de Aplicaciones actuando como un Proxy SIP

En este modo de operación, las solicitudes SIP de entrada son enviadas por el S-CSCF al AS, el cual redirige las solicitudes de vuelta hacia el S-CSCF después de realizar algún proceso sobre ellas, que puede o no modificar cabeceras SIP, para que así finalmente sean enviadas a su verdadero destino. Este modo de funcionamiento es usado generalmente cuando el servicio tiene como finalidad la supervisión, monitoreo o gestión de las sesiones de servicios de más bajo nivel.

Un diagrama de este modo de funcionamiento puede apreciarse en la figura 20.

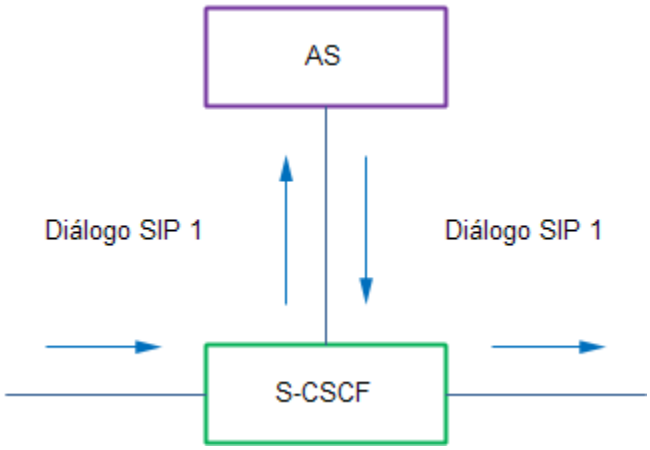

Figura 20. AS actuando como Proxy SIP.

2.3.4 Servidor de Aplicaciones en modo B2BUA

En el modo de operación B2BUA (Back to Back User Agent, Agente de Usuario Espalda contra Espalda), las solicitudes SIP entrantes son dirigidas por el S-CSCF hacia un AS, el cual genera una nueva solicitud SIP lógicamente relacionada con la anterior y la envía al S-CSCF para que esta sea encaminada hacia su destino, tal como puede apreciarse en la Figura 21.

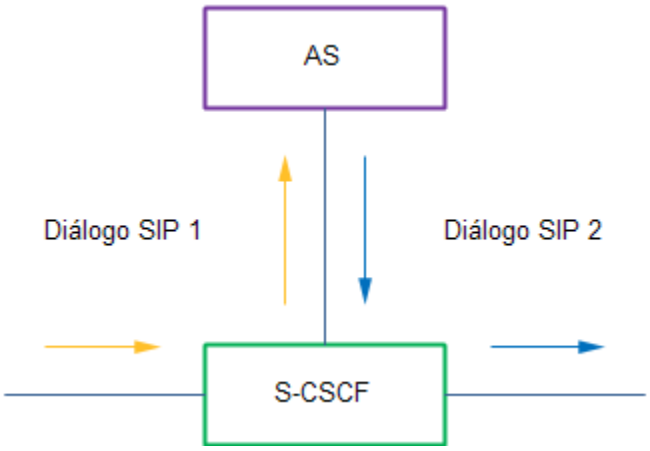

Figura 21. AS actuando como un B2BUA.

2.3.5 Servidor de Aplicaciones en modo B2BUA Activo

En este modo de operación el AS inicia dos solicitudes con dos diferentes diálogos. Estas solicitudes son dirigidas al S-CSCF, el cual las dirige hacia sus destinos, tal como es mostrado en la figura 22.

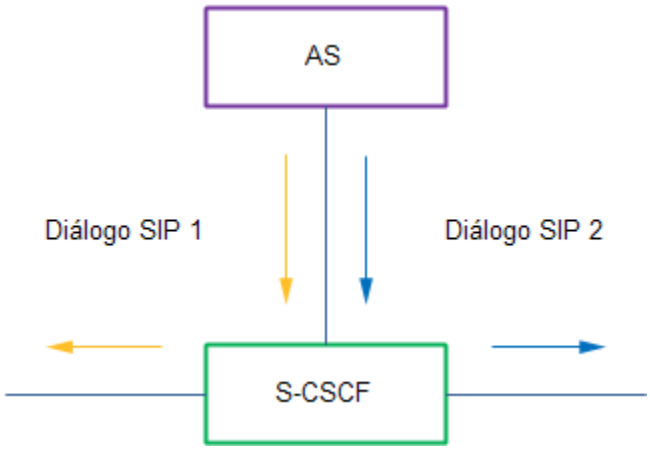

Figura 22. AS actuando como un B2BUA Activo.

2.3.6 Servidor de Aplicaciones no involucrado

En este modo de operación el AS nunca es involucrado en la señalización SIP o tiene configurado no ser usado durante un largo periodo de tiempo. La solicitud SIP de entrada es recibida y enviada por el S-CSCF.

2.4 Conclusión del Capítulo

Hasta el momento, una base conceptual acerca de las dos temáticas principales de este trabajo, que son JAIN SLEE e IMS, ha sido creada. En ella aspectos como arquitecturas, componentes, conceptos, interfaces, conexiones e interacciones han sido destacados; ello tenía como finalidad establecer cada una de las relaciones funcionales necesarias entre una aplicación o servicio JAIN SLEE y un entorno IMS.

De este modo, se ha dado desarrollo y cumplimiento al primer objetivo que se propuso para poder llegar a la definición de los componentes reusables necesarios que conformarían el mecanismo de integración propuesto, el cual será explicado en el siguiente capítulo.

CAPITULO 3. INTEGRACION DE JAIN SLEE EN UN ENTORNO IMS

Este capítulo aborda todo el proceso para la definición de los componentes reusables por aplicaciones o servicios JAIN SLEE que son prestados sobre un entorno IMS. Por lo tanto, primero son destacados las falencias del modo de integración básico existente, para posteriormente definir unos criterios que permitan dar respuesta a las necesidades mostradas y que conduzcan hacia la identificación de unas arquitecturas candidatas, las cuales luego de ser analizadas darán paso a la selección de la arquitectura del mecanismo de integración. Posteriormente, la arquitectura general del mecanismo de integración dentro de un entorno IMS es definida, junto con la descripción de los servicios de las interfaces IMS que lo componen y su funcionamiento.

3.1 Modo de Integración Básico entre JAIN SLEE y un Entorno IMS

La integración básica entre JAIN SLEE y un entorno IMS es lograda haciendo uso de RA que correspondan a cada uno de los protocolos usados por las interfaces IMS que comunican de forma directa un AS con cualquier otra entidad. Por lo tanto, es necesario contar con RA para el protocolo SIP, con un RA para el protocolo Diameter y con otros dos RA para los protocolos XCAP y HTTP [33], esto según la especificación actual de IMS [18].

Observando la figura 23 puede llegarse a pensar que la integración de JAIN SLEE y un entorno IMS es demasiado simple, pero hacerlo de esta forma lleva consigo grandes dificultades para una creación ágil y eficiente de servicios de valor agregado. Entre estos inconvenientes están las particularidades de los protocolos utilizados, además de los problemas propios derivados del hecho de que IMS aún no está completamente definido, por lo que es común encontrarse con adaptaciones hechas por los operadores para solventar necesidades puntuales de su red.

Figura 23. Arquitectura básica de integración.

Teniendo esto en cuenta, es evidente la necesidad de un operador de sacar ventaja a la hora de prestar servicios de valor agregado mediante la reducción de los tiempos de desarrollo asociados al aprendizaje e implementación de los protocolos que usan las interfaces IMS, ya que por una parte, es necesaria la realización de este proceso cada vez que desee prestar un nuevo servicio, lo que hace que estos sean mucho más costosos y propensos a errores, además que estaría desaprovechando todas las facilidades para la reutilización de componentes que ofrece JAIN SLEE.

Por otra parte, este modo de integración tiene el inconveniente que no oculta o separa las entidades de la red IMS de los servicios de valor agregado, algunos de los cuales pueden ser desarrollados por terceros, por lo que el uso y acceso a estas entidades por parte de los servicios no puede ser supervisado o controlado con facilidad, lo que podría derivar en ciertas situaciones de seguridad para el operador.

Es por esto, que aunque JAIN SLEE tiene un gran potencial para cumplir con los requerimientos de servicios de telecomunicaciones, su uso no está masificado hasta ahora de gran manera, ya que el manejo de los protocolos a tal detalle es una barrera de entrada para su utilización por parte de otros actores, como por

ejemplo proveedores de servicios del mundo de las tecnologías de la información.

Es así como la necesitad de enriquecer o mejorar la integración realizada entre JAIN SLEE y un entorno IMS es identificada. Debe ser mediante la utilización de componentes que realicen el ocultamiento del manejo de estas interfaces y sus respectivos protocolos, de tal forma que estas dificultades sean reducidas lo suficiente como para permitir una creación de servicios mucho más rápida, segura y eficiente.

3.2 Consideraciones para la Selección de la Arquitectura

Para la selección de la arquitectura del mecanismo de integración son tenidos en cuenta varios aspectos que permiten evaluar las diferentes arquitecturas candidatas con el fin de poder seleccionar la que mejor responda a las necesidades que un operador de telecomunicaciones podría tener, por ejemplo en rendimiento, seguridad, facilidades, etc [34].

- **Grado de reutilización:** este criterio indica que tan fácil sería para un servicio de valor agregado reutilizar las funcionalidades provistas por el mecanismo de una manera transparente, de tal forma que puedan utilizarse capacidades de la red IMS sin la necesidad de conocer las interfaces o protocolos necesarios para ello.

- **Rendimiento:** la implementación del mecanismo tiene que asegurar que el impacto en el rendimiento del servicio de valor agregado que lo usa sea mínimo.

- **Separación de lógicas de servicio:** el mecanismo debe ocultar en el mayor grado posible toda la lógica de acceso a las entidades IMS, así como también evitar que los servicios de valor agregado necesiten permisos especiales dentro del SLEE.

- **Tiempo de desarrollo asociado:** este aspecto esta referido a que tan fácil es incorporar dentro de un servicio de valor agregado, los llamados necesarios para usar las funcionalidades provistas por el mecanismo, de modo que no requiera mucho tiempo adicional de desarrollo.

- **Favorabilidad para el Operador:** la mejor arquitectura es aquella que le permita desplegar servicios lo más rápido posible, que requiera poco o nulo tiempo de entrenamiento en protocolos de comunicaciones para los desarrolladores, que ofrezca un rendimiento óptimo y sobre todo que no represente ninguna clase de riesgo para el correcto funcionamiento de la red a la vez que no exponga datos sensibles a terceros.

3.3 Arquitecturas Candidatas

De acuerdo al lenguaje de programación Java y a la especificación de JAIN SLEE, las arquitecturas que pueden implementarse dentro del contenedor de SBB son las siguientes [5]:

- **Arquitectura basada en herencia:** en esta arquitectura todas las funcionalidades comunes son implementadas en clases Java normales o abstractas para luego ser extendidas por otras clases más especializadas.

- **Arquitectura basada relaciones SBB padre-hijo:** las funcionalidades comunes son implementadas en SBB padres que deben hacer llamados a SBB hijos mediante interfaces locales.

- **Arquitectura basada en servicios independientes:** las funcionalidades comunes son implementadas en servicios independientes que pueden ser consumidos por otros servicios.

3.3.1 Arquitectura Basada en Herencia

La herencia permite que servicios de valor agregado puedan, muy fácil y rápidamente, contar con funcionalidades para el uso de las capacidades de la red IMS sin requerir prácticamente cambio alguno en el código Java fuente. Sin embargo, muchos cambios si son necesarios en los Documentos Descriptores de Despliegue[9] del servicio dentro del SLEE, ya que los SBB de los que hereda hacen uso de los protocolos de bajo nivel asociados a cada punto de referencia que existe entre el AS y las demás entidades, por lo que los eventos asociados a los mensajes envíados y recibidos de estos protocolos deben ser declarados en estos documentos. Además, en estos documentos también están indicadas las referencias a los RA correspondientes los cuales en la mayoría de los casos están asociados a elementos físicos de la red, por lo que la ganancia en ocultación y reutilización de código no está reflejada en estos descriptores, obligando así, a que los desarrolladores de servicios de valor agregado deban tener un mediano conocimiento sobre los protocolos de comunicaciones y elementos de red subyacentes, a la vez que necesitan tener una buena comprensión del comportamiento de los métodos provistos por las superclases.

Otra desventaja evidenciada sobre este punto, es que los SBB no solo heredan las funcionalidades que quieren usar sino también todas las demás, por lo cual los desarrolladores de servicios de valor agregado deben tener esto muy en cuenta para no tener ningún problema por lógicas que sean ejecutadas automáticamente en las superclases.

9 Documentos Descriptores de Despliegue corresponden a documentos XML que contienen información de configuración y de despliegue.

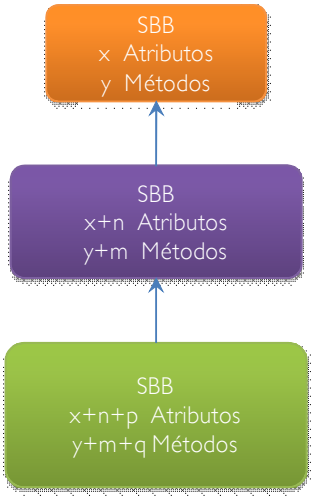

Figura 24. Ejemplo de herencia entre SBB.

A manera de ejemplo, la figura 24 muestra cómo para el caso de la interfaz Sh, una arquitectura basada en herencia tiene primero que todo una clase SBB que debe implementar la interfaz Java SBB definida por la especificación JAIN SLEE, la cual posee un número determinado de atributos y métodos. De este SBB hereda el SBB encargado de las comunicaciones Diameter sobre la interfaz Sh, agregando a la vez sus propios atributos y métodos, de tal forma que este SBB es más complejo que el anterior. Por último, el SBB del servicio de valor agregado hereda del SBB de la interfaz Sh y agrega nuevamente atributos y métodos propios, por lo que es aún más complejo y costoso de administrar computacionalmente para el SLEE que lo contiene.

En cuanto al rendimiento derivado del uso de la herencia en SBB, este es óptimo en cuanto a los tiempos de respuesta (siempre y cuando la clase abstracta resultante después de la herencia no sea demasiado voluminosa como para que sea computacionalmente costosa la serialización y des-serialización de sus instancias), pues no se hacen llamados a otras entidades sino que todas las funcionalidades están ya disponibles en el mismo SBB. Sin embargo, el hecho de que estos SBB tengan dentro de sí toda la lógica de comunicaciones para cada uno de los puntos de referencia de la red IMS al igual que la lógica de servicio, hace que estos sean objetos más grandes y por ende sean más difíciles de administrar para el SLEE, lo que puede hacer que la piscina[10] de objetos SBB que el SLEE pueda manejar simultáneamente sea menor y/o que los tiempos de respuesta asociados a la administración del ciclo de vida de las entidades SBB sea mayor.

10 Una piscina de entidades u objetos es una cantidad determinada de objetos Java previamente creados que se encuentran disponibles para ser usados de modo que se ahorra el tiempo de creación de los mismos.

Respecto a la separación de las lógicas de comunicaciones y la lógica de servicio, la herencia no ofrece un mecanismo muy apropiado para lograr este objetivo, pues las clases Java que heredan de las superclases pueden acceder a la definición de los métodos implementados en dichas superclases, y según el modificador de acceso que estos métodos tengan, inclusive es posible omitir las implementaciones provistas y reemplazarlas por otras implementadas en la clase que hereda. En el caso concreto de las funcionalidades para el uso de las interfaces IMS de comunicación con el AS, esta situación es potencialmente riesgosa sino es tenido en cuenta las precauciones adecuadas, pues un servicio podría omitir las implementaciones provistas. Como ejemplo puede citarse el caso de la actualización de datos de usuario en el HSS y hacer modificaciones erróneas, no permitidas o inclusive malintencionadas.

Debido a esto, es muy posible que un operador no llegue a aprobar el uso de esta práctica, pues el código necesario para acceder y consumir funcionalidades, prestadas sobre los puntos de referencia entre el AS y las demás entidades de la red, contiene información muy crítica y seguramente confidencial, por ejemplo acerca del HSS que es un componente vital del que depende el correcto funcionamiento de toda o una buena parte de la red del operador y que podría dejar sin servicio a miles e incluso a millones de usuarios.

3.3.2 Arquitectura Basada en Relaciones SBB Padre-Hijo

A pesar de que JAIN SLEE define un entorno de ejecución basado en eventos para que sea asincrónico, también define unas interfaces sincrónicas que sirven como un mecanismo para mejorar los tiempos de respuesta en los llamados entre SBB, pero sobre todo como una forma alternativa de comunicación entre SBB que no usa el Enrutador de Eventos del SLEE, para de esta forma ayudar a evitar que este sea congestionado con eventos que no provienen de los RA.

Desde el punto de vista de la reutilización de componentes, este método es muy similar al anterior en cuanto a la facilidad para su implementación, pues los llamados sincrónicos entre diferentes SBB son hechos de manera idéntica. La diferencia radica en que el acceso a las funcionalidades se hace a través de instancias de otros SBB y no a través de la misma instancia que invoca la funcionalidad. Sin embargo, de la misma forma que en la aproximación mediante herencia, el código al igual que los documentos descriptores de despliegue de estos SBB están expuestos totalmente a los desarrolladores de los servicios que los usan, y es posible modificar sus instrucciones.

Además, cuando esta técnica es usada y con el fin de implementar servicios que sean robustos, debe conocerse muy bien todos los detalles del funcionamiento de todos los SBB implicados en dichas relaciones, tal como lo sugiere la especificación de JAIN SLEE en la sección 3.1.2 [5].

Figura 25. Ejemplo de una relación padre-hijo entre SBB.

En la figura 25 puede apreciarse un ejemplo de esta arquitectura, donde un SBB encargado de la lógica de un servicio de valor agregado realiza invocaciones sobre una instancia de un SBB encargado de la interfaz Sh para la comunicación con un HSS, el cual al finalizar el procesamiento respectivo, invoca al SBB de la lógica para de esta manera informarle acerca de los resultados de la operación.

