Universidad de la República
Facultad de Ingeniería.
Ingeniería en Computación

Estudio del Open/Free (GNU/Linux) como plataforma de servicios de red en entornos empresariales

Servicios de Mensajería Instantánea y Presencia en la Empresa

Tutor : Ariel Sabiguero

Daniel Caraballo

Mario Madera

Marcelo Odin

Índice de contenido

1 INTRODUCCIÓN

La proliferación y madurez de paquetes de software de mensajería instantánea (M.I.) como AOL Instant Messenger, Microsoft MSN, Yahoo Messenger impulsó a millones de usuarios en todo el mundo a registrarse y usar activamente dichas herramientas como medio de comunicación, para compartir archivos Peer-to-Peer y aún para video conferencias. Otros de los protocolos más usados es ICQ, que debe leerse como "I Seek You" y no en forma deletreada. Este protocolo fue uno de los primeros en utilizarse, desarrollado por Mirabilis y actualmente en uso.

Es palpable que la mensajería instantánea esta obteniendo una adopción importante dentro del público en general. Cada persona que cuenta con un servicio de M.I. tiende a utilizarlo tanto a nivel hogareño como dentro de la empresa en la que trabaja, no solo con fines personales sino también con compañeros de trabajo con fines puramente laborales. De acuerdo a International Data Corp en el 2001 veinte millones de usuarios usaban M.I. en los negocios prediciendo para fines del 2005, 300 millones[1] y se espera un incremento de mercado para estos servicios de 400 millones de dólares para el año 2008 [20]. Esto muestra que la gente se siente cómoda con este servicio y esta lo suficientemente preparada para utilizarlo dentro de este entorno.

Lamentablemente esta forma informal y sin control de utilización tiene dos grandes impactos en el funcionamiento de las empresas:

- El primero es la pérdida de productividad y mala utilización de recursos de I.T. por parte de los empleados.

- El segundo, considerando que la mayoría de las cuentas de mensajería instantánea están creadas en servidores públicos en Internet, es la seguridad en sus distintos aspectos, por ejemplo escape de información sensible a la empresa.

Gracias a estos factores, a la desinformación por parte de la empresa en si misma acerca de beneficios empresariales de la utilización de esta tecnología y el desconocimiento de sistemas de Mensajería Instantánea privados, la adopción de este servicio a nivel corporativo es mucho más lenta y cautelosa.

Por otro lado la tecnología M.I. tiene varios aspectos ventajosos para cualquier empresa sin importar su porte. Por ejemplo conocimiento de presencia y disponibilidad de compañeros de trabajo en tiempo real, mayor capacidad para trabajo en grupo, contacto con proveedores y clientes más dinámico, etc.

Creemos que los servicios de Mensajería Instantánea propio para un entorno Empresarial tienen un gran potencial para aumentar la productividad de la Empresa y a un bajo costo. Consideramos dicha tecnología como un complemento valioso a los medios actuales de comunicación empresarial y más allá de que su utilización se esté adoptando cada vez con mayor magnitud nos encontramos con una mala utilización de la tecnología o con un desaprovechamiento de todo su potencial y funcionalidades.

Veremos en los siguientes capítulos el 'estado del arte' de dicha tecnología a nivel empresarial abarcando temas como: beneficios para la empresa, mitigación de problemas acarreados por el mal uso de la tecnología, requerimientos empresariales del servicio, protocolos de mensajería propietarios y Open, soluciones comerciales/propietarias existentes y soluciones Open/Free existentes.

[1]Si bien los estudios de IDC son pagos, el dato fue sacado de Blue Coat, empresa dedicada a seguridad de servicios Web. Dicho estudio quedó fuera de línea antes de la finalización de este documento.

2 EL USO DE LA MENSAJERÍA INSTANTÁNEA EMPRESARIAL

2.1 ¿QUE BENEFICIOS LE APORTA A LA EMPRESA?

Como mencionáramos en la introducción, no todos los aspectos de esta tecnologías son adversos. Muy por el contrario se identificaron varios aspectos de su utilización que lo hacen altamente favorable para las empresas. Algunos de estos puntos son:

- **Abarata los costos de telefonía tanto local como de larga distancia**. Con los precios actuales de telefonía, 12 horas de llamadas telefónicas en horario pico"[1] equivale al costo de un ADSL barato[2]. (El valor del computo con las tarifas vigentes a Febrero de 2005 de $1.09 y de 11:00 a 18:00hs la cadencia para la aplicación de cómputos urbanos es de 1 cómputo por minuto, la tarifa única nacional para llamadas al Interior es de $3.23 por minuto y el valor del minuto de llamadas larga distancia internacionales varían de $4.34 a $16.04 por minuto).

- **Contactar a más de una persona a la vez.** Considerando que telefónicamente se puede atender a una sola persona a la vez, con este medio de comunicación se puede estar atendiendo a varios clientes, consultando a varios proveedores a la vez. (p.ej. En una agencia de viaje, estoy con el cliente y me contacto con hoteles, líneas aéreas, etc. para armar el booking; consulta en tiempo real con varios proveedores sobre un producto especifico que no esta en stock).

- **Conocimiento de presencia y disponibilidad de compañeros de trabajo en tiempo real**. Al iniciar el cliente de M.I., la lista de contactos me indica el estado de cada uno de ellos. Se puede saber así si la persona esta no disponible.

- **Mayor capacidad para trabajo en grupo**. Imaginemos la situación donde por falta de espacio no puedo discutir un tema de trabajo con uno o más compañeros sin afectar al resto. Las salas de conferencias, o directamente la comunicación punto a punto, me brindan una solución eficiente a este tema. Observar además que el grupo puede estar conformado por personas geográficamente distantes.

- **Más facilidad para realizar múltiples tareas.** Según el estudio presentado en la conferencia de CSCW **(Computer Supported Cooperative Work)** de 2002 acerca de la utilización de la mensajería en la empresa AT&T, en el 85% de conversaciones laborales mediante M.I. los empleados realizaban otras tareas. Esto es coherente ya que es mucho más fácil realizar más tareas y atender asincrónicamente una conversación de M.I. que cuando se está al teléfono, puesto que al teléfono se espera una conversación más dinámica y más fluida.

- **Contacto dinámico con clientes**, mejorando así la calidad de los servicios ofrecidos. Se puede incluso crearle una cuenta personalizada dentro de la empresa y en algunos casos brindar un servicio de acceso basado en web a fin de no forzar al mismo instalar otro cliente más.

- **Contacto cooperativo con empresas complementarias**, agilizando la cadena de abastecimiento y dándole una mayor flexibilidad a la cooperación comercial entre empresas. Este punto es no menor, ya que facilitaría la "federación" comercial, obteniendo los involucrados una mayor competitividad en el mercado.

[1] El horario pico es de 11:00 a 18:00, coincidiendo prácticamente con el horario de oficinas.

[2] Ver tabla de costos en apéndice 8.10

- **Bajo costo de mantenimiento y administración una vez implantado el servicio**.

- **No requiere recursos adicionales** de infraestructura de red a los ya existentes dentro de la empresa. Exceptuando la transferencia de archivos, se requiere muy poco ancho de banda para los mensajes en cualquiera de los protocolos. (La mayoría de las personas que históricamente utilizan este servicio lo realizaban – muchas todavía lo hacen- a través de líneas discadas).

Todos estos argumentos indican que esta tecnología posee un potencial interesante para incrementar la productividad de las empresas a un bajo costo. También resulta interesante analizar como mitigar los problemas existentes debido a su mala utilización actual y ver que tipos de soluciones existen sobre todo en el mundo Open Source y si éstas están lo suficiente maduras para cumplir los requerimientos necesarios de una implantación corporativa de este servicio.

Naturalmente, como toda innovación se enfrentará a una fuerte resistencia al cambio y su adopción dependerá de las circunstancias particulares de cada empresa y de los beneficios tecnológicos y comerciales que la M.I. pueda presentarles.

2.2 PROBLEMAS ACTUALMENTE ASOCIADOS A LA UTILIZACIÓN DEL SERVICIO

Mala utilización de la tecnología y de recursos de la empresa

Uno de los puntos más notables en cuanto a la mala utilización de la mensajería instantánea dentro de una empresa viene arrastrado por la facilidad de acceso a Internet con la que muchas empresas cuentan y que para el usuario no tiene costo en su utilización (este caso es muy similar a la utilización de la navegación por Internet). Esto generalmente lleva a que el empleado pierda tiempo de trabajo en conversaciones con amigos o familiares. Muchas empresas consideran a el servicio de mensajería instantánea como dispensadores de agua "virtuales". Incluso si se dispone de un servicio propio de mensajería, el uso del mismo puede ser susceptible para fines sociales.

Este tipo de problemas puede ser similar al de estar hablando permanentemente por teléfono pero sin poder detectarlo a menos que se este parado al lado de la persona.

Otro problema es que para que el empleado pueda utilizar su servicio de mensajería favorito debe de obtener el software cliente de Internet lo que implica un abuso de recursos de infraestructura y de ancho de banda de la red corporativa.

Problemas de Seguridad

Como se mencionó anteriormente, la mayoría de los servicios de mensajería utilizados son los de difusión masiva (ICQ, AOL, MSN, Yahoo), dichos servicios se encuentran en servidores públicos en Internet lo cual acarrea con varios problemas de seguridad para la empresa:

Algunos protocolos de Mensajería Instantánea transmiten el tráfico sobre puertos como http (80), entonces la empresa debe de tomar medidas de seguridad más estrictas.

La transferencia de archivos entre 2 clientes en general no es contemplada por las políticas de antivirus centralizadas implementadas en la empresa. Además, la posibilidad de que cualquiera se añada como contacto sin autorización ha servido como un catalizador a virus como el Goner.A (que usa la lista de contactos de ICQ para replicarse). Otro tipo de ataques pueden venir dentro de un mensaje con referencia a URLs maliciosas.

También hay preocupación con un gran número de agujeros de seguridad que presentan los clientes de M.I.. Han habido varias notificaciones de la CERT respecto del AOL instant messenger[1].

A esto se le suma la protección de la propiedad intelectual. Amenazas a la privacidad causada por los clientes de M.I. en el ámbito de trabajo incluyen filtrado de información personal, exposición de direcciones IP, pérdida de información confidencial. La mayoría de los empleados no están al tanto de que si intentan enviar un mensaje a alguien del otro lado de la sala, ese mensaje sale fuera del firewall corporativo y viaja a través de Internet y de vuelta al ambiente corporativo (en algunos protocolos, en texto plano).

Existe además la posibilidad de que se ejecuten"bots" que recogen información sobre cuentas de estos servicios masivos para luego usarlos en técnicas de bombardeo. El resultado, tendríamos un infinito número de pedidos de autorización de inclusión en listas de amigos de gente que no conocemos, con las evidentes molestias que esto ocasiona.

Se puede hacer una analogía entre la utilización de un servicio público de M.I. (Microsoft MSN o Yahoo Messenger) por parte de una empresa y utilizar una cuenta de e-mail pública como Hotmail o Yahoo. Obviamente es mucho mas seguro y confiable tener un servicio propio y controlado.

La mitigación de este tipo de problemas lleva a una serie de requerimientos necesarios por parte de la implementación de esta tecnología para poder convertirlo en una herramienta verdaderamente productiva.

[1] http://www.kb.cert.org/vuls/html/search keyword AOL

2.3 REQUERIMIENTOS CORPORATIVOS PARA SISTEMAS DE MENSAJERÍA INSTANTÁNEA

Tomando en consideración características de productos comerciales para empresas, estudios realizados por empresas como Macroescape Solutions[1] y analizando los problemas detectados anteriormente, establecimos e identificamos distintos requerimientos indispensables para implementar en forma exitosa y segura una infraestructura para mensajería instantánea en el ambiente empresarial. Estos requerimientos fueron desglosados en requerimientos de usuarios, de seguridad, de infraestructura y de los clientes.

2.3.1 Requerimientos de usuario

Historial de mensajes	El usuario necesita poder ver los mensajes que ha ido intercambiando en cada una de las conversaciones.
Transferencia de archivos	Intercambiar archivos P2P es un requerimiento en las redes empresariales (se debe tener la posibilidad de restringir este servicio para el usuario final).
Chat multiusuario	Los usuarios deben poder mantener conversaciones simultaneas con varias personas a la vez, p.e. para poder resolver un problema para el cual se requieren varios equipos.
Invitar gente	Va de la mano con el requerimiento de chat multiusuario.
Directorio de usuarios y facilidades de búsqueda	Se debe poder buscar a cualquier empleado en una lista y añadirlos bajo permisos.
Listas de contactos almacenadas en el servidor	Por motivos de respaldos y movilidad, las listas de contactos deben estar almacenadas en el servidor.
Soporte para múltiples plataformas de mensajerías	El software debe permitir comunicación a través de gateways (en forma de cliente) con sistemas IRC, SMS, SIMPLE, XMPP, SMTP, AIM, ICQ, MSN, Yahoo.
Almacenamiento y entrega de mensajes fuera de línea	Cuando el usuario no está conectado, poder dejarle un mensaje, a los efectos de abaratar los costos de almacenamiento del correo de voz.(en caso de se utilicen centrales telefónicas con almacenamiento en HD)

2.3.2 Requerimientos de seguridad

Encriptación robusta de M.I.	Los mensajes internos y con clientes externos deben estar encriptados.
Auditar / logging / archivar	Es conveniente tratar la M.I. como un e-mail y utilizar capacidades de auditoría y logging para rastrear comportamientos inadecuados en los empleados.
Soporte de certificados digitales PKI	Para uso futuro de modo de evitar fraude de identidad.
Virus scanning	Evitar que virus maliciosos como GONER.A entren al ambiente

Autenticación federada	Forzar nombres de usuario basados en el domino de la empresa (@miempresa.com) y extender la política a los clientes de la misma(@micliente.com).
Control de Contactos y de Salida	Es deseable poder controlar el destinatario del mensaje de manera activa para asegura de que el uso de M.I. sea solo por razones de negocios.
Opción de restringir creación de cuentas de usuario por parte del cliente	Para control total de las cuentas de los usuarios

2.3.3 Requerimientos de infraestructura

Indizado y búsqueda de mensajes	Permitir a los administradores indexar y buscar en los mensajes
Clustering de servidores	Para alta disponibilidad del sistema de mensajería
Integración LDAP	Para utilizar un servidor LDAP para autenticación centralizada junto con los diversos servicios de información de la empresa.
Consola para definir políticas	Se necesita una consola de administración con la capacidad de definir políticas corporativas para la mensajería instantánea, como ser "Servicio de reparación para clientes sólo puede hablar con el cliente A".
Control de identidad	Asegurar que ud. está hablando con quien cree que está hablando
WAP/Wireless/SMS/GPRS	Para utilización de plataformas móviles en General.
Server2Server comunication para mejor escalabilidad horizontal y distribución geográfica (análogamente como un servicio de correo electrónico)	Esta permitiría a empresas que tienen sucursales o a grupos empresariales distribuidos geográficamente, utilizar una misma red de mensajería cerrada. Además de permitir una escalabilidad distribuida (que ayuda a distribuir la carga de tráfico) también mejora la tolerancia a fallas
Soporte de multiples DBMS	Ser capaz de soportar un amplio rango de bases de datos tanto para almacenar información propia del servidor como para rastreo y almacenamiento de mensajes.
Convergencia de Servicios	Capacidad de unir el sistema de M.I. a los servicios de correo electrónico, noticias, videoconferencia y fax, así también como la capacidad de poder interactuar con aplicaciones en la empresa.

2.3.4 Requerimientos de programas clientes

Cliente basado en web	El software de IM necesita poder ser empleado tanto como una aplicación cliente servidor, como una herramienta basada en web para poder soportar el mayor número de clientes remotos.

Interfaz amigable e intuitiva	El usuario generalmente está acostumbrado a clientes como MS Messenger o Yahoo Messenger, es recomendable que el cliente deba de tener interfaz similar con similares funcionalidades
Disclaimer	Debe poder insertarse un disclaimer estándar, lo mismo que sucede con el e-mail enviado desde una empresa.

3 SOLUCIONES PARA EMPRESAS

Si bien existen soluciones tanto comerciales como Open/Free para Mensajería Instantánea, ninguna llega a cubrir aún la totalidad de requerimientos antes presentados.

3.1 SOLUCIONES COMERCIALES/PROPIETARIAS

Cada vez mas proveedores de Software Comercial apuestan a la Mensajería Instantánea como un servicio importante dentro de la Empresa y ofrecen soluciones muy completas, este es un dato no menor ya que generalmente el mercado dicta las tendencias.

Generalmente las soluciones Comerciales basadas en protocolos propietarios o con extensiones propietarias de protocolos abiertos, como por ejemplo Microsoft Live Communications Server (basado en SIP/SIMPLE con extensiones propietarias), si bien son muy completas en servicios y funcionalidades, están muy integradas a software específico como Microsoft SQL Server o Microsoft Active Directory. Incluso implementan funcionalidades extras como ser el de integración con Ofimática que va más allá del alcance de un sistema de Mensajería en si (aunque no deja de tener un valor agregado).

A modo de ejemplo haremos un resumen de las características y funcionalidades ofrecidas por **Microsoft Live Communication Server 2003 estándar y 2005 Enterprise Edition**, se verán los puntos referentes estrictamente a funcionalidades de M.I. y Presencia (M.I. y P.).

Presencia y Contactos: Ambas ediciones soportan control de presencia (activo, ocupado y fuera de oficina) , notificación de Offline y lista de contactos administrada en servidor si se requiere.

Solo la edición 2005 Enterprise da soporte para lista de más de 150 contactos y simular estar Offline.

Servicios de colaboración en tiempo real: Ambos servidores ofrecen los servicios de Mensajería instantánea, Audio y video, transferencia de archivos punto a punto

Performance de Servidores: La edición 2003 soporta hasta 8000 usuarios activos, la 2005 Enterprise hasta 15000 y con posibilidad de clusters hasta 100000 usuario activos, esta última también tiene capacidad de failover tanto como para un mantenimiento de servicio planificado como para fallas no planificadas.

Seguridad: Ambas ediciones cuentan con archivado y loggeo conversacional, encripción de mensajes, audio y video y autenticación por Kerberos o NTLM (NT Lan Manager)

Administración: Administración vía MMC (Microsoft Management Console), Integración con Active Directory y habilitación de servicios (por usuario, por grupo o por feature) es soportada en ambas versiones, la versión 2005 Enterprise cuenta también con una herramienta de administración basada en XML.

API para extensiones: Ambos productos cuentan con una API (Server Application Programming Interface).

Si bien el producto tiene conectividad con MSN, AOL y Yahoo, estos se ofrecen por un precio aparte. EL producto **NO** cuenta con servicios de chatrooms integrados, Gateways para

SMS, Anti-virus y verificación de contenido, alertas, Balanceo de carga o servicios de Backup/Restore, pero existen empresas asociadas a Microsoft que brindan todos estos servicios.

Existen también soluciones comerciales muy completas basadas íntegramente en protocolos abiertos como ser Antepo OPN Server (basadas en XMPP y SIMPLE) y Jabber XCP (basada en XMPP) , con las ventajas de arquitectura que dichos protocolos ofrecen y con mucha independencia del software de base que utilizan en cuanto a Sistemas Operativos, LDAP, Bases de datos.

A modo de ejemplo haremos un resumen de las características y funcionalidades ofrecidas por **Antepo OPN Server.**

- Permite conversaciones 1 a 1 y chatrooms con características mejoradas en los mensajes como ser: prioridad, encripción, entrega garantida, alertas customizables, manejo de lista de contactos, "aparentar estados de Offline" según la conveniencia del usuario.
- Permite creación de Chatrooms públicos o restringidos para grupos reducidos de usuarios
- Integración con Outlook
- Arquitectura distribuida: diseño basado en componentes que pueden ser distribuidos por la red en múltiples equipos, lo cual provee un alto grado de escalabilidad y tolerancia a fallos eliminando puntos de falla centralizados
- Almacenamiento Persistente de datos en Base de Datos como ser lista de contactos, listas anti-spam, etc.
- Capacidad para Failover
- Capacidad de auto recuperación: Los componentes del servidor pueden reiniciar conexiones perdidas de clientes y de reconectarse a los componentes distribuidos existentes para recuperar la disponibilidad del servicio.
- Integración con varios sistemas de LDAP incluido Active Directory y OpenLDAP (tanto para autenticación como para creación y procesamiento de reglas de autorización)
- Dominios virtuales: varios dominios lógicos pueden ser hosteados en un mismo servidor
- Opciones de Seguridad y Control: integración con Kerberos para single-signon (en entornos Windows), Encriptación de tráfico, TLS (Transport Layer Security) para comunicaciones Cliente-Servidor y Servidor-Servidor, autenticación, federación de sistemas foráneos de modo completamente controlado, soporte pasa SASL
- Administración: Consola Web de administración
- Servicios de log y auditoría
- Soporte multi-plataforma : esta desarrollado en JAVA
- Multi-protocolo: Soporte para XMPP, SIMPLE y permite intercambio de mensajes con sistemas Open Source (Jabber), propietarios (MS L.C.S.) y sistemas legados (IBM)
- Sistemas operativos soportados : Windows 2000/2003/XP, Red Hat Linux, Sun Solaris 8 y 9

Encontramos que algunos de estos servidores requieren una buena infraestructura de Hardware ya que están basados en JAVA (Antepo OPN).

También heredan los requerimientos del sistema operativo donde corren, o se necesita infraestructura extra para Sistemas de Base de Datos, también están restringidos requiriendo licenciamientos extra del lado del Sistema operativo para que ciertos Features funcionen correctamente como lo es el Live Communications Server de Microsoft.

3.2 MENSAJERÍA INSTANTÁNEA OPEN /FREE PARA GNU/LINUX

Presentaremos a continuación un análisis de las soluciones existentes de mensajería instantánea Open/Free para GNU/Linux, el análisis abarcará:

- Arquitectura de sistemas de M.I.
- Protocolos abiertos para M.I.
- Servidores de M.I.
- Caso de estudio: Implantación de servicio en Facultad de Ingeniería

3.2.1 Arquitectura de sistemas de M.I.

Tanto en soluciones Open/Free como en comerciales para GNU/LINUX, se puede establecer la existencia de al menos dos clasificaciones basadas en la arquitectura utilizada, dicha arquitectura generalmente va de la mano del protocolo que implementa:

Nos encontramos lógicamente con soluciones cliente-servidor y soluciones punto a punto, también conocidas como **serverless**.

3.2.1.1 Soluciones de tipo serverless

Estas, básicamente, son programas que sirven tanto de cliente como de servidor, están basados en tecnología Peer to Peer y las funcionalidades que ofrecen no las hacen compatibles para soluciones empresariales, algunos puntos a tomar en cuenta son:

- Auditoría – Los mensajes no quedan almacenados en una base de datos que los centralice, impidiendo así el control del flujo de mensajes.
- Control de contactos – Tampoco se pueden controlar los contactos disponibles a los usuarios pudiendo estos mantener conversaciones con personas no habilitadas por la empresa.
- Red Integrada – en virtud de que no es una arquitectura cliente-servidor, es más difícil mantener integrada la mismas, por ejemplo la definición de los nombres de usuario para reflejar la pertenencia a un determinado dominio o por otro lado la integración a otras tecnologías de red.

Las soluciones serverless pueden ser un reemplazo para entornos donde no se necesite más que un reemplazo para WinPopUp, pero no se pueden tomar en cuenta para cubrir los requerimientos antes descritos para un buen servicio corporativo.

Algunos puntos a favor que presenta esta arquitectura son la facilidad de implantación y el casi inexistente costo de mantenimiento.

Algunos paquetes de software Serverless de M.I. son :

- Trillian de Cerullean Studio (utiliza el protocolo Rendevouz de Apple)
- WinPOPUp de Microsoft (utiliza protocolo propietario de Microsoft)

3.2.1.2 Soluciones Cliente-Servidor

Estas representan al grupo mayoritario tanto en paquetes existentes como en difusión. Por sus características arquitectónicas, es de suponer que una solución cliente-servidor brinde mucho más flexibilidad para la implementación de una plataforma empresarial de Mensajería Instantánea. Al tener control sobre el flujo de mensajes que transportan los clientes, se pueden lograr registros de mensajes, auditoría, control de utilización de recursos de red. También se puede lograr mejor control de presencia de usuarios, la implementación de chatrooms es posible sin una estrategia de broadcasting en toda la red, etc. Las ventajas de una arquitectura Cliente-Servidor son notorias.

