PUNTO DE ENCUENTRO VIRTUAL P2P CON ACCESO MÓVIL

RICARDO ALBERTO CAMACHO GÓMEZ
LUIS ERNESTO GARCÍA MARTÍNEZ

POPAYÁN
UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
Febrero de 2005

TABLA DE CONTENIDO

LISTA DE TABLAS Y FIGURAS

INTRODUCCIÓN

El crecimiento de Internet ha generado nuevas necesidades en las personas y a su vez la creación de nuevos servicios que suplan esas necesidades, pues el mundo actual requiere tener acceso a la información a cualquier hora y en cualquier lugar, por lo que los terminales conectados a Internet ya no son sólo computadores personales de ubicación fija, sino que estos terminales pueden ser cualquier unidad de procesamiento de información que posea algún mecanismo de conexión a la red, dándole entrada de esta manera a los equipos móviles, teléfonos celulares, Laptops y PDA, los cuales ya cuentan con estas capacidades de comunicación originando así lo que se conoce como "Internet móvil".
Los dispositivos móviles tienen grandes cualidades que los hacen muy atractivos en el mercado ya que le permiten al usuario movilidad en el acceso a la información, lo cual representa mayor eficiencia y dinamismo en el desempeño de las funciones de cada persona; la gran desventaja son las limitadas características de almacenamiento y procesamiento, lo cual restringe mucho la funcionalidad de las aplicaciones en los dispositivos.

Actualmente, los servicios de Internet móvil son provistos usando la arquitectura Cliente/Servidor, en la que un terminal se conecta a un servidor utilizando un protocolo determinado y de esta manera es posible hacer uso de ciertos recursos o servicios, sin embargo, esta arquitectura presenta algunas desventajas al tener los servicios centralizados en unas pocas máquinas; los clientes juegan un papel muy pasivo en la red al ser capaces de demandar servicios pero no de proveerlos, y no se aprovechan los recursos que cada uno de los clientes podría ofrecer. Por otra parte, se tiene una deficiencia en la información (multimedia, texto e imágenes) disponible para los dispositivos móviles (Teléfonos móviles, Smartphones, PDAs) y además una gran insuficiencia en sistemas especializados en el intercambio de datos entre usuarios móviles y terminales fijos en la red.

El presente proyecto busca explorar la aplicación de tecnologías de computación distribuida descentralizada (*peer-to-peer computing* o P2P) las cuales permiten que cualquier dispositivo sobre una red provea servicios a otros dispositivos[1], de tal forma que cada uno de estos pueda funcionar como cliente y servidor en dicha arquitectura de

[1] Brendon Wilson; "JXTA Book"; http://www.brendonwilson.com/projects/jxta/pdf/jxta02.pdf

red, con el objeto de enriquecer e impulsar la filosofía de las redes, compartir información. De esta manera un terminal en la red P2P puede tener y proveer acceso a diversos tipos de archivos (Música, documentos, imágenes, videos…), recursos de almacenamiento (Disco, memoria) y capacidades de procesamiento. Por esta razón se ve con gran interés el paradigma *Peer to Peer* como el mecanismo para solventar las deficiencias en el acceso a la información que demandan los usuarios móviles de hoy. Este tipo de tecnologías está ganando gran aceptación en aplicaciones innovadoras para Internet como el intercambio y distribución de archivos, mensajería instantánea, distribución de contenidos, compartir la capacidad de procesamiento y espacio de almacenamiento, trabajo colaborativo, y juegos interactivos.

Este proyecto se enmarca en el desarrollo de nuevos servicios para la Internet móvil en redes celulares de última generación (2.5G/3G). Se trata pues de explotar un nuevo modelo de computación mediante la implementación de un sistema interactivo y dinámico entre dispositivos móviles y fijos en una red P2P que facilite la comunicación y el intercambio de información de forma descentralizada y con gran aprovechamiento de recursos en grupos cuya creación, composición e interacción sea dinámica y flexible.

Con el desarrollo de este proyecto se logra una optimización en los servicios de Internet Móvil para los usuarios, explotando las capacidades tanto de los terminales como de Internet, pues al incorporar los dispositivos móviles como *peers* en una red P2P se construye un conjunto virtualmente infinito de recursos de computación y almacenamiento que se pone a disposición de millones de personas conectadas a través de esta.

I. EL PARADIGMA P2P

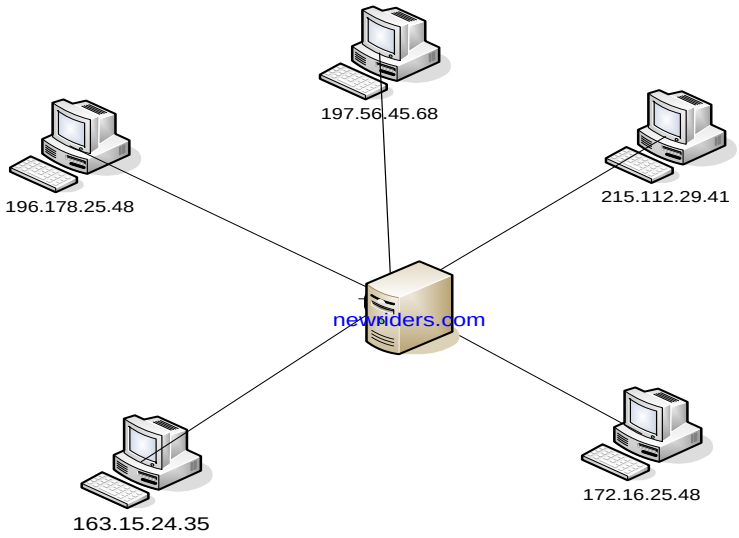
1.1. INTRODUCCIÓN AL PARADIGMA P2P (Peer to Peer).

El paradigma Peer to Peer (P2P) es un modelo que permite a los dispositivos con acceso a red (Computadoras de Escritorio, teléfonos celulares, servidores,…) ofrecer acceso a recursos y/o servicios que residen en otros dispositivos. Un dispositivo en una red P2P puede proveer acceso a cualquier tipo de recurso que tenga a su disposición, como documentos, capacidad de almacenamiento, capacidad de procesamiento, o incluso su propio operador humano.

Aunque P2P podría parecer una novedad, esta tecnología es una extensión de la filosofía de Internet que habla de la robustez mediante de la descentralización.

De la misma manera que Internet proporciona nombres de dominio (DNS), servicios World Wide Web, correo electrónico, y otros servicios mediante la distribución de responsabilidades entre millones de servidores, P2P tiene la capacidad de impulsar un nuevo conjunto de aplicaciones más robustas derivadas del aprovechamiento de los recursos distribuidos a través de todos los rincones de Internet.

La mayoría de los servicios en Internet están distribuidos usando la arquitectura tradicional Cliente/Servidor, ilustrada en la figura 1.1. En esta arquitectura, los clientes se conectan a un servidor usando un protocolo de comunicaciones específico, como por ejemplo el Protocolo de transferencia de archivos (FTP) para obtener acceso a un recurso específico. La mayor parte del proceso de que se lleva a cabo en el ofrecimiento del servicio usualmente ocurre de lado del servidor, dejando al cliente relativamente subutilizado. Gran parte de las aplicaciones más populares en Internet, incluyendo el WWW, FTP, Telnet, y el correo electrónico, usan este modelo para la prestación de su servicio.



Responsabilidades del cliente:
- Enviar comandos de petición de servicio.
- Recibir respuestas de una petición de servicio.

Responsabilidades del servidor:
- Recibir comandos de petición de servicio.
- Procesar la petición de servicios y ejecutar el servicio requerido.
- Enviar respuestas con resultados del servicio requerido.

Figura 1.1 Arquitectura Cliente-Servidor

Desafortunadamente, esta arquitectura tiene un gran inconveniente, en la medida en que el número de clientes aumenta, la carga y el ancho de banda demandados sobre el servidor también se incrementan impidiendo al servidor ocuparse del manejo de clientes adicionales. La ventaja de esta arquitectura es que requiere menos poder computacional del lado del cliente.

Irónicamente, la mayoría de los usuarios han sido persuadidos de equipar sus computadores con capacidades innecesarias que sobrepasan los requerimientos de las aplicaciones de Internet más populares, navegar por la Web y leer el e-mail.

El cliente en la arquitectura cliente/servidor actúa en un papel pasivo, capaz de demandar servicios de los servidores pero incapaz de proporcionar servicios a otros clientes. Este modelo se desarrolló en una época en la cual la mayoría de las máquinas en Internet tenían una dirección IP estática, lo que significaba que todas las máquinas en Internet podían encontrarse unas a otras fácilmente usando un nombre simple (como

sumaquina.com). Si todas las máquinas en la red ejecutaran tanto un servidor como un cliente, ellas formarían la base de una rudimentaria red P2P.

A medida que Internet fue creciendo, la limitación en la cantidad de direcciones IP disponibles llevó a los proveedores de servicios de Internet (ISP – Internet Service Provider) a implementar métodos de asignación dinámica de direcciones para sus clientes de acceso domiciliario a Internet. La naturaleza dinámica en la asignación de direcciones IP impidió que los usuarios pudiesen ofrecer servicios en sus terminales adecuadamente y aunque alguien podría estar ejecutando un servidor, un usuario no podría accederle a menos que supiera de antemano la dirección IP de la máquina que ofrece algún servicio. Estas máquinas se conectan a Internet pero son incapaces de participar en el intercambio de servicios. Por esta razón, muchos de los servicios están centralizados en servidores con direcciones IP estáticas y asociadas a un nombre o dominio fácil de recordar.

Otra razón por la que la mayoría de las máquinas de los clientes no pueden ejecutar servidores es que sean parte de una red privada, las cuales usualmente se aíslan de Internet por un firewall, un dispositivo diseñado para denegar conexiones no autorizadas dentro y fuera de la red privada. Las empresas normalmente crean una red privada para asegurar la información más crítica de la empresa así como para prevenir el abuso o malos manejos de la red. El efecto de esta tecnología es que un computador fuera de la red privada no se puede conectar a un computador dentro de la red privada para obtener servicios.

Considérese la cantidad de procesamiento y capacidad de almacenamiento que las máquinas de los clientes representan. Supóngase que se conectan sólo 10 millones de usuarios con máquinas de 100MHz a Internet en un determinado momento, cada uno tiene libres 100MB de espacio en disco, 1Kbps de ancho de banda y 10% de poder de procesamiento. En este momento, estos clientes representan 10 petabytes (PB) (10^{15} bytes) de espacio de almacenamiento disponible, 10 billones bps de ancho de banda disponible (aproximadamente1.25GBps), y 10^8 MHz de poder de procesamiento desperdiciado. Éstas son estimaciones que indican el enorme potencial sin aprovechar que espera ser liberado en la Internet.

P2P es la clave para aprovechar este potencial, dándole a cada máquina un mecanismo para proporcionar servicios a otras. A diferencia de la arquitectura cliente/servidor, las redes P2P no dependen de un servidor central para ofrecer servicios, éstas redes normalmente operan fuera del sistema de nombres de dominio. Como se muestra en la

Figura 1.2, las redes P2P evitan la organización centralizada de la arquitectura cliente/servidor y en cambio emplean una arquitectura plana, una arquitectura altamente interconectada.

La ventaja principal de las redes P2P es que distribuyen la responsabilidad del ofrecimiento de servicios entre todos los peers en la red; esto elimina las interrupciones del servicio debido a fallas en un único punto de la red y ofrece una solución más escalable para el ofrecimiento de servicios. Además, las redes P2P aprovechan el ancho de banda disponible a través de toda la red usando una variedad de canales de comunicación y ocupando el ancho de banda de la gran Internet.

A diferencia de las comunicaciones cliente/servidor tradicionales en las que las rutas a los sitios más populares, como por ejemplo *www.google.com* podrían saturarse demasiado, P2P pone a disposición muchas más rutas hacia un mismo destino en la red, reduciendo así la congestión por tráfico.

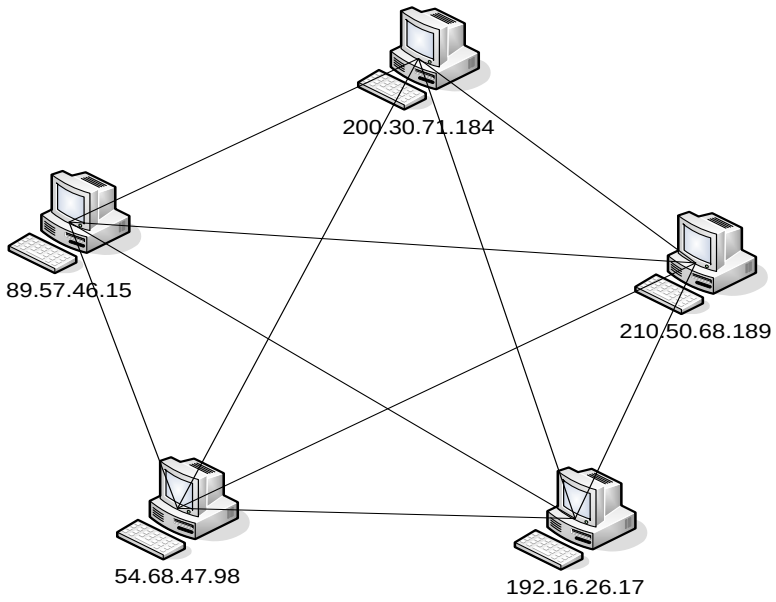


Figura 1.2. Arquitectura P2P

Responsabilidades del cliente:
- Enviar comandos a los peer para solicitar servicios.
- Recibir respuesta a las solicitudes de servicio.

Responsabilidades del servidor:
- Recibir comandos de otros peers solicitando servicios.
- Procesar solicitudes de servicio y ejecutar el servicio solicitado.
- Enviar respuestas con los resultados del servicio solicitado.
- Difundir solicitudes de servicio a otros peers.

P2P tiene la capacidad de prestar recursos con alta disponibilidad a un costo mucho más bajo y a la vez aprovechar el uso de los recursos de cada peer.

En tanto que las soluciones cliente/servidor dependen del ancho de banda, de los equipos, y de la ubicación para mantener soluciones robustas, P2P puede ofrecer un nivel comparable de robustez mediante la distribución de la red y la distribución de los recursos demandados a través de la red de P2P.

Desafortunadamente, P2P presenta algunas desventajas debido a la naturaleza redundante de su estructura de red. La forma de distribución de los canales de comunicación en las redes P2P da como resultado demandas de servicios de manera *no-determinística*. Por ejemplo, clientes que piden exactamente el mismo recurso de red podrían conectarse a diferentes máquinas por diferentes rutas de comunicación, y obtener resultados diferentes. Las solicitudes enviadas a través de una red P2P pueden no producir una respuesta inmediata y, en algunos casos, podrían no dar una respuesta. Los recursos en una red P2P pueden desaparecer en el momento en el que los clientes que almacenan esos recursos se desconectan de la red; a diferencia de los servicios proporcionados tradicionalmente en Internet, en el cual la mayoría de los recursos están continuamente disponibles.

Sin embargo, P2P puede superar todas estas limitaciones. Aunque los recursos pueden desaparecer en algunas ocasiones, una aplicación P2P podría implementar "mirrors", ó puntos de réplica de la información en múltiples peers, y de este modo proporcionar un acceso redundante a los recursos. Un gran número de peers interconectados reduce la probabilidad de que la solicitud de un servicio no sea contestada. Para abreviar, la misma estructura de una red P2P que causa los problemas puede ser usada para resolverlos.

1.1.1. La Importancia de P2P

Aunque P2P ganó popularidad como un medio para distribuir ilegalmente propiedad intelectual con derechos de autor, P2P tiene mucho más para ofrecer al mundo de la informática que el acceso fácil a música o archivos de video. Para ilustrar las ventajas que traen los protocolos P2P sobre los sistemas actuales de localización de contenido y recursos en Internet considérese el siguiente ejemplo.

Para encontrar alguna información específica en Internet, los usuarios se dirigen normalmente a un navegador Web o a su motor de búsqueda favorito, como por ejemplo "Google", y envían una solicitud de búsqueda.

Generalmente se recibe una lista de miles de resultados, muchos de los cuales no se relacionan, están desactualizados, o peor aún, hacen referencia a recursos que ya no existen.

Uno de los problemas con los actuales motores de búsqueda son las falsas soluciones de la centralización de conocimientos y recursos. Google depende de una base de datos central que se actualiza diariamente buscando en todos los sitios de Internet información nueva. Debido al número de páginas Web indexadas en su base de datos (más de 1.6 mil millones), no todas las entradas se actualizan diariamente. Como resultado de esta limitación, la información en la base de datos de Google no muestra la información más actualizada que se encuentra disponible, disminuyendo de esta forma los resultados útiles de cualquier búsqueda.

La tecnología de los motores de búsqueda presenta otras desventajas:

- Requiere una gran cantidad de equipos. Por ejemplo, Google ejecuta un clúster Linux de 10,000 máquinas para proporcionar su servicio.
- Si el motor de búsqueda se desconecta (Por ejemplo debido a un daño en la red), toda la información del motor de búsqueda ya no estará disponible.
- Debido a la gran extensión de Internet, el motor de búsqueda no puede proporcionar un índice global de Internet.
- Los motores de búsqueda no pueden interactuar con información guardada en la base de datos de un sitio Web empresarial, lo cual significa que un motor de búsqueda no puede "ver" toda la información.

Un servicio similar podría implementarse usando la tecnología P2P, mejorando el servicio capacitándolo con características más avanzadas. Imagínese que cada persona pudiera ejecutar un servidor Web personal en su computador de escritorio. Suponga que además publicara contenido de la máquina, este servidor tendría la capacidad de procesar solicitudes de información sobre los documentos que maneja, y responder a dichas peticiones con una lista de los documentos que más se ajustan a la petición.

El servidor del usuario tendría la responsabilidad de indexar los documentos que pone a disposición y por consiguiente estaría en capacidad de proporcionar información más actualizada sobre dichos documentos a cualquiera que solicite una búsqueda. La indexación de los documentos de un solo usuario sería una tarea mucho más manejable que si la comparamos con la gestión del índice de documentos que se hace en un

buscador como Google, estamos comparando algunas docenas de documentos frente a billones de páginas.

Entidades de diversa índole podrían instalar gateways para conectar las bases de datos de sus sitios Web a la red P2P, proporcionando acceso a la información que los motores de búsqueda actualmente no pueden alcanzar.

El sistema tendría esta gran ventaja: Si el servidor del usuario se desconecta de la red, el servicio de búsqueda sabría que dichos recursos no están ahora disponibles; de tal manera que los usuarios que ejecuten búsquedas en la red no recibirían resultados de los recursos que no estén disponibles. Y así, a cualquiera que busque información, se le garantiza que cualquier resultado que se encuentre estará disponible, reduciendo de esta manera el tiempo de búsqueda utilizado.

Se podrían ordenar los resultados de la búsqueda incluso de toda la red para determinar qué información podría satisfacer mejor las necesidades basándose en varias características como la disponibilidad del servidor que almacena el recurso o el número de servidores que almacenan una copia del mismo recurso.

Este ejemplo de aplicación de la tecnología P2P no es perfecto, sin embargo, el ejemplo ilustra el principio subyacente de P2P: *permitir a cualquiera ofrecer servicios en una red*. Hasta ahora, la experiencia tradicional en Internet ha sido bastante pasiva.

Así como la edición de textos por computador fue la revolución a mediados de los 80's, P2P promete revolucionar el intercambio de información en Internet ahora.

1.1.2. Una Breve Historia de P2P

El paradigma P2P siempre ha existido, pero no siempre se ha reconocido como tal; los servidores con IP's fijas siempre han tenido la capacidad de comunicarse con otros servidores para acceder a los servicios. Algunas aplicaciones anteriores a P2P, como la del servicio de correo y el servicio de nombres de dominio (DNS), se basan en estas características para dar un ejemplo de redes distribuidas, pero una de estas aplicaciones, *Usenet*, sobresale entre todas.

Usenet se creó en 1979 por dos estudiantes graduados de la Universidad de Carolina del Norte, Tom Truscott y Jim Ellis, para ofrecer una forma de intercambiar información entre dos computadores en los primeros días antes de la gran expansión de Internet. En su primera versión se le permitía a una computadora comunicarse con otra mediante marcación, comprobar sus archivos, y descargar esos archivos; esto se hacía durante la noche para ahorrar costos de larga distancia telefónica. El sistema evolucionó al sistema

masivo de noticias que existe en la actualidad. *Usenet* tiene algunas propiedades que le ayudan a distinguirse como probablemente la primera aplicación de P2P. *Usenet* no tiene ningún director central, la distribución de contenidos se maneja por cada nodo, y el contenido de la red *Usenet* se reproduce (totalmente o en parte) por sus nodos. Una de las cosas más interesantes sobre *Usenet* es lo que es: Nada!

Usenet no es un bloque de software o una red de servidores, aunque requiere el software y servidores para operar, estas cosas no definen a *Usenet* verdaderamente. En esencia, *Usenet* es un modelo definido para que las máquinas hablen unas con otras y permitan que los mensajes de noticias sean enviados y difundidos por la red, generando así un protocolo, el Protocolo de Transporte de Noticias de Red (NNTP – Network News Transfer Protocol) (lETF RFC 977), un gran número de máquinas puede participar en la red ofreciendo servicios de manera independiente. Esta distribución de responsabilidades es la que distingue a *Usenet* como la primera y verdadera, aunque rudimentaria, aplicación de tecnología P2P.

Desde *Usenet*, las aplicaciones P2P más populares han estado en una de las siguientes categorías: mensajería instantánea (IM), intercambio de archivos (File Sharing), y computación distribuida (Distributed Computing). Considérese a continuación una breve reseña de cada una de estas categorías.

1.1.2.1. Mensajería Instantánea (IM)

Cuando Mirabilis (Empresa de software predecesora de ICQ, fundada por cuatro jóvenes israelíes) sacó a la luz el ICQ (www.icq.com) en noviembre de 1996, puso a disposición de los usuarios una manera más rápida de comunicarse que la que daba el correo electrónico tradicional. ICQ (Pronunciado en inglés de la misma forma que "I seek you" –> Te busco) permite a los usuarios ser notificados cuando sus amigos están en línea y enviar mensajes instantáneos entres sí. Además de sus capacidades para mensajería instantánea, ICQ permite a los usuarios intercambiar archivos. Aunque clasificada como una aplicación P2P, ICQ es un híbrido de P2P y de la arquitectura cliente/servidor a la hora de analizar cómo es que ofrece su servicio, como se muestra en la Figura 1.3.

ICQ usa un servidor central para supervisar qué usuarios están en línea y notificar a las partes interesadas cuándo los nuevos usuarios se conectan a la red. Sin embargo todo el resto de la comunicación entre los usuarios se dirige de una manera P2P, con mensajes que fluyen directamente de la máquina de un usuario al otro sin la intermediación del servidor.

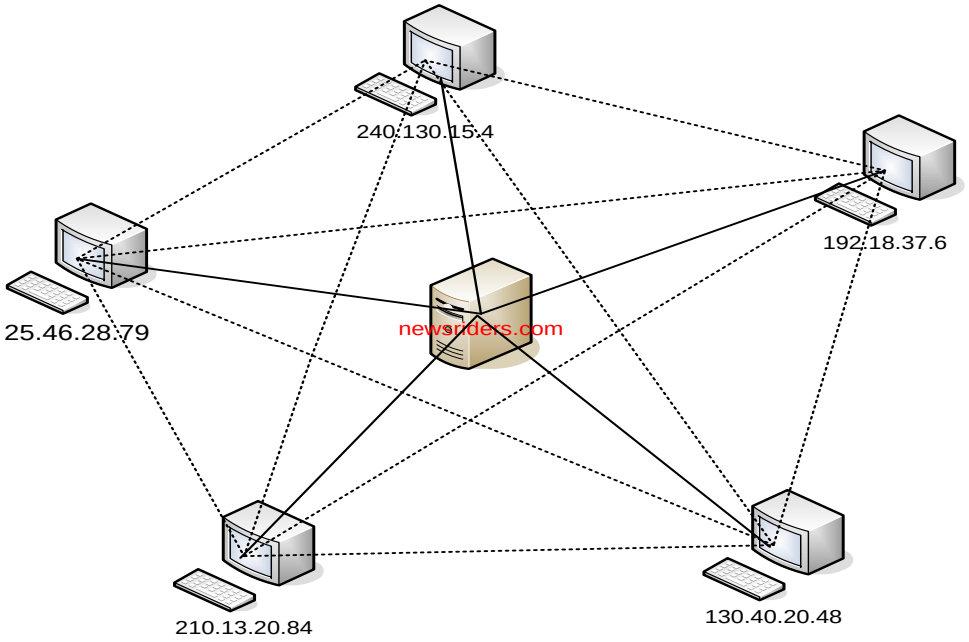


Figura 1.3 Red empleada en la operación de ICQ (Híbrido P2P)

------ : Enlaces por demanda.
_____ : Enlaces temporales necesarios.

