CLIENTE MÓVIL SIP PARA UN SISTEMA IVR BASADO EN LA ARQUITECTURA DE SERVICIOS IMS

JOSÉ FERNANDO MUÑOZ BERMEO
XIMENA VELASCO MELO

ANEXO C

HERRAMIENTAS PARA EL DESARROLLO DE LOS MÓDULOS QUE COMPONEN LA ARQUITECTURA IMS

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telemática
Popayán
2006

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

C1. INTRODUCCIÓN

El presente anexo tiene la finalidad de exponer las tecnologías y herramientas que se utilizaron para implementar los módulos necesarios que componen la arquitectura IMS para la ejecución de servicios IVR.

Para el proceso de selección de las diferentes tecnologías y herramientas, se tuvo en cuenta que: permitiera el desarrollo de los requerimientos planteados, se tratara de software libre, aunque tuviera un costo mínimo posible, se ejecutara en Windows XP, contara con una buena documentación de instalación y uso, facilitara ejemplos de aplicaciones previamente realizadas.

En la sección C2 se describen las características del Servidor de Aplicaciones SIP seleccionado, en la sección C3 se detallan algunas de las características más importantes de los APIs para la creación de la lógica del servicio, Servlet SIP y JavaMail. En la sección C4, se habla de la herramienta y de las clases más importantes que se utilizaron para la creación del módulo MRF, y en la sección C5 se habla acerca del API que se utilizó para la realización de las pruebas de validación del Cliente Móvil, el Servidor de Aplicaciones y el módulo MRF.

C2. SERVIDOR DE APLICACIONES SIP SIPMethod [1]

El 3GPP define un Servidor de Aplicaciones SIP como la entidad que permite el desarrollo y despliegue de nuevos servicios.

SIPMethod es un servidor de aplicaciones SIP que se basa en la JSR-116 (API de Servlets SIP), el RFC-3261 (Especificación de SIP) y soporta las extensiones claves de SIP. Está diseñado para el desarrollo y despliegue de aplicaciones basadas en SIP con la tecnología de Servlets SIP. SIPMethod se basa en una arquitectura de microkernel que provee el rendimiento máximo, robustez, y capacidades de fléxibilidad y extensibilidad.

Este servidor viene en 2 ediciones: la estándar y la edición para proveedores de servicios.

C2.1. REQUERIMIENTOS TÉCNICOS

Los elementos técnicos esenciales que se encuentran en la edición estándar se describen a continuación:

Estandares de Java	Servlet API 2.4, SIP Servlet API 1.0, JMX 1.2
Estandares de IETF	RFC 2246, 2327, 2616, 2617, 2782, 2806, 2976, 3261, 3263, 3265, 3310, 3428, y 3515
Plataformas soportadas	
Sistemas operativos	Windows, Linux y Solaris
SoporteHW	Intel, HP, IBM, Sun, y otros
Java Development Kit	JDK 1.4.2 o posterior
Protocolo y Conectividad	
Protocolos	SIP, SDP, HTTP
Transporte	TCP, UDP, TLS
Cortafuegos Transversal	Entunelamiento TCP, STUN, TURN
Gestión	
Consola Web	Configuración, Monitoreo y Gestión
Gestión de Red	SNMP y otros conectores JMX 1.2 compatibles
Seguridad	
Seguridad de transporte	TLS tanto para SIP y HTTP
Autenticación	Básica, Digest, Certificate
Integración	
Aplicaciones SIP / HTTP	Soporte de aplicaciones que utilizan Servlets SIP / HTTP
Aplicaciones J2EE application	Soporte JNDI, RMI o integración SOAP
Flexibilidad y extensibilidad	
Microkernel	Componentes de arquitectura Plug-n-play
Extensiones	Soporte de Filtros, transportes, etc.

Tabla C1: Requerimientos técnicos del Servidor de Aplicaciones SIPMethod

C2.2. ENTORNO DE CREACIÓN DE APLICACIONES SIPMethod

SIPMethod ACE le permite a los desarrolladores la creación rápida y el despliegue de aplicaciones basadas en Servlets SIP. Las herramientas se construyeron en forma de plug-ins o módulos de Eclipse. Este entorno de creación de aplicaciones se puede descargar de:

http://www.micromethod.com/products/download.htm

Permite a los desarrolladores gestionar completamente el ciclo de desarrollo de aplicaciones, desde el comienzo del proyecto hasta el despliegue final, a través de asistentes, editores y configuraciones.

