CRITERIOS TÉCNICOS PARA EL APROVISIONAMIENTO DE VAS EN UNA NGN DENTRO DEL CONTEXTO COLOMBIANO

Anexos

Carlos Felipe Estrada Solano
Julián Andrés Caicedo Muñoz

Director
Mag. Oscar Mauricio Caicedo Rendón

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telemática
Línea de Investigación en Servicios Avanzados de Telecomunicaciones
Popayán, Agosto de 2010

Tabla de Contenido

Lista de Figuras

Lista de Tablas

Anexo A

SDP comerciales y Organismos de Estandarización

El presente documento tienen como objetivo exponer de forma general las Plataformas de Despliegue de Servicios (SDP, Service Delivery Platfom) más significativas en el mercado para la creación, ejecución y despliegue de Servicios de Valor Agregado (VAS, Value Added Services) en Redes de Próxima Generación (NGN, Next Generation Network) haciendo énfasis en la plataforma del proveedor de Equipos de Telecomunicaciones Zhong Xing (ZTE, Zhong Xing Telecommunication Equipment), de la Empresa Municipal de Cali (EMCALI EICE-ESP), y de la arquitectura de aprovisionamiento de servicios de Opencloud, con RHINO como corazón para el desarrollo de servicios. Además, se realiza una descripción de los organismos de estandarización que conforman el núcleo del concepto de las SDP.

A.1 Estructuras base para la SDP

Es este punto se pretende describir de manera general las principales organizaciones de estandarización detrás de la construcción de la arquitectura de las SDP para el aprovisionamiento de VAS en las NGN

A.1.1 OSE

El Grupo Móvil Abierto (OMA, Open Mobile Alliance) está formado por varias empresas, incluyendo importantes operadores móviles, distribuidores de dispositivos e infraestructura de red, compañías de Tecnologías de la Información (IT, Information Technology) y proveedores de contenido y servicios, que tiene como objetivo consolidar en una sola organización todas las actividades de estandarización para el área de habilitadores de servicios móviles, incluyendo la arquitectura e interfaces abiertas e independientemente de las redes y plataformas de bajo nivel [1].

La especificación de un grupo de habilitadores, la definición de componentes estándar y las relaciones existentes entre estos elementos, en conjunto conforman el Ambiente de Servicios de OMA (OSE, OMA Service Enviroment), el cual representa, una arquitectura estándar, flexible, extensible e interoperable, en donde los servicios pueden ser desarrollados, integrados y desplegados con una complejidad menor [2].

La arquitectura OSE (*ver Figura 1*) se encuentra compuesta por los siguientes elementos [2]:

- *Habilitador*: tecnología para el desarrollo, despliegue u operación de un servicio. Especifica una o más interfaces públicas.

- *Implementación de habilitador*: puede ser visto como una plantilla que representa una implementación de cualquier habilitador OMA. Ofrece funciones estandarizadas de un recurso determinado y expone las interfaces de gestión del ciclo de vida que permiten usar las capacidades para la administración de componentes de los habilitadores.

- *Interfaz*: límite común entre dos sistemas asociados.

- *Enlace de interfaz de habilitador*: provee la sintaxis y los protocolos utilizados para tener acceso a los habilitadores de servicio.

- *Recurso*: elemento que representa una capacidad en el dominio de un proveedor de servicios.

- *Ejecutor de políticas*: elemento encargado de suministrar mecanismos de gestión basado en políticas para la protección de recursos.

- *Aplicación*: implementación de un conjunto de funciones que habilitan uno o más servicios o que realizan algún otro tipo de trabajo útil.

- *Ambiente de ejecución o gestor del ciclo de vida de un habilitador*: módulo en el cual se soportan varias funciones que permiten al dominio OSE controlar los habilitadores de servicio. Entre estas funcionalidades encontramos el proceso de monitoreo, el manejo del ciclo de vida del software, algunos soportes del sistema (manejo de hilos, balance de carga, transparencia, etc.), Operación Y Mantenimiento (O&M, Operation & Maintenance), entre otras.

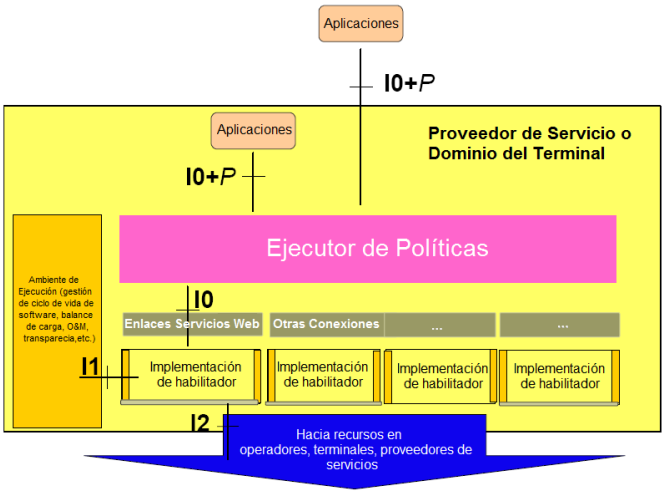


Figura 1 Elementos de la arquitectura OSE
Fuente: OMA[1]

A.1.2 JAIN

La iniciativa para Interfaces de Programación de Aplicaciones (API, Application Programming Interface) de Java para Redes Integradas (JAIN, Java API for Integrated Networks), representa a una comunidad de expertos en comunicaciones que define un conjunto de API Java para migrar las redes e interfaces de comunicaciones propietarias a estándares abiertos, permitiendo un rápido, simple y económico desarrollo, despliegue y mantenimiento de servicios y productos de próxima generación [3].

Dichas especificaciones definidas en la iniciativa JAIN son construidas basándose en tres objetivos principales heredados de la tecnología Java [3]:

- *Portabilidad de servicio*: bajo la filosofía *hecho una vez, corre en cualquier parte*, define un conjunto de interfaces comunes estandarizadas Java que mapean aquellas soluciones que utilizan interfaces propietarias, asegurando una completa portabilidad.

[1] Tomado del documento "*OMA Service Environment*" [2].

- *Independencia de red*: bajo la filosofía *cualquier red*, permite que las aplicaciones sean soportadas y accedidas por cualquier tecnología de red bien sea alambrada, inalámbricas o basada en el protocolo IP.

- *Desarrollo abierto*: bajo la filosofía *por cualquiera*, define las API de comunicaciones de fácil uso permitiendo a diferentes desarrolladores crear nuevas aplicaciones que anteriormente eran de complejo desarrollo y despliegue.

Los anteriores objetivos actualmente se encuentran enmarcados en dos aéreas de desarrollo de especificaciones de API Java [3]:

- *Interfaces de contenedor* que especifican las API del ambiente de ejecución bien sea orientado al dominio de las comunicaciones o al empresarial.

- *Interfaces de aplicación* que especifican las API que permiten el desarrollo de servicios a través de redes de comunicaciones fijas, móviles y de datos basadas en el Protocolo Internet (IP, Internet Protocol).

En la Figura 2 se expone una topología Java típica y recomendada para las redes de comunicaciones, en donde se puede observar la forma como diferentes tecnologías e interfaces Java, tanto propietarias como desarrolladas por la misma comunidad, pueden complementar las API definidas en la iniciativa JAIN para ofrecer una solución mejorada y completa [3].

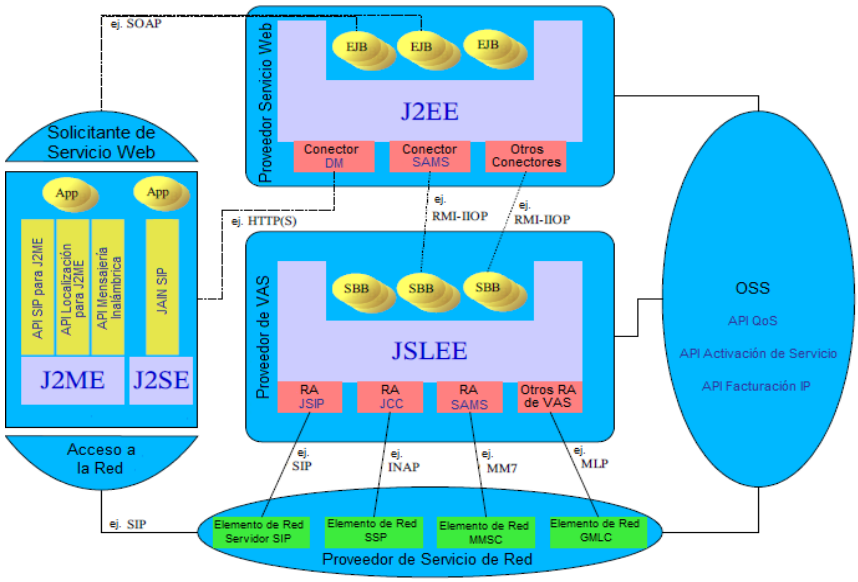


Figura 2 Java en la topología de red de comunicaciones
Fuente: Sun Microsystems[2]

A.1.3 OSA/Parlay

Las especificaciones de Acceso a Servicios Abierto (OSA, Open Service Access) definen una arquitectura que permite a los desarrolladores de aplicaciones implementar servicios que hagan uso de las funcionalidades de red a través de interfaces estandarizadas abiertas [4]. Estas funciones de red se definen en términos de un conjunto de Características de Capacidades de Servicio (SCF,

[2] Tomado del documento "*JAIN and Java in Communications*" [3].

Service Capability Features) que son accesibles desde dichas interfaces y son soportadas por diferentes Servidores de Capacidad de Servicio (SCS, Service Capability Servers) [5].

El objetivo de OSA es proveer una interfaz estándar, extensible y escalable, que permita la inclusión de nuevas funcionalidades y mejoras posteriores en la red, causando un mínimo impacto en las aplicaciones [5].

La arquitectura OSA (*véase Figura 3*) consta de tres partes, los cuales se describen a continuación:

Figura 3 Visión general de OSA
Fuente: ETSI[3]

- *Aplicaciones*: componentes software que suministran servicios a los usuarios finales a través de la utilización de las SCF. Estas aplicaciones son implementadas en uno o más servidores de aplicación.

- *Framework*: proporciona a las aplicaciones mecanismos básicos que les permiten hacer uso de las capacidades de servicio en una red.

- *Servidor(es) de capacidad de servicio*: los SCS contienen las SCF que son ofrecidas a las aplicaciones.

OSA propone un framework que contiene las diferentes interfaces que relacionan las partes que componen la arquitectura de la Figura 4 [6].

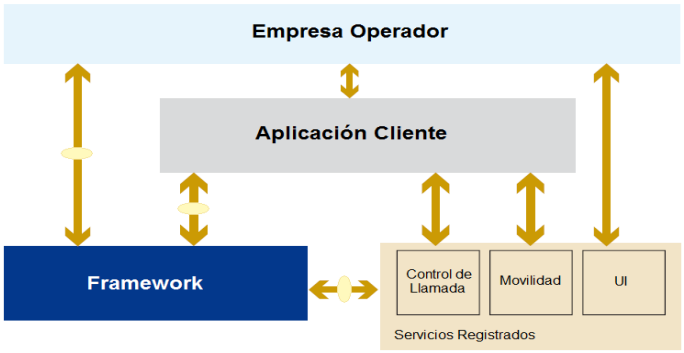


Figura 4 Relaciones del framework de OSA/Parlay
Fuente: ETSI[4]

[3] Tomado de la especificación ETSI TS 123 127 V6.1.0 [5].

- *Interfaz framework-aplicación*: proporciona a las aplicaciones los mecanismos necesarios que les permiten hacer uso de de las capacidades de servicio en la red [4]. Dentro de estos mecanismos podemos encontrar autenticación, autorización, descubrimiento de capacidades, establecimiento de acuerdo de servicio y acceso a las capacidades de servicio [6].

- *Interfaz framewok-capacidad de servicio*: proporciona el soporte multivendedor [4]. El mecanismo básico encontrado es el de registro de las capacidades de servicio de red, incluyendo las generadas recientemente debido a la instalación o actualización de un SCS [6].

Interfaz framework-operador: proporciona al operador los mecanismos básicos que le permiten administrar las cuentas de los clientes y gestionar tanto los perfiles como los contratos de servicio [4]. La función elemental es la de suscribir el servicio en donde se represente el acuerdo contractual entre el operador y el framework, de tal forma que el operador actúa como cliente/suscriptor del servicio y las aplicaciones cliente como usuarios o consumidores del servicio [6].

A.2 SDP Comerciales

A.2.1 Oracle

La industria Oracle es una de las principales empresas que desarrollan sistemas de gestión de base de datos, software de mediador (fusion middleware), planificación de recursos empresariales (ERP, Enterprise Resource Planning) y gestión de relación cliente (CRM, Custumer Relationship Management), entre muchas otras tecnologías [7].

El despliegue rápido de servicios de telecomunicaciones usando estándares abiertos es el concepto que ha venido tomando Oracle para la tecnología de las comunicaciones. La plataforma de despliegue de esta compañía, reúne la flexibilidad y madurez del mediador de fusión y de sus bases de datos, con una fuerte funcionalidad de telecomunicaciones. Esto permite a los operadores un rápido y eficiente despliegue de nuevos servicios de voz, datos y multimedia [7].

Esta SDP consiste de una familia de productos que permiten la creación, ejecución, exposición y gestión de servicios de telecomunicaciones. El servidor de aplicaciones convergentes y el GateKeeper de servicios de comunicaciones son los dos productos clave de esta compañía para el desarrollo de VAS en NGN.

Servidor de aplicaciones de comunicaciones convergentes: este servidor de aplicación, basado en la tecnología de la Edición Empresarial de Java (JEE, Java Enterprise Edition) con soporte para el Protocolo de Inicio de Sesión (SIP, Session Initiation Protocol), para el Subsistema Multimedia IP (IMS, IP Multimedia Subsystem) y construido bajo el paradigma de Arquitectura Orientada a Servicios (SOA, Service Oriented Architecture), es una poderosa herramienta con altos índices de disponibilidad, fiabilidad y desempeño para suministrar una gran variedad de VAS en entornos de nueva generación. La Figura 5 muestra la arquitectura del servidor Oracle para la creación, despliegue y ejecución de servicios convergentes de telecomunicaciones [8].

[4] Tomado de la especificación ETSI ES 204 915-3 V1.1.1 [6].

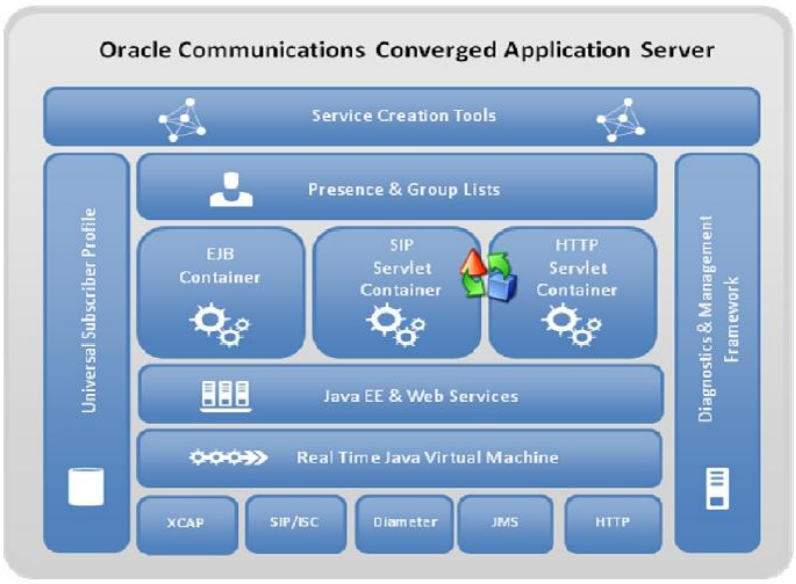


Figura 5 Servidor de aplicaciones de comunicaciones convergentes
Fuente: Oracle[5]

GateKeeper de comunicaciones convergentes: Este componente es una plataforma de exposición de servicios de telecomunicaciones convergentes basado en WEB-SOA, el cual permite a los operadores de red, potencializar sus servicios a través de la vinculación de nuevos actores que enriquezcan el contenido de las diferentes aplicaciones. Este modulo comprende todos los mecanismos de seguridad que requiere un operador al momento de exponer sus capacidades de red a terceros. Es aquí donde procesos como la Calidad de Servicio (QoS, Quality of Service) son ejecutados para cumplir con los altos requerimientos que los VAS demandan. En la Figura 6 puede apreciarse la arquitectura funcional de este componente.

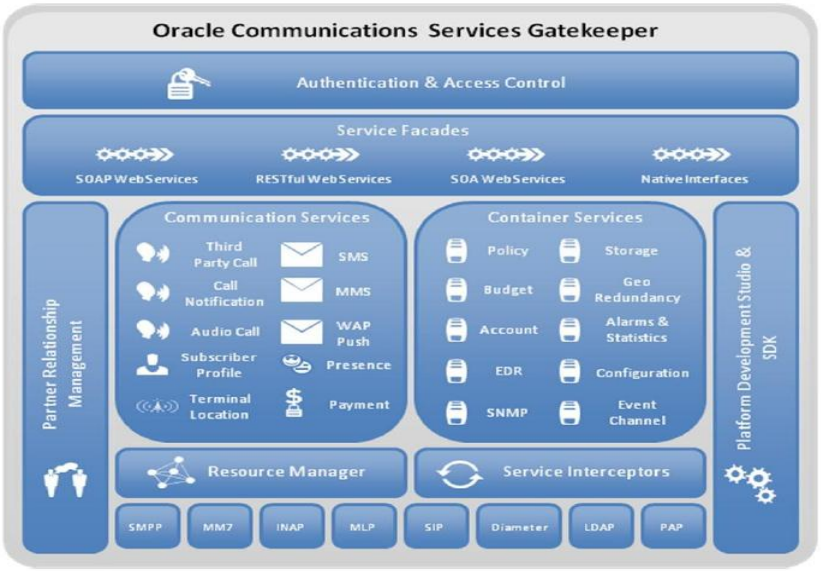


Figura 6 Vista funcional del GateKeeper de servicios de comunicaciones
Fuente: Oracle[6]

[5] Tomado del documento "*Oracle Communications Converged Application Server*" [8].

A.2.2 IBM

La industria IBM es sin duda la más grande y antigua compañía del mundo de la tecnología de la computación que junto con su grupo de soluciones en comunicaciones ha desarrollado un entorno de despliegue de servicios para que los operadores ofrezcan una gran variedad de servicios emergentes. Conocido como ambiente de despliegue de servicios del operador (SPDE, Service Provider Delivery Enviroment), el Framework de esta compañía suministra un completo mecanismo de seguridad, gestión, mantenimiento, creación y despliegue de servicios. Gracias a su orientación SOA, esta arquitectura flexible y escalable, permite una rápida creación y bajos costos de implementación. Entre las características más relevantes encontramos:

- Integración horizontal entre sus funciones y modelos de negocio
- Basado en estándares de IT (SOA, WEB 2.0, etc) y en la industria de la comunicaciones (eTOM, SID, NGOSS, IMS, SIP)
- Plataforma de servicios independiente de la red del operador
- Soporte de múltiples ambientes de creación de servicios
- Abstracción de red y exposición a terceros a través de las API y servicios web
- Bajo acoplamiento entre los componentes software
- Capacidad de adaptación y facilidad de crecimiento frente a la evolución tecnológica y la convergencia

En la Figura 7 puede apreciarse los diferentes módulos funcionales de la arquitectura SPDE de IBM

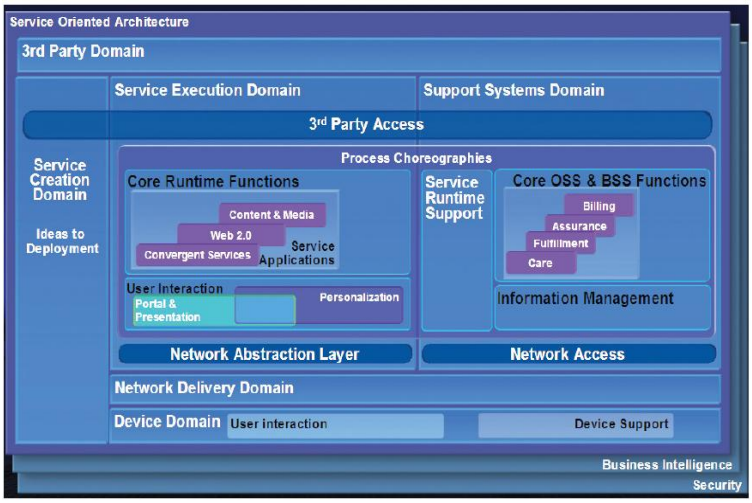


Figura 7 Ambiente de despliegue de servicios del Operador
Fuente: IBM[7]

A.2.3 Microsoft

La corporación Microsoft es una multinacional de la tecnología de la información, con un nivel impresionante de ventas anuales y más de 71 mil empleados en todo el mundo. Microsoft, conocido como uno de los principales impulsadores de la revolución digital, está adoptando un nuevo concepto en el desarrollo y entrega de servicios. Dentro de sus grandes variedades de productos de

[6] Tomado del documento "*Oracle Communications Services Gatekeeper*" [38].
[7] Tomado del documento "*IBM Service Provider Delivery Environment Framework*" [26].

comunicaciones, se destaca el Framework diseñado para la capa de servicio donde es realizada la integración de las infraestructuras legadas de Sistemas de Soporte de Operaciones (OSS, Operations Support Systems)y Sistemas de Soporte de Negocio (BSS, Business Support System). Este entorno de servicios interconectados, denominado Framework de Servicios Conectados (CSF, Connected Services Framework) por la compañía, ofrece un ambiente para la creación, ejecución, despliegue y gestión de servicios. Bajo el paradigma SOA, la SDP de Microsoft, facilita a los operadores de telecomunicaciones la adaptación ante los cambios tecnológicos. La Figura 8 expone los diferentes módulos de la arquitectura CSF

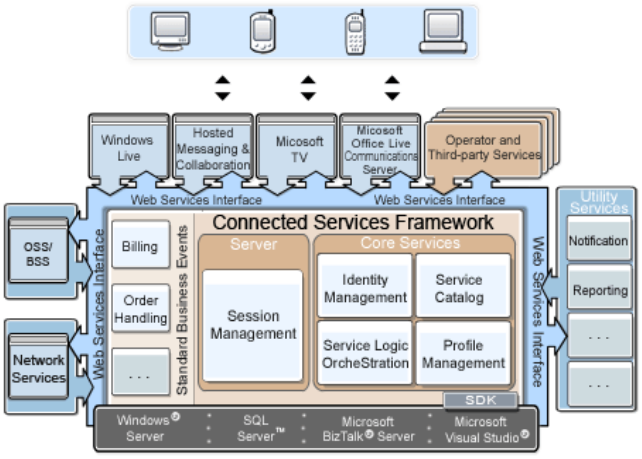


Figura 8 Arquitectura CSF
Fuente: Microsoft[8]

A.2.4 BEA

Esta compañía líder a nivel mundial en el desarrollo de software para infraestructura de comunicaciones y entonarnos empresariales, trae al mercado su plataforma de comunicaciones WebLogic (WLCP, WebLogic Comunications Platforms), la cual habilita la convergencia del mundo telco con el IT. Esta plataforma, Figura 9, consiste de dos productos principales, un servidor SIP y un GateKeeper de red.

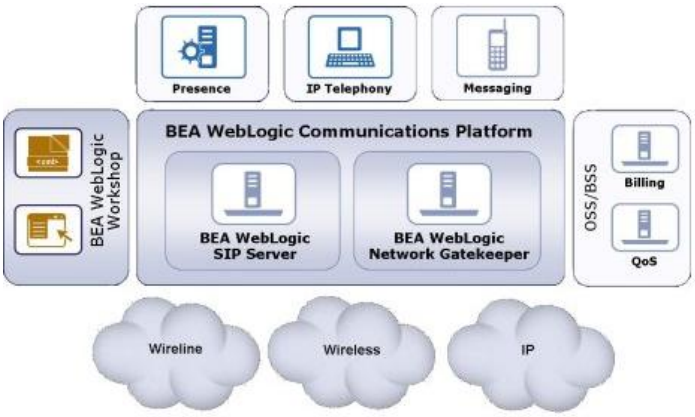


Figura 9 Plataforma de comunicaciones WebLogic
Fuente: BEA[9]

[8] Tomado del documento "*Connected Services Framework 3.0*" [27].
[9] Tomado del documento "*An Introduction to the BEA WebLogic Communications Platform*" [40].

Servidor SIP WebLogic: Este componente es un servidor de aplicación SIP que junto con la tecnología JEE, permite el desarrollo de una gran variedad de servicios. La Figura 10 muestra la arquitectura de este producto.

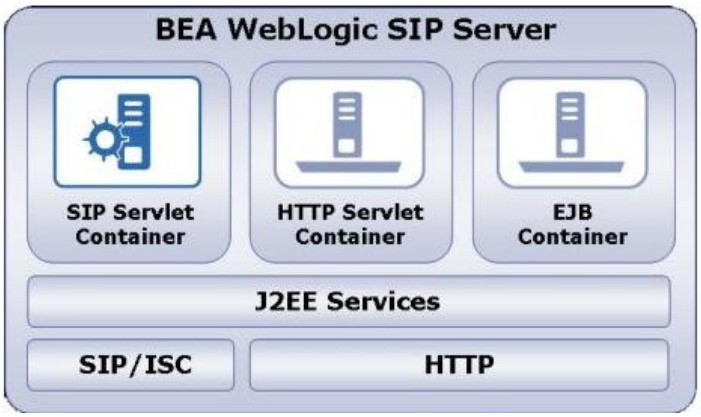


Figura 10 Arquitectura servidor SIP WebLogic
Fuente: BEA[10]

GateKeeper de red: en este componente son aplicadas las políticas de seguridad para la manipulación de las capacidades de red del operador por parte de terceros. La selección de las capacidades se basa en el uso apropiado de las diferentes API que encapsulan las funcionalidades para el desarrollo de aplicaciones externas al dominio de confianza del operador, además, dentro de este componente, está la posibilidad de introducir nuevos componentes (SIP, Interfaces IMS, componentes redes inteligentes, etc) para ampliar el portafolio de VAS.