El rendimiento de esta solución también es bueno en cuanto a los tiempos de respuesta, ya que aunque los llamados no son hechos sobre la misma instancia que invoca las funcionalidades, son hechos sobre instancias que residen en la misma JVM[11] y sin pasar por el Enrutador de Eventos. Así mismo, otro beneficio de esta arquitectura, es que las instancias de los SBB de los servicios son menos voluminosas, pues ya no incluyen lógicas adicionales a las de negocio, por lo que la serialización y des-serialización de las entidades SBB es más rápida. Igualmente, la JVM puede invocar al recolector de basura para que elimine las instancias de los SBB que proveen las funcionalidades de comunicación con la red IMS cuando estas ya no son necesarias y por consiguiente liberar recursos del sistema.

Por otra parte, a pesar de que esta arquitectura separa bien las lógicas de negocio y de comunicaciones IMS en diferentes SBB, esta separación refiere al hecho de que los códigos fuentes están separados en diferentes clases Java y a que sus correspondientes instancias de SBB funcionan por separado aunque dentro de la misma JVM. Sin embargo, durante la etapa de implementación de los servicios, los desarrolladores tienen acceso irrestricto sobre el código fuente de los SBB que proveen las prestaciones de comunicaciones IMS y de hecho deben estudiarlo constantemente para así poder implementar los SBB propios de manera adecuada. Es decir, la separación de lógicas provista por esta arquitectura no provee un aislamiento suficiente para permitir que los desarrolladores de servicios de valor agregado puedan crear servicios sin tener que conocer y entender todos los detalles de bajo nivel asociados a las comunicaciones entre el AS y las correspondientes entidades IMS, a la vez que los operadores no podrían proteger el código sensible que accede a su infraestructura crítica de red.

11 Máquina Virtual de Java (JVM, Java Virtual Machine).

Teniendo en cuenta los anteriores argumentos, lo más probable es que un operador siga considerando de mayor importancia garantizar la seguridad de su infraestructura de red mediante la no divulgación de información crítica, por lo que a pesar de las ventajas que ofrece esta arquitectura desde el punto de vista del rendimiento, sus demás falencias hacen que no sea factible su aprobación para implementación. Además, si ésta arquitectura donde cada servicio tiene toda la lógica de comunicaciones con las entidades IMS llegara a ser implementada, el operador de telecomunicaciones debería auditar este código si quiere asegurarse de que haga uso correcto de la red, lo que indiscutiblemente retrasaría enormemente el proceso de despliegue de nuevos servicios a la vez que demandaría recursos adicionales.

3.3.3 Arquitectura Basada en Servicios Independientes

Tal como su nombre lo indica, en esta arquitectura las funcionalidades de acceso a las entidades IMS que tienen interfaces hacia el AS están disponibles como servicios independientes de fácil consumo alojados en el SLEE y que ofrecen eventos personalizados como medio de comunicación con otros servicios.

De este modo, para que un servicio de valor agregado pueda acceder a estas funcionalidades, sólo tiene que enviar eventos del tipo adecuado hacia el Enrutador de Eventos, y este automáticamente esta encargado de instanciar un SBB raíz del servicio independiente apropiado, el cual realiza el procesamiento respectivo para posteriormente iniciar otro evento informando acerca del resultado de la operación solicitada. Aquí puede apreciarse la gran ventaja de esta arquitectura, que consiste en el hecho de que el servicio de valor agregado no tiene acceso a los detalles acerca de la implementación de las funcionalidades y tampoco lo necesita, por lo que el código de este servicio en efecto solo consiste en la lógica de negocio que le corresponde, además de que sus documentos descriptores de despliegue tampoco tienen ninguna referencia a los RA ni a los eventos de los protocolos de comunicaciones empleados.

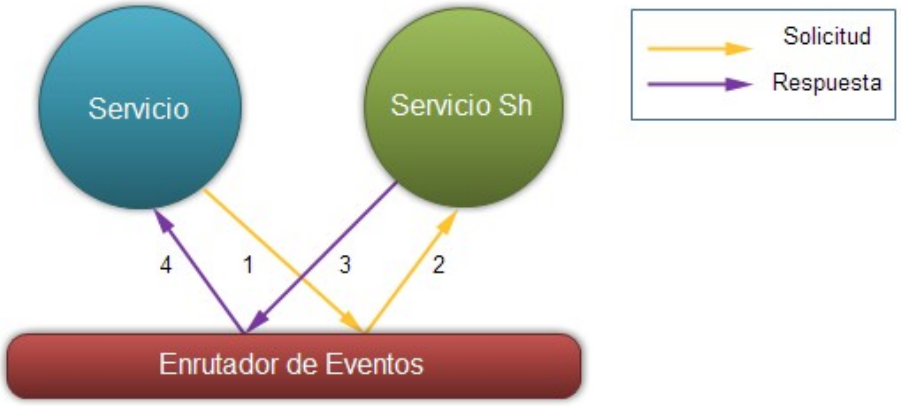

Figura 26. Ejemplo de llamados asincrónicos entre SBB.

En el ejemplo de la figura 26, puede apreciarse como la lógica de un servicio y la lógica de acceso a un servidor HSS están totalmente separadas gracias al manejo de eventos personalizados y la intermediación del Enrutador de Eventos.

Desde el punto de vista del rendimiento, esta arquitectura tiene una desventaja respecto a las dos anteriores, la cual consiste en la necesidad de que todos los eventos personalizados enviados por los servicios de valor agregado, pasen por el Enrutador de Eventos del SLEE, para que este pueda crear una instancia de SBB del servicio independiente de comunicaciones para el procesamiento, lo que implica un retardo asociado al tiempo que le toma al Enrutador de Eventos realizar este proceso. Además, la cantidad de eventos que este componente del SLEE entraría a manejar podría incrementarse notablemente, pues aparte de los eventos que generan los RA, estarían los de los servicios de valor agregado alojados en el SLEE, lo que finalmente hace que esta arquitectura demande más recursos hardware.

Por otra parte, al igual que en la arquitectura basada en relaciones padre-hijo entre SBB, el uso de memoria RAM en esta arquitectura es más eficiente que en la arquitectura basada en herencia, pues las funcionalidades están distribuidas en objetos más pequeños que se usan por periodos más cortos de tiempo y que por ende pueden ser removidos por el recolector de basura de la JVM una vez que estos han dejado de usarse, por lo que la cantidad de memoria RAM disponible en el tiempo es en promedio mayor.

La arquitectura basada en servicios independientes se diferencia de las otras dos opciones en que es la única que efectivamente permite una separación total entre las lógicas de negocio y de comunicaciones, es decir, un servicio de valor agregado alojado en un SLEE que desee consumir funcionalidades provistas por entidades IMS conectadas al AS, no tendría el más mínimo conocimiento acerca de los protocolos necesarios para tal fin, además de que tampoco requeriría saber

los detalles acerca de estas entidades, ni tener acceso a los RA correspondientes, por lo que los documentos descriptores de despliegue de estos servicios solo contendrían la configuración propia.

Entonces, desde el punto de vista de la facilidad de reutilización de componentes que ofrece esta arquitectura, ésta es la opción más adecuada, ya que permite el desarrollo de servicios de valor agregado que en efecto solo consisten de las lógicas y los documentos descriptores de despliegue propios del servicio que quiere prestarse, evitando así dedicar recursos en lógicas dependientes de la especificación de IMS y de protocolos de bajo nivel, lo que hace que el proceso de creación y despliegue de nuevos servicios sea mucho más ágil.

Muy relacionado con el anterior aspecto, es que el tiempo de desarrollo adicional necesario para utilizar estas funcionalidades es mínimo, ya que por una parte el código fuente necesario para el envío y recepción de eventos es muy pequeño, y por otro lado, no es necesario configuraciones ni permisos de seguridad adicionales en los documentos descriptores de despliegue, por lo que este tiempo es el menor de las 3 alternativas de arquitectura.

Por otra parte, para un operador esta arquitectura también ofrece una característica muy importante que las demás arquitecturas no tienen, que es el hecho de que el código crítico que accede a las entidades IMS está bajo su total dominio y no es necesario que sea conocido por los desarrolladores de servicios. De esta manera, información vital acerca de la infraestructura de red es protegida.

Finalmente, estos servicios de acceso a las funcionalidades de la red, deberían ser gestionados por personal de confianza dentro del operador, de modo que puedan ser configurados como en el caso del establecimiento de límites o restricciones independientes por servicio, para limitar la carga sobre un HSS y mantenerlo en operación en rangos seguros de volumen de transacciones.

3.4 Selección de la Arquitectura del Mecanismo de Integración

Como puede apreciarse en la tabla 2, la arquitectura basada en servicios independientes presenta la mayor cantidad de beneficios, con excepción del rendimiento ya que en este caso la arquitectura basada en relaciones padre-hijo entre SBB es la mejor opción. Sin embargo, dado los resultados obtenidos en las pruebas realizadas sobre el prototipo implementado contenidas en el capítulo 5 de este trabajo, demuestra que el rendimiento es muy bueno y de llegar a ser necesario, la capacidad del sistema puede ser fácilmente escalada gracias a las facilidades de *clustering*[12] que ofrece JAIN SLEE.

12 Clustering se refiere a la técnica que permite usar un conjunto de computadoras como si fueran una sola de mayores prestaciones.

	Herencia	Relaciones padre-hijo	Servicios Independientes
Reutilización	medio	medio	alto
Separación de lógicas	bajo	medio	alto
Rendimiento	medio	alto	medio
Tiempo de Desarrollo Asociado	medio	medio	bajo
Favorabilidad para el Operador	bajo	bajo	alto

Tabla 2. Tabla comparativa de Arquitecturas Candidatas.

3.5 Arquitectura Genérica del Mecanismo de Integración de JAIN SLEE y un entorno IMS

Según la comparación de las arquitecturas expuestas en la sección anterior, la arquitectura basada en servicios independientes es seleccionada como la mejor opción para la creación de los componentes reutilizables dentro del SLEE. Con esta alternativa el uso de las interfaces IMS a un nivel mucho mayor es facilitado, de tal manera que la integración entre JAIN SLEE y un entorno IMS sea hecho de una forma más rápida y sencilla para favorecer la creación de los servicios con un *Time To Market*[13] mucho menor.

Por lo tanto, estos servicios son los encargados de manejar las interfaces IMS y sus protocolos de bajo nivel con sus respectivas extensiones. En adelante se hace referencia a este conjunto de servicios como “el Conjunto de Servicios de Interfaces IMS”.

La arquitectura del mecanismo se puede apreciar en la figura 27.

13 Time to Market es un termino comúnmente usado para referirse al tiempo que toma llevar un producto desde su conceptualización hasta la comercialización.

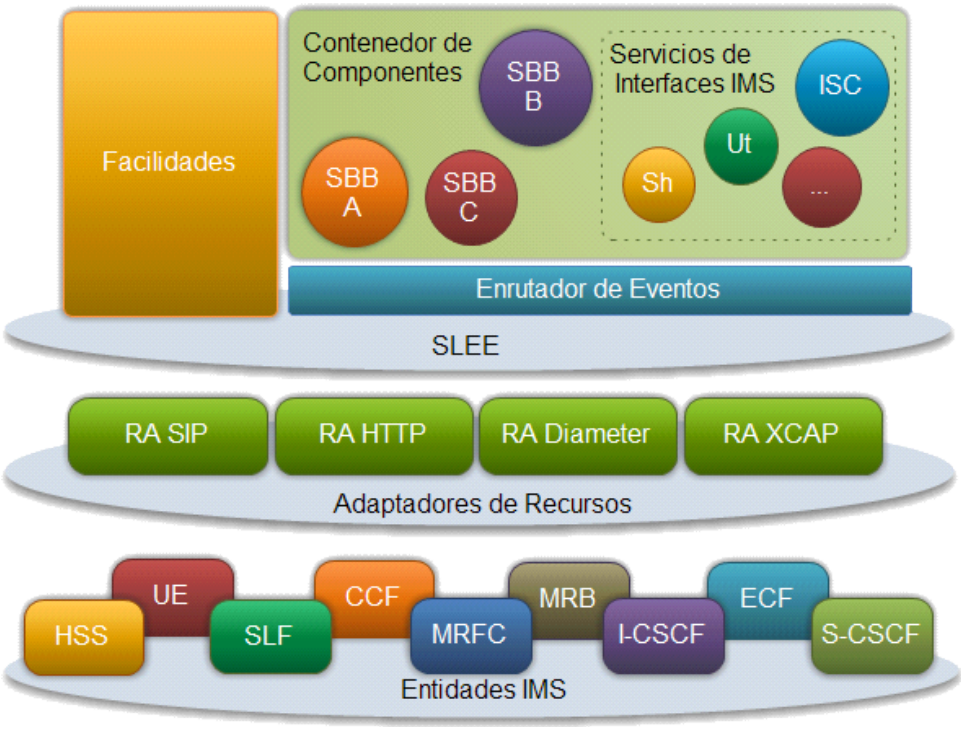

Figura 27. Arquitectura Genérica del Mecanismo de Integración.

3.5.1 Descripción de los Servicios de Interfaces IMS

Cada uno de estos servicios puede estar compuesto por uno o varios SBB que usan relaciones padre-hijo entre sí con el fin de optimizar los tiempos de respuesta al interior del servicio, pero que hacia el exterior ofrecen un comportamiento asincrónico. Aunque esta comunicación mediante servicios independientes es realizada de forma asincrónica, el tiempo empleado por el enrutador de eventos para enviar los mensajes es tan pequeño que no representa un problema mayor, lo cual permite seguir garantizando el correcto uso de las características que ofrece JAIN SLEE para la ejecución de servicios de telecomunicaciones.

Como el objetivo es que ningún servicio de valor agregado alojado en el SLEE tenga acceso directo a las entidades IMS, los Servicios de Interfaces IMS son los únicos que manejan los Adaptadores de Recursos SIP, Diameter y demás, por lo que en sus documentos descriptores de despliegue tienen datos específicos acerca de las entidades de la red, los cuales son solo conocidos y controlados por el operador de telecomunicaciones.

Toda la lógica correspondiente al funcionamiento de estos Servicios de Interfaces IMS es implementada en clases Java abstractas normales, y los contratos sincrónicos son implementados en Interfaces Java. Es así, como cada uno de los

SBB que componen a estos Servicios de Interfaces IMS, está compuesto por una o más clases, las cuales contienen los métodos necesarios para el procesamiento de cada uno de los mensajes de protocolo entrantes, y los métodos correspondientes a la devolución de los resultados del procesamiento hacia los servicios de valor agregado. Idealmente, después de realizada esta tarea, las Entidades SBB deberían ser eliminadas del SLEE, a menos que el servicio de valor agregado exija lo contrario, lo cual hace que el ciclo de vida de las Entidades SBB de los Servicios de Interfaces IMS sea muy corto y los recursos del sistema sean liberados rápidamente.

Una parte importante de estos Servicios de Interfaces IMS son sus documentos descriptores de despliegue, ya que en ellos deben estar contenidos datos concretos acerca de los RA que cada servicio emplea, como también permisos de seguridad necesarios para acceder a ellos. Es de mucha importancia la correcta referencia a los mensajes específicos de cada protocolo a soportar, ya que de estas definiciones depende totalmente el grado de compatibilidad que el Servicio de Interfaz IMS tiene con la interfaz IMS que implementa.

Es así, como puede visualizarse que estos servicios son convertidos de cierta forma en una abstracción adicional de los protocolos de comunicación a un nivel mucho más alto, facilitando en gran medida la creación y despliegue de nuevos servicios.

3.5.2 Funcionamiento General de la Arquitectura Genérica de Integración de JAIN SLEE y un entorno IMS

Esta propuesta de integración hace posible que ningún servicio de valor agregado necesite manejar los protocolos que manejan las interfaces IMS entre el AS y cualquier otra entidad. De este modo, cuando un servicio de valor agregado desea por ejemplo manipular los datos de la sesión SIP del usuario, envía un evento indicando los cambios requeridos al servicio encargado de la interfaz ISC y este hará los cambios de manera transparente, tras lo cual es opcional que le devuelva un evento informando acerca del resultado de la operación. Otro ejemplo, puede apreciarse cuando un servicio de valor agregado desea almacenar datos arbitrarios de servicio asociados a la personalización, esta operación es realizada sobre un servidor HSS haciendo uso del protocolo Diameter tal y como debe usarse en la interfaz Sh. Para este caso de uso, el servicio de valor agregado debe enviar un evento indicando que desea realizar esta operación para que el Enrutador de Eventos automáticamente inicialice una instancia del Servicio de Interfaz Sh para que la lleve a cabo y opcionalmente informar al servicio de valor agregado sobre el resultado de la operación mediante otro evento. De igual forma, debería hacerse para cada operación que el operador de telecomunicaciones desee proveer de manera similar en sus servidores de aplicaciones JAIN SLEE.

Los detalles concretos acerca de las características de las implementaciones de estos servicios están a discreción del operador, es decir, un operador puede personalizar estos servicios para que satisfagan sus necesidades especificas, por ejemplo agregarles capacidades de generación de reportes para saber con detalle las estadísticas de uso por parte de los demás servicios de valor agregado o capacidades de gestión, de tal forma que sea posible activar, desactivar y/o limitar el uso de estos Servicios de Interfaz IMS por parte de otros.

3.6 Conclusión del Capítulo

Con la definición de los Servicios de Interfaces IMS, ha logrado quedar definido el mecanismo de integración que buscaba este trabajo y por lo tanto, el segundo objetivo específico planteado ha sido cumplido, restando solo confirmar su eficacia a través de un prototipo de validación.

Es de suma importancia resaltar, que el mecanismo definido debería ser tomado solo como una referencia para implementaciones reales, ya que los criterios seleccionados para su definición no necesariamente van a ser adecuados para todos los operadores.

CAPITULO 4. CASO DE ESTUDIO PARA LA VALIDACION DEL MECANISMO DE INTEGRACION

Después de la definición del mecanismo de integración de JAIN SLEE y un entorno IMS es necesario realizar la validación del funcionamiento de dicha propuesta con el fin de verificar su efectividad. Así, este capítulo contempla primero la definición del escenario IMS en el que está enmarcado el presente trabajo y que de ahora en adelante es llamado el “entorno IMS” el cual determina las interfaces que serán usadas, lo que también define los Servicios de Interfaces IMS necesarios. También explica como un servicio sencillo consume los Servicios de Interfaces IMS para facilitar su comunicación e interacción con el entorno IMS establecido. Así mismo, explica los criterios con los que fueron seleccionadas las herramientas necesarias para llevar a cabo este caso de estudio.