El SIPMethod ACE incluye el servidor de aplicaciones SIPMethod.

Requerimientos

Plataforma : Windows, Mac o Linux
Java : JRE 1.4.2 o posterior
Eclipse : Eclipse 3.1 o posterior

Elementos esenciales

- Asistente de Proyectos SIP
- Asistente de Servlets SIP
- Asistente de Clases de escucha SIP
- Editor para el Descriptor de Despliegue
- Sincronización de Contenido: sincroniza automáticamente los cambios del proyecto con el sip.xml.
- Despliegue de Aplicaciones: despliega las aplicaciones directamente en el servidor SIPMethod o en otros servidores.
- Ejecución y depuración de aplicaciones

C3. APIs PARA LA CREACIÓN DE LA LÓGICA DEL SERVICIO

C3.1. SERVLET SIP

Un Servlet SIP es un componente de aplicación basado en Java, el cual realiza el proceso de señalización SIP, y es dirigido por un contenedor de Servlets SIP, quien ejecuta la lógica del servicio y soporta opcionalmente HTTP y la plataforma J2EE. Como otros componentes de Java, los servlets son independientes de la plataforma. Los Servlets interactúan con clientes SIP mediante el intercambio de mensajes de peticiones y respuestas, a través de un contenedor de servlets [2].

Esta API soporta la especificación de SIP, definida en el RFC 3261, y también soporta las siguientes extensiones: el entorno de notificación de eventos (RFC 3265) y el método de mensajes para mensajería instantánea (RFC 3428).

La especificación del API pretende permitir a las aplicaciones el control completo sobre la señalización SIP, mientras al mismo tiempo esconde mucho de la complejidad no esencial de SIP, la cual no es relevante para los desarrolladores de aplicaciones.

C3.1.1. Arquitectura [3]

- **Contenedor de Servlets**
    - Este es un entorno en el cual un servlet puede existir.
    - Aquí se carga y se inicializa un servlet.
    - Cuando un mensaje SIP llega, se encarga de invocar los métodos apropiados.
    - El contenedor de Servlets SIP administra recursos tales como puertos de escucha, los hilos, transacciones y diálogos, el estado de la sesión y los componentes de la aplicación.

- **Servlets**

Son clases con métodos de servicio que se compilan en un archivo SAR (Servlet Archive File).

- **Descriptores de despliegue**

Son archivos basados en XML y que contienen la información de configuración y las reglas para orientar el procesamiento de los mensajes SIP.

C3.1.2. Características de los Servlets SIP [4]

- Son una aplicación directa de SIP al modelo de servlets HTTP.
- Los Servlets SIP procesan peticiones y respuestas SIP particulares. El ciclo de vida lo maneja el contenedor.

- Los Servlets SIP pueden crear y acceder a los datos de sesión, los datos de llamada y los datos de transacción.
- Los Servlets SIP pueden ordenarle al contenedor que: redireccione una petición, inicie una nueva petición, reenvíe una respuesta, genere una respuesta.
- Los Servlets SIP le permiten a las aplicaciones actuar como Agente de Usuario Cliente (UAC), Agente de Usuario Servidor (UAS), o servidor Proxy.

C3.1.3. Mapeo de Servlets SIP [2][4]

Un servidor único de aplicaciones puede soportar muchas de ellas. Es por ello que se necesita el mapeo de servlets, lo cual se define como reglas que crean enlaces entre los mensajes SIP y los servlets. Este mapeo se basa en expresiones que concuerdan con algún campo(s) de un mensaje.

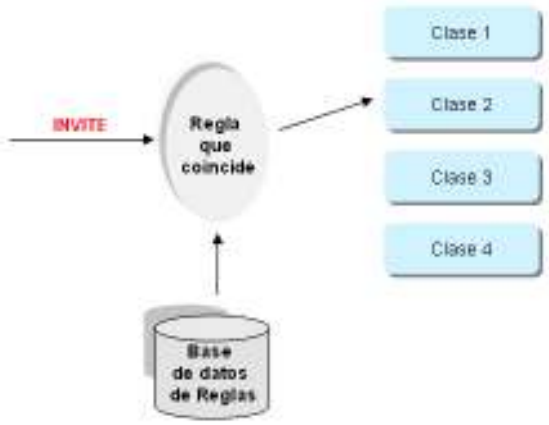

Figura C1: Mapeo de Servlets SIP

Descriptores de despliegue

Los descriptores de despliegue permiten:

- Nombres descriptivos y utilización de clases
- Mapeo de servlets
- Manejo de los parámetros de contexto, inicialización
- Hacer referencia a recursos que necesiten las aplicaciones
- Hospedaje para EJBs
- Contextos JNDIs
- Tiempo de expiración de sesión

En el mundo de los Servlets SIP, el descriptor de despliegue tiene el nombre de `sip.xml`, y su sintaxis es similar al `web.xml` de los servlets HTTP, excepto por la etiqueta `<servlet-mapping>`. Esta etiqueta no dirige una URL a un servlet sino que describe una condición (que se basa en el contenido de los campos o subacampos de un mensaje SIP entrante) que una petición

SIP debe cumplir, para que pueda dirigirse aun servlet. En otras palabras, si una regla coincide con una petición, el contenedor puede invocar el servlet correspondiente. Se debe tener en cuenta que el mapeo se utiliza para las peticiones iniciales, ya que las siguientes peticiones que se encuentren dentro del mismo diálogo o sesión se van a manejar por el mismo servlet inicial.

C3.1.4. Construcción, empaquetamiento y despliegue [2]

Los Servlets SIP se tienen que compilar y luego empaquetar en archivos SAR (Archivos Servlet). Esta es una funcionalidad equivalente a los archivos WAR, para el dominio de los servlets HTTP, y utiliza la misma estructura. Ver Figura C2.

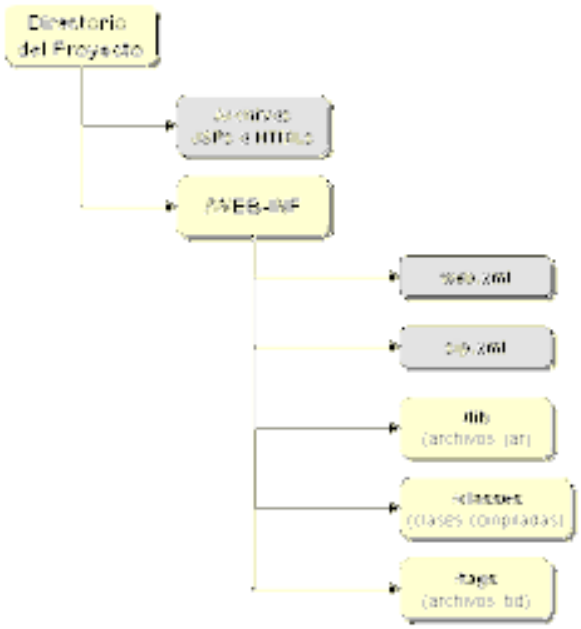

Figura C2: Estructura del archivo SAR

El último paso es el despliegue, el cual depende del servidor de aplicaciones SIP. Por lo general consiste en copiar el archivo SAR dentro de la carpeta de despliegue del servidor de aplicaciones.

C3.1.5. Métodos para el procesamiento de peticiones y respuestas [2]

Con los servlets HTTP se procesa una petición de entrada y se envían mensajes de respuesta, mientras que con los Servlets SIP se puede tanto enviar como recibir peticiones y respuestas.

Los siguientes métodos se llaman por el contenedor cuando se reciben mensajes, tanto de peticiones como de respuestas. Estos se llaman por el contenedor como se muestra a continuación, además se pueden sobrescribir.

<table>
<tr><td colspan="2"><code>void service(ServletRequest, ServletResponse)</code>

Si se sobrescribe no se debe olvidar llamar a <code>super.service()</code>.

La implementación por defecto llama los siguientes métodos:</td></tr>
<tr><td><code>void
doRequest(SipServletRequest)</code>

Si se sobrescribe no se debe olvidar llamar a <code>super.doRequest()</code>.

La implementación por defecto llama uno de los siguientes métodos de petición:</td><td><code>void doResponse(SipServletResponse)</code>

Si se sobrescribe no se debe olvidar llamar a <code>super.doResponse()</code>.

La implementación por defecto llama uno de los siguientes métodos de respuesta:</td></tr>
<tr><td><code>doAck(SipServletRequest)
doBye(SipServletRequest)
doCancel(SipServletRequest)
doInfo(SipServletRequest)
doInvite(SipServletRequest)
doMessage(SipServletRequest)
doNotify(SipServletRequest)
doOptions(SipServletRequest)
doPrack(SipServletRequest)
doRegister(SipServletRequest)
doRequest(SipServletRequest)
doResponse(SipServletResponse)
doSubscribe(SipServletRequest)</code></td><td><code>doProvisionalResponse(SipServletResponse)</code>
Corresponde a los mensajes de respuesta del tipo 1xx.