A.2.5 Ericsson

Esta compañía es una multinacional líder proveedora de equipos para telecomunicaciones y servicios para operadores de redes fijas y móviles. Dentro de sus tres aéreas de trabajo (redes, servicios y multimedia), la compañía ha establecido una SDP multimedia para el despliegue de servicios avanzados en redes móviles de telecomunicaciones. Dentro de las características más importantes se encuentran:

- Soporte de funciones de entrega de servicios
- Orquestación y procesos de negocio
- Soporte de funciones comunes
- Ejecución de servicios
- Control de servicios

La Figura 11 presenta la arquitectura de la SDP de esta compañía

[10] Tomado del documento "*An Introduction to the BEA WebLogic Communications Platform*" [40].

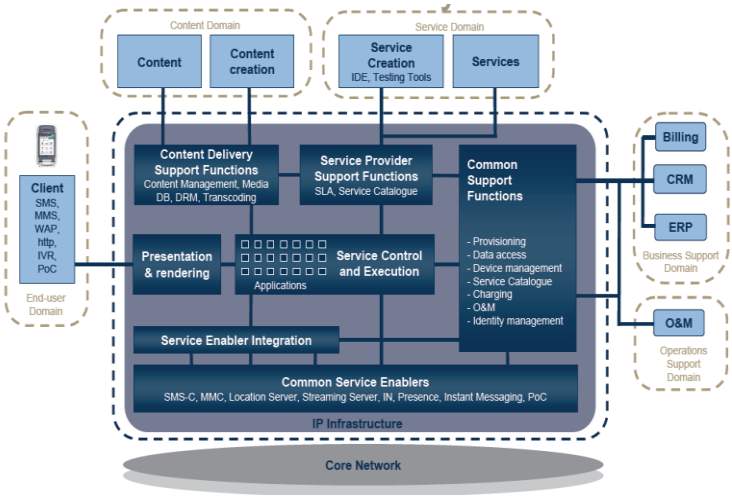


Figura 11 Arquitectura SDP Ericsson
Fuente: Ericsson[11]

A.2.6 Nokia Siemens

Las SDP de esta compañía, conocida como Framework para el despliegue de servicios (SDF, Services Delivery Framework), reúne los conceptos adoptados por el grupo Moriana [9] y expone los siguientes beneficios [10]:

- Implementación de nuevos servicios de manera rápida y a mas bajo costo
- Reducción los costos de integración y desarrollo a través del reúso de funciones comunes entre servicios
- Decremento del costo de gestión de VAS
- Incremento en la calidad de aprovisionamiento de VAS
- Integración transparente con los BSS/OSS

La figura 12 presenta la arquitectura propuesta por la compañía para una rápida creación, ejecución y despliegue de servicios avanzados de comunicaciones, y presenta los siguientes componentes.

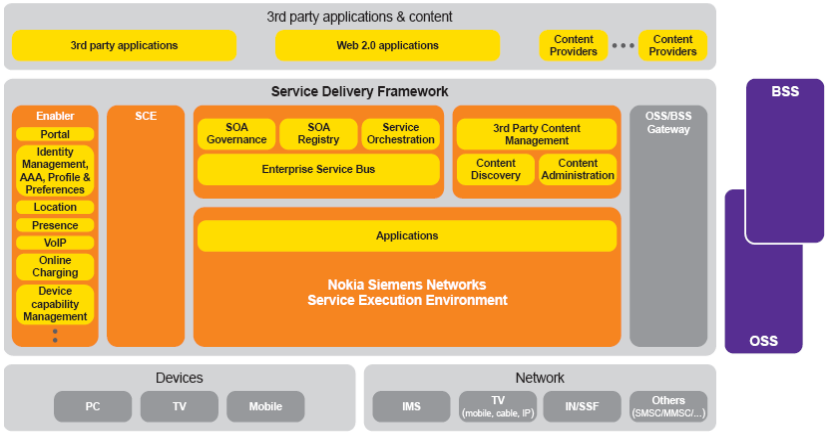


Figura 12 Arquitectura SDP Nokia Siemens Network
Fuente: Nokia Siemens Network[12]

[11] Tomado del documento "*Ericsson Service Selivery Platform*" [28].

Ambiente de ejecución de servicio (SEE): suministra el entorno de ejecución para los servicios y aplicación.

Ambiente de creación de servicio (SCE): provee el conjunto de herramientas para construir todo tipo de servicios sobre la plataforma.

Gateways OSS/BSS: representa los componentes para la integración trasparente ente la SDF y los módulos OSS/BSS del proveedor de servicios de telecomunicaciones.

Ambiente SOA: contiene los módulos necesarios para la creación de servicios basados sobre el concepto SOA.

Gestión y entrega de contenido: proporciona el desarrollo y gestión de contenido por terceros.

Habilitadores: hace referencia a todo el conjunto de funciones comunes a los servicios.

A.2.7 Red Hat

Red Hat es la compañía responsable de la creación y mantenimiento de una distribución del sistema operativo GNU/Linux que lleva el mismo nombre: Red Hat Enterprise Linux, y de otros más. Fedora, así mismo, en el mundo del middleware patrocina jboss.org y distribuye la versión profesional bajo la marca JBoss Enterprise. Algunos años atrás, esta empresa adquirió completamente a Mobicents, una herramienta tecnológica en código abierto para la implementación de JAIN SLEE en un servidor Telco.

Con la combinación de Jboss[13] y el servidor Mobicents, Red Hat ha creado su plataforma de comunicaciones, una herramienta para la creación, ejecución y despliegue de VAS en la industria de las telecomunicaciones. La Figura 13 muestra la arquitectura para la SDP de Red Hat

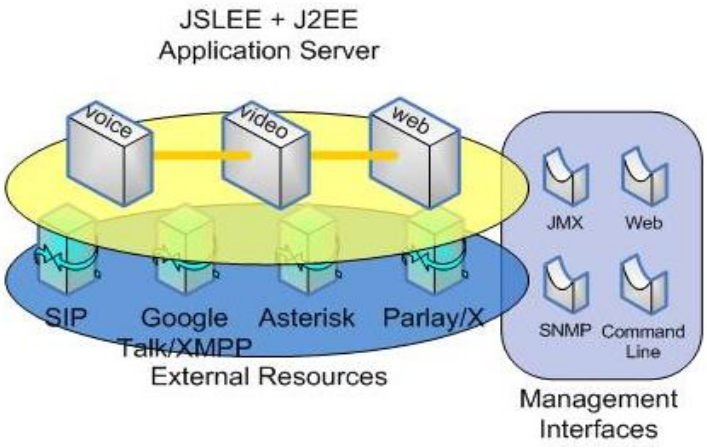


Figura 13 Plataforma de comunicaciones Jboss
Fuente: Mobicents[14]

[12] Tomado del documento "*Service Delivery Framework*" [29].
[13] Servidor de aplicación JEE de código abierto e implementado en Java puro.
[14] Tomado de la página oficial de Mobicents [39].

La ventaja de Mobicents recae en la implementación de la tecnología JAIN SLEE como motor de alto desempeño de una SDP. Las características más importantes de esta tecnología son tratadas en el Anexo B.

A.2.8 ZTE

La arquitectura de despliegue de servicios de la empresa ZTE se adapta al movimiento del mercado y a las nuevas tendencias tecnológicas. La plataforma contiene ambientes de ejecución, de creación de servicio y un nivel de abstracción de red. Algunas de las características más relevantes son mencionadas a continuación:

- Plataforma que adapta tecnologías y servicios (GSM, WCDMA, CDMA2000, IMS, NGN).
- Soporte al modelo de OSA.
- Interfaces abiertas multinivel como las API de Parlay y Servicios Web (WS, Web Services).
- Soporte de herramientas de desarrollo como el Lenguaje de Ejecucion de Proceso de Negocio(BPEL, Business Process Execution Language), SCE para Proveedor de Servicio(SPSCE, Service Provider SCE), SCE para Redes Inteligentes (INSCE, Intelligent Network SCE).
- Soporte de gestión para plataformas como Servicio de Mensajería Corta (SMS, Short Message Service), Servicio de Mensajería Multimedia (MMS, Multimedia Messaging System), Servicio Basado en Ubicación (LBS, Location Based Service), Java, respuesta de voz interactiva (IVR, Interactive, Voice, Response) , etc.

En la Figura 14 se observa la arquitectura funcional de referencia de la plataforma de despliegue de ZTE.

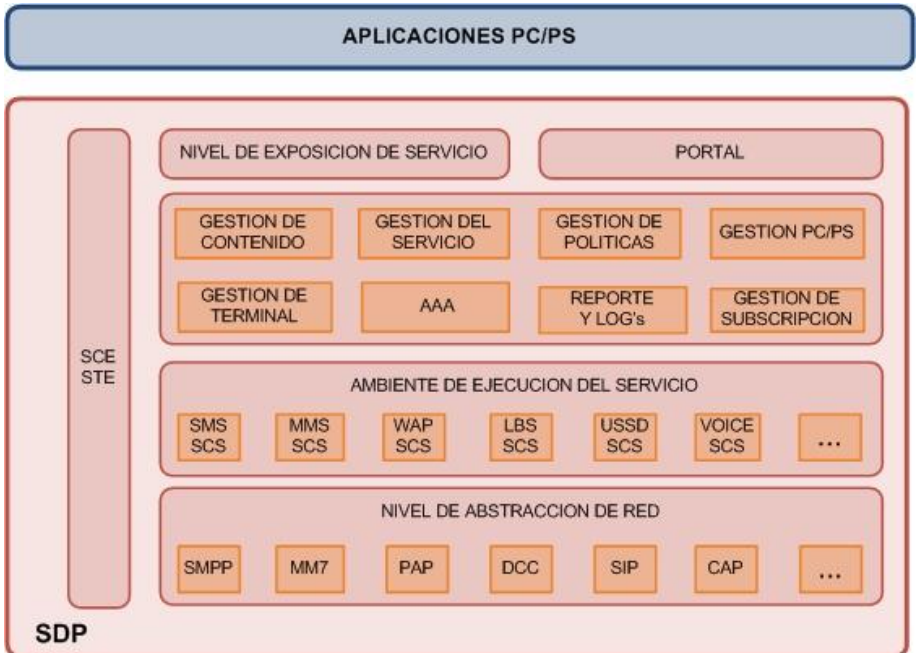


Figura 14 ZTE SDP
Fuente: ZTE[15]

[15] Tomado del documento "*ZXMC SDP Product Description*" [30].

SLEE ZTE

Este componente es el centro de la arquitectura y realiza los procesos de gestión, configuración y composición con servicios básicos. Aquí la lógica es desarrollada y permite a diferentes parámetros como la definición de listas blancas[16] , listan negras[17] e implementación de los acuerdos de servicio ser configurados y ejecutados.

SCE ZTE

Este componente es una interfaz entre la red del operador y el diseñador del servicio. Es una entidad funcional usada para introducir rápidamente servicios de valor agregado en redes inteligentes. Este ambiente de creación, Figura 15, usa componentes básicos con el fin de construir nuevas aplicaciones las cuales serán finalmente probadas en herramientas como el Ambiente de Pruebas de Servicio (STE[18], Service Test Environment) para comprobar el buen funcionamiento del servicio dentro de la red de telecomunicaciones.

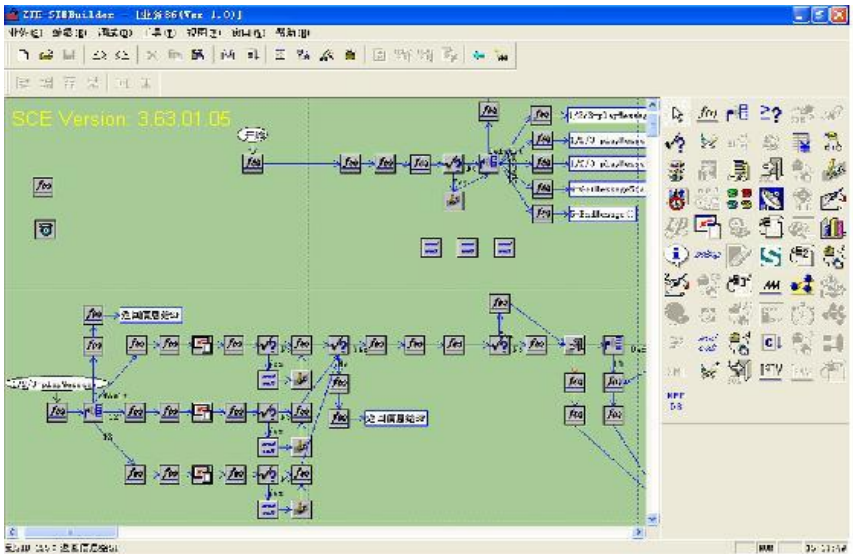


Figura 15 Parlay SCE
Fuente: ZTE[19]

Nivel de abstracción de red

Este es el punto de acceso al centro de la SDP. Es el responsable para que los habilitadores del servicio accedan a los protocolos de red. Este nivel está compuesto de dos partes, (1) el módulo relacionado con interfaces de adaptación de servicio como mensajería corta, mensajería multimedia, localización, medios multimedia, presencia e IMS y (2) el módulo de interfaces de adaptación de voz como INAP, CAMEL, ISC, SIP y MAP.

[16] Listas generadas por el usuario con el fin de configurar los permisos de aceptación de números para recibir llamada. Servicio ONLY de ZTE.
[17] Listas generadas por el usuario con el fin de configurar los permisos de negación de números para no recibir llamada, Servicio ONLY de ZTE.
[18] Ambiente de prueba del servicio proporcionado por ZTE en la SDP con el fin de suministrar al desarrollador un conjunto de herramientas para comprobar la integridad del servicio.
[19] Tomado del documento "*ZXMC SDP Product Description*" [30]**.**

A.2.9 Plataforma Opencloud

La arquitectura de RHINO, distribución comercial de esta compañía, está construida con los componentes más importantes de una plataforma de despliegue de servicios. La capa de creación de servicios, la de ejecución, integración y nivel de atracción conforman su completa estructura. La Figura 16 muestra los componentes de la plataforma.

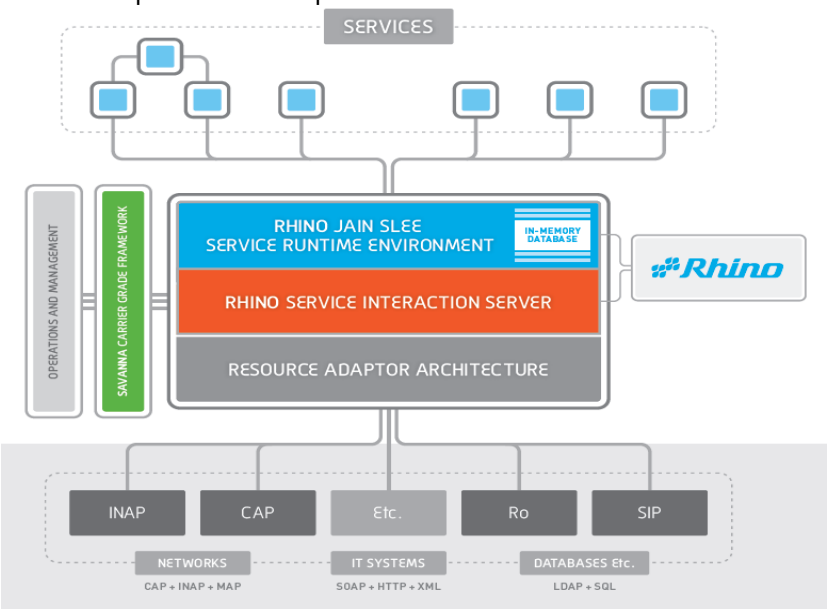


Figura 16 Plataforma de despliegue de RHINO
Fuente: OpenCloud[20]

RHINO JAIN SLEE

Es el corazón de la plataforma RHINO, el cual está constituido por un servidor de aplicaciones en tiempo real, diseñado y optimizado para aplicaciones de comunicaciones conducidas por eventos (*modelo seguido en las aplicaciones del dominio telco*). Este servidor soporta los requisitos de un operador en telecomunicaciones y es compatible totalmente con las especificaciones JAIN SLEE 1.0 (*JSR 22*) y JAIN SLEE 1.1 (*JSR 240*) [11].

Servidor de Interacción de Servicios RHINO

El Servidor de Interacción de Servicios RHINO (RHINO SIS[21], RHINO Service Interaction Server) es una poderosa, flexible y extensible plataforma de integración de servicios manejada por script[22] e interfaces de usuario. Este componente permite al desarrollador componer y gestionar nuevos servicios de redes SS7 e IMS. En la Figura 17 puede verse el contexto de implementación de servidor[23] de aplicaciones [12].

[20] Tomado del documento "*A service delivery platform for next generation telecommunications services*" [12].
[21] Adopta la interacción de servicios para ampliar el concepto IMS SCIM incluyendo la composición de nuevos servicios como IN/SS7, IMS/IP, Web Services.
[22] Conjunto de instrucciones para automatizar tareas.
[23] RHINO SIS corre sobre el servidor de aplicaciones RHINO, RHINO JAIN SLEE.

Figura 17 RHINO SIS
Fuente: RHINO[24]

Este servidor nace con la idea de extender la ventaja del Gestor de interacción de capacidades de Servicio (SCIM, Service Capability Interaction Manage. SCIM fue pensado con la intención de lograr la interacción entre las capacidades de una red IMS dentro de un servidor de aplicación SIP. Fue propuesto y desarrollado por el Proyecto Conjunto de Tercera Generación (3GPP, 3rd Generation Partnership Project) dentro de especificación 6 de su arquitectura de servicios. Por su parte, RHINO SIS acoge esta propuesta y lo extiende a tecnologías de redes inteligentes basadas en conmutación de circuitos. La Tabla 1 resume las principales características de RHINO SIS.

Tabla 1 Características RHINO SIS

CARACTERÍSTICA	DESCRIPCIÓN
Manejo de interacción IN-SIS y SIP-SIS	SIP –SIS: interacción de servicio para servicios IMS IN-SIS: Interacción de servicios para servicios conmutados IN
Combinación de servicios locales y/o externos	Con el RHINO SIS, los servicios pueden ser locales (desplegados en el SIS) o estar contenidos en plataformas externas. La ubicación física de los servicios es tranparente al SIS. Esto permite que nuevos servicios sean combinados y que los servicios contenidos en redes heredadas migren suavemente a la plataforma RHINO.
Gestión y operación manejables	El uso de RHINO SIS permite: Reducir los costos de operación de la red debido a que usa los recurso de red existentes de manera más eficiente y que sean usados de múltiples maneras Permite que nuevos servicios sean implementados como servicios puros JAIN SLEE o como combinaciones de servicios existentes. Esto suministra una mejor gestión de interacción y una migración a estructuras basadas en IMS.

Fuente: RHINO[25]

[24] Tomado del documento "*A service delivery platform for next generation telecommunications services*" [12].

Arquitectura de adaptador de recursos

Este nivel se compone de la arquitectura de Adaptador de Recursos (RA, Resources Adapter) de JAIN SLEE. Esta suministra un conjunto de tecnologías que son introducidas[26] dentro de la plataforma con el fin de inter-operar diferentes sistemas. La Tabla 2 resume las tecnologías soportadas dentro de la plataforma RHINO las cuales son utilizadas para desarrollar servicios de próxima generación [11].

Tabla 2 RA de RHINO

RA	DESCRIPCIÓN
SIP	Protocolo de Inicio de Sesión
ISC	3GPP/Soporte de Extensión
Diameter	Interfaz IMS
MM7	Mensajería MMS
CAP	SS7
INAP	SS7
MAP	SS7
HTPP	Integración Empresarial
SOAP	Servicios Web
JEE	Integración Empresarial
LDAP	Integración de servidor de directorio
JDBC	Integración a base de datos

Fuente: RHINO[27]

Dominio del operador

La plataforma RHINO reúne los requerimientos[28] específicos al operador como desempeño, fiabilidad, disponibilidad, facilidad de crecimiento, contención contra fallas, redundancia, gestión y mantenimiento. Este conjunto de parámetros son tenidos en cuenta con el fin de crear servicios de valor agregado desde la perspectiva de un operador de telecomunicaciones y la unión de tecnologías IT. La Figura 18 muestra la relación entre IT y Telco.

---

[25] Tomado del documento "*Overview And Concepts*" [21].
[26] Arquitectura basada en "plugged-into". introducción de módulos sobre cierta infraestructura, en este caso JAIN SLEE.
[27] Tomado del Documento "*Data Sheet Rhino Core Platform*" [11].
[28] Los requerimientos generalmente aceptados proceden de una serie de documentos de sistemas de conmutación de Bellcore, Investigación de comunicaciones BELL.

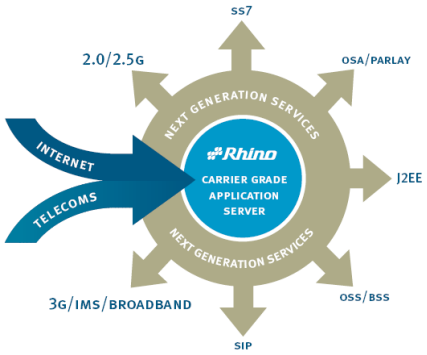


Figura 18 VAS en NGN
Fuente: RHINO[29]

Resumen

Son muchas las SDP que existen el mercado, y muchas maneras de implementarlo. Debido a la no estandarización del concepto, las SDP terminan siendo ofrecidas como un producto especifico a un proveedor para la creación, ejecución, desligue, facturación y gestión de servicios de telecomunicaciones, y no como un modelo general para el aprovisionamiento de VAS en cualquier NGN. Sin embargo, todas estas SDP introducen cuatro componentes clave en la oferta de servicios: Nivel de ejecución, Nivel de creación, Nivel de adaptación de red y acceso a terceros. Todos ellos analizados en detalle dentro de la monografía

A manera de resumen, se presenta en la Tabla 3 una comparación entre las SDP comerciales según las características más importantes de toda plataforma para la creación, ejecución y despliegue de VAS en NGN.

Tabla 3 Comparativa entre servidores de telecomunicaciones utilizados o proporcionados por diferentes compañías

ÍTEM	ORACLE	MICROSOFT	ERICSSON	BEA	IMB	NOKIA SIEMENS	RED HAT
Lenguaje base de programación	Java	.NET	Java	Java	Java	Java	Java
Servidores para el ambiente de ejecución	SIP Servlet	SIP Servlet .NET	SIP Servlet	SIP Servlet	SIP Servlet	Rhino JAIN SLEE jNetX	Mobicents JAIN SLEE
Políticas de Administración	SI	SI	SI	SI	SI	SI	SI
Servidores de preferencias/perfiles	SI	SI	SI	SI	SI	SI	SI
Soporte Parlay	SI	NO	SI	SI	SI	SI	SI
Localización / Presencia	SI	SI	SI	SI	SI	SI	SI

[29] Tomado del Documento "*Data Sheet Rhino Core Platform*" [11].

Herramientas /componente IT	SI	SI	SI	SI	SI	SI	SI
Compatibilidad con IMS	SI	SI	SI	SI	SI	SI	SI
Soporte para redes fijas y móviles	SI	SI	SI	SI	SI	SI	SI
Soporte OSS/BSS	SI	SI	SI	SI	SI	SI	SI
Servidor de ambiente de ejecución basado en modelo de composición y reúso	NO	NO	NO	NO	NO	NO	SI
Servidor de ambiente de ejecución con soporte de más de un protocolo	SIP HTTP	SIP HTTP	SIP HTTP	SIP HTTP	SIP HTTP	SIP SS7 INAP CAP MAP Diameter Otros	SIP SS7 INAP CAP MAP Diameter Otros

Anexo B

Tecnologías dentro de las SDP para la creación, ejecución y despliegue de VAS en NGN

El presente documento pretende abordar las principales tecnologías aplicadas, adoptadas y adaptadas al interior de una SDP para la creación, ejecución y despliegue de servicios de telecomunicaciones. El análisis realizado se enfoca en el contexto de implementación de un servidor de aplicación para ambientes orientados al operador, haciendo énfasis en las principales ventajas y desventajas de estas tecnologías.

B.1 Lenguaje de Procesamiento de Llamada (CPL, Call Processing Language)

Esta tecnología es un lenguaje de scripting que define una manera de gestionar las llamadas salientes y entrantes de una NGN. Fue diseñada para implementar servicios dentro de un servidor SIP Proxy con soporte de protocolos SIP y H.323. Además, es suficientemente poderosa para describir el comportamiento de un gran número de servicios. Su contexto de aplicación esta sobre dominios de no confianza donde usuarios finales a través de script creados por ellos, cambian las características de los servicios desplegados en los respectivos servidores. Esta forma de configuración hace posible una personalización más eficiente y más fácil de implementar. Dentro de sus ventajas se destacan [13]:

- Fácil implementación y configuración debido a alto nivel de abstracción
- Diseñado para manejar la capacidad de control de llamada
- Permite que terceros creen servicios personalizados para sus clientes
- Aplicación en diferentes escenarios: (1) scripts creados y cargados en servidores de aplicación por usuarios finales, (2) scripts creados y desplegados en servidores de aplicación por administradores de red en nombre de sus clientes y (3) creación de los scripts desde una aplicación web la cual se encarga de enviarlos al servidor que aloja la tecnología CPL.
- Lenguaje suficientemente maduro

Aunque CPL sea un lenguaje suficientemente evolucionado, presenta desventajas en relación a al desarrollo de VAS como:

- No origina llamadas hacia dos o más usuarios (llamada tripartita y múltiples sesiones de llamada)
- Solo implementa la capacidad de control de llamada[30]
- Diseñado solo para control y descripción de servicios de telefonía en internet
- Solo para creación, no es un entorno de ejecución
- No tiene un control ante fallos
- No expresa una arquitectura de componentes
- Tecnología solo para establecer sesiones, no para soportar alto tráfico

B.2 Lenguaje de Marcado Extensible en Telefonía (XTML, Extensible Telephony Mak-up Language)

[30] Una de muchas capacidades de red del operador para componer servicios avanzados de telecomunicaciones.