Responsabilidades del cliente:
- Registrar o borrar del registro los servicios disponibles con el servidor.
- Enviar comandos al servidor para encontrar algún servicio específico.
- Recibir respuestas del servidor con la lista de los peers que ofrecen el servicio solicitado.
- Recibir respuesta a las peticiones de servicio de un peer específico.
- Recibir comandos de otros peers solicitando un servicio en especial.
- Procesar peticiones de servicio y ejecutar las acciones correspondientes a dichas peticiones.
- Enviar respuesta a las peticiones de servicio ejecutadas por otros peers.

Responsabilidades del servidor:
- Registrar y eliminar del registro los servicios peer disponibles.
- Recibir comandos de localización de servicios y recursos.
- Buscar recursos disponibles registrados por peers.
- Enviar respuestas con la ubicación de *newriders.com*

ICQ desde su aparición ha tenido muchos imitadores, incluso MSN Messenger, (www.messenger.msn.com), AOL Internet Messenger (www.aol.com/aim), y Yahoo Messenger (www.messenger.yahoo.com). Lamentablemente, estas aplicaciones no son compatibles; cada uno depende de su propio protocolo de comunicación propietario. Como resultado de esta incompatibilidad, los usuarios deben descargar el software cliente para cada red de mensajería y deben ir a través de un proceso de registro separado para cada red. La mayoría de usuarios a pesar de esto se han tenido que

ajustar a las condiciones que imponen dichas compañías, teniendo como consecuencia un crecimiento independiente de cada comunidad sin interacción alguna entre sí.

Recientemente, varios desarrolladores de software han intentado interconectar estas comunidades separadas usando ingeniería inversa de los protocolos de IM y haciendo un nuevo software del cliente. Dicha aplicación, *jabber* (www.jabber.com), proporciona gateways para los principales servicios de mensajería, permitiéndoles a los usuarios interactuar con otros de diferentes redes de IM. Este intento ha encontrado oposición por parte de los proveedores de servicio, por ejemplo AOL ha cambiado su protocolo de comunicación en un intento de bloquear a los clientes de Jabber y mantener su red libre de ellos.

1.1.2.2. Intercambio de Archivos (File Sharing)

El popular Napster (www.napster.com) tuvo su fase de despliegue en Internet a lo largo de 1999, dando a los usuarios la capacidad de intercambiar archivos de MP3. Napster empleaba una solución de P2P híbrida similar a la de ICQ, contando con un servidor central para guardar una lista de los archivos de MP3 que había en cada una de las máquinas de los usuarios. Este servidor también era responsable de responder a las consultas a cerca de los archivos de música que había disponibles y la dirección de la máquina que los contenía.

La funcionalidad del intercambio de archivos era coordinada directamente entre los peers sin un servidor intermediario.

Además de compartir archivos, Napster proporcionaba funciones de Chat que permitían a los usuarios enviar mensajes de texto entre sí.

Tomando como ejemplo a Napster, pero teniendo en cuenta las implicaciones legales del violar los derechos de autor, el proyecto Gnutella (www.gnutelliums.com) tomó el concepto de intercambio de archivos establecido por Napster y fue un paso más allá al eliminar la necesidad de un servidor central para proporcionar las funciones de búsqueda. La independencia de un servidor de la red de Gnutella, combinada con su capacidad para compartir cualquier tipo de archivo, hace de este proyecto una de las demostraciones más potentes de la tecnología P2P.

Los peers en la red de Gnutella son responsables no sólo de servir los archivos, sino también de responder a las peticiones y enrutar los mensajes a otros peers. Aunque Gnutella no requiere un servidor central para proporcionar búsquedas y resolver direcciones IP, conectarse a la red de Gnutella solo requiere que un peer sepa la

dirección IP de otro peer que ya esté conectado a la red P2P. Por esta razón, varios peers con direcciones IP estáticas o conocidas han sido establecidos para proveer a nuevos peers un punto de inicio para descubrir otros peers sobre la red.

Al eliminar la dependencia de un servidor central se han evidenciado nuevos problemas, por ejemplo:

- ¿Cómo los peers distribuyen mensajes a otros sin inundar la red?
- ¿Cómo los peers proporcionan contenido seguro y anónimo?
- ¿Cómo la red puede estimular los recursos compartidos?

Otras variantes de intercambio de archivos de P2P, incluyendo *Freenet* (freenet.sourceforge.net), Morpheus (www.musiccity.com), y MojoNation (www.mojonation.net), tienen estos mismos problemas. Cada una de estas aplicaciones aborda un problema específico. Freenet proporciona almacenamiento de contenido anónimo y descentralizado protegido por criptografía contra saboteadores. Morpheus ofrece capacidades de búsqueda mejorada basadas en metadatos embebidos en formatos comunes. MojoNation usa una moneda artificial, llamada Mojo, para incentivar los recursos compartidos.

La Tragedia del Pueblo…

En muchas comunidades que comparten los recursos, hay un riesgo de padecer la "Tragedia del Pueblo": el uso excesivo de un recurso compartido al punto de su destrucción. La Tragedia del Pueblo se refería originalmente al problema de sobre pastoreo en tierras públicas, pero el término puede aplicar a cualquier recurso público que se use sin restricción.

En algunos sistemas P2P, los peers pueden usar los recursos (el ancho de banda y el espacio de almacenamiento) de otros en la red sin poner a su disposición sus propios recursos en la red, reduciendo así la característica más importante de la red, el ser fuente de recursos e información para toda la comunidad. Cuando más usuarios deciden no compartir sus recursos, esos peers que si comparten los recursos se les incrementa la carga y la red empieza a revertirse a la arquitectura clásica cliente/servidor. Llegando a la conclusión lógica de que la red en el futuro colapse y no beneficie a nadie.

Las soluciones P2P más actuales han intentado prevenir "La tragedia del Pueblo" incorporando chequeos que aseguren que los usuarios comparten los recursos. Lime Wire (www.limewire.com) por ejemplo, permite restringir a los usuarios descargas basado en el número de archivos que un cliente está compartiendo en la red.

MojoNation (www.mojonation.net) toma este modelo un paso más allá e incorpora un sistema de dinero virtual, el cual los usuarios ganan en la medida en que compartan sus recursos y después lo gastan accediendo a otros recursos.

1.1.2.3. Computación distribuida (Distributed Computing)

La computación distribuida es una manera de resolver problemas difíciles dividiendo el problema en sub-problemas que pueden ser resueltos independientemente por un gran número de computadores.

Aunque las aplicaciones más populares de computación distribuida no han sido soluciones P2P, es importante notar el avance y la contribución que han hecho proyectos como *SETI@Home* (setiathome.berkeley.edu) y *Distributed.net* (distributed.net) y compañías como *United Devices* (www.ud.com).

En 1996, *SETI@Home* empezó distribuyendo una aplicación de protector de pantalla a los usuarios, para poder procesar los datos del radio-telescopio y de esta manera contribuir a la búsqueda de vida extraterrestre. Desde entonces, ha conseguido más de 3 millones de usuarios (de los cuales más de medio millón son contribuyentes activos).

En un proyecto similar que comenzó en 1997, Distributed.net usó el poder de cómputo de sus usuarios para hackear mensajes encriptados. En ambos casos, el software del cliente contacta a un servidor para descargar su porción del problema y empezar a resolverlo; hasta que el problema se solucione, no se requiere ninguna comunicación con el servidor.

En un futuro cercano, se espera que la computación distribuida evolucione para aprovechar la tecnología P2P y crear así el mercado de la CPU o capacidades de procesamiento como valor a comerciar.

1.2. CONCEPTOS P2P

En esta sección se introduce la terminología y los conceptos usados para describir la arquitectura general la cual es común a toda red P2P, los componentes habituales en todas las soluciones que siguen este paradigma (incluyendo aquellas que no están construidas usando la tecnología de JXTA). Por otra parte, se dispone de una profundización en la comunicación P2P en el Anexo I.

1.2.1. Elementos Esenciales en las Redes P2P

P2P es la solución al deseo de los usuarios de conectar sus terminales de tal manera que puedan compartir información, recursos, y servicios.

Esto, parece en un principio simple, pero para poder entenderlo se requiere saber algunas cosas, por ejemplo: Cómo un terminal se entera de la presencia de otro terminal?, cómo los terminales se organizan en torno a intereses comunes?, cómo un terminal hace que los demás conozcan sus capacidades?, que información se requiere para identificar a un determinado terminal? y finalmente cómo los terminales intercambian datos.

Todas las redes P2P tienen elementos fundamentales que proporcionan respuesta a estas y otras preguntas. Desafortunadamente, muchas de las implementaciones de estos elementos están codificadas en las redes P2P propietarias, produciendo inflexibilidad en las aplicaciones. Por ejemplo, la mayoría de soluciones P2P actuales asumen el uso de TCP como el protocolo de transporte, lo cual limita las aplicaciones al no poder operar en cualquier ambiente de red. Las soluciones P2P flexibles necesitan un lenguaje que explícitamente declare todas las variables para cualquier solución P2P.

1.2.1.1. Los Peers

Un peer es un nodo en una red P2P que forma la unidad de proceso fundamental de cualquier solución P2P. Es cualquier entidad capaz de realizar algún trabajo útil y de comunicar los resultados de ese trabajo a otra entidad sobre una red, directamente o indirectamente.

La definición de trabajo útil depende del tipo de peer. Existen tres posibles tipos de peers en cualquier red P2P:

- Peers Simples (Simple peers).
- Peers Rendezvous (Peers de punto de encuentro).
- Peers Routers (Peers de enrutamiento).

Cada peer en la red puede actuar como uno o más tipos de peer, con cada tipo definiendo un conjunto de responsabilidades diferente en la red P2P.

1.2.1.1.1. Peers simples

Un peer simple esta diseñado para funcionar en un terminal de usuario final, permitiéndole a este usuario proporcionar servicios desde su terminal y utilizar servicios

proporcionados por otros peers en la red. Es muy probable, que un peer simple este localizado detrás de un firewall, separándolo así de la red; y ocasionando que los peers fuera del firewall probablemente no sean capaces de comunicarse directamente con el peer simple localizado dentro del firewall.

Debido a su accesibilidad limitada a la red, los peers simples tienen poca responsabilidad en cualquier red P2P. A diferencia de otros tipos de peers, estos no son responsables de manejar la comunicación en nombre de otros peers o de servir la información de terceros a otros peers.

1.2.1.1.2. Peers Rendezvous

Tomado literalmente, Rendezvous traduce encuentro o punto de encuentro; en P2P, un peer Rendezvous proporciona a los peers un sitio en la red que sirve para descubrir otros peers y los recursos de éstos. Los peers generan consultas discovery a un peer Rendezvous, y el Rendezvous les proporciona información sobre los peers que él conoce en la red. Un peer Rendezvous puede aumentar sus capacidades almacenando en caché información sobre los peers para usarla en el futuro o reenviando consultas discovery a otros peers Rendezvous. Este esquema tiene la ventaja de mejorar el grado de reacción, reducir el tráfico de la red, y proporcionar mejores servicios a los peers simples.

Un peer Rendezvous normalmente existe fuera del firewall de una red privada. Un Rendezvous puede estar detrás de un firewall, pero necesita poder pasarlo usando un protocolo autorizado por éste o mediante un peer router que esté fuera del firewall.

1.2.1.1.3. Peers Router

Un peer Router provee un mecanismo para que los peers se comuniquen con peers que están separados de la red por un firewall o por un equipo de traducción de dirección de red (NAT). Un peer Router proporciona la forma para qué peers que están fuera del firewall pueden comunicarse con peers que están detrás del firewall, y viceversa. Esta técnica de firewall y NAT se discute en detalle mas adelante.

Para enviar un mensaje a un peer por medio de un router, el peer que envía el mensaje debe primero determinar cual peer router usar para comunicarse con el peer destino. Esta información de enrutamiento proporciona un mecanismo en P2P para remplazar a los DNS tradicionales, permitiéndole a un dispositivo que no esta conectado permanentemente y que usa una dirección IP dinámica, ser encontrado en la red. De la misma forma en que un DNS traduce un nombre a una dirección IP, la información de

enrutamiento proporciona un mapeo entre un identificador único que especifica a un peer remoto en la red y una representación que puede ser usada para contactar al peer remoto por medio de un peer router.

En los sistemas simples, enrutar la información podría consistir solamente en resolver una dirección IP y un puerto TCP para dar un identificador único. Un sistema más complejo proporciona información de enrutamiento mediante una lista ordenada de peers router que se usan para enrutar un mensaje hacia un peer. Para permitir que dos peers se comuniquen estando en redes de transporte diferente, es necesario enrutar el mensaje a través de múltiples peers router, ya que los peers router son los encargados de pasar la información entre redes de transporte diferentes e incompatibles.

1.2.1.2. Peer Groups

Hasta ahora la naturaleza especializada de las soluciones P2P y de sus protocolos asociados han divido el uso de la red según el tipo de aplicación. Por ejemplo para ejecutar archivos compartidos, probablemente se usaría el protocolo de Gnutella y sólo se tendría la capacidad de comunicase con otros peers que usaran el mismo protocolo; de forma similar, para ejecutar mensajería instantánea, se usaría ICQ y sólo se podría comunicar con otros peers que también usaran ICQ.

La incompatibilidad de protocolos ha dividió el espacio de la red dependiendo del tipo de aplicación usada por los peers involucrados. Si se considera un sistema de P2P en el cual todos los clientes pueden hablar el mismo conjunto de protocolos, el concepto de peer group se hace necesario para subdividir el espacio de la red. Es así como un peer group se define de la siguiente forma:

Un conjunto de peers formado para servir a un interés común o a los objetivos estipulados por los peers involucrados. Los peer groups pueden proporcionar servicios a sus peers miembros que no son accesibles por otros peers en la red P2P.

Los peer groups dividen la red P2P en grupos de peers con objetivos comunes basándose en:

- **La aplicación con la que se desea colaborar este en función de un grupo:** Un peer group se forma para intercambiar servicios que los miembros no quieren hacer disponibles a toda la población de la red P2P. Una razón para hacer esto podría ser la naturaleza privada de los datos usados por la aplicación.

- **Los requerimientos de seguridad de los peers involucrados:** Un peer group podría emplear la autenticación de servicios para restringir quién puede entrar en el grupo y acceder a los servicios ofrecidos por el grupo.
- **La necesidad de la información de estado sobre los miembros del grupo:** Los miembros de un peer group pueden supervisar a otros miembros. La información de estado puede usarse para mantener un nivel mínimo de servicio para la aplicación del peer group.

Los miembros de un peer group pueden proporcionar acceso redundante a un servicio, asegurando de esta forma que un servicio siempre esté disponible al peer group mientras haya al menos un miembro proporcionando el servicio.

1.2.1.3. Transporte en la red

Para intercambiar datos, los peers deben emplear algún tipo de mecanismo para manejar la transmisión de datos sobre la red. Esta capa, conocida como "Capa de Transporte de Red" es responsable de todos los aspectos de transmisión de datos, incluso la ruptura de datos dentro de los paquetes, agregando cabeceras apropiadas a un paquete para controlar su destino, y en algunos casos, asegurando que el paquete llegue a su destino. El transporte en la red podría ser un transporte de bajo nivel, como UDP o TCP, o un transporte de alto nivel, como HTTP o SMTP.

El concepto de transporte en la red P2P puede dividirse en tres partes fundamentales:

- **Endpoints**: La fuente o el destino de cualquier segmento de datos que se transmite a través la red. Un endpoint corresponde a las interfaces de red usadas para enviar y recibir datos.
- **Pipes**: Canales de comunicación virtuales unidireccionales y asíncronos, que conectan dos o más endpoints.
- **Messages**: Contenedores de datos que se transmiten sobre un pipe desde un endpoint a otra.

Para comunicarse usando un pipe, definido arriba, un peer necesita primero encontrar los endpoints, uno para la fuente del mensaje y otro para cada destino del mensaje, y conectarlos uniendo un pipe a cada uno de los endpoints. Cuando se traza este camino, el endpoint que actúa como fuente de datos es llamado *output pipe* y la terminación que actúa como sumidero de datos es llamada *input pipe*. El pipe en si no es responsable de llevar realmente los datos entre los endpoints; es solamente una abstracción usada para representar el hecho de que dos endpoints están conectados. Los endpoints

proporcionan el acceso a la interfase de red usada para transmisión y recepción de datos.

Para enviar los datos de un peer a otro, un peer empaqueta los datos que van a ser transmitidos en un mensaje y envía el mensaje usando un *output pipe*; en el endpoint opuesto, un peer recibe el mensaje desde un *input pipe* y extrae los datos transmitidos.

Cabe anotar que un pipe proporciona comunicación en una sola dirección, así que se requieren dos pipes para lograr la comunicación bidireccional entre dos peers.

Aunque la comunicación bidireccional es normal en las redes modernas, no hay ninguna razón para excluir la posibilidad de un canal de comunicación unidireccional porque en definitiva cualquier red de transporte bidireccional puede modelarse fácilmente usando dos pipes unidireccionales.

1.2.1.4. Los Servicios

Los servicios proporcionan funcionalidad que los peers pueden emplear para ejecutar "el trabajo útil" sobre un peer remoto. Este trabajo podría incluir transferencia de archivos, información de estados, realización de cálculos, o hacer básicamente cualquier cosa que se pueda desear en un peer sobre una red P2P y que este tenga la capacidad de hacerlo. Los servicios son el motivo por el cual se reúnen los dispositivos en una red P2P; sin servicios, no se tendrían redes P2P (solo se tendría un conjunto de terminales incapaces de ayudarse unos a otros con sus recursos.

Los servicios pueden ser divididos en dos categorías:

- **Servicios Peer**: Son servicios ofrecidos por un peer en particular a otros peers en la red. Las capacidades de estos servicios son únicas del peer y sólo estarán disponibles mientras el peer este conectado a la red. Cuando el peer se desconecta de la red, el servicio es deshabilitado.
- **Servicios de peer groups:** Son servicios ofrecidos por un peer group a los miembros del grupo. La funcionalidad de los servicios puede ser proporcionada por varios miembros del peer group, y de este modo proporcionar acceso redundante a los servicios. Con tal de que un miembro del grupo este conectado a la red y esté proporcionando el servicio, el servicio es habilitado para todos los miembros del peer group.

La mayoría de la funcionalidad requerida para crear y mantener una red P2P, esta en los protocolos fundamentales, los cuales son necesarios para encontrar a los peers y a sus

recursos, los protocolos pueden ser también considerados como servicios. Este núcleo de servicios proporciona los cimientos básicos P2P usados para construir otros servicios más complejos.

1.2.1.5. Advertisements

Hasta ahora, las aplicaciones P2P han usado una forma informal de advertisements. En Gnutella, el resultado devuelto por una consulta de búsqueda podría ser considerado un advertisement que especifica la localización de un archivo de música específico sobre la red de Gnutella. Estos advertisements primitivos están sumamente limitados en su propósito y aplicación. En su núcleo, un advertisement se define de la siguiente forma:

Representación estructurada de una entidad, servicio o recurso que se hace disponible por un peer o por un peer group como parte de una red P2P.

Todos los bloques de construcción discutidos hasta este punto pueden ser descritos por advertisements, incluso los peers, los peer groups, los pipes, los endpoints, los servicios y el contenido. Los advertisements son una poderosa herramienta para describir los recursos y para simplificar la tarea de organizar las redes P2P.

1.2.1.6. Los Protocolos

Cada intercambio de datos depende de un protocolo para dictaminar qué datos se envían y en qué orden estos se envían. Un protocolo es simplemente:

Una manera de estructurar el intercambio de información entre dos o más partes que usan reglas que han sido previamente convenidas por todas las partes.

En P2P, los protocolos deben definir cada tipo de interacción que un peer puede realizar como parte de la red P2P:

- Encontrar los peers en la red.
- Encontrar qué servicios proporciona un peer.
- Obtener información del estado de un peer.
- Invocar un servicio de un peer.
- Crear, unirse, y dejar los peer groups.
- Crear conexiones de datos con los peers.
- Enrutar los mensajes hacia otros peers

La organización de la información en advertisements simplifica los protocolos requeridos para hacer el trabajo en P2P. Los advertisements dictaminan la estructura y la

representación de los datos, simplificando la definición de un protocolo. En lugar de pasar los datos de un lado a otro, los protocolos simplemente organizan el intercambio de advertisements que contienen la información requerida para realizar cualquier funcionalidad.

1.2.1.7. Designación de Entidades

La mayoría de los objetos en una red P2P necesita algún segmento de información que los identifique en la red:

- **Peers:** Un peer necesita un identificador que otros peers pueden usar para localizarlo o direccionarlo en la red. Identificar a un peer en particular podría ser necesario para permitir el enrutamiento de los mensajes a través de un tercero hacia el peer correcto.
- **Peer groups:** Un peer necesita de alguna manera identificar cual peer group desea usar para realizar alguna acción. Las acciones podrían incluir la unión, hacer consultas, o dejar el peer group.
- **Pipes:** Para permitir que haya una comunicación, un peer necesita de alguna manera poder identificar un pipe que conecte los endpoints en la red.
- **Contenido:** Una parte del contenido debe ser definido como único para permitir a los peers reflejar el contenido en la red, proporcionando así un acceso redundante. Los peers pueden usar así este único identificador para encontrar contenido en cualquier peer.

En las redes P2P tradicionales, algunos de estos identificadores son usados en detalles específicos sobre el transporte en la red; por ejemplo, un peer podría identificarse por su dirección IP. Sin embargo, usar representaciones en sistemas dependientes es inflexible y no proporciona un sistema de identificación que sea independiente del sistema, de la operación o del transporte en la red. En la red P2P ideal, cualquier dispositivo debe ser capaz de participar, sin tener en cuenta su sistema operativo o el transporte en la red.

Un esquema de designación de entidades independiente del sistema, es un requisito indispensable para una red P2P flexible.

1.2.2. Comparación entre las Soluciones P2P Existentes

Usando los bloques básicos de las redes P2P definidos en este capítulo, es posible interpretar las soluciones propietarias P2P existentes, como Napster y Gnutella, o aun aplicaciones que no sean P2P, como la arquitectura cliente/servidor.

1.2.2.1. Napster

La red híbrida P2P de Napster, consiste en un servidor centralizado que realiza las búsquedas, podría ser modelado como un solo peer rendezvous y múltiples peers simples, todos usando TCP como red de transporte. El peer rendezvous proporciona a los peers simples la capacidad de localizar advertisements de archivos MP3 los cuales consisten del nombre del archivo, la dirección IP, e información sobre el puerto. Los peers simples usan esta información para conectarse directamente y descargar el archivo hasta su peer.

Napster no provee una completa solución para evitar los firewalls, y solo son capaces de atravesar un único firewall. Cada peer actúa como un router, capaz de enviar contenido a un peer protegido por un firewall cuando la petición es hecha a través de HTTP. Napster no proporciona capacidades para el enrutamiento de mensajes, lo que significa que un peer simple en la red no puede actuar como un peer router para habilitar a otros peers a traspasar un firewall doble.

1.2.2.2. Gnutella

En la red de Gnutella, cada peer actúa como un peer simple, un rendezvous peer, y un peer router, usando TCP para el transporte de los mensajes y HTTP para la transferencia de archivos.

Las búsquedas en la red son propagadas por un peer a todos sus peers vecinos conocidos, los cuales después propagan la petición a otros peers. Los advertisements de Gnutella consisten en una dirección IP, un numero de puerto, un número de índice que identifica el archivo en el peer host, y detalles del archivo como nombre y tamaño. Los peers de Gnutella no proporcionan todas las capacidades de los peers router, lo cual significa, que como Napster, los peers de Gnutella son capaces de cruzar sólo un firewall.

1.2.2.3. Cliente/Servidor

La arquitectura tradicional Cliente/Servidor puede ser interpretada en términos de los bloques de construcción P2P. Los clientes actúan como peers simples, y el servidor actúa como un peer rendezvous capaz de proporcionar advertisements que varían de acuerdo a la aplicación. No se proporciona ninguna capacidad para traspasar firewalls o NAT, ya que los servidores poseen direcciones IP reales y para ofrecer sus servicios estos deben

ser habilitados explícitamente en el firewall. La red de transporte usada depende de la aplicación.

II. EL PROYECTO JXTA

En la actualidad, lamentablemente, las aplicaciones P2P tienden a usar protocolos que son propietarios e incompatibles por naturaleza, reduciendo la ventaja ofrecida por los dispositivos de las redes P2P. Cada red forma una comunidad completamente cerrada, independiente de las otras redes e incapaz de impulsar sus servicios.

Hasta ahora, la emocionante exploración de aplicación de la tecnología P2P ha opacado la importancia de la interoperabilidad y reutilización de software para convertir a P2P en una plataforma madura de soluciones, los desarrolladores necesitan reenfocar sus esfuerzos de programación fundamentalmente a redes P2P para crear aplicaciones P2P sobre una base sólida y bien definida. Para hacer esto, los desarrolladores P2P necesitan un lenguaje común que le permita a los peers comunicarse y realizar las actividades básicas en una red P2P.

2.1. QUÉ ES EL PROYECTO JXTA

Teniendo en cuenta la necesidad de un lenguaje P2P común, Sun Microsystems creó el Proyecto JXTA (pronunciado yuxtapous o yuxta), un pequeño grupo de desarrolladores bajo la dirección de Bill Joy y Mike Clary, para diseñar una solución que sirviera a todas las aplicaciones P2P. En su interior, JXTA es simplemente un conjunto de especificaciones protocolares sobre las cuales se pretende soportar una extensa gama de aplicaciones P2P, de tal forma que cualquiera que desee producir una nueva aplicación se ahorre la dificultad de diseñar sus propios protocolos para manejar el núcleo de funciones de una comunicación P2P.