<code>doSuccessResponse(SipServletResponse)</code>
Corresponde a los mensajes de respuesta del tipo 2xx.

<code>doRedirectResponse(SipServletResponse)</code>
Corresponde a los mensajes de respuesta del tipo 3xx.

<code>doErrorResponse(SipServletResponse)</code>
Corresponde a los mensajes de respuesta del tipo 4xx, 5xx y 6xx.</td></tr>
</table>

Tabla C2: Métodos para el procesamiento de peticiones y respuestas

C3.2. JavaMail [5][6]

El API JavaMail es un paquete opcional de la plataforma Java que sirve para leer, componer, y enviar mensajes electrónicos. Esta API proporciona una estructura independiente de la plataforma e independiente de los protocolos para construir aplicaciones de correo y mensajería.

Esta API se utiliza para crear programas del tipo Agente de Usuario de Correo (Mail User Agent, MUA), similares a Pine y Microsoft Outlook. Su propósito principal no es transportar, enviar, o re-enviar mensajes como *sendmail* u otros programas del tipo Agentes de transferencia de Correo (Mail Transfer Agent, MTA), sin embargo, los programas MUA tratan con los programas MTA para el envío real.

Los protocolos que se utilizan con el API son básicamente cuatro:

- SMTP
- POP
- IMAP
- MIME

Entender lo básico de cada protocolo ayuda a entender mejor cómo usar el API JavaMail. Aunque el API sea independiente del protocolo, no se puede evitar las limitaciones de los protocolos subyacentes. Si una capacidad no está soportada por el protocolo elegido, el API JavaMail no hará que esa capacidad se proporcione.

Para el caso del acceso al correo electrónico del usuario de la Universidad del Cauca se utilizó el protocolo IMAP (Internet Message Access Protocol), el cual se define en la RFC 2060.

Las clases más importantes que componen el API son: `Session`, `Message`, `Address`, `Authenticator`, `Transport`, `Store`, y `Folder`. Todas estas clases se encuentran en el paquete del nivel superior del API JavaMail: `javax.mail`.

- **Clase `Session`**

La clase `Session` define una sesión de correo básica. Es a través de ésta que todas las demás funcionan. El objeto `Session` utiliza un objeto `java.util.Properties` para obtener información como el servidor de correo, el nombre de usuario, la contraseña, y otra información que puede compartirse a lo largo de toda la aplicación.

- **Clase Message**

Una vez que se tiene el objeto `Session`, es hora de empezar a crear un mensaje para enviar. Esto se hace con un objeto `Message`.

- **Clases Store y Folder**

Para obtener los mensajes se empieza con la creación de una `Session`. Sin embargo, después de obtener la sesión, se debe proceder a conectarse a un `Store`, con un nombre de usuario y una contraseña. A `Store` se le deberá decir qué protocolo utilizar, a través del objeto Session.

Después de conectarse al `Store`, se puede obtener un `Folder`, que debe estar abierto antes de poder leer los mensajes que hay en su interior: Para POP3, la única carpeta disponible es INBOX, mientras que con IMAP se puede disponer de otras carpetas.

Una vez que se tiene un `Message` para leer, se puede obtener su contenido con `getContent()` o escribir contenido en un `Stream` con `writeTo()`.

C4. HERRAMIENTA PARA LA CREACIÓN DEL MÓDULO MRF [7][8]

Los productos *Verbio* actualmente están disponibles únicamente para sistemas operativos Windows en sus variantes 2000/XP, aunque actualmente se está en fase de pruebas internas de una versión para Linux.

Antes de proceder a la instalación del software es conveniente verificar que el hardware utilizado reúne los requisitos mínimos indispensables para soportar la carga computacional y de memoria consumidos por los sistemas de reconocimiento y síntesis del habla *Verbio*.

C4.1. Library SDK - PLATAFORMA GENÉRICA

Library SDK proporciona un conjunto de funciones a bajo nivel que permiten utilizar cualquier dispositivo de audio (tarjetas CTI o tarjetas de sonido) siempre y cuando el desarrollador conozca cómo intercambiar muestras de audio con dicho dispositivo. Es decir, el desarrollador deberá encargarse de obtener las muestras de audio y pasarlas al sistema de reconocimiento (en procesos de reconocimiento del habla) o bien reproducir las muestras de audio que le entregue el sistema de síntesis (en procesos de síntesis del habla).