Esta tecnología es un lenguaje de descripción de servicio basado en Lenguaje de Marcado Extensible (XML, eXtensible Mark-up Language) y asociado a un Framework de ejecución de aplicación diseñado para proveer servicios de telecomunicaciones mejorados de próxima generación. Esta tecnología está pensada en adaptar el gran potencial de las de tecnologías de internet a las tecnologías de comunicaciones [13][14]. Modularmente, la tecnología XTML consta de un editor visual, un documento XTML, un servidor XTML, Plug-ins del servidor XTML y una plataforma de implementación.

Editor Visual XTML: es una herramienta de diseño gráfico para crear documentos XTML. Con ella es posible introducir, retirar y configurar bloques de servicio en un diagrama de flujo que representa la totalidad de la lógica de la aplicación. Esto permite una rápida creación y ágil modificación de los componentes que conforman el servicio.

Documento XTML: es un archivo de texto que contiene la descripción[31] del servicio. Estos documentos son generados con los editores visuales o cualquier editor XML y son almacenados en un repositorio los cuales típicamente son accedidos por un servidor web.

Servidor XTML: este componente se encarga de recuperar los documentos XTML almacenados en el repositorio y ejecutar la lógica del servicio descrita en ellos.

Plug-ins: son componentes de software independientes que contienen una lógica específica a fin de extender las capacidades del servidor XTML y soportar nuevas características y lenguajes. Estos componentes son descritos dentro de otro documento XMTL el cual es invocado una vez el servidor ejecute el archivo XMTL que contiene la referencia.

Plataforma de implementación: componente que suministra una interfaz hacia los recursos del operador. Esta plataforma envía los eventos producidos en la solicitud de un servicio a un componente específico.

Los anteriores módulos se estructuran dentro de la arquitectura general. La Figura 19 presenta el diagrama de bloques que la componen.

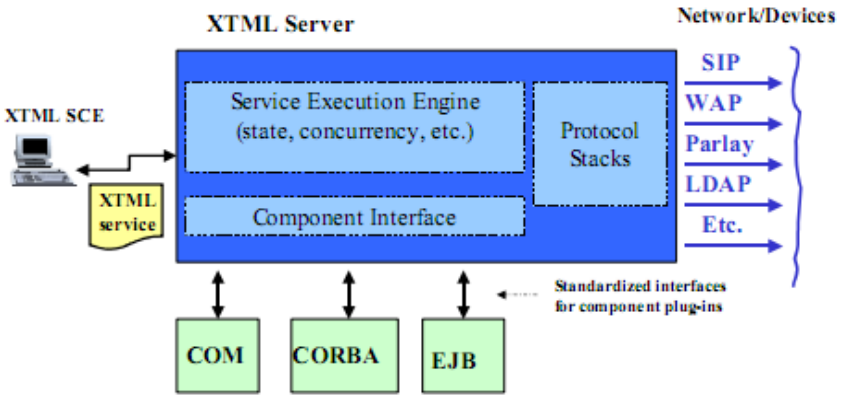


Figura 19 Arquitectura XTML
Fuente: EURESCOM[32]

[31] La descripción se hace usando vocabulario XTML.
[32] Tomado del documento "*Next Generation Networks: the service offering standpoint. Service creation analysis in an NGN context*" [32].

XTML expone las siguientes ventajas y desventajas:

Ventajas:

- Acercamiento a un entorno de ejecución
- Tecnología basada en eventos
- Enrutador de eventos
- Rápida creación de servicio
- Acercamiento a una arquitectura plug and play
- Contiene un entorno gráfico de creación

Desventajas:

- No está optimizado para "carrier-grade"[33]
- Su entorno de ejecución no tiene control ante fallos
- Basado para aplicaciones solo de voz
- Basado en soluciones propietarias
- Poca facilidad de crecimiento y bajo desempeño
- No contiene un propio modelo de arquitectura en clúster

B.3 Lenguaje de marcado extensible de voz (VoiceXML)

Esta tecnología permite al usuario interactuar con un servidor web a través del reconocimiento de la voz. VoiceXML aprovecha las capacidades de un servidor de medios para prestar las funciones de audio en relación a la reproducción del habla. Las aplicaciones que pueden desarrollarse con este tipo de tecnología van desde simples servicios de telefonía hasta formar parte de una de las funciones del servicio de mensajería unificada [15]. Dentro de sus características más importantes se destacan:

- Los documentos XML son accedidos desde cualquier servidor web. Esto brinda una solución de gestión de contenido distribuido
- La Capa de Conexión Segura (SSL, Secure Sockets Layer) puede ser usado entre el navegador de voz y el servidor web haciendo a las transacciones VoiceXML bastante seguras
- El código es procesado en el lado del cliente permitiendo a cualquier usuario manipular la aplicación
- Total nivel de abstracción de gestión de recursos
- Estándar bastante aceptado para el desarrollo de aplicaciones de voz sobre Internet.
- Su lenguaje de alto nivel de abstracción hace posible crear rápidamente aplicaciones sin un aprendizaje exhaustivo
- Flexibilidad de configuración para el usuario
- Manejo de aplicaciones asíncronas

Las anteriores características son soportadas al interior de la arquitectura, Figura 20, para el aprovisionamiento de servicios.

[33] Se le llama al conjunto de características esperadas dentro de un operador telco: alta disponibilidad, redundancia, desempeño, fiabilidad.

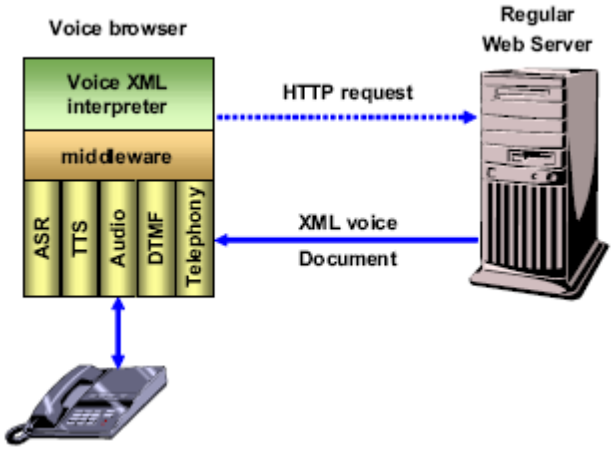


Figura 20 Arquitectura de Voice XML
Fuente: iiWAS2008[34]

Browser de voz: este componente es el encargado de ejecutar e interpretar el archivo XML por medio de diferentes tecnologías (ASR, TTS, DTMF).

Servidor web: componente donde residen las páginas de aplicación. Estas páginas pueden ser archivos VoiceXML, ASP, JSP, o paginas dinámicas PHP para generan archivos VoiceXML.

Aunque esta tecnología sea muy usada, su extensibilidad y dominio de aplicación es limitado, en gran parte por la falta de creación de aplicaciones que demandan más capacidades de red, como por ejemplo, la creación de aplicaciones de conferencia, mensajería, presencia, acceso a contenidos especializados, etc. Por este motivo, tecnologías como CCXML han sido desarrolladas para complementar la capacidad de VoiceXML. Dentro de otras desventajas se pueden listar:

- No contiene un entorno de ejecución de servicio.
- Solo para interpretar archivos XML.
- Basado para aplicaciones de telefonía
- No contienen una arquitectura para el reúso de componentes
- No cumple con características de "carrier-grade"
- No manejo de múltiples sesiones
- No contiene control ante fallos
- El alcance de desarrollo es limitado

B.4 Lenguaje de marcado extensible de control de llamada (CCXML, Call-Control eXtesible Markup Language)

Esta tecnología es un lenguaje de control de llamada que ofrece capacidades de manejo de sesiones integrando un centro de telefonía y un nodo de conferencia multipartita (servidor de medios). Esta tecnología ha sido diseñada con el objetivo de complementar la tecnología VoiceXML. Este complemento, suministra conferencia avanzada, control de sesiones de audio y gestión sobre el intérprete de VoiceXML. Sin duda, es un gran avance en la creación de servicios mejorados, aportando características importantes en el aprovisionamiento de los mismos [13]. Entre las funciones más relevantes se encuentran:

[34] Tomado del documento "*Conceptual Framework for Services Creation/Development Environment in Telecom Domain*" [15].

- Control de múltiples llamadas: realiza un control independiente de las llamadas asignado un enlace de diálogo para cada una de ellas
- Manejo de conferencia: permite que se entallezcan sesiones simultaneas entre diferentes usuarios
- Integración con el interprete VoiceXML para la creación de servicios mejorados
- Gestor de eventos asíncronos

Aunque esta tecnología exponga un panorama en la creación de servicios mejorados, no contiene una completa arquitectura para soportar el aprovisionamiento de VAS. Esto es:

- Diseñado para soportar aplicaciones solo basadas en el establecimiento de llamadas
- No tiene un entorno con características de "carrier-grade"
- No posee una arquitectura optimizada para reúso de componentes
- Crecimiento limitado al momento de crear servicios especializados

B.5 OSA/Parlay

Esta tecnología ha definido una arquitectura[35] que hace posible el inter-funcionamiento entre las aplicaciones IT y las características de red de un operador de telecomunicaciones, a fin de construir servicios convergentes tanto en redes fijas como móviles [13]. Para brindar el acceso a los recursos del operador por terceros, OSA/Parlay definió un conjunto de API que encapsulan las funciones de control y señalización usadas al interior de la infraestructura telco. Estas interfaces gestionan el acceso a capacidades como: el establecimiento de llamadas de voz, gestión de sesiones de datos, gestión de movilidad, mensajería y presencia [16]. Estas interfaces fueron definidas en el Lenguaje de Modelado Unificado (UML, Unified Modeling Language) y el Lenguaje de Definicion de Interfaz (IDL, Interface definition language) dejando la implementación a criterio del desarrollador [15].

Aunque hoy en día, tanto OSA como Parlay sean usados indistintamente, en un principio no fue así. Estas tecnologías eran aplicadas en contextos diferentes pero con objetivos similares. Por un lado, Parlay fue pensado con la intención de permitir a empresas terceras ofrecer sus servicios sobre las redes de telefonía conmutada, mientras que OSA, al mismo tiempo, planteaba una arquitectura similar para el desarrollo de VAS en redes de comunicaciones móviles de tercera generación. Debido a esta similitud de intereses, Parlay y OSA se fusionaron en un único estándar, OSA/Parlay. Las principales características y puntos fuertes son

- Hace transparente la complejidad de la red presente en los protocolos de comunicaciones y sus implementaciones del desarrollo de aplicaciones.
- Permite el desarrollo de aplicaciones a terceros.
- Suministra un acceso controlado y seguro de las capacidades del operador de red a proveedores de servicios terceros.
- Expone varias capacidades de red implementadas por distintos protocolos a l fin de crear nuevas aplicaciones a través de la combinación de ellos.

Las API definidas dentro de las especificaciones de OSA/Parlay son típicamente suministradas sobre Parlay Gateways. Esto hace que muchas de las implementaciones dependan del equipo de red suministrado por el vendedor. Para un desarrollador, este tipo de dependencia en la

[35] Para más información de la arquitectura y del framework de implementación referirse a la sección 2.1.3.5 de la monografía.

implementación de las interfaces, dificulta la creación de diferentes servicios. Además de este inconveniente, esta tecnología presenta las siguientes desventajas:

- El servidor Parlay (Parlay Gateway) no está diseñado para cumplir con los requerimientos de "carrier-grade"
- El Gateway parlay no suministra un entorno de ejecución de aplicación. Esta tarea es asignada a servidores de aplicación externos.
- No contiene un robusto modelo de fallos
- Solo provee un acceso estándar entre en nivel de aplicación del servicio y los niveles de red subyacentes
- Son interfaces, no un lenguaje
- Posee cierta complejidad técnica que no es adecuada para aplicaciones relativamente sencillas
- Son de bajo y medio nivel de abstracción
- Aunque este diseñada para acceso a terceros, su alcance[36] no es tan amplio debido a la necesidad de disponer de una habilidad en telecomunicaciones y de un conocimiento en protocolos específicos (SIP,MAP, INAP)
- Aunque la filosofía de esta tecnología este concentrada en hacer accesible la red del operador a proveedores de servicio externos y a las infraestructuras IT, OSA/Parlay, en la mayoría de los casos, es aplicada para crear servicios dentro del domino del operador más que hacer efectiva la apertura de la red..

B.6 OSA Parlay-X

Esta tecnología define un conjunto de servicios Web para brindar un acceso más abstracto tanto de comunicaciones basadas en SS7 como en redes basadas en SIP. El modelo seguido para el despliegue de estas interfaces se basa en el mecanismo tradicional de publicación de servicios web. Los servicios de Parlay-X son habilitados a través de sus interfaces publicadas en un registro e implementadas sobre el Parlay Gateway. Estas interfaces web no son más que un nivel de abstracción más alto que las API de OSA/Parlay. El mecanismo de autenticación y a autorización sigue el mismo modelo de la arquitectura OSA/Parlay, Figura 21. Dentro de sus principales ventajas se encuentran:

- Alto nivel de abstracción
- Aumento de la capacidad en el de desarrollo de nuevos servicios
- Beneficios del uso de tecnologías web
- Composición, orquestación y reúso de servicios
- Entorno más amplio de desarrolladores
- Orientada a dominios IT

[36] Criterio de usabilidad establecido en la monografía, sección 3.2.3

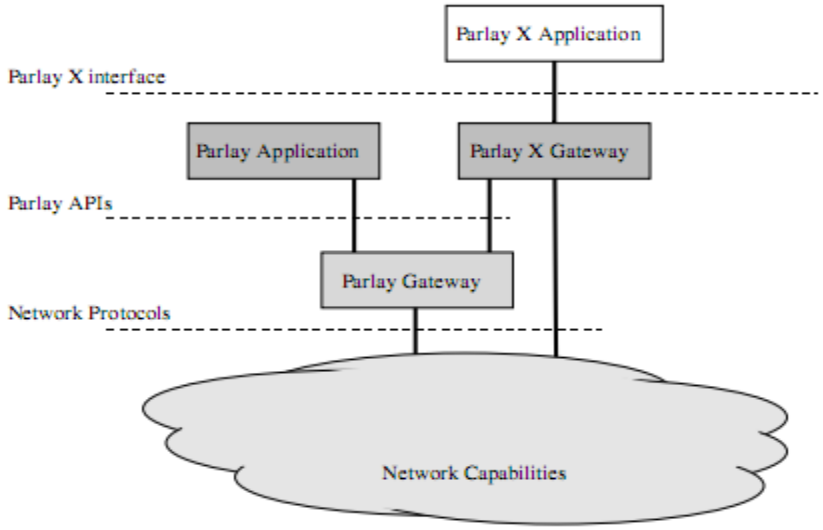


Figura 21 Parlay X y relación con OSA/Parlay
Fuente: Lucent Technologies[37]

Aunque este tipo de tecnologías generen un nivel más alto de abstracción y estén disponibles a una comunidad más amplia de desarrolladores debido a su relación directa en el dominio IT, presenta algunas desventajas en cuanto a la creación, ejecución y despliegue de VAS en NGN. Entre ellas se destacan:

- Soporte solo para el Protocolo de Acceso a Objetos Simple (SOAP, Simple Object Access Protocol), y el Protocolo de Transferencia de Hipertexto (HTTP, Hypertext Transfer Protocol).
- Diseñado principalmente para aplicaciones síncronas
- Su implementación depende las capacidades del Parlay Gateway
- Tecnologías para desarrollar servicios, no para soportar la lógica del servicio
- Parlay Gateway no proporciona un entorno de ejecución de servicio
- Son interfaces, no lenguajes

B.7 Servicios Web

El principal objetivo de una arquitectura de servicios web, es tener una red interoperable basada en el reúso de servicios, adecuada para interactuar con terceros y flexible en el ofrecimiento de servicios por parte de un operador de red o un proveedor de servicios [15].

Estos servicios son usados en la medida que su interfaz de Lenguaje de Descripción de WS (WDSL, Web Services Description Language) sea invocada. Con el fin de encontrar las diferentes WSDL, existen unos repositorios donde son alojadas. Este repositorio es conocido como Integración, Descubrimiento y Descripción Universal (UDDI, Universal Discovery, Descripotion and Integration). Una vez la interfaz ha sido localizada, es posible comunicarse con el servicio requerido. Esta comunicación se da por medio de SOAP[38] soportado sobre el protocolo HTTP. La Figura 22 muestra como esta tecnología se ubica dentro de un ambiente de creación e interactúa con los recursos de red de un operador de telecomunicaciones

[37] Tomado del documento "*Parlay/OSA, From Standars to Reality*" [31].
[38] El lenguaje usado por este protocolo consta de la tecnología XML.

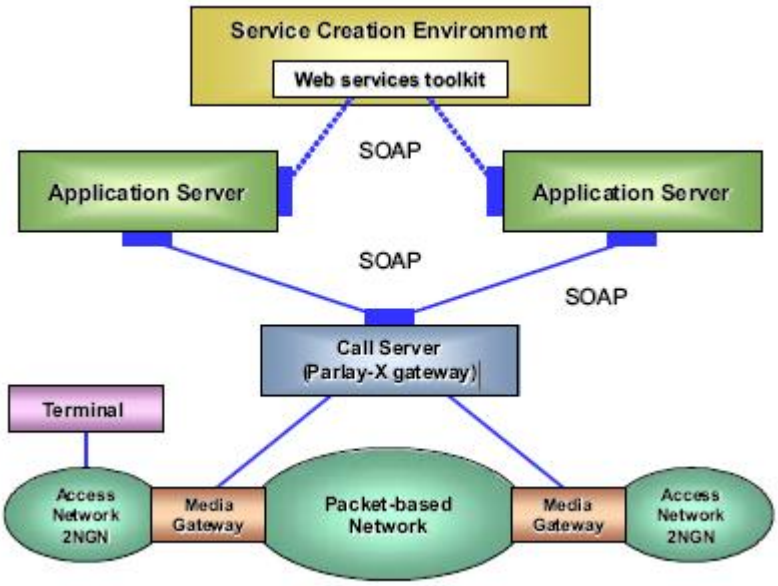


Figura 22 Servicio Web en la arquitectura de referencia del proyecto P1109
Fuente: iiWAS2008[39]

Las implementaciones de los servicios web necesitan que el API correspondiente sea trasladado a WSDL, mientras que los servicios web desplegados en los servidores de aplicación deberán hacer la traducción de mensajes SOAP a las interfaces subyacentes.

Esta tecnología ha tenido un gran éxito dentro del dominio IT. Su gran expansión se debe a la manera precisa de adaptación a paradigmas como SOA. Aunque SOA puede ser implementado sobre diferentes plataformas como CORBA, UDDI y arquitecturas tradicionales Cliente/servidor, la tecnología de servicios web, ofrece el ambiente de creación más competente para implementar este paradigma [17]. La aplicación directa de los servicios Web se ve reflejada sobre servidores de aplicación JEE

Los operadores telco hacen enormes esfuerzos para aprovechar las ventajas de SOA dentro de sus modelos de negocio. Desarrollos como SOAP RA en entornos JAIN SLEE [18], hacen posible que la tecnología de servicios web pueda ser usada para la creación de VAS en NGN. Sus principales ventajas recaen en:

- Alto novel de abstracción
- Cumplimiento del paradigma SOA
- Conectividad a servidores terceros
- Disponibilidad de terceros para crear nuevos servicios
- Rápida creación de nuevas aplicaciones

Aunque con esta tecnología sea posible la creación de nuevas aplicaciones a través del reúso, composición y orquestación de servicios, no es factible una aplicación directa al dominio Telco para el desarrollo de VAS. Entre sus desventajas se pueden destacar:

[39] Tomado del documento "*Conceptual Framework for Services Creation/Development Environment in Telecom Domain*" [15].

- Desarrollada para soportar aplicaciones síncronas
- Soporte solo del protocolo SOAP/HTTP
- Solo para crear servicios, no para contener la lógica de ellos.
- No maneja tolerancia a fallos ni alta disponibilidad

B.8 JEE

Esta tecnología ampliamente difundida en el entorno IT introduce un modelo de programación simplificado en el cual se desarrolla una amplia gama de aplicaciones. Dentro de su modelo se destacan los componentes Servlet/Paginas Java (JSP, JavaServer Pages) y de Negocio Empresarial (EJB, Enterprise Java Beans)

Servlet/JSP: este componente es usado ampliamente para aplicaciones web. El elemento JSP realiza la interacción de usuario con en el nivel de presentación definido dentro de la arquitectura JEE [19], mientras que el Servlet, implementa parte de la lógica de la aplicación y gestión la navegación del usuario.

EJB: este elemento proporciona una manera estándar de implementar la lógica del servicio en sistemas "back-end"[40]. El estándar EJB provee las capacidades para resolver los retos típicos de aplicaciones empresariales, entre ellos: mapeo de objetos tradicionales e integración transaccional de sistemas distribuidos

Esta tecnología presenta enormes beneficios, su difusión comercial es bastante amplia y muchos servidores de aplicación la implementan. Sin embargo, su arquitectura y capacidades están orientadas a cumplir con los requisitos de aplicaciones empresariales, es decir, JEE esta optimizado para dominios IT. Por esta razón, JEE no puede ser aplicada como un servidor telco. Los esfuerzos para adaptar los grandes beneficios de esta tecnología, recaen en extender su alcance a aplicaciones SIP basadas en contenedores SIP servelt. Estas modificaciones, fuera del alcance del estándar, producen soluciones de carácter propietario. Esto implica perdida de la portabilidad de las aplicaciones.

En conclusión, este modelo brinda portabilidad, seguridad y productividad en la creación y ejecución de aplicaciones, pero no es adecuada para entornos telco debido que no está optimizada ni diseñada para cumplir con los requerimientos que este contexto demanda.

B.9 SIP Servlets

Esta tecnología es una extensión del modelo Servlet empresarial y diseñado para simplificar el desarrollo de aplicaciones basas en SIP [20] Este modelo de programación permite que las aplicaciones estén alojadas en un contenedor de servlets y respondan a solicitudes SIP de un usuario o nodo especifico de red que maneje este protocolo [15]. Para ampliar el alcance y desarrollo de nuevos servicios, usualmente se integra con HTTP servlet con el fin de lograr operabilidad con servidores basados en JEE [21]. La gran ventaja de esta tecnología es su diseño para soportar aplicaciones asíncronas, principal característica de aplicaciones en dominios telco. Otra gran ventaja, es su amplia difusión en las SDP de grande compañías. Oracle, Microsoft, Ericsson, BEA e IBM han optado por implementar este paradigma como corazón de su plataforma para la creación, despliegue y ejecución. Sin embargo, SIP Servlet no es la tecnología más adecuada para implementarse en una NGN de un operador telco, principalmente por no estar

[40] Hace referencia al estado final de un proceso.

optimizada a cumplir con los requerimientos como alta disponibilidad, desempeño, fiabilidad y facilidad de crecimiento

La Figura 23 representa la arquitectura definida para esta tecnología

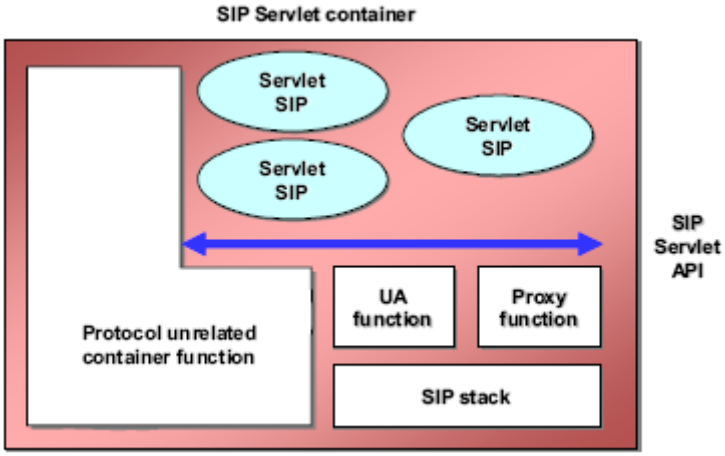


Figura 23 Módulos de SIP Servlet
Fuente: iiWAS2008[41]

Sin duda, SIP Servlet parece atractiva a los desarrolladores por la similitud con el mecanismo de programación HTTP Servlet (basado en solicitud/respuesta). Sin embargo, aunque sintácticamente sean parecidos, su significado semántico es completamente diferente. El modelo de solicitud/respuesta es síncrono, mientras que el protocolo de comunicaciones SIP es asíncrono, haciendo que la tecnología SIP servlet también lo sea [21]. Por tal razón, los desarrolladores de aplicaciones telco y empresariales deben diseñar y codificar según el paradigma presentado. En este caso, para crear servicios en dominios telco, es necesario que la creación de aplicaciones este basada en la naturaleza asíncrona[42] de la comunicaciones. Aunque SIP servelt facilite la creación de aplicaciones basadas en SIP y brinde un entorno de ejecución apropiado para el despliegue de este tipo de aplicaciones, presenta las siguientes limitaciones [21]:

- Las versiones 1.0 y 1.1 de esta tecnología no definen un control de concurrencia, gestión de estado o modelo de fallas. Este tipo de funciones se pueden encontrar frecuentemente en soluciones propietarias.SIP Servlet soporta solo aplicaciones basadas en el protocolo SIP, por lo que componer y dar soporte a otros protocolos no es posible. Un ejemplo claro, es el no soporte para Diameter, uno protocolo definido para lo que hoy es quizás la NGN con más futuro, IMS. Esto quiere decir, que no se pueden crear aplicaciones prepago[43] SIP dentro del estándar Sip servlet sin recurrir a extensiones propietarias para lograrlo.
- Al no tener mecanismos de control para evitar fallos de Hardware y software en arquitecturas de clúster, los desarrolladores deben componer estructuras como servicios complejos los cuales no son suministrados por la misma tecnología. Debido a esto, el desarrollo no alcanza la misma robustez, disponibilidad y poder de recuperación que un sistema telco demanda en sus equipos de red
- No ofrece interfaces de Operación Administración y Gestion (OA&M, Operation Administration & Maintenance)

[41] Tomado del documento "*Conceptual Framework for Services Creation/Development Environment in Telecom Domain*" [15].
[42] Este paradigma asíncrono es seguido en el desarrollo del servicio CRBT descrito dentro del caso de estudio.
[43] Este sistema es el mecanismo de pago preferido por el 60% de las telecomunicaciones en el mundo.

