Plataforma de acceso universal a mensajería instantánea móvil

O. M. Caicedo*
Miembro del Grupo de Ingeniería Telemática (GIT), Universidad del Cauca

D. A. Caicedo*
Miembro del Grupo de Ingeniería Telemática (GIT), Universidad del Cauca

E. Figueroa*
Miembro del Grupo de Ingeniería Telemática (GIT), Universidad del Cauca

F. O. Martínez,*
Miembro del Grupo de Ingeniería Telemática (GIT), Universidad del Cauca

J. A. Hurtado*
Miembro del Grupo de Ingeniería Telemática (GIT), Universidad del Cauca

Fecha de recepción: 30-05-06 Fecha de selección: 30-10-06 Fecha de aceptación: 30-08-06

ABSTRACT

One of the most succesful services is Instant Messaging (IM), due to the versatility and simplicity offered to users. Because of the success of mobile telephony, new horizons have been developed for IM. However, the diversity existing in protocols and transport technologies have lead the technology to a low interoperability environment.

This papers describes a platform called PAUMIM , because its spelling in Spanish, proposed by our research group, that allows the communication between different providers of IM services using mobile devices.

KEYWORDS

Interoperability, Instant Messaging, Mobility, Jabber, Transport

RESUMEN

En la actualidad uno de los servicios más exitosos es la mensajería instantánea, gracias a la versatilidad y simplicidad de comunicación que ofrece a los usuarios, una característica clave en el exigente entorno de conectividad y movilidad que demanda la sociedad. Con el éxito de la telefonía móvil, nuevos horizontes hicieron su aparición para este tipo de servicios; sin embargo, la diversidad de protocolos y tecnologías de transporte también se hizo más diversa y

* O. M. Caicedo, E. Figueroa, D. A. Caicedo, F. O. Martínez y J. A. Hurtado son miembros del grupo de interés en Desarrollo de Aplicaciones Móviles e Inalámbricas W@PColombia perteneciente al Grupo de Ingeniería Telemática (GIT), Universidad del Cauca, Calle 5 No. 4-70 Popayán, Colombia. (e-mail: {omcaicedo, dacaicedo, efigueroa, fomarti, javhur}@unicauca.edu.co).

dio origen a un ambiente de baja y compleja interoperabilidad.

El presente artículo describe la plataforma PAUMIM (Plataforma de Acceso Universal a Mensajería Instantánea Móvil), propuesta por el grupo de interés en Desarrollo de Aplicaciones Inalámbricas Móviles e Inalámbricas (W@PColombia) del Grupo de Ingeniería Telemática (GIT) de la Universidad del Cauca, para la comunicación entre distintos proveedores de este servicio, a través del uso de dispositivos móviles.

PALABRAS CLAVE

Interoperabilidad, Mensajería Instantánea, Movilidad, Transportes, Jabber.

Clasificación Colciencias: A

I. INTRODUCCIÓN

En la actualidad el continuo crecimiento de internet brinda a los usuarios nuevos servicios soportados sobre tecnologías de banda ancha, los cuales les ofrecen alternativas para agilizar su trabajo, incursionar en el mundo de la información y el entretenimiento. Para llevar a cabo comunicaciones simples y de forma inmediata, uno de los servicios más utilizados es el de Mensajería Instantánea caracterizado por su amplia acogida y rápida expansión alrededor del mundo.

Las ventajas del servicio de mensajería instantánea se pueden evidenciar en el entorno empresarial y en el sector del entretenimiento. Dentro de una empresa es vital contar con herramientas que permitan una comunicación interactiva entre los trabajadores, y compartir recursos y conocimientos, como una alternativa a los servicios de comunicación actuales como la telefonía fija o móvil. En el campo del entretenimiento, en los últimos años el uso de este servicio ha cobrado mucha importancia y ofrece a los usuarios un medio inmediato para comunicarse con la familia, compañeros y amigos [1].

El principal problema que presenta la mensajería instantánea móvil y fija es la coexistencia de varias redes [2], cada una de ellas con aplicaciones y protocolos distintos que dificultan la interoperabilidad entre los diferentes proveedores, imposibilitando una verdadera comunicación universal entre los usuarios a cualquier hora y lugar. Surge entonces la necesidad de garantizar la interoperabilidad entre los diferentes proveedores de mensajería instantánea y proveer movilidad a los usuarios del servicio.

El grupo de interés en el desarrollo de aplicaciones móviles e inalámbricas w@pcolombia del Grupo de Ingeniería Telemática (GIT) plantea la creación de una plataforma de acceso universal a mensajería instantánea móvil (PAUMIM), que será descrita de la siguiente forma: en la sección II se presentan los conceptos asociados a la mensajería instantánea tanto fija como móvil; en la sección III se describe con más detalle el protocolo de mensajería instantánea Jabber, como núcleo de la plataforma construida; en la sección IV se describe la plataforma en sí; en la sección V, el entorno de experimentación y pruebas; y finalmente en la sección VI se presentan las conclusiones del trabajo realizado.

II. MENSAJERÍA INSTANTÁNEA

A. *Definición*

El término mensajería instantánea hace referencia en internet a la posibilidad de establecer conversaciones de texto en directo entre individuos [3]. Pero a diferencia de los chat, esta comunicación no se establece en una sala en la que hay más personas, sino de forma exclusiva entre los dos individuos involucrados, por lo tanto se considera como un punto intermedio entre los sistemas de chat y los mensajes de correo electrónico.

Las herramientas de mensajería instantánea son programas regularmente gratuitos y versátiles, de fácil instalación y uso, que requieren una conexión a Internet para su activación. Dichos programas permiten realizar conversaciones de texto,

envío de archivos y videoconferencia entre otros servicios [3].

B. *Proveedores de mensajería instantánea*

La mensajería instantánea ha ganado popularidad en forma abrumadora. En los últimos años el número de usuarios de los distintos proveedores como AOL (American OnLine), Microsoft Messenger, Yahoo e ICQ (I Seek You), se ha incrementado en forma sustancial y ha alcanzado casi el número total de usuarios de Internet [4]. A continuación se identifican las herramientas de mensajería instantánea más importantes.

ICQ: Fue el primer sistema de mensajería instantánea que salió al mercado y proporcionó una nueva alternativa a los sistemas de chat convencionales, lográndolo con éxito [3]. La última versión disponible, además de funciones como ICQPhone que incorpora telefonía IP para hacer llamadas entre computadores o de estos a teléfonos convencionales, tecnología SMS (Short Message Service) [5], integración con Outlook y más, ofrece una nueva colección de iconos (emoticons), que facilitan el envío de mensajes y las intenciones de comunicación.

MSN Messenger: Pertenece a Microsoft Networks, es un sistema eficiente cuya principal ventaja, además de su sencillez de uso, es su integración con Hotmail y MSN, la red de contenido de Microsoft, con la estrategia. Net. [3]. El servicio de mensajería de Microsoft ofrece chat, video llamadas, conferencia, transferencia de archivos, iconos gestuales, pizarra, envío de mensajes SMS, entre otras funcionalidades.