3.3 PROTOCOLOS ABIERTOS PARA M.I.

Habiendo seleccionado una implementación cliente-servidor, estudiaremos los protocolos de mensajería disponibles, se realizará una breve reseña sobre protocolos propietarios y luego se presentarán los protocolos SIMPLE, XMPP (antes Jabber) e IRC, protocolos abiertos

Los sistemas de mensajería basados en protocolos propietarios o cerrados, tales como los que utilizan MSN de Microsoft, AIM de AOL, ICQ de Mirabilis y Yahoo entre otros, también son soluciones cliente-servidor pero el hecho de que dichos protocolos sean propietarios, hace muy difícil la tarea de implementación de servidores Open/Free que soporten los mismos, generalmente se requiere de ingeniería reversa lo cual puede resultar muy costoso sobre todo si el protocolo cambia seguido para añadir nuevas funcionalidades (otro *lei-motive* por parte de empresas comerciales acerca del cambio de sus protocolos es justamente dificultar la tarea de ingeniería reversa para que servidores que no sean implementados por la empresa propietaria queden obsoletos)

A pesar de esta característica, existen algunos servidores (ICQ) para plataforma GNU/Linux. Los mismos han sido desarrollados utilizando estas técnicas y al momento de la investigación (07/2004) estaban discontinuados.

El siguiente paso lógico de la investigación fue sobre el estudio de los protocolos abiertos para mensajería instantánea.

3.3.1 Protocolos Abiertos (SIMPLE, XMPP, JABBER, IRC)

A continuación presentamos cada uno de los 4 protocolos abiertos más importantes en lo que respecta a mensajería instantánea.

Nota: XMPP deriva del protocolo Jabber por lo cual se realizará específicamente el análisis de XMPP explicando su evolución desde Jabber y sus principales diferencias.

3.3.2 IRC

IRC (RFC1459) o "Internet Relay Chat". Fue originalmente escrito por Jarkko Oikarinen en 1988 y fue designado como un reemplazo del arcaico programa “talk”.

IRC proporciona un medio de comunicación en tiempo real basado en chatrooms o canales donde un usuario puede unirse a canales para hablar con todas las personas presentes en dichos canales, o también brinda la posibilidad de entablar una conversación entre 2 personas de el mismo canal, existen redes de servidores IRC públicas, la red más grande es Efnet (la red

originaria de IRC, muchas veces conteniendo más de 32.000 usuarios a la vez), seguida por Undernet, IRCnet, DALnet y NewNet

IRC ganó fama internacional durante la Guerra del Golfo Pérsico en 1991 donde usuarios de todo el mundo se reunían en un canal donde se registraban los reportes en vivo de forma online

La arquitectura propuesta por el protocolo es cliente servidor

3.3.3 SIMPLE

Las Session Initation Protocol for Instant Messaging and Presence Leveraging Extensions (SIMPLE) son un conjunto de extensiones que le permitan al Session Initation Protocol (SIP) brindar las funcionalidades básicas para el manejo de presencia y de mensajería instantánea, tal como su nombre lo indica.

Estas extensiones fueron desarrolladas por la IETF y están siendo propuestas por el grupo de trabajo sobre SIMPLE de la Internet Engineering Task Force (IETF). De todos los internet-drafts propuestos (21), 3 fueron pasados a RFC :

- A Presence Event Package for the Session Initiation Protocol (SIP) (RFC 3856)
- A Watcher Information Event Template-Package for the Session Initiation Protocol (SIP) (RFC 3857)
- An Extensible Markup Language (XML) Based Format for Watcher Information (RFC 3858)

El resto de las especificaciones continúan como Drafts

SIMPLE provee como primitivas las siguientes señales (*notar que el término "señal" lo hereda de su origen en la telefonía*):

INVITE	Es la encargada de marcar el comienzo de sesión.
BYE	Usada para finalizar la sesión.
Estas dos son heredadas de SIP.	
MESSAGE	Se utiliza para enviar un mensaje de tipo disparo o llamado de modo paginador (pager mode).
SUSCRIBE	Solicita información de presencia, la cual será enviada al solicitante.
NOTIFY	Transporta la información de presencia.

Para las sesiones de M.I. normales, es decir conversaciones, SIMPLE inicia utilizando INVITE y luego pasa a utilizar el protocolo de transporte llamado Message Session Relay Protocol (MSRP).

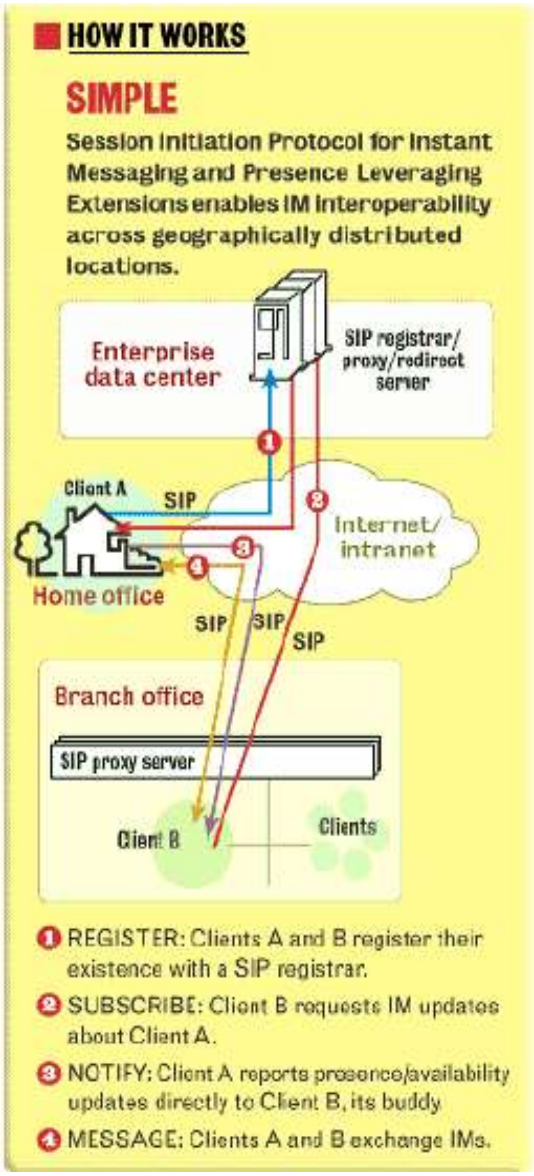
SIP (así también como SIMPLE) es orientado end-to-end, lo que implica que toda la lógica es manejada por los dispositivos cliente (salvo el ruteo de las señales), no hay u único punto de falla y las redes pueden escalar de buena manera. el precio a pagar es mucho más overhead de los mensajes (señales) y mucho más proceso del lado de los clientes, que a su vez tienen una parte de cliente servidor. La parte cliente (UAC - User Agent Client) es la que manda requests y recibe

responses de otros servidores y la parte de servidor (UAS User Agent Server) recibe requests externos y envía responses.

Como la comunicación entre 2 usuarios es de estilo peer-to-peer, también agrega un poco más de complejidad si se quiere implementar algún requerimiento específico , por ejemplo : registro de conversaciones entre dos usuarios para auditoría.

SIP está basado en el protocolo HTTP , el cual hereda el formato de cabezales de los mensajes del RFC822. SIP también hereda la codificación de cabezales de mensajes del RFC822

Las entidades SIP se identifican utilizando SIP URI's (Uniform Resourse Identifiers) la forma de una SIP URI es sip:usuario@dominio, en un estilo de dirección de e-mail .



Envío de Mensajes

Los clientes de M.I. envían el tráfico real a sus pares directamente o a través de Servidores Proxy SIP o Servidores de Redireccionamiento SIP.

Los servidores proxy SIP reenvían los pedidos entre elementos del sistema SIP tales como teléfonos SIP, mientras que los servidores de redireccionamiento son usados para comunicarle a los clientes acerca de los movimientos de los participantes.

Un cliente M.I. usa MIME para enviar pedidos de multimedia. M.I. multiparte y salas de chat son soportados, ya que SIP fue diseñado para rutear señales tanto a más de un punto final como a uno solo.

La relación de M.I./presencia con SIP es análoga a la de SMS con la telefonía celular. SMS utiliza piggybacking sobre la red de tecnología celular y M.I./presencia lo hace sobre SIP, el cuál es el la forma de señalización telefónica de Internet con mas momentum en la actualidad.

Al utilizar SIMPLE, M.I./presencia automáticamente gana los beneficios de SIP, los cuales combinan multimedia, funcionalidades de multiparte y groupware, así como la ventaja de soportar estas mismas funcionalidades para usuarios "móviles" (PDAs, celulares, etc.)

Figura 1 : Todo el contenido con copyright 1995-2003 Network World, Inc. http://www.nwfusion.com.
Este texto corresponde a una traducción realizada de la sección Spreading Messages de [8].

3.3.4 XMPP y Jabber

En enero del año 1998 Jeremie Miller, hace de público conocimiento la especificación un protocolo de mensajería instantánea abierto basado en XML , bautizado como Jabber (que significa algo así como "Parloteo"). El origen de Jabber se da gracias a que Miller tenía que utilizar varios clientes de M.I. para mantener todos sus contactos lo cual le resultaba problemático.

La solución obvia podría ser construir un cliente único que soportara todos los protocolos de sistemas de M.I. , pero Miller encontró los siguientes problemas:

- La naturaleza de los protocolos propietarios hacían difícil la implementación para un soporte completo en el cliente y lo hacia demasiado complicado.
- Cada vez que se hiciera cambio de protocolos (algo que estaba fuera de su control) o se implementara un nuevo protocolo, el cliente debería ser modificado y reinstalado , una tarea no muy práctica en sitios con muchos usuarios
- Se suma que la programación de interfaces gráficas no es el fuerte de todos los programadores, como el caso de Miller, entonces prefirió concentrar esfuerzos en problemas más específicos y dejar a otros la construcción de Interfaces Gráficas

Miller entonces decidió crear una solución con las siguientes características:

- Debía tener su propio protocolo interno basado en XML

El protocolo debería de ser:

- Simple de entender y de implementar
- Fácil de extender
- Abierto.

- La complejidad para interactuar con protocolos propietarios debía de resolverse a nivel de servidor en forma de plug-in, cada módulo (o plug-in) sería un transporte específico para un protocolo externo específico
- Todos los clientes tendrían que implementar solo el sencillo protocolo interno, el resto sería implementado en el servidor

Llamó a esta solución Jabber. Una parte fundamental de la arquitectura propuesta por Miller fue que el servicio sería descentralizado, tal vez porque no se consideró a priori para grandes organizaciones, donde cualquiera podría tener su propio servidor con su propio dominio, muy similar a servidores SMTP, esta desisión de Miller (tal vez casual), tiene un efecto colateral muy beneficioso en cuanto a la capacidad de implementar la solución en cualquier tipo de red de una forma tan distribuida como se quiera (una empresa puede tener un servidor con domino propio por sección, por departamento, por sucursal)

En agosto de 1999 Miller pide a la comunidad de Jabber en Internet soporte para que Jabber sea contemplado por la IETF. Una año y medio después en Junio del 2000, Miller y otros miembros del proyecto Jabber enviaron un 'Internet draft' al grupo de trabajo de la IETF encargado de protocolos de Mensajería instantánea y Presencia (IMPP – Instant Messaging and Presence Protocol), dicho borrador (draft) documentaba el protocolo Jabber, el primer intento fue infructuoso ya que el draft expiró sin ninguna contribución por parte del grupo de trabajo.

A principios de 2002, la comunidad de Jabber decidió una vez más enviar el protocolo Jabber a la IETF y esta vez hubo éxito pues la IETF formó un grupo de trabajo llamado XMPP (eXtensible Messaging and Presence Protocol).

El grupo de trabajo XMPP produjo cuatro Internet drafts, que recientemente (Octubre de 2004) pasaron a ser RFC's (cuya categoría actual es la de estándar Track)

Los cuatro documentos corresponden a :

- **Extensible Messaging and Presence Protocol (XMPP): Core**

Antes *draft-ietf-xmpp-core-22*, actualmente RFC3920, especifica el núcleo del protocolo, a saber, como el protocolo utiliza flujos de datos en XML para intercambio de información estructurada en tiempo real entre dos puntos dentro de una red informática.

- **Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence**

Antes *draft-ietf-xmpp-im-21,* actualmente RFC3921, especifica como extender el núcleo del protocolo (XMMP:core) para proveer funcionalidad básica de mensajería instantánea y de información de presencia de acuerdo con el RFC2779 (Instant Messaging / Presence Protocols Requirements)

- **Extensible Messaging and Presence Protocol (XMPP): Mapping the Extensible Messaging and Presence Protocol (XMPP) to Common Presence and Instant Messaging (CPIM)**

Antes *draft-ietf-xmpp-cpim-04* , actualmente RFC3922, especifica el mapeo entre XMPP y la especificación CPIM (Common Profile Instant Messaging), principalmente para poder implementar gateways intermedios para comunicación entre servicios XMPP y servicios no-XMPP

- **Extensible Messaging and Presence Protocol (XMPP): End-to-End Signing and Object Encryption for the Extensible Messaging and Presence Protocol**

Antes *draft-ietf-xmpp-e2e-07*, actualmente RFC3923, define los métodos para habilitar a un usuario a firmar y/o encriptar un mensaje enviado a un recipiente específico.

En el Apéndice 8.11 se presenta un análisis mas detallado de dicho protocolo.

3.3.5 Conclusión

Luego de el análisis realizado en el apéndice 8.11 acerca de el protocolo XMPP, encontramos que el mismo es muy elegante, sencillo, entendible, potente y extensible. XMPP cuenta con todas las características para volverse estándar de la IETF y si bien la mayoría de implementaciones de servidores de mensajería están basados en el protocolo Jabber, es de esperar que en siguientes versiones de los productos todos tienda a ser compatibles con la versión 1.0 de XML propuesta.

La arquitectura cliente-servidor propuesta por XMPP también es muy elegante y sencilla, pudiendo obtener sistemas distribuidos inclusive con componentes de los servidores distribuidos, escalabilidad horizontal, clusters de servidores, implementación sencilla para failover y control total sobre el flujo de la información.

Notamos que XMPP brinda todas las herramientas y especificaciones para extender el protocolo y para poder lograr interoperabilidad con (y entre) aplicaciones empresariales como correo, groupware, sistemas de ERP, CRM, etc.

4 SELECCIÓN DEL PROTOCOLO

4.1 IRC

Si bien IRC cuenta con muchos años en el área de chat en Internet, entendemos que el protocolo toma como funcionalidad fundamental los canales de chat multiusuarios y luego brinda la posibilidad de una conversación entre dos personas, el otro problema que encontramos en IRC es el manejo de presencia , no fue construido para los requerimientos y conceptos de presencia de mensajería moderna. Tampoco tiene el concepto de lista de contactos, aunque se puede simular una funcionalidad parecida teniendo varios canales de IRC y en cada uno de ellos habilitar los contactos deseados, pero no es una funcionalidad de lista de contactos.

Históricamente IRC siempre fue utilizado por personas relacionadas con la informática, las personas no relacionadas al área en general no conocen IRC y el concepto de mensajería que tienen es el de MS Messenger o Yahoo Messenger, lo cual, en una implantación corporativa, el usuario final puede sentirse menos cómodo para su uso.

Debemos destacar también que IRC puede ser beneficioso para determinados entornos de trabajo o entornos académicos, pudiendo ser muy efectivo por ejemplo para tener canales (chatrooms) de consulta dentro de un instituto, se podría pensar en un newsgroup en tiempo real pero si arbitraje ni control de mensajes posteados.

4.2 SIMPLE Y XMPP

Dados que estos protocolos apuntan a ser los estándares de la IETF para servicios de Mensajería Instantánea, veremos las principales diferencias a nivel del protocolo para la selección de uno de ellos para estudiar una implementación real del mismo, también se tomará en cuenta la cantidad de proyectos Open/free de servidores que implementan dichos protocolos.

4.2.1 Diferencias entre SIMPLE y XMPP

Las diferencias más importantes que encontramos entre estos dos protocolos son:

- SIMPLE propone una arquitectura bastante compleja para cliente-servidor ya que está construida sobre la arquitectura SIP. Existen varios servidores SIP con una tarea designada específica lo cual lleva a que se deben de encontrar implementaciones de todos estos servidores. SIMPLE no implementa protocolo para comunicaciones servidor-servidor.

  XMPP propone una implementación cliente-servidor mucho más elegante y sencilla en la cual especifica ambas interfases cliente-servidor y servidor-servidor, esta implementación de cliente–servidor facilita la implementación de soluciones para requerimientos de auditoría y log de conversaciones.

- Vimos que la comunicación entre dos clientes en SIMPLE es en un estilo peer-to-peer, lo cual hace muy difícil el control de tráfico. Los creadores de SIP alegan que una sesión peer-to-peer entre dos clientes esto no es tan malo ya que se pueden tener muchas conversaciones sin sobrecargar los servidores ya que sólo la iniciación de las conversaciones pasan a través de los servidores

- De acuerdo a estudios hechos en Jabber Inc. (desarrolladores del servidor Jabber XCP, el servidor más completo al día de hoy basado en XMPP), los requerimientos de ancho de banda entre los protocolos pueden diferir significativamente. Un mensaje en SIMPLE puede ser de más del doble del tamaño que en XMPP y en el caso de mensajes de presencia la diferencia es mucho mas grande, el estudio dice que la diferencia es 1 a 10 a favor de XMPP. Sin embargo es obvio que estos estudios pueden resultar tendenciosos, en la práctica, la relación 1 a 10 puede variar de acuerdo a como los clientes y servidores XMPP implementen la información opcional de las Stanzas.

- SIMPLE estrictamente no contempla en su especificación funcionalidades como lista de contactos o capacidades de chatrooms (aunque se pueda implementar enviando de uno a varios clientes). Esto ha obligado a empresas como Microsoft e IBM (que han adoptado a SIMPLE) a desarrollar extensiones propietarias.

- Otra diferencia y problema potencial de SIMPLE es heredado de SIP ya que utiliza TCP y UDP en el sentido de que TCP incluye control de congestión pero UDP no. Esto abre la posibilidad de pérdida de Datagramas (UDP) en caso de congestión en la red.

La extensibilidad de XMPP y su estructuración basada en XML, hace de este protocolo una plataforma de comunicación en tiempo real no solo para utilizarse en tecnologías de Mensajería Instantánea sino también para comunicación en tiempo real entre aplicaciones, esto en una empresa puede utilizarse tanto para aplicaciones internas como para ofrecer servicios a otras empresas. Bell South por ejemplo, utiliza XMPP para la construcción de aplicaciones con requerimientos de interoperabilidad en tiempo real para call-centers.

4.3 IMPLEMENTACIONES DE CLIENTES Y SERVIDORES OPEN/FREE

4.3.1 Servidores

Encontramos que existen varias implementaciones de servidores basadas en Jabber y XMPP, cada proyecto actualiza de manera muy activa, existen sitios de desarrollo de plugins, herramientas de administración, auditoría, registro de conversaciones, clientes web.

Encontramos que existen también muchas implementaciones de servidores XMPP/Jabber tanto para servicios públicos como en empresas (en apéndice 8.5 presentamos la lista de servidores públicos y algunas implantaciones empresariales que vale la pena mencionar son el caso de Boeing y BellSouth), no así en el caso de servicios de mensajería basados en SIMPLE.

4.3.2 Clientes

También encontramos que existen muchos clientes que soportan Jabber en plataformas Microsoft y tipo-UNIX, algunos muy completos como el caso de Gaim. También encontramos clientes web para Jabber

4.4 CONCLUSIÓN

Habiendo relevado y analizado los distintos protocolos abiertos para mensajería, optamos que la implantación como caso de estudio aplicado, sería una solución de Mensajería Instantánea que estuviera basada en el protocolo Jabber/XMPP.

Cabe mencionar también que muchas empresas importantes han elegido dicha tecnología para implantar Servicios de Mensajería y Presencia , siendo uno de los más notorios el caso de la empresa Boeing, la cual utiliza el producto Jabber XCP (basado en XMPP) como sistema de mensajería y presencia embebido en el sistema táctico militar S.O.S.C.O.E. (Systems Common Operating Environment). Dicho sistema forma parte del proyecto multi-billonario de la armada de los Estados Unidos, el sistema F.C.S. (Future Combat Systems)

Así también fue ya mencionado el caso de BellSouth extendiendo el protocolo para requerimientos de aplicaciones específicos.

En el siguiente punto analizaremos las soluciones existentes de servidores, cuales son las más completas, seleccionaremos una de ellas y se describirá la implantación real dentro de Facultad de Ingeniería.

5 SELECCIÓN DE SERVIDOR

La selección del servidor a estudiar y para posterior implantación de servicio fue hecha en base a las siguientes consideraciones:

- Que tanto soporte del protocolo Jabber/XMPP que brinda cada producto
- Opciones de autenticación: debe ser seguro (soportar encriptación de mensajes que evite que terceros puedan leer los mensajes, y autenticación que permita restringir entre quienes se puede intercambiar mensajes; su uso no debe comprometer la seguridad de la empresa mediante el empleo de puertas traseras o caballos de Troya)
- Opciones para almacenamiento de datos
- Debe ser popular para asegurar el futuro activo del producto
- Debe ser interoperable (evitar aislamiento al poder intercambiar mensajes con usuarios de otros servicios de mensajería)
- Documentación de la que se dispone
- Capacidades adicionales para poder abarcar la mayor cantidad de requerimientos para brindar servicios empresariales (vistos anteriormente), ya sea de forma interna o mediante componentes adicionales (que también deben ser Open/Free).

Para poder tener una idea de la diferencia entre soluciones Open/Free y Comerciales , se incluirá también el análisis de 2 de las más completas Soluciones basadas en XMPP/Jabber.

Los servidores a relevar son los más completos y populares:

	Antepo OPN	**Jabber XCP**	**Jive-Messenger**	**Jabberd**	**Ejabberd**
Desarrollador	Antepo Inc	Jabber Inc	Jive Software	Proyecto jabberd	Proyecto ejabberd
URL	www.antepo.com	www.jabber.com	www.jivesoftware.org	jabberd.jabberstudio.org	ejabberd.jabberstudio.org
Licencia	Comercial	Comercial	GPL (antes comercial)	GPL	GPL
Plataformas	AIX,HP-UX, Linux, Solaris, Windows	Linux,Solaris,Windows	AIX,HP-UX, Linux,MacOS X, Solaris, Windows	AIX,*BSD,HP-UX, Linux,MacOS X,Solaris,Windows	AIX,*BSD,HP-UX, Linux,Solaris,Windows
Versión	4.0	4.0	2.0.0	2.0s6	0.7.5
Lenguaje	Java	C	Java	C	Erlang

Tabla comparativa de características relevar:

	Antepo OPN	Jabber XCP	Jive-Messenger	Jabberd	Ejabberd
Soporte de protocolo					
Seguridad Cliente-Servidor					
TLS	Si	Si	No	Si	Si
SASL	No	Si	No	Si	Si
SSL (Viejo)	Si	Si	Si	Si	Si
Seguridad Servidor-Servidor					
TLS	Si	Si	No	Si	No
SASL	Si	Si	No	No	No
Dialback	Si	Si	No	Si	Si
Internacionalización					
Para dominio	Si	Si	Si	Si	Si
Para usuario	Si	Si	Si	Si	Si
Para Recurso	Si	Si	Si	Si	Si
xml:lang	Si	Si	No	No	Si
Funciones XMPP Básicas					
Asignación de recursos	Si	Si	Si	Si	Si
Sesiones de MI	Si	Si	Si	Si	Si
Reglas de privacidad	Si	Si	No	Si	Si
Errores de XMPP	Si	Si	Si	Si	Si
Contactos y Presencia					
Manejo de Rosters	Si	Si	Si	Si	Si
Subscripciones	Si	Si	Si	Si	Si
Ultima actividad	Si	Si	No	Si	Si
Manejo de mensajes					
Entrega demorada	Si	Si	No	Si	Si
Expiración de mensajes	Si	No	No	Si	Si
Recuperación flexible de mensajes offline	No	Si	No	No	No
Protocolos para "descubrimiento"					
Descubrimiento de servicios	Si	Si	Si	Si	Si

Soporte para Cliente					
Autenticación no-SASL	Si	Si	Si	Si	Si
Registración con otros servidores y servicios	Si	Si	Si	Si	Si
Almacenamiento XML privado	Si	Si	Si	Si	Si
Almacenamiento de v-cards	Si	Si	Si	Si	Si
Información e interacción con el Servidor					
Entidad v-card	Si	Si	Si	Si	Si
Entidad Tiempo	Si	Si	Si	Si	Si
Versionado de software	Si	Si	Si	Si	Si
Comandos ad-hoc	No	No	Si	No	No
Métodos alternativos de conexión					
Protocolo para componentes externos	Si	Si	No	Si	Si
HTTP Polling	Si	Si	No	No	Si
Opciones de autenticación					
Fuentes para autenticación					
LDAP	Si	Si	Si	Si	Si
Certificados	Si	Si	Si	No	No
Radius	Si	No	Si	Si	No
PAM	Si	Si	Si	Si	No
Dominio NT	Si	No	Si	No	No
Standalone	Si	Si	Si	Si	Si
Mecanismos SASL					
PLAIN	Si	Si	No	Si	Si
DIGEST-MD5	Si	Si	No	Si	Si
Kerberos_V4	Si	Si	No	No	No
ANONYMOUS	No	Si	No	Si	No
Opciones para almacenamiento de datos					
File-system	Si	Si	Si	Si	Si
LDAP	Si	Si	Si	Si	No
BerkleyDB	No	Si	Si	Si	No
PostgreSQL	Si	Si	Si	Si	No
MySQL	Si	No	Si	Si	No

MS SQL Server	Si	No	Si	No	No
Oracle	Si	Si	Si	Si	No
MS Active Directory	Si	Si	Si	No	No
Otros	No	No	Si (DB2)	Si	Si (Mnesias – BD impl en Erlang)
Documentación					
Archivo README	Si	Si	Si	Si	Si
guía de usuario	Si	Si	Si	Si	Si
Documentación de APIs	Si	Si	Si	Si	Si
Capacidades adicionales					
Almacenamiento de Mensaje Offline	Si	Si	Si	Si	Si
Archivado de Mensajes (log y auditoría)	Si	Si	Si	Si (componente externo)	Si (componente externo)
Herramientas de administración	Si	Si	Si	Si (componente externo)	Si
Estadísticas del servidor	Si	Si	Si	No	Si

Paralelamente se realizaron instalaciones y pruebas con Jive-XMPP, ejabberd y Jabberd.