El nombre JXTA se deriva de la palabra "juxtapose" (Yuxtapuesto), que significa colocar dos entidades en paralelo o en proximidad, una al lado de la otra. Escogiendo este nombre, el equipo de desarrolladores de Sun reconoció que las soluciones P2P siempre existirían junto a las actuales soluciones cliente/servidor en lugar de reemplazarlas completamente.

Si bien es verdad que el modelo del proyecto JXTA se basa en el paradigma P2P, este proyecto implica mucho más que una iniciativa de este tipo de tecnología. De hecho, el proyecto JXTA facilita un mecanismo que permite crear y proporcionar una nueva y completa gama de servicios y aplicaciones, al tiempo que promete ampliar los servicios web desde ubicaciones fijas o centralizadas hasta cualquier punto de la red. Por ejemplo, los desarrolladores pueden utilizar el código del proyecto JXTA para crear aplicaciones y servicios que permitan a los clientes:

- Realizar búsquedas en toda la Web y en todos los dispositivos conectados (no sólo en los servidores) para encontrar la información que se necesita.
- Guardar archivos e información en los puntos distribuidos de la red, no sólo en las unidades de disco locales
- Conectar sistemas de juegos para que varios usuarios, ubicados en distintos lugares, puedan disfrutar del mismo juego de forma interactiva
- Participar en subastas entre grupos de usuarios seleccionados
- Colaborar en proyectos desde cualquier lugar, utilizando un dispositivo conectado
- Compartir servicios de computación, tales como procesadores o sistemas de almacenamiento, independientemente del lugar en el que estén ubicados físicamente los usuarios

El Proyecto JXTA no ha sido diseñado para sustituir las tecnologías y los modelos de computación actuales, sino para ampliarlos con el fin de satisfacer un nuevo conjunto de necesidades. "Hoy en día, la Web tiene más usuarios que nunca. Existe una gran proliferación de dispositivos y métodos de acceso. Hay más ancho de banda y poder de cómputo, al tiempo que los usuarios exigen acceso a mucho más contenido".

Los protocolos JXTA están diseñados para ser independientes del lenguaje de programación y de los protocolos de transporte. Los protocolos JXTA pueden ser implementados en JAVA, C/C++, Perl, y en muchos otros lenguajes de programación. Dichos protocolo se pueden basar en TCP/IP, HTTP, Bluetooth, Home PNA u otros protocolos de transporte.

JXTA como proyecto de software libre (Open Source)

Durante los últimos meses, el modelo de licencia "Open Source" sobre el cual se pone el Proyecto JXTA a disposición de los desarrolladores ha llevado a un notable enriquecimiento y refinamiento de la tecnología por parte de la comunidad del Proyecto JXTA, así como de los ingenieros de Sun.

Las licencias de la tecnología del Proyecto JXTA se otorgan utilizando un modelo open source similar al que emplea "Apache Software Foundation". Esto permite a los desarrolladores descargar tanto los binarios como el código fuente sin costo alguno, modificarlos y enriquecerlos, y volver a distribuir el código a otros miembros de la comunidad, siempre y cuando todos los archivos den crédito a quienes hayan colaborado y el nombre JXTA se use sólo con previa autorización.

Como prueba de que la comunidad del Proyecto JXTA está asumiendo como algo propio el desarrollo de la tecnología JXTA, considérense los siguientes puntos:

- La mayoría de las preguntas técnicas que se plantean en el sitio JXTA.org reciben respuesta de miembros de la comunidad que no pertenecen a Sun.
- La mayoría de los manuales de tutoría del sitio JXTA.org fueron aportados por miembros de la comunidad que no pertenecen a Sun.
- Muchos de los más amplios artículos y publicaciones acerca de la tecnología JXTA fueron escritos por miembros independientes de la comunidad.
- Instituciones académicas tienen investigadores que están usando o investigando la tecnología JXTA.

2.2. ARQUITECTURA JXTA

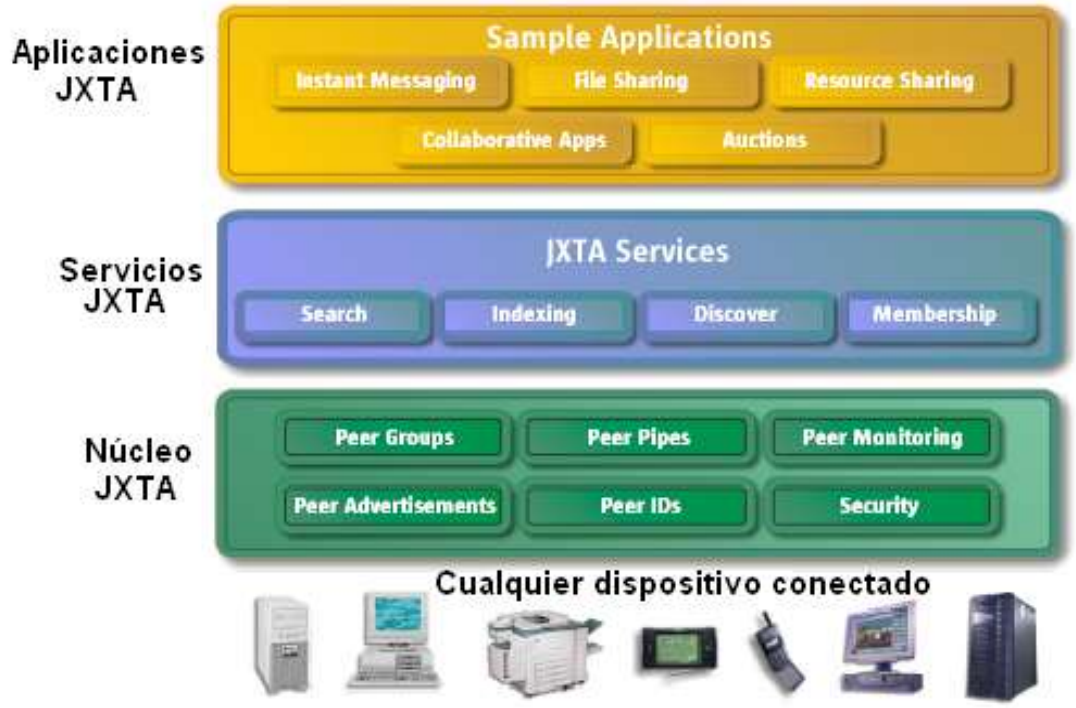


Figura 2.1 Arquitectura JXTA

- Nivel de Plataforma (Núcleo JXTA): El nivel de plataforma, también conocido como "Núcleo JXTA", encapsula las primitivas mínimas y esenciales para el trabajo en redes P2P. Esto incluye el soporte para las operaciones fundamentales de las aplicaciones P2P, incluyendo descubrimiento, transporte (manejo de firewalls), la creación de peers, peer groups, y las primitivas de seguridad asociadas.
Idealmente, los elementos de esta capa son compartidos por todas las soluciones P2P, constituyen el fundamento de JXTA. Todos los demás aspectos de una solución de P2P de JXTA en los servicios o las capas de aplicación se basan en esta capa para suministrar su funcionalidad.

Los elementos del núcleo JXTA son listados a continuación:

- Peers
- Grupos Peer
- Red de transporte (Tuberías, terminales, mensajes)
- Anuncios (Advertisements)

- Entidades de etiquetado (Identificadores)
- Protocolos (Descubrimiento, comunicación, monitoreo)
- Primitivas de autenticación y seguridad.

Esta capa incluye además seis protocolos básicos y esenciales provistos por JXTA. Aunque estos protocolos son implementados como servicios, están ubicados en la capa de plataforma y son designados como servicios de núcleo para distinguirlos de las soluciones de la capa de servicios.

- Nivel de Servicios JXTA: este nivel incluye los servicios de red, los cuales no son absolutamente necesarios para la operación de la red P2P, pero son comunes y ofrecen funciones deseables para un entorno P2P.
Entre los servicios de red más comunes están: búsqueda e indexado, directorio, sistemas de almacenamiento de información, intercambio de archivos, sistemas de archivos distribuidos, almacenamiento y alquiler de recursos, traducción de protocolos, autenticación, servicios de infraestructura de llave pública (*PKI*), entre otros.

- Nivel de Aplicación: este nivel incluye la implementación de aplicaciones integradas, tales como mensajería, intercambio de documentos y recursos, reparto y gestión de contenidos, sistemas de correo electrónico P2P, sistemas de subasta P2P, y muchos otras posibles aplicaciones P2P.

El límite entre servicios y aplicaciones no es rígido. Una "aplicación" para un cliente podría ser vista como un "servicio" para otro. El sistema en general está diseñado para ser modular, permitiendo a los desarrolladores escoger de una gran cantidad de aplicaciones y servicios lo que mejor se ajuste a sus necesidades.

Componentes JXTA

La red JXTA consiste en una serie de nodos interconectados, más conocidos como *peers.* Dichos peers se organizan en *peer groups* (Grupos de peers), los cuales proveen un conjunto de servicios en común. Por ejemplo, en algún grupo peer se podrían prestar servicios de intercambio de archivos, mensajería instantánea, subasta, entre otros.
Los peers en JXTA anuncian sus servicios en documentos XML llamados *advertisements.* Los advertisements les permiten a otros peers en la red JXTA conectarse e interactuar con los servicios que ofrece un peer en particular.

Los peers JXTA usan *pipes* (tuberías) para enviar *mensajes* hacia otros peers.
Las pipes son un mecanismo de transferencia de mensajes asíncrono y unidireccional usado por el servicio de comunicación. Los mensajes son simples documentos XML los cuales contienen información de enrutamiento, de identificación, o de seguridad. Las pipes están limitadas a *endpoints* (terminales) específicos, tales como un puerto TCP y una dirección IP asociada.

Existen tres aspectos esenciales en la arquitectura JXTA que la distinguen de los otros modelos de red distribuidos:

- El uso de documentos XML (advertisements) para describir los recursos de red.
- Abstracción de pipes a peers, y peers a endpoints sin depender de una autoridad de asignación de nombres tal como un DNS (Servicio de Nombres de Dominio).
- Un esquema uniforme de direccionamiento (PeerIDs).

2.3. PROTOCOLOS JXTA

La base de JXTA es un conjunto común de protocolos definidos por la comunidad JXTA. Estos protocolos pueden ser usados para construir aplicaciones. Diseñados con un overhead bajo, es decir, los protocolos no asumen nada acerca de la topología de red subyacente sobre la cual una aplicación se está ejecutando.
JXTA generalmente define seis protocolos. El número de protocolos que se implementan depende de las capacidades de los nodos. Los nodos también pueden extender o reemplazar algún protocolo, dependiendo de los requerimientos que se tengan en particular. Es importante aclarar que los protocolos JXTA no prometen interoperabilidad, los protocolos que conforman JXTA estandarizan la manera en la cual los nodos:

- Se encuentran unos a otros.
- Se organizan automáticamente en grupos.
- Publican y encuentran recursos.
- Se comunican unos con otros.
- Se monitorean unos a otros.

La especificación de Protocolos de JXTA v2.0 define los fundamentos del networking P2P (Ver Figura 2.2):

- **Peer Resolver Protocol (PRP – Protocolo de Resolución de Peer):** es el mecanismo mediante el cual un peer envía una petición a uno o más peers, y recibe una respuesta (o múltiples respuestas) a dicha petición. PRP implementa un protocolo petición/respuesta. El mensaje de respuesta se produce de acuerdo

a un identificador único incluido en el cuerpo del mensaje. Las consultas pueden ser dirigidas a un grupo entero o a un peer en especial.

- **Peer Discovery Protocol (PDP – Protocolo de Descubrimiento):** a través de este protocolo un peer puede anunciar todos sus recursos, y descubrir los recursos que otros peers ofrecen (Pipes, peer groups, servicios…). Cada recurso se describe y se publica usando un advertisement. Los advertisements son metadatos independientes de un lenguaje de programación que describen recursos de red. Los advertisements se representan mediante documentos XML.
- **Peer Information Protocol (PIP – Protocolo de Información):** es el mecanismo que le permite a un peer obtener información sobre el estado de otros peers. Esta información puede incluir tiempo on line (uptime), carga de tráfico, capacidades, estado, entre otros datos.
- **Pipe Binding Protocol (PBP – Protocolo de Asociación de Tuberías):** es el mecanismo mediante el cual un peer puede establecer un canal de comunicación virtual o tubería (*pipe*) entre uno o más peers. Los peers usan el PBP para asociar dos o más terminales a una conexión (terminales de pipe). Las pipes son el mecanismo fundamental para lograr comunicación entre peers.
- **Endpoint Routing Protocol (ERP – Protocolo de Enrutamiento de Terminal):** es el mecanismo mediante el cual un peer puede descubrir una ruta (secuencia de saltos) para enviar mensajes a otro peer. Si un peer "A" quisiera enviar un mensaje a un peer "C", y no hay una ruta directa entre "A" y "C", entonces el peer "A" necesita encontrar peers intermediarios que enruten sus mensajes hacia "C". El ERP se usa para determinar información acerca de dicha ruta. Si la topología de red cambia y las rutas previamente establecidas ya no están disponibles, los peers pueden usar ERP para encontrar una ruta alterna al destino.
- **Rendezvous Protocol (RVP – Protocolo de Punto de Encuentro):** es el mecanismo mediante el cual los peers se registran o se retiran de un servicio de difusión. Dentro de un peer group, los peers pueden ser peer rendezvous o peers que escuchan peer rendezvous. El RVP permite a un peer enviar mensajes a todos los clientes del servicio rendezvous. El PRP (Peer Resolver Protocol) y el PBP (Pipe Binding Protocol) usan al RVP cuando necesitan propagar sus propios mensajes de protocolo.

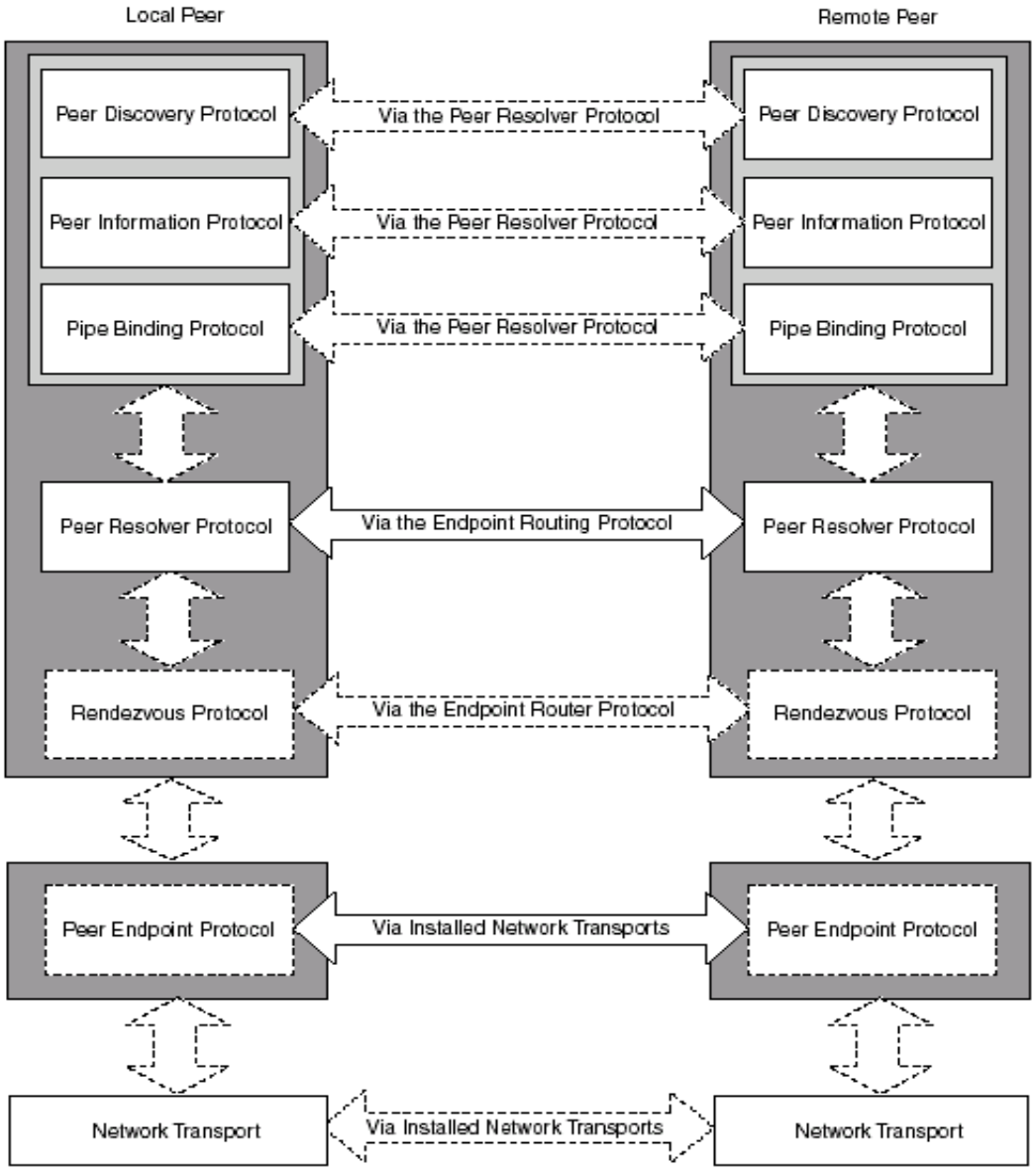


Figura 2.2. Pila de protocolos JXTA

2.4. CONCEPTOS BÁSICOS EN JXTA

En esta sección se detallarán los componentes fundamentales de la plataforma JXTA para tener un mejor entendimiento de cómo está constituida y de su forma de operación. Se especifica la forma en que JXTA implementa los conceptos P2P detallados en el apartado 1.2.1 *"Conceptos P2P".*

2.4.1. Peers

Un peer es cualquier dispositivo interconectado que implementa uno o más protocolos JXTA. Peers pueden ser sensores, teléfonos, PDAs, también como lo pueden ser computadores, servidores y supercomputadoras. Cada peer opera de manera asíncrona e independiente de otros peers y posee un identificador único, conocido como PeerID. Los peers publican una o más interfaces de red para usar los protocolos JXTA, cada una de estas interfaces es publicada como un *peer endpoint* , el cual identifica de manera exclusiva su interfaz de red. Los peers usan sus *endpoints* para establecer conexiones punto a punto entre sí. Sin embargo, los peers no necesariamente tienen que interconectarse directamente ya que se pueden utilizar peers intermedios para enrutar los mensajes hacia un destino que se encuentra aislado debido a disposiciones de red físicas que lo impiden o a causa de diversas configuraciones de red (Por ejemplo: proxy, NATs, Firewalls)

Los peers se configuran mediante un descubrimiento espontáneo de los otros peers en la red formando relaciones transitorias o persistentes llamadas *peer groups* (Grupos de peers).

2.4.2. Peer Groups (Grupos peer)

Un *peer group* es una colección de peers que han acordado un grupo común de servicios. Los peers se organizan automáticamente en peer groups, cada uno identificado por un ID único de grupo. Cada peer group puede establecer sus propias políticas de acceso y privacidad, desde abierta (cualquiera puede añadirse al grupo) hasta altamente segura y protegida (Solicitud de suficientes credenciales para el ingreso al grupo).

Los peers podrían pertenecer a más de un peer group simultáneamente, por defecto, el primer grupo que es instanciado y al cual pertenecen todos los peers es el *NetPeerGroup.* Los peers pueden ingresar a cualquier otro grupo disponible.

Existen tres diferentes motivaciones para la creación de un grupo:

- **Crear un entorno seguro:** los grupos crean un dominio local de control en el cual se aplica una política de seguridad específica. Dicha política podría ser tan simple como una autenticación en texto plano de login y password, o tan sofisticada como la criptografía de llave pública. Los límites del peer group permiten acceder a los peers miembros a contenidos protegidos. Los peer groups forman regiones lógicas cuyos límites restringen el acceso a los recursos del peer group.

- **Delimitar un entorno:** Los grupos permiten establecer dominios locales de especialización. Por ejemplo, los peers podrían unirse para implementar una red de CPU compartida o de intercambio de archivos. El peer group sirve para subdividir la red en regiones abstractas y ofrecer un mecanismo de delimitación de áreas especializadas. Los límites del peer group definen el alcance de búsqueda cuando se trata de encontrar algún contenido en especial.
- **Crear un entorno de monitoreo:** Los peer groups permiten a los peers monitorear un conjunto de peers con algún propósito en especial, por ejemplo, ver su estado (activo/desconectado), introspección de tráfico, y características en general.

Los grupos también forman relaciones jerárquicas, en las cuales cada grupo tiene un solo padre. Cuando se realicen búsquedas de contenidos o recursos, dichas solicitudes de búsqueda se propagan dentro del grupo, el advertisement para el grupo se publica en el grupo padre y además al interior de sí mismo. Un peer group ofrece un conjunto de servicios llamado *peer group services.* JXTA define un conjunto básico de servicios pero se pueden desarrollar más servicios para añadir mayor funcionalidad a una aplicación en especial que lo requiera.

Con el propósito de que dos peers interactúen a través de un servicio, ellos deben ser parte del mismo peer group.

2.4.3. Servicios de Red

Los peers cooperan y se comunican para publicar, descubrir e invocar servicios de red. Los peers pueden publicar múltiples servicios de red, cabe recordar que dichos servicios de red se descubren mediante el PDP – Peer Discovery Protocol.

Los protocolos JXTA manejan dos niveles de servicios de red:

- Servicios Peer: son los ofrecidos por cada peer en particular, lo que implica que si el peer llega a fallar, el servicio ya no estaría disponible tampoco. Múltiples instancias del mismo servicio pueden ejecutarse en diferentes peers, pero cada instancia publica su propio advertisement.
- Servicios de group Peer: este tipo de servicio está compuesto por una colección de instancias de un servicio ejecutándose en múltiples miembros de un peer group. Si cualquiera de los miembros llegase a fallar, el servicio ofrecido por el colectivo no se afecta (asumiendo que el servicio está aún disponible en otros

miembros del grupo).Los servicios del grupo peer son publicados como parte del advertisement. Los servicios pueden ser ya sea pre-instalados en un peer, o cargados desde la red. Con el propósito de ejecutar un servicio, un peer tendría que localizar una implementación adecuada para su entorno de ejecución. El proceso de encontrar, descargar e instalar un servicio desde la red es similar a realizar una búsqueda en Internet de una página, descargar la página y luego instalar el plugin requerido para su respectivo despliegue.

El conjunto básico de servicios que ofrece JXTA consiste en:

- Servicio de descubrimiento (Discovery Service): Este servicio es usado por los miembros del grupo para buscar recursos, tales como peers, pipes, otros peer groups y servicios.
- Servicio de membresía (Membership Service): este servicio es usado por los miembros del grupo para rechazar o aceptar el acceso de algún peer al grupo. Un peer que desee ingresar a un grupo primero debe localizar a un miembro actual de dicho grupo y luego realizar una solicitud de aceptación. La solicitud podría ser aceptada o rechazada por los miembros actuales, mediante una una votación que realizarían todos los miembros del grupo o un sector representativo de ellos para las nuevas solicitudes de ingreso.
- Servicio de acceso (Access Service): este servicio se usa para validar las solicitudes hechas por un peer a otro. El peer que realiza la solicitud debe mostrar sus respectivas credenciales y la información requerida para determinar si se le permite o no el acceso. Sin embargo, cabe anotar que no todas las operaciones dentro de un peer group necesitan ser chequeadas con el servicio de acceso.
- Servicio de Pipes (Pipe Service): este servicio es utilizado para crear y gestionar las conexiones entre los miembros del peer group.
- Servicio Resolver (Resolver Service): este servicio se usa para enviar peticiones genéricas a otros peers. Los peers pueden definir e intercambiar peticiones para encontrar cualquier información que pueda necesitarse (Por ejemplo, el estado de un servicio, o el estado de un pipe endpoint).
- Servicio de monitoreo (Monitoring Service): este servicio es usado para permitir a un peer monitorear otros miembros del mismo grupo.

No todos estos servicios se tienen que implementar en todos los peer groups, un peer group está en la libertad de implementar solamente los servicios que le sean de utilidad y de depender del grupo por defecto, el NetPeerGroup, para proveer implementaciones genéricas de servicios fundamentales no críticos.

2.4.4. Módulos

Los módulos en JXTA son una abstracción usada para representar una pieza de "código" usado para implementar alguna característica en el mundo JXTA. Los servicios de red son el ejemplo más común de estos casos. El módulo abstraído no especifica qué es este "código": este puede ser una clase java, un archivo jar, una librería dinámica DLL, un conjunto de mensajes XML, o un script. La implementación del comportamiento del módulo se deja a quienes implementan el módulo. De esta manera, los módulos pueden ser usados para representar diferentes implementaciones de un servicio de red en diferentes plataformas, tal como la plataforma Java, Microsoft Windows, o el sistema operativo Solaris.