Yahoo Messenger: Es una de las alternativas de mensajería instantánea más fresca, y mejor integrada con la plataforma de servicios de la marca (Yahoo!) como el correo electrónico y geocities, de manera que su uso resulta transparente [3].

AIM (American-On-Line Instant Messenger): Sus prestaciones avanzadas incluyen la comunicación entre computadores o de computadores a teléfonos convencionales; para hacerlo requiere un proveedor que soporte el servicio. También permite configurar alertas para correo electrónico y leer los mensajes de cualquiera de sus cuentas [3].

C. *Protocolos abiertos de mensajería instantánea*

Dentro de las iniciativas más importantes que se han propuesto para la normalización y estandarización del servicio de mensajería instantánea figuran los protocolos de la IETF SIMPLE (SIP for Instant Messaging and Presence Leverage) [6]-[7] y Jabber/XMPP (Extensible Messaging and Presence Protocol) [8]-[9] basado en XML.

SIMPLE: Es un protocolo que permite el intercambio de mensajes y manejo de presencia [6]. Está basado en el protocolo SIP [10], es utilizado para establecer, administrar y finalizar sesiones con el objetivo principal de ayudar a los usuarios a enviar invitaciones hacia los posibles participantes de una sesión, dondequiera que se encuentren. Estas sesiones permiten establecer comunicación entre los usuarios para poder intercambiar distintas clases de información como voz, video, mensajes y datos.

Jabber: Es un conjunto de protocolos XML de flujos de descarga (Streaming) y tecnologías que permiten que dos entidades en internet intercambien mensajes, información de presencia, y otra información estructurada en tiempos cercanos al real [11]. Jabber se encuentra soportado en miles de servidores de internet y es usado por más de seis millones de personas en todo el mundo; a pesar de esto, se encuentra menos difundido que muchos sistemas propietarios.

D. *Mensajería móvil*

JIMM: Jimm es un clon ICQ para dispositivos móviles, específicamente para teléfonos celulares, basado en Java 2 Micro Edition [12]. Jimm no representa ningún producto de ICQ, inc. La distribución de Jimm es pública con la licencia GNU General Public License (GPL) [15].

JXME – JXTA for J2ME: El proyecto JXTA es una plataforma abierta de programación para servicios y aplicaciones P2P (Peer to peer). El propósito del proyecto JXTA para J2ME es proveer funcionalidades compatibles JXTA usando dispositivos que soporten Java [16]-[17]-[18].

Mobber: Es un sistema de mensajería instantánea propietario basado en el protocolo Jabber, orientado al acceso desde dispositivos móviles. En la actualidad se encuentra en desarrollo, no hay versiones con funcionalidad estable y la documentación es muy escasa. El sistema Mobber está compuesto por un servidor propietario y una aplicación J2ME de libre distribución para el cliente móvil, desde la cual se puede establecer comunicación con contactos de diferentes proveedores de mensajería, con la restricción de que el acceso se debe hacer desde una cuenta del servidor Mobber [19].

III. JABER

Jabber es un conjunto de protocolos de libre distribución, cuenta con una comunidad de desarrollo muy grande y dentro de sus principios fundamentales está la interoperabilidad con otros sistemas de mensajería[14], razón por la cual ha sido elegido como el stack de protocolos para la plataforma PAUMIM. Entre sus características más importantes, Jabber cuenta con un stack de protocolos basado en XML, lo cual hace posible su interpretación por diferentes sistemas operativos y plataformas, no es centralizado y es altamente extensible a través de la adición de componentes. Por otro lado, Jabber brinda soporte SSL (Secure Socket Layer) para comunicaciones cliente/servidor y para algunos clientes soporta la extensión GPG (GNU Privacy Guard) para firmar información de presencia y cifrar las comunicaciones punto a punto usando modelo asimétrico [20]. A continuación se realiza una breve descripción del stack de protocolos de Jabber.

A. *Protocolo de mensajería*

El protocolo de gestión de mensajes Jabber es simple pero poderoso. Por defecto, no se recibe confirmación sobre la llegada de un mensaje a su destino para reducir el tráfico en el servidor; por otro lado, si el receptor no se encuentra disponible el servidor guardará el mensaje hasta que éste se conecte [9].

B. *Protocolo de presencia*

Una de las grandes diferencias que existen entre la mensajería instan-

tánea y el correo electrónico es que los usuarios tienen la posibilidad de conocer el estado de otros usuarios. Jabber ofrece la posibilidad de establecer tantos estados diferentes como el usuario desee. Una de las prioridades más altas tenidas en cuenta en el protocolo Jabber 3 es la intimidad de los usuarios. Por lo tanto, si un usuario quiere agregar a otro en su lista de contactos y recibir su información de presencia, deberá hacer una petición al servidor, y si el otro usuario acepta, entonces se podrá ver su estado [21].

C. *Protocolo de grupo de chat*

Gestiona la comunicación entre usuarios de un grupo de chat [11]. Existen cuatro acciones fundamentales:

Unirse al grupo: Enviando un mensaje de presencia de tipo "disponible (available)" al grupo.

Enviar mensajes a todo el grupo: Enviando un mensaje al grupo deseado, al cual previamente el usuario ha debido unirse.

Enviar mensajes a un único miembro del grupo: Enviando un mensaje a una persona específica del grupo.

Abandonar el grupo: Enviando un mensaje "no disponible (unavailable)" de presencia al grupo.

D. *Protocolo de información-solicitud (IQ- Inf o- Query)*

Es un protocolo sencillo y extensible de petición/respuesta que permite a los usuarios hacer peticiones y almacenar o cambiar datos. IQ es simplemente el conductor de esas peticiones y respuestas, y gestiona los datos que son necesarios de acuerdo con las conveniencias particulares de cada servidor. La mayoría de las peticiones IQ son entre un cliente y un servidor. Sin embargo, hay algunos protocolos de IQ que van estrictamente de un cliente a otro, como el protocolo de petición de versión del cliente, en el cual un cliente le pide a otro su versión del programa [11].

E. *Protocolo de registro*

El primer paso para albergar usuarios en el servidor Jabber es la creación de cuentas mediante una extensión del protocolo IQ. En los servidores públicos, y en su más pura esencia, las cuentas de usuario no son más que unos repositorios de credenciales que los clientes usan para autenticarse con el servidor. Además de estos datos básicos, muchos servidores asocian otros a la cuenta de usuario.

Aunque el manejo y almacenamiento de las cuentas de usuario puede ser algo complicado, la implementación del protocolo de registro no lo es. El protocolo de registro normalmente suele ser junto con el de autenticación el único protocolo que un usuario sin autenticar puede usar de un servidor Jabber [11].

F. *Protocolo de autenticación*

El protocolo de autenticación de Jabber, al igual que el de registro, es una extensión del protocolo IQ. Es uno de los protocolos dedicados únicamente a la seguridad en Jabber basado en SASL (Simple Authentication and Security Layer). Este protocolo permite a un usuario demostrar al servidor que realmente es quien dice ser.