(En el caso de Jive Messenger (o Jive XMPP), las pruebas realizadas fueron a modo de curiosidad técnica ya que el producto pasó a se Open-Source en Noviembre de 2004 y ya se había comenzado con la implantación de un servidor XMPP en facultad.)

5.1 JIVE MESSENGER

JIVE Messenger ofrece una solución totalmente integrada de un servidor basado en XMPP. Jive Messenger está hecho en Java y tiene como prerrequisito el JDK1.5, la instalación es muy sencilla y la configuración inicial (nombre de servidor, de dominio, clave de usuario administrador, tipo de conexión JDBC, etc) es muy intuitiva, igualmente las tareas de administración, Jive utiliza un aplicación web basada en servlets y se puede acceder a la misma vía Web browser.

Para almacenar datos de la aplicación Jive puede utilizar una base de datos embebida propia (H.S.Q.L.D.B. , sistema de Base de Datos relacional enteramente escrita en Java que soporta en buena forma ANSI-SQL92 y mejoras de las especificaciones de SQL 99 y SQL 2003) , lo cual lo independiza de buena forma, pero recomiendan que para implantaciones grandes para soportar muchos usuarios se utilice un sistema de Base de Datos.

El producto brinda scripts para la creación su esquema para las bases de Datos soportadas, la conexión del servidor hacia la BD se realiza mediante JDBC. Uno de los puntos fuertes de Jive es que soporta una gran cantidad de sistemas de Bases de Datos, tanto comerciales como Open/Free (MySQL, PostgreSQL, Oracle, MS SQL Server, DB2).

Tiene integradas funciones de auditoría, loggin, estado de usuario y de creación de Chatrooms con opciones de seguridad para los mismos. También tiene soporte para Service Discovery y LDAP.

Encontramos que el producto es muy bueno para una implantación rápida en un ambiente empresarial de porte mediano o pequeño.

Una de las principales desventajas que encontramos en el producto es que aún no implementa los módulos para conexiones server-to-server y requiere la infraestructura necesaria para correr Java 5 (JDK1.5), lo cual puede ser un costo adicional en caso de no poseer el hardware adecuados.

Si bien sus desarrolladores aseguran que el servidor tiene una robustez y escalabilidad probadas, encontramos que en las pruebas de carga, en algunos casos, el proceso del servidor finalizaba (finalizaba toda la JVM) quedando el servicio de Mensajería no disponible.

5.2 EJABBERD

Ejabberd es un servidor multiplaforma desarrollado en Erlang[1] y de código abierto. Puede ejecutarse en clusters proveyendo de esta manera tolerancia a fallas, ya que la información de los nodos del cluster se replica en cada uno. Si uno cae, los otros siguen trabajando normalmente. La instalación es sencilla, ya que la distribución de Erlang provee de todos los componentes que necesitará eJabberd. Soporta Ipv6. Implementa casi de forma completa el XMPP y varias de las extensiones J.E.P (Jabber Extension Protocol).

De todas formas, el número de sitios que lo utilizan como solución no es muy elevado respecto a Jabberd, y tampoco las listas de noticias crecen tanto como las de este último.

Podríamos decir entonces que Jabberd cuenta con una comunidad de usuarios más amplia lo que normalmente se traduce en mejor soporte.

En cuanto a la parte de desarrollo, creemos que el estar implementado en Erlang es lo que lo pone en desventaja con respecto a servidores como Jive y Jabberd (hechos en java y C/C++ respectivamente) ya que está limitado el desarrollo a una comunidad más reducida de personas que conozcan el lenguaje.

Por otro lado, la configuración de eJabberd no parece tan sencilla como los archivos xml de Jabberd . Pero posee una interfaz gráfica que permite configurarlo, la cual es accesible vía web a la dirección http://127.0.0.1:5280/admin donde la dirección IP localhost puede sustituirse por la dirección o nombre del host donde corre el servicio, para acceso remoto.

Esta interfaz permitirá administrar las ACL (Access Control List), crear o dar de bajas a usuarios y ver información estadística del servidor, por otro lado no permite la configuración de Salas las cuales deben de ser creadas y configuradas desde los clientes. Recomiendan los desarrolladores que se utilice PSI o Tkabber (clientes para protocolo Jabber) para estas tareas.

[1]Erlang/OTP es un ambiente de desarrollo para construir aplicaciones distribuidas de tiempo real. Tiene dos versiones: una registrada con soporte y una de código abierto sin soporte. Más información se puede obtener en http://www.erlang.se/

5.3 JABBERD

Es la implementación original del servidor para el protocolo Jabber (fue desarrollado por la comunidad Jabber, podríamos decir que es el "original").

Dado que Jeremie Miller comenzó la primera implementación en 1998, tenemos que el producto ya tiene 7 años en el mercado y esta en continuo desarrollo logrando cada vez más features y más soporte a la especificación XMPP, la versión actual es la 2.0s6.

Es un servidor ampliamente usado en el mundo, existen varias implantaciones de servidores Jabberd públicos que ofrecen servicios gratuitos de M.I. (ver apéndice A5). Algo a destacar es que la mayoría de dichos servidores están aún en la versión 1.4.x y que también la mayoría corre bajo sistemas GNU/Linux.

Tiene una comunidad muy activa de administradores y usuarios, esto queda demostrado por la gran cantidad de mensajes que llegan a diario a la lista de administración al servicio de noticias (news) gmane.org, este hecho a sido de gran utilidad para nosotros a la hora de buscar referencias para la instalación, configuración y resolución de problemas.

La arquitectura de Jabberd es sencilla (cliente-servidor) exactamente igual a la vista anteriormente propuesta por XMPP.

5.4 ARQUITECTURA DE JABBERD2

Jabberd2 distribuye sus servicios a través de 5 componentes desarrollados en C, los cuales se comunican vía TCP/IP soportando encriptación en el flujo interno y externo de datos mediante el uso de OpenSSL, cada componente es configurable mediante archivos XML:

- **`Router`**
  Es el componente más importante de servidor, acepta conexiones de los demás componentes y pasa paquetes XML (los paquetes pueden ser mensajes de usuario o paquetes del servidor mismo) entre ellos

- **`S2S`**
  Maneja la comunicación con servidores externos

- **`Resolver`**
  Componente utilizado por S2S, resuelve hostnames para el S2S como parte de la autenticación vía dialback

- **`SM`**
  Implementa la mayor parte de funcionalidades de la parte de mensajería instantánea en sí:
  - Pasa mensajes
  - Presencia
  - Listas de personas
  - Subscripciones

El SM se conecta al Software de datos de la aplicación (BD) para obtener los datos de persistencia. SM también maneja las extensiones de Discovery (dichas extensiones son para descubrimientos de servicios que brinda el servidor como por ejemplo descubrimiento Gateways disponibles) y listas privadas

- **`C2S`**
  maneja la comunicación con clientes Jabber2
  - Se conecta con los clientes
  - pasa paquetes para el SM
  - Autentica Clientes
  - Registra usuarios
  - Dispara actividades (triggers activity) con el SM

El componente C2S se conecta con el software de datos de Autenticación (DB, LDAP) , para autenticar y registrar usuarios

Jabber2 también requiere de componentes externos:

- Application Data Store: base de datos para almacenar datos del servidor y de los usuarios como ser: lista de contactos, v-cards, mensaje del día, información de creación de cuentas, logout de usuarios, encolamiento de mensajes, mensajes offline, información de licencia vacacional del usuario.

- Authentication Data Store: base de datos o LDAP para mantener información de registro y autenticación de usuarios

- Gateway(s) de M.I. externos: actualmente existen gateways para MSN, AIM,Yahoo, ICQ, IRC, SMS, SMTP y otros menos conocidos

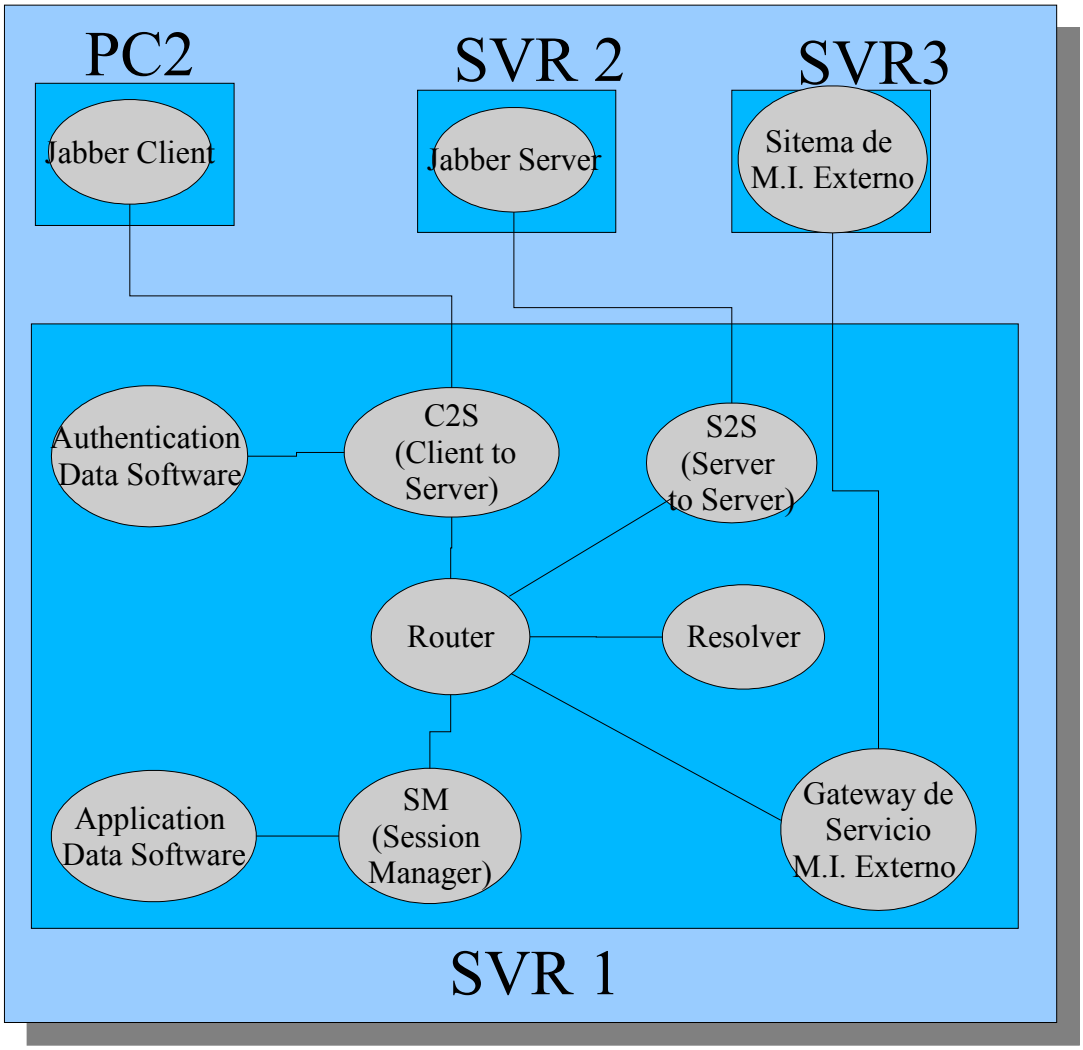


Figura 2 : Diagrama de componentes de Jabberd2

Cabe destacar que la implementación basada en componentes que se comunican vía TCP/IP permite una solución con una granularidad de distribución excepcional. No solo se puede lograr escalabilidad vertical en un solo servidor Jabberd (esto es mejorando el HD del equipo donde corre el servidor), sino que se permite una escalabilidad horizontal para el mismo servidor con el mismo dominio, esto es corriendo cada componente en Servidores físicos distintos. Gracias a esto se puede incrementar la seguridad en buena manera, por ejemplo el componentes S2S y los Gateways pueden colocarse en servidores separados dentro de la DMZ entre dos firewalls, y los componentes para mensajería local dentro de la red local.

Gracias a la arquitectura que propone XMPP y la implementación descentralizada de Jabberd. Se puede obtener gran flexibilidad al momento de implementación de solución de mensajería de acuerdo a el caso en que nos afrontemos, prácticamente se pueden distribuir los servidores y componentes de los mismos de la manera que más convenga.

5.5 PAQUETES EXTERNOS PARA SOLUCIÓN A DEMÁS REQUERIMIENTOS EMPRESARIALES

5.5.1 Configuración y Herramientas de Administración de Jabberd

Como mencionamos anteriormente, Jabberd utiliza una serie de archivos basados en XML para la configuración de cada componente (un archivo por componente) y los servicios que brindan, estos archivos, son auto explicados y de fácil entendimiento haciendo muy intuitiva la comprensión acerca de que se debe modificar para lograr el cambio deseado en el servidor.

A la fecha de hoy Jabberd en su versión 2.0 carece de herramientas administrativas basadas en interfaces gráficas o scripts de administración. (para la versión 2.0 sólo se encontraron frontends php para creación de usuarios , no así su versión previa 1.4, la cual cuenta con scripts de administración más maduros que abarcan más tareas sobre el servidor).

El hecho de que dichos archivos de configuración sean basados en XML y estén especificados y descriptos de una forma muy completa, hace relativamente sencilla la implementación de herramientas administrativas gráficas (probablemente web) por ejemplo basadas en servlets J2EE utilizando parsers de XML como por ejemplo Xerces del proyecto Jakarta o también se puede utilizar PHP o perl (en el caso de este último, actualmente permite el manejo de XML en forma de objetos, lo cual hace la programación y manipulación de los archivos de una manera muy fácil y conveniente).

De la misma forma , dado que el esquema utilizado en la base de Datos es muy sencillo, la implementación de herramientas gráficas para administración de usuarios, manejo de listas de contactos, etc., también se hace relativamente sencilla.

5.5.1.1 Jabber Registration Tool

La Herramienta de Registración para Jabber (o JRT por su sigla en Inglés) es una herramienta basada en web para la alta y baja de usuarios en servidores Jabber; así como para la modificación de las claves de acceso de los mismos. Todo lo que se necesita es un web server con soporte para PHP 4.0.1 o posterior y XML Puede obtenerse de jrt.jabberstudio.org. Está licenciado bajo los términos de la licencia GPL.

5.5.2 Auditoría y Log de conversaciones :

Bajo la premisa de que "si una persona sabe que sus actividades son públicamente visibles, dicha persona limitará sus actividades a las que el público deba ver", la auditoría y log de conversaciones son fundamentales dentro de un entorno corporativo para evitar abusos de la tecnología para fines personales, así como también evitar potenciales fugas de información hacia fuera de la empresa o dentro de la misma. Según el estudio presentado en la CSCW mencionado anteriormente, en una implantación de servicio de Mensajería con auditoría y log de conversaciones se encontró que es muy raro que los empleados utilicen este medio para chat sobre cuestiones personales, solo un 13 % de las conversaciones laborales contenían algún asunto personal y un 6.4 % de las conversaciones contenían solo temas personales.

Dentro de los paquetes de Software de logging de conversaciones y auditoría Open/Free para servidores basados en XMPP, el más completo que encontramos fue Bandersnatch

Bandersnatch se conecta al componente router de jabberd2 y captura los paquetes XML que pasan a través del servidor Jabberd y los procesa para luego insertarlos en una base de datos de forma que luego sea sencillo acceder a dichos datos para representarlos de forma estadística, las

vistas de los datos se realizan mediante un frontend Web hecho en php. Dicho frontend de acceso web permite acceder públicamente a datos estadísticos de uso del servidor por usuario como ser cantidad de mensajes internos o hacia gateways externos. Para poder visualizar mensajes de usuarios , bandersnatch implementa un usuario administrador que debe acceder mediante autenticación.

5.5.2.1 Información ofrecida por Bandersnatch

Consulta al JID de Bandersnatch

Si se le es enviado un mensaje al JID del componente, este responderá con las estadística de uso del día de el usuario que realizó el pedido. En caso de que el usuario pertenezca a la lista de Administradores definida en Bandersnatch, este enviará información acerca de los 20 usuarios locales y remotos que más han estado utilizando el servicio en el día

Consultas al Frontend

El frontend ofrece información estadística a nivel de servidor y de usuario, registra el total de mensajes locales y remotos por día, total de mensajes hacia gateways externos (por cada gateway) y una lista de usuarios, si se cliquea en uno de los usuarios , se muestran las estadísticas correspondientes a dicho usuario. En caso de que se quiera acceder al registro de conversaciones de un usuario, se debe de ingresar al frontend como Administrador.

5.5.2.2 Configuración del componente

La configuración de Bandersnatch consta de un archivo XML para el producto en si y de un archivo php para el frontend. El archivo de configuración es bastante sencillo de entender y contiene varias opciones logrando una flexibilidad bastante buena del componente.

La configuración del componente consta de:

- **Opciones del servidor :**
  - hostname: nombre del servidor jabber
  - port : puerto del servidor por el cual se conecta el componente
  - secret: palabra clave con la cual el componente autentica contra el servidor
  - connectiontype: tipo de conexión que el componente establecerá con el servidor (por defecto es TCP/IP)

- **Opciones del componente:**
  - name: es el JID del componente, además este usuario es el que se autentica con el router para proveer el servicio.

- **MySQL:**
  - server: el nombre del host donde se encuentra el servidor

- dbname: el nombre de la Base de Datos que utiliza el componente
- user: usuario de la Base de Datos
- password: clave del usuario de la Base de Datos

- **Opciones de debug:**
  - level: nivel de registro de debug en caso de ser necesario
  - file : archivo de salida de debug

- **Opciones de logging:**
  - local_server: nombre completo del servidor local , con este valor Bandersnatch discierne acerca de cuales mensajes son remotos y cuales locales y para registrar presencia de los usuarios locales.
  - local_domains: lista de dominios considerados locales (para poder reflejarlos de ese modo en las estadísticas)
  - admin_jids: lista de JID los cuales Bandersnatch debe considerarlos como administradores. Normalmente un usuario puede enviar un mensaje a Bandersnatch para ver sus estadísticas del día. Los Administradores pueden recibir información más detallada como por ejemplo un ranking de usuarios y su utilización del chat.
  - ignore_jids: lista de JID's que Bandersnatch debe ignorar y no registrar información de log y auditoría para ellos.
  - Privacy: nivel de privacidad (de 0 a 3) implementa la granularidad de lo que se quiere registrar, desde no registrar estadísticas para usuarios remotos o no registrar conversaciones a una registración completa de mensajes y presencia
  - aggressive_presence: Bandersnatch por defecto solo registra presencia de tipo Offline y Online, para que pueda obtener estadísticas de presencia de forma completa (away, chat, DND) , debe suscribirse en la lista de contactos de un usuario, con presencia agresiva activada, cada vez que un usuario envíe un mensaje, Bandersnatch chequea si esta suscripto en sus contactos, en caso negativo, envía pedido de suscripción.

5.5.2.3 Bandersnatch y Base de Datos

El producto utiliza MySQL como sistema de Bases de Datos aunque en Internet hemos encontrado que algunas implantaciones se han hecho con PostgreSQL (con modificaciones del código fuente de bandersnatch).

La estructura en la Base de Datos es muy sencilla y solo consiste en 4 tablas :

- **message**

Todos los mensajes son registrados en esta tabla, se registra el texto del mensaje, el timestap del mismo, desde que usuario y hacia que usuario , la misma cuenta con índices sobre estos últimos tres campos.

- **presence**

registra los cambios de presencia de los usuarios

- **user**

Esta tabla mantiene información sobre el estado actual de los usuarios y a cuales están subscritos(para el uso de presencia agresiva)

- **auth**

Esta tabla solo es utilizada por el frontend PHP para determinar que usuarios pueden ingresar al frontend como administradores para poder leer el log de los mensajes.

5.5.2.4 Problemas encontrados

Los principales problemas encontrados con el producto fueron los siguientes:

Tanto la instalación cono la puesta en marcha del componente no es trivial, la documentación no esta del todo correcta. Los paquetes prerequisitos de Bandersnatch son muchos y todos no están documentados, mediante los errores en la puesta en marcha, se debieron deducir que paquetes prerrequisito faltaban además de los especificados en la documentación.

Se debió modificar el script de generación de la Base de Datos pues no estaba correctamente implementado (ver Apéndice 8.4.2)

Se tuvieron problemas para poder lograr conectar el componente hacia el router ya que la documentación no especifica claramente como hacer este paso y los ejemplos que otorgaba la misma no funcionaron en nuestras implantaciones.

Una vez que se pudo habilitar el servicio, vimos que el componente registraba los mensajes 2 veces en la base de datos, se debieron hacer modificaciones en los fuentes para corregir el problema. Si bien el problema fue corregido en base a una sugerencia que se encontró en la 'mailing list' de Bandersnatch, luego de modificar el fuente, se encontró un upgrade sobre el script de registro en el sitio de Bandersnatch que además arreglaba otro problema similar para los paquetes hacia Gateways de servicios de mensajería externos (ver Apéndices)

El componente a veces no se conecta contra el Router porque da fallo en la autenticación mediante SASL y se debe correr el script de inicio nuevamente.

(Todas las modificaciones y correcciones sobre los fuentes están en el Apéndice 8.4.2)

5.5.2.5 Conclusión

Si bien se tuvieron bastantes problemas para lograr un correcto comportamiento del componente y existen posibles mejoras al componente como por ejemplo el archivado y depuración automática de la Base de Datos, encontramos que Bandersnatch es capaz de generar un conjunto de estadísticas (útiles para tener una noción sobre la utilización del servicio de Mensajería y de loggeo de conversaciones) adecuado para un ambiente corporativo, se le agrega la facilidad con la cual se pueden acceder a estos datos vía frontend Web.

5.5.3 Salas de chatrooms (Multiconferencia):

Si bien se puede suponer que la ventaja única de la multiconferencia dentro de la M.I. empresarial podría ser para discusiones específicas donde se requieran varios integrantes, de todas formas es un requerimiento válido para la empresa disponer de dicho servicio.

Un servicio de multiconferencia podría verse como uno un news board pero en tiempo real y sin datos histórico (se puede implementar datos históricos extrayendo datos de los sistemas de log como el Bandersnatch). Cada usuario podría ingresar (si está autorizado) a distintos chatrooms creados para discutir temas específicos donde se requiere la opinión o desisión de más de dos personas. Las salas de multicinferencia pueden ser públicas o privadas (se acceden con clave).

La especificación de XMPP contempla los mensajes para multiconferencia, pero no especifica en si como debe de implementarse el manejo de las salas multiconferencias (cada servidor lo puede implementar como quiera), sin embargo la comunidad Jabber propone una extensión al XMMP/Jabber de se discute la implementación de un protocolo robusto para conferenciar en base a texto para utilizarse como guía para desarrolladores. Esta extensión es la JEP-0045: Multi-User Chat. (ver apéndice 8.2.1)

Este tipo de componente es uno de los que revisten mayor funcionalidad. Nos permitirá establecer salas de conferencia (salas de chat) las cuales podrán ser tanto públicas como privadas, moderadas o no, etc.

Jbberd2 no dispone nativamente de esta característica, para ello utilizamos el paquete **mu-conference**. Este componente es uno de los que revisten mayor funcionalidad. Nos permitirá establecer salas de conferencia (salas de chat) las cuales podrán ser tanto públicas como privadas, moderadas o no, etc.