Los módulos proveen una abstracción genérica que permite a un peer instanciar un nuevo comportamiento. Por ejemplo, en el caso de ingresar a un peer group, un peer podría tener que aprender un nuevo servidor de búsqueda que sea usado únicamente en ese peer group. Con el propósito de unirse a este peer group el peer debe instanciar este nuevo servicio, el framework de módulos permite la representación y el anuncio de comportamientos independientes de la plataforma, y permite a los peers describir e instanciar cualquier tipo de implementación de comportamiento.

Por ejemplo, un peer tiene la habilidad de instanciar ya sea un comportamiento implementado en Java o C. La habilidad de describir y publicar comportamientos independientes de la plataforma es esencial para soportar peer groups compuestos por peers heterogéneos. Los advertisements de módulo permiten a los peers JXTA describir un comportamiento sin importar la plataforma. La plataforma JXTA usa advertisements de módulo para auto describirse.

La abstracción de módulo incluye la definición de una clase de módulo, una especificación del módulo, y una implementación del módulo:

- **Clase de módulo:** se usa para anunciar la existencia de un comportamiento o característica. La definición de clase representa un comportamiento esperado y una asociación esperada para soportar el módulo. Cada módulo de clase es identificado por un ID único, llamado *ModuleClassID.*

- Especificación de Módulo: usado para acceder a un módulo, contiene toda la información necesaria para acceder o invocar cualquier módulo. Por ejemplo, en el caso de un servicio, la especificación del módulo podría contener el anuncio de un pipe para ser usado en la comunicación con el servicio.
Una especificación de módulo es un método usado para proveer la funcionalidad que la clase de un módulo implica.
Puede haber múltiples especificaciones de módulo para una determinada clase. Cada especificación de módulo es identificada por un ID único, el *ModuleSpecID.* El *ModuleSpecID* contiene el *ModuleClassID,* es decir el *ModuleClassID* está embebido en el *ModuleSpecID,* indicando la clase de módulo asociada.
Una especificación de módulo implica compatibilidad en la red. Todas las implementaciones de un módulo deben usar los mismos protocolos y ser compatibles, aunque estén escritas en un lenguaje diferente.

- **Implementación de módulo:** es la implementación de una determinada especificación de módulo. Pueden existir múltiples implementaciones de una misma implementación de módulo. Cada implementación de módulo contiene el *ModuleSpecID* de la especificación que implementa.

Los módulos son usados por los servicios de los peer groups, y pueden también ser usados por servicios stand alone. Los servicios JXTA pueden usar la abstracción de módulo para identificar la existencia del servicio (su Clase de módulo), la especificación del servicio (Su especificación de módulo), o una implementación del servicio (Una implementación del módulo). Cada uno de estos componentes tiene asociado un advertisement, el cual puede ser publicado y descubierto por otros peers JXTA.
Por ejemplo, considere el servicio JXTA Discovery (Servicio de descubrimiento), el cual tiene un *ModuleClassID,* identificándolo como un servicio de descubrimiento, su funcionalidad abstracta. Pueden haber múltiples especificaciones del servicio discovery, posiblemente incompatibles entre sí, una podría usar estrategias que se ajusten exactamente al tamaño de un grupo y a su disposición en la red, mientras que otro experimenta con nuevas estrategias. Cada especificación tiene un solo *ModuleSpecID* el cual referencia el servicio de descubrimiento, *ModuleClassID.* Para cada especificación, puede haber múltiples implementaciones, cada una de las cuales contiene el mismo *ModuleSpecID.*

En resumen, puede haber múltiples especificaciones de un módulo de clase, implementaciones, las cuales podrían ser incompatibles. Sin embargo, todas las implementaciones de cualquier especificación se asume que son compatibles.

2.4.5. Pipes (Tuberías)

Los peers JXTA usan *pipes* para enviar mensajes a otros. Las pipes son simplemente un mecanismo asíncrono y unidireccional para la transferencia de mensajes usado en el servicio de comunicación. Las pipes no discriminan el contenido que transportan ya que soportan la transferencia de cualquier objeto, desde código binario, cadenas de datos, y cualquier otro objeto que java pueda soportar.

Los *pipe endpoints* (terminales de tubería), pueden ser tanto de entrada (*input pipes*), para la lectura de datos proveniente de otra fuente, como de salida (*output pipes*) para el envío de datos a cualquier destino. Los *pipe endpoints* corresponden a las interfaces (Por ejemplo puertos TCP asociados a una dirección IP) que cada peer pone a disposición para el intercambio de mensajes con otros. Los pipes pueden tener endpoints conectados a diferentes peers en diferentes momentos, o podrían no estar conectados a ninguno.

Los pipes son canales virtuales de comunicación, pueden conectar peers que no tienen un enlace físico directo. En este caso, uno o más peers intermedios deben intervenir como repetidores de los mensajes que se desean intercambiar entre los dos endpoints de un pipe.

Los pipes ofrecen dos modos de comunicación, punto a punto y difusión, como se pueden ver en la gráfica 2.3. El núcleo de JXTA también ofrece pipes unicast seguras, una variante de la pipe punto a punto.

- **Pipes punto a punto (Unicast pipes):** este tipo de pipe conecta dos endpoints de una pipe: una pipe de entrada por la que el peer recibe mensajes provenientes de una pipe de salida desde otro peer.
- **Pipes de difusión (Propagate pipes):** pipe que conecta una pipe de salida con múltiples pipes de entrada. Los mensajes fluyen desde la pipe de salida (output pipes), la fuente de difusión, hacia las pipes de entrada (input pipes). Todas las difusiones se hacen dentro de los límites de un peer group. Es decir, *la pipe de salida y todas las pipes de entrada deben pertenecer al mismo grupo.*
- **Pipes unicast seguras (Secure Unicast Pipes):** este es un tipo de pipe punto a punto que provee un canal de comunicación seguro. Se pueden construir otros tipos de pipes a partir de los tipos básicos.

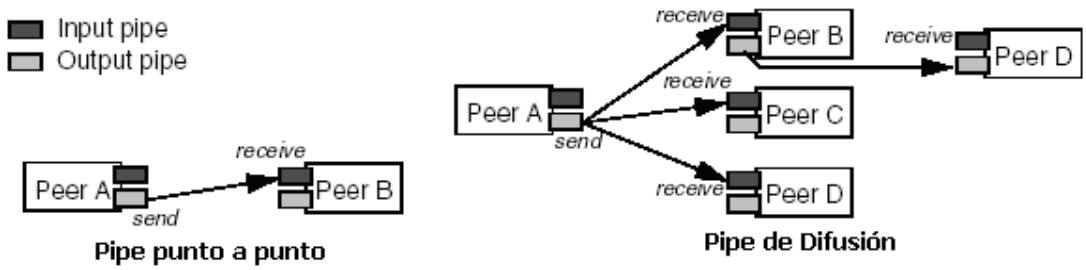


Figura 2.3. Pipes de difusión y pipes punto a punto

2.4.6. Mensajes

Un mensaje es un objeto enviado entre dos peers JXTA; es la unidad básica de intercambio de datos entre dos peers. Los mensajes se envían a través del servicio Pipe y del servicio Endpoint. Generalmente, las aplicaciones usan el servicio pipe para crear, enviar y recibir mensajes. (No se espera que las aplicaciones tengan que hacer uso del servicio endpoint directamente, pero puede usarse en caso de que la aplicación necesite entender o controlar la topología de la red JXTA).

Un mensaje es una secuencia ordenada de contenidos con nombre y tipo llamados *Elements.* De esta manera, un mensaje (*Message*) es un conjunto de pares nombre/valor con un contenido que puede ser de cualquier tipo.

Los protocolos JXTA son un conjunto de mensajes intercambiados entre peers. Cada plataforma asociada describe cómo se traduce un mensaje a y desde una estructura de datos nativa como un objeto Java o como una estructura C.

Hay dos representaciones para los mensajes: XML y binario. La plataforma JXTA para J2SE usa un formato binario para encapsular el contenido del mensaje. Los servicios pueden usar el formato más apropiado para el transporte de datos. Por ejemplo, un servicio que requiera una representación compacta para los mensajes puede usar la representación binaria, mientras que otros servicios pueden usar XML. Los datos binarios podrían ser codificados usando el esquema Base64 en el cuerpo del mensaje XML. El uso de mensajes XML en la definición de protocolos permite que diferentes clases de peers participen en un protocolo.

Ya que los datos están etiquetados, cada peer está en la libertad de implementar el protocolo de la manera que mejor se ajuste a sus capacidades y a su rol. Si un peer solo necesita un subconjunto de los mensajes de protocolo, las etiquetas de los datos en XML permiten que el peer identifique qué partes del mensaje son de su interés. Por ejemplo, un peer congestionado y sin suficiente capacidad para procesar alguna o la mayor parte

de un mensaje puede usar las etiquetas de los datos del mensaje para procesar las partes que sí alcanza a procesar e ignorar el resto.

2.4.7. Advertisements (Anuncios)

Todos los recursos de red en JXTA, tal como peer, peer groups, pipes y servicios, son representados por un *advertisement.*

Los advertisements son metadatos escritos en un lenguaje neutral representados en documentos XML. Los protocolos JXTA usan advertisements para describir y publicar la existencia de un recurso en la red P2P. Los peers descubren recursos mediante la búsqueda de sus correspondientes advertisements, los cuales pueden ser puestos en caché una vez descubiertos para registrar así su localización.

Cada advertisement es publicado con un temporizador (*lifetime*), el cual especifica la disponibilidad de sus recursos. Los temporizadores permiten la eliminación de recursos obsoletos sin necesidad de un control centralizado. Un advertisement puede ser publicado nuevamente, antes de que el temporizador original expire, para extender el temporizador de un recurso. Los protocolos JXTA definen los siguientes tipos de advertisements:

- **Advertisement de peer (Peer Advertisement):** describe el recurso de peer, se usa principalmente para mantener información específica sobre peers, tal como el nombre, peerID, endpoints disponibles y cualquier otro atributo característico del peer que se quiera publicar dentro del peer group.
- **Advertisement de peer group (Peer Group Advertisement):** describe recursos de un peer group específico, tal como nombre, peer group ID, descripción, especificación y parámetros de servicio.
- **Advertisement de Pipe (Pipe advertisement):** describe una pipe, el servicio pipe lo utiliza para crear los endpoints asociados a la tubería de salida y de entrada. Cada advertisement de pipe contiene un ID simbólico opcional, un tipo (Punto a punto, de difusión, seguro, etc..) y un único pipe ID.
- **Advertisement de Clase de Módulo (Module Class Advertisement):** describe una clase de módulo, su propósito esencial es formalizar la existencia de una clase de módulo. Esto incluye nombre, descripción, y un ID único, el *ModuleClassID.*
- **Advertisement de Especificación de Módulo (Module specification Advertisement):** define una especificación de módulo, su propósito es proveer

referencias a la documentación necesaria para crear implementaciones conformes a la especificación. Un uso secundario, es crear instancias utilizables de forma remota, mediante la publicación de información tal como el advertisement de pipe, esto incluye el nombre, un ID único (*ModuleSpecID*), el advertisement de pipe y el campo de los parámetros el cual contiene parámetros arbitrarios para ser interpretados por cada implementación.

- **Advertisement de Implementación de módulo (Module Implementation Advertisement):** define una implementación de una especificación de módulo determinado. Incluye nombre, ModuleSpecID asociado, también el código, paquete y parámetros los cuales permiten a un peer obtener los datos necesarios para ejecutar la implementación.
- **Advertisement de punto de encuentro (Rendezvous Advertisement):** describe un peer que actúa como un peer rendezvous para un peer group.
- **Advertisement de Información de Peers (Peer Information Advertisement):** da información acerca de los recursos de un peer. El uso principal de este advertisement es mantener información específica sobre el estado actual de un peer, tal como tiempo on line, contéo de mensajes entrantes y salientes, hora del último mensaje recibido, y el último mensaje enviado.

Cada advertisement es representado por un documento XML. Los advertisements se componen de una serie de elementos organizados jerárquicamente. Cada elemento puede contener sus datos o elementos adicionales. Un elemento puede también tener atributos, indicado en parejas "nombre-atributo/valor-atributo". Un atributo es usado para almacenar metadatos, los cuales ayudan a describir los datos dentro del elemento.

Un advertisement JXTA puede tener la siguiente forma:

```
<?xml version="1.0"?>

<!DOCTYPE jxta:PipeAdvertisement>

<jxta:PipeAdvertisement xmlns:jxta="http://jxta.org">
        <Id>
            urn:jxta:uuid-
596162616461626 14E504720503250338E3E786229EA460DADC1A176B69B731504
        </Id>
        <Type>
            JxtaUnicast
        </Type>
        <Name>
            TestPipe.end1
        </Name>
</jxta:PipeAdvertisement>
```

La especificación completa de los advertisements JXTA es dada en "*JXTA Protocols Specification*"[2]

2.4.8. Seguridad

Las redes P2P dinámicas tal como la red JXTA necesitan soportar diferentes niveles de acceso a los recursos. Los peers JXTA operan en un modelo basado en confianza, en el cual un peer actúa bajo la autoridad otorgada por otro peer declarado como confiable para desarrollar alguna tarea en especial.

La seguridad consta de 5 condiciones:

- Confidencialidad: garantizar que el contenido de un mensaje no se muestra a individuos no autorizados.
- Autenticación: garantizar que el remitente es quien dice ser.
- Autorización: garantizar que el remitente está autorizado para enviar un mensaje.
- Integridad de los datos: garantizar que el mensaje no ha sido modificado en su viaje del origen al destino.
- Refutabilidad: garantizar que el mensaje fue transmitido por un remitente identificado apropiadamente y no es la repetición de un mensaje transmitido previamente.

Los mensajes XML tienen la capacidad de adicionar metadatos tales como credenciales, certificados, digests y llaves públicas a los mensajes JXTA, haciendo

[2] Sun Microsystems; "JXTA Protocols Specification"; http://spec.jxta.org/nonav/v1.0/docbook/JXTAProtocols.pdf

posible la implementación de estas características de seguridad y su utilización en la red JXTA. Los digests de los mensajes garantizan la integridad de los mismos.

Los mensajes pueden ser también encriptados (usando llaves públicas) y firmados (usando certificados) para confidencialidad y refutabilidad. Las credenciales pueden ser usadas para permitir la autorización y autenticación de mensajes.

Una credencial es un token (ficha) usado para identificar al remitente, y puede ser usada para verificar si un remitente tiene permisos suficientes para enviar un mensaje a un endpoint determinado. La credencial es un token que debe ser presentado cada vez que se envía un mensaje. La dirección desde la que se envía el mensaje, la cual está contenida en el mensaje JXTA, está asociada a la identidad del remitente en la credencial. Cada implementación de la credencial se especifica como una configuración de fácil manejo, la cual permite múltiples configuraciones de autenticación coexistentes en la misma red.

Este es el intento de los protocolos JXTA para lograr compatibilidad con una gran cantidad de mecanismos de seguridad a nivel de transporte para arquitecturas basadas en mensajes, tal como SSL (Secure Socket Layer) y IPSec (Internet Protocol Security).

Sin embargo, protocolos de transporte seguros, como SSL e IPSec, solo garantizan la integridad y confidencialidad del mensaje transferido entre dos peers. Con el propósito de tener comunicaciones seguras en una red de múltiples saltos como JXTA, se debe establecer una relación de confianza con todos los peers intermediarios. La seguridad se encuentra comprometida si cualquiera de los enlaces de comunicación no es seguro.

2.4.9. IDs (Identificadores)

Los peer, peer groups, pipes y otros recursos JXTA necesitan tener una identificación única. Un ID JXTA identifica exclusivamente una entidad y sirve como una forma canónica de referirse a dicha entidad. Actualmente, hay seis tipos de entidades JXTA, la cuales tienen definidos sus respectivos tipos de identificación: peers, peer groups, pipes, contenidos, definiciones de clase de módulo y especificaciones de módulo.

Los IDs JXTA se presentan en forma de texto.

Un ejemplo de Peer ID JXTA podría ser:

urn:jxta:uuid-2614A78746150325033F3BC76FF13C2414CBC0AB663666D03

Un ejemplo de Pipe ID JXTA puede ser:

urn:jxta:uuid-4E504720503250338E3E786229EA460DADC1A176B69B731504

Se generan IDs únicos aleatoriamente por la plataforma JXTA. Hay dos IDs JXTA especialmente reservados:
El ID nulo y el ID del NetPeerGroup (grupo Universal al que todos los peers pertenecen por defecto).

La terminación de los ID JXTA define qué tipo de recurso especifica, por ejemplo, se tiene una terminación de 02 para los peerGroupID, 03 para los peerID, y 04 para los pipeID.

III. EL PROYECTO JXME (JXTA para J2ME)

El propósito del proyecto JXME es facilitar la compatibilidad entre la red JXTA y los dispositivos móviles con soporte para J2ME enmarcados dentro de la especificación del CLDC 1.0 (Configuración de Dispositivos Limitados Interconectados) y el MIDP 2.0/1.0 (Perfil de Dispositivos Móviles de Información).

Gracias a JXME, cualquier dispositivo MIDP está en capacidad de participar en redes P2P con otros dispositivos MIDP y además usar el soporte que ofrece JXTA para que pueda participar en actividades P2P con peers ubicados en computadores de escritorio, estaciones de trabajo y servidores.

JXME implementa las capacidades de búsqueda de peers, búsqueda e ingreso a peer groups y localización de pipes. La tecnología JXME ha sido probada en dispositivos PalmOS, teléfonos móviles con soporte MIDP 2.0/1.0 CLDC 1.0 y en diversos simuladores J2ME, lo que facilita la verificación de prototipos desarrollados con la tecnología y la puesta en funcionamiento en entornos reales.
J2ME es una versión de Java reducida a la mínima expresión, adecuada para pequeños dispositivos con capacidades limitadas, tales como teléfonos celulares, handhelds, smartphones y PDAs. Mientras se considere J2ME como un peer JXTA, se deben tener en cuenta las siguientes limitaciones importantes en estos dispositivos:

- J2ME no tiene ninguna clase o API de procesamiento XML. Por lo tanto los requerimientos de procesamiento XML de JXTA son un poco truculentos de manejar. Otros motores de procesamiento XML de terceros, tales como kXML están disponibles y pueden ser usados para procesamiento XML en aplicaciones JXTA. Sin embargo, esto resulta en una aplicación sobrecargada, ocupando mucha de la memoria del dispositivo y tiempo de procesamiento, sin dejar suficiente memoria para otros procesos. Es razonable asumir que J2ME necesita actuar como un peer JXTA sin ninguna asistencia XML.

- Un dispositivo J2ME solo puede enviar solicitudes HTTP pero no puede abrir puertos para escuchar y recibir solicitudes HTTP entrantes.

3.1. FUNCIONAMIENTO

Teniendo en cuenta los aspectos mencionados hasta aquí, véase a continuación una detallada especificación de la comunicación de los dispositivos J2ME con la red JXTA y una introducción al desarrollo de aplicaciones JXTA para dispositivos móviles J2ME.

Para empezar, obsérvese en la figura 3.1 cómo un relay JXTA ayuda a los terminales basados en J2ME a interactuar con la red JXTA.

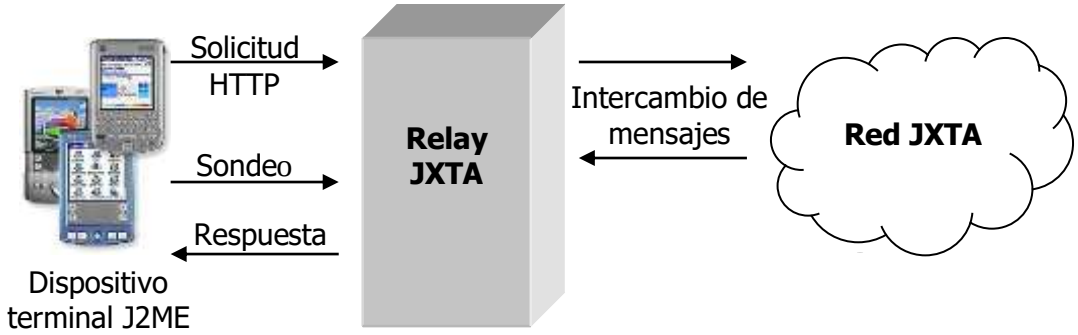


Figura 3.1. Interacción entre dispositivos J2ME con la red JXTA a través de un relay.

Un peer J2ME envía peticiones HTTP a un relay JXTA, el *mensaje* de petición contiene uno o más *elementos*, donde cada elemento está compuesto de una pareja "nombre – valor" viajando como parte de la cabecera de una petición HTTP.

Cuando el relay JXTA recibe una petición desde un peer J2ME, analiza cada par "nombre – valor" en la petición HTTP y genera los mensajes XML de acuerdo con el formato JXTA para ejecutar los comandos que recibe desde el cliente móvil sobre la red P2P.

El caso contrario, cuando un relay JXTA recibe un mensaje desde la red JXTA y el destino es un peer J2ME, analiza el formato del mensaje entrante y produce una respuesta HTTP correspondiente a dicho mensaje. Sin embargo, un relay JXTA no tiene forma de enviar la respuesta HTTP al peer J2ME (como se había mencionado anteriormente, los dispositivos J2ME no tienen funcionalidad de servidor HTTP y por ende no pueden abrir puertos para escuchar mensajes de peticiones entrantes[3]). Por lo tanto, un relay JXTA esperará que el peer J2ME envíe una petición de sondeo para enviarle una respuesta a dicha solicitud.

[3] En la actualidad se adelantan trabajos sobre la versión *JXME proxyless* (JXME sin proxy) para dispositivos CLDC1.0/MIDP2.0 los cuales cuentan con mayores capacidades y pueden trabajar con mayor autonomía en una red JXTA.

Cabe anotar que JXTA ha definido los protocolos de comunicación entre un host relay y un cliente de tal manera que el relay no solo sirva a clientes J2ME sino también a cualquier clase de dispositivo que se pueda comunicar implementando estos protocolos.

Ahora, a través de toda esta sección se enfatizará en la comunicación de los peers J2ME con la red JXTA, se estudiará a fondo el formato general de los mensajes JXME y todos los procesos asociados a la generación e intercambio de mensajes de protocolo.

3.1.1. Estructura de los Mensajes JXME

Figura 3.2 Representación de un mensaje JXME

Como se puede ver, un mensaje JXME contiene básicamente dos cosas, un encabezado y un cierto número de elementos que dan al mensaje su significado.

Tanto el encabezado como los elementos del mensaje JXME contienen varios campos, los cuales se discutirán más adelante. Para esto obsérvese la representación textual del mensaje JXME de la Figura 3.2 representado en la Figura 3.3.

```
jxmg    0    01   05  proxy                        07
jxel    2    0    07  request                      0006  create
jxel    2    0    04  name                         0004  test
jxel    2    0    09  requestId                    0001  1
jxel    2    0    04  type                         0004  PIPE
jxel    2    0    03  arg                          0007  Unicast
jxel    1    0    26  EndpointDestinationAddress   xxxx  destination address
jxel    1    0    21  EndpointSourceAddress        xxxx  source address
```

Figura. 3.3. Lista de todos los elementos en un mensaje JXME

En la práctica se ubicarían todos los datos en una sola línea, pero para un mejor entendimiento de la estructura de los mensajes JXME, el encabezado y todos los elementos del mensaje se muestran en líneas separadas.

Se puede comparar la Figura 3.2 con la lista de la Figura 3.3 para encontrar los valores de los diferentes campos. Por ejemplo, la primera línea en la Figura 3.3. corresponde al encabezado del mensaje, los demás son el contenido del mensaje, los elementos, los cuales se explican a continuación.

1. Todos los mensajes JXME inician con la cadena *jxmg*. Esta es la "*Firma del Mensaje",* mostrada en el encabezado de la Figura 3.2.
2. El "*0*" después de *jxmg* especifica la versión del mensaje.
3. El tercer campo del encabezado es un valor de dos dígitos, "01" en la lista de la figura 3.3, este valor especifica el número de namespace*s* que este mensaje JXME maneja (El namespace se especificará más adelante, en el cuarto campo del encabezado). Los diferentes elementos del mensaje están clasificados dentro de algún namespace y cada uno de estos puede pertenecer a un namespace diferente. El concepto de namespace en JXTA es el mismo que el que se tiene en XML: cada elemento en un mensaje JXME pertenece a un cierto namespace*,* si no es así, entonces pertenece a un namespace predeterminado o a un namespace vacío.
4. El cuarto campo en el encabezado declara los namespace*s* usados por el mensaje actual. El campo de declaración de namespace empieza con un valor de dos bytes (05 en la lista de la figura 3.3), el cual especifica el número de caracteres en el nombre del primer namespace (proxy). El nombre del namespace actual (proxy) sigue al valor entero de dos bytes.

Ya que se puede tener cualquier cantidad de namespaces en un mensaje JXME, el cuarto campo puede contener cualquier cantidad de namespaces declarados. Cada declaración es una pareja de valores y una cadena que especifica el nombre del namespace.