Las funciones de este SDK empiezan con el prefijo `vox_` y están disponibles desde entornos de programación en C/C++, aunque es posible su invocación directa a la DLL, desde entornos distintos, como por ejemplo Visual Basic.

Library SDK se mantiene por compatibilidad con las antiguas versiones, a pesar de que se le han incorporado nuevas funcionalidades.

C4.2. Advanced SDK - PLATAFORMA GENÉRICA C++

Advanced SDK proporciona un conjunto de funciones a más alto nivel que las funciones contenidas en *Library SDK* sin que esto suponga perder el nivel de control en las funcionalidades en que se interese.

De este modo, se pretende facilitar el uso de los sistemas de reconocimiento y síntesis del habla, especialmente en aquellos aspectos que suelen conllevar una mayor complejidad, tales como la gestión de los resultados devueltos por el reconocedor o la gestión de las desconexiones entre cliente y servidor.

Existe una gran dualidad entre las funciones de ambos SDK, por lo que sería recomendable, siempre que sea posible, migrar de *Library SDK* a *Advanced SDK* para aprovechar las nuevas (y futuras) funcionalidades.

Advanced SDK se desarrolló a partir de *Library SDK* y está formado por un conjunto de objetos definidos mediante clases C++, desarrollados inicialmente para ser utilizados desde Microsoft Visual Studio.

C4.3. VERBIO GRAMMAR MANAGER

El Gestor de la Gramática de Verbio es una herramienta destinada a diseñar y probar las gramáticas de reconocimiento utilizadas en la aplicación.

C4.4. GRAMÁTICAS DE RECONOCIMIENTO

Una gramática de reconocimiento define el universo de posibles respuestas que el sistema puede reconocer en un determinado momento. Es el subconjunto del "lenguaje" aceptado por el reconocedor. Por lo tanto debe haber una gramática adecuada para cada instante del diálogo entre la persona y la máquina. Una gramática especifica el vocabulario de palabras que el sistema puede reconocer así como las posibles secuencias en que éstas se pueden combinar.

Por ejemplo, dado un vocabulario de palabras "cerrar, abrir, la, ventana, puerta", una gramática podría ser "una de las palabras del vocabulario" (es decir, reconocimiento de palabras aisladas permitiendo que el usuario diga locuciones como "abrir" o "ventana" por ejemplo); otra gramática podría ser "una o varias de las palabras del vocabulario" (es decir, reconocimiento de palabras conectadas permitiendo al usuario decir locuciones como "puerta" o "abre la puerta", pero también "la cerrar abrir"); finalmente, otra gramática "basada en patrones" como "[abrir | cerrar] [la] [puerta | ventana]" imponiendo el lenguaje natural en una sintaxis de gramática de contexto libre.

Los diferentes tipos de gramáticas definen el lenguaje y la terminología para indicar cómo combinar las palabras.

- **Listas de palabras**

La gramática más sencilla para el reconocimiento del habla es la anteriormente mencionada como: "una de las palabras del vocabulario". *Verbio* permite usar archivos de texto consistentes en listas de entradas (cada una con una o más palabras), una por línea. Cada una de estas entradas se denotará como *palabra del vocabulario*. Veáse en el siguiente ejemplo:

```
Azul Claro
Azul Oscuro
Azul Celeste
```

Opcionalmente, las listas de palabras se pueden formatear en columnas separadas por un carácter tabular ("\t").

Las *palabras de relleno* (entradas que empiezan con el símbolo #) son útiles para indicar qué palabras acompañantes pueden usarse en el contexto de la gramática pero no son propiamente palabras con contenido (significantes) (una palabra de relleno reconocida no cuenta como palabra devuelta por las funciones de procesado del resultado).

1023 -> Luís Fernández
1024 -> José Pérez
1024 -> Pepe Pérez
1032 -> María Sancho
#ponme con
#por favor

Estas gramáticas tipo "una de las palabras del vocabulario" se denotan en *Verbio* como *palabras aisladas*.
Similarmente, las gramáticas tipo "una o varias de las palabras del vocabulario", denotadas en *Verbio* como *palabras*) usan los mismos archivos pero se permiten múltiples entradas por elocución del usuario.