4.1 Definición del Escenario IMS

Debido a que la cantidad de entidades que pueden tener interrelación con el AS es considerable y a que las interfaces respectivas en varios casos no están completamente definidas en las especificaciones del 3GPP para IMS, es necesario seleccionar un entorno IMS reducido pero que al mismo tiempo sea representativo y que sirva de modelo base para entornos más grandes.

Teniendo esto en cuenta es definido el siguiente entorno IMS, ver figura 28.

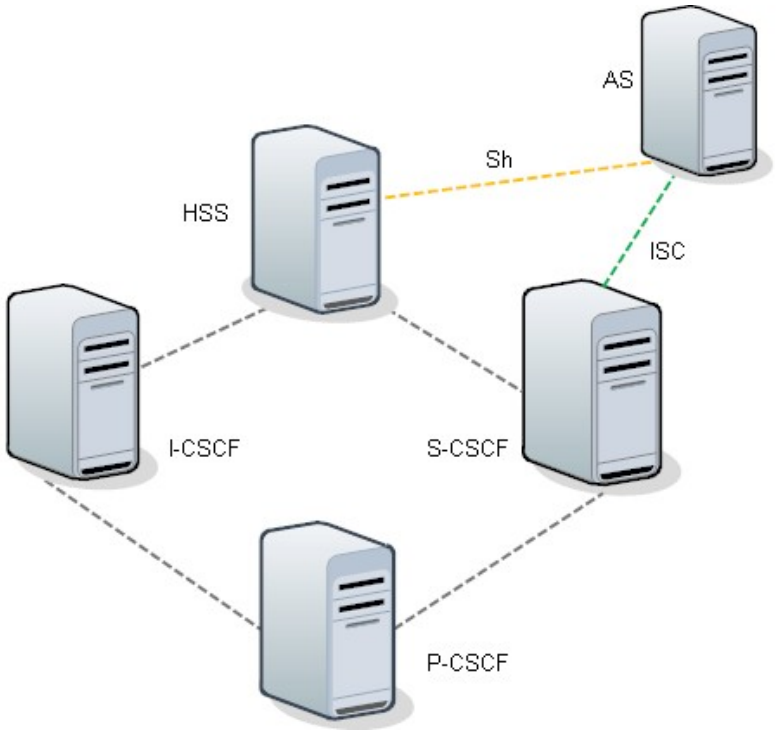

Figura 28. Entorno IMS seleccionado.

El entorno IMS seleccionado consta de un AS que implementa el estándar JAIN SLEE y que está conectado al S-CSCF y al HSS, además de la participación del P-CSCF y el I-CSCF como parte del núcleo de IMS. Este entorno es sin duda alguna el más general que puede darse para un AS SIP, pues sin la interfaz ISC los servicios no podrían ser disparados en forma selectiva, y sin la interfaz Sh los servicios no podrían ser personalizados con base en la información detallada del usuario almacenada en el HSS.

Es importante resaltar que las entidades seleccionadas para la conformación del entorno cumplen funciones de gran importancia como es el caso del S-CSCF quien, por ser el nodo central del plano de señalización, tiene como funciones principales el procesamiento y gestión del estado de todas las sesiones a la vez que tiene la función de identificar cuando una llamada o sesión debe encaminarse hacia un AS. Otra función importante que tiene el S-CSCF es la gestión y monitoreo del uso de recursos por parte de las llamadas.

De igual forma, el HSS también realiza un papel muy importante dentro de un entorno IMS ya que es el repositorio de todos los datos referentes al perfil de los usuarios, como son la información de autenticación para poder utilizar la red y los datos temporales correspondientes a los estados de registro y de localización [25] [35].

Las razones para la selección de este entorno son las siguientes:

- Sin la interfaz ISC no es posible conectar un AS SIP a una red IMS de tal forma que puedan consumirse servicios con base en las necesidades dinámicas de los usuarios, ya que al omitirse al S-CSCF en el camino de comunicación hacia el AS no pueden usarse los iFC (Initial Filter Criteria, Criterios de Filtro Iniciales).

- Sin la interfaz Sh un servicio alojado en un AS no puede personalizarse de acuerdo a la información que el operador posee acerca del usuario que desea consumir el servicio, perdiéndose así la gran ventaja que tienen los operadores de telecomunicaciones sobre las empresas que prestan sus servicios sobre Internet.

- Las entidades de red S-CSCF y HSS hacen parte del núcleo de una red IMS, mientras que otras entidades son opcionales 3GPP 23228 [17] [18].

- Los protocolos base que son usados en la mayoría de las interfaces son SIP y Diameter, con algunas extensiones dependiendo de la interfaz. Entonces, el usar la interfaz ISC (SIP) y la interfaz Sh (Diameter) hace que extender el presente trabajo a las demás interfaces pueda hacerse de forma natural.

- La interfaz Dh es una ligera variación de la interfaz Sh que aparece cuando existen múltiples HSS en la red.

- La interfaz Ma también usa el protocolo SIP y es muy similar a la interfaz ISC. Además es necesario recordar que al utilizar esta interfaz, la intervención del S-CSCF es omitida, lo cual es poco común.

- La interfaz Ut implica la participación de otros componentes adicionales encargados de realizar una autenticación del terminal de usuario antes de permitir su conexión con el AS [18], además requeriría en el AS de un servicio encargado de la personalización y configuración de los servicios, lo cual aleja la temática trabajada y de la intención de definir un escenario reducido de IMS.

- Las interfaces Ro y Rf son usadas para el intercambio de información de tarificación *offline* y *online*, pero al igual que la interfaz Sh también usan el protocolo Diameter con diferentes comandos que no representan cambios a nivel de arquitectura sino solo de implementación.

4.2 Definición de los Servicios de Interfaces IMS

Según las interfaces establecidas en el entorno IMS es necesario definir dos servicios independientes en donde cada uno maneja los protocolos correspondientes a su interfaz y expone unos eventos personalizados que permiten el uso de las funcionalidades implementadas.

A continuación son presentados los servicios definidos.

4.2.1 Servicio para la Interfaz ISC

Maneja los mensajes SIP que pueden ser intercambiados entre un AS y un S-CSCF. En este caso los mensajes ocurren principalmente en la etapa previa al consumo del servicio, ya que están encargados del establecimiento de la sesión del mismo, por lo cual el servicio recibe mensajes SIP INVITE y crea diálogos SIP [14] que los representan de forma lógica. De igual forma, el servicio maneja la finalización de las sesiones con la correspondiente eliminación de los diálogos.

El servicio tiene como función principal ahorrar tiempo a los desarrolladores de servicios de valor agregado que desean usar una red IMS para desplegar sus servicios, así como también de servir como un ocultador de información incluida en ciertas cabeceras SIP que el operador pueda estar interesado en esconder.

Los eventos personalizados que el servicio puede recibir desde un servicio de valor agregado son mostrados en la figura 29.

Figura 29. Eventos de entrada al Servicio ISC.

- **messageEvent:** este evento contiene una cadena de texto plano que un servicio de valor agregado desee enviar a manera de mensaje de texto a un usuario.

- **callEndedEvent:** este evento no contiene ningún campo y se usa solo

14 Un dialogo SIP es una secuencia de transacciones que conservan los mismos datos en los campos Call-ID (Id. de llamada), From (De) y To (Para).

como una indicación para terminar una llamada o sesión.

Los eventos personalizados que el servicio puede enviar hacia un servicio de valor agregado pueden observarse en la figura 30.

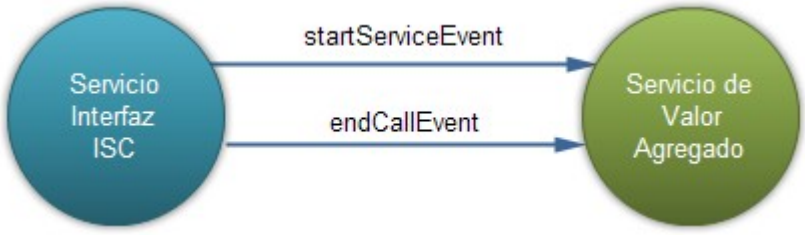

Figura 30. Eventos de salida del Servicio ISC.

- **startServiceEvent:** Este evento contiene una cadena de texto que indica la identidad del usuario que desea iniciar la llamada o sesión y la identidad del servicio o usuario al que es deseado llamar.

- **endCallEvent:** Este evento no contiene ningún campo y solo se usa como un indicador de que el usuario ha terminado la llamada o sesión.

4.2.2 Servicio para la Interfaz Sh

Es encargado del manejo de todos los mensajes que pueden intercambiarse entre un AS y un HSS sobre la interfaz Sh. Ellos pueden ser operaciones de consulta de datos de perfiles, actualización de datos de perfiles, subscripción a los cambios de la información almacenada y notificación de dichos cambios. Igualmente es encargado de ocultar toda la información extra que llega con las respuestas a las solicitudes hechas al HSS, restringiendo así su acceso y garantizando además la obtención de solo la información requerida, la cual es enviada en un formato mucho más entendible hacia el servicio de valor agregado que reutiliza este servicio.

Además del significativo ahorro en el tiempo de desarrollo que es logrado reutilizando este componente, este servicio cumple la función de ocultar totalmente el HSS hacia los servicios de tal modo que los posibles riesgos de seguridad asociados a esta interfaz sean eliminados.

Dentro de las características principales que el servicio implementa están el manejo del protocolo Diameter según la aplicación Sh y la adecuación de documentos XML de tal forma que el HSS no recibe documentos que no cumplen

con el esquema requerido por las especificaciones de IMS.

Los eventos personalizados que el servicio puede recibir desde un servicio de valor agregado son mostrados en la figura 31.

Figura 31. Eventos de entrada del Servicio Sh.

- **queryEvent:** es usado para realizar una consulta sobre un HSS y contiene un campo que indica el tipo de consulta a realizar (datos arbitrarios de servicio, estado del usuario o localización del usuario).

- **updateEvent:** es empleado para realizar una actualización de datos arbitrarios de servicio y contiene dos campos, uno para los datos a actualizar y otro que es el indicador del servicio.

- **subscribeEvent:** es usado para realizar una suscripción sobre cambios en datos almacenados en el HSS y contiene dos campos, uno que es el indicador del dato sobre el que la suscripción es realizada y otro para indicar si es necesario realizar una suscripción o una cancelación.

Los eventos personalizados que este servicio puede enviar hacia un servicio de valor agregado son mostrados en la figura 32.

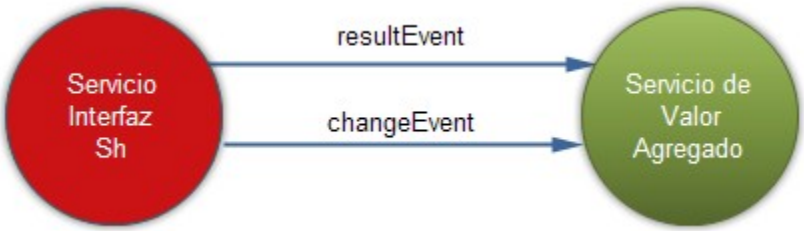

Figura 32. Eventos de salida del Servicio de Interfaz Sh.

- **resultEvent:** representa el resultado del procesamiento de un evento

(queryEvent, updateEvent o subscribeEvent) y contiene un campo que indica si este fue exitoso o fallido.

- **changeEvent:** es usado cuando ha ocurrido un cambio en algún dato en el HSS sobre el cual existe una suscripción y contiene un campo que indica los cambios sufridos por dicho dato.

De esta manera todas las capacidades disponibles en la interfaz Sh son provistas.

4.3 Definición del Servicio de prueba

La descripción del servicio es realizada desde el punto de vista de 3 diferentes actores como sigue:

- **Usuario final:** El servicio debe permitir, a un cliente de un operador de IMS, recibir mensajes de texto que contengan comentarios en vivo generados por un comentarista deportivo acerca de los partidos del mundial de Sudáfrica. Así mismo, el usuario debe tener la capacidad de iniciar y cerrar la sesión del servicio, de modo que solo reciba mensajes de texto mientras esté conectado, y cada vez que el usuario inicia sesión en el servicio, el sistema debe enviarle un mensaje de texto que le indique cuantos mensajes le quedan en su saldo.

- **Operador IMS**: El servicio debe permitir a un operador IMS hacer difusión masiva de mensajes de texto a los usuarios conectados al servicio a partir de los mensajes de mensajería instantánea que un comentarista deportivo envía a través de su cuenta personal de Gtalk[15].

  El saldo de los usuarios es manejado mediante un número que indica la cantidad de mensajes de texto por los que un usuario ha pagado, de modo que el servicio nunca envíe mensajes a usuarios sin saldo.

  Además, los datos de servicio de cada usuario deben almacenarse en el HSS, para lo cual el servicio debe hacer uso del Servicio de Interfaz Sh provisto en el AS JAIN SLEE, debido a que el acceso al HSS está restringido. De igual manera, el operador desea que las cabeceras SIP relacionadas con los datos de tarificación y que contienen las direcciones físicas de los equipos encargados de esta función esten ocultas, para ello exige el uso del Servicio de Interfaz ISC para el manejo de las sesiones.

  El servicio no debe requerir acceso a ningún RA adicional al de XMPP (Extensible Messaging and Presence Protocol, Protocolo Extensible de

15 Gtalk es un servicio de mensajería instantánea que se presta sobre Internet de la empresa Google Inc.

Mensajería y Comunicación), encargado de manejar la parte de la mensajería instantánea con Gtalk, ni tampoco permisos especiales de ejecución dentro del SLEE.

- **Desarrollador del servicio:** El servicio debe poder iniciar sesión en el servicio Gtalk y recibir mensajes de mensajería instantánea a través del RA de XMPP, para posteriormente reenviar su contenido al Servicio de Interfaz ISC, para que este cree los mensajes SIP apropiados y los envíe a los usuarios conectados.

  Cuando un usuario inicia sesión, el servicio debe verificar que el usuario tiene saldo disponible para poder continuar, y en caso de no tenerlo se debe terminar la sesión inmediatamente e informarle mediante un mensaje de texto que debe recargar saldo. El servicio también debe descontar saldo cada vez que el usuario haya recibido uno o varios mensajes.

4.4 Diagrama de Secuencia del Caso de Estudio

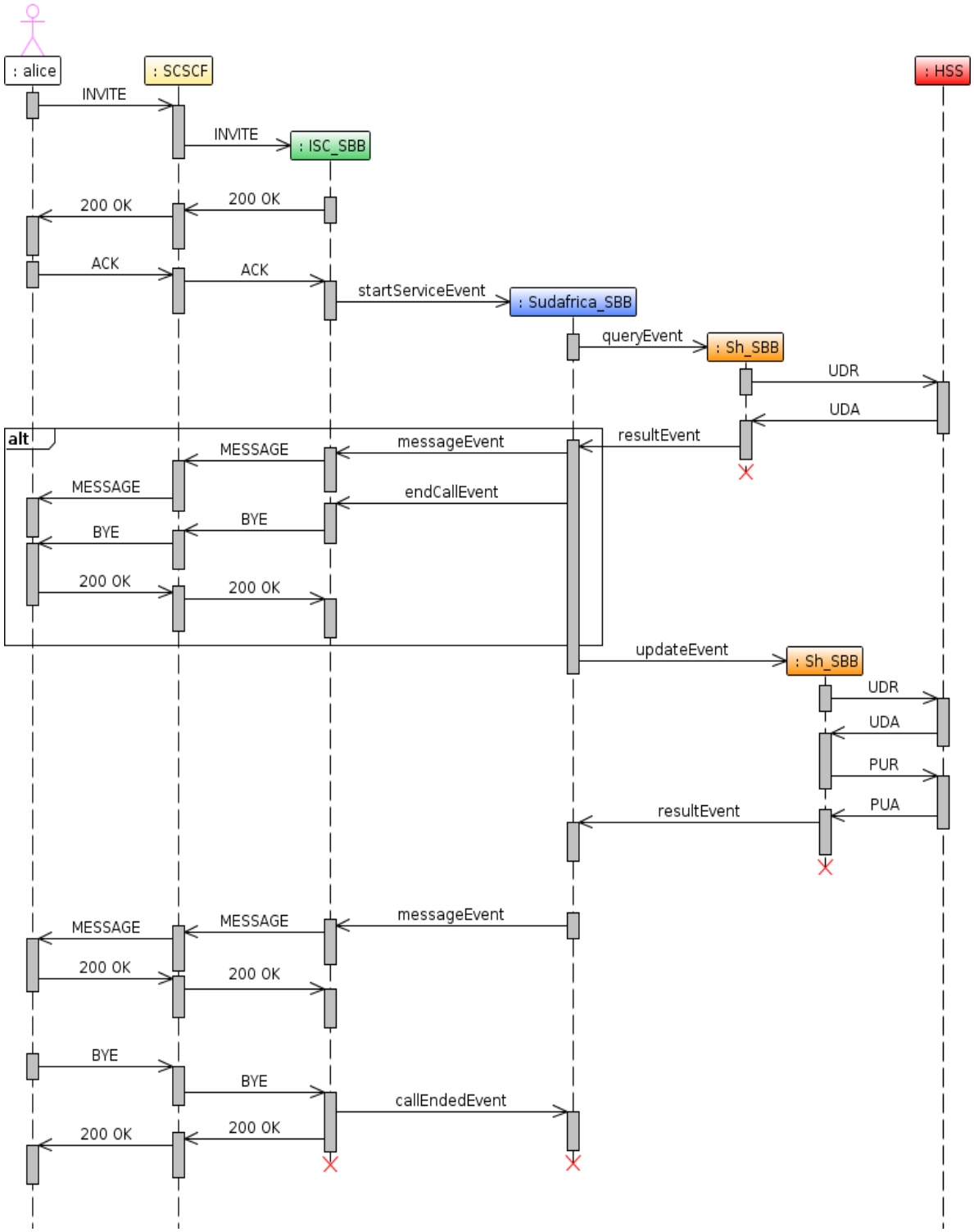

Figura 33. Diagrama de Secuencia del Prototipo.