El sistema de autenticación y acceso a un servidor es simple: los clientes no autenticados tienen una serie de permisos restringidos, y los clientes

autenticados tienen un completo acceso al uso de todos los protocolos Jabber que estén implementados en el servidor del dominio al que pertenecen. El protocolo de autenticación ofrece cuatro niveles diferentes de autenticación:

Anonymous: Si el servidor admite usuarios anónimos basta con el envío de la petición IQ sin ningún otro tipo de dato y el usuario puede validar una sesión con el servidor.

Plain: Funciona enviando dentro del mensaje de autenticación la contraseña sin codificar en formato de texto.

Digest: Mediante este tipo de autenticación el servidor envía un identificador de sesión junto con la etiqueta de inicio de sesión. Para generar la autenticación, el cliente concatena el identificador de la sesión con la contraseña del usuario. La cadena resultante es codificada usando el algoritmo SHA-1 (Secure Hash Standard - 1).

Zero-knowledge: Es el formato más seguro y más complicado soportado por los protocolos Jabber. Este método de autenticación es complejo y su adopción tanto en servidores como en clientes ralentiza el proceso de autenticación. Este tipo de autenticación ya no guarda la contraseña del usuario en el servidor. De hecho, la información que el servidor guarda son las credenciales que sólo sirven para una única autenticación del cliente. El servidor irá creando nuevas credenciales de único uso [11].

G. *Protocolo Roster*

Para no enviar los cambios de presencia entre todos los usuarios del sistema, Jabber ha creado el concepto de suscripción de presencia. Como su nombre lo indica, la suscripción de presencia determina los suscriptores que recibirán las actualizaciones de presencia de cada usuario.

Los suscriptores piden una suscripción a un usuario, y el usuario acepta o deniega dicha suscripción. Cada usuario se suscribe a los usuarios que desea y puede aceptar que dichos usuarios u otros se suscriban a los cambios de su presencia. Para realizar esta tarea, Jabber ha definido unas estructuras de datos estándar conocidas como Jabber Roster, que no son más que una lista de otros usuarios identificados por su Jabber ID [11].

H. *Transportes entre Jabber y otros servidores de mensajería instantánea*

Debido a que Jabber es un protocolo libre basado en el intercambio de paquetes en formato XML, los sistemas Jabber están concebidos como un sistema genérico de transporte de mensajería instantánea. Su diseño simple ha sido explotado por servidores Jabber para conectarse con otros sistemas de mensajería instantánea como MSN Messenger y Yahoo! Messenger.

El servidor Jabber de referencia jabberd utiliza módulos llamados transportes (transport) que proveen un puente entre Jabber y los demás sistemas de Mensajería Instantánea. Los transportes tratan a cada sistema de mensajería instantánea propietario como un dominio Jabber con sus propios clientes identificándolos por su "Jabber ID". Al enviar un mensaje Jabber a uno de esos módulos, los Jabber IDs especiales hacen que

sean transportados por el módulo de transporte. Los módulos de transporte conectan con el sistema de mensajería 4 instantánea correspondiente y actúan como clientes de ese servidor para intercambiar mensajes y presencia entre los dos sistemas. Cada transporte debe hacer la traducción de los mensajes Jabber al formato respectivo del sistema de mensajería instantánea correspondiente [11].

IV. PLATAFORMA DE ACCESO UNIVERSAL A MENSAJERÍA INSTANTÁNEA MÓVIL - PAUMIM

A. *Arquitectura*

La Figura 1 ilustra la arquitectura de la Plataforma de Acceso Universal a Mensajería Instantánea Móvil, PAUMIM, la cual consta de los siguientes módulos:

Figura 1. Arquitectura de la plataforma PAUMIM.

Módulo administrativo: Permite realizar la administración y mantenimiento de la plataforma PAUMIM mediante acceso Web. Entre las funcionalidades que brinda se encuentra la capacidad de gestionar usuarios mediante su adición, edición y eliminación de la plataforma, y el establecimiento de comunicación directa entre el administrador de la plataforma y el usuario por medio de mensajes. El administrador tiene a su disposición información estadística con respecto al uso de la plataforma a través de gráficos que le permiten visualizar el número de usuarios conectados, tráfico cursado e información de tarificación.

Módulo de conexión móvil: Permite a los clientes móviles acceder a los servicios de mensajería instantánea que ofrece PAUMIM. Este módulo

es el encargado de gestionar las conexiones y manejar la sesión de los clientes soportados (Clientes J2ME MIDP1.0 y MIDP2.0) [12]. Para realizar la gestión de conexiones, este módulo cuenta con dos submódulos. El primero se encarga de administrar las conexiones de los clientes MIDP1.0 a través de un componente de actualización de mensajes HTTP que mantiene en cola los mensajes de intercambio entre el servidor y los clientes, los cuales se encuentran en un proceso continuo de sondeo para actualizar su estado. El segundo se encarga de administrar las conexiones de los clientes MIDP2.0, recibiendo peticiones de conexión y realizando la asignación de las mismas a través de sockets TCP.

Módulo de interoperabilidad: Es el encargado de llevar a cabo la implementación del protocolo de mensajería instantánea Jabber. Entre sus funciones más importantes se encuentran el registro de usuarios, la gestión de información de presencia, mensajería y contactos de los usuarios, y garantiza la interoperabilidad con proveedores de mensajería externos. Cuenta con dos submódulos descritos a continuación:

Servicios Jabber: Encargado de prestar los servicios definidos por el protocolo Jabber, entre ellos la conexión, envío de mensajes y manejo de presencia. Se comunica con el módulo de control central para efectos de notificación, transporte de mensajes y presencia, y con el módulo transporte para establecer comunicación con los sistemas de mensajería instantánea MSN, AOL e ICQ.

Módulo de transportes: Es una extensión del módulo de servicios Jabber, la cual permite establecer una comunicación con otras redes IM. Este módulo desempeña el papel de representación de los clientes móviles ante los servidores de mensajería instantánea propietarios, con el fin de dar a la plataforma una de sus principales características, la interoperabilidad.

Módulo de control central: Se encarga de coordinar toda la funcionalidad de la plataforma. Se compone de tres submódulos:

Control: Encargado de implementar la lógica del negocio del sistema y llevar a cabo la coordinación de mensajes entre los módulos restantes.

Tarificación: Lleva a cabo las tareas de registro de datos y generación de información de utilidad para la administración de la plataforma. La primera tarea consiste en registrar el uso de la plataforma por parte de los usuarios, en donde se tiene en cuenta el tráfico cursado y las horas pico de uso. La segunda es registrar el consumo por parte de los usuarios en donde se tiene en cuenta el número de sesiones iniciadas, mensajes enviados, mensajes recibidos y sesiones de chat establecidas.

Gestión de usuarios: Encargado de gestionar la información de los usuarios. Provee las funcionalidades de registro, eliminación, edición y búsqueda de usuarios.