5.5.3.1 Componente Mu-Conference (Multi User Conference)

Este componente fue diseñado originalmente para ejecutarse junto a las versiones 1.4x del servidor jabberd. Actualmente puede ejecutarse junto con las versiones 2.0sX del servidor pero necesita unas bibliotecas conocidas como Jabber Component Runtime o JCR para compilarse, hablaremos de esto más adelante.

Según se menciona en el archivo README de la distribución de Mu-Conference actualmente posee las siguientes características

- Compatibilidad hacia atrás con Groupchat 1.0
- Roles y Afiliación de usuarios
- Logging nativo de salas
- Configuración dinámica de salas
- Salas moderadas
- Salas protegidas por clave
- Salas no anónimas
- Salas exclusivas para miembros
- Exclusión de usuarios de las salas en modos Ban y Kick
- Salas persistentes reconfigurables

Los requisitos así como los procesos de instalación y configuración se muestran en el apéndice 8.2.2.

En cuanto a la administración, este componente viene con un script PERL que actúa a modo de "wizard" para la creación de salas persistentes, guiando al usuario en la creación de las mismas.

De toda maneras, por experiencia de los integrantes del grupo, la salas persistentes pueden ser generadas manualmente creando los archivos que definen su configuración y el archivo **`rooms.xml`**. Esto se traduce en una muy fácil administración ya que scripts o front-end web que faciliten estas tareas pueden ser fácilmente desarrollados.

5.5.4 Servicios para búsqueda de usuarios

Los componentes de tipo JUD (Jabber User Directory) proveen de un sistema que permite hallar fácilmente las JID de usuarios los cuales podrán ser buscados por datos personales tales como Nombre, Apellido, eMail, nick, etc. Se detalla en apéndice 8.3 el proceso de instalación de uno de los paquetes que brindan dicho servicio.

5.5.5 Clientes Web

Un requerimiento interesante para un servicio de Mensajería Instantánea en una empresa es el de acceso al servicio mediante una interfaz Web. El cliente se puede integrar en la página de la empresa o en un Portal.

Las ventajas más destacables que ofrece un cliente Web son tres :

- Incrementa la seguridad en caso de comunicación con Internet ya que no es necesario habilitar los puertos que utiliza Jabber para conexiones desde Internet, todo pasa por los puertos http.
- No obliga al usuario a instalar un cliente de XMPP, esto puede ser muy beneficioso y bien visto si la empresa tiene clientes con JID propio en su servicio de Mensajería, la empresa le está brindando servicios de forma personalizada y además no requiere la instalación de ningún paquete extra en el PC del cliente (generalmente los clientes tendrán ya clientes de servicios de mensajería públicos como MS Messenger pero es poco probable que tengan un cliente que soporte el protocolo XMPP)

5.5.5.1 Web Messenger

Web Messenger brinda un cliente Web hecho en PHP completamente funcional exceptuando la transferencia de archivos y las conexiones vía ssh, fue hasta la fecha el más completo y maduro que encontramos, utiliza una base de datos propia (mySQL) para llevar registro de usuarios que utilizan el servicio así como a que Gateways el usuario habilitado vía web. Las desventajas que encontramos son que las conexiones son solo por el puerto no encriptado del servidor, lo que implica un riesgo extra si el servidor HTTP que brinda el servicio no es el mismo que el servidor de Mensajería, y que a Setiembre de 2004 el producto se descontinuó como proyecto Open-Source y paso a ser Comercial.

Si bien la configuración del componente en si mismo es muy sencilla, lo que requiere una dificultad extra es la configuración del entorno donde corre el componente (se requieren conocimientos de administración de servidores HTTP y PHP entre otros). Al testearlo, encontramos que es un producto muy amigable y no requiere de conocimientos extras para ser utilizado por un usuario regular de M.I. acostumbrado a un cliente propio de Mensajería.

6 IMPLANTACIÓN DEL SERVICIO EN FING (REQUERIMIENTOS DE HARDWARE, DISPONIBILIDAD, PERFORMANCE Y PRUEBAS DE CARGA)

Se realizaron varias instalaciones y pruebas de Jabberd en distintos equipos con diferentes especificaciones de hardware:

- Pentium 120 Mhz con 32Mb de RAM
- Pentium Celerón de 266 Mhz con 64Mb de RAM (equipo de Facultad de Ingeniería)
- Athlon XP 2400 de 2 Ghz con 512 MB de RAM

Los requerimientos de disco no fueron relevantes para las pruebas de stress (si se debe de contemplar esto en caso de registro de conversaciones y presencia sobre una base de datos)

El servidor de mensajería respondió de buena manera en todos los casos inclusive en el equipo de menos recursos.

6.1 CASO DE ESTUDIO – IMPLANTACIÓN DE SERVICIO EN FACULTAD DE INGENIERÍA

Si bien se habían realizado varias pruebas en equipos aislados y 2 equipos conectados punto a punto, entre las cuales se probaron los componentes de auditoría y log de conversaciones, salas multi-conferencia, cliente web, pruebas de carga, etc., no se había podido simular una implantación de tipo empresarial con acceso desde varios clientes en una L.A.N. real.

El servicio de mensajería pudo implantarse a mediados de diciembre de 2004, el servicio fue implantado dentro de la red de Facultad con posibilidad de conversaciones dentro de la misma y hacia el exterior, pasando por 2 firewalls.

El servicio sigue disponible a la fecha actual (Marzo de 2005) y no sufrió ningún corte no planificado, solamente para instalaciones de upgrades del servidor Jabberd, brindando ya más de 100 días de servicio ininterrumpido. Se probaron funcionalidades de comunicación persona a persona desde dentro y fuera de Facultad (parte de la tutela del proyecto se realizó a través del servicio de M.I. con nuestro tutor estando en Francia), conferencia multiusuario, servicios de auditoría, pruebas de carga extremas con grandes cargas de tráfico de mensajes y operaciones de presencia sobre el servidor.

No se pudieron hacer pruebas de funcionalidades server-to-server ya que implicaba habilitar mas puertos en el firewall y en principio no lo consideramos como una funcionalidad básica.

6.2 PRUEBAS DE CARGA

Se presentará a continuación un resumen de los resultados de las pruebas de carga hechas sobre el equipo implantado en Facultad:

Bajo el supuesto de simular un caso extremo en una PyME se consideraron 50 usuarios activos, estableciendo conversaciones todos a la vez, añadiendo y eliminando contactos cada cierto tiempo, cambiando de estado de presencia. La herramienta de carga utilizada es el JabSimul. Con dicha herramienta se pueden configurar todas estas propiedades de las pruebas, dicha configuración es mediante archivo XML con los siguientes elementos:

- **`user_names_generator`** (generados de nombres de usuario)
    - **`range`**: rango de usuarios (Ej test01, test02,....test50)
    - **`server`**: Nombre del servidor
- **`connect`**: cada cuanto se conecta un usuario (en ms)
- **`add_roster`**: cada que frecuencia (en ms) se debe añadir un usuario a lista de contactos hasta que el valor 'max_roster_count' sea alcanzado
- **`del_roster`**: cada que frecuencia se borra un usuario de la lista de contactos
- **`send_message`**: Mensaje a enviar (se utiliza mensaje de 50 caracteres)
    - **`range`**: rango de usuarios a los cuales puede un usuario enviar
- **`change_status`** : cada cuanto tiempo (en ms) el usuario cambia el estado de presencia
- **`logout`** : cada cuanto se desconecta un usuario (en ms)
- **`kill_connection`**: cada cuando una conexión es abortada
- **`send_raw_bytes`**: envío de bytes aleatorios (ruido)

Para llegar a un valor coherente de largo de mensaje se realizaron conteos de palabras (promedio de 6 letras) de conversaciones en nuestros ambientes laborales y se tuvieron en cuenta datos del estudio hecho en el sistema de Mensajería de la empresa AT&T.

Según dicho estudio, en una conversación normal entre 2 usuarios que utilizan M.I. en una forma importante en un entorno laboral, los mensajes son de 12 palabras en promedio y cada mensaje es enviado cada 13 segundos.

Según nuestros conteos, los mensajes en promedio son de 7 u 8 palabras en promedio y se envían cada unos 20 segundos aprox..

Los valores para las pruebas del simulador fueron :

1 - Mensajes de largo 50 caracteres (8.3 palabras)

2 - En caso de la prueba de estabilidad, los mensajes eran enviados cada 7 segundos

En las pruebas de stress para medir retardo de mensajes y diferencia cuando estos son loggeados por Bandersnatch, los mensajes fueron enviados cada 15 segundos bajo el supuesto de que un usuario divida ese tiempo en leer un mensaje que le envía y contestar en 7 segundos otro mensaje de 50 caracteres.

6.2.1 Prueba de carga para estudiar estabilidad del Servidor:

Se realizo una prueba de carga de mas de 117 horas continuadas, y además se realizaron mediciones de trafico para estimar el uso de ancho de banda de red por usuario, las pruebas se realizaron sin registro de conversaciones con Bandersnatch.

Resultados:

```
prueba a las 63:54.39 hs de corrida

Conn stat:    conns: total: 51 estabilished: 51
              kills: total: 0 unexpected: 0
Messages:     tot.sent: 3346120          tot.rcvd: 3346119
              rcvd.offline: 0     rcvd.admin: 0
              rcvd.normal: 3346119 fwd: 1673060 avg.time: 56 [ms]
              diff check: 1 stability: 1
Roster:       tot.adds: 43301 avg.time: 87 [ms]
              tot.dels: 75353 avg.time: 98 [ms] glob_rost: 678
Presences:    tot.sent: 390934 tot.rcvd: 3185947
Packets:      created: 3935890 sent: 3935890
              canceled: 0 in queues: 0
```

```
luego de 117:21.50 hs, se corta la prueba para realizar mediciones

Conn stat:    conns: total: 51 estabilished: 51
              kills: total: 0 unexpected: 0
Messages:     tot.sent: 6144702   tot.rcvd: 6144702
              rcvd.offline: 0     rcvd.admin: 0
              rcvd.normal: 6144702 fwd: 3072351 avg.time: 55 [ms]
              diff check: 0 stability: -2
Roster:       tot.adds: 79370 avg.time: 90 [ms]
              tot.dels: 138459 avg.time: 94 [ms] glob_rost: 688
Presences:    tot.sent: 717944 tot.rcvd: 5859908
Packets:      created: 7227220 sent: 7227220
              canceled: 0 in queues: 0
```

Durante dichas las pruebas el servidor Jabberd utilizó un promedio de 30% de CPU y un 50 % de memoria.

El tráfico total de paquetes XML en los 7042 minutos fue de 4.3 GB aprox.

En promedio el ancho de banda que fue utilizado por los 50 usuarios fue de unos 85 kbps.

Cada usuario utilizo en promedio 1,7 kbps de ancho de banda.

Notamos que el servidor se comporta de excelente manera y que el consumo de recursos no es exagerado ni crece de forma significativa a lo largo de la prueba. También vemos que la utilización del recurso de Red no es de ninguna forma excesivo, una red de 10Mbps esta sobrada para soportar dicho servicio sin afectar la performance en la misma.

Aunque los tiempos de entrega de mensajes y operaciones del cliente son medidos en las siguientes pruebas simulando un entorno menos extremo de intercambio de mensajes, vemos en dicha prueba extrema que la entrega de cada mensaje demora en promedio 55 milisegundos.

6.2.2 Prueba de demora de tiempos de entrega de mensaje y de operaciones de usuarios (con y sin loggeo de presencia y conversaciones)

Se realizaron 4 pruebas de aproximadamente 8 minutos con características similares a la prueba de estabilidad pero con la diferencia que dos de ellas fueron con intervalos más razonables adecuados a una dinámica más real de un entorno empresarial.

Repetimos que la diferencia básica es que el intervalo de espera de envío de mensajes por parte de los usuarios, que era cada 7 segundos en las pruebas de estabilidad pasa a ser de 15 segundos.

Consideramos razonable el contar con una demora de 15 segundos ya que es normal leer un mensaje de 50 caracteres en 7 segundos y responder con otro mensaje de ese tamaño en otros 7. La idea de esta prueba es ver las demoras que puede introducir el loggeo de conversaciones por Bandersnatch y cuanto afecta en la performance del servidor Jabber y del Hardware la activación de dicho registro.

Pruebas con valores de jabsimul idénticos a pruebas de estabilidad:

Resultado Prueba 1 – Sin Bandersnatch

```
Conn stat:   conns:  total: 51    estabilished: 51
             kills:  total: 0     unexpected:   0
Messages:    tot.sent:     6546   tot.rcvd:   6545
             rcvd.offline: 0      rcvd.admin: 0
             rcvd.normal:  6545   fwd: 3273       avg.time:   55 [ms]
             diff check:   1      stability:  0
Roster:      tot.adds:     86     avg.time:   92 [ms]
             tot.dels:     148    avg.time:   93 [ms]     glob_rost: 693
Presences:   tot.sent:     815    tot.rcvd:   6183
Packets:     created:      8008       sent:   8008
             canceled:     0      in queues:   0
```

Resultado Prueba 2 – Con Bandersnatch

```
Conn stat:  conns:  total: 51    estabilished: 51
            kills:  total: 0     unexpected:   0
Messages:   tot.sent:     6578   tot.rcvd:   6514
            rcvd.offline: 0      rcvd.admin: 0
            rcvd.normal:  6514   fwd: 3257       avg.time:   72 [ms]
            diff check:   64     stability:  1
Roster:     tot.adds:     90     avg.time:   112 [ms]
            tot.dels:     146    avg.time:   122 [ms]    glob_rost: 694
Presences:  tot.sent:     831    tot.rcvd:   6483
Packets:    created:      8073       sent:   8072
            canceled:     0      in queues:   1
```

Pruebas con valores adecuados a entorno más real empresarial

Resultado Prueba 3 – Sin Bandersnatch

```
Conn stat:  conns:  total: 51    estabilished: 51
            kills:  total: 0     unexpected:   0
Messages:   tot.sent:     3048   tot.rcvd:   3047
            rcvd.offline: 0      rcvd.admin: 0
            rcvd.normal:  3047   fwd: 1524       avg.time:   53 [ms]
            diff check:   1      stability:  1
Roster:     tot.adds:     86     avg.time:   76 [ms]
            tot.dels:     146    avg.time:   89 [ms]     glob_rost: 705
Presences:  tot.sent:     812    tot.rcvd:   6298
Packets:    created:      4512       sent:   4512
            canceled:     0      in queues:   0
```

Resultado Prueba 4 – con Bandersnatch

```
Conn stat:  conns:  total: 51    estabilished: 51
            kills:  total: 0     unexpected:   0
Messages:   tot.sent:     3039   tot.rcvd:   3006
            rcvd.offline: 0      rcvd.admin: 0
            rcvd.normal:  3006   fwd: 1503       avg.time:   56 [ms]
            diff check:   33     stability:  0
Roster:     tot.adds:     87     avg.time:   95 [ms]
            tot.dels:     145    avg.time:   102 [ms]    glob_rost: 707
Presences:  tot.sent:     817    tot.rcvd:   6504
Packets:    created:      4499       sent:   4499
            canceled:     0      in queues:   0
```

Notamos que en las pruebas realizadas con Bandersnatch, dicho componente utilizaba el 50% de CPU del sistema para realizar las tareas de registro y que en las pruebas de carga excesiva (dos primeras de este juego), se ve topeado el sistema por Bandersnatch , Jabberd, Jabbsimul y MySQL, de allí se puede ver la demora incrementada de 20 milisegundos en la entrega de mensajes, en las siguientes 2 pruebas adecuadas de carga extrema pero no excesiva, notamos que la entrega de los mensajes prácticamente no se ve afectada por el componente Bandersnatch.

Cabe destacar que tanto con o sin es sistema topeado por CPU, las operaciones de cambios en lista de contactos se ven demorados (en 20-30 ms), es de suponer que es por la utilización de ambos componentes (Jabberd y Bandersnatch) sobre el mismo recurso (MySQL).

Si bien por falta de tiempo no se realizaron todos los tests necesarios para probar todos los límites y los detalles de la implantación (por ejemplo, prueba de escalabilidad sumando usuarios al servicio, pruebas utilizando motores de datos transaccionales o no en las Bases de Datos, etc.) , las pruebas realizadas las consideramos de gran importancia, ya que pusimos a prueba un servicio empresarial implementado en base a Software Open/Free en un hardware antiguo y se logró que el servicio estuviera: activo, bajo una situación de stress y estable durante aproximadamente 5 días. Además durante dicha prueba, la performance fue mas que satisfactoria.

7 CONCLUSIÓN

A lo largo del estudio nos encontramos que la Mensajería Instantánea en el entorno Empresarial es viable y puede otorgar beneficios a la empresa a un bajo costo de implantación e infraestructura, observamos la existencia de productos comerciales/propietarios y proyectos Open/Free muy maduros.

Si bien en algunos casos en estos productos Open/free encontramos que una implementación seria del servicio requiere un conocimiento importante de sistemas operativos, redes y programación, también se encontraron productos muy completos para utilización 'out of the box' .

Notamos también que las empresas Comerciales de Software promueven dicha tecnología a nivel empresarial y desarrollan productos de gran porte para implementar dicho servicio lo cual marca una tendencia.

Tanto soluciones comerciales cono Open/Free mayormente se están basando en protocolos abiertos (en algunos casos con extensiones propietarias) certificados por IETF lo que garantiza interoperabilidad entre las distintas implementaciones de los productos y con posibilidad de interfaces estándar con otros servicios de otra índole.

Vimos que existen problemas acarreados al servicio, en general muchos derivados de una mala utilización del mismo:

- Utilización para conversaciones personales, distracción del empleado.
- Escape de información sensible a la empresa.
- Abusos de recursos de infraestructura informática de la Empresa.

Pero que también contamos con la existencia de las herramientas necesarias para mitigar todo este tipo de problemas.

Estudiamos los requerimientos necesarios para implantar un servicio que contemple puntos como: seguridad, maximización de productividad, interoperabilidad, estabilidad, escalabilidad, etc.

Se estudiaron los protocolos de mensajería abiertos (todos propuestos por la IETF como protocolos estándares) existentes mas importantes (IRC, SIMPLE y XMPP) y se llega a que XMPP es el que está mas desarrollado, y con mas soluciones implementada existentes. Las ventajas de este protocolo como la extensibilidad y su estructuración basada en XML, hace del mismo una plataforma de comunicación en tiempo real no solo para utilizarse en tecnologías de Mensajería Instantánea sino también para comunicación en tiempo real entre aplicaciones, esto en un Empresa puede utilizarse tanto para aplicaciones internas como para ofrecer servicios a otras Empresas.

Luego de el análisis presentado a cerca de el protocolo XMPP, encontramos que el mismo es muy elegante, sencillo, entendible, potente y extensible.

La arquitectura cliente-servidor propuesta por XMPP también es muy elegante y sencilla, pudiendo obtener sistemas distribuidos inclusive con componentes de los servidores distribuidos, escalabilidad horizontal, clusters de servidores, implementación sencilla para failover y control total sobre el flujo de la información.

Notamos que XMPP brinda todas las herramientas y especificaciones para extender el protocolo y para poder lograr interoperabilidad con (y entre) aplicaciones empresariales como correo, groupware, sistemas de ERP, CRM, etc.

Estudiamos las alternativas más maduras existentes en el mundo Open/Free, basadas en el protocolo abierto XMPP y nos encontramos que mas allá de que una implementación del servicio completa empresarial tiene un buen grado de dificultad (sobre todo por las herramientas complementarias), no es imposible y es viable.

Concluimos que dichos productos Open/Free están ya aptos para brindar un servicio empresarial de porte de PyME y que dependiendo de los requerimientos de la empresa, se puede llegar desde una implementación sencilla con funcionalidades básicas de Mensajería Instantánea y Presencia hasta funcionalidades avanzadas como salas multiusuario, auditorías, etc. y que tomando las precauciones necesarias de seguridad, se puede utilizar el servicio como una herramienta de comunicación fiable (tanto interna como hacia otras empresas proveedoras, socios de negocios o clientes) con bajos costos asociados y que puede suponer ahorros de telefonía para la empresa, también puede suponer un incremento de productividad en cuanto a la utilización del servicio de una forma de multi-tarea realizando otras actividades ya que no requiere tanta atención como cuando se está utilizando otro medio de comunicación como por ejemplo: el teléfono.

Es claro que los beneficios que aporta un servicio empresarial de Mensajería Instantánea en una PyME depende del tamaño de la empresa (tamaño en cuanto a cantidad de personal que utiliza servicios informáticos), del rubro de la misma y de su estrategia a futuro de la calidad de los servicios que brinde a sus clientes. De todas formas, dada la flexibilidad en cuanto a utilización para distintos fines (uso interno, servicios a clientes, consultas a proveedores o combinación de ellos), es prácticamente un servicio conveniente para muchas empresas.

Repetimos que, como toda innovación, su adopción dependerá de las circunstancias particulares de cada empresa y de los beneficios tecnológicos y comerciales visibles que dicha tecnología pueda aportarle.

8 APÉNDICES

8.1 INSTALACIÓN DE JABBERD Y CONFIGURACIÓN BÁSICA

Una vez obtenido el tarball con los fuentes de jabberd2 (http://jabberstudio.org/projects/jabberd2/releases/) debemos recoger ciertos datos, presentados en la tabla, siguiente a fin de proceder con la instalación.

Información requerida para la instalación de Jabberd2.

Parámetro	Requerido	Descripción	Su información
Jabberd User and Group	Sí	Usuario y grupo que será usado para correr jabberd	Usuario: jabber Grupo: jabber
Directorio PID	Sí	Directorio donde jabberd guarda los archivos PID	/usr/local/var/jabberd/pid
Directorio Log	No[1]	Directorio para los log de jabberd. Si no es especifica en los archivos de configuración por defecto el log es syslog.	/usr/local/var/jabberd/log
Paquete de Autenticación	Sí	Paquete de terceros para usarse en la autenticación de los usuarios	MySQL, PostgreSQL, Berkeley DB, OpenLDAP con PAM
Paquete para almacenamiento de Datos	Sí	Paquete de terceros para almacenar información de jabberd	MySQL, PostgreSQL o Berkeley DB
Directorio de Datos	Berkeley DB	Directorio para los archivos de Berkeley DB	/usr/local/var/jabberd/db
Usuario y clave para MySQL	MySQL	Usuario y clave MySQL que jabberd usara para conectarse	Usuario: jabberd2 Clave: secreto
Usuario y clave para PostgreSQL	PostgreSQL	Usuario y clave PostgreSQL que jabberd usara para conectarse	Usuario: jabberd2 Clave: secreto
Seteos de Conexión OpenLDAP	OpenLDAP	Seteos de conexión para su servidor OpenLDAP : FQDN del servidor o IP, puerto y versión (v2 o v3)	
Usuario y clave OpenLDAP	OpenLDAP	Usuario y clave para conectarse al servidor OpenLDAP. Necesario solamente si el servidor no acepta conexiones anónimas.	
Seteos de consulta OpenLDAP	OpenLDAP	ND (Nombre distinguido) Base y atributo ID de usuario usado para construir consultas para el servidor. El ND Base puede ser un el ND raíz del servidor o un NDR (nombre distinguido relativo) bajo el cual se encuentra el ID del usuario	
Nombre del Host	Sí	Nombre del host donde el servidor jabberd será instalado. Para servidores accesibles desde Internet, este debe ser algo parecido a **midominio.com**	

[1]Nosotros aconsejamos usar un archivo de log distinto de syslog. Esto ayudará en la administración y mantendrá acotado el log del sistema.

Parámetro	Requerido	Descripción	Su información
Ubicación de la Clave SSL	No	Ubicación para el archivo pem OpenSSL, necesario para comunicación encriptada	/usr/local/etc/jabberd/server.pem
Usuario y clave para el Router	No	Usuario y clave para la conexión de componentes con el componente Router de Jabberd	Usuario: jabberd2 Calve: secret

Una vez recabados estos datos, se está en condiciones de comenzar el proceso de instalación del servidor Jabberd.

8.1.1 Instalación de Jabberd2

8.1.1.1 Creación del usuario y del grupo Jabber

Para comenzar, se deberá crear un grupo y un usuario **`jabber`** para ejecutar el servidor

```
su
groupadd jabber
useradd -g jabber jabber
```

8.1.1.2 Creación de los directorios para PID's y Logs

Luego como superusuario se deberá ejecutar la siguiente secuencia de comandos con el fin de crear es directorio donde jabberd guardará los archivos pid

```
su
mkdir -p /usr/local/var/jabberd/pid/
chown -R jabber:jabber /usr/local/var/jabberd/pid/
```

Puede ahora (si lo desea) crear el directorio para los archivos de log (recomendado).

```
mkdir -p /usr/local/var/jabberd/log/
chown -R jabber:jabber /usr/local/var/jabberd/log
```

Nota: por defecto jabberd escribe los mensajes de log a syslog. Para que lo haga en directorio recién creado debe explicitarse en los archivos de configuración de los componentes del servidor. Más adelante veremos uno por uno estos archivos.