Nótese que el orden de los namespaces en el encabezado es importante. Al primer namespace (proxy en la lista de la figura 3.3) se le ha asignado automáticamente un identificador "*2".* Todas las entradas subsecuentes obtienen un identificador con un incremento en uno del identificador precedente. Por ejemplo, si se crea otro namespace, por ejemplo namespace*prueba,* este tendrá 3 como identificador.

El identificador *0* y *1* son preasignados al namespace vacío y al namespace *jxta* respectivamente.

Nota: El uso de identificadores de namespaces en los elementos de los mensajes JXME y el mapeo de sus nombres a identificadores enteros reduce el tamaño del mensaje y optimiza el uso del ancho de banda en las comunicaciones inalámbricas.

5. Los últimos dos dígitos (07) especifican el número de elementos (Objetos de la clase Element) que hay en el mensaje. Como se puede ver en la lista de la figura 3.3, se tienen siete elementos después del encabezado del mensaje. Como ya se ha visto, la lista de la figura 3.3 incluye siete elementos, cada elemento contiene cierta información y todo el conjunto de los diferentes elementos tiene un significado especial para el destinatario del mensaje. Por ejemplo, esta información podría ser una petición para crear un grupo o para unirse a uno. A continuación se especifican los diferentes campos de cada elemento.

3.1.1.1. Estructura de los Elementos JXME

1. Cada elemento inicia con el prefijo "*jxel",* el cual indica el inicio del elemento de un mensaje.
2. Lo que sigue al prefijo característico de los elementos jxme (*jxel*) es el identificador de los elementos del namespace. En los primeros cinco elementos del mensaje el valor del campo es "2", el cual especifica el namespace *proxy* (haciendo referencia al cuarto campo del encabezado, donde se definió "*2"* en el

encabezado como el identificador para el namespace *proxy*). El namespace con identificador *"1"* en los dos últimos elementos de la lista especifica el namespace predefinido, *jxta.*

3. Contiguo al identificador del namespace está un indicador (*flag*) de 1 dígito. Este valor en bits actúa como un indicador de campos opcionales en el elemento, tal como MIME type o la codificación. Un valor de *"0"* para este campo indica que usarán valores por defecto para los campos opcionales (el valor por defecto para MIME type es *application/octet-stream* y el valor por defecto para la codificación de contenidos es *base64*)
4. Seguido al byte indicador está el nombre del elemento. El nombre del elemento está constituido por dos subcampos; el primero es un valor de dos bytes (07 en el primer elemento) el cual especifica la longitud del nombre del elemento, su número de caracteres. Seguido a este campo se tiene el nombre del elemento en sí, en este caso *request* en el primer elemento.
5. Luego de especificar el nombre del elemento se tiene el contenido en sí del elemento. Los contenidos del campo consisten en dos subelementos: un valor de cuatro bytes que especifica el número de caracteres en el campo de contenidos (0006 en el primer elemento) y el contenido en sí (*create* en el primer elemento).

A continuación se van describir los mensajes que un cliente móvil envía al relay para desempeñar las diferentes operaciones en la red JXTA.

3.1.2. Comunicación de los Peers JXME con el Relay JXTA

Cuando un peer JXME se conecta a la red JXTA, primero solicita un identificador de peer (*peerId*). El relay le asigna entonces un peerId en respuesta a la solicitud, el cual utilizará durante el resto de la comunicación. Una vez que el cliente JXME obtiene su *peerId,* solicita una conexión con el relay y éste entonces le responde confirmando y aceptando su petición. La figura 3.4 ilustra este proceso.

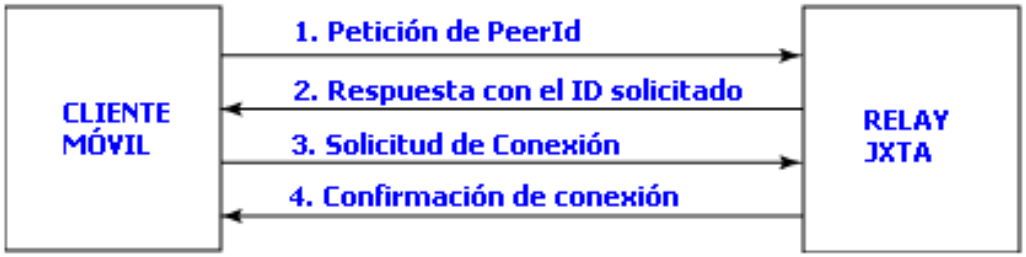


Figura 3.4. Establecimiento de conexión

Después de establecida la conexión, el peer JXME puede ahora hacer otra clase de peticiones al relay para llevar a cabo las operaciones involucradas en la aplicación. Ahora, obsérvese en detalle todo lo que ocurre durante el proceso descrito hasta aquí.

3.1.2.1. Petición de Identificador de Peer (PeerId)

Como se muestra en la figura 3.5, el cliente móvil envía la solicitud de un peerId como una URL en un mensaje HTTP usando el método HTTP GET. El peer guarda este peerId que el relay le asigna para su uso durante toda la comunicación.

```
GET /unknown-unknown?0,-1,http://172.16.0.37:2481/
EndpointService:jxta-NetGroup/
uuid-DEADBEEFDEAFBABAFEEDBABE0000000F05/pid HTTP/1.1
Connection: close
Content-Length: 0
User-Agent: UNTRUSTED/1.0
Host: 172.16.0.37:2481
```

Figura 3.5 Petición de PeerId

La URL en la petición HTTP es como la siguiente:

```
unknown-unknown?0,-1,http://172.16.0.37:2481/
EndpointService:jxta-NetGroup/uuid-DEADBEEFDEAFBABAFEEDBABE0000000F05/pid
```

Examinando esta URL, se pueden identificar los siguientes componentes:

1. ***unknown- unknown***: especifica que el solicitante (*requestor*) no tiene un *PeerId.*
2. ***0***: Este es el valor por defecto del *timeout* en milisegundos. Un valor de 0 indica que el cliente espera hasta que reciba una respuesta.

3. ***-1***: Esto indica cuanto tiempo (en milisegundos) el peer JXME permanece conectado al relay después de que la respuesta de la petición de PeerId ha llegado. Un valor de -1 indica que el cliente se desconecta inmediatamente después de recibir la respuesta. Para los clientes JXME este valor siempre es -1.
4. ***http:/172.16.0.37:2481***: esta es la dirección IP y el número de puerto por el cual el host relay está escuchando.
5. ***EndpointService:jxta-NetGroup***: esta parte es un identificador para el relay. Este valor siempre es el mismo en todas las peticiones JXME para los PeerId.
6. ***uuid-DEADBEEFDEAFBABAFEEDBABE0000000F05:*** este es un identificador de clase para el servidor relay. JXTA ha definido identificadores de clase para los diferentes tipos de servicios JXTA.
7. ***pid:*** la última parte especifica el comando en sí. En este caso *pid* indica que el cliente está solicitando un nuevo PeerId.

La lista de la figura 3.6 muestra una respuesta típica desde el relay a una petición de PeerId, el relay envía la respuesta a un mensaje JXTA embebido en el cuerpo (body) de una repuesta HTTP.

```
jxmg    0     01    05 relay                          04
jxel    2     0     08 response                       0003  pid
jxel    2     0     06 peerid                         xxxx  peer identifier
jxel    1     0     26 EndpointDestinationAddress     xxxx  destination address
jxel    1     0     21 EndpointSourceAddress          xxxx  source address
```

Figura 3.6. Respuesta a la petición de un PeerId

La respuesta contiene cuatro elementos, dos en el namespace *relay* y dos en el namespace *jxta:*

1. El primer elemento indica que el mensaje es la respuesta a una solicitud de PeerId.
2. El segundo elemento especifica el PeerId asignado al peer.
3. El tercer elemento muestra la dirección del terminal de destino, que en este caso es el cliente JXME.
4. El último elemento especifica la dirección del teminal de origen, que en este caso es el host relay.

3.1.2.2. Petición de Conexión

Después de que el cliente solicita un peerId luego solicita una conexión con el relay JXTA. La lista de la figura 3.7 muestra una petición típica de conexión, la cual es enviada como un mensaje de petición HTTP usando el método GET.

```
GET
/uuid-59616261646162614A787461503250033F55704B199754D3E9076A2947775A6EE03?0,
-1,http://172.16.0.37:8010/
EndpointService:jxta-NetGroup/uuid-DEADBEEFDEAFBABAFEEDBABE0000000F05/
connect,3600000,keep,other HTTP/1.1
Connection: close
Content-Length: 0
User-Agent: UNTRUSTED/1.0
Host: 172.16.0.37:8010
```

Figura 3.7 Solicitud de establecimiento de conexión

La siguiente figura muestra que fácilmente se puede extraer la URL de una petición HTTP:

```
uuid-59616261646162614A787461503250033F55704B199754D3E9076A2947775A6EE03?0,
-1,http://172.16.0.37:8010/EndpointService:jxta-NetGroup/uuid-DEADBEEFDEAFBABAFEEDBABE
0000000F05/connect,3600000,keep,other
```

Este mensaje es muy parecido al de petición de PeerId, con algunos detalles importantes a tener en cuenta:

1. La URL empieza con un PeerId válido (el mismo que se recibió en la respuesta a la solicitud de PeerId)
2. Ahora se usa un comando diferente en la petición de establecimiento de conexión, ahora es *connect,* en la solicitud de PeerId esta cadena era *pid.*
3. El valor que hay después del comando *connect*, en este caso *3600000* indica el tiempo en milisegundos que el relay conservará los mensajes entrantes para el cliente JXME. Si el cliente no lee sus mensajes dentro del tiempo especificado, el relay los descarta.

En respuesta a este mensaje, (ver figura 3.8) el relay envía una confirmación indicando que la conexión ha sido establecida exitosamente.

```
jxmg    0    01    05  relay                       04
jxel    2    0     08  response                    0009  connected
jxel    2    0     05  lease                       0007  3600000
jxel    1    0     26  EndpointDestinationAddress  xxxx  destination address
jxel    1    0     21  EndpointSourceAddress       xxxx  source address
```

Figura 3.8. Respuesta a una petición de conexión

Después de que el cliente se conecta con el relay, el cliente ya está en capacidad de ejecutar las siguientes operaciones:

- Unirse a Peer Groups
- Buscar recursos
- Crear pipes
- Enviar mensajes por una pipe
- Abrir una pipe para leer mensajes o cualquier clase de información

3.1.2.3. Petición de Ingreso a Peer Groups (join)

Para unirse a un peer group, el cliente envía un mensaje al relay en el cual especifica su deseo de conexión a un grupo. Dicha petición y su correspondiente respuesta gráficamente se pueden ver en la figura 3.9.

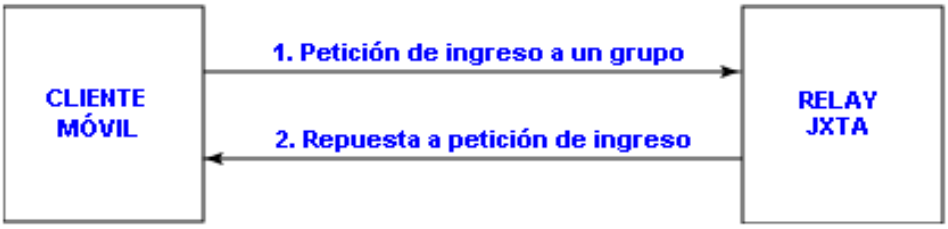


Figura 3.9 Petición de ingreso a un peer group

```
jxmg    0    01    05  proxy                       06
jxel    2    0     07  request                     0004  join
jxel    2    0     02  id                          xxxx  peer group identifier
jxel    2    0     03  arg                         xxxx  password
jxel    2    0     09  requestId                   0001  1
jxel    1    0     26  EndpointDestinationAddress  xxxx  destination address
jxel    1    0     21  EndpointSourceAddress       xxxx  source address
```

Figura 3.10 Mensaje de petición de ingreso a un peer group

Este mensaje (Figura 3.10) está compuesto de seis elementos. Los primeros cuatro elementos pertenecen al namespace *proxy* y los últimos dos al namespace *jxta..* Los elementos de este mensaje son explicados a continuación:

1. El elemento *request* especifica el propósito de la solicitud (unirse a un peer group - *join*).
2. El elemento *id* contiene el identificador de peer group (*PeerGroupId*)
3. El elemento *arg* especifica un password requerido para el ingreso al peer group. Esto ayuda a autenticar un peer JXME en un grupo. Si no se tiene un password (O si no se necesita) se envía una cadena vacía en este elemento.
4. El elemento *requestId* contiene un número entero asociado con la petición y se usa para hacer corresponder las respuestas que da el relay. Note que las respuestas del relay vienen en cualquier orden, por lo que se necesita algún mecanismo para hacer corresponder las peticiones a las respuestas.
5. Los elementos *EndpointDestinationAddress* y *EndpointSourceAddress* especifican la dirección del relay y del peer que hace la solicitud, respectivamente.

La lista mostrada a continuación corresponde a la respuesta de un servidor relay a una petición de ingreso (*join*) a un peer group.

```
jxmg    0    01    05  proxy                       04
jxel    2    0     08  response                    0007  success
jxel    2    0     09  requestId                   0001  2
jxel    1    0     26  EndpointDestinationAddress  xxxx  destination address
jxel    1    0     21  EndpointSourceAddress       xxxx  source address
```

Figura 3.11 Repuesta a una petición de ingreso a un peer group.

Esta lista indica que la petición fue recibida exitosamente y que el cliente se ha unido al nuevo peer group, lo que indica que ahora puede empezar a operar en dicho grupo.

3.1.2.4. Búsqueda de Recursos (*search*)

Se utilizarán los mensajes de búsqueda para localizar pipes, peers, peer groups y otros recursos JXTA. La figura 3.12 representa el proceso de petición de búsqueda y la(s) respuesta(s) del relay. El número de mensajes a una petición de búsqueda que puede retornar un relay depende de la cantidad de recursos que se ajustaron a los criterios de búsqueda. Esto se representa por una línea punteada en la figura 3.12.

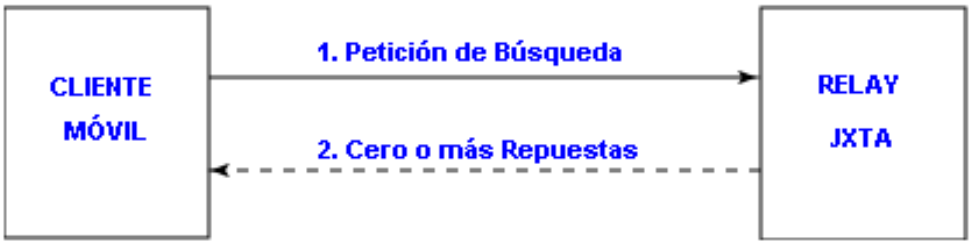


Figura 3.12 Proceso de petición de búsqueda de un peer group

Obsérvese a continuación un mensaje de búsqueda de peer group en detalle.

```
jxmg    0    01    05  proxy                       08
jxel    2    0     07  request                     0006  search
jxel    2    0     04  type                        0005  GROUP
jxel    2    0     04  attr                        0004  name
jxel    2    0     05  value                       0003  myPeerGroup
jxel    2    0     09  threshold                   0001  1
jxel    2    0     09  requestId                   0001  3
jxel    1    0     26  EndpointDestinationAddress  xxxx  destination address
jxel    1    0     21  EndpointSourceAddress       xxxx  source address
```

Figura 3.13 Mensaje de búsqueda de un peer group

Entre los elementos que componen este mensaje se tienen:

1. El elemento *request* especifica que el mensaje corresponde a una petición de búsqueda.
2. El elemento *type* especifica que el tipo de recurso que se está buscando. En este caso se tiene GROUP, lo cual indica que se está buscando un peer group. Otros valores que puede tomar este campo son PIPE (si se está buscando una pipe) o PEER (si se está buscando un peer).
3. El elemento *attr* especifica un atributo del recurso buscado. Por ejemplo, si se está buscando un peer group en particular por su nombre, el contenido de *attr* es *name*, como se puede ver en el la lista mostrada en la figura 3.13.
4. El elemento *value* especifica el objeto de búsqueda. Por ejemplo, si se está buscando un peer group por su nombre en particular (por ejemplo, *myPeerGroup*), este se especifica como el contenido del elemento *value.* También es posible utilizar un carácter comodín (asterisco, " * ") en la cadena de búsqueda para obtener como resultado todos aquellos patrones que se ajusten.

5. Todos los peers JXTA están en capacidad de responder a mensajes de búsqueda, el elemento *threshold* se usa para limitar el número de respuestas que un peer simple puede enviar en respuesta a una petición de búsqueda.

Cada respuesta a un mensaje de búsqueda viene en un mensaje separado. La lista mostrada en la figura 3.14 representa un mensaje típico de respuesta a una petición de búsqueda.

```
jxmg      0       01      05 proxy                          07

jxel      2       0       08 response                       0006  result
jxel      2       0       04 type                           0005  GROUP
jxel      2       0       04 name                           xxxx  name of group found
jxel      2       0       01 id                             xxxx  id of group found
jxel      2       0       09 requestId                      0001  2
jxel      1       0       26 EndpointDestinationAddress     xxxx  destination address
jxel      1       0       21 EndpointSourceAddress          xxxx  source address
```

Figura 3.14 Mensaje de respuesta a una petición de búsqueda

Esta respuesta contiene un solo resultado a una respuesta de búsqueda. El elemento *type* especifica el tipo de recurso buscado (peer group), el nombre especifica el nombre del recurso, y el *id* especifica el identificador del peer group (*peerGroupId*). El cliente que realiza la petición puede ahora usar el peerGroupId para sus comunicaciones con el peer group y para su ingreso a dicho peer group.

3.1.2.5. Creación de Pipes (*create*)

Como se mencionó anteriormente, si un peer necesita recibir mensajes de toda la red JXTA primero necesita crear una *pipe,* una vez creada, cualquier otro peer en la red utilizará esa pipe para enviarle mensajes a ese terminal en especial.

La figura 3.15 representa gráficamente el proceso de creación de una pipe.

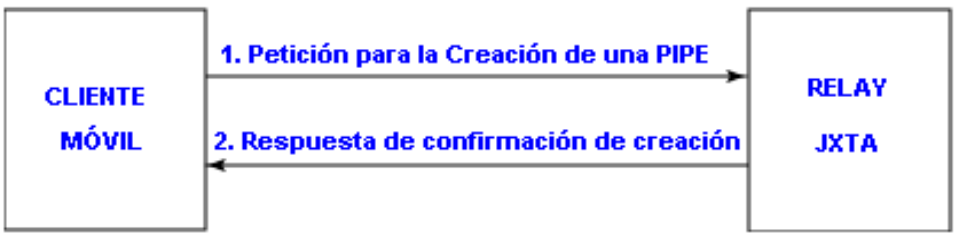


Figura 3.15 Representación del proceso de creación de una PIPE

```
jxmg    0    01    05  proxy                        07
jxel    2    0     07  request                      0006  create
jxel    2    0     04  name                         0008  testPipe
jxel    2    0     09  requestId                    0001  4
jxel    2    0     04  type                         0004  PIPE
jxel    2    0     03  arg                          0011  JxtaUnicast
jxel    1    0     26  EndpointDestinationAddress   xxxx  destination address
jxel    1    0     21  EndpointSourceAddress        xxxx  source address
```

Figura 3.16 Mensaje de petición para la creación de una pipe

La petición está compuesta por siete elementos:

1. El primer elemento especifica que el mensaje corresponde a una solicitud para la creación de un recurso (*create*).
2. El segundo elemento dice el nombre del recurso.
3. El *requestId* es un entero asociado con la petición, usado por el autor de para hacer corresponder las respuestas provenientes del relay.
4. El elemento *type* especifica el tipo de recurso a crear, en este caso *pipe.* Si se estuviera creando un peer group, el valor de este campo sería GROUP.
5. El elemento *arg* especifica el tipo de pipe. Para los clientes JXME, este puede ser *Unicast* o *JxtaPropagate*
6. El sexto y séptimo elemento especifican la dirección de destino y origen respectivamente.

```
jxmg    0    01    05  proxy                        08
jxel    2    0     08  response                     0007  success

jxel    2    0     09  requestId                    0001  4
jxel    2    0     04  type                         0004  PIPE
jxel    2    0     04  name                         0008  testPipe
jxel    2    0     02  id                           xxxx  pipe identifier
jxel    2    0     03  arg                          0011  JxtaUnicast
jxel    1    0     26  EndpointDestinationAddress   xxxx  destination address
jxel    1    0     21  EndpointSourceAddress        xxxx  source address
```

Figura 3.17 Mensaje de respuesta a una petición de creación de pipe

La mayor parte de los mensajes de respuesta son parecidos a su mensaje de petición. De hecho, existen básicamente dos diferencias:

1. El primer elemento dice que este es un mensaje de respuesta (*response*) y su contenido indica que se ejecutó efectivamente el comando correspondiente a la petición hecha.
2. El elemento *id,* da un identificador de la nueva pipe creada.

Los mensajes de creación de *peer groups* son muy parecidos a los mensajes de creación de *pipes*, la única diferencia está en el elemento *type (tipo).* Si se desea crear un nuevo peer group, el contenido del atributo *type* es GROUP. No es necesario especificar el elemento *arg* en una petición de creación de peer group.

3.1.2.6. Abrir una Pipe de Lectura (listen)

Para abrir una pipe de lectura (*inputPipe*), o pipe de escucha como se conoce en la terminología JXTA, se necesita enviar un mensaje de petición de escucha (*listen*) al relay. El relay empieza entonces a escuchar y a retener mensajes para que quien hizo la petición los pueda leer después.

En la figura 3.18 se representa gráficamente el proceso de petición /respuesta para que el relay empiece a estar pendiente de mensajes para el peer solicitante.

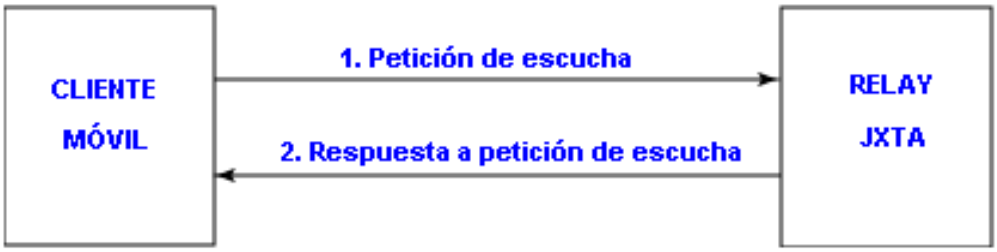


Figura 3.18 Secuencia de petición/respuesta para abrir una pipe de lectura

La siguiente gráfica muestra un mensaje típico de petición de apertura de una pipe de lectura:

```
jxmg    0    01    05  proxy                        05
jxel    2    0     07  request                      0006  listen
jxel    2    0     02  id                           xxxx  pipe Id
jxel    2    0     09  requestId                    0001  1
jxel    1    0     26  EndpointDestinationAddress   xxxx  destination address
jxel    1    0     21  EndpointSourceAddress        xxxx  source address
```

Figura 3.19 Mensaje de petición de apertura de una pipe de lectura (input pipe)

El elemento *request* en la lista mostrada en la figura 3.19 indica que la petición es para que se empiece a escuchar por dicha pipe. El elemento *id* especifica el identificador de la pipe (*pipeId*) sobre la cual se quiere escuchar.

A continuación se muestra el mensaje que el relay devuelve como respuesta a esta petición. Dice que la solicitud fue aceptada exitosamente, lo cual significa que el relay está dispuesto a escuchar mensajes enviados por esta pipe y a guardarlos hasta que sean leídos.

```
jxmg    0    01    05  proxy                       04
jxel    2    0     08  response                    0007  success
jxel    2    0     09  requestId                   0001  2
jxel    1    0     26  EndpointDestinationAddress  xxxx  destination address
jxel    1    0     21  EndpointSourceAddress       xxxx  source address
```

Figura 3.20 Mensaje de repuesta de un servidor relay a una petición de creación de pipe de lectura (*inputPipe*).

3.1.2.7. Envío de Mensajes a través de Pipes (send)

Antes de que se puedan enviar mensajes por una pipe, primero se tiene que encontrar dicha pipe. Un mensaje de búsqueda de pipe es muy similar al mensaje de búsqueda de un peer group, por esta razón no se detallarán los mensajes de búsqueda de pipes.

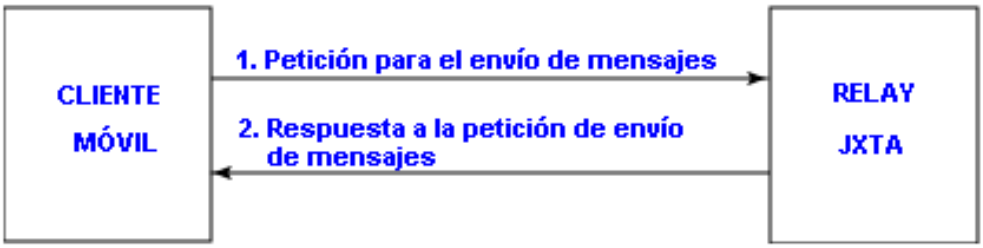


Figura 3.21 Secuencia petición/respuesta para el envío de mensajes a través de una pipe.