Cliente móvil (CUMI): Es una aplicación J2ME que se ejecuta en el dispositivo móvil, la cual constituye la interfaz entre el usuario y el sistema. A través de un módulo de almacenamiento persistente, se guardan preferencias y datos del usuario. Para establecer la comunicación, el usuario

no necesita crear una nueva cuenta en el servidor PAUMIM, ya que puede iniciar una sesión a través de la cuenta de su proveedor de mensajería tradicional. Existen dos versiones de *CUMI*, una para dispositivos móviles de gama media (teléfonos celulares y PDAs Palm, basado en la implementación MIDP 1.0 [13]) y otra para dispositivos móviles de gama media-alta (teléfonos celulares y smartphones, basado en la implementación MIDP 2.0 [13]).

B. *Diseño*

Las Figuras 2 y 3 muestran el diseño de la arquitectura del sistema. El diagrama de paquetes del servidor muestra una arquitectura en tres capas. En la primera están los paquetes que contienen las clases que fueron implementadas por completo durante el desarrollo de este proyecto. La segunda corresponde a API (Application Program Interface) de terceros adicionadas como librerías, y en la tercera se encuentran los paquetes que corresponden a módulos funcionales de la plataforma externos al ambiente de desarrollo Java que fueron utilizados directamente o con pequeñas adaptaciones. En contravía, el diagrama de paquetes del cliente móvil muestra una arquitectura mucho más simple.

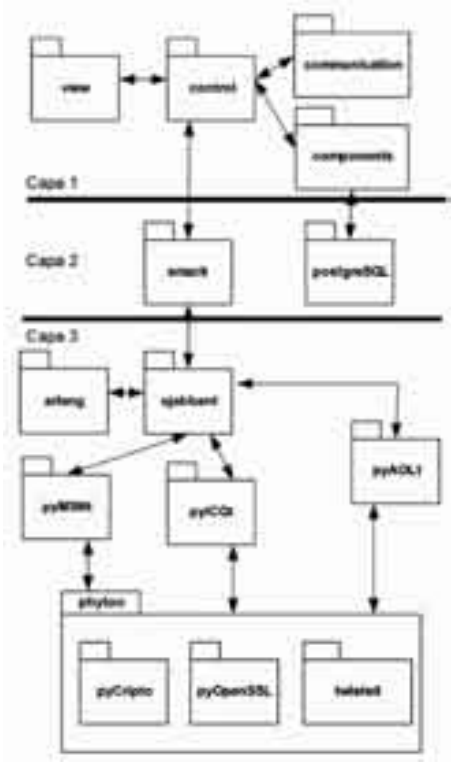

Figura 2. Diagrama de paquetes del servidor PAUMIM.

A continuación se realiza una breve descripción de cada uno de los paquetes:

Servidor PAUMIM python: Lenguaje interpretado utilizado para el desarrollo del módulo de transportes [22].

Figura 3. Diagrama de paquetes del cliente móvil PAUMIM .

pyOpenSSL: Provee extensiones a Python para crear conexiones seguras. Conforma una capa de alto nivel alrededor de una configuración de la librería OpenSSL de Python [23].

pyCripto: Provee herramientas de cifrado de información a Python [24].

twisted: Es un framework de código abierto implementado en Python especializado para el desarrollo de aplicaciones basadas en red. Sirve de soporte al paquete python para establecer y administrar conexiones de red [25].

ejabberd: Es un servidor Jabber multiplataforma. Desarrollado en el lenguaje Erlang, cuenta con un soporte total de las características del protocolo Jabber [26].

erlang: Lenguaje funcional utilizado especialmente para desarrollo de aplicaciones distribuidas. Tiene soporte para concurrencia, distribución y tolerancia de fallos. Utilizado para el desarrollo del servidor Ejabberd [27].

pyMSNt: Es un transporte desarrollado en Python. Provee una pasarela con la cual el servidor puede comunicarse con la red de MSN Messenger [28].

pyAOLt: Es un transporte desarrollado en Python. Provee una pasarela con la cual el servidor puede comunicarse con la red de AOL. El transporte debe estar instalado en el servidor Jabber, y su operación es transparente para el usuario [29].

pyICQt: Es un transporte desarrollado en Python. Provee una pasarela con la cual el servidor puede comunicarse con la red de ICQ Messenger. El transporte debe estar instalado en el servidor Jabber, y su operación es transparente para el usuario [30].

smack [31]: Es una API de código abierto para construir clientes de mensajería instantánea en Java. Puede ser embebida en aplicaciones para crear componentes de presencia y mensajería instantánea basados en el protocolo Jabber. De esta forma se logra la comunicación entre los paquetes implementados en Phyton y los creados en Java.

postgreSQL: Este conjunto de librerías permite a los programas Java conectarse al motor de base de datos PostgreSQL [32].

view: En este paquete se encuentran las páginas JSP, a través de las cuales se lleva a cabo la comunicación entre la plataforma y el administrador. Por medio de ellas, el administrador puede observar las variables de estado de la plataforma y así realizar funciones de administración. Todas las páginas JSP se comunican con el paquete de control.

control: Contiene las clases que gestionan la lógica de la aplicación y la coordinación de la comunicación entre módulos y entre la plataforma y el administrador.

components: Tiene inmersas las clases Java propietarias que implementan todas las funcionalidades de la plataforma. Cada una de ellas representa un componente encargado de llevar a cabo una funcionalidad específica. En este paquete se cuenta con componentes que llevan a cabo tareas de tarificación, gestión de usuarios, funcionalidades de administración y manejo de mensajería y presencia de usuarios.

communication: Contiene las clases involucradas en la creación de un servicio que escucha peticiones concurrentes provenientes de clientes móviles, las cuales pueden utilizar HTTP (para clientes MIDP1.0) o Sockets TCP (para clientes MIDP2.0).

Cliente Móvil PAUMIM

view: Contiene las clases necesarias para llevar a cabo la comunicación directa entre el sistema y el usuario a través de interfaces gráficas.

control: Agrupa las clases que sirven para gestionar la lógica de la aplicación y la coordinación de la comunicación entre ésta, el usuario y el

servidor. Además, la clase principal de este paquete permite manejar el ciclo de vida de la aplicación.

communication: Contiene las clases involucradas en la comunicación entre el cliente móvil y el servidor PAUMIM. Para establecer dicha comunicación se utilizan clases propietarias para el establecimiento y mantenimiento de la conexión y las clases definidas en JXME (JXTA for J2ME) para la codificación y decodificación de los mensajes de comunicación.

C. *Funcionamiento*

A continuación se describe cada uno de los pasos involucrados en el uso de la plataforma PAUMIM:

Inicialmente se descarga *el cliente móvil (CUMI)* al dispositivo, por medio de la interfaz OTA (Over The Air), o a través de los diferentes medios de comunicación disponibles como cable de interfaz serial, USB, Bluetooth o IrDA.

La primera vez que el usuario inicia la aplicación debe configurar su login y contraseña de ingreso, definida para PAUMIM o para alguno de los sistemas propietarios de mensajería instantánea en los que tiene cuenta, y establecer las direcciones de los contactos con los que va a establecer la comunicación (opcional). Toda esta información es enviada al servidor de mensajería instantánea que es el encargado de almacenarla.