La figura 33 es un diagrama de secuencia que explica el funcionamiento del prototipo, omite algunas entidades (especialmente el Enrutador de Eventos, ya que este solo hace la función de puente entre dos entidades), mensajes y casos de error con el fin de hacer énfasis en las relaciones entre los servicios y entidades involucradas en el prototipo.

Todo comienza cuando un usuario previamente registrado en la red IMS, decide ejecutar su aplicación cliente del servicio de valor agregado “Sudafrica Live!”, tras lo cual la aplicación inmediatamente procede a enviar un mensaje SIP solicitando el inicio de la sesión, ello es efectuado mediante un mensaje SIP INVITE enviado al contacto sudafricalive@open-ims.test.

Cuando este mensaje SIP llega hasta el núcleo de la red IMS, es decir el S-CSCF, este procede a evaluar los iFC y encuentra que hay uno que coincide con ese mensaje SIP en particular, entonces este ejecuta las acciones necesarias indicadas por el iFC y reenvía la petición directamente hacia el AS correspondiente. Cabe resaltar que el S-CSCF debe tener configurado correctamente el iFC para que sean encaminados por la misma ruta todos los mensajes SIP que pertenezcan a una misma sesión.

Cuando el mensaje SIP INVITE llega al AS, específicamente al Enrutador de Eventos, este procede a evaluar si existe algún servicio activo en el SLEE que haya indicado en sus descriptores de despliegue que los eventos SIP INVITE son eventos que requieren de la creación de una Entidad SBB de esos servicios. Como resultado de esta evaluación, una Entidad SBB del Servicio de Interfaz ISC es creada.

Una vez la Entidad SBB del Servicio de Interfaz ISC ha sido creada, el Enrutador de Eventos procede a enviarle el evento SIP INVITE para que ésta lo procese. Luego de que este evento ha sido procesado, el servicio continua y envía un mensaje SIP 200 OK y queda a la espera del ACK de parte de la aplicación cliente en el terminal de usuario. Cuando el ACK llega, entonces el inicio de la sesión previa es completado correctamente y el Servicio de Interfaz ISC crea un dialogo SIP persistente que representa toda la sesión. Durante el procesamiento del evento ACK, el Servicio de Interfaz ISC procede a crear un evento simplificado que solo contiene la identidad SIP del usuario que envió el mensaje original y el destinatario del mensaje, de modo que el evento solo contiene los campos alice@open-ims.test como remitente y sudafricalive@open-ims.test como destinatario. Una vez el evento personalizado ha sido creado, la Entidad SBB procede a enviarlo al Enrutador de Eventos y este repite el procedimiento de evaluación de los servicios activos que puedan estar interesados en tal evento y como resultado una nueva Entidad SBB del SBB raíz del servicio de valor agregado “Sudafrica Live!” [16] es creada.

16 El servicio debe tener un método InitialEventSelector definido, debe encargarse de filtrar solo los mensajes personalizados que tengan como destinatario a sudafricalive@open-ims.test.

Después de que la Entidad SBB del servicio de valor agregado ha sido creada, el Enrutador de Eventos procede a enviarle el evento simplificado creado previamente por el Servicio de Interfaz ISC. Aquí, lo que haga esta Entidad SBB depende totalmente de la lógica del servicio, que para este caso, lo primero que hace es verificar que el usuario que ha solicitado iniciar la sesión tiene saldo disponible en su cuenta, y en caso de no tenerlo, terminar la sesión inmediatamente después de enviar un mensaje de texto informándole que debe recargar. Para hacer la consulta, crea un evento simplificado que solo contiene la identidad del usuario sobre el que va a realizar dicha consulta y el tipo de consulta que desea realizarse sobre el HSS, que para este caso, es una consulta sobre datos arbitrarios del servicio “Sudafrica Live!”. Tras crear el evento simplificado, este es enviado al Enrutador de Eventos.

De la misma forma que con los anteriores eventos, el evento personalizado es procesado por el Enrutador de Eventos y como resultado crea una Entidad SBB del Servicio de Interfaz Sh. Luego, esta Entidad SBB recibe el evento personalizado y lo procesa, lo que resulta en la identificación completa del tipo de consulta que desea hacerse sobre el HSS para así crear un mensaje Diameter apropiado para iniciar la transacción de consulta. Cuando este mensaje es enviado por esta Entidad SBB hacia el Enrutador de Eventos, este lo identifica como un evento que debe ser enviado a través del RA de Diameter sobre la interfaz Sh del AS hacia el HSS. De aquí en adelante pueden ocurrir varias cosas dependiendo del HSS y de las cuales el Servicio de Interfaz Sh es encargado completamente. Posteriormente, cuando la Entidad SBB y el HSS han terminado la transacción, la Entidad SBB procede a crear un evento personalizado de respuesta que contiene el resultado y los datos de la operación de consulta, y lo envía a través del Enrutador de Eventos de regreso a la Entidad SBB del servicio de valor agregado, tras lo cual la Entidad del Servicio Sh es eliminada y el objeto SBB que la representada vuelve a la piscina de objetos disponibles del SLEE.

Cuando la Entidad SBB del servicio de valor agregado recibe el evento personalizado, obtiene los resultados sobre la consulta que realizó, específicamente son dos valores, el primero es un indicador de si la consulta pudo ser realizada correctamente y el segundo son los datos XML correspondientes a los datos arbitrarios de servicio, los cuales están presentes solo si el resultado de la consulta fue positivo. Si al analizar el documento XML, su resultado arroja que el usuario no tiene saldo suficiente, entonces la Entidad SBB crea un evento personalizado que contiene un mensaje de texto para el usuario y lo envía al Enrutador de Eventos para que este lo entregue a la Entidad SBB del Servicio de Interfaz ISC, para que a su vez, esta cree un mensaje SIP MESSAGE a semejanza del evento personalizado recibido y lo envíe al terminal de usuario. Luego, la Entidad SBB del servicio de valor agregado, crea otro evento personalizado que indica el termino de la sesión con el usuario y lo envía igualmente al Enrutador de Eventos para que lo entregue a la Entidad SBB del Servicio de Interfaz ISC, tras lo cual la Entidad SBB del servicio de valor agregado

es eliminada inmediatamente. Al mismo tiempo, la Entidad SBB del Servicio de Interfaz ISC crea un mensaje SIP BYE que envía hacia el Enrutador de Eventos para que sea enviado al terminal de usuario, tras lo cual esta Entidad SBB también es eliminada y todo habrá terminado. Para el caso en que el usuario sí tiene saldo disponible, entonces la Entidad SBB del servicio de valor agregado procede a conectar al usuario al servicio de narración para luego descontar un partido de la cuenta personal del usuario, para lo cual crea un evento personalizado que contiene la identidad del usuario y el nuevo documento XML con los nuevos datos de servicio, y lo envía al Enrutador de Eventos.

El Enrutador de Eventos procesa este evento personalizado de la misma manera realizada anteriormente y finaliza creando una nueva Entidad SBB del Servicio de Interfaz Sh. De manera similar al caso de la consulta, esta Entidad esta encargada de realizar toda la operación de actualización de datos arbitrarios de servicio sobre el HSS, para lo cual realiza una serie de intercambios de mensajes un poco más elaborada que para el caso anterior. Al finalizar, y si todo ha resultado correctamente, esta Entidad SBB procede a crear un evento personalizado de respuesta para enviarlo de regreso a la Entidad SBB del servicio de valor agregado a través del Enrutador de Eventos, este evento contiene en este caso solo un indicador acerca del resultado de la operación. Tras enviar este evento, la Entidad SBB del Servicio de Interfaz Sh es eliminada inmediatamente y el objeto que la representaba vuelve a la piscina de objetos disponibles.

En este punto, el usuario está conectado al servicio y ya se le ha descontado el presente partido de su cuenta personal, por lo que cada vez que la Entidad SBB reciba un mensaje XMPP desde el servicio Gtalk por el comentarista, ésta crea un evento personalizado con el contenido del mensaje instantáneo recibido y lo envía al Enrutador de Eventos para que lo entregue a la Entidad SBB del Servicio de Interfaz ISC y así crea un mensaje SIP MESSAGE para que sea enviado al terminal de usuario.

Por último, cuando el usuario desea terminar la sesión, la aplicación cliente de su terminal envía el correspondiente mensaje SIP BYE que es encaminado, a través de la red de acceso del caso, hacia el S-CSCF, luego es entregado al AS y recibido por la Entidad SBB del Servicio de Interfaz ISC que a su vez, crea un evento personalizado que indica el fin de la sesión y lo envía al Enrutador de Eventos para que lo entregue a la Entidad SBB del servicio de valor agregado, la cual al recibirlo es eliminada. Previamente, la Entidad SBB del Servicio de Interfaz ISC al recibir el mensaje SIP BYE, responde automáticamente con un mensaje SIP 200 OK y también es eliminada pues la sesión ha terminado. Después de haber sido eliminadas todas las entidades SBB, los objetos SBB que las representaban regresan a la piscina de objetos SBB disponibles.

4.5 Selección de la Implementación del Núcleo de IMS

Para la implementación del prototipo es necesario un entorno IMS que soporte idealmente las últimas versiones de las especificaciones del 3GPP, dentro de las cuales están las siguientes dos opciones:

- **Ericsson Service Development Studio, SDS:** es una herramienta provista por Ericsson que puede integrarse con el IDE[17] Eclipse, lo cual permite que a medida que los servicios IMS sean desarrollados pueden probarse de forma inmediata, ya que provee un ambiente de pruebas potente y muy fácil de usar, el cual puede realizar un seguimiento de las secuencias de eventos intercambiadas entre actores en tiempo real, al igual que los contenidos de los mensajes de señalización [36].

  Esta herramienta aprovisiona al entorno con un solo componente que integra un P-CSCF, un I-CSCF, un S-CSCF y un HSS, con lo cual las principales entidades son englobadas, negando de esta forma la posibilidad de configurar y estudiar el comportamiento de los eventos entre ellas, así como también la manipulación de las interfaces de interconexión entre los mismos y el AS por separado.

  Es importante resaltar que dada esta condición, la conexión entre el HSS y el AS es realizada a través del S-CSCF, por lo tanto la interfaz Sh no es expuesta, así como también solo es permitido la lectura de los perfiles almacenados en el HSS, más no su modificación, ni ninguna de las operaciones nombradas en el Capítulo 2 [37].

  A pesar de que existen bastantes ventajas sobre todo a nivel de configuración, el SDS cuenta con la desventaja de estar contenido dentro de eclipse y solo estar disponible para sistemas Windows. El hecho de estar inmerso dentro de Eclipse afecta de forma muy negativa al rendimiento especialmente durante mediciones de tiempos de respuesta y uso de memoria RAM. Sin embargo, el principal inconveniente de esta herramienta radica en el hecho de no proveer acceso a las interfaces hacia las entidades IMS con excepción de la ISC.

- **Open IMS Core:** es una implementación libre del núcleo de IMS, donde cada una de las entidades que lo conforman están implementadas de forma separada.

  Es necesario destacar que esta herramienta no fue creada con el propósito de actuar como un producto en un contexto comercial, sino que tiene como fin proveer una implementación de referencia del núcleo de IMS que

17 Entorno de Desarrollo Integrado (IDE, Integrated Development Environment)

permita llevar a cabo pruebas de concepto sobre tecnologías IMS, así como también de prototipos de aplicaciones [38].

Esta herramienta permite manejar interfaces como la ISC, lo cual faculta a los desarrolladores para hacer uso de las características de enrutamiento, de los lanzadores de aplicaciones IMS y de otras características relacionadas al S-CSCF. De igual forma, ofrece la interfaz de conexión hacia el HSS, por lo cual un desarrollador tiene acceso irrestricto a los datos de perfiles de identidades IMS.

El Open IMS Core cuenta con una interfaz web muy potente que permite la configuración de una gran cantidad de opciones como son: la creación de identidades públicas y privadas, la creación de lanzadores de servicios, la creación de perfiles, la configuración relacionada al servidor de aplicaciones, los permisos de acceso a los datos contenidos en el HSS, entre otros.

Dada la necesidad del prototipo de hacer uso de la interfaz Sh para el manejo de perfiles de usuario almacenados en el HSS, el SDS queda descartado en favor del Open IMS Core. Por otra parte, debe tenerse en cuenta que la implementación tiene un carácter completamente investigativo, por lo cual las pruebas realizadas sobre este entorno no deben tomarse como referencia a nivel de rendimiento en futuros trabajos.

4.6 Selección de la Implementación de la Especificación de JAIN SLEE

A pesar de que la primera especificación de JAIN SLEE fue publicada en el año 2004, a la fecha solo existen dos implementaciones certificadas, sin tener en cuenta la de referencia. Aunque existen algunos productos parcialmente basados en JAIN SLEE y que son usados en la actualidad, estos han sido descartados pues no garantizan que los desarrollos realizados sobre estas plataformas son aplicables a otras [5].

Los aspectos a tener en cuenta para la selección de la implementación a usar para el desarrollo del prototipo es mostrada en la tabla 3:

Adaptadores de Recursos	Los RA disponibles deben implementar los protocolos requeridos por la interfaces ISC y Sh. Adicionalmente debe contar con una buena base de RA que permitan implementar el servicio de valor agregado.
Documentación	La cantidad de documentación disponible en línea acerca del producto debe ser suficiente.
Soporte	Deben existir medios de comunicación para hacer consultas acerca del producto, ya sea directamente con el proveedor o con su comunidad de usuarios.
Costo	Idealmente el producto debe ser gratuito al menos en versiones académicas.
Facilidades de Configuración	El producto debe ser configurable al menos mediante archivos de texto plano, XML o mediante una consola de configuración con interfaz gráfica.
Especificación que se Implementa	Es necesario que el producto implemente la última versión de la especificación de JAIN SLEE.

Tabla 3. Principales características de una implementación de JAIN SLEE.

Las implementaciones consideradas son las siguientes:

- **Rhino Application Server:** es un producto privativo de la empresa OpenCloud, la cual es destacada por ser la principal patrocinadora de la especificación de JAIN SLEE.

  Los RA disponibles para Rhino provistos por OpenCloud son en su mayoría para redes basadas en SS7 (Signaling System No. 7, Sistema de Señalización No. 7) e IMS, y algunos pocos para la integración con otros sistemas. Para el caso del prototipo planteado, cumple el requerimiento de los RA para la interfaz ISC y Sh pero no para el protocolo XMPP [15].

  La documentación disponible para Rhino es abundante y de calidad gracias al portal para desarrolladores de OpenCloud, donde está la información

actualizada sobre cada característica del AS, sus RA y librerías, además, de una gran cantidad de ejemplos disponibles.

Un aspecto realmente destacable acerca de OpenCloud es el soporte que brinda para Rhino, pues en sus foros públicos casi siempre existe personal calificado dispuesto a responder preguntas y hacer sugerencias, además de la misma comunidad de usuarios que también colabora activamente.

OpenCloud ofrece una versión gratuita de prueba de Rhino que viene limitada en rendimiento y está pensada exclusivamente como una plataforma para realizar pruebas de concepto, además, ofrece una versión de producción sin limitantes, la cual no está disponible abiertamente al público y de la que su precio de comercialización es desconocido.

Rhino ofrece dos facilidades principales para su configuración, la primera es una consola de comandos interactiva, y la segunda es una consola web que ofrece las mismas capacidades que su contraparte, pero mediante una interfaz gráfica.

La especificación de JAIN SLEE que implementa Rhino es la 1.1.

- **Mobicents JAIN SLEE:** El AS Mobicents JAIN SLEE es un producto libre y totalmente gratuito de la empresa Red Hat Inc. distribuido bajo la licencia GPL[18] v2, la cual mantiene un modelo de negocio basado en soporte y no en licenciamiento.

  Mobicents JAIN SLEE tiene una base de adaptadores de recursos muy similar a Rhino, y para el caso del prototipo, cumple totalmente los requerimientos, ya que cuenta con los RA para las interfaces ISC y Sh, además del RA para el protocolo XMPP.

  La documentación disponible para Mobicents JAIN SLEE es buena, aunque no siempre está corresponde con las últimas versiones del software. Respecto a los ejemplos disponibles, estos son menos que para el caso de Rhino, aunque son un poco más completos y elaborados.

  El soporte que brinda la comunidad de Mobicents es bueno pero bastante más limitado que el brindado por OpenCloud, debido a que son los mismos desarrolladores quienes lo prestan por lo que la cantidad de tiempo que dedican a esta labor es bastante menor.

  A la fecha, Mobicents JAIN SLEE acaba de finalizar su actualización desde la versión 1 de JAIN SLEE a la versión 1.1, a la vez que está bajo

18 Licencia Pública General (GPL, General Public License)

reestructuración total de la plataforma, por lo que de momento las facilidades de configuración disponibles no están totalmente depuradas, encontrándose algunas carencias y deficiencias en su funcionamiento.

Mobicents JAIN SLEE implementa la versión 1.1 de JAIN SLEE [39].

Al finalizar la exploración, Rhino es seleccionada debido a que la documentación, soporte y facilidades de configuración son un poco mejores que en Mobicents JAIN SLEE, además de que el costo de la versión de producción fue asumido por OpenCloud y le fue otorgada a la Universidad del Cauca una licencia de uso valida por 3 meses.

Respecto al RA de XMPP, este fue adecuado desde Mobicents JAIN SLEE con relativamente poco esfuerzo, logrando de esta forma contar con todos los RA necesarios para que el prototipo pueda ser implementado sobre Rhino.

4.7 Selección del Cliente IMS

Dada la necesidad de contar con un cliente IMS que pueda manejar las funcionalidades del servicio planteado como son la creación de inicios de sesión fácilmente al igual que la recepción de mensajes SIP MESSAGE dentro de la sesión, varias pruebas con los principales clientes IMS existentes hasta el momento fueron realizadas, ellos son: Mercuro IMS Client [40], UCT IMS Client [41], IMS Communicator [42] y Monster [43].

De las pruebas realizadas, los clientes Mercuro, UCT e IMS Communicator lograron satisfacer las necesidades en cuanto al registro de usuarios en la red IMS y el establecimiento de sesiones, pero no lograron recibir mensajes SIP MESSAGE dentro de las sesiones.