- Necesita de extensiones propietarias
- No maneja tareas de programación complejas
- Su arquitectura no brinda reúso
- No suministra control de concurrencia
- No existe soporte para redes SS7 por lo que es necesario de otras tecnologías para realizar la integración de estas redes con NGN
- La unión de SIP Servlet con tecnología JEE recae en soluciones propietarias de forma tal que es necesario de cambios importantes en la arquitectura.
- No utiliza técnicas orientadas a objetos para la construcción de servicios

B.10 JAIN SLEE

Tecnología de la iniciativa JAIN diseñada específicamente para permitir el desarrollo de aplicaciones que reúnan los más estrictos requerimientos del las aplicaciones de comunicaciones. JAIN SLEE permite un fácil crecimiento, alta disponibilidad, gestión para el control del estado de los servicios y un entono de ejecución lo suficientemente robusto y con altos niveles de desempeño [21].

JAIN SLEE propone un modelo de programación estándar que puede ser usado por una gran comunidad de desarrolladores JAVA. Este modelo simplifica el trabajo en la creación de aplicaciones eliminado los errores de programación comunes y asegurando que los servicios sean desplegados con rapidez [22].

La Figura 24 representa el modelo de referencia por capas de la especificación JAIN SLEE

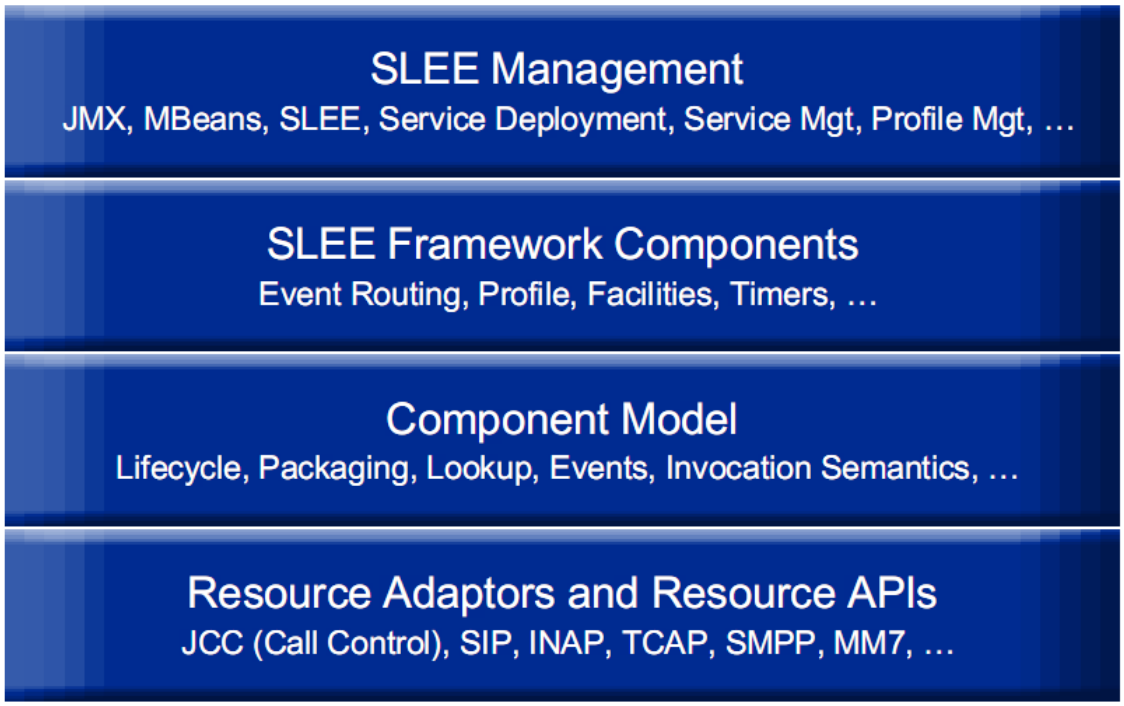


Figura 24 Modelo en capas del sistema JAIN SLEE
Fuente: JAIN SLEE.org[44]

El punto central de modelo de referencia es el nivel de componente, el cual especifica cómo la lógica del servicio es construida, la manera de empaquetar los servicios y la manera de recibir y

[44] Tomado del documento "*JAIN SLEE Tutorial. Introducing JAIN SLEE*" [33].

ejecutar los eventos externos. El nivel de gestión[45] representa el mecanismo por el cual un administrador manipula el SLEE y una interfaz para que los desarrolladores definan los datos necesarios para un servicio en particular. El nivel de framework SLEE, introduce las capacidades necesarias para que servicios creados por diferentes desarrolladores puedan interactuar. Finalmente, los adaptadores de recursos son responsables de la comunicación con un recurso de red en particular externo a JAIN SLEE. Esta comunicación se hace mediante la implementación del API adecuada para manipular dicho recurso. Dentro de otras ventajas[46] se tienen [23]:

- Portabilidad del servicio: permite que los componentes de la aplicación sean desarrollados y desplegados en plataforma compatibles con JAIN SLEE de diferentes vendedores sin necesidad de realizar una nueva compilación y modificaciones de código.
- Robustez: el modelo de programación de esta tecnología hace posible eliminar muchos errores comunes al momento de la codificación. Esto se debe al uso del fuerte tipado[47] y a la adopción de un modelo donde el SLEE tiene percepción de todas las llamadas o sesiones relacionadas al estado de la aplicación.
- Fiabilidad: dentro del modelo de programación existe una semántica bien definida para condiciones de fallos
- Modelo de eventos dinámico, flexible y consistente: un modelo que en tiempo de ejecución toma decisiones de enrutamiento de peticiones a nuevos componentes de servicio en donde cada uno de los enlaces de comunicación puede ser configurados como una transacción asíncrona.
- Arquitectura de componentes orientada a objetos: el modelo de componentes de esta tecnología estructura la lógica de la aplicación como una colección de componentes orientados a objetos reusables a fin de componer servicios más avanzados. Este modelo también define una especie de contrato para la comunicación entre los componentes.
- Desarrollo de aplicaciones simple: esta tecnología hace posible que el desarrollador se enfoque en la creación de la lógica de aplicación sin la necesidad de entender las diferentes acciones de bajo nivel que corren tras la ejecución del servicio.
- Independencia de red: cualquier aplicación desarrollada con esta tecnología es independiente de cualquier protocolo, API o topología de red. Esto se consigue gracias a la arquitectura de adaptadores de recursos.
- Soporte de servicio web y voz enriquecida: gracias a la complementariedad con tecnologías como JEE, es posible crear gran variedad de servicios avanzados de telecomunicaciones. La convergencia de dominios IT con dominios telco es más realizable con la integración de JSLEE y JEE.

Quizás la única gran desventaja hasta el momento es la implementación relativamente nueva frente a otras tecnologías como OSA/Parlay. Aunque estas dos tecnologías hayan surgido en el mismo tiempo, su aplicación en especificaciones difiere. En este sentido, JSLEE es joven y su paradigma de programación es diferente al adoptado extensamente sobre aplicaciones JEE y de la Edición Estándar de Java (JSE, Java Estándar Edition). Actualmente, JSLEE solo tiene dos especificaciones, siendo el jsr 240 su última actualización. En un inicio, la primera versión[48] de esta tecnología, tenía problemas en cuanto a la portabilidad entre sus adaptadores de recursos. Sin embargo, en la segunda y última[49] especificación, este problema es resuelto. La actual arquitectura de adaptación de recursos permite que cualquier adaptador sea completamente compatible con

[45] La gestión incluye tareas como subscripción, instalación y manejo del sistema del ciclo de vida del servicio, entre otras.
[46] Para más información detallada referirse a la especificación JAIN SLEE 1.1 [34].
[47] Característica de un lenguaje de programación para controlar que no sea violado los tipos de datos al tratar de hacer cualquier operación dentro del código.
[48] Jsr 22, especificación JAIN SLEE 1.0.
[49] Jsr 240, especificación JAIN SLEE 1.1.

cualquier implementación de la especificación 1.1. Solo pequeños cambios en los permisos de seguridad deben hacerse para desplegar sin problemas estas unidades.

A primera vista, JAIN SLEE solo tiene en contra su reciente surgimiento como tecnología aplicada y además, con la segunda versión de la especificación, se hubiese hecho más fuerte en relación a la portabilidad de las aplicaciones y demás elementos que lo componen. Sin embargo, en la medida que sea implementada y usada por la amplia comunidad JAVA en diferentes sistemas de producción, se irán encontrando inconvenientes que serán tratados y corregidos en próximas versiones de la especificación.

Es importante resaltar que JAIN SLEE es más que una tecnología para la creación de servicios. Es una completa arquitectura que permite la integración de diferentes tecnologías, sistemas de de negocio y operación (BSS, OSS). Motivo por el cual, aunque haya sido considerado como una tecnología para la creación, su contexto de aplicación está más orientado a una SDP, principalmente para el ambiente de ejecución, corazón de una arquitectura de despliegue de servicios.

Resumen

Las anteriores tecnologías fueron descritas en relación a las características más importantes para la creación, ejecución y despliegue de servicios avanzados de telecomunicaciones. Su análisis más detallado en cuanto a la arquitectura y módulos no es objeto de este estudio. La comparación presente en este anexo va orientada a establecer un contexto sobre la tecnología más adecuada para ser parte de un servidor de aplicación en el dominio telco.

Para un operador de telecomunicaciones, contar con un servidor de aplicación dentro de su NGN, es vital para el aprovisionamiento de servicios. Este elemento, suministra características y facilidades que son comunes a todos los servicios. Dentro de estas características se encuentran: replicación, control de concurrencia, transacciones, gestión de estado de las aplicaciones, entre otras. Esta inclusión de facilidades, libera al desarrollador de la construcción de más código y genera una reducción en los costos de desarrollo y en las fases de mantenimiento. Un servidor de aplicación está orientado a construir el modelo de negocio de la empresa independientemente del domino implementado (IT o Telco). Por esto, para un operador telco, es indispensable adoptar un servidor de aplicación que cumpla con sus requerimientos y se adapte a su NGN.

La mayoría de las tecnologías descritas no son precisamente servidores de aplicación, si no tecnologías que permiten la construcción de servicios de telecomunicaciones sin la existencia de un contenedor de la lógica del servicio. Algunos servidores de aplicación usan estas tecnologías y las integran al ambiente de ejecución que este tipo de elementos proporciona. De esta manera, solo queda en consideración las tecnologías SIP Servlet, JEE y JAIN SLEE. Aunque parezca raro, OSA/Parlay no es tenido en cuenta por la simple razón de no ser una tecnología para ejecutar la lógica del servicio, por el contrario, es una gran herramienta de interfaz para realizar la abstracción de red y creación de nuevas aplicaciones por terceros. La tarea de ejecución de la lógica de negocio es asignada a los servidores estrictamente de aplicación.

En conclusión, JAIN SLEE es sin duda la tecnología más pertinente dentro de un dominio telco, superando las ventajas de SIP Servlet, su principal rival[50] hasta el momento.

[50] En [20], se aprecia un análisis más detallado sobre las ventajas que tiene JAINSLEE en relación a SIP Servelt y su capacidad para migración a redes IMS.

Es importante resaltar que la integración e inter-operabilidad de estas tecnologías es viable y de hecho, ampliamente usado. Esto quiere decir, si un operador adopta e implementa alguna de ellas, es posible adaptarse a nuevas tendencias sin la necesidad de desechar su antigua tecnología. JAIN SLEE por ejemplo, puede adaptarse con tecnologías como JEE, SIP Servlet y OSA/Parlay. La principal razón para esto es que tanto JEE como OSA son tecnologías complementarias a JSLEE. Las Figuras 25 y 26 representan de manera general la conexión entre JAIN SLEE-JEE y JAIN SLEE-OSA/Parlay respectivamente.

Figura 25 Integración entre JAIN SLEE y JEE

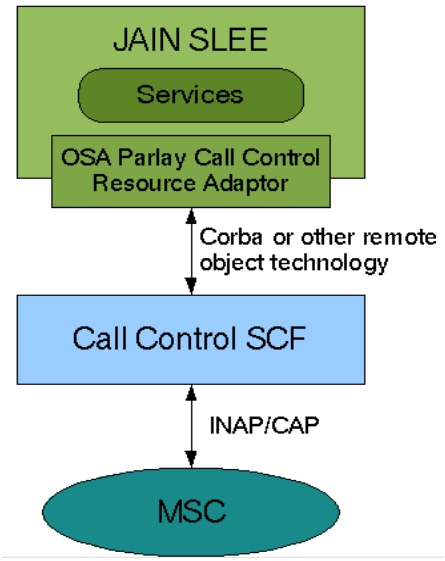


Figura 26 Integración JAIN SLEE y OSA/Parlay
Fuente: JAIN SLEE org[51]

Otro punto de análisis es el contexto de aplicación, donde la presencia de una tecnología es más adecuada sobre otra. Por ejemplo, una integración de JEE y JAIN SLEE expone un ambiente apropiado para el uso de tecnologías de servicios web a fin de componer servicios IT con capacidades telco. Este tipo de aplicación según el contexto, ha sido abordado por diferentes proyectos de investigación como SPICE, OPUCE, LOMS, MAMS, SPAGIC, SeCSE y PLASTIC, todos ellos citados en [24].

A manera de resumen, la Tabla 4 representa la comparación de las tecnologías principales para la construcción, ejecución y despliegue de VAS en NGN.

[51] Tomado de la página oficial de JAIN SLEE [35].

Tabla 4 Tecnologías candidatas para un servidor de aplicación Telco

ÍTEM	OSA/PARLAY	PARLAY X	SIP SERVLET	JAIN SLEE
Orientación	Enterprise-grade[52] on-network[53]	Enterprise-grade on-network	Enterprise-grade on-network	Carrier-grade[54] in-network[55]
Cumplimiento de requerimientos del operador	NO	NO	NO	SI
Soporte de capacidades de red	SI (depende de la especificación y vendedor)	SI (depende de la especificación y vendedor)	NO	SI (independiente del vendedor)
Protocolos manejados	SS7 INAP CAP MAP Otros (depende de la especificación)	SOAP WSDL (sólo para WS)	SIP HTTP	SS7 Megaco Diameter SOAP SIP Otros
Ambiente de ejecución	NO (implementado sobre el Parlay Gateway)	NO (implementado sobre el Parlay Gateway)	SI (contenedor SIP Servlet, aplicaciones basadas en SIP)	SI (estándar)
Nivel de abstracción	Bajo-Medio	Alto (sólo para WS)	Alto	Alto
Acceso a terceros	SI	SI	NO	SI (solución con JSPA)
Arquitectura orientada a eventos	NO	NO	NO	SI
Modelo de componente orientado a objetos	NO	NO	NO	SI
Arquitectura de abstracción de red	NO	NO	NO	SI
Modelo de control ante fallos (contenedor)	NO	NO	NO	SI
Herramientas JEE	SI (servidor externo)	SI (servidor externo)	Extensión de JEE	SI

[52] Conjunto de características esperadas en sistemas IT.
[53] Aplicaciones de terceros desplegadas en equipos que usan capacidades Telco.
[54] Conjunto de características esperadas dentro un operador Telco: alta disponibilidad, redundancia, fiabilidad, entre otras.
[55] Aplicaciones desarrolladas en equipos orientados estrictamente al domino del operador.

Anexo C

Plataforma multiservicio de EMCALI

El presente anexo brinda una descripción general del NGN del operador de telecomunicaciones EMCALI EICE-ESP, de su componente principal para el aprovisionamiento de VAS y se presenta la lista de servicios desplegados dentro de la plataforma mutisevicio.

C.1 NGN EMCALI

La red de EMCALI es el claro ejemplo de la migración tecnológica que demanda el mercado de los nuevos servicios de telecomunicaciones, pasando de una red tradicional conformada por estructuras verticales, la clásica Red telefónica Publica Conmutada (PSTN, Public Switched Telephone Network), hasta modelos horizontales como las NGN. En la medida que el mercado de las comunicaciones ha evolucionado, EMCALI tuvo la necesidad de adaptarse a las nuevas tendencias tecnológicas con el fin de permanecer en el negocio. Motivo por el cual, la empresa se adentra a las redes de nueva generación[56] migrando sus estructuras tradicionales a un mundo basado en IP. EL camino es largo para este operador principalmente porque su modelo de ofrecimiento de servicios de nueva generación pareciera aun no estar concebido en plenitud y peor aún, su modelo de negocio sigue sin ser orientado al mundo NGN manteniendo el implementado en su red PSTN.

La Figura 27 muestra la arquitectura de referencia de red de próxima generación de EMCALI.

Nivel de acceso

Este nivel lo componen tecnologías heredadas como PSTN[57], ATM, RDSI; móviles como GSM/GPRS 2G[58], inalámbricas como WIMAX[59], y telefonía IP sobre dispositivos SIP y Softphones [60].

En relación a los dispositivos de interconexión entre las tecnologías presentes y el nivel de transporte de la NGN de EMCALI, ZTE provee los siguientes equipos de acceso:

Media Gateway ZXMSG 9000: es un equipo de transporte del flujo de datos y señalización. Según la configuración establecida, estos dispositivos actúan como Gateway de Troncal (TG, Trunk Gateway), Gateway de señalización (SG, Signaling Gateway) y Gateway de acceso (AG, Access Gateway).

ZXMSG 9000 como AG: cuando es utilizado este modo, el equipo se ubica en el nivel de acceso de la red de paquetes y permite a suscriptores PSTN, RDSI, V5 y DSL, acceder a la red IP.

- ZXMSG 9000 como TG: cando se utiliza con este fin, el dispositivo es ubicado en el nivel central de la red de paquetes. Esta configuración permite el acceso tanto a los suscriptores de troncal [61] número 7 como a los suscriptores PRI[62].

[56] la red NGN lleva en operación 3 años
[57] En este punto, se realiza la interconexión con las empresas de telefonía móvil celular. La conexión se lleva a cabo entre los equipos de conmutación de esta red y de de la red del operador móvil.
[58] para trabajadores de la empresa y telefonía rural
[59] está implementando en cierta porción no muy amplia y solo para la parte urbana de Cali
[60] Para usuarios NGN de EMCALI
[61] Altos volúmenes de tráfico sobre troncales que manejan señalización numero 7.

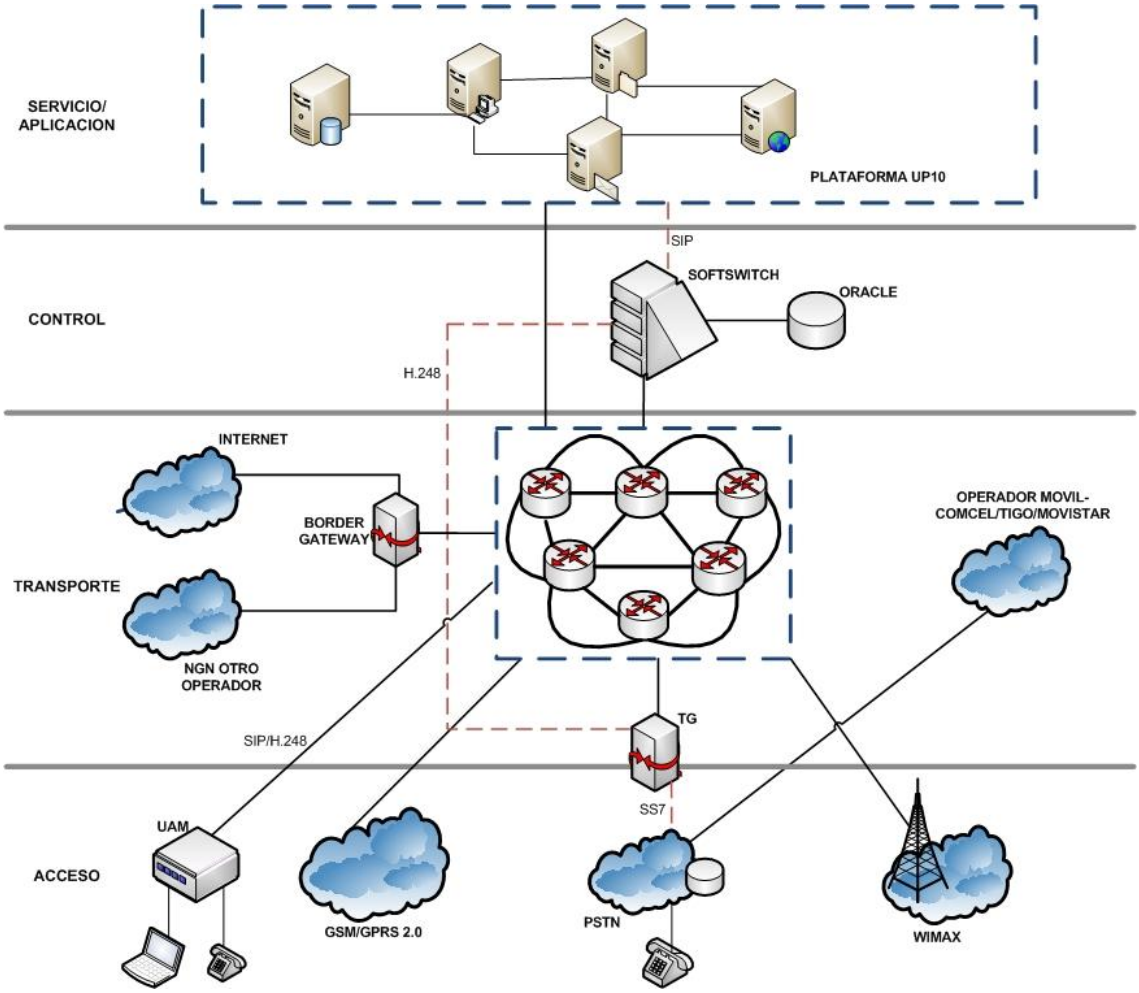


Figura 27 NGN de EMCALI

- ZXMSG 9000 como SG: cuando el equipo es implementado en este modo, se ubica en el núcleo de la red de paquetes y realiza la interoperabilidad entre los mensajes SS7 de la PSTN y los mensajes de la familia de protocolos SIGTRAN[63] del la red IP.

- Border Gateway ZXSS10 B100: es un dispositivo de frontera que enlaza la red pública de un proveedor de servicios a la red privada del proveedor. Este direccionamiento de redes públicas y privadas es obtenido por la traslación de dirección de red (NAT, Network Adress Translation) o vía HTTP.

Dispositivo de acceso integrado (IAD, Integrated Access Device): dispositivo de acceso diseñado para usuarios individuales y pequeñas organizaciones empresariales. Este permite a dispositivos como PC, teléfonos y maquinas de fax acceder la red de conmutación de paquetes.

Nivel de Transporte

Este nivel está conformado por el núcleo de Conmutación de Etiqueta Multiprotocolo (MPLS[64], Multiprotocol Label Switching) con elementos de red de capa 3. Los seis anillos (formados por los

[62] Canal primario definido en RDSI para altos volúmenes de tráfico
[63] Grupo de trabajo del IETF que ha desarrollado una serie de protocolos para transportar SS7 por redes IP.

nodos Colón, Guabito, San Fernando, Versalles, Poblado y Parcelaciones) dan completo soporte a toda la red multiservicio. En este nivel se realiza la conexión con la red de Internet y con redes NGN de otro operador. Los enrutadores están sujetos a un diseño de redundancia 1 a 1 y diseñados para soportar una gran variedad de protocolos. La Tabla 5, resume los protocolos manejados por estos dispositivos.

Tabla 5 Protocolos soportados por los dispositivos del nivel de transporte de la NGN

TIPO DE PROTOCOLO	PROTOCOLOS
De nivel de enlace	PPP, MPPP
Nivel de Red	IP, ICMP, ARP, V-SWITCH, SMARTGROUP
Nivel de Transporte	TCP y UDP
De Enrutamiento	RIP v1/v2, OSPF v2, BGP4, IS-IS, IPv6, RIPng, OSPF v3, BGP4+, IS-IS6, MPLS/VPN, VPWS, VPLS, QoS, RSVP TE, políticas de enrutamiento y funciones de carga compartida
Tunel	GRE
Nivel de aplicación	Telnet, FTP y TFTP
Control de red y aplicación	NAT, ACL y URPF
Protocoloes NM	SNMP v1/v2/v3, RMON v1 y NTP

El border Gateway conecta los clientes NGN que usan herramientas software para el establecimiento de llamadas como el Xlite y el EMVOZ[65] a través de la red de internet.

Nivel de Control

Este nivel es el más importante de toda red NGN. En este punto se realiza el control sobre las llamadas, señalización, control de flujos de información, funciones AAA[66], entre muchas otras. El componente esencial en esta arquitectura es el Softswitch, encargado de realizar todas funciones de procesamiento de control integrado como procesamiento de llamadas, adaptación de los protocolos de acceso, autenticación, enrutamiento, asignación de recursos, almacenamiento de registro detallado de llamada (CDR, Call Detail Record) y la interconexión e inter-funcionamiento de todos los nodos de la plataforma multiservicio de EMCALI, de igual manera, el Softswitch suministra todos los servicio básicos de llamada, servicios suplementarios, servicios de valor agregado basados en web y servicios multimedia punto a punto de la red PSTN.

En cuanto a su arquitectura, el nivel físico está constituido por un componente de procesamiento en tiempo real, un servidor de base de datos, un componente de OSS y un sistema de conmutación de red que interconecta todos los módulos internos. La capa lógica esa dividida en un de un nivel de servicio, un nivel de control y un nivel de adaptación.