Una vez el servidor reciba la información de los usuarios móviles se encarga de establecer la comunicación con los proveedores de servicio de mensajería propietarios, para solicitar el registro de los contactos y la validación del usuario en ellos.

El usuario puede iniciar una sesión de mensajería instantánea móvil en cualquier terminal que cuente con el cliente móvil (CUMI) instalado mediante el uso de su login y contraseña.

Cuando el usuario inicia una sesión, el servidor de mensajería instantánea se encarga de enviar al móvil toda la información pertinente entre la cual se tiene la lista de contactos, estado de los contactos y mensajes de entrada. De igual forma, el servidor se encarga de recibir los mensajes de los móviles y encaminarlos hacia sus contactos si pertenecen al mismo sistema o a los proveedores propietarios a través del módulo de interoperabilidad.

Cuando el usuario realice la desconexión, PAUMIM se encarga de notificar a todos sus contactos y a los servidores propietarios de mensajería instantánea del cambio de estado a desconectado. Además, almacena todos los mensajes que los contactos le envíen hasta que se puedan entregar en el próximo inicio de sesión.

V. ENTORNO DE EXPERIMENTACIÓN

La Figura 4 muestra el entorno de experimentación utilizado para la realización de las pruebas de PAUMIM sobre ambientes inalámbricos.

A. *Cliente Móvil*

En el cliente móvil se realizaron tres tipos de pruebas:

La primera, en emuladores de Nokia, Sony Ericsson y el Wireless Toolkit de Sun, cada uno equipado con herramientas que permiten observar el consumo de memoria de la aplicación

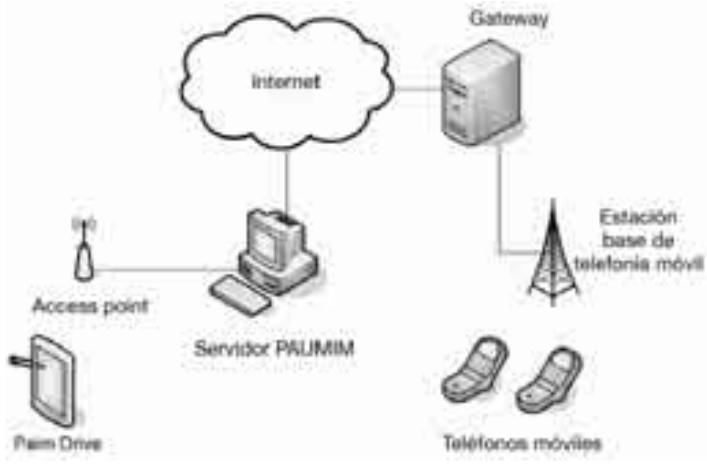

Figura 4. Entorno de experimentación PAUMIM.

y la cantidad de datos enviados y recibidos por la red.

La segunda, en una WLAN (Wireless Local Area Network), se utilizó un punto de acceso inalámbrico y PDAs Palm Drive y Tungsten T5.

La tercera, a través de la infraestructura de datos GPRS de la red celular, instalando la aplicación servidora en una máquina con IP real y el acceso a través de teléfonos celulares Sony Ericsson T610 para el cliente MIDP 1.0 y Nokia 6230 para el cliente MIDP 2.0.

Consumo de memoria

A continuación se muestra el rendimiento experimental observado para los clientes MIDP 1.0 y 2.0. Figuras 5 y 6 presentan datos de consumo de memoria de la aplicación en tiempo de ejecución para las funcionalidades más relevantes.

Las Figuras 7 y 8 muestran el consumo de memoria en el inicio de sesión de la aplicación y actualización de presencia en cuatro ocasiones para los dos tipos de clientes.

El principio de funcionamiento de los clientes MIDP 1.0 y 2.0 es distinto. Para los dos clientes hay un periodo de transición en el cual el dispositivo carga en memoria la aplicación y comienza el periodo de conexión y validación de sesión.

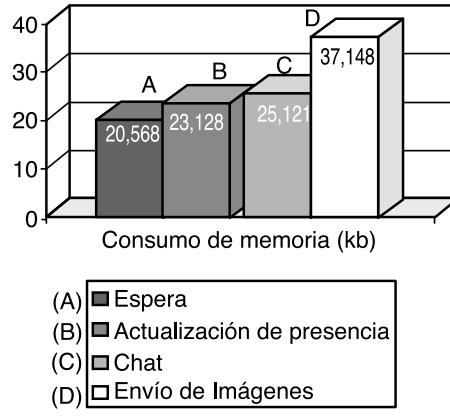

Figura 5. Consumo de memoria para el cliente MIDP 1.0.

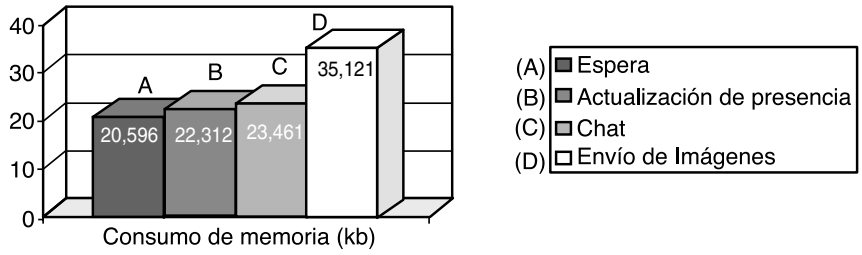

Figura 6. Consumo de memoria para el cliente MIDP 2.0 .

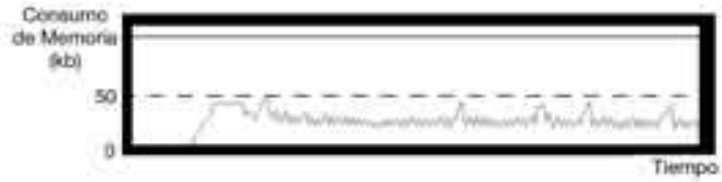

Figura 7. Consumo de memoria para el inicio de sesión y actualización de presencia (cliente MIDP 1.0).

Figura 8. Consumo de memoria para el inicio de sesión y actualización de presencia (cliente MIDP 2.0).

Una vez finalizada esta primera parte, el consumo de memoria toma cierta estabilidad dependiendo del tipo de cliente. En el cliente móvil MIDP1.0 se genera periódicamente un mensaje de sondeo y se envía al servidor, en este momento se crean objetos y se hace uso de la red, por esta razón la gráfica muestra un comportamiento de diente de sierra ya que después de consumir recursos y enviar el mensaje al servidor, se limpia la memoria a través del recolector de basura (garbage collector). Para el cliente móvil MIDP 2.0 la gráfica de consumo de memoria es mucho más estable debido a que la comunicación con el servidor se realiza mediante un socket TCP y no hay que crear objetos ni hacer uso de la red para conservar la conexión. En general el consumo de memoria para las dos aplicaciones es similar, teniendo mayor estabilidad y menor consumo el cliente MIDP2.0 por el principio de funcionamiento.