C4.5. CLASES MÁS IMPORTANTES PARA EL MANEJO DE VERBIO

- **`VerbioMRF`**: Contiene toda la funcionalidad, tanto de síntesis como de reconocimiento de voz.
- **`EngineASR`**: Se encarga de la conexión con el servidor Verbio para ASR.
- **`EngineASRResource`**: Se encarga de enviar las muestras de audio en formato PCM Ley A, al motor de reconocimiento.
- **`VerbioResult`**: Entrega los resultados del ASR con su respectivo nivel de certeza.
- **`EngineTTS`**: Esta clase se encarga de configurar los parámetros relacionados con el servidor TTS.
- **`EngineTTSResource`**: Entrega el audio (PCM Ley A o μ) correspondiente al texto suministrado previamente.

C4.6. CLASES MÁS IMPORTANTES PARA EL MANEJO DE SIP

OSIP

Es un API C/C++ para SIP que opera en el nivel de transacción. Las clases principales son:

- **`osip`**: Representa el stack del API, es decir, habilita toda la funcionalidad de SIP.
- **`osip_message`**: Encapsula el formato del mensaje SIP, lo que facilita su manipulación.
- **`osip_parser`**: Se encarga de convertir los mensajes de texto SIP en estructuras.
- **`osip_event`**: Representa un evento SIP, que puede ser un mensaje o una respuesta dentro de una transacción.

- **`Sock_udp`**: Se encarga del nivel de transporte, es decir, de los sockets de envío y recepción de mensajes SIP.

C4.7. CLASES MÁS IMPORTANTES PARA EL MANEJO DE RTP

JRTPLIB

Es un API C/C++ para RTP. Las clases principales son:

- **`Rtpconnection`**: Es la clase encargada de manipular los sockets de entrada y salida para el envío y recepción de paquetes RTP.
- **`Rtpsesion`**: Encapsula toda la funcionalidad del protocolo RTP.
- **`RTPPacket`**: Es una abstracción de un paquete RTP.
- **`Rtppacketprocessor`**: Se encarga de validar los paquetes RTP de llegada, y de salida.

C5. API PARA LA REALIZACIÓN DE PRUEBAS

JAIN SIP

JAIN SIP es una especificación para transacciones de propósito general para el protocolo SIP, la cual se basa en interfaces Java. Es rico tanto en semántica como en la definición del protocolo SIP, y se desarrolló como una interfaz estándar para el protocolo que puede ser usado independientemente del nivel de programación y el entorno. JAIN SIP puede ser usado de multiples formas:

- Como una especificación para plataformas J2SE que permiten el desarrollo de aplicaciones independientes de agentes de usuario, aplicaciones de registro y proxy.
- Como base para la implementación SIP para contenedores de SIP Servlets, que permite el desarrollo de agentes de usuario, aplicaciones de registro y proxy dentro de un entorno basado en Servlets.
- Como base para la implementación SIP para contenedores de EJBs (Enterprise JavaBeans), que permiten el desarrollo de agentes de usuario, aplicaciones de registro y proxy en un entorno EJB. [9]

C5.1. CARACTERÍSTICAS DE JAIN SIP [9]

- Proporciona una implementación de referencia con una funcionalidad completa de la implementación SIP.
- Estandariza la interfaz para el modelo de transacciones genéricas definidos por el protocolo SIP y proporciona acceso para la funcionalidad de diálogo desde la interfaz de transacciones.
- El diseño es extensible.

C5.2. INFORMACIÓN GENERAL DE LA ARQUITECTURA [9]

JAIN SIP soporta la funcionalidad del RFC 3261 y de las siguientes extensiones de SIP: el método INFO (RFC 2979), la fiabilidad para proporcionar respuestas (RFC 3262), el marco para la notificación de eventos (RFC 3265), el método UPDATE (RFC 3311), el encabezado Reason (RFC 3326) y el método MESSAGE (RFC 3428) definido para la mensajería instantánea.

La arquitectura se desarrolla para el entorno J2SE por tanto se basa en eventos utilizando el modelo de eventos `Listener` / `Provider`. La especificación es asíncrona por lo cual identificadores de transacciones para los mensajes correlacionados, también define varias clases "`factory`" para la creación de mensajes `Request`, `Response` y encabezados SIP.

JAIN SIP define una interfaz para cada encabezado que soporta, los cuáles se pueden adicionar a los mensajes `Request` y `Response`. Estos mensajes se pasan al `SipProvider` con una

transacción para ser enviados a través de la red, mientras el `SipListener` escucha los eventos que llegan y encapsula los mensajes, los cuáles pueden ser respuesta a los diálogos iniciados o nuevos diálogos que comienzan.