La lista de la figura 3.22 muestra un mensaje direccionado a una pipe específica.

```
jxmg    0    02    05  proxy 07 records            08
jxel    2    0     07  request                     0004  send
jxel    2    0     09  requestId                   0001  2
jxel    2    0     02  id                          xxxx  target pipe-Id
jxel    0    0     06  sender                      0006  tester
jxel    0    0     07  message                     0005  Hello
jxel    3    0     05  fileReference               0001  3
jxel    1    0     26  EndpointDestinationAddress  xxxx  destination address
jxel    1    0     21  EndpointSourceAddress       xxxx  source address
```

Figura 3.22 Mensaje de petición de envío de un mensaje

Ya se han explicado los primeros tres elementos (*request, requestId, y Id*). Los elementos *sender y message* pertenecen al namespace vacío (*empty*), el identificador para los elementos de este namespace es "*0".* Estos dos elementos son realmente específicos de la aplicación, lo que significa que el desarrollador puede inventar sus propios elementos del mensaje.

También se permite crear otros namespaces e incluir elementos específicos de la aplicación que pertenezcan a un namespace nuevo en el mensaje. Por ejemplo, obsérvese el elemento *fileReference,* cuyo identificador de namespace tiene un valor de "*3".* El encabezado del mensaje define el namespace como *records.*

La lista de la figura 3.23 muestra la respuesta del relay a este mensaje. El elemento *response* indica que la petición fue aceptada exitosamente, lo cual indica que el relay ha aceptado el mensaje y lo tiene listo para su entrega. Ahora el relay debe despachar el mensaje a su destino.

```
jxmg    0    01    05  proxy                       04
jxel    2    0     08  response                    0007  success
jxel    2    0     09  requestId                   0001  2

jxel    1    0     26  EndpointDestinationAddress  xxxx  destination address
jxel    1    0     21  EndpointSourceAddress       xxxx  source address
```

Figura 3.23 Respuesta a un mensaje de petición de envío

3.1.2.8. Lectura de mensajes de respuesta

La comunicación JXME con el relay es de naturaleza asíncrona, es decir, el relay no envía sus respuestas inmediatamente. En el proceso de envío de mensajes otros peers responderán, mientras que el relay tan solo almacena los mensajes entrantes de los clientes JXME. Los clientes JXME contactan con el relay mediante un mecanismo denominado *polling (*sondeo*)* para recibir sus mensajes entrantes.

El cliente JXME siempre tiene que contactar el relay; el relay nunca inicia una conexión. Esto es de especial importancia ya que un dispositivo J2ME solo puede actuar como un cliente HTTP (enviando peticiones HTTP) y no puede actuar como un servidor HTTP (escuchando peticiones de otros clientes).

JXME no define un mensaje de petición en especial que explícitamente le solicite al relay el envío de todos los mensajes que tenga para un cliente JXME. Los clientes JXME envían peticiones u otros mensajes hacia el relay, por ejemplo solicitudes de ingreso a grupos, o peticiones de búsqueda, mientras que el relay se dispone a recibir respuestas a dichas peticiones y a reenviarlas de tal manera que lleguen al cliente que hizo la petición. El cliente JXME usa el elemento *requestId* de los mensajes de respuesta para determinar cual de ellas corresponde a la petición hecha previamente.

Qué pasa si el cliente JXME no tiene ningún mensaje para enviar pero quiere seguir en espera de mensajes entrantes?, sencillamente el cliente envía periódicamente peticiones HTTP las cuales no contienen ningún mensaje JXTA que puedan causar alguna clase de procesamiento en el relay, mientras que el relay continuará enviando mensajes como usualmente lo hace.

Hasta aquí se ha visto la manera en que los terminales JXME se comunican con la red JXTA a través de los servidores relay, se ha detallado la estructura de los mensajes y comandos fundamentales para la implementación de soluciones P2P con los protocolos JXTA y cualquier clase de dispositivo que pueda implementarlos, en este caso dispositivos móviles J2ME.

Para profundizar un poco más en los conceptos vistos hasta el momento se aconseja revisar el anexo 4, PROGRAMACIÓN CON JXME, en el cual se construyen dos prototipos sencillos que demuestran las funciones más elementales en la comunicación de los peers JXME en la red P2P.

IV. EL PUNTO DE ENCUENTRO VIRTUAL P2P CON ACCESO MÓVIL

4.1. DESCRIPCIÓN GENERAL

El punto de encuentro virtual P2P es una aplicación implementada en el lenguaje de programación java para dispositivos móviles (J2ME) haciendo uso de los paradigmas de la orientación a objetos y la pila de protocolos JXTA para terminales con soporte Java (JXME).
El sistema le facilita a los usuarios de telefonía móvil el intercambio de información de interés e interactuar en un ambiente distribuido P2P. Los dispositivos móviles sobre los cuales de ejecuta la aplicación deben soportar la tecnología J2ME, y cumplir con la configuración CLDC 1.0 y el perfil MIDP1.0 o MIDP2.0.

En términos generales, el punto de encuentro es un sistema de mensajería instantánea en el cual los usuarios pueden gestionar comunidades (peer groups) especializadas en determinados temas (Deportes, tecnología, amistades, entre otros), crearlas, unirse a las mismas e intercambiar información personal de cada usuario especificada en un perfil configurado y almacenado previamente en cada terminal. Por defecto existen unas comunidades base a las cuales los usuarios pueden ingresar cuando deseen mientras se encuentren conectados a la red JXTA.

El sistema permite a los usuarios saber qué peers están activos en un momento dado, y también anunciar la llegada o la salida de un peer a un grupo mediante la difusión multicast de mensajes informativos de este tipo de eventos. Por otra parte, los peers una vez conectados a un grupo tienen la posibilidad de enviar mensajes bien sea a todos los miembros o a un solo peer en particular.

Teniendo en cuenta que los medios de escritura de los dispositivos móviles no son muy cómodos y ágiles, se ha implementado también una lista de palabras con las expresiones más utilizadas por los usuarios facilitando de esta manera la escritura desde los terminales JXME. Por otra parte se tiene a disposición un conjunto de *Emot-icons* que abrevian y hacen más agradable las sesiones de conversación en el Chat.

4.2. EL PROCESO DE DESARROLLO

El análisis se realizó teniendo como punto de partida varios sistemas de mensajería existentes en la actualidad. Observando sistemas como el MSN Messenger para retomar un poco la parte de las opciones más usadas por los usuarios tales como el uso de emoticons, la configuración de perfiles, nicknames y características muy generales que ayudaron a entender la importancia de tener una aplicación de fácil manejo para el usuario.

Por otra parte, también se observó el prototipo "*InstantP2P*" del proyecto *myJXTA*, un proyecto de la Sun Microsystems encargado de desarrollar un sistema de mensajería para peers de escritorio que pretende mostrar todas las ventajas que ofrece el protocolo JXTA en la construcción de aplicaciones P2P. Las funcionalidades prestadas por este sistema van desde la simple comunicación de mensajes escritos, haciendo una completa gestión de peer groups creando, buscando y permitiendo el ingreso autenticado a dichos grupos, hasta el intercambio y búsqueda distribuida de contenidos (Fotos, música, videos, documentos, …). Básicamente estas mismas funciones se pensaron para el punto de encuentro, teniendo como resultado final de esto una extensión a dispositivos móviles de la red P2P conformada por dispositivos de escritorio que ejecutan el *InstantP2P*.

Del proceso de análisis surgieron nueve casos de uso, los cuales se detallan a continuación de forma general. Para obtener una especificación más detallada de estos el lector debe remitirse al Anexo II.

Por otra parte, si se desea tener mayor detalle acerca de la instalación, configuración y operación del Punto de Encuentro virtual, se invita al lector a revisar el Manual de Instalación y de operación del sistema en el Anexo III.

4.2.1. Casos de Uso del Sistema

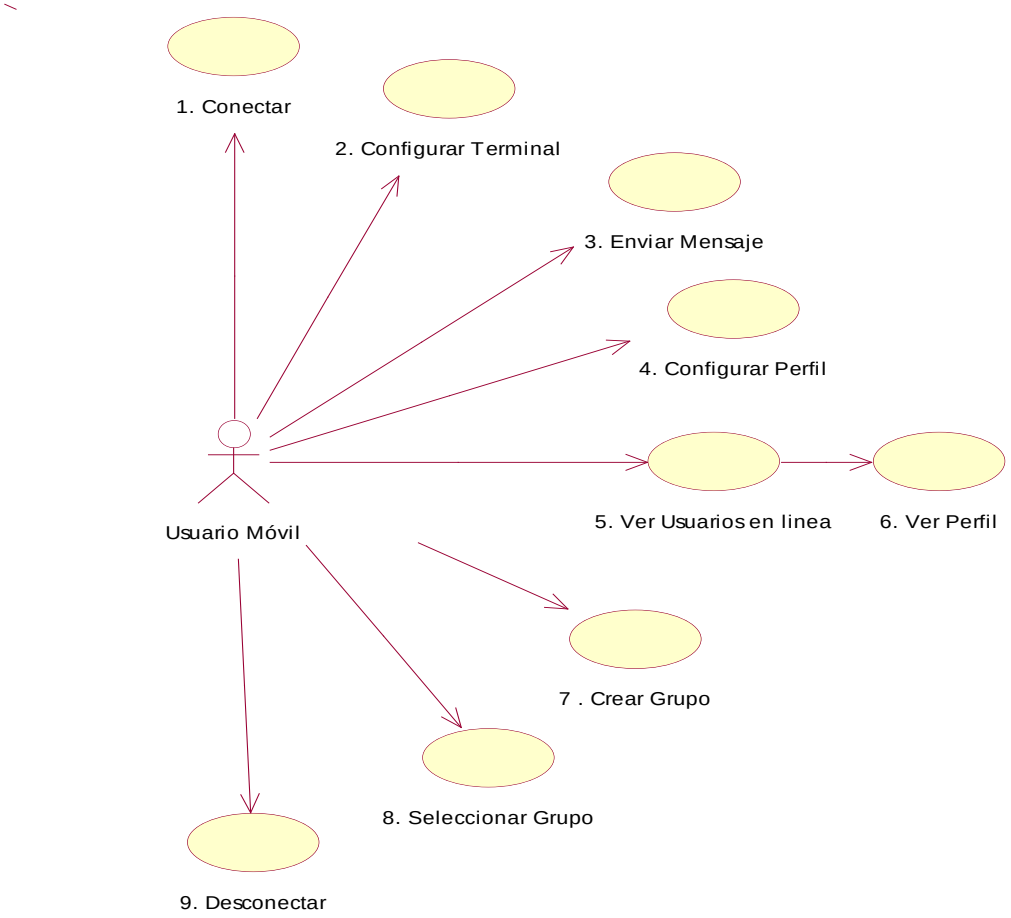


Figura 4.1 Diagrama de casos de uso del sistema

4.2.1.1. Caso de uso: Conectar

Actores: Usuario Móvil (Iniciador)

Tipo: Primario

Resumen: este caso de uso le permite al Usuario Móvil ingresar y ser parte de la comunidad JXTA conectándose con un host relay e iniciando un proceso de sondeo para enviar y recibir mensajes.

4.2.1.2. Caso de uso: Configurar Terminal

Actores: Usuario Móvil (Iniciador)

Tipo: Primario

Resumen: el usuario establece los parámetros necesarios para su conexión con la red JXTA, entre los datos ingresados están la dirección IP del host relay, el

puerto de conexión con el relay (9700 Por defecto), el nickname del usuario y el intervalo de sondeo especificado en segundos. el cual determina cada cuanto se consulta el relay para enviar o recibir nuevos mensajes.

4.2.1.3. Caso de uso: Enviar Mensaje

Actores: Usuario Móvil (Iniciador).

Tipo: primario

Resumen: mediante este caso de uso el usuario redacta un mensaje de texto, con la posibilidad de usar plantillas y adjuntar emoticons, para posteriormente enviarlo ya sea a todo el grupo al cual se encuentra asociado o a un usuario específico.

4.2.1.4. Caso de uso: Configurar Perfil

Actores: Usuario Móvil (Iniciador)

Tipo: Primario

Resumen: este caso de uso permite al usuario establecer su información personal, especificando su nombre real, su ubicación, edad, género y estado civil. Cabe anotar que la especificación de estos datos es opcional.

Nota: *El **perfil** de un usuario corresponde al listado de la información personal más relevante del mismo.*

4.2.1.5. Caso de uso: Ver usuarios en línea

Actores: Usuario Móvil (Iniciador)

Tipo: Primario

Resumen: el usuario puede ver todos los peers conectados en un grupo y elegir uno para establecer una conversación privada o ver su perfil.

4.2.1.6. Caso de uso: Ver Perfil

Actores: Usuario Móvil (Iniciador).

Tipo: Primario

Resumen: mediante este caso de uso el usuario puede consultar la información personal de otro dentro de un grupo.

4.2.1.7. Caso de uso: Crear grupo

Actores: Usuario Móvil (Iniciador).

Tipo: Primario

Resumen: este caso de uso le permite a un usuario crear una nueva comunidad o grupo de interés, al cual se puede unir cualquier usuario.

4.2.1.8. Caso de uso: Seleccionar grupo

Actores: Usuario Móvil (Iniciador).

Tipo: Primario

Resumen: mediante este caso de uso el usuario escoge un grupo de interés en el cual desee participar y a continuación le facilita su ingreso de tal manera que pueda interactuar con todos los demás usuarios en línea.

4.2.1.9. Caso de uso: Desconectar

Actores: Usuario Móvil (Iniciador).

Tipo: primario

Resumen: mediante este caso de uso, el usuario cierra todas las conexiones que tiene establecidas en un grupo y envía un mensaje para informar a todos los miembros que ha abandonado el mismo.

Nota: para mayor detalle en cuanto a los casos de uso remitirse al Anexo 2 "Análisis y diseño detallado del Punto de Encuentro".

4.2.2. Arquitectura del Sistema

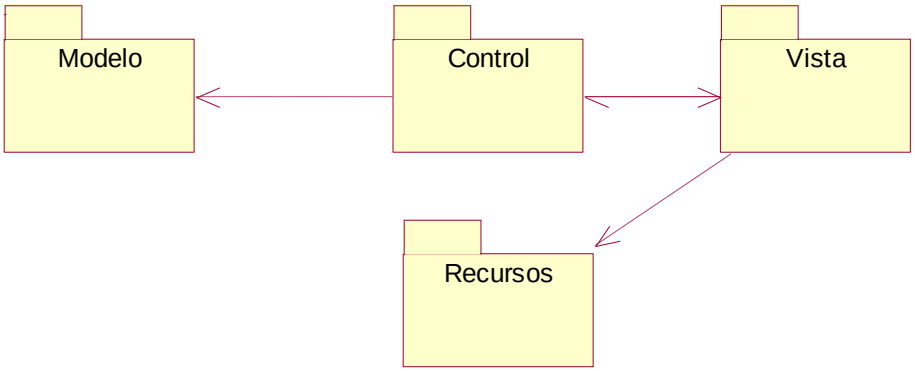


Figura 4.2. Diagrama de paquetes de la arquitectura propuesta

Para el desarrollo adecuado de las diferentes funcionalidades del sistema se ha decidido usar una arquitectura basada en el paradigma vista - control – modelo que nos permite separar en forma adecuada los diferentes módulos del sistema diferenciando la lógica de la presentación.

Este tipo de arquitectura asigna las funcionalidades de un sistema de la siguiente manera:

- **Vista:**
  - Contiene las interfaces con las que el usuario interactúa para recibir e ingresar la información.

- **Control:**
  - Responde a los comandos recibidos por parte del usuario traduciéndolos en procedimientos que involucran al modelo y las vistas.
  - Controla la persistencia de datos en la aplicación (RMS).
  - Gestiona el despliegue de las interfaces de usuario en el dispositivo.
  - Gestiona los procesos de comunicación de la aplicación.

- **Modelo:**
  - Contiene los elementos y entes básicos de JXME sobre los cuales se construye la aplicación.

- **Recursos:**
  - Este paquete contiene todas las imágenes y sonidos que hacen parte de la aplicación.

4.2.2.1. Descripción detallada

La aplicación desarrollada se distribuye en 4 paquetes principales los cuales contienen las clases que constituyen la aplicación y permiten realizar la funcionalidad requerida.
El primero de estos paquetes es *Vista,* este contiene las interfaces gráficas de usuario, las cuales deben ser muy intuitivas en su navegación de tal forma que el usuario pueda hacer un fácil uso de la aplicación, característica de la cual depende en gran parte el éxito de los desarrollos para dispositivos móviles.
Este paquete contiene interfaces gráficas para la captura y despliegue de datos al usuario, además del manejo de imágenes en la aplicación.

El segundo de estos paquetes corresponde al *Control*, aquí se encuentran las clases que permitirán responder a los eventos recibidos en las interfaces gráficas, gestionar el despliegue de las diferentes interfaces de usuario, controlar el acceso a los datos que se

guardan persistentemente para la aplicación y toda la funcionalidad de comunicación que se requiere para interactuar con JXTA utilizando las clases base de JXME.

El tercer paquete corresponde al *Modelo*, que contiene las clases J2ME base para interactuar en un entorno JXTA. Estas clases constituyen la abstracción de la red JXTA para los dispositivos móviles, permiten la creación de mensajes para la comunicación entre peers móviles y la red P2P e implementan los mecanismos para la gestión de grupos e intercambio de archivos en un ambiente distribuido. Otra de las funcionalidades en este paquete es el establecimiento de la conexión HTTP con el host relay, quien actúa a nombre de los peers móviles en la red JXTA.
Por último el paquete recursos contiene todas las imágenes utilizadas por la aplicación.

A. Paquete VISTA

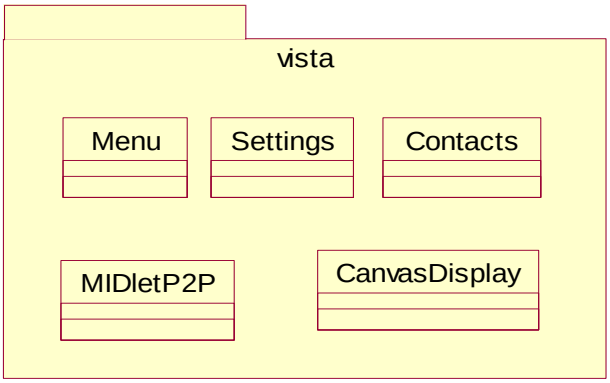


Figura 4.3. Clases del paquete vista

- **CanvasDisplay:**

  Esta clase se encarga de desplegar la interfaz de presentación principal de la aplicación y el conjunto de *emot-icons* utilizados en el punto de encuentro.

- **Settings:**

  Esta clase se encarga de mostrar el formulario de configuración del terminal y el formulario de configuración de perfil.

- **Menu:**

  Esta clase permite desplegar en pantalla los mensajes recibidos con su remitente y el contenido respectivo. Contiene además todos los posibles comandos ejecutables en el menú principal.

- **Contacts:**

  Esta clase permite gestionar los grupos JXTA, contiene la funcionalidad de buscar, crear y unirse a grupos. También permite monitorear el estado de los peers en un momento dado (En línea ó Sin conexión) y establecer sesiones privadas de conversación con cualquiera de los peers.

- **MIDletP2P:**

  Esta clase es la MIDlet principal en la cual se crea una instancia de todas las clases a utilizar a través de la aplicación y se hace un manejo centralizado del despliegue de las alertas.

B. Paquete CONTROL

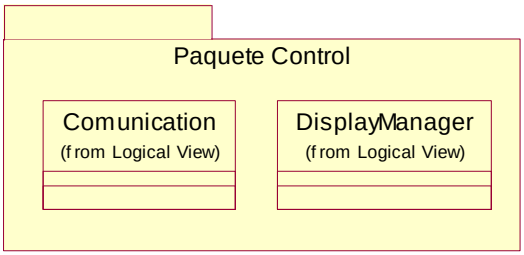


Figura 4.4. Clases del paquete Control

- **Comunication:**

  Esta clase es fundamental, ya que se encarga de manejar todos los procesos involucrados en la comunicación con la red JXTA basándose en la funcionalidad ofrecida por las clases del paquete modelo (PeerNetwork, Message, Element, HttpMessenger, ByteCounterOutputStream). Por otra parte, crea los mensajes de protocolo propios de la aplicación y analiza las respuestas enviadas por el host relay.

- **DisplayManager:**

  Esta clase es la encargada de gestionar el despliegue de las interfaces de usuario, facilitando la navegación a través ellas.

C. Paquete MODELO

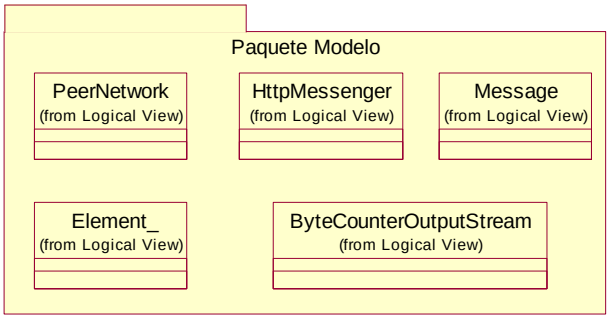


Figura 4.5. Clases del Paquete Modelo

- **PeerNetwork:**

  Esta clase es una abstracción de la red JXTA y especifica las operaciones que una aplicación móvil puede invocar sobre la misma.

- **Message:**

  Esta clase representa un mensaje (*Message*) JXTA. Un mensaje está compuesto de varios elementos (*Elements)*. Los *Elements* pueden estar en cualquier orden, pero ciertos elementos están reservados para ser usados por la red JXTA. Estos *Elements* privados usan un *namespace*[4] privado.

  Esta clase también define métodos especiales para acceder a las propiedades más usadas y para manejar las repuestas a las operaciones definidas en la clase PeerNetwork. Esta clase es inmutable.

- **Element:**

  Esta clase representa un elemento (*Element)* de un mensaje (*Message)* JXTA. Un Mensaje JXTA esta compuesto de muchos *Elements*. Esta es una clase inmutable.

- **HttpMessenger:**

  Esta clase proporciona el servicio de mensajería para los peers JXME, facilitando el envío y la recepción de mensajes a través del host relay.

- **ByteCounterOutputStream:**

  Esta clase hereda de la clase java "*OutputStream*" e implementa la funcionalidad de conteo de bytes sin realmente tener un buffer del flujo en memoria. Es decir, se precalcula el tamaño de un *Message* para fijar el encabezado *Content-Length* en una petición HTTP.

[4] El concepto de *namespace* en JXTA tiene el mismo significado que *namespace* en XML.

4.2.2.2. Diagrama de Clases

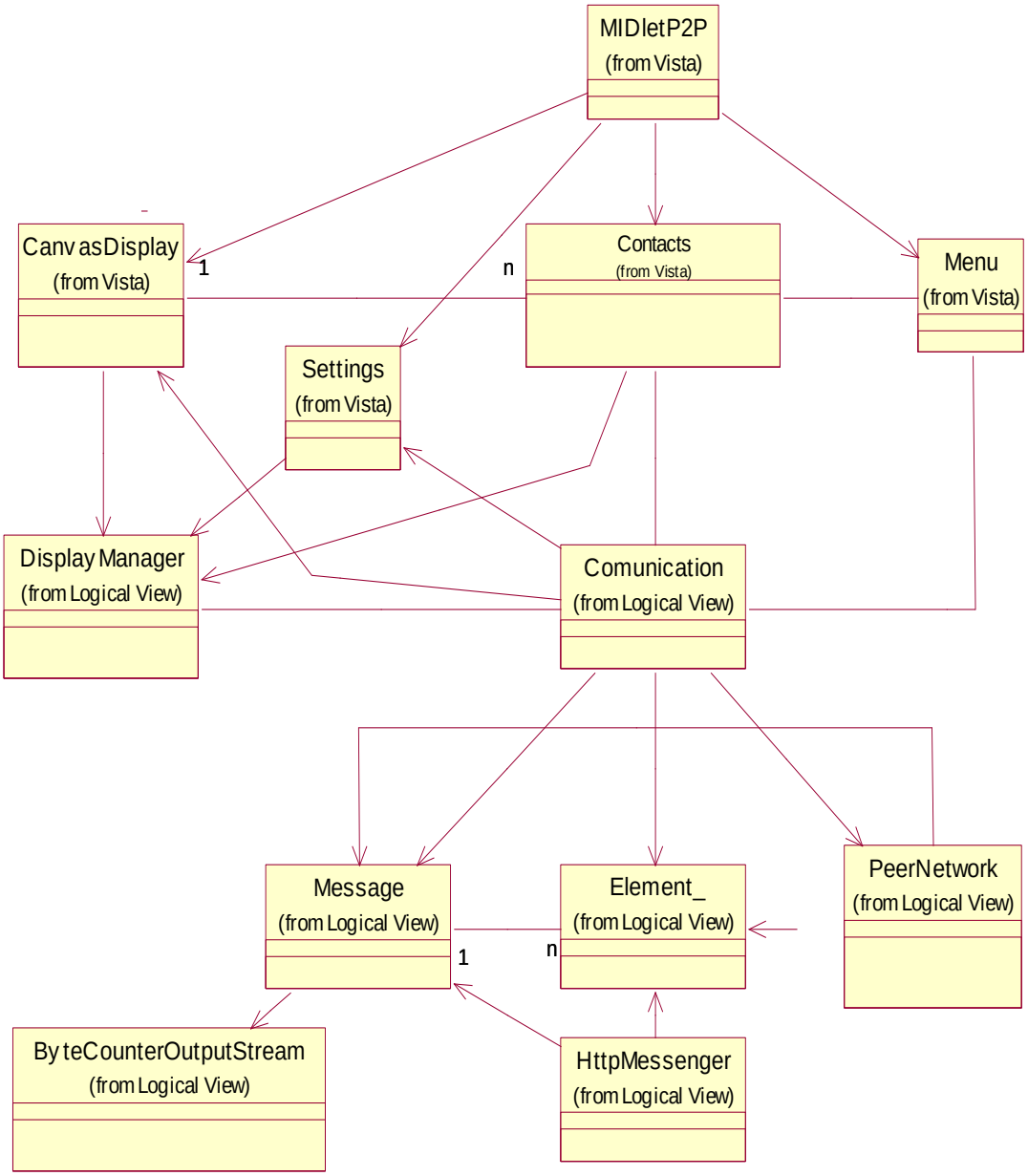


Figura 4.6. Diagrama de Clases

En este diagrama se representan todas las relaciones entre las clases descritas en el apartado anterior, para tener una visión general de todas las clases que componen el sistema.