Por su parte, el cliente IMS llamado Monster fue el único que respondió de forma correcta a los requerimientos del servicio, al poder manejar tanto el inicio de la sesión como la recepción de los mensajes SIP MESSAGE dentro de la misma, por lo cual fue seleccionado como el cliente IMS ha utilizar.

Es necesario resaltar que este último cliente hace parte de la misma iniciativa que desarrolla el Open IMS Core, por lo cual es comprensible que hayan funcionado de forma correcta en conjunto sin requerir configuraciones o modificaciones adicionales.

4.8 Conclusión del Capítulo

Como es posible observar, la arquitectura basada en servicios independientes

permite que el servicio de valor agregado no tenga la necesidad de entrar en contacto con los protocolos SIP y Diameter, y aun así pueda usar las funcionalidades prestadas sobre las interfaces ISC y Sh del entorno IMS. De esta forma, el mecanismo prueba ser muy efectivo sobre todo para servicios que quieran migrar desde otras redes hacia una red IMS que tenga un AS que implemente la especificación de JAIN SLEE, ya que la reutilización de los Servicios de Interfaces IMS elimina el tiempo de desarrollo asociado a la adecuación de los servicios para que puedan usar los nuevos protocolos.

Con esto el tercer objetivo específico propuesto por el presente trabajo es cumplido y parte del objetivo general ya que el mecanismo propuesto ha probado que facilita la creación y prestación de servicios de telecomunicaciones de valor agregado. Sin embargo, respecto a la prestación de los servicios, en el siguiente capítulo un análisis de rendimiento del mecanismo propuesto es hecho de modo que las inquietudes planteadas en el capítulo 3 sean abordadas.

CAPITULO 5. VALIDACION DEL CASO DE ESTUDIO DEL MECANISMO DE INTEGRACION

Es de interés en primer lugar medir la desventaja que tiene la arquitectura basada en servicios independientes respecto a la basada en relaciones padre-hijo entre SBB, es decir, medir el retardo que introduce el Enrutador de Eventos cuando este es incorporado entre dos entidades SBB. Para realizar esta medición, varios servicios simples de prueba son usados, teniedo como función realizar el registro del tiempo del envío y de llegada de eventos.

En segundo lugar, el prototipo implementado es colocado a prueba bajo condiciones de alta carga de usuarios dentro de un entorno que representa el escenario descrito en capítulos anteriores, con el fin de determinar si el mecanismo es apropiado o no para desplegar en entornos reales.

5.1 Escenario para la Realización de las Pruebas

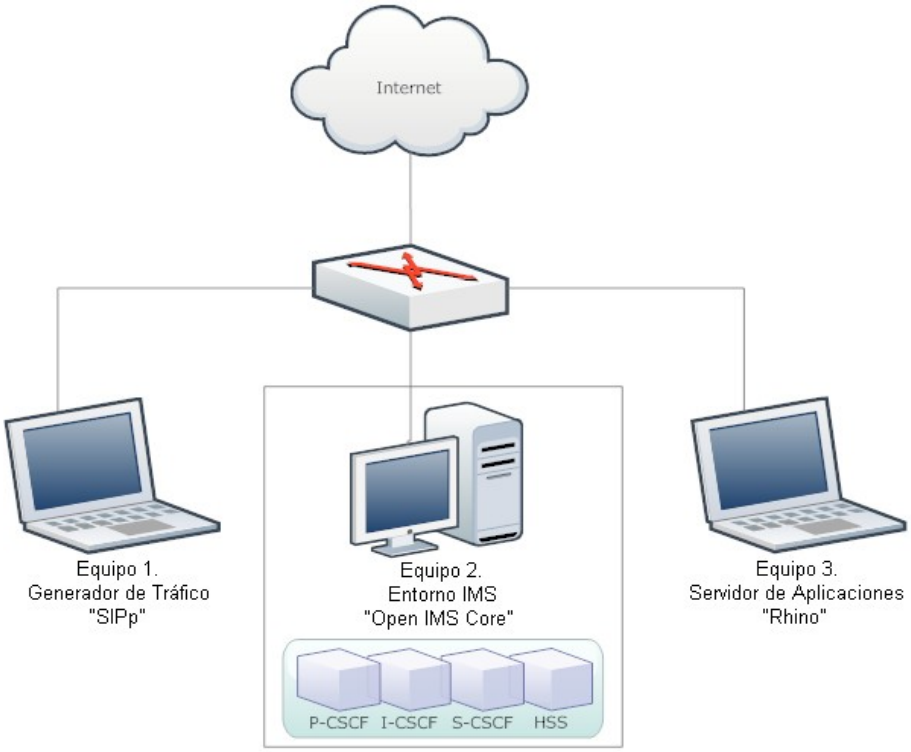

Figura 34. Escenario de pruebas.

El escenario en el cual fueron llevadas a cabo las pruebas es mostrado en la figura 34 y corresponde a una red conformada por 3 computadores y un router, el cual permite la comunicación entre ellos y provee además el acceso a internet. A continuación está la descripción de cada uno de los computadores usados.

5.1.1 Generador de tráfico

Las características hardware y software de este equipo son las siguientes:

- Procesador: Intel Core 2 Duo, 2.1GHz, 3MB de Cache y FSB 800Mhz.
- Memoria RAM: 2GB DDR2 Dual Channel, 667 Mhz.
- Disco duro: 250GB, 5400 rpm.
- Sistema Operativo: Ubuntu 9.10 x86.
- Cliente SIP: SIPp version 3.1.

El generador de tráfico usado es SIPp, el cual es un software libre para el testeo de herramientas que funcionan con el protocolo SIP. Esta puede actuar tanto como un Agente de Usuario Cliente o Servidor.

SIPp permite el uso de escenarios de prueba definidos en documentos XML con la descripción de los flujos de los mensajes SIP a utilizar durante las pruebas. Además de ello, es capaz de crear sesiones simultáneas de tal forma que puedan realizarse pruebas de carga.

Es importante resaltar que SIPp es una herramienta de línea de comandos, por lo que no cuenta con una interfaz gráfica que permita realizar de forma sencilla las pruebas o la composición de los escenarios.

5.1.2 Entorno IMS

Las características hardware y software de este equipo son las siguientes:

- Procesador: AMD Athlon 64, 2GHZ, 512KB de Cache y FSB 800MHz.
- Memoria RAM: 1GB DDR2 Single Channel, 533 Mhz.
- Disco duro: 80GB, 7400 rpm.
- Sistema Operativo: Ubuntu 9.10 x86.

El entorno IMS implementado corresponde a la descripción realizada en el capítulo 4 del presente trabajo y que en resumen consiste de 4 entidades IMS provistas por el Open IMS Core, las cuales son el P-CSCF, el I-CSCF, el S-CSCF y el HSS.

5.1.3 Servidor de Aplicaciones

Las características hardware y software de este equipo son las siguientes:

- Procesador: Intel Core 2 Duo, 2Ghz, 4MB de Cache y FSB 800MHz.
- Memoria RAM: 2GB DDR2 Dual Channel, 667 Mhz.
- Disco duro: 80GB, 5400rpm.
- Sistema Operativo: Ubuntu 10.04 x86.
- AS: Rhino Application Server de Opencloud, versión de producción.

Adicionalmente, el AS ha sido provisto con los RA de los protocolos SIP, Diameter, XMPP y Telnet además de los servicios implementados para el caso de estudio.

5.2 Medición del Impacto de la Arquitectura Basada en Servicios Independientes

Como fue mencionado en el capítulo 3, debido a la necesidad de la arquitectura basada en servicios independientes de usar el Enrutador de Eventos, es previsible que los tiempos de respuesta en esta arquitectura sean mayores a los tiempos de respuesta de la opción basada en relaciones padre-hijo entre SBB. Por lo tanto, algunas pruebas fueron realizadas con la orientación de cuantificar el impacto que tiene sobre el rendimiento de la comunicación, la arquitectura seleccionada para el mecanismo.

5.2.1 Prueba de Retardos en SBB con Relaciones Padre-Hijo

Con el fin de evaluar el modo de funcionamiento sincrónico, el servicio de prueba está compuesto de dos SBB, uno que actúa como padre y otro como hijo. El SBB padre es el que inicia la comunicación realizando un llamado sobre el SBB hijo, para que este proceda inmediatamente a efectuar un llamado de CallBack[19] sobre el SBB padre. De esta manera, el SBB padre es capaz de medir el tiempo que transcurrió desde que realizó la invocación sobre el SBB hijo hasta que éste le regrese el llamado. Como puede observarse en la figura 35, la comunicación es realizada de forma directa, es decir, sin pasar por el enrutador de eventos.

19 Los métodos de CallBack son aquellos que permiten a una entidad que es invocada por otra, invocar a la entidad llamante directamente

Figura 35. Esquema de funcionamiento del servicio de prueba.

Para indicar el inicio de la prueba, un RA de Telnet fue utilizado, de modo que cuando el SBB padre recibe un mensaje por este medio, procede a tomar el tiempo e invocar al SBB hijo, para después computarlo con el tiempo medido en el llamado de CallBack.

Con el fin de poder obtener un dato promedio de este tiempo, la toma de 30 datos que corresponden a 30 mensajes enviados a través de telnet fue realizada. En la figura 36 están los tiempos medidos durante esta prueba, y los datos obtenidos son muestrados en la tabla 4.

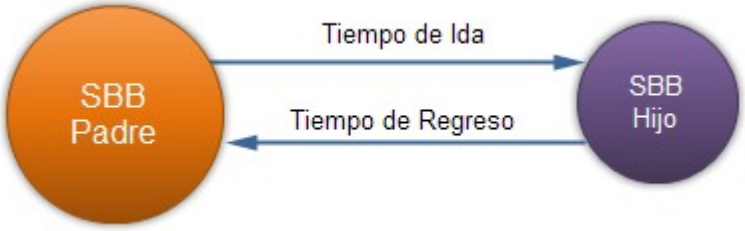

Figura 36. Tiempos de respuesta de llamados sincrónicos.

Prueba	Tiempo de ida (ms)	Tiempo de regreso (ms)
1	1	0
2	0	0
3	2	0
4	0	1
5	0	1
6	0	1
7	2	0
8	0	0
9	3	0
10	1	0
11	1	1
12	1	1
13	1	1
14	1	0
15	1	0
16	2	1
17	0	0
18	1	0
19	1	0
20	3	0
21	0	1
22	2	1
23	1	1
24	0	0
25	0	1
26	1	0
27	1	0
28	2	1
29	2	1
30	1	1

Prueba	Tiempo de ida (ms)	Tiempo de regreso (ms)
Promedio	**1.03**	**0.47**

Tabla 4. Tiempos medidos en la comunicación padre-hijo.

Como es posible observar, el tiempo de ida es siempre mayor o igual que el tiempo de regreso, lo cual se debe a que el SBB padre para poder enviar el mensaje al SBB hijo debe primero obtener su interfaz local, la cual esta almacenada en un CMP que debe ser des-serializada para poder ser usada. De igual forma, el SBB padre debe obtener un objeto local que represente la interfaz local del SBB hijo que permita el acceso a este. Por el contrario, para el tiempo de regreso el SBB hijo solo invoca el método de CallBack del SBB padre, y como todos los objetos e interfaces han sido pre-cargados en memoria, el tiempo empleado es menor.

Analizando los resultados, es posible concluir que los tiempos de respuesta obtenidos son buenos e incluso pueden ser considerados despreciables para la mayoría de las aplicaciones, así como también que la principal causa del retardo es la des-serialización de campos almacenados en CMP.

5.2.2 Prueba de Retardos en Llamados Asincrónicos con Servicios Independientes

Esta prueba mide el efecto que tiene el Enrutador de Eventos cuando es introducido en el camino entre dos SBB que pertenecen a diferentes servicios.

Para esta prueba dos servicios independientes fueron utilizados, ellos emplean eventos personalizados para su comunicación de forma asincrónica. El primer servicio es el encargado de crear y enviar un evento personalizado al segundo servicio, para que este enseguida envíe también un evento personalizado de respuesta, tal como puede verse en la figura 37.

Figura 37. Esquema de funcionamiento para la segunda prueba.

Los tiempos medidos en esta prueba fueron en primer lugar el tiempo que transcurre desde que el primer servicio envía el primer evento personalizado hasta que el segundo servicio lo recibe, ellos corresponden a los eventos señalados con el número 1 en la figura 36; el segundo tiempo medido es el que transcurre desde que el segundo servicio envía el segundo evento personalizado hasta que el primer servicio lo recibe, corresponden al número 2 en la misma figura.

Nuevamente 30 mediciones fueron realizadas y los datos obtenidos son mostrados en la tabla 5.

Prueba	**Tiempo de ida (ms)**	**Tiempo de regreso (ms)**
1	15	2
2	14	2
3	15	1
4	13	2
5	14	2
6	11	1
7	12	1
8	16	3
9	15	2
10	13	3
11	13	2
12	14	2

Prueba	Tiempo de ida (ms)	Tiempo de regreso (ms)
13	12	1
14	12	1
15	14	1
16	11	2
17	11	1
18	12	2
19	13	1
20	13	2
21	11	2
22	13	1
23	14	1
24	11	1
25	15	1
26	14	2
27	14	2
28	12	1
29	12	2
30	11	1
Promedio	**13**	**1.6**

Tabla 5. Tiempos medidos en la comunicación servicio-servicio.

Observando estos resultados, es posible apreciar como el tiempo correspondiente a la comunicación inicial entre los dos servicios toma mucho más tiempo que todas las anteriores mediciones realizadas, ello es debido a que en este caso el Enrutador de Eventos tiene que analizar el evento entregado por el primer servicio para determinar si existe algún otro servicio instalado y activo en el SLEE que esté interesado en recibir dicho tipo de evento. Adicionalmente, también debe agregarse en el tiempo de ida, el tiempo empleado en la creación de una Entidad SBB del segundo servicio.

Por el contrario, el tiempo de regreso es mucho menor y es inclusive comparable con los tiempos medidos para el caso de llamados sincrónicos entre SBB, lo cual en gran parte es debido al hecho de que todas las entidades SBB necesarias ya

han sido creadas y procesamiento adicional no es realizado por parte del Enrutador de Eventos, por lo que este último solo debe realizar la función de enrutamiento y no la de creación de nuevas entidades.

Entonces, puede concluirse que el efecto del Enrutador de Eventos en general es muy pequeño y es incluso despreciable a partir del segundo evento, cuando todas las entidades involucradas en la comunicaciones ya han sido creadas, por lo que es posible considerar que el impacto del uso de esta arquitectura en el mecanismo no es considerable.

Además, es importante resaltar que los equipos sobre los que fueron realizados estas pruebas no fueron creados para tal propósito, por lo que es de esperarse que sobre equipos adecuados con mejores características estos tiempos tiendan a bajar mucho más.

5.3 Pruebas de carga

Con el objetivo de probar si el mecanismo propuesto es adecuado para un operador de telecomunicaciones desde el punto de vista de la cantidad de usuarios que este puede manejar, es de vital importancia determinar si el prototipo implementado presenta buenas características en cuanto a altas cargas de usuarios simultáneas y que permanecen durante un tiempo considerable.

Por otra parte, debe tenerse en cuenta que los resultados obtenidos en pruebas de carga realizadas sobre el AS que implementan la especificación JAIN SLEE, pueden ser extrapolados con un buen grado de confiabilidad para entornos más grandes, ya que una de las principales características de JAIN SLEE es su excelente escalabilidad horizontal y vertical [44].

5.3.1 Valores objetivo de la prueba

La tabla 6 muestra los valores objetivos con los que se configuró el generador de tráfico.

Total de sesiones a realizar	150000
Máximo número de sesiones simultáneas	45000
Tasa de creación de sesiones	194 por segundo
Duración de la sesión	180 segundos (3 minutos)

Tabla 6. Valores objetivo de la prueba de carga.

5.3.2 Prueba sobre el Escenario Completo

El escenario total corresponde a la figura 33. Para realizar esta prueba y lograr los valores objetivo, el generador de tráfico fue ejecutado de la siguiente manera:

sipp 192.168.0.11:5060 -inf users.csv -sf uac.xml -i 192.168.0.12 -p 10100 -r 194 -m 150000 -l 45000 -trace_stat

Como primer resultado, es posible identificar que el S-CSCF provisto por el Open IMS Core era el cuello de botella de todo el sistema y limitaba el funcionamiento de este muy negativamente, obligando a tener tasas de creación de sesiones inferiores a 10 por segundo, lo cual en cierta medida era de esperarse dado de que el Open IMS es un producto orientado totalmente a la investigación. Este dato fue obtenido mediante la reconfiguración en tiempo de ejecución del generador de tráfico. Por lo tanto la ejecución del comando de SIPp generaba un tráfico que el S-CSCF no era capaz de manejar, por lo cual se decidió realizar una segunda prueba en base a la configuración de un nuevo entorno IMS.

5.3.3 Prueba sobre el Escenario Reducido

Debido a lo anterior, la realización de una segunda prueba fue decidida, en este caso el S-CSCF, con el objetivo de probar si el mecanismo podía ofrecer mejores rendimientos cuando éste fuera desplegado en un ambiente de producción. Sin embargo, el HSS fuemantenido de modo que pudieran probarse los dos Servicios de Interfaces IMS[20]. Por lo tanto, el nuevo escenario de prueba es que se muestra en la figura 38.