Tiempo de respuesta

En las Figuras 9 y 10 se puede apreciar el tiempo de respuesta promedio de las aplicaciones para las funcionalidades más importantes en entorno de emulación.

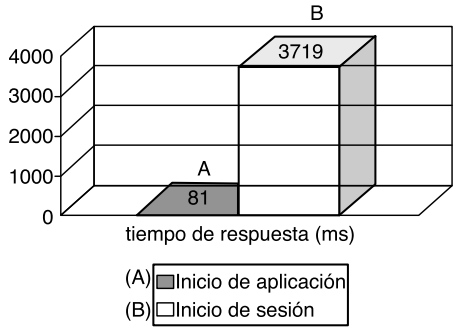

Figura 9. Tiempo de respuesta para el cliente MIDP 1.0.

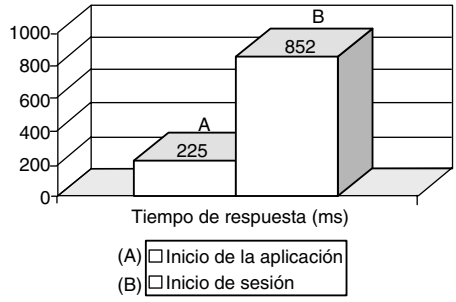

Figura 10. Tiempo de respuesta para el cliente MIDP 2.0.

El inicio de sesión consta del ensamble del mensaje con los datos de sesión para su envío al servidor, recepción de respuesta y actualización de la interfaz gráfica. El comienzo de la aplicación es más rápido para el cliente MIDP1.0 pero su tiempo de respuesta al principio de sesión es mucho más lento que para el cliente MIDP2.0. Lo anterior se debe a que el principio de funcionamiento para la recepción de conexiones por parte para los dos clientes es distinto. Para el cliente MIDP1.0 el servidor tiene que recibir y tratar información transportada en el protocolo HTTP, para lo cual debe empaquetar y desempaquetar la información según el stack de protocolos HTTP. Además de esto, el usuario debe descargar un mensaje por petición, lo cual incrementa el tiempo de respuesta de la aplicación. Para el cliente MIDP2.0 el servidor recibe la conexión TCP/IP y trata la información directamente sin necesidad de realizar empaquetamiento y los mensajes se envían a medida que se vayan creando, por lo tanto no hay cola de espera y el tiempo de retardo presente es únicamente el de transporte por la red.

Tamaño de los mensajes

La Figura 11 muestra el tamaño de los mensajes para cada una de las funcionalidades del cliente móvil. La diferencia entre el cliente MIDP1.0 y MIDP2.0 es que el primero debe realizar el sondeo al servidor y por esta razón consume más ancho de banda.

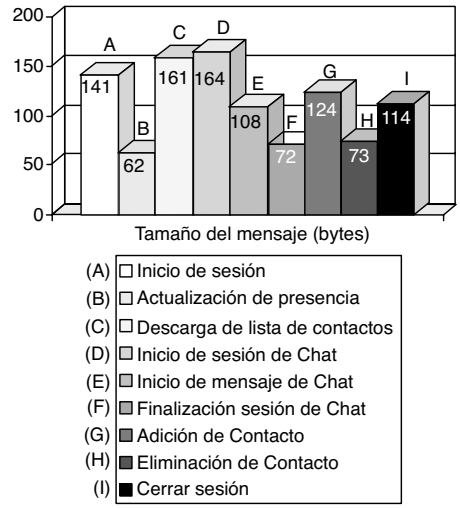

Figura 11. Tamaño de mensajes para el cliente móvil (MIDP 1.0 y 2.0).

B. *Módulo de conexión móvil*

Consumo de memoria y uso de CPU de servicios Jabber

El servidor Ejabberd 1.0 se instaló en el sistema operativo Linux distribu-

ción Debian. El número de usuarios que soporta concurrentemente depende de la configuración y del hardware en el cual se está ejecutando. Con respecto a la configuración, es necesario definir el número máximo de puertos habilitados por Erlang y la cantidad máxima de conexiones.

La eficiencia del servicio tiene una dependencia directa con el ambiente de ejecución del servidor, si corre en una sola máquina o en un cluster de máquinas. Para ambos casos la memoria RAM y la CPU son factores determinantes en el rendimiento. Cuando el servidor es configurado para correr en un cluster de máquinas, el rendimiento del sistema depende, además de la configuración y disposición del cluster, de máquinas. En este proyecto se configuró el servidor Ejabberd para correr en una sola máquina, y se estableció el número máximo de puertos en 1000.

A través de una prueba de estrés, en la cual a partir de una aplicación de escritorio se iniciaron sesiones anónimas en el servidor hasta que éste cerró la recepción de nuevas conexiones, se determinó que con el entorno de ejecución utilizado el servidor puede aceptar entre 720 y 770 usuarios dependiendo de la actividad de cada una de las sesiones de usuario. Las Figuras 12 y 13 muestran el uso de la CPU y el consumo de memoria respectivamente, en los tres intervalos en las cuales se ejecutó la prueba.

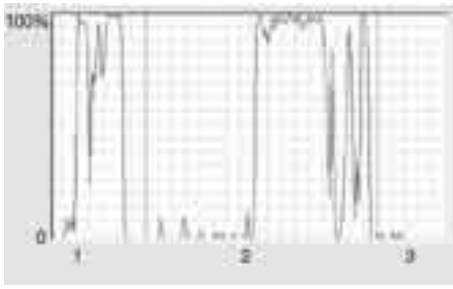

Figura 12. Uso de CPU para el servidor Ejabberd.

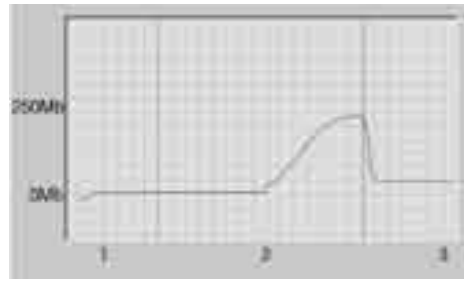

Figura 13. Consumo de memoria para el servidor Ejabberd.

El primero corresponde al arranque del servidor Ejabberd. El segundo a la prueba de estrés, y el tercero, al momento de cierre de las sesiones de usuario iniciadas.

Se puede observar que el uso de la CPU es máximo cuando se inicia el servidor, en el registro de las cuentas y al cerrar el servidor. El consumo de memoria aumenta en la medida en que se empiezan sesiones anónimas hasta el punto en el cual el servidor ya no tiene memoria suficiente, suspendiendo en ese momento la recepción de nuevas conexiones y dejando de incrementar el consumo de memoria RAM.