---

[64] Es un mecanismo de transporte de datos estándar creado por la IETF y definido en el RFC 3031. Opera entre la capa de enlace de datos y la capa de red del modelo OSI. Fue diseñado para unificar el servicio de transporte de datos para las redes basadas en circuitos y las basadas en paquetes. Puede ser utilizado para transportar diferentes tipos de tráfico, incluyendo tráfico de voz y de paquetes IP.

[65] Programa desarrollado por ZTE con el fin de suministrar mensajería multimedia para los clientes de la empresa. Este programa aparte de la facilidades de mensajería permite hacer llamadas locales ilimitadas

[66] Siglas de Autenticación, Autorización, Facturación (en inglés Accounting)

- *Nivel de servicio:* en este nivel se lleva a cabo el enlace con los niveles superiores y a los sistemas de información empresarial por medio del gestor del servicio y del gestor de datos respectivamente. El gestor del servicio sirve como punto de control para la interacción entre el Softswitch y el Punto de Control de Servicio (SCP, Service Control Point) del servidor de aplicaciones, y el gestor de datos suministra una interfaz unificada a la base de datos de EMCALI.

- *Nivel de control:* este permite el control sobre la funciones de procesamiento de llamada, de acceso a protocolos, interconexión, funcionamiento y soporte a toda la infraestructura de EMCALI. Este lo componen los módulos de gestor de recursos, control de llamada y conexión con otros nodos de control. el primero asigna los recursos más adecuados para manejar la solicitud de algún servicio en particular, el segundo, se encarga de realizar un control de llamada unificado y finalmente, el modulo SIP/SIP-T soporta la interconexión entre diferentes Softswitch.

- *Nivel de adaptación:* son todos aquellos protocolos manejados por el nodo que permiten la conexión y señalización con toda la infraestructura NGN del operador (SIP, H.323, SS7-INAP/CAP/MAP, H.248. entre muchos otros).

La Figura 28 muestra la distribución de los bloques lógicos de la arquitectura del Softswitch

Figura 28 Componentes lógicos del Softswitch
Fuente: ZTE[67]

Este componente en conjunto con los servidores de aplicación, proveen a la NGN de EMCALI, una variedad de servicios avanzados de telecomunicaciones, desde servicios tradicionales inteligentes hasta servicios mejorados en redes IP.

Nivel de aplicación y servicio

Este nivel está conformado por la plataforma de despliegue UP10. Esta plataforma la componen un número de servidores y módulos de control para la prestación de servicios. A continuación se detalla

[67] Tomado del documento "*ZXSS10 SS1b SoftSwitch Control Equipment Technical Manual*" [25].

más a fondo este nivel en razón que el análisis del presente proyecto se enfoca en la creación, ejecución y despliegue de servicios de valor agregado soportados sobre el nivel de aplicación y servicio.

C.2 Plataforma UP10

Esta plataforma, conocida a nivel productivo como ZXUP10, cumple con las especificaciones de la ITU, ETSI y el IETF. La arquitectura, Figura 29, se basa principalmente en la especificación de OSA/Parlay versión 3 (ETSI ES 201 915) y se encuentra dividida en cuatro bloques funcionales, (1) el nivel de aplicación, (2) el nivel de control, (3) el nivel de adaptación y (4) el nivel de recursos.

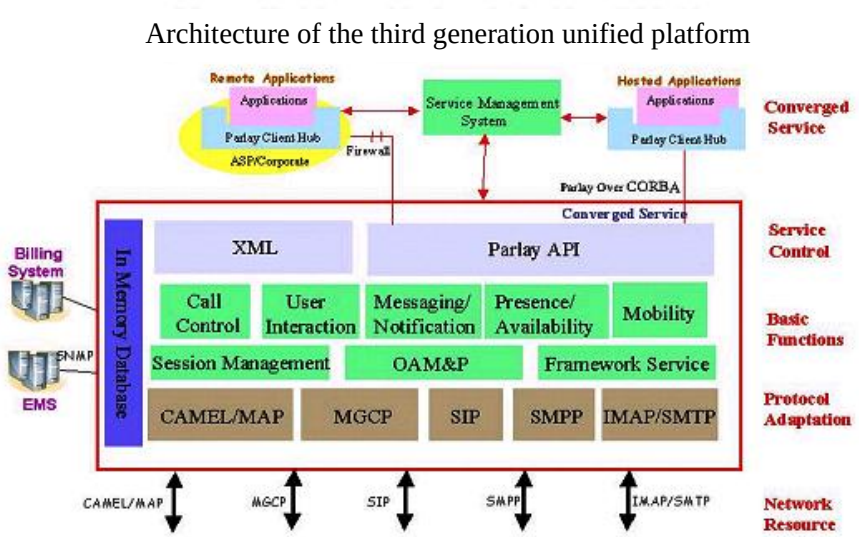


Figura 29 Arquitectura de referencia ZXUP10
Fuente: ZTE[68]

Componentes lógicos

1) Nivel de aplicación

En este nivel residen las aplicaciones que usa una o más SCF[69]. Estas aplicaciones son registradas y autenticadas dentro del Framework API y luego (véase sección 2.1.3.5), la aplicación tiene la autorización para el uso de las SCF sobre el servidor Parlay. Este mecanismo de seguridad implementado por la plataforma a través del cliente parlay (PCH, Parlay Client Hub), permite a terceros hacer uso de los recursos y capacidades del operador para construir nuevos servicios. Sin embargo, aunque se cuente con dicho componente, el acceso está totalmente cerrado en gran parte por problemas contractuales presentes entre el operador EMCALI y uno de sus proveedores de equipos de telecomunicaciones, ZTE. Este inconveniente ha traído todo tipo de restricciones en cuanto al uso y accedo de la tecnología OSA/Parlay para desarrollo de terceros, motivo por el cual, esta tecnología no ha podido ser implementada dentro de trabajos externos a los desarrollados y ejecutados por la compañía ZTE.

[68] Tomado del documento "*ZXUP10 System introduction*" [36].
[69] control de llamada, interacción de usuario, gestión de movilidad, mensajería, facturación, gestión de conectividad, framework y operación y mantenimiento [36].

2) Nivel de control de servicio.

Este nivel de control implementa la API de Parlay y registra los módulos SCF con el Framework de forma tal, que una aplicación sabe si puede obtener cierto servicio del servidor Parlay.

3) Nivel de adaptación.

Este nivel mapea los diferentes protocolos de red[70] a una interfaz unificada suministrada por la API Parlay con el fin de ser usada por cualquier modulo SCF.

4) Nivel de recursos.

Conformado por el Softwicth, centro de conmutación móvil, servidor de medios, servidor de correo y centro de mensajería corta.

Todo este conjunto de niveles se encuentran encapsulados en componentes físicos dentro de la plataforma. A continuación se describe la arquitectura física de la plataforma ZXUP10.

Componentes físicos

La plataforma ZXUP10 es un infraestructura integrada conformada por elementos como el Parlay Gateway, servidor de medios, servidor de aplicaciones, servidor Texto a Habla (TTS, Text to Speech), Gateway de señalización y consola de operación, administración y mantenimiento.

1) Parlay Gateway[71]

Este es el componente más importante de la plataforma. De él depende la adaptación de red, el suministro de la API de acuerdo al protocolo y el acceso a terceros para el desarrollo de nuevos servicios. Lo conforman tres módulos funcionales, (1) el adaptador de protocolo, (2) el bloque de funciones básicas y (3) el bloque de control de servicio. Sus diferentes niveles se aprecian en la Figura 30

Adaptador de protocolo: corresponde al nivel de adaptación de red. Este punto permite encapsular los diferentes protocolos que serán usados para la construcción de nuevos servicios.

Módulo de Función Básica: este componente se encarga de la gestión y control del Parlay Gateway. También contiene las capacidades básicas para la construcción de servicio (SCF) y el Framework según lo establecido por la especificación de OSA/Parlay.

- Framework: este sub-modulo ofrece funciones de gestión orientadas al usuario y orientada a la aplicación. La primera es en función de la confiabilidad y seguridad, y la segunda, en relación al registro, autenticación y autorización.

- Call Control: sub-modulo encargado de suministrar las capacidades básicas de control de llamada y del establecimiento de sesiones.

- Interacción de usuario: parte que suministra la comunicación de información entre usuarios.

[70] soporta protocolos como CAMEL, SIP, MGCP,IMAP, SMTP, SMPP [36].
[71] reúne los niveles de control y adaptación de la plataforma ZXUP10

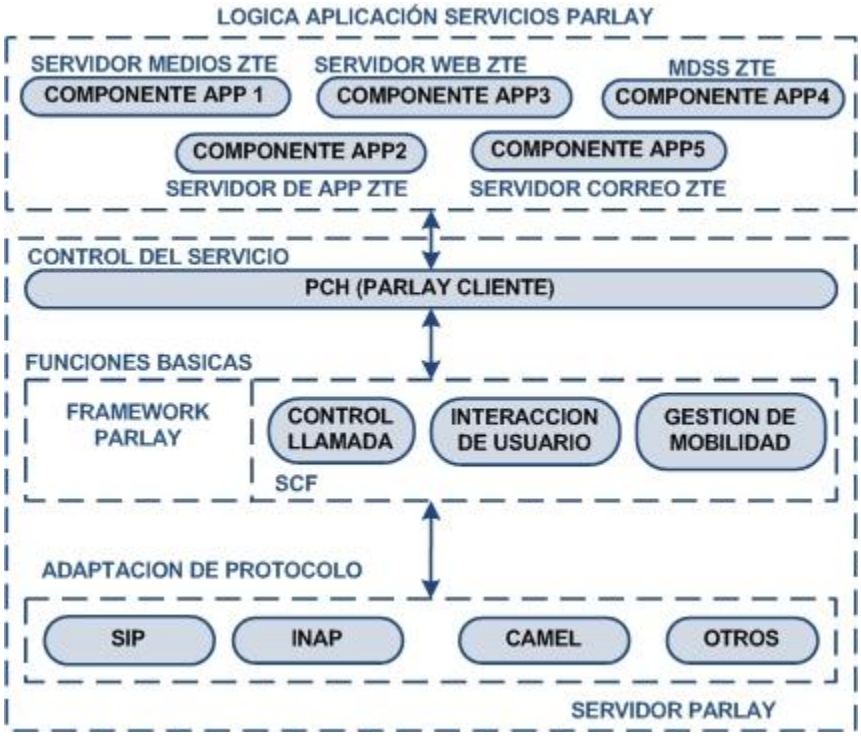


Figura 30 Módulos del Parlay Gateway
Fuente: ZTE[72]

- Gestión de movilidad: provee el servicio de ubicación geográfica y estado de actividad del usuario.

- Mensaje general: representa el sub-modulo para enviar, almacenar y recibir mensajes de correo de voz o electrónicos.

- Charging: modulo que genera y gestiona las sesiones de cobro de los servicios. En este punto se realiza la generación de CDR.

- Gestión de conectividad: sub-modulo usado para gestionar la conectividad entre la red de la empresa y la red del proveedor del servicio. En este punto se configurar los parámetros de QoS para enviar los paquetes de información ente las diferentes redes.

- OA&M: sub-modulo encargado de realizar funciones generales de gestión, administración y mantenimiento.

Módulo de control de Servicio: este componente es el encargado de implementar la API de Parlay, lo que hace posible, registrar las SCF al framework de forma que cualquier aplicación haga uso de ellas. Este módulo implementa la tecnología PCH con el fin de que terceros accedan a las SCF para desarrollar una gran variedad de servicios sin la necesidad de conocer los protocolos específicos de red. La Figura 31 representa la manera como las aplicaciones convergentes a través del API Parlay usan las SCF para suministrar servicios.

[72] Tomado del documento "*ZXUP10 System introduction*" [36].

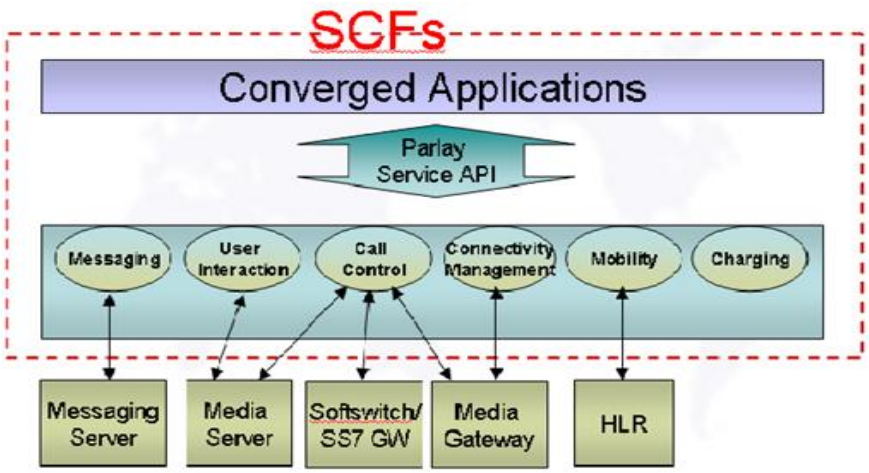


Figura 31 Interacción de las aplicaciones con las SCF
Fuente: ZTE[73]

2) Servidor de aplicaciones

En este componente se ejecuta la lógica de las aplicaciones. Los servidores aquí contenidos hacen uso de las capacidades del Gateway habiendo primero pasado por los mecanismos de registro, autenticación y autorización. Los servidores de aplicaciones pueden ser implementados en maquinas diferentes o integrados dentro el mismo servidor Parlay [74] pero generalmente, estos elementos son desplegados por terceros en el modelo OSA.

3) Servidor de medios.

Brinda el soporte para las funciones especializadas que requieren servicios de alto nivel como conferencia, gestión, mantenimiento, conversión de diferentes algoritmos de codificación, etc.

4) Servidor TTS.

Es un servidor que convierte un conjunto de palabras en flujo de media la cual es transmitida al servidor de medios para ser reproducido a los usuarios.

5) Gateway de señalización.

Componente encargado de conectar los protocolos de señalización SS7 (INAP, CAP, MAP) con los protocolos de señalización IP (SIP, MGCP).

6) Consola de operación, mantenimiento y gestión.

Este componente permite el manejo y mantenimiento del sistema ZXUP10. Implementa funciones como configuración y modificación de datos, señalización de ruta, mantenimiento de alarmas, solicitud de información de llamada, gestión del operador, entre otras.

La Figura 32 muestra la como se realiza la conexión del Parlay Gateway con el Softswitch de ECMALI.

[73] Tomado del documento "*ZXUP10 System introduction*" [36].
[74] para la implementación de servicio, RHINO JAIN SLEE estaría dentro de otra máquina y actuaria como tercero

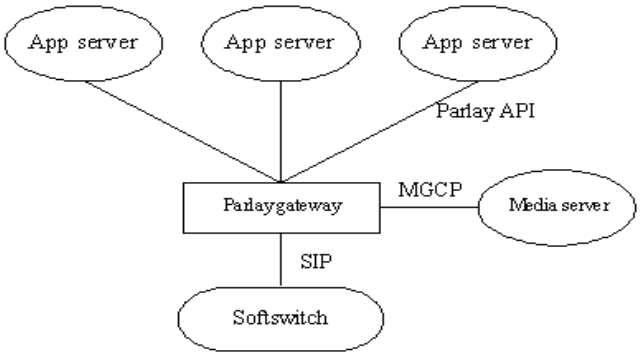


Figura 32 Conexión entre Parlay Gateway y Softswitch
Fuente: ZTE[75]

C.3 Servicios soportados por la plataforma ZXUP10

Esta plataforma suministra una variedad de servicios personalizables sobre NGN como llamada web, conferencia web, web 800, correo de voz, Un Numero de enlace (ONLY, One Numer Link You) y servicio de mensajería unificada (UMS, Unified Messaging Service). A continuación se realiza una breve descripción de cada uno de ellos.

Correo de voz: un suscriptor de este servicio puede escuchar a través de correo (acceso vía web) o por llamada telefónica convencional, el mensaje que cualquier usuario deja por medio del lanzamiento de la respuesta de voz interactiva (IVR, Interactive, Voice, Response) en el buzón de voz del suscriptor del servicio. Normalmente, este servicio es usado en conjunto con el servicio ONLY y UMS.

Conferencia Web: Este servicio está desarrollado para múltiples redes y consiste principalmente en el establecimiento de llamadas para conferencia basada en la web. Un usuario puede atender una conferencia a través del terminal soft, conjunto de teléfonos normales, terminales SIP o teléfonos móviles. Toda la información es desplegada en páginas web en tiempo real. Es posible también crear sub-conferencias a través de estas páginas.

Web 800: este servicio es similar al servicio prestado sobre la red inteligente en relación a la llamada 01 8000 con la diferencia que se realiza sobre la web. Entre las compañías que pueden incluir este servicio se destacan las agencias de viajes, compañías de aerolíneas, empresas de transporte e instituciones educacionales.

UMS: este servicio es pensado para usuarios individuales donde se conjuga una serie de características para brindar acceso unificado a varios servicios. Entre ellos, acceso a terminales de usuarios (teléfonos móviles, teléfonos SIP, softphones), e integración de servicios de voz, video, datos y fax. La información almacenada en un punto central del sistema, es accedida por cualquiera de los siguientes dispositivos: PC (vía correo electrónico), fax, telefonía (vía servicio correo de voz), y aplicaciones de mensajería (MSN, Xlite, EMVOZ, SMS)

ONLY (): este es un servicio de movilidad en cual el suscriptor usa un único número personal de telecomunicaciones (PTN, Personal Telecommunicatin Number) a fin de acceder a cualquier red y

[75] Tomado del documento "*ZXUP10 System introduction*" [36].

recibir llamadas a través de ellas. Según la configuración del usuario, la llamada entrante al PTN, puede ser direccionada a números diferentes. Esto permite al usuario estar siempre en contacto y disponible, bien sea en su teléfono SIP, en su teléfono móvil, en la oficina, en la casa, etc. La configuración de los números telefónicos se hace vía web por el suscriptor del servicio

Llamada Web: este servicio permite al usuario originar llamadas con solo hacer click sobre alguno de los números desplegados en una página web. La conexión RTP se realiza entre dispositivos SIP, teléfonos soft, teléfonos convencionales PSTN, telefonía IAD y dispositivos móviles.

SoftDA: este servicio conocido comercialmente como EMVOZ[76], presta características semejantes al MSN de la compañía Microsoft. Entre sus funciones se encuentran: mensajería, sesiones de llamada, sesiones de videoconferencia, transferencia de archivos, video llamada, entre muchas otras. La característica diferencial en relación a otros clientes de mensajera comerciales, radica en el establecimiento de llamadas locales bien sea a usuarios NGN o usuarios PSTN.

[76] Actualmente no está comercializado por EMCALI.

Anexo D

Modelado del Servicio - Línea Base Arquitectónica

En este apartado se realizara el análisis de la línea base arquitectónica desde las respectivas vistas de funcionalidades, de referencia, de implementación, de distribución física de le elementos y de componentes modulares, exponiendo y describiendo los diferentes modelos (de casos de uso, de análisis, de diseño, de implementación y de despliegue) que componen la implementación del servicio de Tono de Timbre Personalizado (CRBT, Color Ring Back Tone) en la NGN de operador EMCALI.

D.1 Características del servicio

El desarrollo del piloto consiste en la implementación de algunas de las funcionalidades del servicio CRBT. No es objetivo del proyecto desarrollar un completo VAS, sino por el contario, aplicar los criterios establecidos.

En esta parte se sugiere utilizar una metodología de desarrollo software que facilite el análisis, diseño, implementación y despliegue, de una solución. En el presente trabajo de grado, la metodología utilizada para el desarrollo del prototipo CRBT se basó en el MCS.

A continuación se mencionan las características más importantes con las cuales cuenta el prototipo CRBT:

- Funcionales:
  - Reproduce RBT creados a partir de componentes de audio, reemplazando el RBT tradicional.
  - Ofrece una mínima personalización del servicio, la cual permite seleccionar un RBT por defecto para cada suscriptor, el cual se reproduce para cualquier llamada entrante del mismo.
- No funcionales:
  - Permite la suscripción a los clientes que pertenecen a la NGN de EMCALI.
  - Es independiente del dispositivo final (teléfono convencional, móvil, softphone, SIP) y de la red (NGN, PSTN, GSM). Es decir, el RBT se reproduce hacia cualquier terminal de usuario que tenga conexión con la NGN de EMCALI (**convergencia a nivel de terminal y de red**).

Permite al administrador de Rhino SLEE suscribir nuevos usuarios y modificar la personalización del servicio a través de las herramientas de creación y gestión de perfiles que ofrece el ambiente de ejecución

De acuerdo al análisis de las características generales del servicio CRBT y las condiciones de red del operador EMCALI (ver Anexo C) se presenta la línea base arquitectónica del sistema solución.

D.2 Línea base arquitectónica

D.2.1 Vista de funcionalidades

Modelo de caso de uso

Este modelo, Figura 33 y Tabla 6, describe las funcionalidades y el escenario del servicio desarrollado según los criterios establecidos.

Figura 33 Diagrama de casos de uso del prototipo CRBT

Tabla 6 Descripción del escenario de caso de uso Enviar Audio

<table>
<tr><th>Caso de Uso</th><th>Reproducir RBT audio</th></tr>
<tr><td>Actores</td><td>Sistema Llamada (iniciador).
Usuario A.</td></tr>
<tr><td>Importancia</td><td>Alta</td></tr>
<tr><td>Propósito</td><td>Reproducir al actor Usuario A un componente de audio, previamente definido, que reemplaza el RBT tradicional.</td></tr>
<tr><td>Resumen</td><td>Cuando el Usuario A inicia una solicitud de llamada a un usuario suscriptor del servicio, la entidad Sistema Llamada detecta el número destino y direcciona la señalización de llamada al prototipo CRBT, el cual verifica el RBT de audio definido en el perfil CRBT del suscriptor, y reproduce al Usuario A el archivo de audio correspondiente, en reemplazo del RBT tradicional.</td></tr>
<tr><td colspan="2">Precondiciones</td></tr>
<tr><td colspan="2">El Usuario B deber estar suscrito al servicio y tener el archivo audio de su preferencia</td></tr>
<tr><td colspan="2">Escenario
<table>
<tr><th>Actores</th><th>Sistema</th></tr>
<tr><td>Recibe solicitud de llamada del usuario llamante (usuario A)
Envía solicitud de inicio de llamada</td><td>Recibe la solicitud de inicio de llamada
Verifica que el usuario llamado tenga activo el servicio CRBT (E1)
Carga el perfil de usuario suscriptor
Retorna solicitud de envío de llamada</td></tr>
<tr><td>Recibe solicitud de inicio llamada
Envía solicitud de inicio de llamada a usuario B
Recibe estado disponible de usuario llamado
Envía estado disponible de usuario llamado (usuario B)</td><td>Recibe estado disponible de usuario llamado
Envía archivo de audio al usuario llamante</td></tr>
</table></td></tr>
<tr><td colspan="2">Poscondiciones</td></tr>
<tr><td colspan="2">Llamada establecida.</td></tr>
<tr><td colspan="2">Excepciones</td></tr>
</table>

E1: Servicio inactivo. No se carga el perfil y el proceso de establecimiento de llamada continua normalmente.

Dentro del modelo de diseño será detallado con más claridad el funcionamiento del CRBT (Figura 37)

Modelo de análisis

Este modelo representa la estructura del servicio final. Según la relación usuario-sistema se exponen los siguientes diagramas, Figuras 34 y 35.

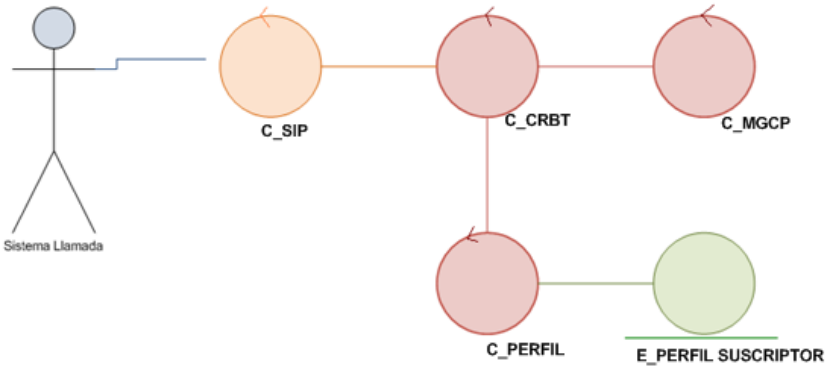


Figura 34 Diagrama de clases de análisis del prototipo CRBT

Tabla 7 Clases de análisis

Nombre de la Clase	Tipo	Responsabilidad
C_Sip	Control	Manejo de la señalización SIP
C_CRBT	Control	Realiza la lógica del servicio CRBT
C_Perfil	Control	Ejecuta las tareas de control sobre el perfil de usuario suscriptor
C_Mgcp	Control	Controla la comunicación con el servidor de medios
E_Perfil-Suscriptor	Entidad	Contiene la información del perfil del usuario suscriptor

Figura 35 Diagrama de paquetes de análisis del prototipo CRBT

Tabla 8 Paquetes de análisis

Nombre	Descripción
Control	Contiene todas las clases que realizan el control de señalización SIP, mensajes MGCP y control de la lógica del servicio (C_SIP, C_MGCP, C_CRBT)
Persistencia	Contiene las clases que acceden al perfil del suscriptor (C_perfil)
Bd	Agrupa la clases que contienen la información del perfil del suscriptor (E_Perfil-Suscriptor)

D.2.2 Vista del modelo de referencia

Modelo de diseño

Según el modelo de análisis se ha desarrollado un modelo final de diseño a través del refinamiento de responsabilidades de las clases y del funcionamiento del sistema. Las Figuras 36 y 37, muestran la estructura y comportamiento final del servicio implementado.