20 A pesar de que se aisló el S-CSCF, el generador de trafico es capaz de generar mensaje SIP de acuerdo a IMS, por lo que el servicio de Interfaz ISC es aplicable

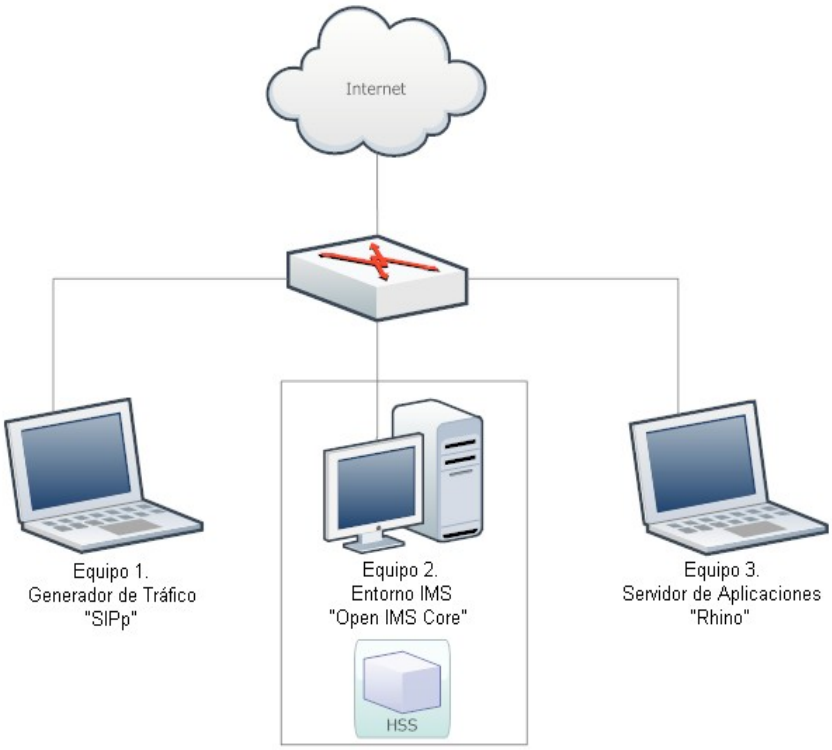

Figura 38. Escenario de prueba reducido.

Después de realizar la prueba de carga bajo las anteriores condiciones, se obtuvieron los siguientes resultados:

Cantidad de sesiones creadas por segundo

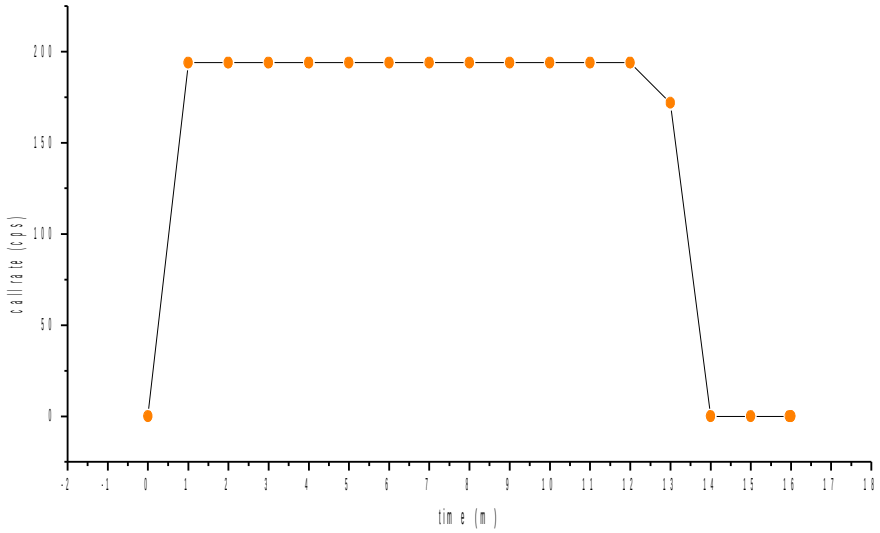

Figura 39. Tasa de creación de sesiones.

Como puede verse en la figura 39, la tasa de creación de sesiones corresponde exactamente a la tasa objetivo, lo que indica que el prototipo funcionó perfectamente y no rechazó, retrasó o encoló ningún inicio de sesión. Al final la curva empieza a caer debido a que el total de sesiones estaba por alcanzarse.

Sesiones simultáneas

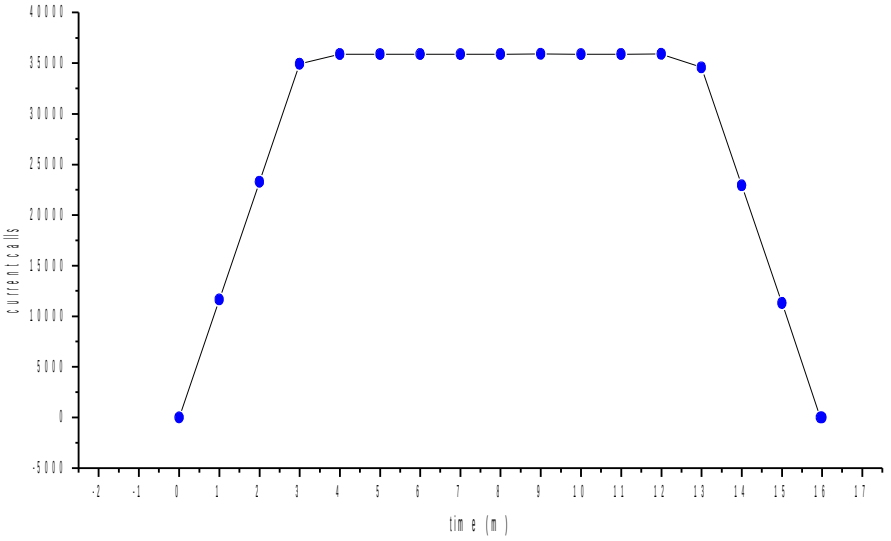

Figura 40. Sesiones simultáneas.

En la figura 40 puede observarse como el prototipo fue capaz de mantener alrededor de 36000 sesiones simultáneas sin problemas. Debe tenerse en cuenta que no alcanzó el numero objetivo propuesto debido a que a partir del minuto 3 la tasa de creación y de eliminación de sesiones pueden cancelarse perfectamente, lo que hace que el sistema este en equilibrio entorno a 36000. Igualmente se encuentra que a partir del minuto 13 la curva empieza a caer hasta que llega a cero.

Tiempos de Respuesta

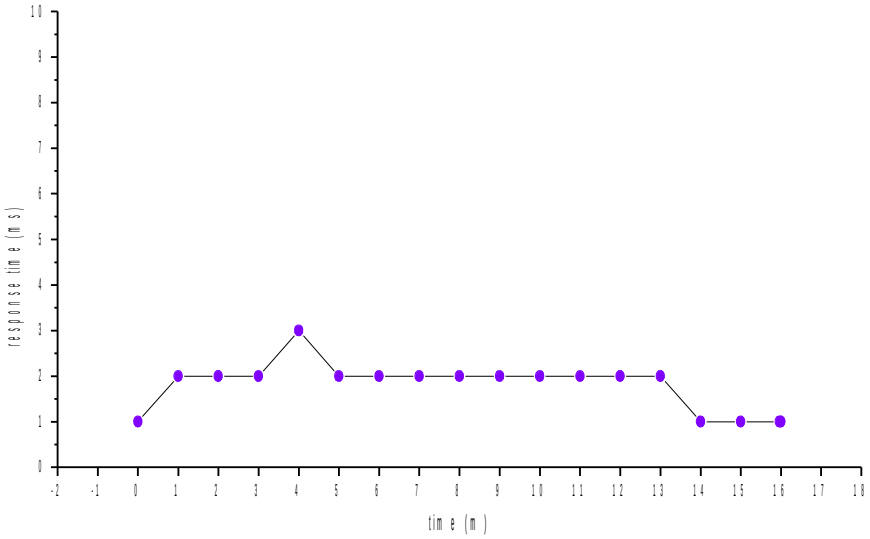

Figura 41. Tiempos de respuesta.

El tiempo de respuesta medido por el generador de tráfico corresponde al tiempo percibido entre dos mensajes SIP, como por ejemplo la pareja INVITE – 200 OK. En la figura 41 puede apreciarse que los tiempos de respuesta siempre fueron menores a 3 milisegundos, lo que es bastante bueno. Sin embargo debe tenerse en cuenta que los equipos estabán en una red LAN.

Retransmisiones

La cantidad total de retransmisiones de mensajes SIP fue 0, lo que implica un funcionamiento totalmente ideal del sistema, pero al igual que para el caso anterior, esto es sustentado fundamentalmente en el hecho de que los equipos involucrados en el flujo de los mensajes estaban dentro de una red LAN, la figura 42 muestra este comportamiento. Es de esperarse que en un ambiente real y donde aparte del trafico SIP existan tráfico adicional, la cantidad de retransmisiones se incremente.

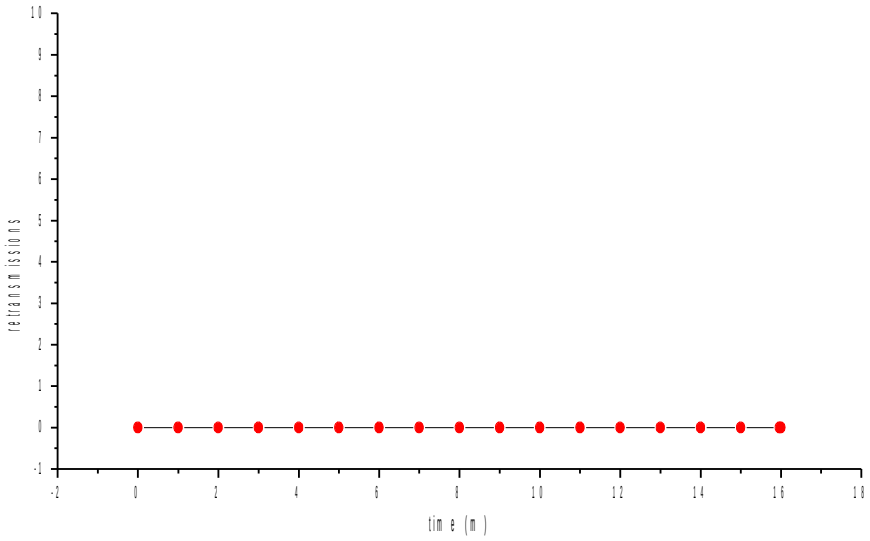

Figura 42. Retransmisiones.

Duración de las sesiones

Cada sesión estaba programada para durar 180 segundos, mas una pausa de 5 segundos de guarda en el escenario de SIPp. De este modo, cualquier retraso sobre este tiempo sería el resultado del tiempo de viaje de los mensajes dentro de la red LAN y de la inestabilidad de los temporizadores del SLEE.

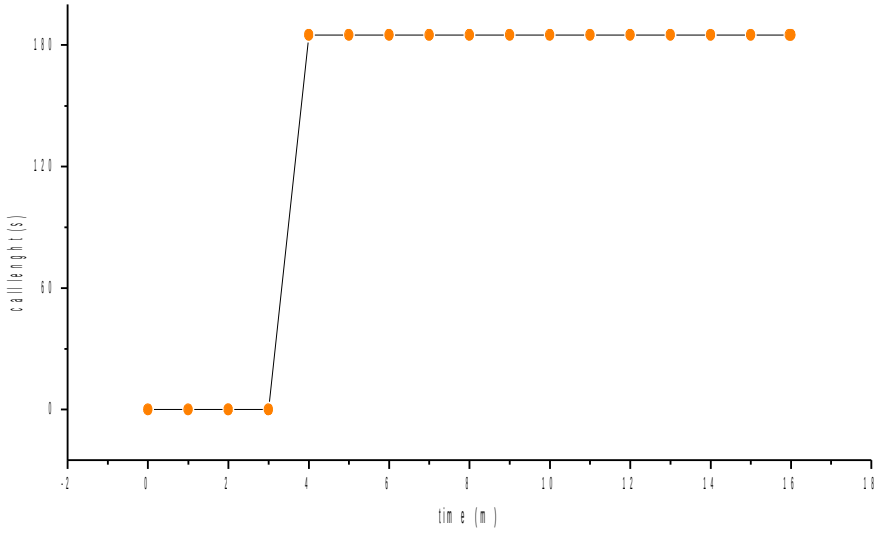

Figura 43. Duración de las sesiones.

Como puede observarse en la figura 43, la duración de las sesiones fue muy cercana a los 185 segundos, donde el retraso promedio percibido por el generador de tráfico fue de 20[21] milisegundos, lo que indica que en general los temporizadores del SLEE son estables y precisos.

5.3.4 Comportamiento Interno del SLEE durante la Prueba

A continuación se muestra el comportamiento del SLEE durante la prueba del segundo escenario descrito en el anterior ítem.

Para todos los casos presentados, las gráficas se analizan de igual manera, donde el eje vertical representa la cantidad de entidades SBB medidas en un instante de tiempo, es decir, la cantidad de instancias del servicio correspondiente que se están monitoreando en tiempo real, y el eje horizontal representa el tiempo, el cual en total corresponde a 15 minutos medidos desde las 11:14 am hasta las 11:29 am aproximadamente. En todos los casos, una instancia o entidad SBB corresponde a un usuario en el sistema.

5.3.4.1 Comportamiento del Servicio de Interfaz ISC

En la figura 44 puede apreciarse la tasa de creación de entidades SBB del Servicio de Interfaz ISC. Debe notarse que esta figura está relacionada directamente con la figura 39 que corresponde a la de tasa de creación de sesiones entregada por el generador de tráfico, dado que por cada sesión generada por SIPp, una entidad SBB que se encarga de ella es creada.

Como puede observarse, el SLEE creó entidades SBB del Servicio de Interfaz ISC al mismo ritmo que el generador de tráfico SIP solicitaba inicios de sesión. Debido a esto los tiempos de respuesta logrados fueron tan bajos. Una característica importante que revela esta gráfica, es el hecho de que no se presentan picos altos o bajos, lo que indica que el comportamiento del servicio fue bueno y que no se encoló o retrasó el inicio de sesión de manera notoria de ningún usuario. De esta manera, cuando la curva cae al finalizar el minuto 15 aproximadamente, se han creado 150000 entidades SBB del Servicio de Interfaz ISC, una para cada usuario.

21 Este valor corresponde al promedio de los valores adicionales en la duración de las llamadas.

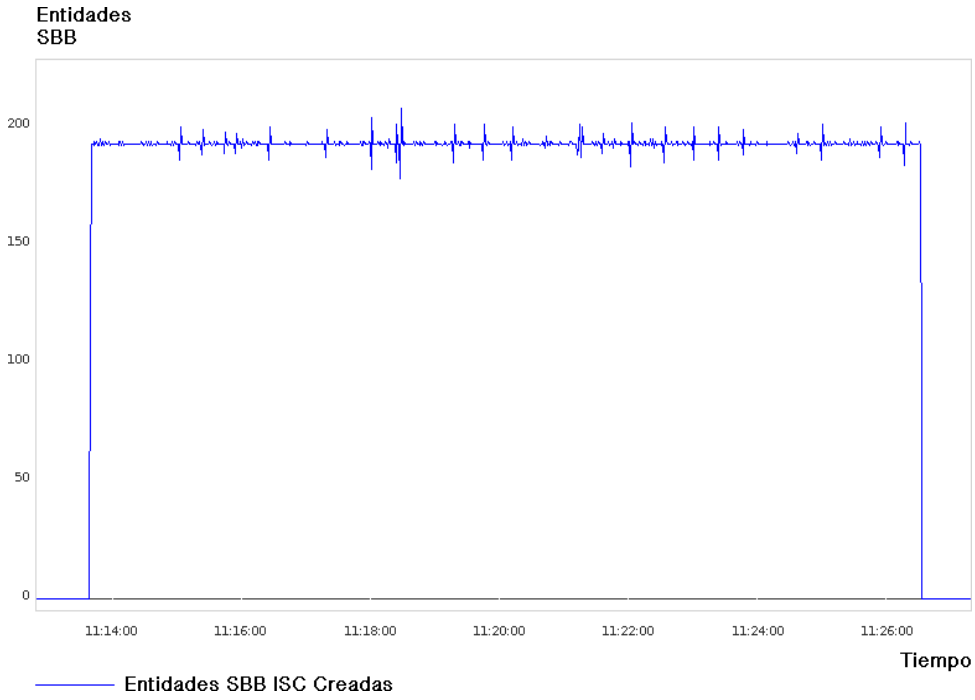

Figura 44. Entidades SBB del Servicio de Interfaz ISC creadas.

Por otra parte, como puede verse en la figura 45, la curva que muestra la cantidad de entidades SBB activas del Servicio de Interfaz ISC tienen 3 fases, la primera donde la cantidad de entidades SBB activas crece de manera lineal, es decir, la cantidad de usuarios que el servicio está atendiendo simultáneamente se está incrementando a un ritmo fijo, para este caso de 194 nuevos usuarios activos por segundo de acuerdo a la gráfica anterior, luego viene una fase donde la cantidad de usuarios simultáneos se estabiliza en torno a un valor entre 36000 y 37000 usuarios, esto debido a que a partir del minuto 3 de la prueba empiezan a terminar su sesión los primeros usuarios, por lo que la tasa de inicios de sesión se iguala con la tasa de terminación de sesiones, impidiendo así que la cantidad total de usuarios simultáneos en el servicio se incremente. En la última fase, ya no son creadas nuevas sesiones de usuario porque ya se han atendido el total de usuarios de la prueba, pero si se continua el proceso de terminación de sesiones para los usuarios que permanecían activos, por lo cual la curva decae hasta 0 cuando ya no queda ningún usuario por atender. Como puede observarse, esta figura corresponde fielmente con los datos de llamadas simultáneas entregados por el generador de tráfico SIPp.

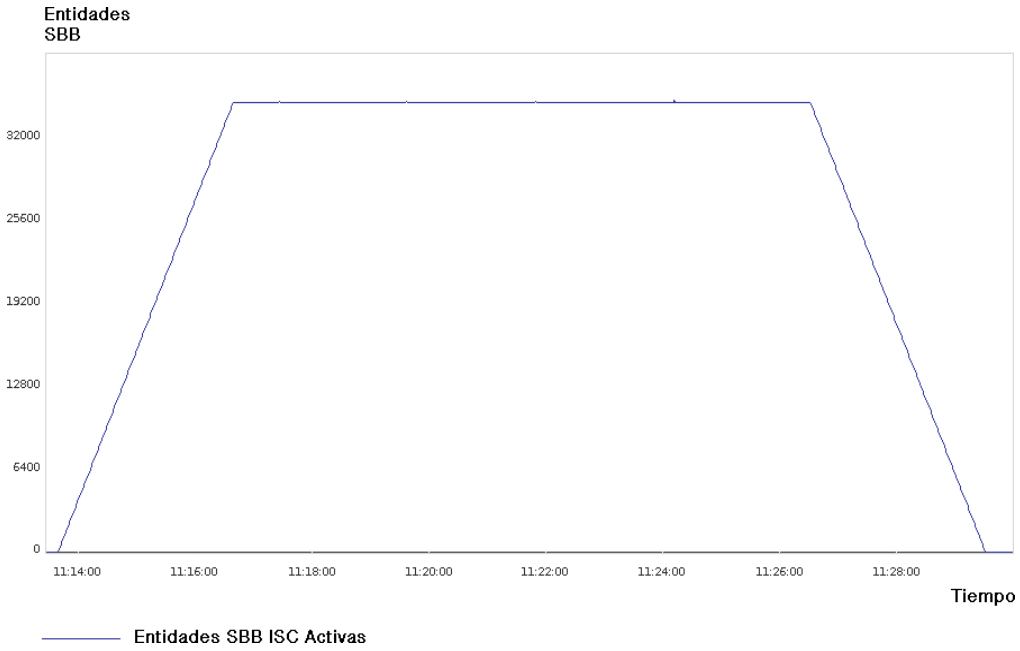

Figura 45. Entidades SBB del Servicio de Interfaz ISC activas.