Submódulo de transportes

Cada uno de los transportes se encuentra en un estado de desarrollo diferente y no tienen todas las funcionalidades implementadas. Sin embargo, el desempeño de cada uno de los transportes fue satisfactorio ya que permitieron llevar a cabo exitosamente la interoperabilidad

con los proveedores de mensajería instantánea externos. En la Tabla I se pueden observar las funcionalidades disponibles actualmente para cada uno de los transportes:

Tabla I. Funcionalidades disponibles de los transportes Jabber.

Funcionalidad	pyMSN	pyICQ	pyAIM
Mensajería	✓	✓	✓
Presencia	✓	✓	✓
Grupos de chat	X	X	✓
Soporte para Vcard	✓	✓	✓
Presencia invisible	✓	X	✓
Notificaciones de escritura	✓	✓	✓
Mensajes HTML	X	X	✓

El principal inconveniente presente en la comunicación de PAUMIM con otros proveedores de mensajería instantánea radica en la modificación regular del protocolo de comunicación que éstos realizan con el objeto de obstaculizar la interoperabilidad con sus competidores. La consecuencia directa es que los desarrolladores deben recurrir a un recurso de ingeniería inversa de los módulos de transporte de cada servidor, para adaptarlos a los cambios, de modo tal que sigan funcionando correctamente.

VI. PAUMIM VS SMS

Para realizar la comparación entre PAUMIM y el Servicio de Mensajería Corta (SMS) [5] se van a abordar tres aspectos fundamentales: precio, acceso al servicio e interoperabilidad.

Precio: En términos generales, los mensajes de texto a través de SMS tienen un precio (para el usuario final) superior al kilobyte de descarga a través de GPRS. Tomando como referencia los precios de un operador de telefonía móvil en Colombia, el precio de un SMS es de 146 pesos, mientras que el precio del Kb equivale a 3.73 pesos. Con base en estos datos se puede decir que en promedio un mensaje por medio de PAUMIM cuesta 2.23 pesos. Con lo anterior se concluye que el precio por envío de mensajes en la plataforma PAUMIM es mucho más bajo con relación a SMS. Además, para el caso del Cliente Móvil MIDP 2.0 el tráfico a cursar es reducido, lo que también contribuye a disminuir costos, haciendo el servicio más atractivo para el usuario final.

Acceso al servicio: El servicio de mensajería corta es prácticamente un servicio universal, soportado por todos los teléfonos que se conectan a la red celular. Para que los 9 dispositivos móviles soporten el cliente PAUMIM deben tener soporte para aplicaciones Java MIDP 1.0 o 2.0 y acceso a la red de datos GPRS.

Interoperabilidad: El objetivo principal de PAUMIM es brindar interoperabilidad entre múltiples proveedores de mensajería instantánea, mediante la adición de módulos funcionales a la plataforma. Por otro lado, para lograr interoperabilidad entre proveedores de servicio de SMS es necesario llegar a acuerdos comerciales de interconexión de redes. Dichos acuerdos son dependientes de las políticas de cada operador y de la regulación de cada país, y son determinantes en el momento del establecimiento de dichos acuerdos.

VII. CONCLUSIONES

A pesar de que J2ME es una especificación, la implementación de la máquina virtual para cada gama de dispositivos tiene algunas variaciones y por tanto los desarrolladores

de aplicaciones móviles basadas en Java Micro Edition deben tenerlas en cuenta.

Por medio de la construcción de un protocolo liviano de bajo consumo de ancho de banda se llevó a cabo la comunicación entre el cliente móvil y el servidor PAUMIM de forma eficiente.

El uso de protocolos universales como TCP/IP en las comunicaciones entre los dispositivos móviles y el servidor PAUMIM hace que se puedan implementar y soportar diferentes tipos de clientes Web y de escritorio, en una gran variedad de tecnologías de desarrollo como Java, PHP, Perl, Python,. Net, C#, VisualBasic, entre otras.

El perfil MIDP1.0 es muy restringido en cuanto al soporte de comunicaciones, debido a que cuenta únicamente con conexiones HTTP, que al ser un protocolo sin estado, obliga a un sondeo permanente del servidor lo cual incrementa el consumo de ancho de banda, disminuye el rendimiento de la aplicación e incrementa los costos de utilización.

El soporte de conexiones MIDP 2.0 es mucho más completo que su antecesor, lo cual permite implementar comunicaciones a bajo nivel por medio de sockets TCP mediante el establecimiento de un circuito lógico entre el cliente móvil y el servidor, reduciendo costos e incrementando en gran medida la eficiencia de la aplicación, a costa de un incremento en la complejidad para el desarrollador.

En cuanto a servidores, Ejabberd ofrece un soporte completo para el protocolo Jabber, proporciona una configuración robusta y adaptable a los diferentes transportes de comunicación como PyMSN, PyICQ y PyAOL, los cuales fueron seleccionados para el desarrollo del proyecto por sus características de libre distribución, alto rendimiento, alta adaptabilidad y fácil configuración.

BIBLIOGRAFÍA

[1] N. Flynn, *Instant Messaging Rules: A Business Guide to Managing Policies, Security, and Legal Issues for Safe IM Communication*, 1era. ed. Broadway, New York: Amacon, 2004, p. 28-36.

[2] J. Sabaté. (2002, Oct.). Mensajería Instantánea. Del placer a los negocios. Revista Consumer EROSKI. [Online]. Disponible: http://revista.consumer.es/web/es/20021001/internet/.

[3] E. Lopez. Mensajería instantánea. PC Magazine. [Online]. Disponible: http://www.x-extrainternet.com/messengers.asp

[4] E. Shiu y A. Lenhart. (2004, Sep). How Americans use instant messaging. Pew Internet and American Life Project, Washington, D.C. [Online].Disponible: http://www.pewinternet.org/pdfs/PIP_Instantmessa ge_Report.pdf

[5] O. M. Caicedo, F. O. Martínez, M. J. Gómez and J.A. Hurtado, "Architectures for Web Services Access from Mobile Devices," in *Proc. 2005 3rd Latin American Web Congress La-Web 2005*, pp. 93-97

[6] SIMPLE Working Group. (2006, Ene.). SIP for Instant Messaging and Presence Leveraging Extensions. IETF. [Online]. Disponible: http://www.ietf.org/

html.charters/simple-charter.html

[7] J. Alan, *SIP: Understanding the Session Initiation Protocol*, 2da. ed. Norwood, MA: Artech House, 2004, p. 17-42.

[8] *Extensible Messaging and Presence Protocol (XMPP): Core*, IETF Standard RFC 3920, Oct. 2004.

[9] *Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence*, IETF Standard RFC 3921, Oct. 2004.

[10] *Session Initiation Protocol (SIP) Extension for Instant Messaging*, IETF Standard RFC 3428, Dec. 2002.

[11] O. Lage. (2005, Jun.).Jabber/XMPP. Facultad de Ingeniería, Universidad de Deusto. [Online]. Available: http://www.jabberes.org/node/529.