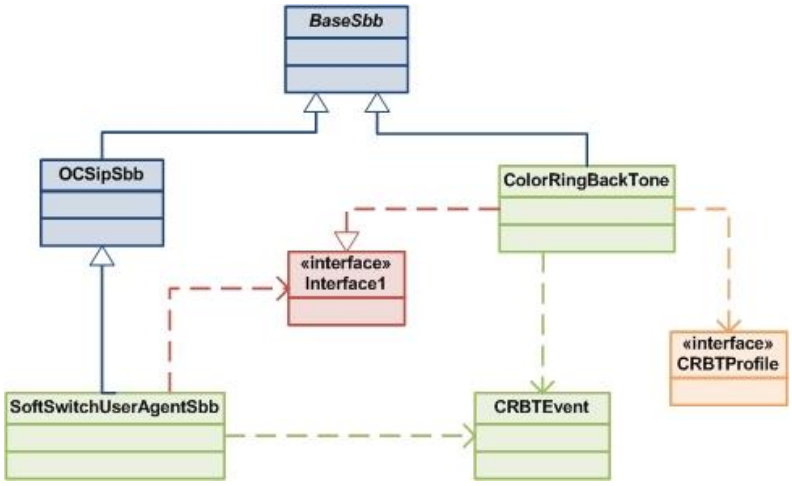


Figura 36 Diagrama de clases de diseño del prototipo CRBT

El anterior diagrama representa la estructura general del servicio CRBT. Para un claro entendimiento se realiza a continuación la descripción de cada uno de los componentes.

- Las *clases BaseSbb y OCSipSbb* son clases re utilizadas con el fin de implementar métodos necesarios y comunes para el funcionamiento del servicio. Mientras la primera contiene métodos típicos en la construcción de elementos SBB, la segunda está más orientada al manejo de mensajes SIP.

- *SoftSwitchUserAgentSbb:* esta clase basada en el código fuente del BackToBackUserAgentSbb de Opencloud, se ajusta para los requerimientos particulares del servicio CRBT dentro de la NGN de EMCALI. Su función es el control de los mensajes SIP generados durante tolo el establecimiento y finalización de la llamada para el servicio CRBT. Con el fin de mantener el estado de los Bloques de Construcción del Servicio (SBB,Service Building Block), esta clase usa los campos de Persistencia Gestionada por Contenedor (CMP, Container Managed Persistence). A diferencia de la programación normal en Java, este nuevo paradigma basado en eventos, requiere del manejo de estructuras diferentes. Esta es la razón de no usar parámetros dentro de la clase si no los campos CMP.

  El control de los mensajes SIP se realiza sobre la manipulación de eventos dentro de la clase. En este punto, se manejan los eventos de tipo Request y de tipo Responses. Los primeros son todos los mensajes SIP como INVITE, CANCEL, BYE y ACK. Los segundos, todos aquellos con respuesta 1xx y 2xx, como mensajes OK (código 200), Ringing (codigo180) y Sesion Progress (código 183).

- *ColorRingBackToneSbb:* esta clase se encarga del control del servicio relacionado con la lógica y la comunicación con el servidor de medios. En este punto se majean los mensajes MGCP como el CRCX RESPONSE que realiza la interacción con el MMS. Estos mensajes son atendidos por los eventos respectivos guardando su estado en los campos CMP.

- *CRBTSbbLocalObject:* interfaz que contiene los métodos que la clase SoftSwitchUserAgentSbb usa para la comunicación con la lógica del servicio. Los métodos descritos por esta interfaz son:createCRBTConnection, playCRBTMedia, isCRBTSubscriber, deleterCRBTConnection y cancelCRBTProcess.

- *CRBTProfileCMP:* interfaz para acceder a los campos de perfil. Esta clase contiene los métodos necesarios para realizar la lectura del archivo media relacionado al suscriptor.

- *CRBTEvent*: clase encargada de comunicar todos los eventos producidos dentro de las transacciones del servicio. Este punto realiza el control del flujo de mensajes necesarios entre las clases que manejan la señalización SIP y los mensajes MGCP.

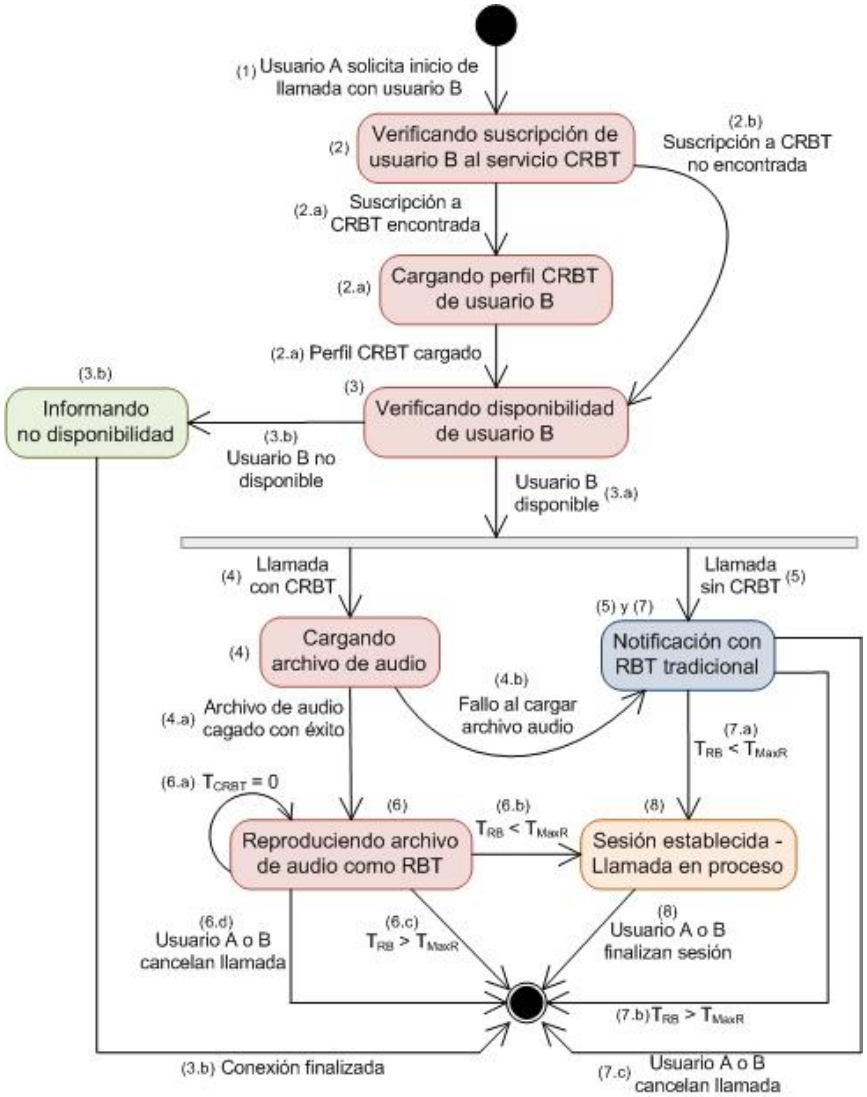


Figura 37 Diagrama de estados del caso de uso Reproducir RBT de audio

El anterior diagrama (Figura 37) representa el prototipo CRBT implementado. Este se caracteriza por ser un servicio en tiempo real, reactivo y basado en protocolos, razones por las cuales es importante presentar un diseño basado en diagramas de estados, ya que describen de una mejor manera éste comportamiento.

A continuación se exponen y describen los distintos estados, eventos y transiciones, que ocurren en el caso de uso Reproducir RBT de audio en la NGN de EMCALI

(1) El flujo del prototipo inicia cuando un usuario A solicita establecer una sesión de llamada con un usuario B.
(2) Cuando el sistema verifica que el usuario B tenga suscrito el servicio CRBT:
    a. Si la suscripción del usuario B al servicio CRBT es encontrada, el sistema carga su perfil CRBT y procede a verificar la disponibilidad del usuario B.
    b. Si la suscripción del usuario B al servicio CRBT no es encontrada, el sistema procede directamente a verificar la disponibilidad del usuario B.
(3) Cuando el sistema verifica la disponibilidad del usuario B:
    a. Si el usuario B se encuentra disponible, el sistema procede a iniciar la sesión de llamada hacia este usuario: con CRBT, si la suscripción al servicio CRBT fue encontrada; sin CRBT, si tal suscripción no fue encontrada.
    b. Si el usuario B no se encuentra disponible (ocupado, no accesible o no conectado), el sistema informa al usuario A dicho estado y finaliza la conexión.
(4) Para una llamada con CRBT, el sistema carga el archivo de audio, previamente definido en el perfil CRBT del usuario B, el cual fue obtenido en el punto 2-a.
    a. Si el archivo de audio es cargado exitosamente, el sistema procede a reproducir el componente de audio como RBT de la llamada.
    b. Si el archivo no es encontrado o existe un error en la carga del mismo, el sistema procede con la llamada sin CRBT.
(5) Para una llamada sin CRBT, el sistema emplea el RBT tradicional para notificar al usuario A acerca del proceso de la llamada.
(6) Cuando el sistema reproduce el archivo de audio como RBT al usuario A:
    a. Si el tiempo de duración del archivo de audio (T_{CRBT}) llega a su fin ($T_{CRBT} = 0$) y el usuario B aún no responde la llamada, el archivo de audio se reproduce de nuevo.
    b. Si el tiempo de respuesta del usuario B (T_{RB}) es menor al tiempo máximo de espera para la respuesta de la llamada (T_{MaxR})[77] ($T_{RB} < T_{MaxR}$), el sistema detiene la reproducción del RBT de audio y establece la llamada, permitiendo el intercambio de voz entre los usuarios.
    c. Si $T_{RB} > T_{MaxR}$, el sistema detiene la reproducción del RBT de audio y finaliza la conexión.
    d. Si el usuario A o B cancelan la llamada, el sistema detiene la reproducción del RBT de audio y finaliza la conexión.
(7) Cuando el sistema notifica con el RBT tradicional al usuario A:
    a. Si $T_{RB} < T_{MaxR}$, el sistema establece la llamada.
    b. Si $T_{RB} > T_{MaxR}$, el sistema finaliza la conexión.
    c. Si el usuario A o B cancelan la llamada, el sistema finaliza la conexión.
(8) Una vez la llamada se establece y se encuentra en proceso, si el usuario A o B terminan la sesión, el sistema finaliza la conexión.

[77] Para el caso de EMCALI, el T_{MaxR} es de 1 minuto, valor definido por el responsable del Softswitch de esta empresa.

Modelo de implementación

Este modelo representa los módulos funcionales en la prestación del servicio CRBT dentro de la NGN de EMCALI, la Figura 38 muestra los diferentes bloques que componen el VAS construido y los elementos lógicos que dan soporte al servicio.

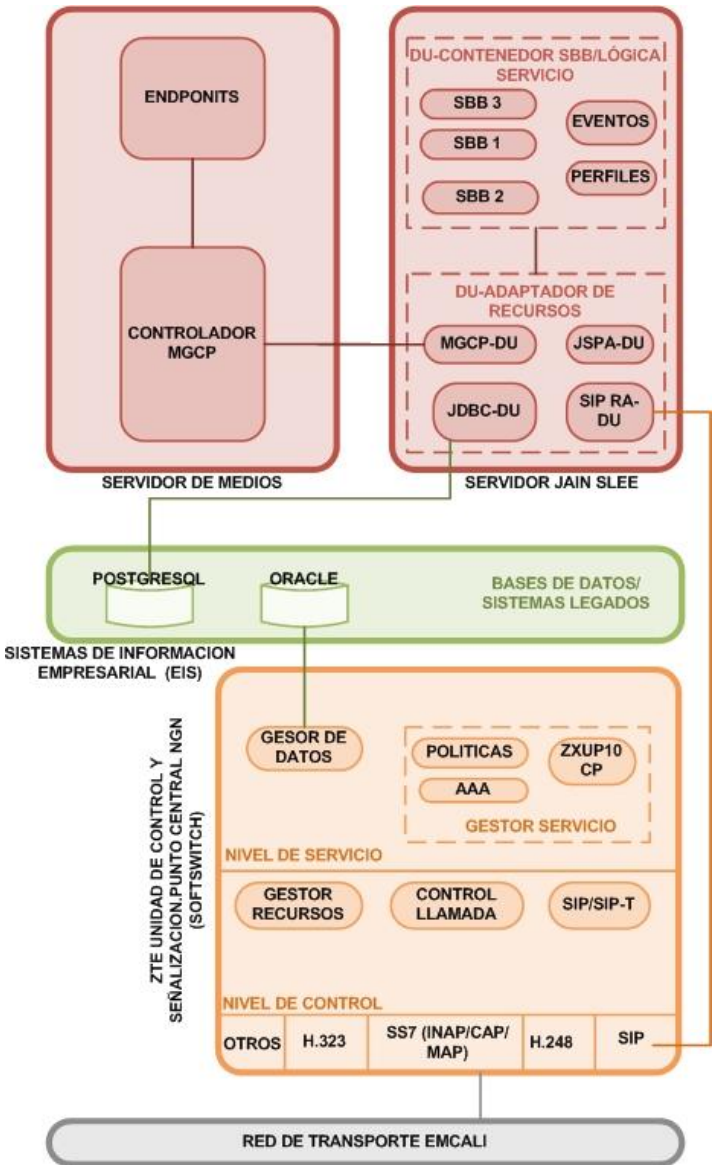


Figura 38 Arquitectura modular del sistema solución

- *Servidor JAIN SLEE (RHINO SLEE):* este componente contiene módulos funcionales denominados unidades desplegables. En ellas se encuentran elementos necesarios para la prestación del servicio CRBT. La primera unidad incluye los bloques que ejecutan la lógica de la aplicación. Estos bloques estos construidos por eventos, perfiles y los SBB. Las otras dos unidades la componen los bloques encargados de realizar la comunicación con el modulo de control de la NGN de EMCALI y el modulo de medios (SIP RA y MGCP RA)

- *Servidor de Medios (MMS):* este componente modular es el encargado de contener los archivos de audio que son reproducidos al usuario llamante. Dentro del él se destaca el bloque de controlador MGCP encargado de implementar la interfaz del estándar para este protocolo. El bloque EndPoint es una representación lógica de una entidad física como un enlace troncal o una fuente de audio. Para el prototipo se selecciono el "*custom endopoint"* el cual, a criterio del desarrollador, es posible crear una propia representación lógica del recurso físco (archivo de audio)

- *Unidad de control y señalización:* este componente es el punto central de la arquitectura NGN de EMCALI, encargado del flujo de información y control desde los usuarios hasta las capas superiores (Servidores de aplicaciones) y viceversa. Está compuesto de un nivel de servicio, un nivel de control y un nivel de adaptación (ver Anexo C).

- *Sistemas de información empresarial (EIS, Enterprise Information System):* este modulo lo conforman todos aquellos recursos de carácter empresarial que manejan la información del negocio como por ejemplo, los recursos de planeación empresarial, el procesamiento de transacciones, bases de datos, etc. En este caso particular la base de datos implementada es PostgreSQL.

D.2.3 Vista de Implementación

Modelo de implementación

A continuación se describe los componentes ejecutables finales del sistema para el funcionamiento del servicio CRBT en la NGN de EMCALI.

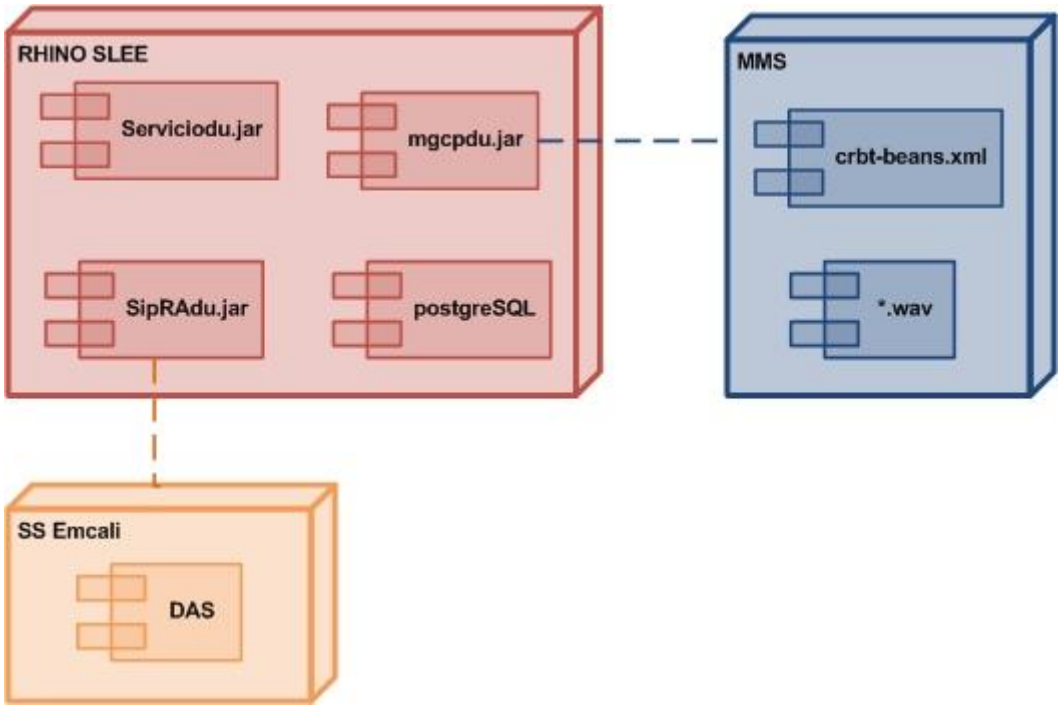


Figura 39 Diagrama de componentes del sistema solución

Tabla 9 Descripción de componentes del sistema solución

Componente	Descripción
Servicio.jar	Componente ejecutable de clases que realizan la lógica del servicio
SipRA.jar	Componente ejecutable que maneja la interconexión con el nodo control de la NGN de EMCALI y los mensajes SIP al interior del servicio
MGCP.jar	Componente ejecutable encargado de realizar la comunicación con servidor de medios por medio de los mensajes MGCP
DAS	Elemento encargado de llevar a cabo el análisis de dígitos de un número entrante a la NGN de ECMALI. En caso de que el usuario llamado sea suscriptor del servicio CRBT, la solicitud de inicio de sesión será direccionada al servidor RHINO. En este punto se puede configurar un número o una serie de números los cuales serán enrutados al servidor JSLEE.
Profile.sql	Componente que se ejecuta en la base de datos. Permite el acceso al perfil del suscriptor.
Crbt-beans.xml	Archivo ejecutable que contiene la configuración del número de endpoint o conexiones simultaneas permitidas dentro del MMS
*.wav	Componente que contiene todos los archivos de audio a ser reproducidos.

D.2.4 Vista de distribución física de elementos

Modelo de despliegue

Dentro de este modelo, se presenta el diagrama de despliegue, Figura 40, de los componentes físicos que representan la interacción con la NGN de EMCALI y los elementos que prestan el servicio CRBT.

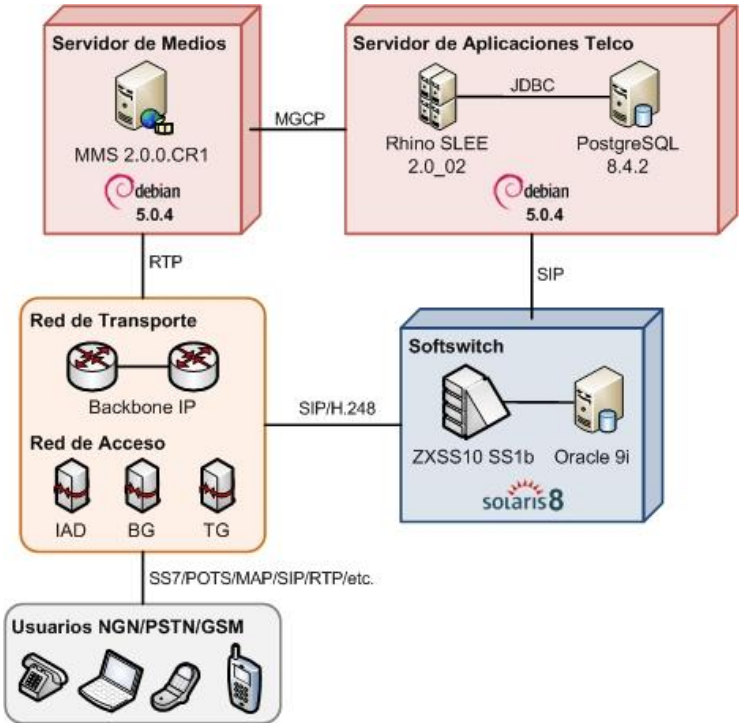


Figura 40 Diagrama de despliegue del sistema solución

- *Softswitch*: nodo de control de la NGN de EMCALI que implementa funciones de procesamiento de control integrado, tales como control de llamadas, adaptación a protocolos de acceso, interconexión e interoperabilidad. El Softswitch ZXSS10 SS1b de ZTE se encuentra instalado sobre un OS Sun Solaris 8, y utiliza una base de datos Oracle9i. Las características de la máquina dependen del contrato de venta del equipo, documento al cual no se tiene acceso. Sin embargo, en [25]se especifican los siguientes parámetros máximos: un BHCA alrededor de 17 millones de llamadas, una capacidad de soportar hasta 1500 puntos de 64K o 100 puntos de 2M de SS7, y un manejo de una variedad de protocolos, como SS7: INAP/CAP/MAP/TUP/ISUP, SIP, SIGTRAN, H.248, MGCP, H.323, entre otros.

- *Servidor de aplicaciones Telco*: nodo que controla el ciclo de vida completo, la lógica de servicio y el almacenamiento de datos, del prototipo CRBT. Establece la conexión SIP con el Softswitch para el manejo de señalización de llamada, y MGCP con el servidor de medios para el control de la reproducción de componentes de audio. Se instaló Rhino SLEE versión 2.2_02 de OpenCloud, sobre un PC de escritorio HP Compaq dc 7800p con Intel Core 2 Duo E8300 @ 2.83GHz y 2GB de Memoria de Acceso Aleatorio (RAM, Random Access Memory). Se utilizó el OS Debian 5.0.4 (lenny), y la Máquina Virtual Java (JVM, Java Virtual Machine) versión 1.6.0_18 Java HotSpot 64-bit Server. El motor de base de datos instalado para este ambiente de ejecución es PostgreSQL 8.4.2, al cual se accede a través de JDBC. En este caso, Rhino SLEE no se implementó en la configuración de nodo por máquina, debido a escasos recursos computacionales y de red.

- *Servidor de medios*: nodo encargado de reproducir el archivo de audio al usuario que inicia la llamada a través del Protocolo de Transporte en Tiempo real (RTP, Real-time Transport Protocol). Se instaló el MMS versión 2.0.0.CR1 de Mobicents, sobre un PC portátil Gateway M-1631U con AMD Turion X2 TL-60 2.0 GHz y 4GB de RAM. Se utilizó el OS Debian 5.0.4 (lenny) y la JVM versión 1.6.0_18 Java HotSpot 64-bit Server.

- *Red de transporte*: proporciona todos los elementos de red del nivel de transporte de la NGN de EMCALI. Entre ellos se destaca el backbone IP (información detallada en el Anexo C).

- *Red de acceso*: proporciona todos los elementos de red del nivel de acceso de la NGN de EMCALI. Entre ellos se destacan el Dispositivo de Acceso Integrado (IAD, Integrated Access Device), la Pasarela de Frontera (BG, Border Gateway) y la Pasarela de Troncal (TG, Trunk Gateway) (información detallada en el Anexo C).

- *Usuarios:* este componente hace referencia a todos los usuarios que interactúan con le red del operador EMCALI. Entre ellos se encuentran usuarios PSTN, NGN, clientes móviles, usuarios web y usuarios SIP.

D.2.5 Vista de componente modulares

Modelo de diseño

Este modelo expone la forma del sistema desde el punto de vista de una organización estructurada de paquetes, Figura 41

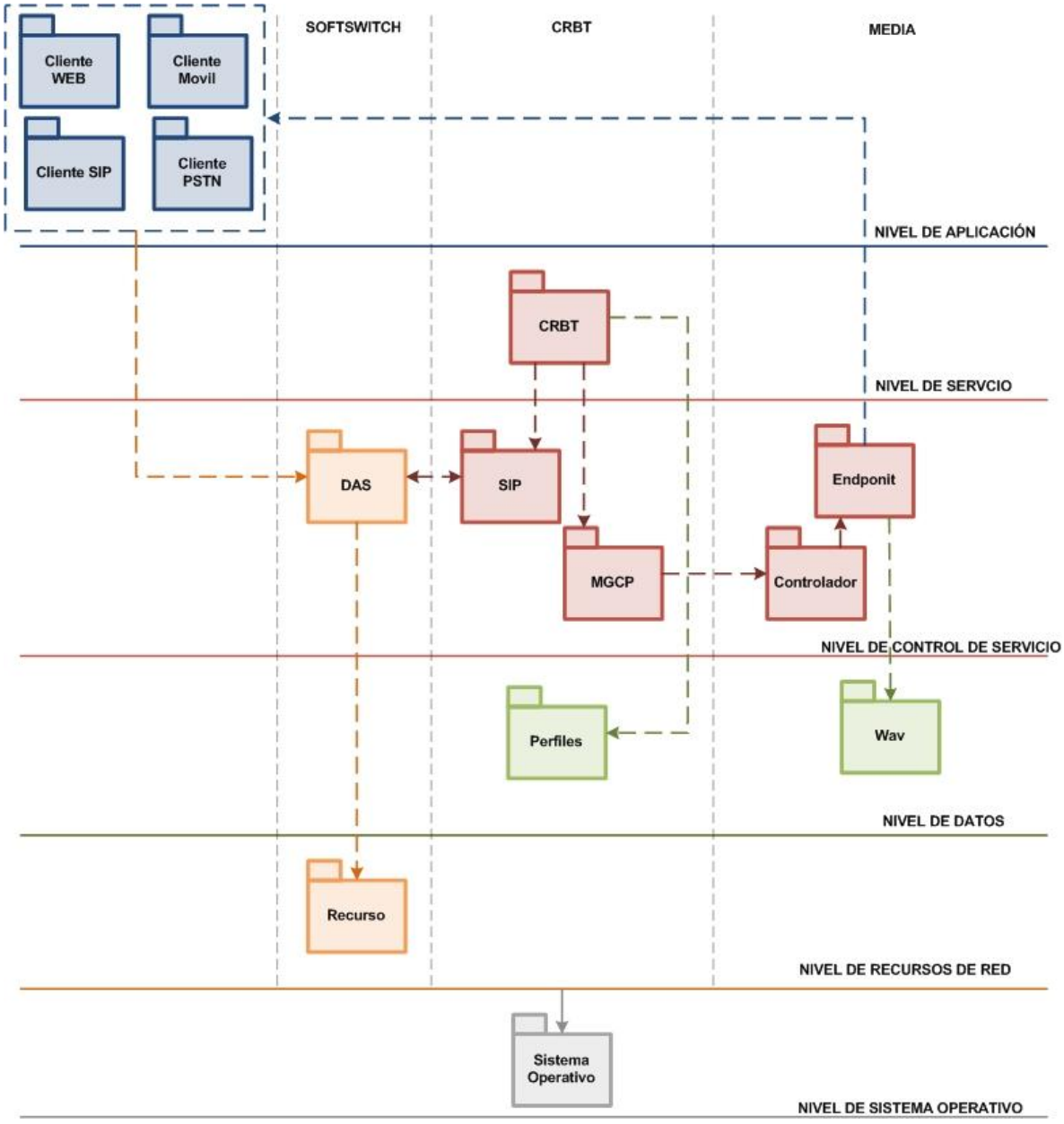


Figura 41 Diagrama de paquetes de diseño del sistema solución

- *Nivel de aplicación:* representa los diferentes usuarios s que acceden a la NGN de EMCALI. Ellos son suscriptores del servicio CRBT o clientes finales.