El diseño JAIN SIP es extensible, definiendo una interfaz genérica de encabezados que se pueden usar para aplicaciones que utilicen encabezados que no son soportados directamente por JAIN SIP. El diseño también define mecanismos que soportan métodos de creación de diálogos futuros en un entorno JAIN SIP, usando las propiedades Java. JAIN SIP puede ser manejado estáticamente considerando las direcciones IP, la función de encaminamiento, y la especificación dinámica para puertos y método de transporte (UDP, TCP).

En la Figura C3 se puede apreciar la Arquitectura de Mensajes SIP [10]:

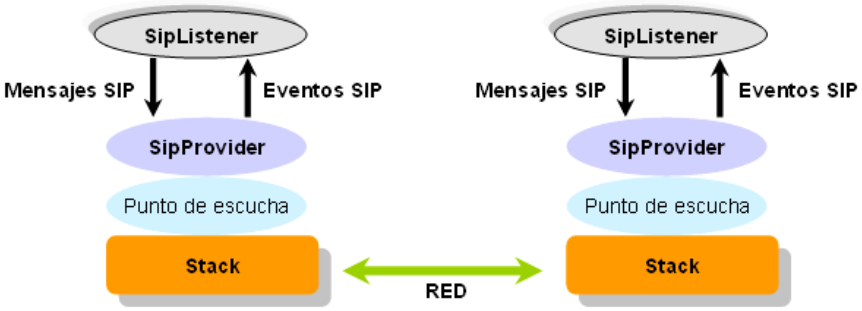

Figura C3: Arquitectura de mensajes SIP

C5.3. IMPLEMENTACIÓN DE REFERENCIA: NIST-SIP [11][12]

Una implementación de la plataforma JAIN SIP de dominio público es NIST-SIP. La implementación NIST-SIP, distribución 1.2, contiene las siguientes librerías:

- La Implementación de Referencia oficial del API JAIN SIP (JSR 32)
- El Equipo de Compatibilidad Tecnológica (TCK "Technology Compatibility Kit") para el JSR 32.
- Una implementación del API JAIN SDP (JSR 141)

Plataformas Soportadas
J2SE 1.4.x, se recomienda utilizar J2SE 1.4.2 o superior.

Librerías y Grupo de Paquetes

En la Tabla B3 se listan los paquetes incluidos en la jerarquía `gov.nist.` y `javax.sip` que constituyen la implementación de referencia para JAIN-SIP 1.2

Paquete	Descripción
`gov.nist.core`	Contiene las clases del núcleo de las cuales dependen el resto de implementaciones.
`gov.nist.core.net`	Contiene las clases e interfaces de red.
`gov.nist.javax.sdp`	Implementación anterior del paquete `javax.sdp`
`gov.nist.javax.sip`	Raíz de la implementación JAIN de SIP.
`gov.nist.javax.sip.address`	Implementación del paquete `address` del API JAIN SIP.
`gov.nist.javax.sip.header`	Contiene la implementación de los encabezados SIP como se definen en JAIN-SIP-1.1, y una implementación de la `Factory` de encabezados JAIN-SIP.
`gov.nist.javax.sip.parser`	Traductores para los encabezados SIP, URL's y direcciones.
`gov.nist.javax.sip.stack`	Este paquete contiene las clases para la construcción de la pila del protocolo SIP.
`javax.sdp`	Esta es una versión preliminar del paquete `javax.sdp` basado en la versión pública JSR141.
`javax.sip`	Este paquete contiene las interfaces principales del modelo de la arquitectura JAIN SIP.
`javax.sip.address`	Este paquete contiene interfaces que representan los componentes de direccionamiento del protocolo SIP.
`javax.sip.header`	Este paquete contiene todas las interfaces de encabezados y extensiones soportados por esta especificación.
`javax.sip.message`	Este paquete contiene las interfaces que representan los mensajes SIP de `Request` y `Response`.

Tabla C3: Paquetes JAIN-SIP 1.2

Factories

Las “factories” que define JAIN SIP son cuatro, cada una con una responsabilidad respectiva, como se muestra en la Tabla C4:

Factory	Descripción
`SipFactory`	Esta interfaz define métodos para crear los nuevos objetos `Stack` y otros objetos `Factory`.
`AddressFactory`	Esta interfaz define métodos para crear direcciones SIP y telefónicas.
`HeaderFactory`	Esta Interfaz define métodos para crear nuevos objetos del tipo `Headers`.
`MessageFactory`	Esta interfaz define métodos para crear nuevos objetos de mensajes SIP de `Request` y `Response`.