8.1.2 Instalación de prerequisitos

Para su instalación mas básica, Jabberd 2 tiene 2 prerequisitos:

- Paquete de Almacenamiento de Datos (uno de los siguientes)
  - MySQL
  - PostgreSQL
  - Berkeley DB
- Paquete de Autenticación (uno de los siguientes)
  - MySQL
  - PostgreSQL
  - Berkeley DB
  - OpenLDAP
  - PAM

Opcionalmente se necesitan:

- OpenSSL (versión 0.9.6b o superior)
- Libidn (versión 0.3.0 o superior)

Jabberd 2 puede ser instalado sin estos paquetes; aunque es fuertemente recomendado que se instalen (sobre todo OpenSSL en implantaciones empresariales).

Jabberd 2 requiere paquetes de almacenamiento de datos y para autenticación de los usuarios que a él se conectan ya que no dispone de almacenamiento propio. (esto no es una carencia y es beneficioso en grandes implantaciones. De hecho los demás productos de M.I. que implementan almacenamiento propio de datos, recomiendan que en grandes instalaciones, aproximadamente de más de 50 usuarios, se utilicen paquetes específicos de almacenamiento de datos). Observar que tanto MySQL, PostgreSQL como Berkeley DB cumplen con ambos requerimientos.

La instalación realizada en el marco de este proyecto utiliza MySQL versión Max (puede utilizarse la estándar) para ambos propósitos, pues este gestor es requisito también de otros componentes auxiliares tales como Bandersnatch (log de mensajes) y Users-Agent (Componente JUD – Jabberd User Directory).

Dado los bajos recursos de la PC utilizada, decidimos minimizar al máximo posible el uso de distintos paquetes, optando por aquellos que fueran comunes a varios servicios.

8.1.2.1 Instalación de OpenSSL

OpenSSL provee de encriptación a las comunicaciones cliente-servidor y servidor-servidor. El protocolo XMPP **RFC3920** requiere que los servidores Jabberd soporten TLS (Transport Security Layer), TLS está basado en el protocolo SSL 3.0 de Netscape. OpenSSL implementa SSL v2 y v3 y TLS v1.

Se requiere OpenSSL versión 0.9.6b o superior

Se hace hincapié en el cuidado tomado en este paso. Si se debe actualizar OpenSSL, probablemente se tenga que recompilar y reinstalar el software que dependa de OpenSSL, pues de otra manera, podría dejar de funcionar adecuadamente. Por más información, habrá que dirigirse al sitio web de OpenSSL (www.openssl.org).

Básicamente deben seguirse estos pasos:

Bajar el archivo **openssl-0.9.nz.tar.gz** del sitio OpenSSL, donde "**nz**" es la última versión estable. El software que contiene el archivo es un paquete de encriptación fuerte por lo tanto deberá chequear el marco legal[1] local antes de bajarlo.

Se deberá extraer los archivos fuentes de OpenSSL. Para ello se deberá cambiar de directorio pasando al mismo donde se haya realizado la descarga. Se deberá extraer los archivos ejecutando el siguiente comando

```
tar -zxvf openssl-0.9.nz.tar.gz
```

Esto creará un directorio **openssl-0.9.nz.** Se deberá cambiar de directorio a este último.

```
cd openssl-0.9.7b
```

Luego se deberá configurar OpenSSL para el sistema que se esté actualizando

```
./config --prefix=/usr
```

Por defecto OpenSSL se instala en **/usr/local/openssl**. Se usará la opción **--prefix=/usr** para que las bibliotecas OpenSSL se instalen en el directorio **/usr**. Se recomienda el uso de esta opción para que OpenSSL este más accesible, no solamente a Jabberd sino a otras aplicaciones también.

Se compilará usando **make**, pudiéndose testear antes de la instalación con **make test.**

No debería de haber errores en la salida. Ahora utilizando la cuenta de superusuario se procederá a a instalar usando **make install.**

8.1.2.2 Instalación de Libidn

Libidn provee funcionalidades de manipulación de strings a Jabberd 2. Anteriormente a Jabberd 2 stable 3, libidn estaba incluida en la distribución del servidor, pero por problemas de licencias debió ser excluida de la misma realizándose actualmente de manera independiente.

La versión mínima requerida es 0.3.0 o superior. Los fuentes y más información pueden obtenerse en www.gnu.org/software/libdn.

Pasos básicos de instalación

- Descargar el archivo **libidn-0.n.n.tar.gz**
- Descomprimir con **tar -zxvf libidn-0.n.n.tar.gz**
- Cambiar de directorio **cd libidn-0.n.n**
- Configurar Libdn para el sistema. Use la opción **--prefix=/usr** para que se instale en el directorio **/usr/lib**. El comando sería entonces
  ./configure --prefix=/usr
- Compilar con **make**
- Cambiar a superusuario e instalar

[1]En países como Estados Unidos este tipo de software es considerado un arma y por lo tanto se halla bajo la reglamentación del Departamento de Defensa. El uso de software no habilitado es un delito federal grave. [21]

```
su
make install
```

Ahora resta instalar el paquete de Datos y el Autenticación, en este caso nosotros usamos MySQL MAX y se instalo usando los archivos **.rpm** provistos por la distribución Mandrake 10.0 CE. Los mismos corresponden a la versión 4.0.1.8-1mdk

La siguiente tabla muestra los distintos paquetes instalados

Nombre	Descripción
libmysql12	Este paquete contiene las bibliotecas compartidas (*.so*) las cuales son necesarias para que ciertos lenguajes y/o aplicaciones las carguen al utilizar MySQL.
libmysql12-devel	Este paquete contiene las bibliotecas y archivos de cabecera para el desarrollo de aplicaciones clientes. Necesarias para la compilación de Jabberd.
MySQL-bench	Contiene los scripts y los datos para benchmark. No es necesaria para la instalación de Jabberd.
MySQL-client	Contiene los clientes estándar de MySQL.
MySQL-common	Archivos comunes del servidor de base de datos MySQL.
MySQL-Max	Puede utilizarse la versión estándar pues no se están utilizando los features adicionales (que básicamente es incluir también el storage Engine de BerkleyDB)

8.1.3 Compilación e Instalación de Jabberd2

Los siguientes pasos intentan ser una descripción paso a paso de la compilación e instalación de Jabberd2 para el caso particular de nuestro proyecto. Información más detallada puede encontrarse en http://www.jabberdoc.org.

Primero de deberá descargar la última versión disponible del servidor jabberd2. El nombre del archivo será de la forma j**abberd-2.0sn.tar.gz** donde "**n**" es la última versión estable.

Luego se deberá de descomprimir los fuentes a algún directorio (es recomendable en `$HOME` o en el propio "`/home/jabber`") con **`tar -zxvf jabberd-2.0sn.tar.gz`**.

Procederemos ahora a la configuración. Primero nos cambiamos al directorio creado durante la descompresión **`cd jabberd-2.0sn`**. Se puede obtener información sobre las opciones de configuración realizando **`./configure –help`**.

En nuestro caso usamos OpenSSL, MySQL, Libidn, se habilito la opción de depuración y se instaló en el directorio por defecto **`/usr/local/bin`**.

Para ello el comando es

```
./configure enable-debug --enable-mysql --enable-ssl --enable-idn \
--with extra-include-path=/usr/local/include:/usr/local/include/mysql
\
--with-extra-library-path=/usr/local/lib:/usr/local/lib/mysql
```

Luego de configurado se ejecuta la siguiente secuencia a fin de compilar e instalar

```
make
su
make install
```

La ubicación de los archivos por defecto es la siguiente

```
Archivos de configuración ..................... /usr/local/etc/jabberd
Binarios (jabberd, c2s, resolver,
router, s2s, sm) ............................. /usr/local/bin
```

Una vez instalados debemos establecer como propietario al usuario jabber

```
chown -R root:jabber /usr/local/etc/jabberd/*
chmod -R 640 /usr/local/etc/jabberd/*
ln -s /usr/local/etc/jabberd/ /etc/jabberd
ln -s /usr/local/etc/jabberd/ /home/jabber/etc
```

Este último es opcional, pero facilita las tareas de configuración. Se recomienda realizar también un enlace simbólico al directorio de logs previamente creado. Ahora ya tenemos el servidor instalado.

8.1.4 Configuración de MySQL para jabberd

Utilizando el script **`db-setup.mysql`** en la distribución de jabberd crearemos la base de datos en el servidor MySQL donde Jabberd guarda la información por él necesitada.

```
mysql -u root -p
mysql>\. db-setup.mysql
```

Luego deberemos crear el usuario para que jabberd se conecte al servidor MySQL. Desde la consola MySQL ejecutamos el siguiente comando

```
GRANT select,insert,delete,update ON jabberd2.*
to jabberd2@localhost IDENTIFIED by 'secret';
```

Nota: Socket MySQL por defecto. Jabberd 2 estable 3 se conecta al servidor MySQL vía socket en **`/tmp/mysql.sock.`** *. El socket por defecto, cuando se instala MySQL usando el código fuente, está en* **`/var/lib/mysql/mysql.sock.`** *por lo tanto se deberá crear un enlace simbólico:*

```
ln -s /var/lib/mysql/mysql.sock /tmp/mysql.sock.
```

Si no está seguro donde puede estar el socket de MySQL revise el archivo de configuración del servidor, usualmente ubicado en

`/etc/my.cnf` *o* **`/etc/mysql/my.cnf.`**

8.1.5 Configuración del Servidor Jabberd2

Comenzamos por una configuración básica para testear que el servidor levante en forma adecuada. Para ello deberemos editar el contenido de los archivos de configuración de cada componente de jabberd2 según sea necesario.

Comenzaremos por los archivos **sm.xml** y **c2s.xml** responsables de la configuración de los componentes Session Manager y Client to Server respectivamente.

El primer paso es el nombre del host en ambos archivos, este formará parte las JID (Jabber ID), por lo tanto deberá ser resoluble por la red. Es decir, si usamos algo del estilo **jabber.fing.edu.uy** este debe ser resuelto adecuadamente por el DNS.

Nota : *se puede usar solamente el nombre del dominio (fing.edu.uy) para las ID de Jabberd2, pero en este caso el servidor DNS debe estar correctamente configurado para resolver las ID al host donde corre el servicio Jabber. Más adelante veremos el agregado de los registros SRV a la DNS.*

Nota 2 : en la práctica realizada en el In.Co. logramos autorización para conectar nuestro servidor jabber (jabber.lab-inco.fing.edu.uy) dentro de la subred del laboratorio interno del Instituto. Para poder acceder al mismo dentro de facultad, debemos conectarnos al firewall (tortuga.fing.edu.uy) de dicha subred, en el cual, mediante port-forwarding se redirige la conexión al servidor. Por esta razón, las JID son del estilo tortuga.fing.edu.uy

En el archivo **sm.xml** deberemos ir a la siguiente sección

```
<!-- Session manager configuration -->
<sm>
<!-- Our ID on the network. Users will have this as the domain part
     of their JID. If you want your server to be accessible from
     other Jabber servers, this ID must be resolvable by DNS.s
     (default: localhost) -->
 <id>tortuga.fing.edu.uy</id>
```

En el archivo **c2s.xml** la sección correspondiente es

```
<!-- Local network configuration -->
<local>
  <!-- Who we identify ourselves as. This should correspond to the
       ID (host) that the session manager thinks it is. You can
       specify more than one to support virtual hosts, as long as you
       have additional session manager instances on the network to
       handle those hosts. The realm attribute specifies the auth/reg
       or SASL authentication realm for the host. If the attribute is
       not specified, the realm will be selected by the SASL
       mechanism, or will be the same as the ID itself. Be aware that
       users are assigned to a realm, not a host, so two hosts in the
       same realm will have the same users.
       If no realm is specified, it will be set to be the same as the
       ID. -->
    <!-- <id>somemachine.somedomain.com</id> -->
    <id>tortuga.fing.edu.uy</id>
```

Luego se deberá configurar la sección correspondiente al paquete de almacenamiento de datos y el de autenticación, en ambos casos se trata de MySQL (es la opción a usarse por defecto en ambos).

Comenzaremos con el almacenamiento, el cual se realiza en el **sm.xml**. Deberemos buscar la sección storage y editarla para que contenga la siguiente información

```
<!-- Storage database configuration -->
<storage>
 <!-- By default, we use the MySQL driver for all storage -->
 <driver>mysql</driver>
```

```
<!-- MySQL driver configuration -->
   <mysql>
     <!-- Database server host and port -->
     <host>localhost</host>
     <port>3306</port>
     <!-- Database name -->
     <dbname>jabberd2</dbname>
     <!-- Database username and password -->
     <user>jabberd2</user> <!-- database user for jabberd    -->
     <pass>secret</pass>   <!-- database password for <user> -->
     <!-- Transacation support. If this is commented out,
          transactions will be disabled. This might make database
          accesses faster, but data may be lost if jabberd crashes.

          This will need to be disabled if you are using a MySQL
          earlier than v3.23.xx, as transaction support did not
          appear until this version. -->
     <transactions/>
   </mysql>
 </storage>
```

Pasamos ahora a la configuración del paquete de autenticación. La misma se realiza en el archivo **c2s.xml**. La sección correspondiente dentro del archivo contiene más información (configuración completa en el apéndice 8.8.1.1), pero aquí por razones de espacio, dejaremos sólo lo pertinente.

```
<!-- Authentication/registration database configuration -->
 <authreg>
 <!-- Backend module to use -->
   <module>mysql</module>
 ...
 <!-- MySQL module configuration -->
 <mysql>
   <!-- Database server host and port -->
   <host>localhost</host>
   <port>3306</port>

   <!-- Database name -->
   <dbname>jabberd2</dbname>

   <!-- Database username and password -->
   <user>jabberd2</user>
   <pass>secret</pass>
 </mysql>
  ...
</authreg>
```

Luego de configurado el nombre del host, la aplicación encargada del almacenamiento de datos y la de autenticación, estamos en condiciones para testear el servidor jabberd.

Para ello debemos seguir la siguiente secuencia de comandos

```
su
su jabber
cd /usr/local/bin
./jabberd
```

El comando ps debería mostrar algo similar a lo siguiente

```
[jabber@localhost]$ ps
  PID TTY          TIME CMD
 2674 pts0     00:00:00 bash
 2724 pts0     00:00:00 perl
 2725 pts0     00:00:00 router
 2726 pts0     00:00:00 resolver
 2727 pts0     00:00:00 sm
 2728 pts0     00:00:00 s2s
 2729 pts0     00:00:00 c2s
 2730 pts0     00:00:00 ps
```

La conectividad puede ser testeada mediante algún cliente (GAIM, Gabber) registrando dos cuentas de prueba y estableciendo comunicación entre ellas.

8.2 INSTALACIÓN Y CONFIGURACIÓN DE MU-CONFERENCE

8.2.1 Extensión JEP-0045

A modo de introducción a la implementación de el componente MU-Conference presentamos la extensión LEP-0045. En la misma se discuten aspectos relacionados a la configuración, participación y administración de salas de conferencia basadas en texto. Los requerimientos planteados fueron bien tratados dentro de la comunidad Jabber o familiares en ambientes del estilo como en el Internet Relay Chat (IRC).

8.2.1.1 Requerimientos

En el momento de plantear esta extensión se utilizo como punto de partida las funcionalidades ofrecidas por los servicios de multiconferencia para Jabber existentes al momento los cuales utilizaban el protocolo original "groupchat 1.0". Esos requerimientos fueron extendidos hasta alcanzar la siguiente lista:

Requerimiento	Tipo
Los mensajes dentro de las salas de chat son del tipo especial "groupchat"	B
Cada sala esta identificada como sala@servicio donde "sala" es el nombre de la misma y "servicio" es el nombre del servidor en el cual esta corriendo el servicio de multiconferencia.	B
Cada ocupante de la sala es identificado por sala@servicio/apodo. Apodo (nick en inglés) es especificado por el usuario al entrar a la sala y puede cambiarlo en el transcurso de la "visita" a la misma	B
El usuario entra a una sala (convirtiéndose en ocupante de la misma) enviando "presence" a la misma.	B
El usuario sale de una sala enviando "unaviable" a la misma	B
El usuario puede cambiar su nick o estado de disponibilidad dentro de la sala enviado "update"	B
Logeo nativo de las conversaciones	E
Habilitar que el usuario solicite membresía a la sala	E
Habilitar a los ocupantes para que puedan ver la JID completa de otro ocupante en salas no anónimas.	E
Habilitar a los moderadores ver el JID completo de un ocupante de forma semi anónima	E
Permitir que solo los moderadores cambien el tema de la sala	E
Habilitar a los moderadores a expulsar temporalmente a participantes y visitantes de las salas	E
Habilitar a los moderadores a otorgar y revocar el derecho a voz (es decir a hablar) en las salas moderadas y manejar la "lista de voces"	E
Habilitar a los administradores otorgar y revocar permisos de moderador y manejar la lista de moderadores	E
Habilitar a los administradores a expulsar definitivamente a usuarios de una sala y manejar la lista de expulsados definitivos	E
Habilitar a los administradores a otorgar y revocar privilegios de membresía y manejar la lista de miembros en las salas exclusivas	E

Habilitar a los dueños de la sala limitar el número de ocupantes	E
Habilitar a los dueños especificar otros dueños	E
Habilitar a los dueños otorgar y revocar privilegios de administración y manejar las lista de administradores	E
Habilitar a los dueños de las salas destruirlas	E
Las salas podrán ser publicas o escondidas	E
Persistentes o temporales	E
Protegidas por clave o inseguras	E
Abiertas o solo para miembros	E
Moderadas o no	E
No anónimas o semi anónimas	E

B significa funcionalidad Básica y E propuesta como extensión por la JEP-0045

Por otro lado se establecieron dos dimensiones mutuamente ortogonales que permiten determinar la calidad de un usuario, la cual dura el tiempo de visita a la sala por parte del usuario. No hay mapeo uno a uno entre estas afiliaciones y los roles de los ocupantes de una sala. Las afiliaciones son otorgadas, revocadas y mantenidas en base al JID común de cada usuario. Ellas son:

- **Afiliación** que es una asociación o conexión de larga vida con una sala. Las afiliaciones posibles son
- **dueño (owner)** significado normal o común.
- **administrador (admin)** idem.
- **miembro (member)** usuario incluido en la "whitelist" o lista de habilitados a entrar a una sala cuando esta es solo para miembros o para quienes están registrados en salas abiertas.
- **descastado (outcast)** usuario expulsado definitivamente (banned) de una sala.

Si un usuario entra a una sala sin una afiliación definida estos se le define como "none" pero esta afiliación no persiste durante la visita.

Dueños (Owners) y administradores (admins) son inmunes por definición a ciertas acciones. Específicamente **no pueden** ser expulsados temporal (kicked) o definitivamente (banned) de las salas. Un administrador debe perder su afiliación como tal antes de ser sujeto a tales acciones.

Las afiliaciones permiten a los administradores mantener "whitelists" en salas con membresía y a los usuario en salas abiertas un registro efectivo en las mismas.

Información sobre la afiliación **debe** ser enviada en todas las stanzas de presencia generadas por la sala y enviadas a los ocupantes de la misma

- **Rol** – posición temporal o nivel de privilegio de un usuario en la sala. Estos son
  - **moderador (moderator)** significado normal.
  - **participante (participant)** aplica a los ocupantes que no tienen privilegios administrativos; en salas moderadas se los definen como "los que tienen voz" en contraste a los visitantes.

- **visitante (visitor)** en salas moderadas los ocupantes sin derecho a voz.

También es posible no tener un rol definido y estos duran el tiempo de visita del ocupante a la sala.

Privilegios asociados a los Roles

Privilegio	Ninguno[1]	Visitante	Participante	Moderador
Presencia en la sala	No	Si	Si	Si
Recibir mensajes	No	Si	Si	Si
Cambiar el estado de disponibilidad	No	Si	Si	Si
Cambiar el Nick de la Sala	No	Si*	Si	Si
Envío de mensajes privados	No	Si*	Si	Si
Invitar a otros usuarios	No	Si*	Si*	Si*
Enviar mensajes a todos	No	No**	Si	Si
Modificar el tema de la sala	No	No	Si*	Si
Expulsar visitantes y participantes	No	No	No	Si
Otorgar derecho de voz	No	No	No	Si
Revocar derecho de voz	No	No	No	Si***

*** Por defecto; los seteos de configuración podrían restringir más aún estos privilegios**
**** Una implementación puede otorgar derecho de voz por defecto a los visitantes en salas no moderadas**
***** Un moderador NO DEBE ser capaz de revocar el derecho a voz de un administrador o de un dueño de sala.**

Privilegios asociados a la afiliación

Privilegio	Descastado	Ninguno	Miembro	Administrador	Dueño
Entrar a salas abiertas	No	Si**	Si	Si	Si
Registrarse en salas abiertas	No	Si	N/A	N/A	N/A
Entrar a salas exclusivas	No	No	Si*	Si	Si
Expulsar (ban) miembros y usuarios no afiliados	No	No	No	Si	Si
Editar la lista de miembros	No	No	No	Si	Si
Editar la lista de moderadores	No	No	No	Si**	Si**
Editar la lista de administradores	No	No	No	No	Si
Editar la lista de dueños	No	No	No	No	Si
Cambiar las definiciones de la sala	No	No	No	No	Si
Destruir la sala	No	No	No	No	Si

*** Por defecto, usuarios no afiliados entran en salas moderadas como visitantes, y a salas abiertas como participantes. Un miembro entra a salas como participante. Administradores y dueños como moderadores.**

**** Un administrador o un dueño NO DEBE ser capaz de revocar privilegios de moderación de otro administrador o dueño.**

[1]Significa la ausencia de Rol.

8.2.1.2 Consideraciones adicionales

- Si el usuario entra a una sala protegida por clave y el mismo no la suministra, el servicio debe negar la entrada al usuario a la sala.
- Si un usuario intenta entrar a una sala exclusiva para miembros y no lo es, se le negará la entrada
- Si el usuario fue expulsado definitivamente (es decir, se le cambio el rol a outcast) se le negara el ingreso a las salas.
- Si se produce un conflicto de nickname durante el ingreso, se negará el acceso

En cualquiera de estos casos se notificará adecuadamente del motivo de la negación.

Como se puede apreciar, esta extensión posee funcionalidades y mecanismos que permiten uso más que adecuado de un servicio de estas características. Si el lector desea profundizar en el tema, sobre todo en el modo que se implementan los distintos casos de usuario con las stanzas, deberá referirse al texto de la JEP-0045

8.2.1.3 Glosario de términos usados en Multiconferencia

- **Ban** – remover a usuario de una sala de tal forma que el usuario quede deshabilitado a re-entrar a la mima (hasta o al menos que se removido este estado). El usuario " banned" posee una afiliación de descastado.
- **JID Plan (Bare JID)** – la dirección user@host por el cual un usuario es identificado fuera del contexto de cualquier sesión o recurso existente, en contraste con **JID Completa** o **JID de Sala**.
- **JID Completa (Full JID)** – la dirección user@host/resource por el cual se conoce a un usuario fuera del contexto de una sala.
- **Invitación (Invitation)** – mensaje especial enviado por un usuario a otro pidiendo al receptor que entre a una sala.
- **IRC** -- Internet Relay Chat.
- **Kick** – remover temporalmente a un participante o a un visitante de una sala. El usuario tendrá permitido el acceso en cualquier momento. Tiene rol "none".
- **Logging** – almacenamiento de la discusión que ocurre dentro de una sala para recuperación dentro o fuera de la misma.
- **MUC** – protocolo de conferencia multi usuario documentado por el JEP-0045.
- **Ocupante (Occupant)** – cualquier usuario Jabber dentro de una sala.
- **Mensaje Privado (Private Message)** – mensaje enviado por un ocupante a otro directamente.
- **Sala (Room)** – espacio virtual en el cual entran figuradamente los usuarios Jabber para participar en tiempo real de discusiones basadas en texto con más de un usuario.