La tasa de eliminación de entidades SBB del Servicio de Interfaz ISC puede observarse en la figura 46.

Como es posible apreciar, la tasa de eliminación de entidades SBB osciló la mayoría del tiempo en torno a un valor de 194 entidades eliminadas por segundo, que era la misma tasa de creación de sesiones original. Sin embargo, se presentaron picos altos y bajos donde las entidades SBB a eliminar fueron significativamente superiores e inferiores, lo que puede asociarse a causas como: perturbaciones en el SLEE en su funcionamiento interno, al comportamiento del recolector de basura de la maquina virtual de Java o del SLEE, y en mayor medida al hecho de estar monitoreando el SLEE en tiempo real por las herramientas de gestión[22] empleadas para generar las anteriores gráficas.

22 La herramienta de gestión utilizada fue rhino-stats, una herramienta de OpenCloud

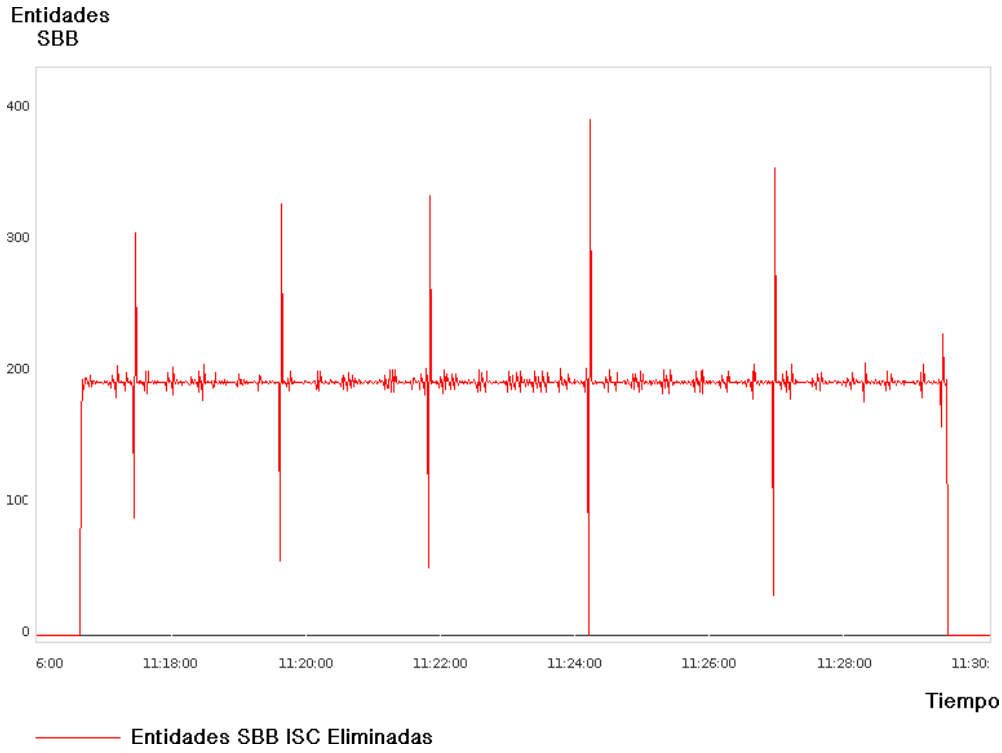

Figura 46. Entidades SBB del Servicio de Interfaz ISC eliminadas.

5.3.4.2 Comportamiento del Servicio de Interfaz Sh

A diferencia del Servicio de Interfaz ISC que solo dependía del generador de tráfico SIP, el Servicio de Interfaz Sh dependía también del funcionamiento del HSS, ya que el tiempo de vida de las entidades SBB de este servicio es tan largo como el tiempo que el HSS demora en procesar una solicitud y retornar el resultado.

En la figura 47 puede apreciarse la tasa de creación de entidades SBB del Servicio de Interfaz Sh. Debe notarse como la forma de esta gráfica está dada por la cantidad de consultas de saldo que deben realizarse sobre el HSS de la red, y que corresponden de forma directa a la cantidad de usuarios que inician sesión en el servicio. Es por esto que las gráficas tienen comportamientos tan similares y permiten verificar como el Servicio de Interfaz Sh funciona de una manera óptima.

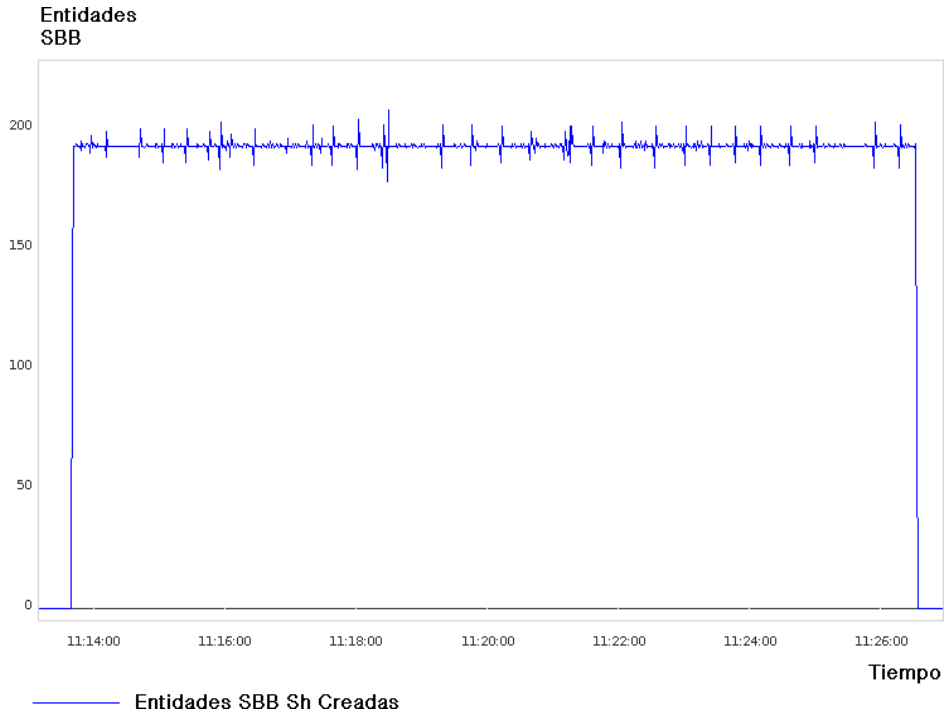

Figura 47. Entidades SBB del Servicio de Interfaz Sh creadas.

Otro aspecto importante que revela la anterior gráfica, es su estabilidad en torno al valor de 194 consultas de saldo por segundo, lo que indica que no se encolaron o represaron grandes volúmenes de transacciones Diameter, que en determinados casos podrían generar picos peligrosos de carga para el HSS.
Debido a que la granularidad de este servicio es muy fina (tal como se puede ver en la figura 33), el tiempo de vida de estas entidades SBB es muy corto, por lo que tan pronto como el SLEE las crea y estas realizan su función, el SLEE procede a eliminarlas, por lo que su tiempo de vida es muy corto.

La cantidad de entidades SBB activas simultáneamente del Servicio de Interfaz Sh es mucho menor comparada con la cantidad de entidades SBB activas del Servicio de Interfaz ISC como era de esperarse, lo cual muestra que efectivamente la cantidad de memoria ocupada por este servicio es en promedio mucho menor, como puede verse en la figura 48. Esto se debe principalmente a que el HSS realiza las tareas de consulta y actualización de datos muy rápidamente.

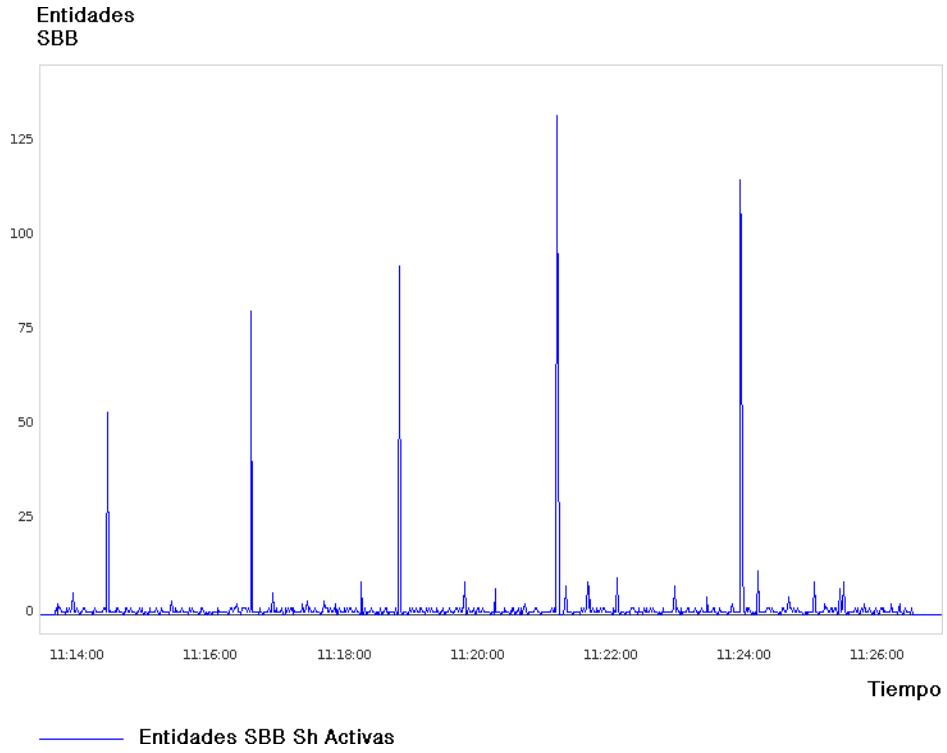

Figura 48. Entidades SBB del Servicio de Interfaz Sh activas.

El comportamiento de las entidades SBB activas del Servicio de Interfaz Sh es un poco menos estable si se compara con la del Servicio de Interfaz ISC principalmente debido a que el tiempo de vida de estas entidades no es fijo, sino que depende totalmente del HSS, por lo cual se presentaron los picos altos. Sin embargo, se puede observar como para la gran mayoría del tiempo que duro la prueba, la cantidad de entidades SBB activas que tenían conexiones con el HSS no fueron superiores a 10, lo cual es un muy buen resultado ya que revela que a pesar de que el servicio de valor agregado atendió a una gran cantidad de usuarios en un corto periodo de tiempo, el HSS de la red no se congestiono o tuvo que atender altas cargas de trabajo.

Por otra parte, si el presente mecanismo de integración se desplegara en un ambiente real, estas gráficas y en particular la que se acaba de presentar, deben constituir una guía fundamental para el dimensionamiento del AS y para imponer límites de trabajo razonables sobre el HSS de la red de acuerdo a políticas impuestas por el operador.

Finalmente en la figura 49 esta la tasa de eliminación de entidades SBB del Servicio de Interfaz Sh, la cual muestra el ritmo al que se eliminaron las entidades SBB que ya no se necesitaban. Aquí se nota la causa de porque la cantidad de entidades SBB activas presento picos altos, la cual son los picos bajos donde no

se eliminaron suficientes entidades SBB seguidas de picos altos donde se eliminaron muchas más entidades. Este comportamiento se debe principalmente a la forma en que funciona el recolector de basura de la maquina virtual de Java y del SLEE, a la vez que a la latencia presentada en el HSS cuando se realizaban consultas o actualizaciones de saldo.

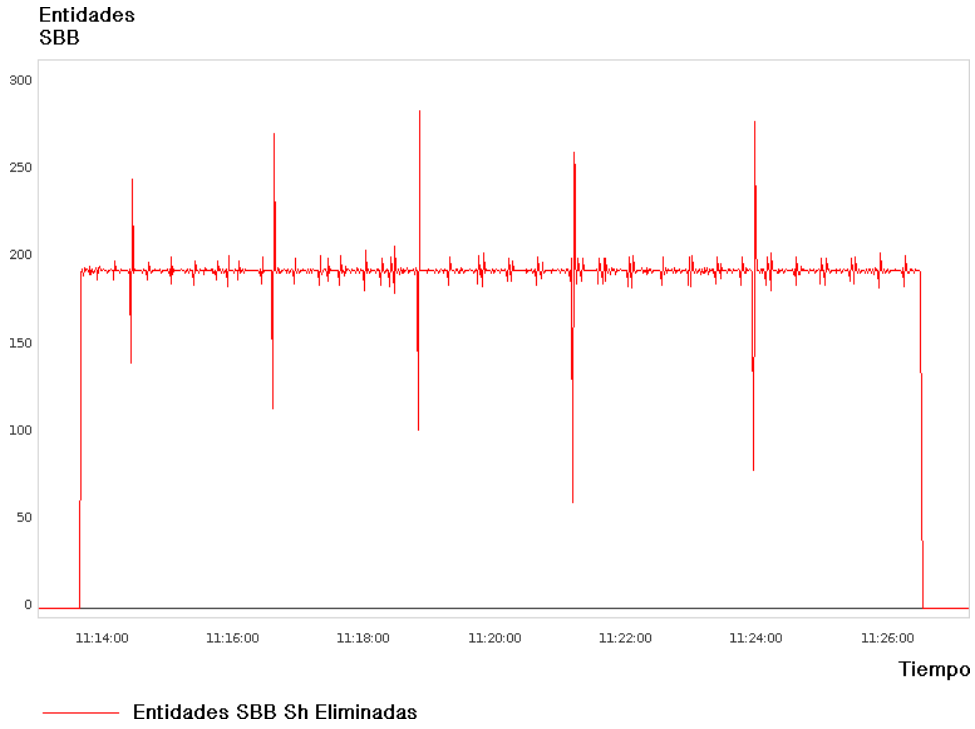

Figura 49. Entidades SBB del Servicio de Interfaz Sh eliminadas.

5.3.5 Comportamiento de los Recursos Hardware

5.3.5.1 Uso de CPU

El uso de la CPU del equipo donde estaba el AS puede apreciarse en la figura 50, donde es destacable que a pesar de la intensidad de la prueba, el procesador funcionó lejos de su límite, por lo que pueden preverse límites de funcionamiento con una cantidad de sesiones simultáneas y tasas de creación de sesiones superiores.

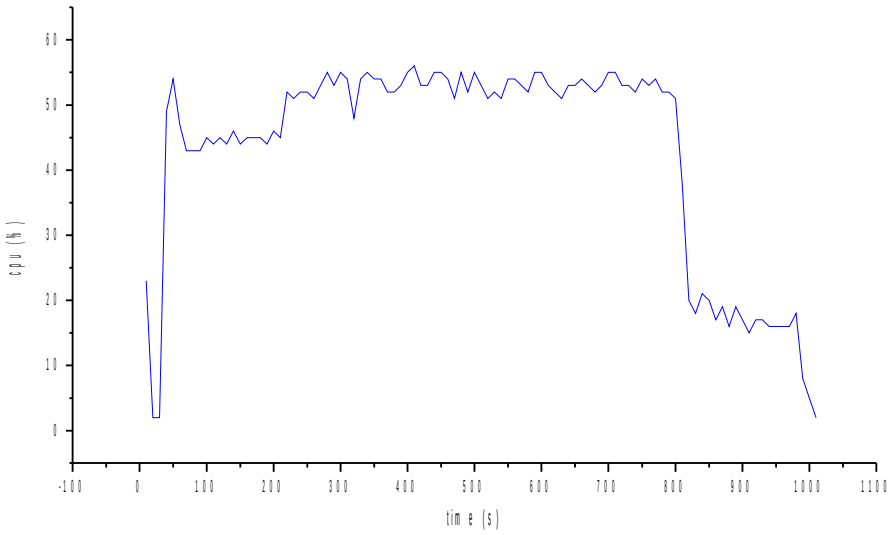

Figura 50. Uso de CPU en el AS.

5.3.5.2 Memoria RAM

El uso de memoria RAM fue más intensivo y estuvo muy cerca de alcanzar el límite disponible en el equipo. Sin embargo, puede apreciarse cómo el uso de este recurso no fue muy variable en el tiempo sino que tendió a mantenerse constante, lo que permite concluir que toda la memoria RAM había sido separada previamente por el SLEE para la piscina de objetos.
Sin duda alguna, este modo de funcionamiento es bastante bueno y ayuda a garantizar tiempos de respuesta bajos. La figura 51 muestra el uso de la memoria RAM durante toda la prueba.

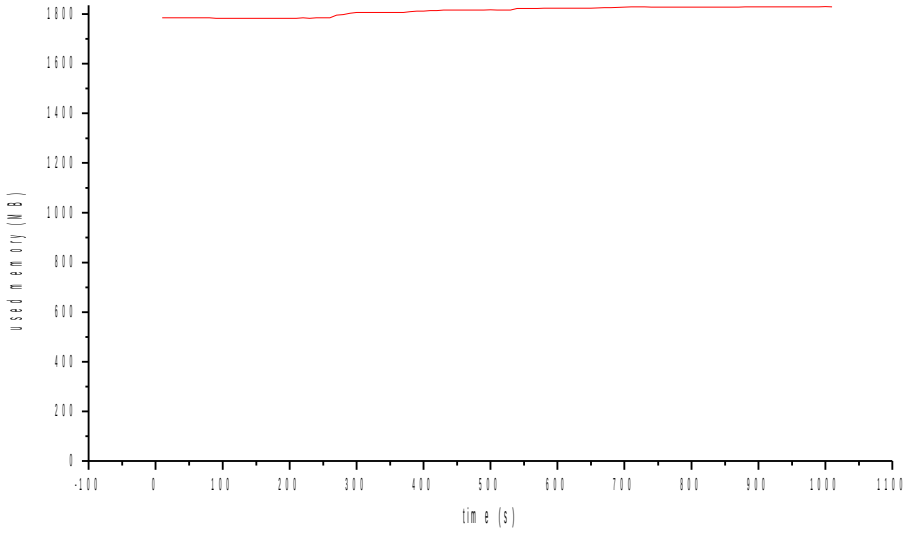

Figura 51. Uso de memoria RAM en el AS.