  - Cliente web: cliente que contiene un software en su PC para realizar llamadas a través de la red IP hacia la NGN de EMCALI

  - Cliente PSTN: usuario de una red PSTN bien sea de ECMALI o cualquier otra que tenga acuerdos con este operador

  - Cliente móvil: cliente asociado a una empresa móvil telecomunicaciones (TIGO, Movistar, COMCEL) que establece conexión con un usuario NGN de EMCALI

- Cliente SIP: usuario que accede desde un teléfono SIP a la NGN de ECMALI

- *Nivel de servicio:* expresa todo los componentes encargados del manejo de aplicación

  - CRBT: contiene las clase que ejecutan la lógica del servicio CRBT y el control del flujo de información entre los di referentes componentes

- *Nivel de control del servicio:* representa los componentes que realizan el manejo de la señalización SIP y MGCP con los nodos de control y de medios respectivamente

  - DAS: contiene la lógica de análisis de dígitos para el direccionamiento de los números destino hacia el nodo del servicio CRBT

  - SIP: contiene las clases que controla la señalización SIP. Estas clases reciben mensajes del nade de control e envían peticiones hacia el nodo del servicio

  - MGCP: contiene las clases que realizan la gestión de los mensajes del protocolo MGCP para la comunicación con el nodo de medios de servicio.

  - Controlador: contiene las clases para recibir solitudes e enviar respuestas a peticionas basadas en el protocolo MGCP.

  - Endpoint: contiene los archivos de configuración para establecer el Protocolo de Transporte en tempo Real (RTP, Real-time Transport Protocol) con los usuarios finales y seleccionar el archivo de audio de un perfil en particular.

- *Nivel de datos:* este unto maneja la persistencia de la información relacionada al servicio y al usuario

  - Perfiles: contiene las clases que incluyen el perfil de suscriptor del servicio

  - Wav: contiene los archivos de audio configurados dentro de los perfiles del suscriptor

- *Nivel de recursos red:* este nivel proporciona los recursos de red necesarios para el aprovisionamiento del servicio

  - Recurso: asigna los recursos más adecuados para manejar la solicitud de algún servicio en particular. Estos recursos varían según la red (NGN, Móvil, PSTN) en la que se envía la solicitud de llamada por parte de cualquier usuario del nivel de aplicación.

- *Nivel de sistema operativo:* en este punto esta contenido el software base para el despliegue de los servicios y componentes del sistema

  - Sistema operativo: hacer referencia a los sistemas Linux como Debain 5.0 y Sun solaris 8 sobre el cual corre tanto el servicio como los componentes de control de la NGN de EMCALI

Anexo E

Instalación y configuración RHINO

El presente anexo proporciona una guía básica para la instalación y configuración de RHINO como Kit de Desarrollo de Software (SDK, Software develompment Kit) en Windows y como SLEE en el sistema operativo Linux.

E.1 RHINO SLEE en Linux

Antes de instalar y configurar RHINO se debe tener en cuenta el hardware y software necesario para la ejecución del servidor JSLEE. Es importante señalar que en caso de realizar pruebas de desempeño, alta disponibilidad y fiabilidad sobre el servidor, es necesario tener equipos hardware con soporte para 64 bits. De esta manera, el sistema operativo, JAVA y Postgres, deberán ser descargados según esta característica. Todo el software[78] requerido deber ser instalado con versiones de 64 bits a fin de realizar pruebas de rendimiento estables y adecuadas.

Requerimientos de hardware, software y sistema operativo

Sistema operativo

La versión de producción de RHINO está oficialmente soportada por Solaris 10 y RedHat AS4. Sin embargo, ha sido desplegado exitosamente sobre otras versiones de Linux. Para el caso especifico del proyecto, la especificación 5.0 de Debian se tomó como sistema operativo base para el desarrollo del prototipo debido a su gran robustez, rendimiento y soporte frente otras versiones gratuitas encontradas en el mercado.

Requisitos hardware

Los requerimientos mostrados en la Tabla 10 corresponden a los recursos solicitados por la especificación 2.1 de RHINO.

Tabla 10 Requisitos hardware

HARDWARE	MÍNIMO	RECOMENDADO
Tipo de Maquina	Producto actual en hardware y CPU	
Numero de Maquinas (un nodo del clúster por maquina)	1	2 o mas
Numero de núcleos CPU por maquina	2	2 a 16
RAM por maquina	1 GB	2+ GB
Interfaz de red	Ethernet	
Requerimientos de interfaz de red por maquina	2 interfaces de 100MB (una interfaz para la comunicación del clúster)	2 o más interfaces de 1 GB (una interfaz para la comunicación del clúster)

Fuente: OpenCloud[79]

[78] Para el desarrollo del prototipo se emplearon las versiones de Linux, Java y PostgreSQL, de 64 bits.
[79] Tomado del documento "*Production Getting Started*" [37].

Requisitos software

- Instalación de Java

RHINO SLEE 2.1 requiere Java SE 5 o 6. Es recomendable usar la versión más actualizada. Esta puede ser descargada desde http://java.sun.com. La instalación solo requiere ejecutar desde la ventana de comandos el archivo .bin descargado y seguir el wizard de configuración. Los pasos mostrados a continuación resume el proceso de instalación

- Ingresar como súper usuario en el Shell
- Ubicarse en la dirección donde esta descargado el instalador
- Ejecutar el archivo .bin ; Esto se hace digitando la sentencia ./Nombre completo del instalador incluida la extensión
- Seguir el wizard de configuración

Configuración de variables de entorno

Para ejecutar cualquier comando java sin ir hasta la dirección de instalación se debe configurar las variables de entorno. Los pasos abajo mencionados describen el proceso

- Ingresar en el Shell (como usuario normal o super usuario)
- Ejecutar la sentencia: nano .bashrc; nano puede cambiarse por gedit o por cualquier editor de texto
- Agregar al archivo abierto el siguiente texto:

  #Java configuration
  export PATH= ruta completa de la carpeta donde se encuentra instalado java/bin:$PATH
  JAVA_HOME= "ruta completa de la carpeta donde se encuentra instalado java" (comillas incluidas)
  export JAVA_HOME

- Guardar cambios
- Desde el Shell ejecutar la sentencia: java -version; deberá aparecer algo como

- Instalación de PostgreSQL

RHINO SLEE requiere de un sistema de gestión de base de datos postgres para almacenar el estado de memoria de trabajo. La memoria de trabajo principal de RHINO contiene el estado de ejecución de los nodos, el estado de configuración de los adaptadores de recursos, perfiles, etc. La base de datos postgres solo suministra un respaldo de la memoria de trabajo del servidor RHINO.

RHINO SLEE ha sido probado sobre las versiones 7.4.*, 8.1.* y 8.3.*. Este gestor de base de datos puede instalarse en una maquina independiente aunque generalmente es instalado sobre el mismo equipo. A continuación se mencionan los pasos para la instalación.

- Descargar el archivo bin de Postgres desde http://www.postgresql.org/download/. (para este caso se uso la versión 8.3.9-1)
- Ingresar como súper usuario en el shell
- Ubicarse en la carpeta donde se encuentra ubicado el instalador
- Ejecutar la sentencia ./nombre completo del instalador incluida la extensión
- Seguir el setup wizard de configuración

Configuración de variables de entorno PostgreSQL

De igual forma que en la configuración de variables de Java, se edita el archivo .bashrc. El texto adicionado debería quedar de la siguiente forma:

#Java configuration
export PATH= ruta completa de la carpeta donde se encuentra instalado java/bin:/ruta completa donde se encuentra instalado Postgres/bin:$PATH
JAVA_HOME= "ruta completa de la carpeta donde se encuentra instalado java" (comillas incluidas)
export JAVA_HOME

Al utilizar las versiones 8 o superiores, la configuración para que Postgres acepte sockets TCP/IP del servidor RHINO no es necesaria, en caso de que se cuente con una menor, el parámetro tcpip_socket del archivo postgresql.conf ubicado en la carpeta DATA debe ser cambiado a tcpip_socket -1.

La configuración aquí expuesta es realizada bajo el supuesto de que Postgres y RHINO han sido instalados en la misma máquina, si por el contrario, la instalación del gestor de base de datos se realiza en otro componente hardware, el archivo pg_hba.conf ubicado en el directorio DATA deberá ser cambiado. Para más información consultar los documentos que vienen con el instalador de RHINO.

Configuración de direcciones IP, nombre de host y direcciones multicast

Antes de instalar el servidor RHINO es necesario configurar los siguientes parámetros. La Tabla 11 expone dicha configuración

Tabla 11 Parámetros de configuración del equipo y de red

CARACTERÍSTICA	DESCRIPCIÓN
Dirección IP	Asegúrese de que el equipo tenga una dirección IP visible en la red
Nombre de Host	Asegúrese de que el sistema pueda resolver (hacer ping) interfaces de loopback (127.0.0.1) y direcciones externas (diferentes a la de localhost)
Direcciones Multicast	Si el sistema local tiene un firewall instalado, modifique los permisos para permitir tráfico multicast UDP Por definición, las direcciones Multicast están el rango 224.0.0.0/4 (224.0.0.0 a 239.255.255.255) . este rango está separado del rango de direcciones Unicast que la

	maquina usa para sus direcciones de host) RHINO SLEE usa Mulicast UDP para distribuir su memoria principal de trabajo entre los miembros del clúster. Durante la instalación, estos parámetros son configurados. Por defecto, el número de puestos requeridos son: 45601, 45602, 46700 a 46800 Todos los nodos en el clúster deben usar la misma dirección multicast. Esto permite que se encuentren visibles. Asegúrese de que el firewall este configurado para permitir mensajes multicast a través de los puertos y direcciones configuradas durante la instalación.
Reloj del sistema	Debido a que RHNO SLEE trabaja de manera síncrona en sus procesos, la configuración del reloj del sistema es prioritaria para este servidor. Encaso de que se tenga varias maquinas, léase los documentos de instalación para configurar el reloj del sistema. Para este caso solo se tienen un equipo, por la que la sincronización del reloj para este caso no es necesaria

Fuente: OpenCloud[80]

Instalación y configuración de RHINO SLEE

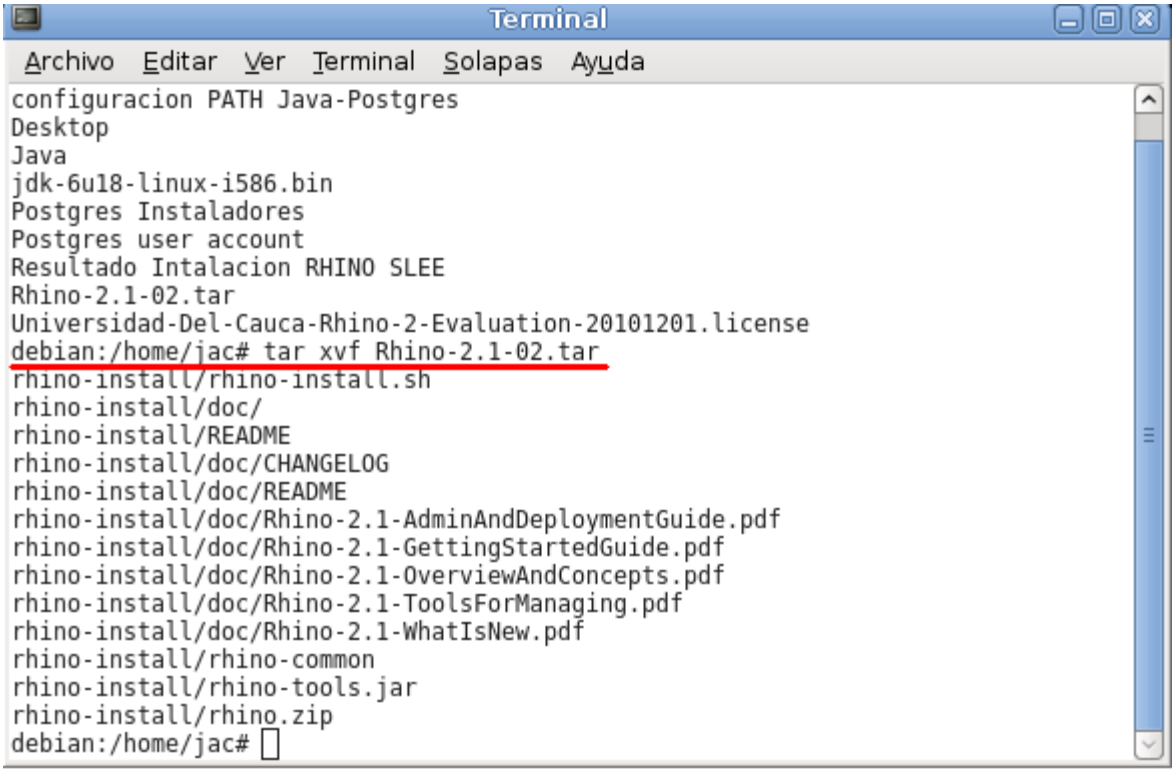
Antes de proceder, es necesario contar con la licencia educativa. Esta licencia es solicitada por escrito a un contacto de Openclud el cual evaluará la solicitud y enviará el link de descarga del servidor y el código de activación en caso de una respuesta afirmativa. La solicitud puede ser diligenciada en la siguiente dirección: https://developer.opencloud.com/devportal/display/OCDEV/Educational+Community+License.

Recuerde que la versión SDK de RHINO y el RHINO SLEE son diferentes. El primero se enfoca en el desarrollo de aplicaciones basadas en JAIN SLEE soportadas por el sistema operativo Windows pero con restricciones en mecanismos contra fallos, configuración en clúster, cantidad de tráfico, entre otras. El segundo, es un servidor en ambiente de producción igualmente soportado por JAIN SLEE pero con la única restricción de volumen de eventos soportados, muchos más que la versión SDK pero menos que la versión licenciada de producción.

Los pasos mencionados a continuación corresponden al proceso de instalación y configuración del servidor RHINO en el sistema operativo debían 5.0

- Ingresar como súper usuario en el Shell
- Ubicarse en la carpeta donde se encuentra el archivo .tar de Rhino
- Ejecutar la sentencia: tar xvf nombre completo del archivo Rhino incluida la extensión. Deberá aparecer algo como:

[80] Tomado del documento "*Production Getting Started*" [37].



- Ubicarse en la carpeta generada al descomprimir el archivo. La carpeta tiene como nombre rhino-install
- Ejecutar el script rhino-install. Recuerde que los archivos .sh .bin y demás son lanzados con la sentencia ./nombre del archivo incluida la extensión. En esto caso sería algo como: ./rhino-instal.sh
- Una vez ejecutado el comando, el script lanza un archivo el cual deber ser configurado según lo solicitado. La Tabla 12 contiene los parámetros de configuración para el servidor RHINO

Tabla 12 Configuración de parámetros de RHINO SLEE

PARÁMETRO	CONFIGURACIÓN POR DEFECTO	DESCRIPCIÓN
Directorio	/RHINO	Ruta del directorio raíz de RHINO.
Postgres Host	Localhost	Dirección IP donde se instalo el servidor Postgres
Puerto Postgres	5432	Puerto de comunicación del servidor Postgres
Postgres User	User	Nombre del usuario que se definió al instalar Postgres (en la mayoría de los casos se instala el servidor Postgres como postgres/postgres-)
Nombre base de datos	rhino	Sera creada en el servidor postgres y configurada con las tablas por defecto del RHINO
Puerto de registro de la interfaz gestión RMI	1199	Se usa para acceder a la gestión desde un cliente Java RMI a fin de usar líneas de comando RHINO
Puerto de objetos de la interfaz de gestión RMI	1200	Se usa para acceder a la gestión desde un cliente Java RMI a fin de usar líneas de comando RHINO

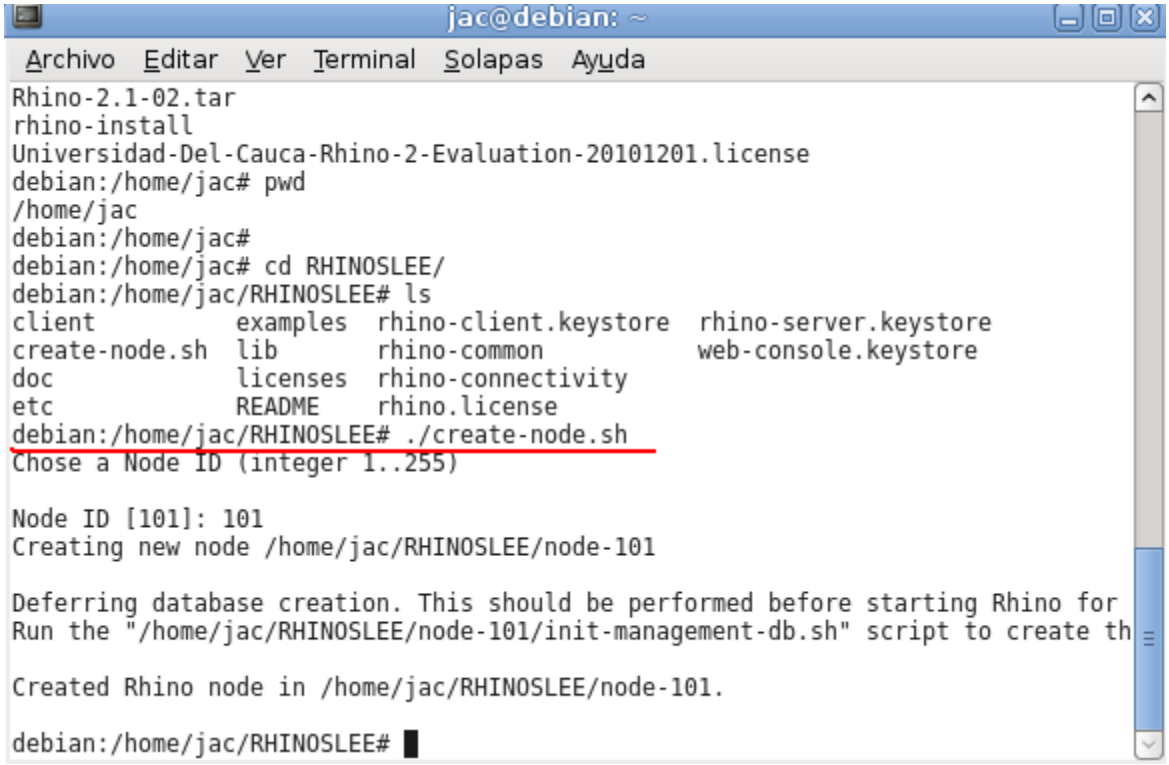
Puerto de servicio remoto para interfaz JMX	1202	Se usa para acceder al servidor remoto JMX. La consola web de rhino usa esto para gestión remota
Puerto de para comunicación HTPPS	8443	Se usa para la consola web y suministrar gestión remota
Directorio JAVA		Directorio de instalación de java JSE/JDK.
Tamaño de heap	512	Argumento dentro del la JVM para especificar la cantidad de memoria (en MegaBytes) para que RHINO se ejecute
Clúster ID	100	Un único numero entero que identifica al clúster
Comienzo del pool de direcciones	224.0.24.1	Conjunto de direcciones multicast que serán usadas para la comunicación
Fin del pool de direcciones	224.0.24.8	
Ubicación del cliente psql	psql	RHINO necesita del cliente interactivo PostgrSQL, psql. Debe ir el PATH completo del cliente psql

Fuente: OpenCloud[81]

En caso de que cometer un error en la configuración o se desee hacer algún cambio es necesario modificar el archivo config_variables ubicado en la dirección $RHINO_HOME/etc/defaults/config

Crear nodos del clúster

- Ubicarse en la dirección base de RHINO
- Ejecutar el script create-node.sh. deberá aparecer algo como:



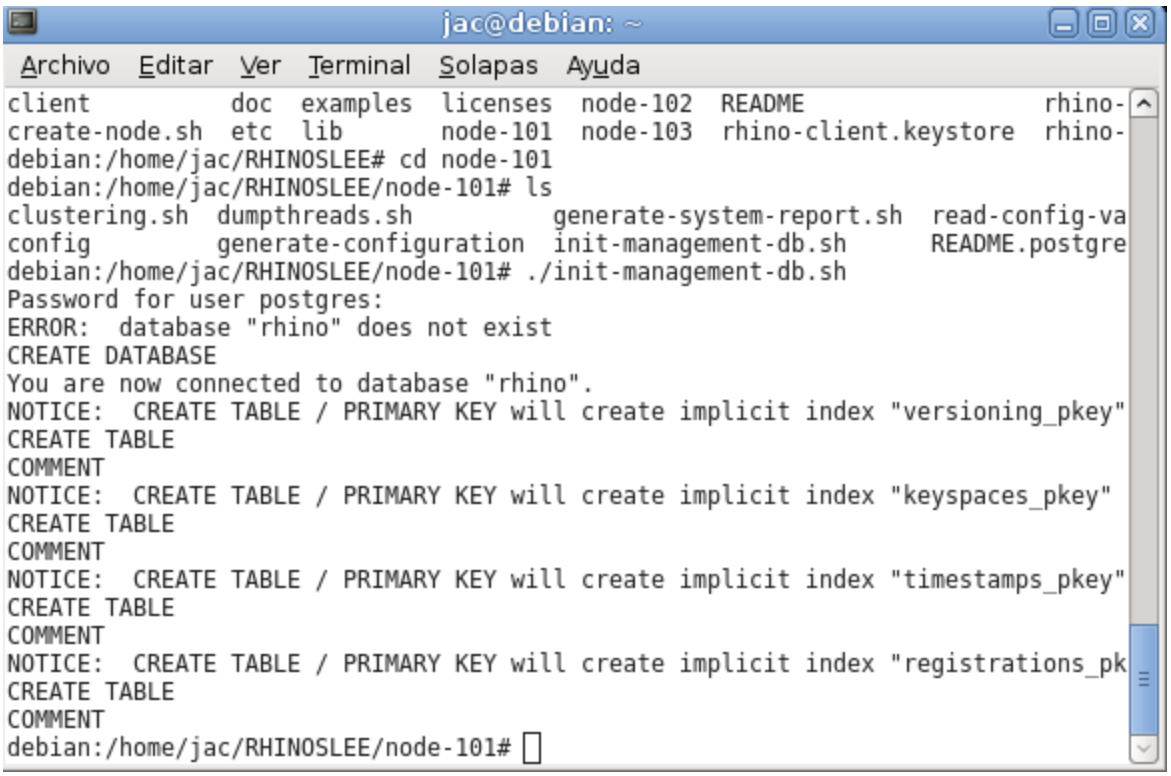
[81] Tomado del documento "*Production Getting Started*" [37].

Repetir lo anterior para crear tantos nodos como desee. Para este caso específico se han creado tres nodos identificados como node-101, node-102, node-103 los cuales formaran el clúster de tres servidores.

Inicializar la base de datos

Los siguientes pasos mencionados son para la creación de la base de datos que RHINO necesita para sincronizar la memoria de trabajo entre los nodos y del almacenamiento de perfiles, entre otras características

- Ubicarse en la dirección del que será el nodo principal. En este caso el nodo con id 101
- Inicializar el servidor postgres. Esto se hace desde la pestaña de aplicaciones en caso de que se halla instalado cualquier ambiente grafico como GNOME. De lo contrario, lanzarlo por el Shell con el comando start
- Dentro de node-NNN ejecutar el script init-management-db.sh; al dar los parámetros solicitados deberá salir algo como:



El script crea la base de datos que necesita el servidor para guardar el estado de trabajo. Esta inicialización de la base de datos solo se corre una vez. En caso de que los nodos sean eliminados es necesario ejecutarlo de nuevo desde un solo nodo.

Arrancar el servidor RHINO

A continuación se describe el procedimiento para arrancar los diferentes nodos que conformaran el clúster.