Como puede observarse, en la parte inferior del diagrama se encuentran las clases fundamentales para la implementación del protocolo P2P, sobre las cuales se soporta todo aspecto relacionado a la comunicación del peer.
En la parte media del diagrama se encuentran todas las clases que realizan las funciones de control y gestión de la aplicación, tanto de almacenamiento, como de comunicación y despliegue de interfaces.
Finalmente en la parte superior del diagrama se puede apreciar la MIDlet[5], en la cual se crea una instancia de todas clases a utilizar en la aplicación.

[5] MIDlet es una aplicación ejecutable desarrollada en lenguaje Java para dispositivos que cumplen con el perfil MIDP (Mobile Information Device Profile).

4.2.2.3. Diagrama de Implantación

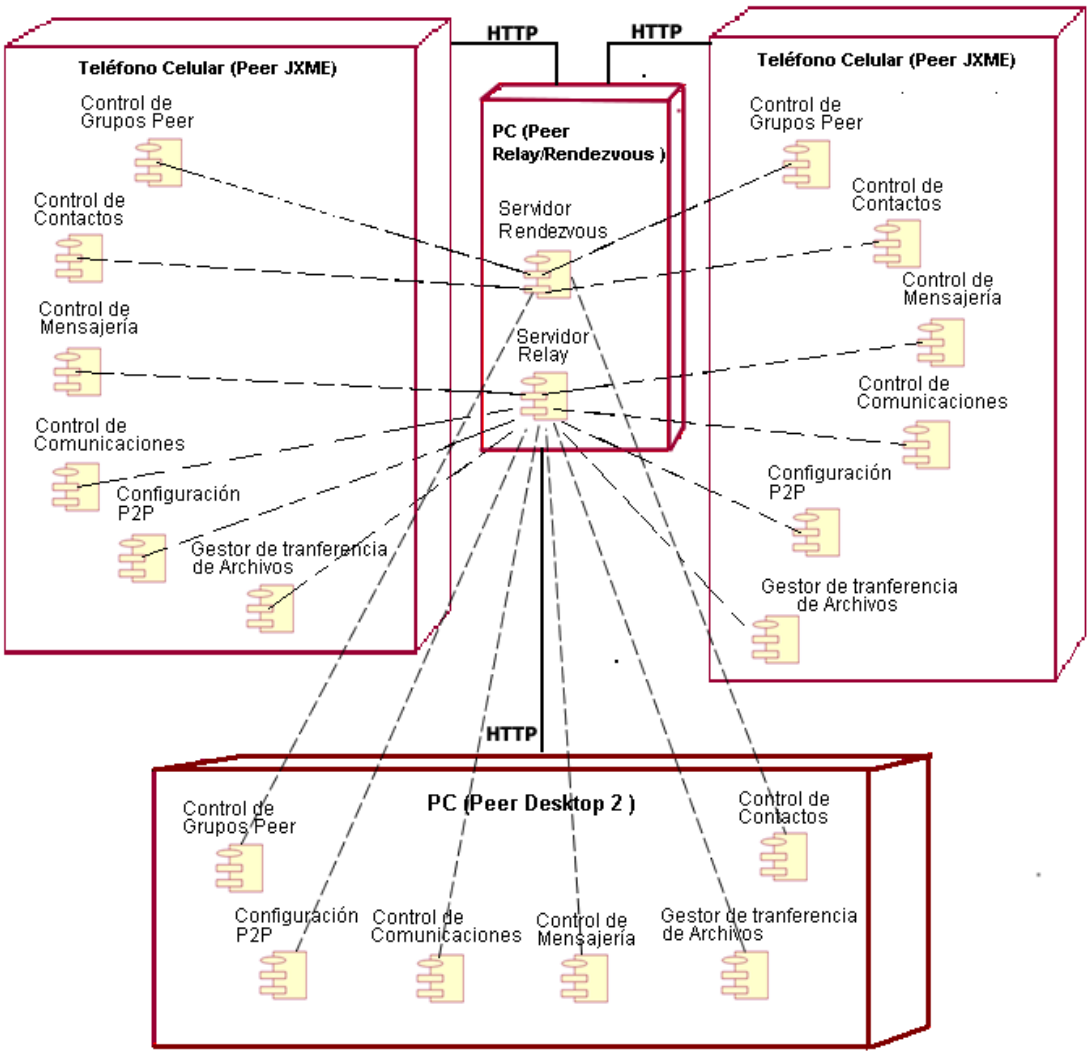


Figura 4.7. Diagrama de Implantación del Sistema

En este diagrama se describe la arquitectura física del sistema y cómo se ejecutan en ella los componentes de programación. Se muestran los nodos, componentes, conexiones y protocolos asociados que la constituyen y cómo se distribuyen todos estos para conformar el Punto de Encuentro Virtual P2P.

Se tienen básicamente los siguientes nodos: PC(Peer Desktop), PC(Peer Relay/Rendezvous), Teléfono Celular (Peer JXME), y uno de ellos se encuentra duplicado, el nodo "Teléfono Celular" para representar la interacción entre peers JXME. A continuación se describen cada uno de estos nodos.

- Teléfono Celular (Peer JXME):

  Este nodo representa cualquier dispositivo móvil con capacidades de soporte a aplicaciones J2ME que ejecuta la aplicación, la cual está compuesta por seis módulos de programación, en los que se distribuye toda la funcionalidad de la aplicación del lado de los peers JXME.

  Entre estos módulos se tienen:

  - Control de Grupos Peer: gestiona la creación, el ingreso, búsqueda y la salida del peer en peer groups.
  - Control de Contactos: maneja todo lo relacionado con la búsqueda y verificación de estado (OnLine/OffLine) de los peers en la red JXME.
  - Control de Mensajería: se encarga de atender todo lo relacionado con el envío y recepción de mensajes de texto entre peers.
  - Control de Comunicaciones: controla la conexión, mantenimiento y desconexión del peer en la red JXTA. Maneja todo lo relacionado con el protocolo P2P de comunicación.
  - Configuración P2P: este módulo almacena persistentemente la configuración del terminal JXME y el perfil de usuario. A partir de esta información se basan otros módulos, como por ejemplo el Gestor de transferencia de archivos quien toma la información del perfil y mediante el protocolo establecido para esto se descargan de un peer a otro los datos personales de un usuario. Y por otra parte, la configuración de terminal es utilizada para determinar los parámetros de conexión del peer con la red JXTA.
  - Gestor de Transferencia de Archivos: gestiona la transferencia de archivos entre peers JXME de tal manera que implementa la funcionalidad de *File Sharing* característica de los sistemas P2P más populares.

- PC(Peer relay/Rendezvous):

  Este nodo representa una máquina conectada a Internet que funciona como servidor Relay/Rendezvous en la red JXTA gracias a los componentes de programación que tiene instalados y en operación. En cuanto a dichos componentes se tienen:

  - Servidor Relay: este componente es el encargado de representar a los peers JXME en la red JXTA ejecutando comandos y acciones a su nombre y comunicándoles todos los eventos de interés de acuerdo a la aplicación que ejecuten. Es un componente impulsado por un

servlet Java de tal manera que cuenta con las características de todo un servidor en Internet.

- Servidor Rendezvous: este componente es el encargado de

- PC (Peer Desktop):

  Este nodo corresponde a un computador conectado a Internet en el cual se ejecuta una aplicación de mensajería instantánea P2P compatible con el sistema de mensajería instalado en los dispositivos móviles (*InstantP2P* propia del proyecto *myJXTA* de la Sun MicroSystems). Hasta el momento dicha compatibilidad permite la interacción con los peers JXME en sesiones de Chat dentro de los diferentes grupos creados. El InstantP2P implementa todas las funcionalidades descritas para el peer JXME descrito anteriormente pero para entornos no móviles.

  La red JXTA es una red *overlay*[6], los peers JXME se soportan en el protocolo HTTP para transportar sus datos (En las cabeceras HTTP). Este es el protocolo mediante el cual se establecen las conexiones entre los diferentes nodos, tal como se muestra en el Diagrama de Implantación del Sistema mostrado en la Figura 4.8.

[6] Overlay: dícese de aquel tipo de redes que se soportan en otras redes para su funcionamiento.

V. ANÁLISIS DE LA VIABILIDAD DE APLICACIONES P2P PRÁCTICAS PARA DISPOSITIVOS MÓVILES

En este capítulo inicialmente se describen algunas de las aplicaciones P2P prácticas que resultarían útiles a un usuario de telefonía móvil en determinadas situaciones, con el objetivo de observar el gran aprovechamiento que se le puede dar a la arquitectura P2P con las ventajas que trae la movilidad y el acceso a servicios en cualquier lugar y a cualquier hora desde un teléfono celular.

Posteriormente se realiza un análisis del desempeño de las aplicaciones P2P en dispositivos móviles con medidas de tiempos de retardo, volúmenes de información, y tarificación de los servicios P2P como parte de la Internet Móvil.

5.1. APLICACIONES DE LA ARQUITECTURA P2P PARA DISPOSITIVOS MÓVILES

La aplicación de tecnologías P2P a entornos móviles reales está sujeta a muchos factores, los cuales dependen del operador, del usuario, de la región a la cual se aplica, entre otros. A continuación se describen algunos de los entornos en los cuales se pueden implementar servicios muy interesantes basados en tecnologías P2P y los cuales promueven el uso de la Internet móvil en los usuarios de telefonía celular impulsando a las personas a que se involucren y participen más cercanamente con tecnologías de esta nueva "Era de la Información".

La arquitectura P2P presenta ciertas ventajas para la implementación de servicios de información, entretenimiento y comercio móvil. Las características de "siempre conectado" (*always on*) de las tecnologías móviles de transmisión de datos para telefonía celular y de los sistemas P2P, permiten el intercambio de información de interés más actualizada, transferencia de diferentes tipos de información como texto, audio, imágenes y video a diferentes destinos sin necesidad de hacer uso de otras tecnologías especializadas como Servicios de mensajería corta SMS o Servicios de mensajería Multimedia MMS.

Mediante la misma arquitectura pero con un diferente modelo de negocio en mente se pueden extender las ventajas de la P2P a diferentes ambientes y situaciones.

De acuerdo a la figura 5.1, se tienen varias comunidades JXTA, las cuales representan por ejemplo la comunidad de restaurantes en una ciudad, la comunidad de cinemas, de supermercados, la comunidad de aerolíneas o empresas de transporte alrededor de un país, entre otras posibles comunidades JXTA o grupos especializados, que por definición son un grupo de peers que ofrecen un conjunto común de servicios los cuales son accesibles por cualquier dispositivo que implemente los protocolos JXTA necesarios para participar en una comunidad.

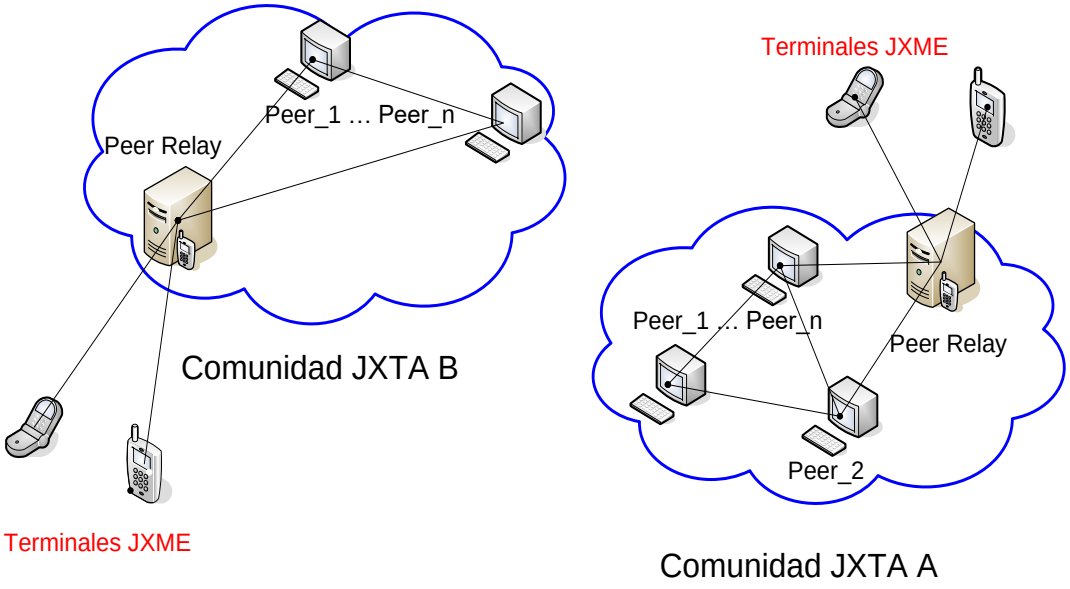


Figura 5.1. Diagrama general de aplicación práctica de la P2P móvil

Véanse a continuación algunas de las posibles aplicaciones de la arquitectura P2P implementada por JXTA para dispositivos móviles.

5.1.1. Servicios de Información

- **Cines y teatros**

Teniendo en cuenta la figura 5.1, y realizando un mapeo entre los elementos mostrados y la realidad se tienen:

La *comunidad* JXTA es un peer group conformado por los terminales peer ubicados en cada uno de los cinemas distribuidos en la ciudad, los cuales son administrados por personal de cada teatro, quienes se conectan a la red JXTA mediante una aplicación de escritorio desarrollada para participar en la red P2P y especialmente diseñada para publicar información de interés acerca del ciclo de películas que se están presentando en la semana, horario de presentación, costo de boletos, lugares disponibles, y aún más interesante, poder hacer reservaciones y no tener que hacer una larga cola de espera para comprar un boleto de entrada al llegar al lugar.

Por otra parte, los terminales JXME son dispositivos móviles que soportan la ejecución de aplicaciones java y tienen instalada una aplicación capaz de interactuar en esta comunidad JXTA de manera que pueden realizar todas las operaciones involucradas en este tipo de aplicación.

De otro lado, los peer Relay siempre estarán involucrados en soluciones P2P para dispositivos móviles mientras no se desarrolle bien el estándar *proxy-less* para la generación JXME 2.0, el cual pretende eliminar toda la infraestructura adaptadora (Relay, Proxy, Rendezvous…) para dispositivos con capacidades limitadas teniendo en cuenta que con el pasar del tiempo empezarán a tener mejores características para trabajo en red y desempeño (CLDC1.1/MIDP2.0) y podrán participar en una red JXTA de manera más autónoma.

- **Museos y Salas de Exposición de Arte**

Todas las ciudades alrededor del mundo cuentan la historia de sus antepasados, las guerras, sus héroes, su arte y cultura. Dichas leyendas son vivificadas en los viejos retratos, utensilios, y demás invaluables objetos puestos a disposición de todo el que quiera adentrarse por un momento en su historia. Para un turista puede resultar de gran interés tener a su disposición información acerca de la

ubicación de dichos centros de historia y arte en la ciudad que visita, su teléfono de contacto, horarios de atención y valor de la entrada, además de poder consultar información adicional directamente con el funcionario del museo encargado de dar mayores informes acerca del centro.

Retomando la figura 5.1, se puede decir que la *comunidad* JXTA es un grupo de peers conformado por los terminales peer de escritorio administrados por personal de cada museo o centro de arte, en el cual pueden interactuar entre sí para intercambiar información de interés para ellos y poder ofrecer a los usuarios más y mejores contenidos, es decir, que un usuario desde su dispositivo móvil pueda tener acceso en línea a fotografías de cuadros, esculturas, breves reseñas de los artistas y descripciones de las exposiciones y contenidos que se están mostrando actualmente en los diferentes museos, además de poder realizar preguntas a los encargados de la exposición y por qué no, hasta a los mismos artistas durante la temporada de la exposición.

- **Hoteles, Restaurantes y Centros de Diversión. (Centro de información turística)**

En las ciudades donde el turismo es una de las actividades económicas más importantes, puede resultar muy útil tanto para los empresarios de la región como para los turistas, el tener un centro virtual de información al que los visitantes tengan acceso en cualquier momento y desde allí puedan consultar las diferentes posibilidades de alojamiento que tienen al llegar a la ciudad, los sitios turísticos más importantes, los centros de diversión que deberían visitar, entre otros.

Una de las ventajas que podría tener este sistema es que los turistas puedan interactuar virtualmente con otros visitantes, compartir sus experiencias en la ciudad y conocer las diferentes opiniones acerca de cualquier lugar antes de y después de la visita, de esta manera la experiencia vacacional sería más divertida y enriquecedora.

Por otra parte, también se podría tener a disposición de los usuarios información específica sobre algún lugar en especial, entre restaurantes, bares, discotecas, parques, los platos especiales en el caso de los restaurantes, horarios de atención, costo de entradas para los parques y atracciones turísticas, breves

descripciones de los sitios con imágenes entre otros detalles de interés con el fin de informar bien a los turistas de todas las posibilidades que tienen al visitar el lugar.

5.1.2. Sistemas de Transporte

A un usuario podría resultarle interesante tener a su disposición información sobre las diferentes rutas que cubren las empresas de transporte alrededor del país, las horas de partida y llegada hacia los diferentes destinos, además del tipo de auto, el costo del pasaje y hasta realizar la compra de pasajes mediante su dispositivo móvil. Por otra parte, las empresas prestadoras de servicios de transporte pueden publicar información acerca de descuentos por la compra de cierta cantidad de pasajes, y hacer sorteos de los mismos en tiempo real entre usuarios de diferentes ciudades.

Si la red celular ofrece buenos niveles de cobertura sobre las rutas nacionales, podría también conocerse el estado de los buses y pasajeros en caso de accidentes, averías mecánicas o situaciones de congestión de tráfico debido a derrumbes o a cualquier clase de eventualidad. De tal forma que se pueda prestar la ayuda más adecuada en cualquier situación.

El modelo se puede aplicar de la misma forma para la consulta de itinerarios, ofertas especiales, rutas cubiertas por las diferentes aerolíneas, compra de pasajes y sincronización de las sucursales alrededor del mundo en cuanto al manejo de maletas y otros detalles que requieran mensajería instantánea y manejo de transacciones en el caso de la compra de pasajes. Los usuarios en tierra antes y después del viaje, pueden consultar un asesor especializado encargado de resolver cualquier inquietud en cuanto a su viaje, sus maletas y el lugar de destino.

5.1.3. Juegos P2P en Redes Celulares

Uno de los mercados con mayor crecimiento en el sector de la telefonía móvil es el de los juegos y el entretenimiento en general. Existen aplicaciones cada vez más interesantes y entretenidas en la Internet móvil por las cuales la gente está dispuesta a pagar, juegos multiusuario, juegos de azar, de estrategia, entre muchas otras clases de juego, que emplean arquitecturas tanto cliente/servidor como P2P y resultan de gran interés para los operadores de telefonía móvil y para los usuarios. A continuación se describen algunas de las clases de juego P2P y su despliegue en redes de telefonía móvil.

- **Ajedrez P2P:**

En este juego, cada usuario inicia su aplicación en el móvil, una vez se conecta a la red P2P, queda a la espera de un oponente, mediante una búsqueda previa se despliega una lista de todos los posibles oponentes, usuarios ubicados en cualquier lugar del mundo conectados a la red P2P ejecutando la aplicación en su terminal. Una vez reconocen los posibles contrincantes, se escoge uno de ellos e inmediatamente se le hace una invitación a dicho usuario, la cual puede aceptar o rechazar. Habiéndose establecido la partida, la aplicación que se ejecuta en cada terminal va actualizándose a través de cada movimiento que se hace verificando la lógica y las reglas del juego para determinar el ganador en cualquier momento.
En cuanto a su funcionamiento, realmente la aplicación se ejecuta en su mayor parte en cada terminal, de tal manera que el intercambio de información a través de la red sea mínimo, comunicando únicamente las actualizaciones del juego a cada uno de los jugadores.

- Triqui o "*TicTacToe*":

Este es el popular juego cuyo propósito es poner tres de los símbolos de forma consecutiva en la rejilla, jugando por turnos intercaladamente. El funcionamiento de este juego es muy similar al descrito para el juego del ajedrez, la aplicación se ejecuta en cada terminal y solamente se transmite a los jugadores sus movimientos, y finalmente un mensaje que indica el ganador.

- Juegos de Póker (BlackJack o veintiuno):

Esta es otra categoría en la que P2P presenta una arquitectura apta para su desarrollo. Los juegos de cartas en los que se requiere de varios jugadores y hasta se hacen apuestas son perfectos candidatos a ser desplegados en redes P2P y a constituir todo un modelo de negocio entre el operador y los clientes móviles o entre los mismos clientes móviles. Es posible crear una moneda virtual con la que se juegue en red, la cual se pueda concretar bien sea en minutos de telefonía al aire, en saldo de navegación por la Internet móvil, o dinero en efectivo mediante un acuerdo entre el operador, los usuarios y los bancos.

Como puede observar en este apartado, la cantidad de aplicaciones que se pueden implementar mediante la arquitectura P2P en entornos reales móviles es amplia y se presentan servicios muy interesantes tanto para el operador de telefonía como para el usuario, postulando la arquitectura P2P como una opción prometedora en el campo de los servicios móviles de 2.5G y 3G.

5.2 FACTORES MÁS SIGNIFICATIVOS QUE AFECTAN EL DESEMPEÑO DE LAS APLICACIONES P2P EN REDES CELULARES

A continuación se realiza una revisión general de los problemas que más afectan el desempeño de las aplicaciones de datos en dispositivos móviles haciendo una estimación sobre celulares de 2.5G y 3G como GSM y UMTS respectivamente, teniendo GPRS como portador de datos. El análisis se podría considerar para redes CDMA, pero para efectos del presente trabajo, se pretende analizar el efecto de la disposición de los terminales en la red celular, por lo cual se estima que se mantendría una proporción similar a la de los resultados mostrados en esta sección.

Los dos factores más importantes a tener en cuenta a la hora de desarrollar aplicaciones de datos para dispositivos móviles son: la latencia y el limitado ancho de banda con el que se cuenta en las redes celulares. Existen otros factores que afectan el desempeño de las aplicaciones en redes celulares pero en este documento se analizan las más significativas.

5.2.1. La latencia en aplicaciones multiusuario.

Los datos mostrados a continuación presentan los resultados obtenidos de experimentar con juegos móviles, tanto en arquitecturas Cliente/Servidor como en arquitecturas P2P.

La latencia en términos simples se conoce como el tiempo que tarda la red en dar una respuesta frente a unos datos de entrada. En el caso especial de las aplicaciones P2P, se tiene un efecto significativo en la latencia que presenta la red ante la aplicación. De acuerdo a la figura 5.13, la latencia P2P es dos veces el

retardo de terminal a terminal (~ 2 x RTT) en tanto que la latencia Cliente/Servidor es dos veces el retardo de Terminal a servidor ~ [7] RTT.