5.4 Conclusión del Capítulo

Este último capítulo demuestra el buen funcionamiento de cada uno de los Servicios de Interfaces IMS desarrollados, ellos conforman el prototipo del mecanismo de integración entre JAIN SLEE y un entorno IMS. Con el puede darse como cumplido el objetivo número 3 que dice “Validar el mecanismo propuesto mediante la implementación de un caso de estudio de un servicio de telecomunicaciones de valor agregado que se preste sobre IMS mediante JAIN SLEE”. Por lo tanto, el trabajo realizado cumplió con cada uno de los objetivos planteados y resalta la importancia que puede tener la solución planteada para la creación de los servicios de valor agregado de telecomunicaciones, ya que al contar con unos servicios base encargados de los protocolos manejados por las interfaces, permite que el desarrollo de servicios sea mucho mas rápido.

CAPITULO 6. APORTES, CONCLUSIONES Y TRABAJOS FUTUROS

6.1 Aportes

- Un mecanismo de integración entre JAIN SLEE y un entorno IMS bastante elaborado que facilita el desarrollo y despliegue de servicios de telecomunicaciones de valor agregado con un Time to Market pequeño.

- Una base de conocimiento sobre JAIN SLEE que explica de forma sencilla el funcionamiento general de la tecnología, a pesar de que esta es bastante compleja.

- Un conjunto de resultados que permiten confirmar que al menos una de las implementaciones de JAIN SLEE disponibles ofrece realmente un rendimiento excelente.

- Un estudio conciso de las interfaces IMS para un AS con sus respectivos protocolos, sirve como referencia rápida para futuros trabajos.

- Los Servicios de Interfaces IMS ISC y Sh, ellos pueden ser reutilizados por otros servicios de forma tal que pueden desarrollarse y desplegarse mucho más rápido.

6.2 Conclusiones

- JAIN SLEE es una alternativa tecnológica excelente para lograr la convergencia en los servicios gracias a su modelo agnóstico de protocolos. Además, también es la mejor opción para operadores que estén realizando cambios de tecnologías en sus redes.

- A pesar que las comunicaciones asincrónicas emplean unos pocos milisegundos más que las sincrónicas, su rendimiento sigue siendo excelente y existe gran ganancia en el tiempo de desarrollo y en la simpleza de las arquitecturas.

- JAIN SLEE es realmente una muy buena tecnología de alto rendimiento aún en equipos con pocos recursos, por ello es una excelente opción para operadores pequeños y medianos.

- Es vital para un operador reducir el *Time to Market* de sus servicios mediante el aprovisionamiento de facilidades que realicen una mayor abstracción de protocolos propios de la red, ya que estos son bastante complejos.

- El protocolo SIP más sus extensiones distan bastante de ser un protocolo simple en la actualidad.

- La curva de aprendizaje de JAIN SLEE es bastante empinada.

- La arquitectura basada en servicios independientes a pesar de incrementar la cantidad de eventos que maneja el Enrutador de Eventos, sigue ofreciendo un muy buen rendimiento.

- La rigidez del modelo de programación de JAIN SLEE favorece el desarrollo de servicios robustos y evita errores comunes de programación.

- Gran parte de la responsabilidad de la dificultad para el aprendizaje de JAIN SLEE es debida a la ausencia de herramientas que automaticen tareas.

- La abstracción que proveen los RA de SIP y Diameter es buena, sin embargo permiten acceso a características que pueden ser sensibles para un operador, por ejemplo, las cabeceras del RFC 3455.

- Un servidor de aplicaciones que implemente la especificación de JAIN SLEE es una muy buena alternativa para un AS en una red IMS, pues los RA permiten comunicación a través de todas las interfaces.

- Construir SBB con pocos campos CMP debe ser un factor de diseño fundamental para lograr buen rendimiento, debido a que la serialización y des-serialización de estos campos es computacionalmente costosa.

- Un buen diseño e implementación son más importantes que la plataforma o el lenguaje de programación empleado.

- Realizar pruebas a cada componente de un sistema por separado es fundamental para encontrar cuellos de botella.

- La metodología de desarrollo XP es bastante eficiente sobre todo cuando hay fases de aprendizaje de por medio.

- En servicios simples, el tiempo de desarrollo de la lógica de negocio es muy pequeño comparado con el necesario para usar las interfaces IMS del AS.

- Es importante para un operador proveer facilidades críticas para evitar que los desarrolladores de servicios de valor agregado tengan que hacerlo ellos mismos, sobre todo teniendo en cuenta que en general los operadores no tienen personal dedicado a auditar código.

- Las especificaciones de los estándares generalmente evitan situaciones relacionadas con la seguridad, simplemente advierten que deben existir convenios y arreglos entre las partes para garantizar la confianza. Por esto es importante la seguridad y la ocultación de información a todo nivel.

- La implementación de los clientes IMS de los servicios no es trivial.

- Debido a la complejidad del código necesario para utilizar las interfaces IMS de un AS, este sería un componente muy costoso que tendría que ser desarrollado una y otra vez para cada servicio que se quiera desplegar.

- Los operadores necesitan con urgencia proveer facilidades para acceder a sus redes y así llamar la atención de desarrolladores de servicios de valor agregado, ya que la dificultad técnica asociada y el carácter ultra proteccionista de los operadores son los principales obstáculos.

- Con un AS JAIN SLEE es muy fácil construir servicios que mezclan diferentes tecnologías.

- Si un operador quiere evitar el “avance” de servicios sobre Internet, debe con urgencia abrir sus redes para la prestación de servicios que ofrezcan una mejor calidad de servicio que la de sus contrapartes gratuitas.

- Ofrecer acceso al HSS de la red es muy importante para poder enriquecer servicios con base en la información personal de los usuarios como por ejemplo la localización.

- El modelado de servicios de telecomunicaciones es realmente fácil de hacer mediante entornos de ejecución orientados a eventos.

- El tiempo de desarrollo asociado al uso de las interfaces IMS ISC y Sh es demasiado alto comparado con el tiempo de desarrollo de servicios de valor agregado sencillos, y puede inclusive ser mucho mayor.

- La arquitectura basada en servicios independientes es una muy buena alternativa para reducir los tiempos de desarrollo, ya que no solo permite reutilizar código fuente sino también documentos descriptores de despliegue.

- El software libre permite, sin duda alguna, realizar tareas de investigación de una manera más provechosa.

- La complejidad de uso de las capacidades de la red deben ser el principal punto a atacar por parte de los operadores si quieren atraer la atención de desarrolladores de servicios de valor agregado.

6.3 Trabajos Futuros

- Implementar el mecanismo propuesto para las demás interfaces de un AS.

- Migrar el desarrollo realizado hacia otras implementaciones de la especificación de JAIN SLEE, como Mobicents, con el propósito de verificar la portabilidad de los elementos desarrollados.

- Hacer una comparación y análisis profundos entre las tecnologías SIP Servlets y JAIN SLEE desde los puntos de vista del rendimiento y facilidad en el desarrollo.

- Proponer un sistema de sincronización de perfiles entre JAIN SLEE y un HSS de una red IMS, que sirva como una memoria cache para servicios con requerimientos de tiempo real y con acceso a datos del HSS.

- Desarrollar una pasarela de señalización para el protocolo SIP genérico y el protocolo SIP de IMS de tal forma que clientes SIP puedan consumir servicios en una red IMS.

- Proponer un patrón de diseño para JAIN SLEE que sirva de referencia para la creación de servicios de valor agregado.

- Crear un conjunto de servicios reutilizables para otras funcionalidades, por ejemplo para enviar mensajes de texto, mensajes multimedia, etc.

REFERENCIAS BIBLIOGRAFICAS

[1] Sun Microsystems, *"*JAIN™ and Java™ Communications,*" Oracle Sun Developer Network, Marzo, 2004. [En Línea]. Disponible: http://java.sun.com/products/jain/*

[2] Sun Microsystems, *"*The JAIN™ APIs: Intergrated Network APIs For Java™ Platform*," Oracle Sun Developer Network,* Noviembre 2000. [En linea]. Disponible:http://java.sun.com/products/jain/

[3] Von-xchange, *"*Network Solutions Softswitch Vendors Showcase Partnerships, SIP*,"*. [En Línea]. Disponible: http://www.von.com/articles/2001/05/network-solutions-softswitch-vendors-showcase-par.aspx

[4] Open Cloud Limited, *"*Rhino 2.0: Overview and Concepts,*" OpenCloud,* Julio 10, 2008. [En Línea]. Disponible: https://developer.opencloud.com/devportal/download/attachments/8325683/OverviewAndConcepts-2.0.pdf?version=1

[5] Universidad de Otago en Asociación con OpenCloud, Harmonic, *"*Página oficial de JAIN SLEE*," JAINSLEE.ORG*. [En Línea]. Disponible: http://www.jainslee.org/slee/slee.html

[6] Christoph Bröcker, Sven Haiges y Michael Maretzke, *"*Ereignisorientierte Komponenten mit JAIN SLEE,*"*. [En Línea]. Disponible: http://www.sigs.de/publications/js/2005/05/broecker_JS_05_05.pdf.

[7] D. C. Page, D. T. Long, *"*Developing Portable Application With JAIN SLEE,*" OpenCloud,* Octubre, 2007. [En Línea]. Disponible: https://developer.opencloud.com/devportal/download/attachments/13927711/DevelopingPortableApplicationsWithJAINSLEE.pdf?version=2.

[8] Swee Lim, Phelim O'Doherty, David Ferry y David page, *"*JAIN SLEE Tutorial*," JAINSLEE.ORG.* [En Línea]. Disponible: www.jainslee.org/downloads/jainslee-tutorial-04.pdf

[9] Phelim O'Doherty, *"*JAIN SLEE Principles*," Oracle Sun Developer Network , Mayo, 2003.* [En Línea]. Disponible: http://java.sun.com/products/jain/article_slee_principles.html

[10] Michael Maretzke, *"*JAIN SLEE Technology Overview,*" MAKETZKE.COM,* Diciembre, 2005. [En Línea]. Disponible: http://www.maretzke.com/pub/lectures/jslee_overview_2005/index.html

[11] Michael Maretzke, *"*Java Telecommunication Aplication Server Technology Comparison,*" MAKETZKE.COM,* Julio 2008. [En Línea]. Disponible: http://www.maretzke.de/pub/whitepapers/telcoappserver_2008/Positioning_TelcoApplicationServer_Technologies_MiMa_v1.0_20080729.pdf

[12] Página Oficial de Mobicents,The Open Source SLEE and SIP Server. [En Línea]. Disponible: http://www.mobicents.org/index.html.

[13] Open Cloud Limited, *"*A Slee for All Seasons,*"* OpenCloud, Mayo 11, 2007. [En Línea]. Disponible: http://www.opencloud.com/documents/Whitepaper%20A%20SLEE%20for%20all%20Seasons.pdf.

[14]_Paolo Falcarin y Claudio Venezia, *"*Communications Web Services and JAIN SLEE Integration Challenges*," ICI GLOBAL,* Diciembre, 2008. [En Línea]. Disponible: http://www.infosci-journals.com/downloadPDF/pdf/ITJ4488_87d0GBTMON.pdf

[15] Página Oficial de Opencloud, Rhino Application Server. Disponible: http://opencloud.com/products/rhino-application-server/real-time-application-server/.

[16] 3GPP Organizational Partners , *"*Documento 3GPP TS 23218:IP Multimedia (mi) Session Handling; IM Call Model,*" 3GPP™ A Global Initiative,* Junio, 2010. [En Línea]. Disponible: http://www.3gpp.org/ftp/Specs/html-info/23218.htm

[17] 3GPP Organizational Partners, *"*Documento 3GPP TS 23002: *Network Arquitecture,*" *3GPP™ A Global Initiative,* Junio, 2010. [En Línea]. Disponible: http://www.3gpp.org/ftp/Specs/html-info/23002.htm

[18] 3GPP Organizational Partners, *"*Documento 3GPP TS 23228: *IP Multimedia Subsystem (IMS); Stage 2," 3GPP™ A Global Initiative,* Junio, 2010. [En Línea]. Disponible: http://www.3gpp.org/ftp/Specs/html-info/23228.htm

[19]3GPP Organizational Partners, *"*Documento 3GPP TS 24623*:Extensible Markup Language (XML) Configuration Access Protocol (XCAP) over the Ut interface for Manipulating Simulation Services," 3GPP™ A Global Initiative,* Diciembre, 2009. [En Línea]. Disponible: http://www.3gpp.org/ftp/Specs/html-info/24623.htm

[20]3GPP Organizational Partners, *"*Documento 3GPP TS 32260: Telecommunication management; Charging management; IP Multimedia Subsystem (IMS) charging,*" 3GPP™ A Global Initiative,* Abril, 2010. [En Línea]. Disponible: http://www.3gpp.org/ftp/Specs/html-info/32260.htm

[21] Miikka Poikselkä, Georg Mayer, Hisham Khartabil y Aki Niemi, The IMS, Ip Multimedia Concepts and Servicces in the Mobile Domain, 2009.

[22] Travis Russell, Session Initiation Protocol (SIP), Cotrolling Convergent Network, 2008.

[23] 3GPP Organizational Partners, *"*Documento 3GPP TS 29328: IP Multimedia (IM) Subsystem Sh interface; Signalling flows and message contents,*" 3GPP™ A Global Initiative,* Mayo, 2010. [En Línea]. Disponible: http://www.3gpp.org/ftp/specs/html-info/29328.htm

[24] 3GPP Organizational Partners, "Documento 3GPP TS 24147:Conferencing using the IP Multimedia (IM) Core Network (CN) subsystem; Stage 3*," 3GPP™ A Global Initiative,* Diciembre, 2009. [En Línea]. Disponible: http://www.3gpp.org/ftp/Specs/html-info/24147.htm

[25] Mark Wuthnow, Matthew Stafford y Jerry Shih, IMS: A New Model for Blending Aplication, 2010.

[26] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley y E. Schooler, *"*RFC 3261: SIP: Session Initiation Protocol,*" The Internet Engineering Task Force*, Junio, 2002. [En Línea]. Disponible: http://www.ietf.org/rfc/rfc3261.txt

[27] M. Garcia-Martin, E. Henrikson y D. Mills, *"*RFC 3455: Private Header (P-Header) Extensions to the Session Initiation Protocol (SIP) for the 3rd-Generation Partnership Project (3GPP),*" The Internet Engineering Task Force*, Enero, 2003. [En Línea]. Disponible: http://www.ietf.org/rfc/rfc3455.txt

[28] C. Jennings, J. Peterson y M. Watson, *"*RFC 3325: Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks,*" The Internet Engineering Task Force*, Noviembre, 2002. [En Línea]. Disponible: http://www.ietf.org/rfc/rfc3325.txt

[29] W. Marshall, *"*RFC 3313: Private Session Initiation Protocol (SIP) Extensions for Media Authorization,*" The Internet Engineering Task Force*, Enero, 2003. [En Línea]. Disponible: http://www.ietf.org/rfc/rfc3313.txt

[30] D. Willis y B. Hoeneisen, *"*RFC 3327: Session Initiation Protocol (SIP) Extension Header Field for Registering Non-Adjacent Contacts,*" The Internet Engineering Task Force*, Diciembre, 2002. [En Línea]. Disponible: http://www.ietf.org/rfc/rfc3327.txt

[31] *D. Willis y* B. Hoeneisen, *"*RFC 3608: Session Initiation Protocol (SIP) Extension Header Field for Service Route Discovery During Registration," *The Internet Engineering Task Force*, Octubre, 2003. [En Línea]. Disponible: http://www.ietf.org/rfc/rfc3608.txt

[32] P. Calhoun, J. Loughney, E. Guttman, G. Zorn y J. Arkko, *"*RFC 3588: Diameter Base Protocol,*" The Internet Engineering Task Force*, Septiembre, 2003. [En Línea]. Disponible: http://www.ietf.org/rfc/rfc3588.txt

[33] Chrightton C., Long D.T. y Page D.C., *"*JAIN SLEE vs SIP Servlet Which is the best choice for an IMS application server?*," IEEE XPLORE Digital Library*, Diciembre, 2007. [En Linea]. Disponible: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4665289.

[34] Sagueza Group, Aplication Server Choices: Important Considerations in Selecting an Application Server, Agosto, 2008.

[35] G. Camarillo, M. Garcia-Martin, The 3G IP Multimedia Subsystem: Merging the Internet and the Cellular Network*, 2006.*

[36] Página oficial de Ericsson, Ericsson Service Development Studio (SDS) 4.2. Disponible: http://www.ericsson.com/developer/sub/open/technologies/ims_poc/tools/sds_40

[37] Ericsson, *"*Service Development Studio (SDS) 4.1 Tutorial,*" ERICSSON,* Noviembre 2008. [En Linea]. Disponible: http://www.ericsson.com/developer/sub/open/technologies/ims_poc/docs/sds40_tutorial

[38] Página Oficial Open IMS Core. Disponible: http://www.openimscore.org/

[39] Página Oficial Mobicents. Disponible: http://www.mobicents.org/

[40] Página Oficial del Cliente IMS Mercuro. Disponible: http://www.mercuro.net/

[41] Página Oficial del Cliente UCT IMS Disponible: http://uctimsclient.berlios.de/

[42] Página oficial del Cliente IMS Communicator. Disponible: http://imscommunicator.berlios.de/

[43] Página Oficial de Cliente IMS Monster. Disponible: http://www.monster-the-client.org/

[44] Opencloud Limited, *"*Rhino Scalability and Performance on Blade Hardware,*" OpenCloud,* [en Línea]. Disponible: http://opencloud.com/documents/Rhino%20Performance%20on%20Blade.pdf