- Ubicarse en la dirección del nodo principal
- Ejecutar el script start-rhino.sh con los parámetros -p -s -k. (./start-rhino.sh -p -s -k). para más información acerca de los parámetros referirse a la documentación de RHINO o abrir el script con cualquier editor de texto
- Acceder como súper usuario desde otro Shell a la dirección del nodo 2.
- Ejecutar el script start-rhino con parámetros -s -k (./start-rhino.sh -s -k)
- Acceder como súper usuario desde otro Shell a la dirección del nodo 3.
- Ejecutar el script start-rhino con parámetros -q -k
- Si todo ha salido bien, en el terminal de cualquier de los nodos deberá aparecer algo como: current membership set: [101,102,103]

Prueba de servicio de llamada

Una vez lanzados los nodos del clúster se procederá a desplegar las diferentes unidades que componen los servicios de Register, Call y SIP Proxy

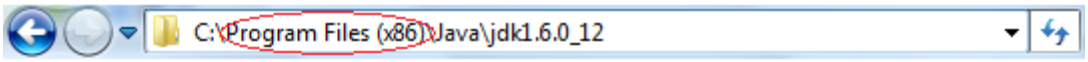
- Ingresar como súper usuario
- Ubicarse en $RHINO_HOME/examples/sip-examples-2.2_02
- Modificar el parámetro PROXY_DOMAINS dentro del archivo sip.properties por la dirección IP del servidor RHINO
- Ejecutar el script deployesexamples.sh
- Si todo sale bien, deberá salir algo como BUILD SUCCESSFULL
- Instalar cualquier softphone y configurarlo con la dirección del servidor RHINO
- Realizar la llamada entre usuarios

E.2 RHINO SDK en Windows

Antes de instalar la versión SDK de RHINO se debe realizar la configuración de Java, Postgres y tener en cuenta requisitos mínimos de memoria de la Tabla 6. En este caso se uso un equipo con 4 GB de memoria y sistema operativo Windows Vista

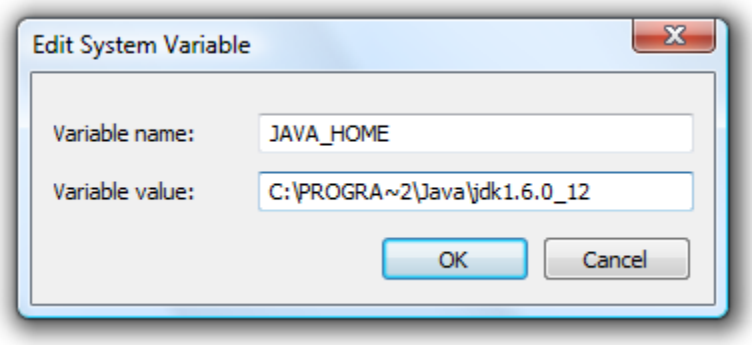
- Configuración Java:

  - JDK 6 actualización 12 fue la versión usada (recuerde que RHINO funciona para la versión 5 o 6.).
  - Crear variable de entorno JAVA_HOME con la ruta de instalación de Java. Si existen espacios en dicha ruta, escribir el nombre corto generado por non-8dot3 para la carpeta o archivo en específico (utilizar comando dir /x dentro de la carpeta que lo contiene). En este caso la ruta de instalación de Java es C:/Program Files (x86)/Java/jdk1.6.0_12:

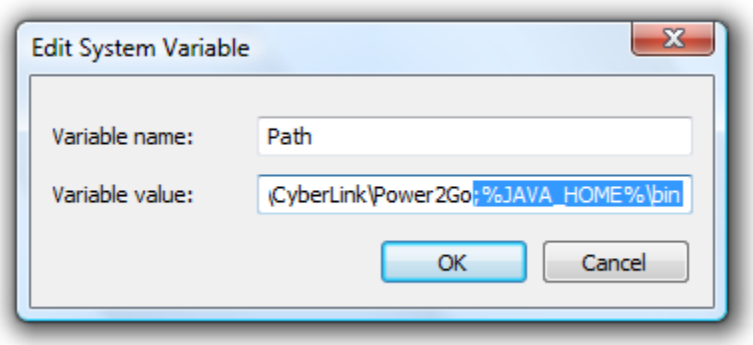


```
C:\>dir /x
 Volume in drive C has no label.
 Volume Serial Number is 2C24-2859

 Directory of C:\
    .
    .
11/08/2009  05:40 p.m.    <DIR>          PROGRA~1      Program Files
15/12/2009  06:52 p.m.    <DIR>          PROGRA~2      Program Files (x86)
    .
    .
```



- Editar la variable de entorno Path, agregando al final del campo valor de variable el texto ;%JAVA_HOME%/bin:



- Verificar que las variables de entorno se encuentran configuradas correctamente, utilizando los comandos java -version y javac -version en cualquier ubicación del equipo:

```
C:\>java -version
java version "1.6.0_12"
Java(TM) SE Runtime Environment (build 1.6.0_12-b04)
Java HotSpot(TM) Client VM (build 11.2-b01, mixed mode, sharing)

C:\>javac -version
javac 1.6.0_12
```

- Configuración de red:

  - Asegurar que el sistema tiene una IP visible en la red.
  - Asegurar que el sistema puede resolver localhost a la interfaz de loopback utilizando el comando ping -4 localhost:

```
C:\>ping -4 localhost

Pinging PIPE-PC [127.0.0.1] with 32 bytes of data:
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128

Ping statistics for 127.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

- Asegurar que el sistema resuelve el nombre de la máquina a una IP externa, no a la dirección de loopback, utilizando el comando ping -4 NombreMáquina (en este caso el nombre de la máquina es PIPE-PC):

```
C:\>ping -4 PIPE-PC

Pinging PIPE-PC [169.254.77.51] with 32 bytes of data:
Reply from 169.254.77.51: bytes=32 time<1ms TTL=128
Reply from 169.254.77.51: bytes=32 time<1ms TTL=128
Reply from 169.254.77.51: bytes=32 time<1ms TTL=128
Reply from 169.254.77.51: bytes=32 time<1ms TTL=128

Ping statistics for 169.254.77.51:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

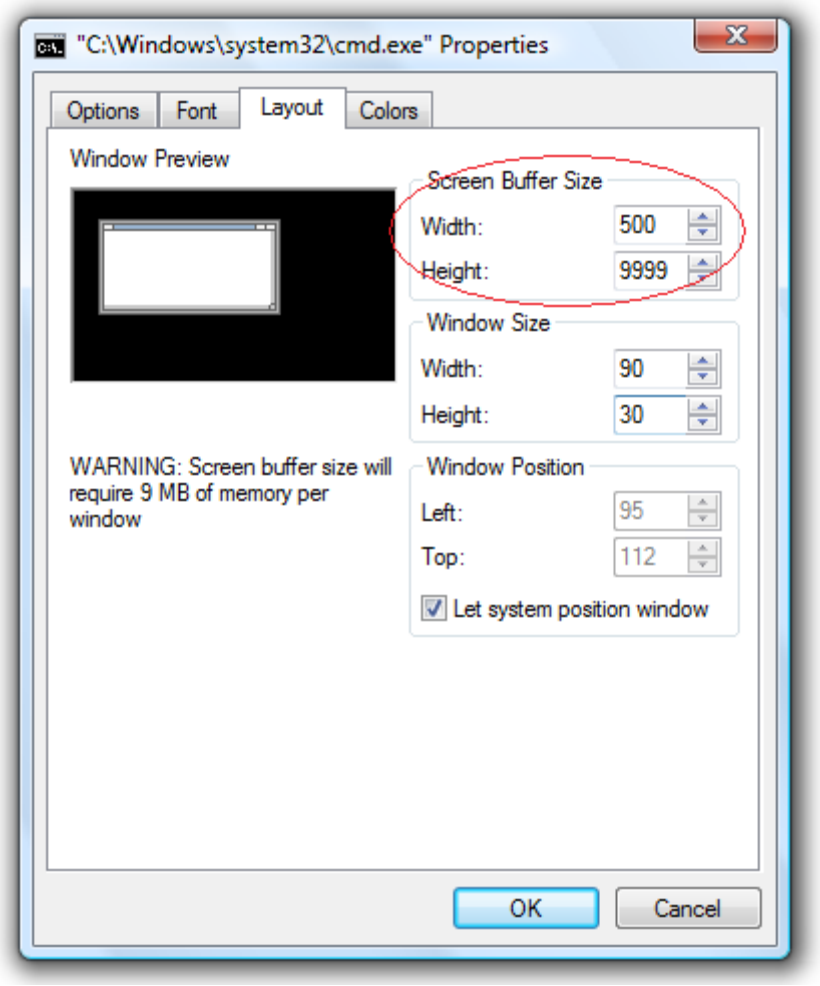
Desempaquetando Rhino SDK

- Descomprimir el archivo zip en $RHINO_HOME, que en este caso es C:/RhinoSDK.
- Leer las notas de la presente versión de Rhino SDK:

  - $RHINO_HOME/doc/index.html >> contiene información de último momento.
  - $RHINO_HOME/doc/CHANGELOG >> resume los cambios realizados entre las versiones previas de Rhino y la presente.

Ejecución de Rhino SDK

- Modificar visibilidad de la ventana de comandos que ejecutará Rhino SDK: clic derecho en la barra de título de la ventana y seleccionar Properties. Luego, en la pestaña Layout, en la sección Screen Buffer Size, modificar el valor de Width a 500 y de Height a 9999, u otros valores que permitan observar mayores registros de Rhino SDK (Precaución: el tamaño del buffer necesitará 9 MB de memoria por ventana, así que sólo realizar esta modificación con la consola de comandos que ejecuta Rhino):



- Iniciar Rhino SDK: en consola ejecutar start-rhino.bat dentro de la carpeta $RHINO_HOME.

- Detener Rhino SDK: en otra consola ejecutar stop-rhino.bat –-[nice|terminate] dentro de la carpeta $RHINO_HOME.

- Utilizar PostgreSQL (en el SDK también es posible utilizar Derby), ya que es la base de datos que utiliza la versión de producción de Rhino:

  - Instalar PostgreSQL v8.4.2 en $POSTGRESQL_HOME, que en este caso es C:/Program Files (x86)/PostgreSQL/8.4.
  - Digitar postgres como contraseña para el súper-usuario postgres y cuenta de servicio postgres.
  - Utilizar el puerto 5432 para el servicio.
  - El servidor acepta por defecto conexiones socket TCP/IP, ya que es una versión posterior a la 8.0. Verificar en $POSTGRESQL_HOME/data/postgresql.conf que el servidor se encuentre configurado de forma que escuche todas las direcciones IP: listen_addresses = '*'.
  - El servidor tiene por defecto configurado el control de acceso para las conexiones locales, necesario para el caso en donde Rhino y PostgreSQL se instalan en el mismo equipo.

Verificar en $POSTGRESQL_HOME/data/pg_hba.conf que se encuentre una configuración similar a la siguiente:

```
# TYPE  DATABASE  USER     CIDR-ADDRESS        METHOD
# IPv4 local connections:
host    all       all      127.0.0.1/32        md5
# IPv6 local connections:
host    all       all      ::1/128             md5
```

NOTA – si Rhino y PostgreSQL son instalados en diferentes máquinas, se debe editar este mismo archivo y adicionar a la configuración, descrita anteriormente, la base de datos y el usuario de la conexión a realizar.

- Si se han realizado cambios a los archivos de configuración de PostgreSQL mencionados, es necesario reiniciar completamente el servidor PostgreSQL para aplicar dichas modificaciones. Decirle al servidor que cargue nuevamente los archivos de configuración no habilita el funcionamiento de red TCP/IP como lo es cuando la base de datos es inicializada.

- Editar $RHINO_HOME/config/config_variables de la siguiente forma:

```
# Management database settings
MANAGEMENT_DATABASE_NAME=rhino_sdk
MANAGEMENT_DATABASE_HOST=localhost
MANAGEMENT_DATABASE_PORT=5432
MANAGEMENT_DATABASE_USER=postgres
MANAGEMENT_DATABASE_PASSWORD=postgres
```

- En $RHINO_HOME/config/rhino-config.xml comentar la configuración Derby: eliminar el final de comentario (-->) que aparece en la línea:

```
<!-- Begin Derby-specific configuration
To use PostgreSQL rather than the embedded Derby database, you'll want to comment out this section and uncomment the PostgreSQL section below. -->
```

, y eliminar el inicio de comentario (<!--) que aparece en la línea:

```
<!-- End of Derby-specific database. -->
```

- En $RHINO_HOME/config/rhino-config.xml descomentar la configuración PostgreSQL: adicionar el final de comentario (-->) inmediatamente después de la línea:

```
<!-- From here on is the configuration for PostgreSQL if you want to use that instead.
-->
```

, y adicionar el inicio de comentario (<!--) inmediatamente antes de la línea:

```
<!-- End PostgreSQL-specific section -->
```

Inicializar la base de datos de PostgreSQL:

En consola ejecutar init-management-db.bat postgres (de ahora en adelante siempre se necesita pasar la palabra postgres a este archivo lote):

```
C:\RhinoSDK>init-management-db.bat postgres
Creating database...
Using PostgreSQL 8.4.2
ERROR: no existe la base de datos ½rhino_sdk¶
   .
   .
   .
CREATE DATABASE
Using PostgreSQL 8.4.2
Database has been initialised.
Press any key to continue . . .
```

Salida cuando se inicializa la base de datos por primera vez.

```
C:\RhinoSDK>init-management-db.bat postgres
Creating database...
Using PostgreSQL 8.4.2
DROP DATABASE
CREATE DATABASE
Using PostgreSQL 8.4.2
Database has been initialised.
Press any key to continue . . .
```

Salida cuando la base de datos ya ha sido inicializada anteriormente.

Al inicializar la base de datos se crean las tablas que utiliza Rhino SLEE y la tabla usada por el ejemplo de conectividad del protocolo SIP (RA y servicios) proveído por OpenCloud. Los comandos SQL ejecutados en esta acción, teniendo en cuenta que la base de datos utilizada es PostgreSQL, se muestran a continuación:

```
create table versioning (
  name text not null primary key,
  application text not null,
  ocbb text not null,
  description text not null,
  version integer not null
);
GO
CREATE TABLE keyspaces (
  dbid text not null,
  key_id text not null,
  mode int4 not null,
  timeout int4 not null,
  table_name text not null,
  primary key (dbid, key_id, mode)
);
GO
CREATE TABLE timestamps (
  dbid text not null primary key,
  era int8 not null,
  last_update int8 not null
);
GO
create table registrations ( -- for SIP location service
  sipaddress varchar(80) not null,
  contactaddress varchar(80) not null,
  expiry bigint,
```

```
  qvalue integer,
  cseq integer,
  callid varchar(80),
  flowid varchar(80),
  primary key (sipaddress, contactaddress)
);
GO
```

Iniciar Rhino SDK con la nueva configuración de base de datos PostgreSQL. Nuevamente en consola ejecutar start-rhino.bat dentro de la carpeta $RHINO_HOME.

BIBLIOGRAFÍA

[1] OMA, "About OMA," *OMA*, 2010. [En línea]. Disponible: http://www.openmobilealliance.org/AboutOMA/Default.aspx. [Consultado: Agosto 26, 2009].

[2] OMA Architecture Working Group (ARC), "OMA Service Environment," OMA, Reporte OMA-AD-Service-Environment-V1_0_4-20070201-A, Febrero 1, 2007. [En línea]. Disponible: http://member.openmobilealliance.org/ftp/public_documents/arch/ARC-OSE/2007/. [Consultado: Agosto 25, 2009].

[3] Sun Microsystems, Inc., "JAIN and Java in Communications," Sun Microsystems, Inc., Santa Clara, California, Estados Unidos, Reporte de Referencia, Marzo 2004. [En línea]. Disponible: http://java.sun.com/products/jain/reference/docs/Jain_and_Java_in_Communications-1_0.pdf. [Consultado: Agosto 26, 2009].

[4] ETSI TISPAN, "Open Service Access (OSA);Application Programming Interface (API);Part 1: Overview (Parlay 6)," ETSI, Estándar ETSI ES 204 915-1 V1.1.1, Mayo 2008. [En línea]. Disponible: http://pda.etsi.org/pda/home.asp?wki_id=y2y_jozhXnACCJBJsnHkX. [Consultado: Agosto 27, 2009].

[5] ETSI 3GPP, "Universal Mobile Telecommunications System (UMTS);Virtual Home Environment (VHE)/Open Service Access (OSA)," Especificación Técnica ETSI TS 123 127 V6.1.0, Junio 2006. [En línea]. Disponible: http://www.3gpp.org/ftp/Specs/html-info/23127.htm. [Consultado: Agosto 27, 2009].

[6] ETSI TISPAN, "Open Service Access (OSA);Application Programming Interface (API);Part 3: Framework(Parlay 6)," Estándar ETSI ES 204 915-3 V1.1.1, Mayo 2008. [En línea]. Disponible: http://pda.etsi.org/pda/home.asp?wki_id=%27IsIT8inGUmopqmrsJsFG. [Consultado: Agosto 27, 2009].

[7] P. A. Muller, et al., "Service Oriented Architectures for convergent Service Delivery Platforms. Applying Service Oriented Architectures to Service Delivery Platforms," EURESCOM, Reporte Técnico EDIN 0532-1652, Diciembre 2006. [En línea]. Disponible: http://www.eurescom.de/public/projectresults/P1600-series/P1652-D2.asp. [Consultado: Octubre 28, 2009].

[8] Oracle Communications, "Oracle Communications Converged Application Server," (hojadedatos) Oracle, Hoja de Datos, 2009. [En línea]. Disponible: http://www.oracle.com/us/industries/communications/045484.pdf. [Consultado: Enero 19, 2010].

[9] Moriana Group, "Service Delivery Platforms – definition and evolution," *MORIANAGROUP.com*, 2010. [En línea]. Disponible: http://www.morianagroup.com/index.php?option=com_content&view=article&id=357&Itemid=190. [Consultado: Enero 22, 2010].

[10] Nokia Corporation Networks, "Service Delivery Platforms," Nokia, Finlandia, Reporte de Referencia, 2005. [En línea]. Disponible: http://www.nokia.com/NOKIA_COM_1/About_Nokia/Press/White_Papers/pdf_files/whitepaper_service_delivery.pdf. [Consultado: Marzo 17, 2009].

[11] OpenCloud, "Rhino Core Platform," (hojadedatos) OpenCloud, Hoja de Datos. [Consultado: Noviembre 19, 2009].

[12] OpenCloud, "A service delivery platform for next generation telecommunications services," OpenCloud, Wellington, Nueva Zelanda, Reporte de Referencia, 2009. [En línea]. Disponible: http://www.opencloud.com/documents/OpenCloud%20Product%20Brochure.pdf. [Consultado: Noviembre 24, 2009].

[13] P. Falcarin y C. A. Licciardi, "Analysis of NGN service creation technologies," en *International Engineering Consortium (IEC) Annual Review of Communication*, vol. 56, Junio 2003, pp. 537-551.

[14] P. Chainho, et al., "Next Generation Networks: the service offering standpoint. Requirements analysis and architecture definition," EURESCOM, Reporte Técnico EDIN 0241-1109, Noviembre 2001. [En línea]. Disponible: http://www.eurescom.de/public/projectresults/P1100-series/1109TI1.asp. [Consultado: Octubre 28, 2009].

[15] N. Kryvinska, C. Strauss, L. Auer, y P. Zinterhof, "Conceptual Framework for Services Creation/Development Environment in Telecom Domain," en *10th International Conference on Information Integration and Web-based Applications & Services (iiWAS) 2008*, Linz, Austria, Noviembre 2008, pp. 324-331.

[16] J. Zuidweg, "Middleware en Telecomunicaciones," *Tecsidel Tic*, Marzo 16, 2009. [En línea]. Disponible: http://www.tecsidel.es/tecsidel/index.php?id=896&L=2. [Consultado: Noviembre 18, 2009].

[17] A. Ryan, et al., "Service Oriented Architectures for convergent Service Delivery Platforms. Service Oriented Architecture and Telcos," EURESCOM, Reporte Técnico EDIN 0531-1652, Septiembre 2006. [En línea]. Disponible: http://www.eurescom.de/public/projectresults/P1600-series/P1652-D1.asp. [Consultado: Octubre 28, 2009].

[18] C. Venezia y P. Falcarin, "Communication Web Services Composition and Integration," en *IEEE International Conference on Web Services (ICWS) 2006*, Chicago, Illinois, Estados Unidos, Septiembre 2006, pp. 523-530.

[19] SunMicrosystems, Inc., "The Java EE 5Tutorial for Sun Java System Application Server 9.1," (tutorial) SunMicrosystems, Inc., Tutorial, Octubre 2008. [En línea]. Disponible: http://java.sun.com/javaee/downloads/index.jsp.

[Consultado: Diciembre 16, 2009].

[20] C. Chrighton, D. T. Long, y D. C. Page, "JAIN SLEE vs SIP Servlet.Which is the Best Choice for an IMS Application Server?," en *Australasian Telecommunication Networks and Applications Conference*, Christchurch, New Zealand, Diciembre 2007, pp. 448-453.

[21] OpenCloud, "Rhino 2.1: Overview and Concepts," OpenCloud, Wellington, Nueva Zelanda, Reporte Técnico, Marzo 30, 2009. [En línea]. Disponible: https://developer.opencloud.com/devportal/display/RD/Rhino+Overview+and+Concepts. [Consultado: Octubre 23, 2009].

[22] OpenCloud, "A SLEE for all Seasons," OpenCloud, Wellington, Nueva Zelanda, Reporte de Referencia, Mayo 11, 2007. [En línea]. Disponible: http://www.opencloud.com/documents/Whitepaper%20A%20SLEE%20for%20all%20Seasons.pdf. [Consultado: Diciembre 2, 2009].

[23] M. Femminella, et al., "Scalability and performance evaluation of a JAIN SLEE-based platform for VoIP services," en *21st International Teletraffic Congress (ITC) 2009*, París, Francia, Septiembre 2009, pp. 1-8.

[24] R. Lasch, B. Ricks, y R. Tönjes, "Service Creation Environment for Business-to-Business Services," en *International Conference on Advanced Information Networking and Applications Workshops (WAINA) 2009*, Bradford, Yorkshire del Oeste, Inglaterra, Reino Unido, Mayo 2009, pp. 512-517.

[25] ZTE Corporation, "ZXSS10 SS1b SoftSwitch Control Equipment Technical Manual," ZTE Corporation, Shenzhen, China, Manual Técnico Versión V2.01.50, Julio 5, 2007.

[26] IBM Corporation, "IBM Service Provider Delivery Environment Framework," IBM Corporation, Somers, New York, Estados Unidos, Reporte de Referencia, Febrero 2009. [En línea]. Disponible: http://www-01.ibm.com/software/industry/telecommunications/. [Consultado: Enero 20, 2010].

[27] Microsoft Corporation, "Connected Services Framework 3.0," , 2006. [En línea]. Disponible: http://msdn.microsoft.com/es-co/library/aa439670%28en-us%29.aspx. [Consultado: Enero 21, 2010].

[28] Ericsson, "Ericsson Service Delivery Platform," Ericsson, Reporte de Referencia, Abril 27, 2005. [En línea]. Disponible: http://archive.ericsson.net/service/internet/picov/get?DocNo=28701-FGB101265. [Consultado: Enero 22, 2010].

[29] Nokia Siemens Networks Corporation, "Service Delivery Framework," Nokia Siemens Networks, Reporte de Referencia, 2008. [En línea]. Disponible: http://www.nokiasiemensnetworks.com/portfolio/services/application-and-systems-integration/applications-solutions. [Consultado: Enero 22, 2010].

[30] ZTE Corporation, "ZXMC SDP Product Description," ZTE Corporation, Shenzhen, China, Especificación Técnica Versión V1.1, Febrero 9, 2009.

[31] M. Unmehopa, K. Vemuri, y A. Bennett, *Parlay/OSA : from standards to reality*. Chichester, Sussex del Oeste, Inglaterra, Reino Unido: John Wiley & Sons Ltd, 2006.

[32] P. Chainho, et al., "Next Generation Networks: the service offering standpoint. Service creation analysis in an NGN context," EURESCOM, Reporte Técnico EDIN 0242-1109, Noviembre 2001. [En línea]. Disponible: http://www.eurescom.de/public/projectresults/P1100-series/1109TI2.asp. [Consultado: Octubre 28, 2009].

[33] S. Lim, P. O'Doherty, D. Ferry, y D. Page, "JAIN SLEE Tutorial. Introducing JAIN SLEE," (tutorial) Sun Microsystems, Inc-OpenCloud, Tutorial, 2003. [Consultado: Diciembre 3, 2009].

[34] JAINSLEE.ORG, "Downloads," *JAINSLEE.ORG*, 2008. [En línea]. Disponible: http://jainslee.org/downloads.html. [Consultado: Diciembre 1, 2009].

[35] JAINSLEE.ORG, "JAIN SLEE and OSA/Parlay," *JAINSLEE.ORG*, 2008. [En línea]. Disponible: http://www.jainslee.org/othertechnologies/osaandslee.html. [Consultado: Diciembre 1, 2009].

[36] ZTE Corporation, "ZXUP10 System introduction," ZTE Corporation, Shenzhen, China, Especificación Técnica Versión V3.63, 2008.

[37] OpenCloud, "Rhino (Production) Getting Started Guide," OpenCloud, Wellington, New Zealand, Manual Técnico, Marzo 31, 2009. [En línea]. Disponible: https://developer.opencloud.com/devportal/display/RD/Rhino+%28Production%29+Getting+Started+Guide. [Consultado: Octubre 23, 2009].

[38] Oracle Communications, "Oracle Communications Services Gatekeeper," (hojadedatos) Oracle, Hoja de Datos, 2009. [En línea]. Disponible: http://www.oracle.com/us/industries/communications/045533.pdf. [Consultado: Enero 19, 2010].

[39] Mobicents, "Introduction," *Mobicents*, 2008. [En línea]. Disponible: http://www.mobicents.org/slee/intro.html. [Consultado: Enero 22, 2010].

[40] Oracle, "An Introduction to the BEA WebLogic Communications Platform," *Oracle*, Febrero 2, 2006. [En línea]. Disponible: http://www.oracle.com/technology/pub/articles/dev2arch/2006/02/communications-platform.html. [Consultado: Enero 21, 2010].