- **ID de Sala (Room ID)** – el identificador de nodo correspondiente al **Room JID**, el cual puede ser opaco y como tal carente de sentido para los humanos a diferencia del **Room Name.**
- **JID de Sala (Room JID)** – la dirección room@service/nick por la cual un ocupante es identificado en el contexto de una Sala.
- **Nombre de Sala (Room Name)** – Nombre para la sala en lenguaje natural y amigable al usuario, configurado por el **Dueño** de la **Sala** y presentado en el **Servicio de Descubrimiento**.
- **Apodo de Sala (Room Nickname)** – identificador de recurso del **JID de Sala**.
- **Lista de usuarios en el chatroom (Room Roster)** – representación a nivel de un cliente Jabber de los ocupantes de un chatroom.
- **Servidor (Server)** – Servidor Jabber que puede o no tener asociado un **Servicio** MUC.
- **Servicio (Service)** – Host que ofrece las capacidades de MUC; a menudo pero no necesariamente un subdominio del **Servidor** Jabber(e.j., conference.jabber.org).
- **Tema (Subject)** – tópico temporario de discusión dentro de una Sala.
- **Visita (Visit)** - "sesión" de un usuario en una sala, comienza cuando entra a la misma y finaliza cuando sale.
- **Derecho de Voz (Voice)**-- en **Salas Moderadas**, es el privilegio de poder enviar mensajes.

8.2.2 Instalación y configuración del componente

Este componente se puede descargar de http://jabberstudio.org/projects/mu-conference/releases/ y al momento de realizar la instalación la última versión estable era la 0.6.0.

Como único requisito requiere el JCR (**Jabberd Component Runtime library**), la cual se puede obtener desde http://jabber.terrapin.com/JCR/ la versión disponible al momento de realizar la instalación era la 0.2.4.

Lo que sigue es una breve secuencia de pasos a seguir para compilar e instalar el componente mu-conference.

```
tar -zxvf jcr-0.2.4.tar.gz
cd jcr-0.2.4
make
```

Luego copiamos el **mu-conferece.0.6.0.tar.gz** dentro del directorio actual y lo descomprimimos

```
tar -zxvf mu-conference-0.6.0.tar.gz
```

Copiamos los siguientes archivos dentro del directorio con los fuentes de mu-conference

```
cp src/main.c     mu-conference-0.6.0/src
cp src/jcomp.mk   mu-conference-0.6.0/src
```

Cambiamos de directorio y compilamos

```
cd mu-conference-0.6.0/src
make -f jcomp.mk
```

Esto generará el ejecutable de **mu-conference** el cual habrá que copiar al directorio donde se encuentran el resto de los componentes del servidor jabber (**/usr/local/bin).**

Luego deberemos copiar el archivo de configuración **muc-jcr.xml** provisto en la distribución de mu-conference al directorio donde se encuentran los archivos de configuración de jabberd **/etc/jabberd**.

Incluimos en Apéndices archivo de configuración muc-jcr.xml

Modificaciones en los demás archivos de configuración de jabber

8.2.2.1 Archivo `router-users.xml`.

```
<user>
  <!-- corresponde al valor de tag <name> en la sección <jcr> de
       muc-jcr.xml -->
  <name>mu-conference</name>
  <!-- corresponde al valor del tag <secret> de la sección <jcr> de
       muc-jcr.xml y al tag <secret> de la seccion <router>
       subsección <local> de de router.xml -->
  <secret>secret</secret>
</user>
```

8.2.2.2 Archivo `router.xml`.

Agregar la siguiente línea en la seccion **<aliases>**.

```
<alias name='conference.tortuga.fing.edu.uy' target='mu-conference'/>
```

donde **name** se corresponde con el valor del tag **<host>** de la sección **<jcr>** y **target** con el valor de **<name>** en la misma sección.

8.2.2.3 Archivo `sm.xml`.

Agregar las siguientes líneas en la seccion **<agents>**

```
<agent jid='conference.fing.edu.uy'>
  <name>Public Conferencing</name>
  <description>Public chatrooms for users.</description>
  <service>public</service>
  <groupchat/>
</agent>
```

Una vez, configurado el servicio y con el servidor jabberd levantado lo iniciamos con el comando

```
/usr/local/bin/mu-conference -c /etc/jabberd/muc-jcr.xml &
```

8.2.3 Creación de salas persistentes

Las salas de conferencia en Jabber pueden ser persistentes o no. Por defecto son no persistentes y pueden ser creadas por los distintos usuarios los cuales, una vez creada la sala, podrán invitar a sus "amigos" a participar en ella.

Las salas permanentes son creadas una sola vez y permanecen abiertas hasta que se las de baja. La distribución de mu-conference facilita un script perl para su creación. El mismo tiene los siguientes módulos como dependencias.

```
XML::Simple
Digest::MD5
```

El proceso general de instalación de módulos perl desde los fuentes y el enlace al sitio donde poder descargarlos se encuentra en el Apéndice B

Este script es de muy fácil manejo, simplemente hay que seguir las instrucciones. Incluimos los pasos seguidos por uno de los integrantes durante una prueba realizada con este script.

```
[root@tortuga rooms]# ./roommaker.pl
Please enter spool directory path (e.g. /usr/local/jabber/spool): /
usr/local/var/jabberd/spool
Please enter jid for the room: chat1@tortuga.fing.edu.uy
/usr/local/var/jabberd/spool/tortuga.fing.edu.uy/  doesn't  exist  -
Create? (Y/N) Y
Creating Directory

Configuring room chat1@ tortuga.fing.edu.uy
Filename:                                                          /
usr/local/var/jabberd/spool/tortuga.fing.edu.uy/2402f2f44290e8505ddac
91e3a0aad89e6ed079a.xml

General Options
---
Room name (text) [Default: chat1]: Chat1
Password (text) [Default: ]: pwdchat1
Room description/MOTD (text) [Default: ]: Sala 1 de prueba - creado
por mario
Room subject (text) [Default: ]: proyecto GNU/Linux
Bare JID of room creator (text)[Default: ]: mario@tortuga.fing.edu.uy
Is room public (0/1) [Default: 0]: 0
Maximum Users  (value) [Default: 0]: 20

Permission Options
---
Allow non-admins to see real jids (0/1) [Default: 0]: 0
Can users change subject (0/1) [Default: 0]: 0
Allow users to IQ query other users (0/1) [Default: 0]: 0

Legacy Options:
---
Consider all clients legacy (0/1) [Default: 0]: 1
Legacy join message (text) [Default: ]: join
Legacy leave message (text) [Default: ]: leave
Legacy rename message (text) [Default: ]: rename

Moderation Options:
```

```
---
Is room moderated (0/1) [Default: 0]: 1
Default entry type of participant (0/1) [Default: 0]: 0
Default entry type of participant (0/1) [Default: 0]: 0

Member-Only Options:
---
Make room member-only (0/1) [Default: 0]: 1
Allow members to send invites (0/1) [Default: 0]: 0

Logging Options:
---
Enable native room logging (0/1) [Default: 0]: 0
Skipping Logging options

Owner List:
---
JID of owner (Empty line to exit): mario@tortuga.fing.edu.uy
JID of owner (Empty line to exit):

Admin List:
---
JID of admin (Empty line to exit): mario@tortuga.fing.edu.uy
JID of admin (Empty line to exit):

Member List:
---
JID of member (Empty line to exit): mario@tortuga.fing.edu.uy
JID of member (Empty line to exit): daniel@tortuga.fing.edu.uy
JID of member (Empty line to exit):

Outcast List:
---
JID of outcast (Empty line to exit):

Writing Room definition file
Room registry not found. Creating

Writing updated Room registry file
```

Luego de terminado el script dentro del directorio **spool** se habrá creado un directorio **tortuga.fing.edu.uy** y dentro de el se hallan los archivos de configuración de la sala y el de registro de salas del servidor.

8.3 DIRECTORIO DE USUARIOS (JUD)

Hemos decidido utilizar para implementar este servicio un script perl llamado user-agent el cual es actualmente el utilizado en jabber.org y por ellos recomendados. Existen algunos otros como pueden ser JUD (el paquete esta nombrado como el servicio) pero actualmente son compatibles con las versiones 1.4.X del servidor jabberd.

User-agent tiene los siguientes requisitos

```
mysql
Net::Jabber v2.0    - for Jabber namespaces
Net::XMPP   v1.0    - for core XMPP support
XML::Stream v1.22   - for handling the XML streams
DBI                 - for MySQL database access
```

Los últimos cuatro son módulos PERL y podrán ser obtenidos desde http://www.cpan.org.

MySQL es utilizado para almacenar la información de los usuarios incluidos en el directorio. Si existían usuarios registrados anteriormente, estos deberán ser incluidos en la base JUD de Users-Agent a mano (mediante scripts). Los usuarios que se creen posteriormente a tener este servicio levantado serán registrados en ambas bases: jabberd2 y JUD.

Detallamos ahora el proceso de instalación de Users-Agent

Primero debemos obtener el tarball de la distribución (al momento de realizarlo era la versión 1.2). El mismo puede ser descargado desde

http://www.jabberstudio.org/projects/users-agent/releases/

Se debe elegir el directorio donde se desea instalar este componente y allí lo guardaremos. Nos cambiamos a ese directorio, descomprimimos y listo

```
tar -zxvf users-agent-1.2.tar.gz
cd users-agent-1.2
```

Comenzamos la configuración creando la base de datos JUD. Para ello disponemos de un script llamado createDB que se encargará de crearla. El mismo deberá poder conectarse en forma temporaria a alguna base dentro del servidor MySQL. La forma más fácil será deshabilitar la clave del root temporalmente

```
mysqladmin -u root -p password ""
```

Como alternativa, se puede editar el script en las líneas 13 y 17 que contiene el usuario y la clave del root. De todas maneras, si no deshabilita la calve deberá exportar la variable de ambiente DBI_DRIVER ante de ejecutar createDB. La misma deberá estar seteada a un valor **dbi:mysql:test:localhost**, donde test es una base de datos MySQL en producción.

Luego de deshabilitada la clave o editado el script podrá ser ejecutado mediante

```
./createDB
```

En caso de haberla deshabilitada NO OLVIDAR resetear la clave de root.

Ahora se podrá chequear que la base fue creada desde la consola de mysql

```
mysql -u root -p
mysql> show databases;
  +----------+
  | Database |
  +----------+
  | JUD      |
  | jabberd2 |
  | mysql    |
  | test     |
  +----------+
  4 rows in set (0.00 sec)
```

Una vez fue verificada la creación, se procederá a otorgar los permisos sobre esta. En caso de usar MySQL con jabberd el comando para esta acción es

```
mysql> use JUD;
mysql> GRANT ALL PRIVILEGES ON *.* TO 'jabberd2'@'localhost'
    ->WITH GRANT OPTION;
```

En caso contrario podrá crear el usuario al mismo tiempo que otorga los permisos mediante

```
mysql> use JUD;
mysql> GRANT ALL PRIVILEGES ON *.* TO 'jabberd2'@'localhost'
    -> IDENTIFIED BY 'secret_password' WITH GRANT OPTION;
```

Pasamos ahora a la edición del archivo de configuración de user-agent, **`config.xml`** .Los valores a setear se encuentran es estos tags:

<hostname>	nombre del servidor jabberd. Debe ser el valor del tag **<id>** al comienzo del archivo **`sm.xml`**
<port>	puerto donde escucha el router de jabberd. Por defecto es 5347
<secret>	es el secret compartido que el router de jabber usa para conectarse a componentes legados. Se encuentra dentro del tag **<secret>** en el archivo **`router.xml`**.
<name>	es el hostname para user-agent. Típicamente es algo del estilo users.somedomain.com, users.jabber.somedomain.com, jud.somedomain.com, etc.
<username>	nombre del usuario para conectarse a la base JUD
<password>	Clave del usuario

Una vez editado estos valores, user-agent está listo para ser usado. El servicio puede levantarse de la siguiente manera

```
su jabber
./users-agent
```

Para testear el componente basta con iniciar un cliente Jabber y chequear en la lista de servicios si user-agent esta disponible. Como muestre esta información depende del cliente pero user-agente provee servicios tanto de búsqueda como de registración.

La elección del cliente Jabber afecta el uso de este agente. Para Linux, PSI aparenta ser el de mejor respuesta; en tanto que para Windows el mejor fueron JAJC y Exodus. Exodus fue de

mejor performance de todos los testeados (PSI, Gabber, Gnome-Gabber, Kopete, Exodus, JAJC and Neos).

Users-Agent soporta correspondencia parcial en las búsquedas por lo tanto no es necesario el uso de comodines. Podría buscarse un usuario cualquiera usando sola la primer letra de cualquiera de los campos completados.

Integración de Users-Agent con los datos vCard Este procedimiento es de uso recomendado exclusivamente en privados y requiere que se este usando MySQL como repositorio de datos.

Es una corrección (patch) escrito por Peter Saint André, que le permite al servidor jabberd2 usar los datos almacenados en la tabla vCards como fuente de datos de Users-Agent. Una vez aplicado, Users-Agent usará esta tabla para búsquedas y registración. De esta manera, la actualización realizada por los usuarios en su vCard será reflejada automáticamente en las búsquedas hechas en Users-Agent JUD . De todas formas, luego de aplicada , cada usuario deberá crear una vCard para estar visible en Users-agent JUD.

Se reitera que esta corrección no sea aplicado en servidores públicos, de lo contrario estaríamos exponiendo información pública de los usuarios sin requerir su autorización. Otra aclaración realizada por el autor es que este patch no ha sido testeado en situaciones de cargas altas.

Para aplicar esta corrección, se debe levantar el componente Users-agent. Luego, descargar el archivo de corrección **`users-agent.vcard.patch`** en el directorio donde se encuentra users-agent y aplicar el parche usando:

```
patch -p0 < users-agent.vcard.patch
```

Se debe de editar el archivo **config.xml** para cambiar el nombre de la base de datos pues ahora esta usando Jabberd2, se debe de editar también el nombre de usuario y clave de ser necesario. Luego se debe de reiniciar el componente users-agent.

8.4 INSTALACIÓN DE BANDERSNATCH

Los requerimientos que presenta la versión 0.3 son los siguientes en lo que respecta al script en modo consola.

- **SQL Server DBI compatible (por defecto usa MySQL)**
- **Perl v 5.6.1 o superior**

y los siguientes módulos PERL

- **Net::Jabber versión 1.0024 o superior**
- **XML::Stream versión 1.16**
- **DBI (mysql)**
- **POE**
- **POE ::Component::Jabber**
  - **AUTHEN::DIGEST::MD5 versión 0.04 o superior**
  - **AUTHEN::SASL versión 2.0.8 o superior**
  - **DIGEST::SHA1 versión 2.10 o superior**
  - **MIME::BASE64 versión 3.0.5 o superior**
  - **NETSSLeay.pm versión 1.25 o superior**
- **POE::Filter::XML**
- **Net::Jabber**
- **PXR**

Una vez instalados las dependencias de bandersnatch, comenzamos con la instalación y configuración del componente siguiendo los pasos establecidos en el manual provisto por la distribución. A pesar de que el manual es bastante completo , lo es para la versión 1.4.x del servidor Jabberd. Información actualizada para la versión 2 se puede encontrar recorriendo los archivos de la mailing list https://sympa.funkypenguin.co.za/index.htm/arc/bandersnatch.

Dado que es un script, solo procedemos a descomprimir el tarball y luego pasamos a la configuración del mismo.

Para generar la base de datos que utiliza para almacenar los mensajes, debemos correr el script **`bandersnatch.sql`**. (Debimos editarlo para corregir errores de sintaxis en el mismo)

La configuración de el componente a nivel de jabberd2 es la siguiente:

En el **`router.xml`** hay que agregar las siguientes líneas dentro de la sección **`<aci>`**

```
<!-- These users can bind names other than their username -->
<acl type='bind'>
  <user>bandersnatch</user>
</acl>
<!-- These users can elect to receive all packets that
     pass through the router -->
<acl type='log'>
  <user>bandersnatch</user>
</acl>
```

El archivo de configuración de bandersnatch según implantado en facultad se encuentra en el Apéndice de archivos de configuración del servicio implantado en Facultad **A8**

Para iniciar este componente solo debemos ejecutar el comando **`./badernstach2.pl config.xml &`**

8.4.1 Instalación del frontend web de bandersnatch

Primero debemos instalar las dependencias del mismo, las cuales son

- PHP 4 **con opción zlib habilitada al momento de compilación.** Esto permitirá la instalación de los módulos PEAR desde los tarball directamente
- PEAR DB 1.3 (included with PHP 4.3.0)
- PEAR HTML_Template_IT 1.0.0
- PEAR Auth 1.1.1
- Servidor web con soporte para PHP

Instrucciones en cuanto a la descarga e instalación de las bibliotecas PEAR pueden ser encontradas en http://pear.php.net/manual/en/installation.php

Una vez instaladas las dependencias, se genera un alias para el directorio frontend dentro de la instalación de bandersnactch en la configuración del servidor web y se otorgan los permisos correspondiente (lectura y ejecución en todo el path.)

Luego debemos editar el archivo config.inc.php dentro del directorio frontend/includes de bandersnactch. Se deberá setear los valores adecuados de: local_server, local_domain, local-transports, etc..

Como último paso debemos editar el archivo de configuración de PHP (**`/etc/php.ini`**) y agregarle dentro de los include-path **`/usr/local/lib/php/.`**

Si el usuario admin de bandersnatch ya esta creado, se deberá cambiar la clave para que ésta este encriptada mediante MD5, a modo de ejemplo se incluye el comando SQL a ejecutar

```
mysql> update auth
    -> set PASSWORD=MD5( 'nueva_password' )
    -> where username='admin';
```

8.4.2 Correcciones en Bandersnatch

Corrección en fuente Perl para solucionar problemas de doble entrada de log de mensajes

.....

```
# Function   : log_message
# Purpose    : Log "message" type nodes
##############################################*/
sub log_message
{
 my $node = shift;
 my %message;
 my $sqlquery;
 my $is_local = 0;
 my $header = $node->get_children()->[0];
 my $body = $header->get_children()->[0];

 return if (!$body->data()); # Don't log empty messages, or "non-<message>" messages :)
  return if ($header->attr('sc:sm')); # Para no duplicar mensajes. (Antes era !$header->attr('sc:sm'))
```

return if ($header->attr('sm:sm')); # (se implementa lo mismo para mensajes con conversaciones a travès de Gateways externos).

```
$message{'to'} = $header->attr('to');
$message{'from'} = $header->attr('from');
$message{'type'} = $header->attr('type');
$message{'subject'} = $header->attr('subject');
$message{'thread'} = $header->attr('thread');
$message{'error'} = $header->attr('error');
$message{'errorcode'} = $header->attr('errorcode');
$message{'body'} = $body->data();
```

.....

8.5 LISTA DE SERVIDORES BASADOS EN JABBER/XMPP PÚBLICOS

Ultima actualización 26/01/2005

La disponibilidad es calculada en base a la cantidad de pings exitosos. Un ping es enviado cada 20 minutos

Servidor	Disponibilidad	Versión	OS
jabber.org	97.1%	jabberd 1.4CVS	Linux 2.4.21-4-686-smp
12jabber.com	92.0%	jabberd 1.4.2	Linux 2.4.10-4GB
akl.lt	99.1%	- -	-
amessage.at	95.6%	jabberd 1.4.3+ipv6	Linux 2.6.8-1-k7
amessage.be	73.4%	jabberd 1.4.3+ipv6	Linux 2.6.8-1-k7
amessage.ch	93.5%	jabberd 1.4.3+ipv6	Linux 2.6.8-1-k7
amessage.de	96.8%	jabberd 1.4.3+ipv6	Linux 2.6.8-1-k7
amessage.info	95.8%	jabberd 1.4.3+ipv6	Linux 2.6.8-1-k7
amessage.li	94.0%	jabberd 1.4.3+ipv6	Linux 2.6.8-1-k7
bloodyxml.com	99.3%	jabberd 1.4.3	Linux 2.6.8.1
borderlinenormal.com	99.3%	jabberd 1.4.3	Linux 2.6.8.1
bulmalug.net	98.4%	- -	-
default.co.yu	95.2%	jabberd 1.4.3	FreeBSD 5.2.1-RELEASE-p9
dhbit.ca	69.5%	jabberd 1.4CVS	Linux 2.6.9
dotgeek.org	98.0%	jabberd 1.4.3	Linux 2.4.25
eifelansinn.net	0.0%		
es.tipic.com	96.7%	TIMP(?)	TIMP NT
geodude.timmins.net	0.0%		
here.dk	99.7%	- -	-
illx.org	97.5%	jabberd 1.4.2	Linux 2.6.5
jabber-fr.net	99.3%	jabberd 1.4.2	Linux 2.4.18-bf2.4

jabber.a.cc.pl	0.0%		
jabber.anywise.com	92.0%	jabberd 1.4.2	Linux 2.4.10-4GB
jabber.at	98.5%	- -	-
jabber.bettercom.de	93.0%	jabberd 1.4CVS	Linux 2.4.19-686
jabber.carter.to	99.7%	jabberd 1.4.2	Linux 2.4.5
jabber.cn	92.3%	jabberd 1.4.2	Linux 2.4.10-4GB
jabber.com	76.4%	XCP jabber2 4.1.0.3	SunOS 5.8
jabber.cz	99.1%	WPJabber 1.1.3	Linux 2.4.20-8smp
jabber.daemon.sh	0.0%		
jabber.dk	98.5%	jabberd 1.4.3	Linux 2.4.25-1-686-smp
jabber.earth.li	96.2%	- -	-
jabber.freenet.de	99.7%	jabberd 1.4.3	Linux 2.6.7
jabber.gda.pl	97.7%	- -	-
jabber.hu	95.2%	jabberd 1.4.3.1	Linux 2.4.27
jabber.kelkoo.net	0.0%	jabberd 1.4.3	Linux 2.4.22-1.2199.nptl
jabber.kicks-ass.org	0.0%		
jabber.linux.it	97.5%	jabberd 1.4.3	Linux 2.6.5
jabber.mailclean.net	99.3%	jabberd 1.4.3	Linux 2.4.18-bf2.4
jabber.mandog.com	0.0%		
jabber.netsample.com	99.8%	jabberd 2.0s1	Linux 2.4.22-10mdksecure
jabber.ngnet.it	0.0%		
jabber.or.id	87.8%	jabberd 1.4.2	Linux 2.4.22-openmosix
jabber.or.kr	0.0%		
jabber.org	97.1%	jabberd 1.4CVS	Linux 2.4.21-4-686-smp
jabber.palomine.net	99.5%	jabberd 1.4.3.1	FreeBSD 5.3-RELEASE-p2
jabber.probonic.com	97.3%	jabberd 1.4.3	Linux 2.6.7-probonic1-skas-sysemu
jabber.projecto-oasis.cx	98.3%	jabberd 1.4.2	Linux 2.2.24-7.0.3

jabber.ro	85.8%	jabberd 1.4.2	Linux 2.6.7
jabber.ru	98.7%	ejabberd 0.8-alpha	unix/freebsd 5.2.0
jabber.sk	95.5%	jabberd 1.4.3	Linux 2.4.20-20.9
jabber.skynet.be	90.6%	jabberd 2.0s4	Linux 2.4.26-skynet
jabber.snc.ru	95.8%	ejabberd 0.8-alpha	unix/linux 2.6.7
jabber.toble.com	0.0%	jabberd 2.0s3	Linux 2.4.26
jabber.unesco.kz	98.7%	jabberd 1.4.2	Linux 2.4.27
jabber.wiretrip.org	98.8%	WPJabber 1.1.4	Linux 2.6.8
jabber.wp.pl	99.2%	WPJabber 1.1.4	Linux 2.4.17-epoll.warf
jabber.zim.net.au	98.4%	jabberd 1.4.3	FreeBSD
jabberes.org	98.8%	ejabberd 0.8-alpha	unix/linux 2.4.25
jabbernet.dk	99.2%	- -	-
jabberpl.org	96.5%	jabberd 1.4.3	Linux 2.6.7-1-386
kanga.nu	0.0%		
mundo-chat.org	0.0%	jabberd 1.4.3.1-ipv6	Linux 2.6.3-4mdk
myjabber.net	98.8%	WPJabber 1.1.3	Linux 2.4.20-28.9
nabla.net	0.0%		
nedbsd.nl	99.4%	jabberd 1.4.3	OpenBSD 3.5
nedlinux.nl	99.5%	jabberd 1.4.3	Linux 2.4.26-gentoo-r9
netmindz.net	95.5%	jabberd 1.4.3	Linux 2.6.9-1.681_FC3smp
njs.netlab.cz	98.4%	jabberd 1.4.3+	Linux 2.6.8-1-686-smp
nureality.ca	76.5%	jabberd 1.4.2	Linux 2.6.9-1.667
oracle.sweeney.ath.cx	97.2%	jabberd 1.4.3	Linux 2.4.18-17.7.xsmp
pan.ubishops.ca	95.3%	jabberd 1.4.3	Linux 2.4.27-sparc
rhymbox.com	0.0%		
sourcecode.de	0.0%		

swissjabber.ch	99.3%	jabberd 1.4.3+ipv6	Linux 2.4.27
swissjabber.li	46.1%	jabberd 1.4.3+ipv6	Linux 2.4.27
syndicon.de	88.1%	jabberd 1.4.3+	Linux 2.6.8-1-686
theoretic.com	99.5%	jabberd 1.4.3-ipv6	Linux 2.4.25_pre6-gss
tipic.com	96.8%	TIMP(?)	TIMP NT
unstable.nl	90.8%	ejabberd 0.7.5	unix/linux 2.6.9

8.6 SOBRE TRANSFERENCIA DE ARCHIVOS

Se decidió no instalar ningún componente adicional para resolver la trasferencia de archivos. De todas maneras, se constato que usando los mismos clientes en ambas puntas de la "conversación", la transferencia de archivos se resuelve sin problemas. El problema de posibles virus en los archivos que se transfieren debe ser controlado por software instalado en el puesto de trabajo, ya que en este caso como la transferencia es punto a punto, el servidor no puede realizar ningún tipo de chequeo.