Tabla C4: Factories de JAIN-SIP

C5.4. RELACIÓN DE LOS SERVLETS SIP CON JAIN SIP [4]

JAIN SIP es una interfaz de bajo nivel para el acceso a servicios SIP, y puede utilizarse en: clientes, servidores, pasarelas, se concentra completamente en el protocolo SIP, solamente soporta transacciones. Un contenedor de Servlets SIP es una aplicación particular de JAIN SIP.

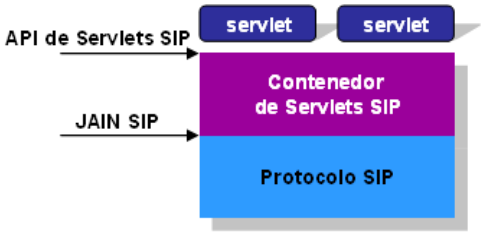

Figura C4: Relación de los Servlets SIP con JAIN SIP

Los servlets se enfocan en servidores que manejan altos grados de volumen. Además adicionan funciones que no hacen parte del protocolo SIP.

C6. BIBLIOGRAFÍA

[1] MICROMETHOD. SIPMethod Platform. Disponible en web: <http://www.micromethod.com/products/sipmethod.htm>

[2] The Java Community Process. *"SIP Servlet API Version 1.0"*. [En línea]. Febrero 2003. Disponible en web: <https://sdlc1b.sun.com/ECom/EComActionServlet;jsessionid=44E0C3137185115CC66BC66F8D0C7874/> [Consulta: Agosto de 2006]

[3] Dinamicsoft. *"SIP Servlets"*. [En línea]. Disponible en web: <http://phoenix.labri.fr/documentation/sip/Documentation/Papers/Programming_SIP/Presentation/SIP_Servlet/Presentation_SIP_Servlets.ppt> [Consulta: Agosto de 2006]

[4] Dinamicsoft. *"SIP Servlets"*. [En línea]. Enero 2001. Disponible en web: <http://www.jdrosen.net/papers/servlet_summit2001.ppt> [Consulta: Agosto de 2006]

[5] Sun Microsystems, Inc. *"JavaMail API documentation"*. [En línea]. Disponible en web: <http://java.sun.com/products/javamail/reference/api/index.html> [Consulta: Agosto de 2006]

[6] Programación en castellano, Juan Antonio Palos. *"API JavaMail"*. [En línea]. Disponible en web: <http://www.programacion.net/java/tutorial/javamail/> [Consulta: Agosto de 2006]

[7] ATLAS, Verbio ASR. *Verbio ASR (Automatic Speech Recognition - Reconocimiento del habla)*. [En línea]. Disponible en web: <http://www.verbio.com/productes.php?id=2> [Consulta: Enero de 2006]

[8] ATLAS, Verbio TTS. *Verbio TTS*. [En línea]. Disponible en web: <http://www.verbio.com/productes.php?id=1> [Consulta: Agosto de 2006]

[9] Phelim O'Doherty – Sun Microsystems, Inc. *"SIP Specifications and the Java Platforms, the look and feel of SIP."*. [En línea]. Enero 2003. Disponible en web: < http://www.cs.columbia.edu/sip/Java-SIP-Specifications.pdf> [Consulta: Septiembre de 2005]

[10] Phelim O'Doherty, Sun Microsystems, Inc., RANGANATHAN, Mudumbi. NIST. *"JAIN SIP Tutorial."*. [En línea]. 2003. Disponible en web: <http://java.sun.com/products/jain/JAIN-SIP-Tutorial.pdf> [Consulta: Agosto de 2006]

[11] NIST/ITL Advanced Networking Technologies Division.. *"NIST-SIP: The Reference Implementation for JAIN-SIP 1.2"*. [En línea]. Disponible en web: <http://snad.ncsl.nist.gov/proj/iptel/jain-sip-1.2/javadoc/index.html> [Consulta: Agosto de 2006]

[12] NIST, National Institute of Standars and Technology. Página principal del Proyecto NIST-SIP. [En línea]. Disponible en web: <http://snad.ncsl.nist.gov/proj/iptel/> [Consulta: Agosto de 2006]