[12] J. Muchow, *Core J2ME technology & MIDP*, 1era. ed. San Antonio Road, California: Prentice Hall, 2001, p. 1-26

[13] O. M. Caicedo, F. O. Martínez, J. A. Hurtado y G. A. Ramírez, "Wireless Trace Service for Latin American Craft Sector", in *Proc. 2006 Wireless Telecommunications Symposium*, a ser publicado.

[14] M. Misek, "Jabber: Communicating in Real Time". *EContent Wilton*, vol. 28, pp. 44-45, Feb. 2005.

[15] Jimm.org. (2006, Mar.). Jimm Mobile Messaging. SourceForge.net. [Online]. Disponible: http://www.jimm.org/

[16] JXTA.org. (2005, Dec.). JXTA for J2ME (CLDC/MIDP). JXTA.org. [Online]. Disponible: http://jxme.jxta.org/

[17] L. García, R. Camacho y F. O. Martínez, "Desarrollo de Aplicaciones P2P para Dispositivos Móviles haciendo uso de los Protocolos JXTA (JXTA + J2ME) – El Punto de Encuentro Virtual P2P", presentado en las V Jornadas de Investigación y Desarrollo en Informática (JIDI) en el marco del evento TECNOCOM 2005, Medellín, Colombia, 2005.

[18] W. Yeager, J. Williams, "Secure peer-to-peer networking: the JXTA example". *IT Professional*, vol. 4, pp. 53-57, Abr. 2002.

[19] Mobber Project. (2005, Jun.). Mobber- mobile jabber communicator. SourceForge.net. [Online]. Disponible: http://mobber.gryf.info/en/

[20] Asociación ADITEL. (2003, Sep.). Introducción a Jabber. Universitat Jaume I de Castellón, España. [Online]. Disponible: http://www.jabberes.org/introduccion

[21] *Instant Messaging / Presence Protocol Requirementse*, IETF Standard RFC 2779, Feb. 2000

[22] A. Downey, J. Elkner y C. Meyers. (2002, Ene.4). *How to Think Like a Computer Scientist Learning with Python*. (1era ed.) [Online]. Available: http://www.ibiblio.org/obp/thinkCSpy/

[23] pyOpenSSL Project. (2004, Ago.). Python interface to the OpenSSL library. SourceForge.

net. [Online]. Disponible: http://pyopenssl.sourceforge.net/

[24] Pycripto. (2005, Dec.). Python Cryptography Toolkit. SourceForge.net. [Online]. Disponible: http://www.amk.ca/python/code/crypto.html

[25] I. Shtull-Trauring. An Introduction to the Twisted Networking Framework. Presented at O'Reilly Emerging Technology Conference. [Online]. http:/ nlamp.com/pub/a/python/2004/01/15/twisted_intro.html

[26] Ejabberd Web site. (2005, Dec.) Ejabberd server. Ejabberd project. [Online]. http://ejabberd.jabber.ru/

[27] Erlang Web site. (2006, Mar.). Erlang Open source. Erlang.org. [Online]. http://www.erlang.org

[28] PyMSNt Web site. (2006, Feb.). Python based MSN Transport for Jabber. Jabberstudio..org. [Online]. http://msntransport.jabberstudio.org

[29] PyAIM-t Web site. (2005, Dec.). AIM transport for Jabber. Blathersource.org. [Online]. http://pyaim-t.blathersource.org/

[30] PyICQ-t Web site. (2005, Dec.). ICQ transport for Jabber. Blathersource.org. [Online]. http://pyicq-t.blathersource.org/ 10

[31] Smack API Web site. (2006, Mar.). Simple and Powerful Java client API for XMPP. Jivesoftware.org. [Online]. http://www.jivesoftware.org/smack

[32] PostgreSQL Web site. (2006, Feb.). PostgreSQL JDBC Driver. Postgresql.org. [Online]. http://jdbc.postgresql.org/

Óscar Caicedo recibió el título de Ingeniero en Electrónica y Telecomunicaciones de la Universidad del Cauca, Colombia, en 2001. En la misma institución recibió el título de Especialista en Redes y Servicios Telemáticos, en 2003. Actualmente realiza su tesis de Maestría en Ingeniería con Énfasis en Telemática en la Universidad del Cauca y se desempeña como docente del Departamento de Telemática de la Facultad de Ingeniería Electrónica y Telecomunicaciones de la misma universidad. Como miembro del Grupo de Ingeniería Telemática (GIT), su área de investigación se centra en la Ingeniería del software aplicada a la construcción de arquitecturas y plataformas para el desarrollo y despliegue de aplicaciones sobre dispositivos móviles, redes inalámbricas y redes móviles celulares.

Andrés Caicedo recibió el título de Ingeniero en Electrónica y Telecomunicaciones de la Universidad del Cauca, Colombia, en 2006. Actualmente realiza su primer año

de estudios de Maestría en Ingeniería con Énfasis en Telemática en la Universidad del Cauca. Además, es miembro del grupo de interés en desarrollo de aplicaciones móviles e inalámbricas w@pcolombia y director de proyectos de tecnología Topp Allians S.A.; donde centra su investigación en el desarrollo de servicios y aplicaciones para dispositivos móviles, desarrollo de aplicaciones empresariales, redes inalámbricas de transmisión de datos, redes globales de información y redes móviles celulares de 2.5G y 3G.

Edwin Figueroa recibió el título de Ingeniero en Electrónica y Telecomunicaciones de la Universidad del Cauca, Colombia, en 2006. Actualmente es miembro del grupo de interés de desarrollo de aplicaciones móviles e inalámbricas w@pcolombia, donde centra su investigación en el desarrollo de servicios y aplicaciones para dispositivos móviles, redes inalámbricas de transmisión de datos, redes globales de información y redes móviles celulares de 2.5G y 3G.

Francisco Martínez recibió el título de Ingeniero en Electrónica y Telecomunicaciones de la Universidad del Cauca, Colombia, en 2003. Actualmente cursa tercer semestre de Maestría en Ingeniería con Énfasis en Telemática en la Universidad del Cauca y se desempeña como docente del Departamento de Telemática de la Facultad de Ingeniería Electrónica y Telecomunicaciones de la misma universidad. Como miembro del Grupo de Ingeniería Telemática (GIT), su área de investigación se centra en la Ingeniería del Software aplicada a la construcción de arquitecturas y plataformas para el desarrollo y despliegue de aplicaciones sobre dispositivos móviles.

Javier Hurtado recibió el título de Ingeniero en Electrónica y Telecomunicaciones de la Universidad del Cauca, Colombia, en 2001. En la misma institución, recibió el título de Especialista en Redes y Servicios Telemáticos, en 2004. Actualmente cursa tercer semestre de Maestría en Ingeniería con Énfasis en Telemática en la Universidad del Cauca y se desempeña como docente del Departamento de Telemática de la Facultad de Ingeniería Electrónica y Telecomunicaciones de la misma universidad. Como miembro del Grupo de Ingeniería Telemática (GIT), su área de investigación se centra en el estudio de los protocolos de señalización en

redes de nueva generación y la construcción de arquitecturas y plataformas para el desarrollo y despliegue de aplicaciones de nueva generación sobre dispositivos móviles.