8.7 SOBRE COMUNICACIÓN CON OTROS SISTEMAS DE MENSAJERÍA

Existen varios gateways implementados para:

- AIM
- MSN
- ICQ
- Yahoo
- SMS
- SMTP

En el caso de los transportes para mensajería instantánea de protocolos propietarios, el usuario debe de tener registrada una cuenta en dicho servicio.

Se realizaron pruebas con el gateway desarrollado en Python para MS Messenger y no se tuvieron mayores problemas, si el servidor cuenta con Service Discovery, se le presentarán al usuario los Gateways disponibles instalados en el servidor.

8.8 ARCHIVOS DE CONFIGURACIÓN DE SERVIDOR IMPLANTADO EN FACULTAD DE INGENIERÍA

A continuación presentaremos las modificaciones de los archivos de configuración de Jabberd 2.0, MU-Conference y Bandersnatch según instalación en equipo de prueba de Facultad de Ingeniería

8.8.1 Jabberd

8.8.1.1 Componente C2S

```
<!-- c2s configuration -->
<c2s>
  <id>c2s</id>
  <pidfile>/usr/local/var/jabberd/pid/c2s.pid</pidfile>

  <!-- Router connection configuration -->
  <router>
    <ip>127.0.0.1</ip>
    <port>5347</port>
    <!-- Username/password to authenticate as -->
    <user>jabberd</user>
    <pass>secret</pass>
    <pemfile>/usr/local/etc/jabberd/server.pem</pemfile>

    <!-- Router connection retry -->
    <retry>
      <init>3</init>
      <lost>3</lost>
      <sleep>2</sleep>
    </retry>
  </router>

  <!-- Log configuration - type is "syslog", "file" or "stdout" -->
  <log type='file'>
    <ident>jabberd/c2s</ident>
    <facility>local3</facility>
    <file>/usr/local/var/jabberd/log/c2s.log</file>
  </log>

  <!-- Local network configuration -->
  <local>
       <id>tortuga.fing.edu.uy</id>
    <ip>0.0.0.0</ip>
    <port>5222</port>
    <pemfile>/usr/local/etc/jabberd/server.pem</pemfile>
    <ssl-port>5223</ssl-port>
  </local>

  <!-- Input/output settings -->
  <io>
    <max_fds>1024</max_fds>
    <!-- Rate limiting -->
    <limits>
      <bytes>0</bytes>
      <connects>0</connects>
    </limits>

    <access>
      <order>allow,deny</order>
```

```
    </access>

    <!-- Timed checks -->
    <check>
      <interval>0</interval>
      <idle>0</idle>
      <keepalive>0</keepalive>
    </check>

  </io>

  <!-- Authentication/registration database configuration -->
  <authreg>
    <!-- Backend module to use -->
    <module>mysql</module>
    <!-- Registration configuration -->
    <register>
          <enable/>
        <instructions>Enter a username and password to register with
this server.</instructions>
       <password/>
    </register>

    <!-- Available authentication mechanisms -->
    <mechanisms>
      <traditional>
        <plain/>
        <digest/>
        <zerok/>
      </traditional>
      <sasl>
        <plain/>
        <digest-md5/>
      </sasl>

    </mechanisms>

    <!-- MySQL module configuration -->
    <mysql>
      <!-- Database server host and port -->
      <host>localhost</host>
      <port>3306</port>
      <!-- Database name -->
      <dbname>jabberd2</dbname>
      <!-- Database username and password -->
      <user>jabberd2</user>
      <pass>jabberd2</pass>
    </mysql>

    <!-- Pipe module configuration -->
    <pipe>
      <!-- Program to execute -->
      <exec>/usr/local/bin/pipe-auth.pl</exec>
    </pipe>
  </authreg>
</c2s>
```

8.8.1.2 Componente SM

```
<!-- Session manager configuration -->
<sm>
  <id>tortuga.fing.edu.uy</id>
  <pidfile>/usr/local/var/jabberd/pid/sm.pid</pidfile>

  <!-- Router connection configuration -->
  <router>
    <!-- IP/port the router is waiting for connections on -->
    <ip>127.0.0.1</ip>
    <port>5347</port>
    <!-- Username/password to authenticate as -->
    <user>jabberd</user>
    <pass>secret</pass>
    <pemfile>/usr/local/etc/jabberd/server.pem</pemfile>
    <!-- Router connection retry -->
    <retry>
      <init>3</init>
      <lost>3</lost>
      <sleep>2</sleep>
    </retry>
  </router>

  <!-- Log configuration - type is "syslog", "file" or "stdout" -->
  <log type='file'>
    <ident>jabberd/sm</ident>
    <facility>local3</facility>
    <file>/usr/local/var/jabberd/log/sm.log</file>
  </log>

  <!-- Storage database configuration -->
  <storage>
    <driver>mysql</driver>

    <!-- MySQL driver configuration -->
    <mysql>
      <host>localhost</host>
      <port>3306</port>
      <dbname>jabberd2</dbname>
      <user>jabberd2</user>
      <pass>jabberd2</pass>
      <transactions/>
    </mysql>
  </storage>

  <!-- Access control information -->
  <aci>
    <acl type='all'>
      <jid>admin@tortuga.fing.edu.uy</jid>
    </acl>
  </aci>

  <modules>
    <chain id='sess-start'/>

    <chain id='sess-end'>
      <module>iq-last</module>
    </chain>

    <chain id='in-sess'>
      <module>validate</module>
```

```
    <module>privacy</module>
    <module>roster</module>
    <module>vacation</module>
    <module>iq-vcard</module>
    <module>iq-private</module>
    <module>disco</module>
    <module>offline</module>
    <module>announce</module>
    <module>presence</module>
    <module>deliver</module>
  </chain>

  <chain id='out-sess'/>

  <chain id='in-router'>
    <module>session</module>
    <module>validate</module>
    <module>presence</module>
    <module>privacy</module>
  </chain>

  <chain id='out-router'>
    <module>privacy</module>
  </chain>

  <chain id='pkt-sm'>
    <module>iq-last</module>
    <module>iq-time</module>
    <module>iq-version</module>
    <module>disco</module>
    <module>announce</module>
    <module>help</module>
    <module>echo</module>
  </chain>

  <chain id='pkt-user'>
    <module>roster</module>
    <module>presence</module>
    <module>iq-vcard</module>
    <module>deliver</module>
    <module>vacation</module>
    <module>offline</module>
    <module>disco-publish</module>
    <module>iq-last</module>
  </chain>

  <chain id='pkt-router'>
    <module>session</module>
    <module>disco</module>
  </chain>

  <chain id='user-load'>
    <module>active</module>
    <module>roster</module>
    <module>privacy</module>
    <module>disco-publish</module>
    <module>vacation</module>
  </chain>

  <chain id='user-create'>
    <module>active</module>
    <module>template-roster</module>
  </chain>
```

```
    <chain id='user-delete'>
      <module>active</module>
      <module>announce</module>
      <module>disco-publish</module>
      <module>offline</module>
      <module>privacy</module>
      <module>roster</module>
      <module>vacation</module>
      <module>iq-last</module>
      <module>iq-private</module>
      <module>iq-vcard</module>
    </chain>

  </modules>

  <!-- Service discovery configuration -->
  <discovery>

    <identity>
      <category>server</category>
      <type>im</type>
      <name>Jabber M.I. server</name>
    </identity>

    <agents>
      <agent jid='conference.tortuga.fing.edu.uy'>
        <name>Public Conferencing</name>
        <description>Public chatrooms for users.</description>
        <service>public</service>
        <groupchat/>
      </agent>
    </agents>

    <items>
    </items>

  </discovery>

  <!-- User options -->
  <user>
    <auto-create/>

    <!-- Templates. If defined, the contents of these files will be
         stored in the users data store when they are created. -->
    <template>
      <!--
      <roster>/usr/local/etc/jabberd/templates/roster.xml</roster>
      -->
    </template>
  </user>

</sm>
```

8.8.1.3 Componente Router

```
<!-- Router configuration -->
<router>
  <!-- ID of the router on the network (default: router) -->
  <id>router</id>
  <pidfile>/usr/local/var/jabberd/pid/router.pid</pidfile>
  <!-- Log configuration - type is "syslog", "file" or "stdout" -->
  <log type='file'>
    <!-- If logging to syslog, this is the log ident -->
    <ident>jabberd/router</ident>
    <facility>local3</facility>
    <!-- If logging to file, this is the filename of the logfile -->
    <file>/usr/local/var/jabberd/log/router.log</file>
  </log>

  <!-- Local network configuration -->
  <local>
    <ip>0.0.0.0</ip>
    <port>5347</port>
    <users>/usr/local/etc/jabberd/router-users.xml</users>
    <secret>secret</secret>
    <pemfile>/usr/local/etc/jabberd/server.pem</pemfile>
  </local>

  <!-- input/output settings -->
  <io>
    <max_fds>1024</max_fds>
    <limits>
      <bytes>0</bytes>
      <connects>0</connects>
    </limits>
    <access>
      <order>allow,deny</order>
    </access>
  </io>

  <aliases>
        <alias   name='conference.tortuga.fing.edu.uy'   target='mu-
conference'/>
  </aliases>

  <!-- Access control information -->
  <aci>
    <acl type='all'>
      <user>jabberd</user>
    </acl>

    <!-- These users can bind names other than their username -->

    <acl type='bind'>
      <user>bandersnatch</user>
    </acl>

    <acl type='log'>
      <user>bandersnatch</user>
    </acl>
  </aci>

</router>
```

8.8.1.4 Componente Router-Users

```
<!-- This is the list of known router users, and their authentication
      secrets. Access control is done via the settings in router.xml
-->
<users>
  <user>
    <name>jabberd</name>
    <secret>secret</secret>
  </user>
  <user>
    <name>muc</name>
    <secret>secret</secret>
  </user>
  <user>
        <name>bandersnatch</name>
        <secret>bandersnatch</secret>
  </user>
</users>
```

8.8.1.5 Componente S2S

```
<!-- s2s configuration -->
<s2s>
  <!-- Our ID on the network (default: s2s) -->
  <id>s2s</id>
  <pidfile>/usr/local/var/jabberd/pid/s2s.pid</pidfile>
  <router>

    <ip>127.0.0.1</ip>
    <port>5347</port>
    <user>jabberd</user>
    <pass>secret</pass>
    <pemfile>/usr/local/etc/jabberd/server.pem</pemfile>
    <retry>
      <init>3</init>
      <lost>3</lost>
      <sleep>2</sleep>
    </retry>
  </router>

  <log type='file'>
    <ident>jabberd/s2s</ident>
    <facility>local3</facility>
    <file>/usr/local/var/jabberd/log/s2s.log</file>
  </log>

  <!-- Local network configuration -->
  <local>
    <ip>0.0.0.0</ip>
    <port>5269</port>
    <resolver>resolver</resolver>
  </local>

  <!-- Timed checks -->
  <check>
    <interval>0</interval>
    <queue>0</queue>
    <invalid>0</invalid>
    <keepalive>0</keepalive>
  </check>
</s2s>
```

8.8.2 Bandersnatch

```
<config>
      <server>
         <connectiontype>tcpip</connectiontype>
         <hostname>tortuga.fing.edu.uy</hostname>
         <port>5347</port>
         <secret>bandersnatch</secret>
      </server>
      <component>
         <name>bandersnatch</name>
      </component>
      <mysql>
         <server>localhost</server>
         <dbname>bandersnatch</dbname>
         <username>bandersnatch</username>
         <password>bandersnatch</password>
      </mysql>
      <debug>
         <level>0</level>
         <file>stdout</file>
      </debug>
      <site>
         <local_server>tortuga.fing.edu.uy</local_server>
         <admin_jids>admin@tortuga.fing.edu.uy</admin_jids>

<confidential_jids>theboss@tortuga.fing.edu.uy</confidential_jids>
         <ignore_jids>chatbot@tortuga.fing.edu.uy</ignore_jids>
         <ignore_jids>bandersnatch@tortuga.fing.edu.uy</ignore_jids>
                                    <ignore_jids>headlines.jabber.lab-
inco.fing.edu.uy</ignore_jids>

<local_domains>conference.tortuga.fing.edu.uy</local_domains>

<local_domains>bandersnatch.tortuga.fing.edu.uy</local_domains>
         <privacy>1</privacy>
         <aggressive_presence>0</aggressive_presence>
      </site>
</config>
```

8.8.3 MU-Conference

```
<jcr>
  <!--
       This is a config file for a copy of MU-Conference, compiled
       against the Jabber Component Runtime (JCR). This is the same
       file that I use to connect to my development server, running
       jabberd2 beta2
       In order to connect to a jabberd v1.4 server, simply change
       the <name> value to muclinker, and make sure the muclinker
       section is in your main jabber.xml file, as per the MU-
       Conference README file.
  -->

  <name>mu-conference</name>
  <host>conference.tortuga.fing.edu.uy</host>

  <ip>192.168.32.50</ip>

  <port>5347</port>
  <user>muc</user>
  <secret>secret</secret>

  <spool>
    /usr/local/var/jabberd/spool/conference.tortuga.fing.edu.uy
  </spool>
  <logdir>/usr/local/var/jabberd/log</logdir>
  <pidfile>/usr/local/var/jabberd/pid/mu-conference.pid</pidfile>
  <!--
     <logstderr/>
  -->
  <loglevel>124</loglevel>

    <conference xmlns="jabber:config:conference">
      <public/>
      <vCard>
        <FN>Public Chatrooms</FN>
        <DESC>This service is for public chatrooms.</DESC>
        <URL>http://foo.bar/</URL>
      </vCard>
      <history>40</history>
      <logdir>/usr/local/var/jabberd/log</logdir>
      <notice>
        <join>has become available</join>
        <leave>has left</leave>
        <rename>is now known as</rename>
      </notice>
      <sadmin>
        <user>admin@tortuga.fing.edu.uy</user>
      </sadmin>
    </conference>
</jcr>
```

8.8.3.1 Ejemplo de Seteo de Salas de Conferencia en MU-Conference

Contenido del archivo 2402f2f44290e8505ddac91e3a0aad89e6ed079a.xml.

```
<xdb>
  <list xdbns="muc:list:owner">
    <item jid="mario@tortuga.fing.edu.uy" />
  </list>
  <list xdbns="muc:list:admin">
    <item jid="mario@tortuga.fing.edu.uy" />
  </list>
  <list xdbns="muc:list:member">
    <item jid="mario@tortuga.fing.edu.uy" />
    <item jid="mvd1304@tortuga.fing.edu.uy" />
  </list>
  <list xdbns="muc:list:outcast">
  </list>
  <room xdbns="muc:room:config">
    <name>Chat1</name>
    <creator>mario@tortuga.fing.edu.uy</creator>
    <defaulttype>0</defaulttype>
    <description>Sala 1 de prueba - creado por mario</description>
    <invitation>1</invitation>
    <invites>0</invites>
    <legacy>1</legacy>
    <logformat>0</logformat>
    <logging>0</logging>
    <maxusers>20</maxusers>
    <moderated>1</moderated>
    <notice>
      <join>join</join>
      <leave>leave</leave>
      <rename>rename</rename>
    </notice>
    <persistent>1</persistent>
    <private>0</private>
    <privmsg>0</privmsg>
    <public>0</public>
    <secret>pwdchat1</secret>
    <subject>proyecto GNU/Linux</subject>
    <subjectlock>0</subjectlock>
    <visible>0</visible>
  </room>
</xdb>

Contenido del archivo rooms.xml

<xdb>
  <registered xdbns="muc:room:list">
    <item name = "chat1@tortuga.fing.edu.uy"
                                                    jid          =
"2402f2f44290e8505ddac91e3a0aad89e6ed079a@tortuga.fing.edu.uy" />
  </registered>
</xdb>
```

Estos dos archivos pueden usarse como plantilla para la generación manual de otras salas con nombres más comprensibles. Por cada sala creada deberá agregarse a **room.xml** otra sección **<registered>** del tipo

```
  <registered xdbns="muc:room:list">
    <item name = "nombreSala@jabber.fing.edu.uy"
          jid  = "nombreSala@jabber.fing.edu.uy" />
  </registered>
```

8.8.4 WebMessenger

```
<?php

// Enter your MySQL database settings.
define("DB_HOST","localhost");
define("DB_USER","webmassenger");
define("DB_PASS","xxxxxxxxxxxx");
define("DB_NAME","webmessenger");


define("JABBER_SERVER","tortuga.fing.edu.uy");
define("JABBER_PORT","5222");


define("TRANSPORT_MSN","");
define("TRANSPORT_ICQ","");
define("TRANSPORT_AIM","");
define("TRANSPORT_YIM","");

define("JABBER_RUNNING","ps ax | grep jabberd");
define("MSN_RUNNING","ps ax | grep msntrans");
define("ICQ_RUNNING","ps ax | grep icqv7ext");
define("YIM_RUNNING","");
define("AIM_RUNNING","");

define("JABBER_RESOURCE","WebMsgr");

$font_family = "Tahoma,Verdana,Arial,Helvetica,sans-serif";
$font_size = 8;

$theme = "default";

define("DEFAULT_LANGUAGE","english");


define("ALLOW_EXISTING_ACCOUNTS",true);


define("DEBUG_MODE",0);
?>
```

8.8.5 Herramienta Jabbsimul

Para prueba de estabilidad

```
<dpsm>

  <events_after>1</events_after>

  <user_names_generator>
    <range>
      <from>10</from>
      <to>60</to>
    </range>
    <name>test%(num:u)</name>
    <server>tortuga.fing.edu.uy</server>
  </user_names_generator>

  <user_properities>
    <filter><name>test.*</name></filter>
    <properities>
     <fullname>test%(num*2:u)-%(3+num%12000/(1+3)+7:u)</fullname>
      <password>password</password>
      <resource>stressingser</resource>
      <Xhost>tortuga.fing.edu.uy</Xhost>
      <!-- <sniff>/usr/local/var/jabberd/log/</sniff> -->

      <event>
        <name>connect</name>
        <frequency>60000</frequency>
        <counter>2000</counter>
        <digest/>
      </event>

      <event>
        <name>add_roster</name>
        <frequency>150000</frequency>
        <user><range><from>1</from><to>50</to></range></user>
        <max_roster_count>20</max_roster_count>
      </event>

      <event>
        <name>del_roster</name>
        <frequency>150000</frequency>
      </event>

      <event>
        <name>send_message</name>
        <frequency>7000</frequency>
        <user><range><from>10</from><to>60</to></range></user>
        <prepend_with_debug_info/>
             <text>Mensaje de envío de prueba para simular carga
stress</text>
      </event>

      <event>
        <name>change_status</name>
        <frequency>30000</frequency>
        <status>Jezdem tera dostepny</status>
        <show>chat</show>
      </event>
```

```
      <Xevent>
        <name>logout</name>
        <frequency>60000</frequency>
      </Xevent>

      <Xevent>
        <name>kill_connection</name>
        <frequency>120000</frequency>
      </Xevent>

      <Xevent>
        <name>send_raw_bytes</name>
        <frequency>300000</frequency>
        <random_stream len="1000"/>
      </Xevent>

    </properities>
  </user_properities>
</dpsm>
```

Configuración para prueba de overhead de Log conversacional

```
<dpsm>

  <events_after>1</events_after>

  <user_names_generator>
    <range>
      <from>10</from>
      <to>60</to>
    </range>
    <name>test%(num:u)</name>
    <server>tortuga.fing.edu.uy</server>
  </user_names_generator>

  <user_properities>
    <filter><name>test.*</name></filter>
    <properities>
     <fullname>test%(num*2:u)-%(3+num%12000/(1+3)+7:u)</fullname>
      <password>password</password>
      <resource>stressingser</resource>
      <Xhost>tortuga.fing.edu.uy</Xhost>
      <!-- <sniff>/usr/local/var/jabberd/log/</sniff> -->

      <event>
        <name>connect</name>
        <frequency>60000</frequency>
        <counter>2000</counter>
        <digest/>
      </event>

      <event>
        <name>add_roster</name>
        <frequency>150000</frequency>
        <user><range><from>1</from><to>50</to></range></user>
        <max_roster_count>20</max_roster_count>
      </event>

      <event>
```

```
        <name>del_roster</name>
        <frequency>150000</frequency>
      </event>

      <event>
        <name>send_message</name>
        <frequency>15000</frequency>
        <user><range><from>10</from><to>60</to></range></user>
        <prepend_with_debug_info/>
             <text>Mensaje de envío de prueba para simular carga
stre</text>
      </event>

      <event>
        <name>change_status</name>
        <frequency>30000</frequency>
        <status>Jezdem tera dostepny</status>
        <show>chat</show>
      </event>

      <Xevent>
        <name>logout</name>
        <frequency>60000</frequency>
      </Xevent>

      <Xevent>
        <name>kill_connection</name>
        <frequency>120000</frequency>
      </Xevent>

      <Xevent>
        <name>send_raw_bytes</name>
        <frequency>300000</frequency>
        <random_stream len="1000"/>
      </Xevent>

    </properities>
  </user_properities>
</dpsm>
```

8.9 INSTALACIÓN DE MÓDULOS PERL

En general todos los módulos perl podrán ser obtenidos desde el sitio web http://www.cpan.org. Se recomienda tener el cuidado de comprobar las dependencias de los mismos antes de bajarlos.

```
tar -zxvf modulo.tar.gz
cd modulo
perl Makefile.PL
make
make test
make install
```

Si posee conexión a internet podrá utilizar los utilitarios provistos por PERL para instalar módulos online.

La manera más fácil para instalar los módulos CPAM es la de utilizar la herramienta de instalación CPAN.pm, la forma de utilización es la siguiente

```
perl -M CPAN -e shell
```

Luego de ingresar en el shell de cpan de debe instalar con el comando install y el modulo a instalar Ej (módulos prerequisitos para JUD)

```
cpan> install Net::Jabber
cpan> install XML::Stream
cpan> install DBI
```

8.10 TARIFAS ANTEL VIGENTES A FEBRERO DE 2005

Tarifa Local

Se aplica a llamadas entre servicios telefónicos ubicados en:

- La misma localidad
- Area Metropolitana de Montevideo
- Localidades de un mismo departamento
- Localidades de distintos Departamentos a distancias menores o iguales a los 60 km.
- Las siguientes localidades fronterizas: Artigas con Quaraí, Bella Unión con Quaraí. Chuy con Chui, Lago Merin con Yaguarón, Río Branco con Yaguarón y Rivera con Livramento.

COMUNICACIONES DE TELEFONO FIJO A TELEFONO FIJO

Valor del cómputo $ 1,09

Cadencia para la aplicación de los cómputos urbanos:

Lunes a Viernes

de 00:00 a 09:00 y de 21:00 a 24:00	1 cómputo c/6 minutos
de 09:00 a 11:00 y de18:00 a 21:00	1 cómputo c/3 minutos
de 11:00 a 18:00	1 cómputo por minuto

Sábados, Domingos y Feriados(1/1, 1/5, 18/7, 25/8, 25/12)

de 00:00 a 24:00	1 cómputo c/6 minutos

Larga Distancia Nacional

SE APLICA SOLAMENTE A LAS LLAMADAS ENTRE SERVICIOS TELEFONICOS UBICADOS EN LOCALIDADES DE DISTINTOS DEPARTAMENTOS A DISTANCIAS MAYORES A 60 KM.

COMUNICACIONES DE TELEFONO FIJO A TELEFONO FIJO POR MINUTO

- LUNES A VIERNES	$3,23 (Tarifa Unica Nacional)
- SABADOS, DOMINGOS	$1,25 (Uruguay Fin de Semana)

Larga Distancia Internacional

LLAMADAS DE LARGA DISTANCIA INTERNACIONAL TELEDISCADA	Tarifa p/min.
Ciudades: Buenos Aires, Porto Alegre, San Pablo, Santiago, Asunción	$4.34
Zona I: Resto Argentina, resto Brasil, resto Chile, Estados Unidos, España, Canadá, Italia, Israel	$ 4,94
Zona II: Francia, Reino Unido, Bolivia, Ecuador, México, Perú, Alemania, Suiza, resto Paraguay	$8,04
Zona III: Costa Rica, El Salvador, Guatemala, Honduras, Nicaragua, Panamá, Puerto Rico y Rep. Dominicana, Venezuela, Colombia	$9,04
Zona IV: Resto del Mundo	$16,04
Zona V: Insular (Ascención, Cook Is., Cuba, Kiribati, Malvinas, Nauru)	$30,75
Tráfico fronterizo Depto. Colonia - Buenos Aires	$4.34

8.11 DESCRIPCIÓN DETALLADA DE PROTOCOLO XMPP

8.11.1 XMPP Core y Arquitectura XMPP

Aunque la funcionalidad de entrega de mensajes y presencia ya existía en el protocolo Jabber original, las extensiones principales de XMPP incluyen funcionalidades de autenticación, seguridad, privacidad y control de acceso según dictamina el RFC2779.

XMPP también soporta el Modelo de M.I. y Presencia del RFC2778 (A Model for Presence and Instant Messaging). También cubre problemas de localización e internacionalización y además extiende las características de los RFC2778 y RFC2779 para, por ejemplo, lograr chatrooms.

La arquitectura XMPP consiste en servidores XMPP, clientes XMPP y gateways para sistemas externos (no-XMPP), cada servidor entrega mensajes a través de una red de servidores XMPP y/o Gateways hacia los clientes, adicionalmente al servicio de ruteo de mensajes y de administración de conexiones, los servidores XMPP ofrecen a los clientes servicios de almacenamiento de información inherentes al cliente como ser listas de contactos (llamados Roosters en XMPP).

La conexión entre servidores (o componentes de servidores), gateways y clientes es sobre TCP/IP, (aunque no es un requerimiento necesario el de utilizar mensajería sobre TCP/IP, es el único método de transporte que hace referencia la especificación).

Los Gateways traducen XMPP al protocolo de dicha red de mensajería externa y también la información de dicha red externa a XMPP. Estos protocolos no-XMPP pueden ser por ejemplo IRC (Internet Relay Chat RFC1459), SMS (Short Message Service), SIMPLE, SMTP, AIM, ICQ, MSN, Yahoo, etc.

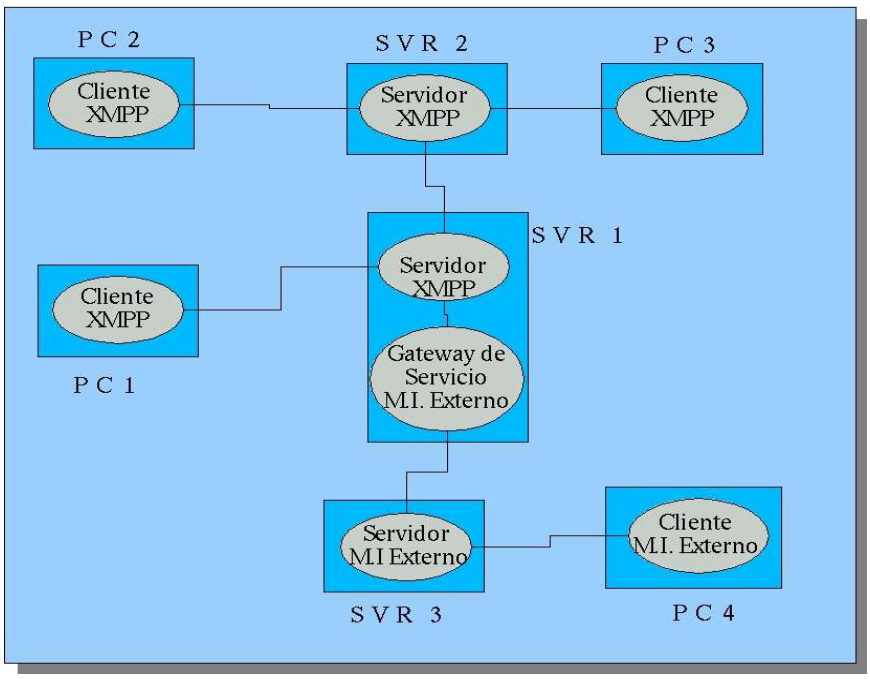


Figura 3: Arquitectura XMPP – Nota: si bien en la figura vemos que el servicio de Gateway corre dentro de uno de los servidores donde se encuentra un servidor XMPP, el gateway puede encontrarse también en otro equipo

8.11.1.1 Modo de referenciamiento de entidades

XMPP define el concepto de una entidad alcanzable en cualquier punto final de una red XMPP llamada JID (Jabber ID) la cual obtiene su nombre del protocolo original Jabber, está compuesto por un nodo, un dominio y un recurso, el uso típico es para representar usuarios o chatrooms:

```
jid  = < nodo"@" dominio"/" recurso>

Ej:   edemberg_gutierrez@xmpp-srv-01.fing.edu.uy/GAIM
      chatroom_consultas_IO@dpto_IO.fing.edu.uy
```

8.11.1.2 XMPP Streams (Flujos de datos XML)

XMPP se construye en base a 2 conceptos fundamentales: XML Streams y XML Stanzas (elementos XMPP). La idea bajo XML Streams es el que en vez de enviar documentos XML separados en una conexión, se crea una especie de conexión persistente entre dos entidades y por donde se envían elementos en forma de XML Stanzas, los streams son unidireccionales, en caso de que se requiera una conexión bidireccional se utiliza otro XML Stream.

Los XML Streams son comenzados por el tag **<stream>**, una vez que se establece el flujo, se esta listo para enviar las XML Stanzas, para cerrar el flujo se utiliza el tag **</stream>**

Un ejemplo de conversación unidireccional entre dos nodos A y B (cliente y servidor por ejemplo) sería :

1-A abre conexión TCP hacia B

2-A inicia el flujo XMPP con **<stream>**

3-A envía elementos XMPP por el flujo

4-A cierra flujo XMPP con **</stream>**

5-A cierra conexión TCP con B

Si se requiere una respuesta de B, se debe abrir un flujo de XMPP de B hacia a, se utiliza la misma conexión TCP.

En caso de problemas en los Streams se aplican las siguientes reglas:

Cualquier error a nivel de stream se considera como irrecuperable y cuando se es detectado un error la entidad que detecta dicho error es responsable de enviar un elemento **<error></error>** y luego inmediatamente **</stream>** y finalmente cerrar la conexión

Si el error sucede cuando se quiere establecer el flujo de deben enviar los siguientes tags : **<stream> – <error></error>** (el elemento error contiene puede contener un tag **<text></text>** que tenga la descripción del error) – **<stream/>** y luego finalizar la conexión TCP.

8.11.1.3 XML Stanzas

Luego que el flujo fuera comenzado empieza el intercambio de XML stanzas

XMPP : core predefine tres **XML Stanzas**:

- message
- presence
- iq (info/query)

Todas las Stanzas comparten un conjunto de atributos comunes:

`to:`
especifica hacia que JID debe ir la Stanza

`from:`
especifica el JID del que envía la Stanza (si el cliente quiere utilizar un JID falso en el atributo from o el stream de dicho JID no ha sido autenticado aún, el servidor debe generar un error)

`id:`
atributo opcional que puede ser utilizado para el rastreo de XML Stanzas en un flujo dado

`type:`
especifica información detallada acerca del propósito de la Stanza

`xml:lang:`
atributo opcional que especifica el lenguaje por defecto del contenido en las Stanzas

Se describen a continuación las tres Stanzas:

Message Stanza: es utilizada para enviar mensajes de una entidad XMPP a otra, por ejemplo, un cliente XMPP utiliza una Message Stanza para enviar un mensaje instantáneo a otro cliente XMPP a través de uno o más servidores XMPP, el servidor coteja el atributo **`'to'`** de la Stanza y rutea apropiadamente el elemento, en caso de no encontrar una ruta válida , el servidor debe enviar un mensaje de error al cliente que generó el mensaje.

Presence Stanza: se considera como una forma básica de broadcast desde una entidad a múltiples recipientes subscritos, generalmente el atributo **`'to'`** no se utiliza, sin embargo si quien envía la Stanza especifica información en el campo **`'to'`**, el servidor debe entregar la Stanza de presencia solo al receptor especificado.

Info/Query Stanza: es una interacción en la cual una entidad pide información a otra entidad. El atributo **`'type'`** juega un papel fundamental en una iq Stanza, ya que es utilizado para especificar la operación de la misma. Estas pueden ser: **`get, set, result`** o **`error`**.

8.11.2 XMPP: Instant Messaging and Presence

El **XMPP: Core** especifica las funciones fundamentales del núcleo del protocolo XMPP como ser XML Streaming, seguridad en los canales, autenticación y las XML Stanzas generales pero no especifica el uso para alguna aplicación o tecnología, en particular para la mensajería instantánea, para ello existe la especificación **XMPP: I.M. & P.** la cual describe las extensiones de **XMPP: Core** para poder lograr servicios básicos de mensajería instantánea y manejo de presencia.

XML Stanzas para Mensajería Instantánea y Presencia

Anteriormente se presento la Message Stanza así como en la especificación **XMPP: Core** , en la especificación **XMPP: I.M. & P.** dicha Stanza es extendida para cubrir tipos y elementos específicos a la mensajería instantánea en si.

8.11.2.1 **Instant Messaging Stanzas***:*

Básicamente se extienden los valores del atributo **'type'** con las siguientes opciones: chat, groupchat, headline, normal o error. Los tipos de mensaje **'chat'** son utilizados para conversaciones uno a uno, los **'groupchat'** para conversaciones uno a muchos como chatrooms o IRC, **'headlines'** son para hacer broadcast de mensajes como por ejemplo "mensaje del día" a todos los usuarios, el tipo de mensaje **'normal'** son para mensajes que no entran en categoría de chat o groupchat y el tipo **'error'** es utilizado cuando ocurre un error en algún intercambio de mensajes anterior.

Adicionalmente las I.M. Stanzas especifican un conjunto de elementos "hijos" para mensajería (**subject**, **body** y **thread**)

Ejemplo de I.M. Stanza:

```
<message

   from='editorial@XMPP-servidor.com'

   to='WilliamSh@XMPP-servidor-2.com'

   type='chat'

   xml:lang='es'>

   <subject>Estimado Señor William</subject>

   <body>He visto su trabajo, no está del todo mal. Hay tensión,
         fantasía, sentido dramático...¿es la primera vez que
         escribe? </body>

   <body xml:lang='en'>I've seen your work, it isn't bad at all.
         There's tension, fantasy, a dramatic sense...this is the
         first time you write??</body>

</message>
```

```
<message

   from='WilliamSh@XMPP-servidor-2.com'

   to='editorial@XMPP-servidor-2.com'

   type='chat'

   xml:lang='en'>

   <subject>Dear Editorial Manager</subject>

   <body>No, I've written another tragedy, is the story of two
         lovers from Verona that...</body>

   <body xml:lang='es'>No, he escrito otra tragedia, es la
         historia de dos amantes de Verona que...</body>

</message>
```

En el ejemplo anterior vemos como puede ser utilizada la internacionalización en la M.I., el mensaje puede tener varios tags **<subject></subject>** y **<body></body>**, cada uno con un lenguaje distinto.

8.11.2.2 **Presence Stanzas**:

Así como la I.M. Stanza, dentro de la extensión para M.I. la Stanza de presencia puede ser de los siguientes tipos: **`unavailable, suscribe, suscribed, unsuscribe, unsuscribed, probe`** o **`error`**.

El tipo **`'unavailable'`** significa que la entidad ya no esta disponible, utilizando una Stanza de presencia de tipo **`'suscribe'`** se utiliza típicamente para agregarse como un contacto a un repositorio de otra entidad, si dicha entidad responde con una stanza de tipo **`'suscribed'`** significa que la suscripción fue autorizada, si la Stanza es de tipo **`'unsuscribed'`**, la suscripción es denegada, si el tipo de la Stanza es **`'unsuscribe'`** se utiliza típicamente para eliminar la presencia del repositorio de contactos de una entidad. El tipo **`'probe'`** es utilizado para obtener la presencia actual de una entidad y el tipo **`'error'`** es enviado si ocurre un error en el envío de stanzas de Presencia.

Una Stanza de Presencia también implementa los siguientes elementos:

`show, status` y **`priority`**. El primero especifica el estado de disponibilidad de la entidad, sus valores pueden ser : **`away`** (la entidad esta temporalmente no disponible), **`chat`** (entidad esta activa para entablar una conversación), **`dnd`** (no molestar "del inglés do not disturb"), **`xa`** (del inglés 'extended away' , significa que la entidad no estará disponible por un largo tiempo). Los valores de el elemento **<show></show>** no fueron creados para lectura humana, se manejan internamente entre las entidades , para ello puede utilizarse el elemento **<status></status>** . La prioridad puede utilizarse para manejo de prioridades de las Presence Stanzas.

Ejemplos:

Una entidad con JID "Sancho@La-Mancha.com" suscribe otra entidad con JID "Quijote@La-Mancha.com" la cual previamente envía el pedido de registro:

```
<presence to='Sancho@La-Mancha.com' type='suscribe'/>

Sancho ahora acepta el registro del Quijote

<presence to='Quijote@La-Mancha.com' type='suscribed/>
```

ejemplo de entidad que especifica su presencia como el de 'no molestar'

```
<presence xml:lang='es'>
   <show>dnd</show>
   <status>estoy en medio de un brete con el gerente</status>
   <status xml:lang='latin'>habemus embarradum gerentae</status>
    <priority>1</priority>
</presence>
```

8.11.2.3 **iq Stanzas**:

Son utilizadas generalmente en M.I. para manejo de listas de contactos o para obtener información de presencia, para mantener privacidad sobre los usuarios u otras entidades, la información de presencia y disponibilidad puede ser restringida solo a entidades que el usuario ha autorizado.

Por medio de iq Stanzas un cliente puede obtener su lista de contactos del servidor

Ejemplo:

```
<iq from='el_pepeMu@xmpp-uruguay.com' type='get' id='roster1'>
   <query xmlns='jabber:iq:roster'/>
</iq>
```

el servidor responde

```
<iq to='el_pepe@xmpp-uruguay.com' type='result' id='roster1'>
   <query xmlns='jabber:iq:roster'>
         <item jid='el_ñato@xmpp-uruguay.com/GAIM'
                     name='Eleuterio'
```

```
                    subscription='both'>
                <group>compañeros</group>
            </item>
            <item jid='el_horno@chatrooms.xmpp-uruguay.com'
                name='El Horno del XMPP'
                subscriptio='from'>
                <group>ChatRooms</group>
            </item>
    </query>
</iq>
```

8.11.3 XMPP : Mapeo de XMPP a CPIM

El grupo de trabajo de IMPP en la IETF especificó un framework interoperable en forma abstracta para M.I. llamado Common Presence and Instant Messaging (CPIM). Para lograr interoperabilidad entre distintas especificaciones de mensajería instantánea, cada una de dichas especificaciones debe definir un sistema de mapeo entre si misma y CPIM.

CPIM esta basado en la definición de mensajes mediante dos tipos de contenido MIME : "`Message/CPIM`" y "`aplication/pidf+xml`".

La especificación del RFC3922 especifica como las XML Stanzas son mapeadas en los tipos MIME de CPIM , el mapeo es bidireccional.

Para no entrar en detalle de los algoritmos y las reglas de mapeo entre los dos protocolos se presenta a continuación como ejemplo el mapeo de un mensaje sencillo, para poder tener una idea más clara:

XMPP XML Stanza:

```
<message

    from='daniel@tortuga.fing.edu.uy'
    to='mario@tortuga.fing.edu.uy'>

   <subject>Hola</subject>

   <body>Hola, que tal ??</subject>

</messsage>
```

```
CPIM MIME

Content-type: text/plain; charset=utf-8

Content-ID: <xyz@tortuga.fing.edu.uy>

From: daniel <im:daniel@tortuga.fing.edu.uy>

To: mario <im:mario@tortuga.fing.edu.uy>

Subject: Hola

Hola, que tal ??
```

8.11.4 XMPP: End-to-End Object Signing and Encryption

El RFC3923 define como las XML Stanzas pueden ser firmadas digitalmente y encriptadas. La idea esta basada en el mapeo XMPP-CPIM descrito anteriormente. Tampoco entraremos en detalle acerca del esquema de encripción del RFC, pero nuevamente, para tener idea del funcionamiento, el procedimiento para firma y encriptación consiste en:

1-Generar un objeto CPIM MIME de la XML Stanza

2-Encriptar y/o firmar el contenido y los cabezales del objeto CPIM generado

3-Poner el objeto encriptado resultante dentro en un elemento **<e2e></e2e>** dentro de un Message Stanza para enviar.

8.11.5 Consideraciones de seguridad en XMPP

Los protocolos SASL y TSL son fundamentales en XMPP para autenticación y utilización de canales seguros, respectivamente.

La especificación de XMPP recomienda que tanto en comunicaciones cliente-servidor como en servidor-servidor, se debe utilizar TLS para la fase de negociación en el momento de establecer una conexión. Dicha negociación es previa a la autenticación de la entidad, que debe hacerse utilizando SASL.

El procedimiento general de una negociación TLS/SASL exitosa se describe a continuación:

1- el cliente abre una conexión TCP con el servidor y comienza un XML Stream

2- el servidor envía una extensión STARTLS al cliente y describe los mecanismos de autenticación que soporta

3- el cliente responde con STARTLS

4- el servidor informa al cliente que puede proceder

5- el cliente y el servidor establecen el canal seguro utilizando TLS sobre la conexión TCP

6- si el paso anterior fue exitoso. El cliente inicia un nuevo XML Stream con el servidor

7- el servidor responde con otro stream enviando los mecanismos de autenticación (como en el paso 2)

8- el cliente selecciona el mecanismo de autenticación adecuado

9- el servidor envía un desafío codificado [Base64] al cliente

10- el cliente responde el desafío también de forma codificada [Base64] (le provee las credenciales adecuadas al servidor)

11- en caso de éxito, el servidor envía otro desafío codificado[Base64] (le envía un "ticket o certificado de sesión")

12- el cliente responde el desafío

13- en caso de éxito, el servidor informa al cliente que la autenticación de realizó de forma exitosa

14- el cliente comienza un XML Stream hacia el servidor para comenzar a enviar las XML Stanzas específicas de la aplicación.

8.11.6 Diferencias entre XMPP y Jabber

Si bien Jabber se podría ver como el padre de XMPP (o XMPP versión 0.9), y existen una gran cantidad de implementaciones de servidores basados en Jabber, la IETF explica detalladamente las diferencias fundamentales entre XMPP y Jabber en el RFC3920 ya que la idea es que los desarrolladores tengan en cuanta dichas diferencias y sea más fácil lograr el soporte de XMPP 1.0 en nuevas versiones de los servidores y clientes, a nivel de **XMPP: Core** las diferencias son siete y a nivel de **XMPP: M.I. & P**., las diferencias son dos.

8.11.6.1 Diferencias principales en el núcleo(core):

1- Encripción de canales

En Jabber, es muy común la utilización de canales encriptados mediante SSL en los puertos 5223 y 5270, en XMPP la encriptación de canales se realiza mediante TLS sobre puertos registrados por IANA (Internet Assigned Numbers Authority) como "xmpp-client" y "xmpp-server" la especificación recomienda que sean los puertos 5222 y 5269 (contrariamente, dichos puertos en la implementación de Jabber original son los puertos para conexiones no-seguras).

2- Autenticación

La autenticación cliente-servidor en Jabber se realiza mediante una iq Stanza, y no implementa una autenticación para conexiones servidor-servidor. XMPP define la utilización de SASL en la autenticación de ambos.

3- Manejo de Errores

En Jabber, los errores referentes a XML Stream son manejados de manera muy simple enviando un texto dentro de un tag **<stream:error></stream>**. XMPP define un manejo de errores mejor desarrollado y extensible con sub-elementos dentro de un elemento **<error></error>.**

En Jabber los errores relacionados con las Stanzas son manejados códigos de error en un estilo HTTP. XMPP implementa un mecanismo más elegante y completo a través de Stanzas de error.

4- Internacionalización

XMPP utiliza el atributo "xml:lang" para soporte multi-lenguaje.

Las otras 3 diferencias tratan sobre la forma de asignación de recursos, proceso de JID's y el atributo 'Versión' de un stream XML (esto es para saber las características del XML Stream según la versión).

8.11.6.2 Diferencias en la extensión para M.I. & P. del protocolo

1- Forma de establecer una sesión

El protocolo de autenticación cliente-servidor desarrollado en la comunidad Jabber, asume que cada cliente es un cliente de M.I. y por tanto inicia una sesión de mensajería instantánea *per sei*. XMPP mantiene una separación estricta entre las funcionalidades del núcleo y las de las extensiones para M.I. y P, entonces una sesión de M.I. no es creada hasta que el cliente especifique que ese será el cometido de la sesión.

2- listas privadas

Si bien la comunidad Jabber comenzó a trabajar en un mecanismo para bloqueo de comunicaciones (llamadas listas privadas) en el 2001, una vez formado el grupo para XMPP, la comunidad Jabber decidió esperar a la implementación de este grupo de la IETF.

9 BIBILOGRAFÍA

[1] Macroscape Solutions Inc. *Instant Messaging in the Enterprise*. 2002. [online]. Disponible desde <http://www.macrospace.com/pdf/IM.pdf> [citado 27/02/2005]
[2] Protocolo SIMPLE - IETF Drafts [online]. Disponible desde <http://www.ietf.org/ids.by.wg/simple.html> [citado 27/02/2005]
[3] Jabberd Studio [online]- Disponible desde <http://jabberstudio.org> [citado 19/01/2005]
[4] Jabberd 2 Installation and Administration Guide [online]- Disponible desde <http://jabberd.jabberstudio.org/2/docs/> [citado 19/01/2005]
[5] Newsgroups de Administración Jabber [online]- Disponible desde <gmane.network.jabber.admin> [citado 19/01/2005]
[6] CPAN Comprehensive PERL Archive Network [online]- Disponible desde <http://www.cpan.org> [citado 27/02/2005]
[7] Funky Penguin Open Source Solutions [online] - Disponible desde <http://www.funkypenguin.co.za/Bandersnatch> [citado 27/02/2005]
[8] Petrack Scott. *SIMPLE aims for M.I. interoperability*. Network World Fusion. Disponible desde <http://www.nwfusion.com/news/tech/2004/0119techupdate.html> [citado 15/10/2004.]
[9] Jabber for Mobile Users - Deploying Jabber Messaging Technology in your mobile enterprise Jabber for Mobile Users.[online]. Disponible desde <http://www.movsoftware.com/whitepapers.htm>[citado 15/10/2004.]
[10]Blue Coat Inc – Empresa dedicada a seguridad de servicios Web. Disponible desde <http://www.bluecoat.com/> [citado 27/02/2005]
[11]Sitio oficial del Working Group de XMPP de la IETF [online]. Disponible desde <http://www.XMPP.org> [citado 27/02/2005]
[12]Peter Saint-Andre. *JEP-0045: Multi-User Chat.Versión 1.16*. Publicado el 30 de junio de 2004 [online] Disponible desde <http://www.jabber.org/jeps/jep-0045.html> [citado 19/01/2005]
[13]Alexey Shchepin. *Ejabberd Installation and Operation Guide*. Publicado Octubre-2004. [online] Disponible desde <http://ejabberd.jabberstudio.org/guide.html> [citado 19/01/2005]
[14]*ejabberd*. [online] Disponible desde <http://ejabberd.jabber.ru/protocols>. [citado 19/02/2005]
[15]Sitio de empresa Jabber Inc [online]. Disponible desde <www.jabber.com> [citado 27/02/2005]
[16]Sitio de empresa Antepo Inc [online]. Disponible desde <www.antepo.com> [citado 27/02/2005]
[17]Sitio de empresa Jive Software [online]. Disponible desde <www.jivesoftware.com> [citado 27/02/2005]
[18]The Character, Functions, and Styles of Instant Messaging in the Workplace (estudio presentado en la conferencia de Computer-Supported Cooperative Work (CSCW), New Orleans, LA, November, 2002) [online]. Disponible desde < http://www.izix.com/pro/lightweight/IM.php > [citado 27/02/2005]
[19]Estadísticas de servidores Jabberd públicos [online]. Disponible desde <http://scriptrepo.jabberstudio.org/jslist/> [citado 19/01/2005]
[20]I.D.C. Abstract sobre estudio de mercado de la Mensajería Instantánea Empresarial [online] Disponible desde <http://www.idc.com/getdoc.jsp?containerId=31809> [citado 15/02/2005]
[21]Software/USExportRegulations - Open Source Code with Cryptography. Disponponible desde <http://www.freedesktop.org/wiki/Software_2fUSExportRegulations>. [citado el 04/05/03]