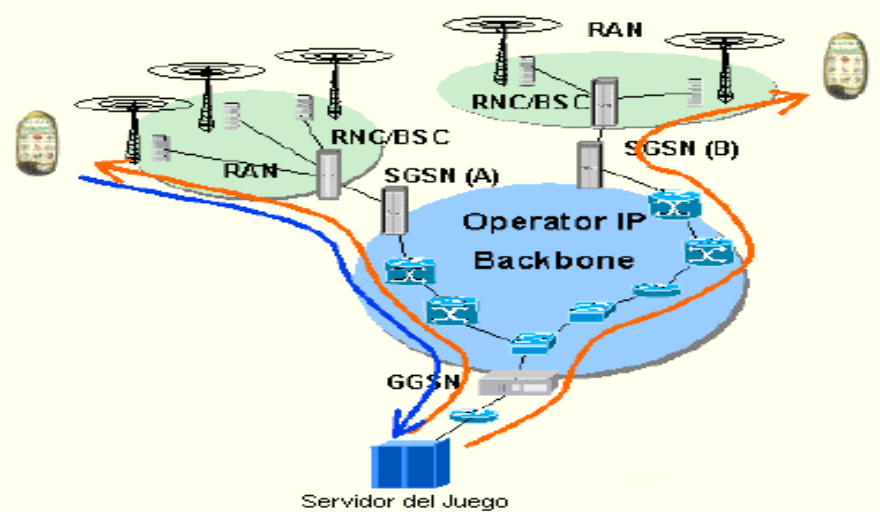


Figura 5.2 Juegos Multijugador en arquitectura Cliente/Servidor

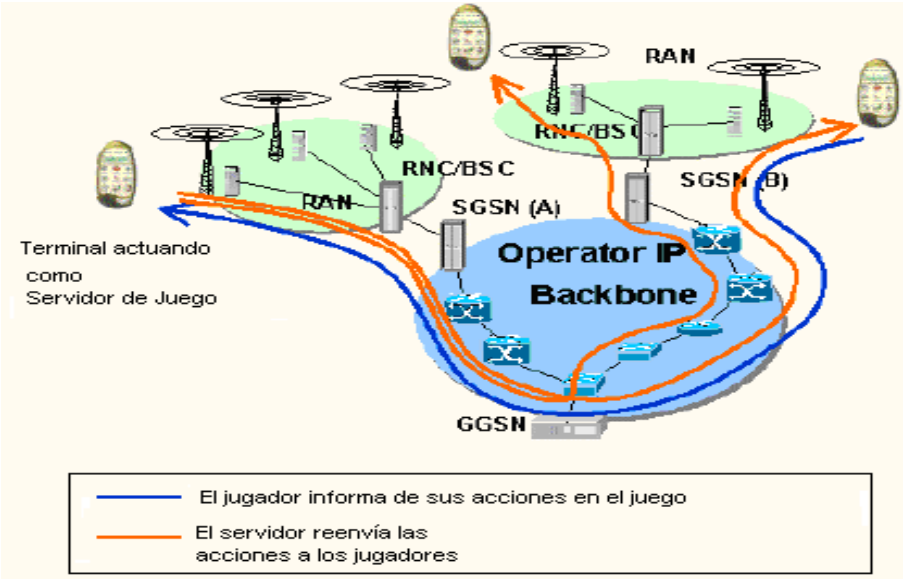


Figura 5.3 Juegos Multijugador en arquitectura P2P

El componente principal del retardo extremo a extremo en redes celulares es generalmente el introducido por la interfaz aérea, este componente se reduce en

[7] El RTT (Round Trip Time) es el tiempo total que un paquete tarda en viajar desde el cliente hasta el servidor más el tiempo que toma en responder del servidor al cliente. Dicha respuesta puede contener datos o simplemente un reconocimiento (ACK) del protocolo TCP, dependiendo de la lógica de la aplicación.

la medida en que se aumenta la tasa de transferencia de bits, la velocidad de transmisión. Sin embargo, además del retardo introducido por la interfaz aérea existen otros factores de importancia en esta medida, por ejemplo, el retardo que se introduce en el establecimiento y la terminación de una conexión de la red móvil con la red cableada. Cada vez que se establece una conexión, el sistema necesita reservar recursos en su red de acceso inalámbrico, en el núcleo de la red, y en su red de transporte. Estas asignaciones obviamente toman tiempo, y siempre estarán allí sin importar la velocidad de transferencia de datos de la interfaz aérea.

Para aprovechar mejor los escasos recursos de acceso inalámbrico, se cierran las conexiones cuando se detecta que no hay actividad. En este proceso de establecer y cerrar rápidamente la conexión de radio se introducen ciertos retardos, particularmente en el caso de conexiones GPRS/EGPRS.

Por lo tanto, para evitar esta clase de retardo durante cada sesión, existen características que también mantendrán el canal por un poco más de tiempo después de que no haya paquetes en espera a ser enviados en los buffers. Esto mejorará los tiempos de RTT, particularmente para conexiones con patrones de transmisión poco regulares.

Otros componentes que introducen el retardo extremo a extremo en redes móviles son el procesamiento en la estación móvil (MS – Mobile Station) y el retardo de los elementos e interfaces de red. La figura 5.15 muestra algunas estimaciones de los valores de RTT en las diferentes redes móviles. Este retardo es el de la interfaz aérea, la estación móvil y los elementos de la red en conjunto. Los tiempos de latencia calculados están basados en una red sin carga, pero en la práctica la carga podría tener un mayor efecto en la latencia de la comunicación entre terminales. Esto se debería a que tanto en CDMA como en GPRS las capacidades de la interfaz aérea son compartidas entre usuarios, en el caso de servicios que no son de tiempo real, ya que solamente en servicios de tiempo real las aplicaciones reservan su ancho de banda[8].

[8] Tomado de Nokia Forum; "*Multiplayer Game Performance over Cellular Networks*"; http://www.forum.nokia.com

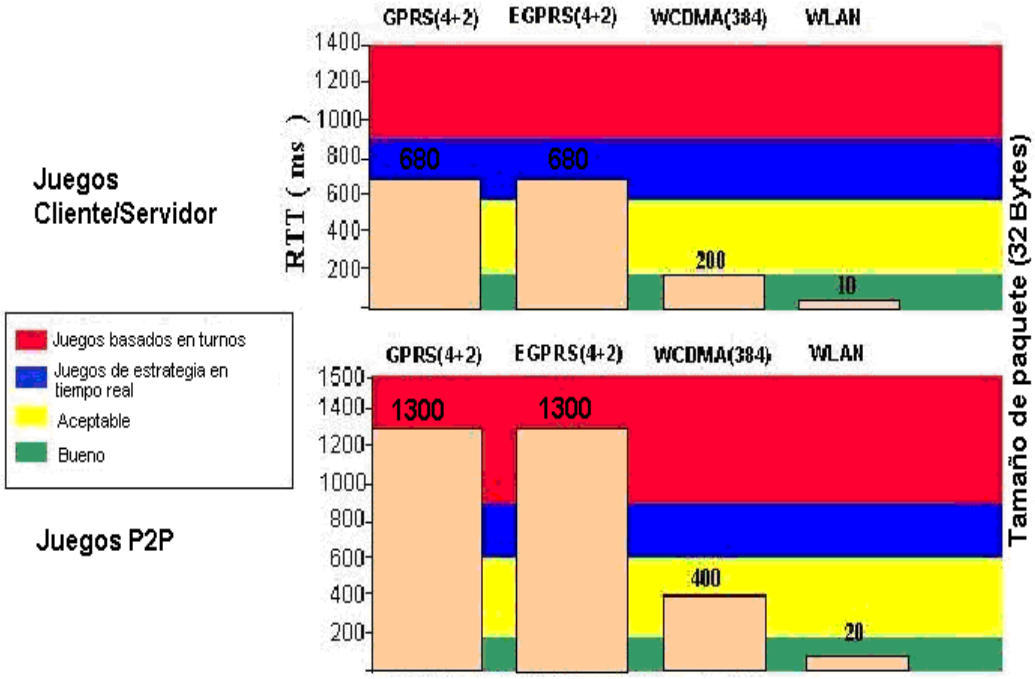


Figura 5.4 Valores de RTT en las redes móviles

Fácilmente se puede observar la notable diferencia en los tiempos de latencia de las aplicaciones en una arquitectura P2P frente a los de la arquitectura Cliente/Servidor. Por lo cual, la arquitectura P2P con las tecnologías actuales disponibles en Colombia no sería la mejor opción para el desarrollo de juegos en tiempo real que requieran una rápida respuesta. Sin embargo, se puede observar que las redes P2P presentan hasta el momento un mejor desempeño en los juegos de turnos, en los cuales no se exige un tiempo de respuesta tan corto como en los juegos de estrategia y tiempo real. La arquitectura P2P no resulta tan aventajada en este aspecto, pero se puede subsanar esta falencia optimizando el protocolo utilizado, es decir, reduciendo la cantidad de datos de protocolo, y mejorando la red de soporte, por ejemplo, como se muestra en la figura anterior, el juego evaluado presenta un mejor desempeño en las redes WCDMA y WLAN respecto a las redes GPRS/EGPRS.

5.2.2. El ancho de banda y las aplicaciones P2P

Otro de los factores a tener en cuenta a la hora de desarrollar aplicaciones de datos para dispositivos móviles es el ancho de banda disponible. Tanto en redes celulares como en las redes fijas cableadas, es un recurso limitado que hay que saber aprovechar para ofrecer a los usuarios aplicaciones con un óptimo rendimiento y rápida respuesta.

Para el análisis del ancho de banda se utilizó una de las aplicaciones P2P más significativas, el Punto de Encuentro. Del Punto de Encuentro la funcionalidad que más tráfico puede generar, la más dinámica y la que podría demandar mayor ancho de banda es el Chat P2P. En primer lugar, porque el texto que puede enviar cualquier usuario puede ocupar hasta 256 caracteres, los mensajes se envían asíncronamente y con una frecuencia promedio de 3 a 4 mensajes por minuto de acuerdo a la velocidad promedio estimada de escritura de los usuarios en el teclado del móvil.
Por otra parte, los peers JXME periódicamente sondean el host relay en espera de mensajes entrantes mediante peticiones HTTP, generando constantemente cierta cantidad de tráfico, lo cual afecta indudablemente el ancho de banda disponible.

Por un momento se consideraron los juegos P2P para realizar este análisis, pero la mayor parte de juegos implementados con JXME utilizan funciones mínimas de comunicación, ya que generalmente se ejecuta la lógica del juego en los terminales y únicamente se comunican las actualizaciones del juego por la red JXTA. Aunque se realiza un sondeo de las mismas características que en la aplicación de mensajería del Punto de Encuentro, para un mejor desempeño del juego, generalmente se procura en diseño minimizar la cantidad de datos de la aplicación que dependan del componente de comunicación. Además, entre las aplicaciones de conversación y los juegos móviles, las primeras son más utilizadas que cualquier otra clase de aplicación.

Analizando la aplicación "Punto de Encuentro Virtual" en este aspecto se realizaron pruebas con el *Series 40 Developer Platform 2.0 SDK*[9] el cual

[9] Herramienta descargada de http://www.forum.nokia.com

incorpora un analizador de tráfico muy útil y permite simular gran parte de las características soportadas por los dispositivos móviles con soporte para aplicaciones J2ME más populares en Colombia (Nokia 6230, Nokia 3595, Sony Ericsson T610/T616, entre otros).

Los resultados de tráfico cursado que aparecen en las figuras corresponden a los datos generados por la aplicación en un teléfono emulado que se ejecuta sobre una red TCP/IP. La veracidad de los datos de tráfico mostrados por el monitor se puede verificar en el contenido de los mensajes cursados, tal y como se muestra a continuación.

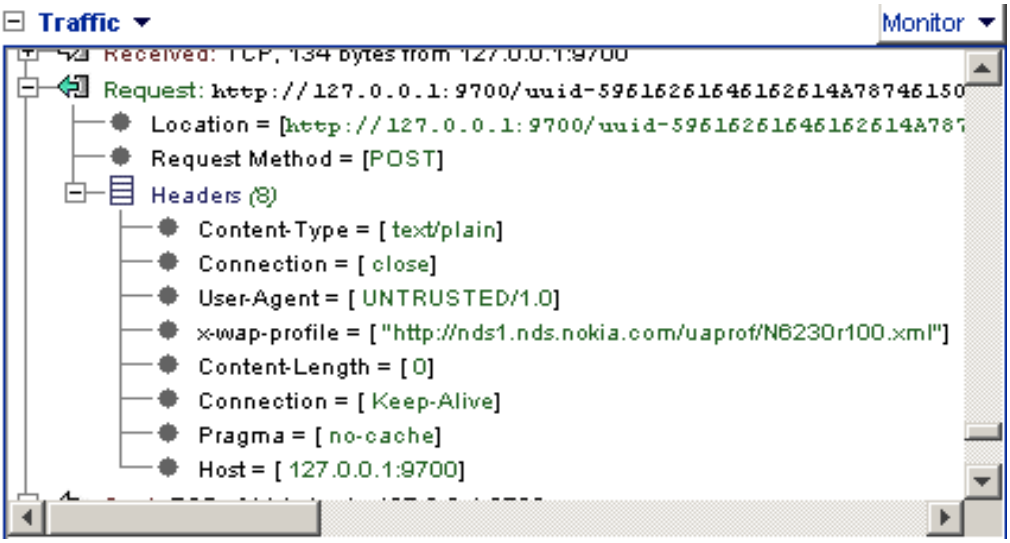


Figura 5.5 Despliegue del contenido de una petición de sondeo HTTP de un peer JXME sobre un relay JXTA

A continuación se analizará el volumen de tráfico cursado en las diferentes etapas por las que atraviesa la aplicación de mensajería del Punto de Encuentro Virtual, especificando los resultados mostrados por el analizador de Nokia, realizando algunos cálculos a partir de estos y finalmente se llegará a determinadas conclusiones con base en los datos obtenidos y otros datos de referencia asumidos.

A. Fase de inicio de la conexión

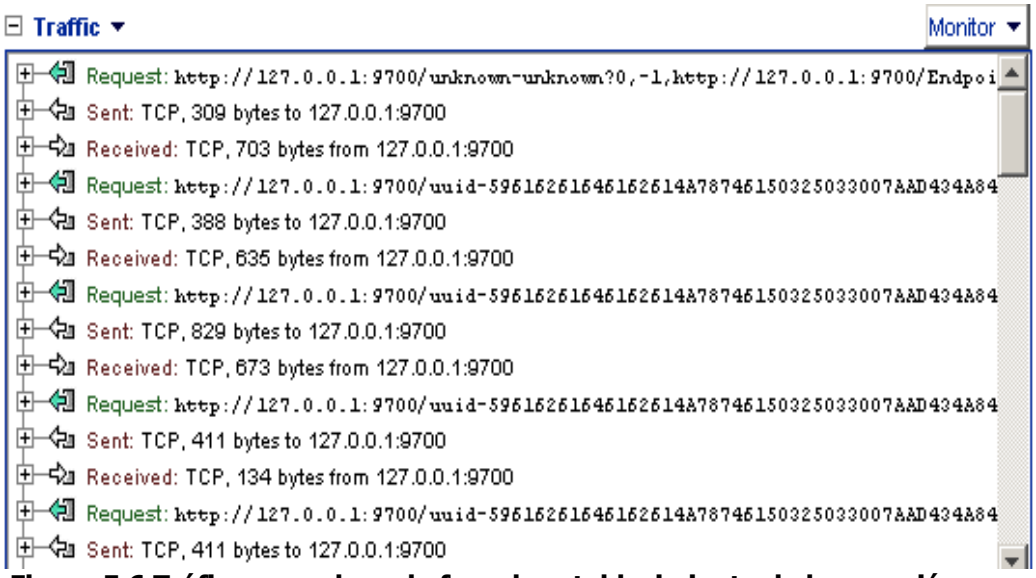


Figura 5.6 Tráfico cursado en la fase de establecimiento de la conexión

En la fase de establecimiento de la conexión se solicita un peerId, un pipeId para el móvil y por otra parte se descarga la lista de peer groups disponibles a los cuales el móvil puede unirse en el momento de su conexión.

Como puede observarse, el volumen de tráfico correspondiente a esta fase está dividido de la siguiente manera:

- Relay a Móvil: (Downlink)

703[Bytes] + 635[Bytes] + 673[Bytes] = 2,011[Kbytes]

- Móvil a Relay: (Uplink)

309[Bytes] + 388[Bytes] + 829[Bytes] = 1,526 [KBytes]

B. Fase de Ingreso a un peer group

Ahora, en el momento en que se establece la conexión con la red JXTA, el usuario debe unirse a alguno de los grupos disponibles, en este proceso se cursa el siguiente tráfico:

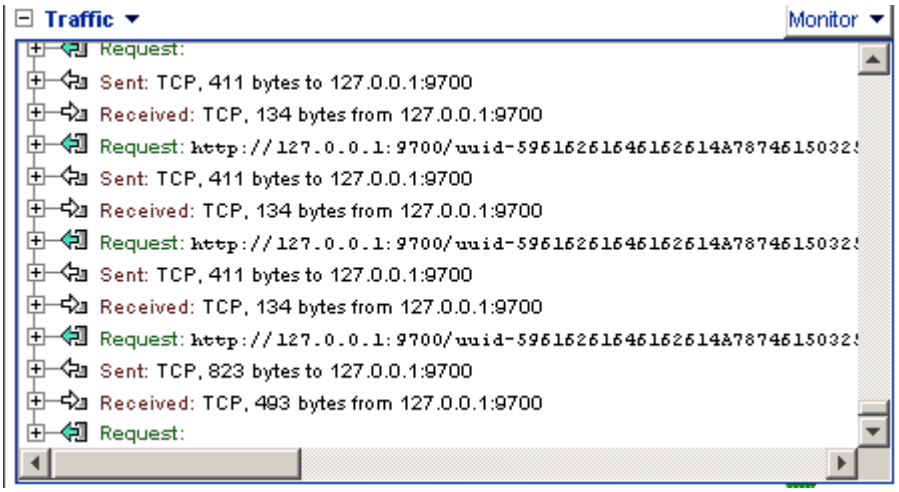


Figura 5.7 Tráfico cursado en el ingreso a un peer group

- Relay a Móvil: (Downlink)

0,493 [KBytes]

- Móvil a Relay: (Uplink)

0,823 [KBytes]

C. Fase de conversación

En la fase de conversación se distinguen dos posibles situaciones, o bien se está sondeando el relay en espera de mensajes entrantes, no hay mensajes ni para enviar ni para recibir y se envían mensajes vacíos para mantener la conexión con el relay, o bien en el sondeo se envían/reciben datos de un mensaje escrito.

- **Tráfico del sondeo del relay sin Envío/Recepción de algún mensaje**

  En este punto se registró la medida de tráfico entre el peer JXME y el host relay cuando no se estaba cursando ninguna otra actividad entre estos, solamente el sondeo periódico fijado cada 10 seg.

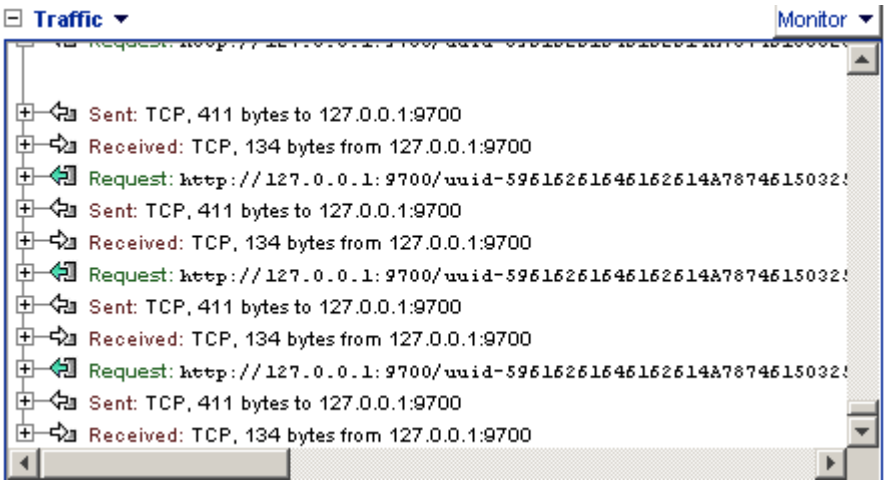


Figura 5.8 Tráfico cursado en el sondeo del relay sin Envío/Recepción de algún mensaje

- Relay a Móvil: (Downlink)

0,134 [KBytes]

- Móvil a Relay: (Uplink)

0,411 [KBytes]

- **Tráfico del sondeo del relay con Envío/Recepción de algún mensaje**

  Para estas pruebas se utilizó un mensaje de un número promedio de palabras *"y@k0>Entra al Punto de Encuentro JXME"*.

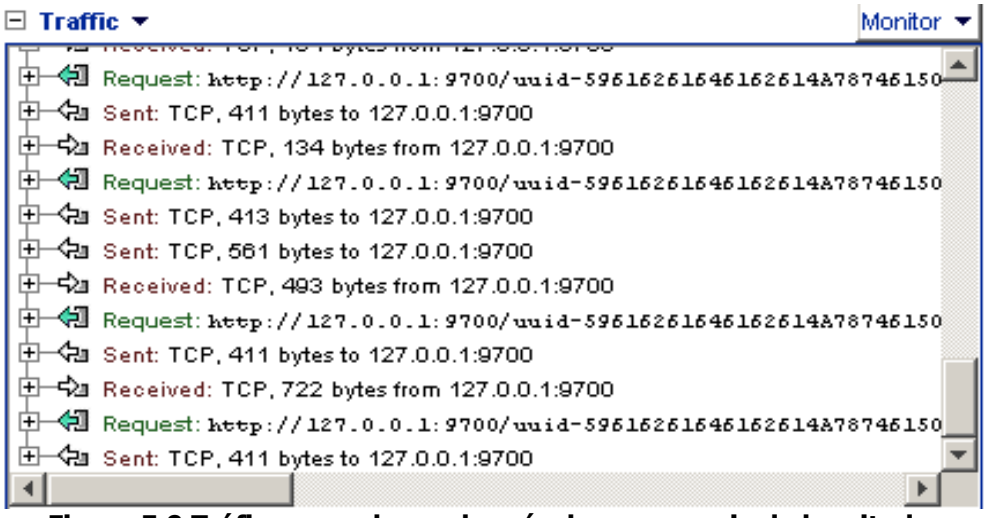


Figura 5.9 Tráfico cursado en el envío de un mensaje de longitud promedio dentro de un grupo

- Relay a móvil: (Downlink)

493[Bytes] + 722[Bytes] = 1,215[KBytes]

- Móvil a Relay: (Uplink)

413[Bytes] + 561[Bytes] = 0,974[Kbytes]

Como se puede observar, el tráfico generado a través de todo el proceso de establecimiento de conexión con la red JXTA, ingreso a un grupo, envío de un mensaje, espera y sondeo del host relay queda así:

Fase	Tráfico Uplink (Móvil a Relay) [Kbytes]	Tráfico Downlink (Relay a Móvil) [Kbytes]
Establecimiento de Conexión	*1,526*	*2,011*
Ingreso a peer group	*0,823*	*0,493*
Sondeo sin Envío/Recepción de mensaje	*0,411*	*0,134*
Sondeo con Envío/Recepción de mensaje	*0,974*	*1,215*

Tabla 5.1 Resultados de tráfico cursado entre peers JXME y host relay para el análisis del aprovechamiento del ancho de banda en una red GSM/GPRS

Ya que la fase de establecimiento y de ingreso a grupos es un evento que solamente ocurre al inicio, mas no representa algo que pueda congestionar la red durante largo tiempo, se tomarán los datos de sondeo ya con el peer conectado a la red en espera de Envío/Recepción de mensajes.

Asumiendo una sesión de 30 minutos en la cual se envían 3 mensajes por minuto (Un mensaje cada 20seg.) y el resto del tiempo el terminal JXME se encuentra haciendo un sondeo al relay cada 10 seg. Se tiene que el tráfico cursado cada 60seg (1min) es:

3 sondeos sin Envío/Recepción de Mensaje (sondeos con mensaje vacío para mantener la conexión con el relay):

Uplink: 1.233KB/min

Downlink: 0.402KB/min

3 sondeos con Envío/Recepción de Mensaje =
Uplink: 0.2922KB/min
Downlink: 3.645KB/min

De acuerdo a estos resultados, se puede observar que el ancho de banda ofrecido por las redes GSM/GPRS que operan en Colombia (14Kbps máximo en el enlace uplink y entre 25Kbps 56Kpbs máximo en el enlace downlink) es suficiente para satisfacer las necesidades de la aplicación, ya que tanto en el Punto de Encuentro como en el tipo de aplicaciones que se proponen en el apartado 5.1, el tráfico de datos por cada transacción no es significativamente alto a pesar de los mecanismos de sondeo periódico que se hacen desde los peers JXME hacia el host relay. Además el ancho de banda requerido es mucho menor que el que se ofrece en la red.
En conclusión, se puede decir que el ancho de banda no resulta ser un factor crítico en el desempeño del tipo de aplicaciones P2P descritas en este documento como si lo es la latencia.

5.2.3. Análisis de la viabilidad económica de las aplicaciones móviles P2P en redes celulares GSM/GPRS en Colombia.

Ahora que se han visualizado de manera general algunas posibles aplicaciones que tendría la arquitectura P2P en entornos móviles y se han observado los problemas más significativos que afectan las comunicaciones de datos en telefonía celular, se realizará a continuación un análisis general de su desempeño en un montaje de prueba sobre dispositivos reales.

A continuación se presentan los resultados de las pruebas realizadas con dispositivos móviles reales, servidores reales y los datos generados en el desempeño de algunas de las aplicaciones móviles P2P más representativas. El Ajedrez JXME[10] en representación de los juegos P2P, y el Punto de Encuentro Virtual P2P en representación de los sistemas de mensajería. Además se tienen en cuenta las tarifas que mantienen los dos operadores GSM del país para comunicaciones de datos GPRS en el cálculo de costos generados por el uso de las aplicaciones para un usuario móvil.

[10] Para mayor información acerca del ajedrez JXME, este se encuentra documentado en el anexo 5

5.2.3.1. Descripción del modelo de prueba

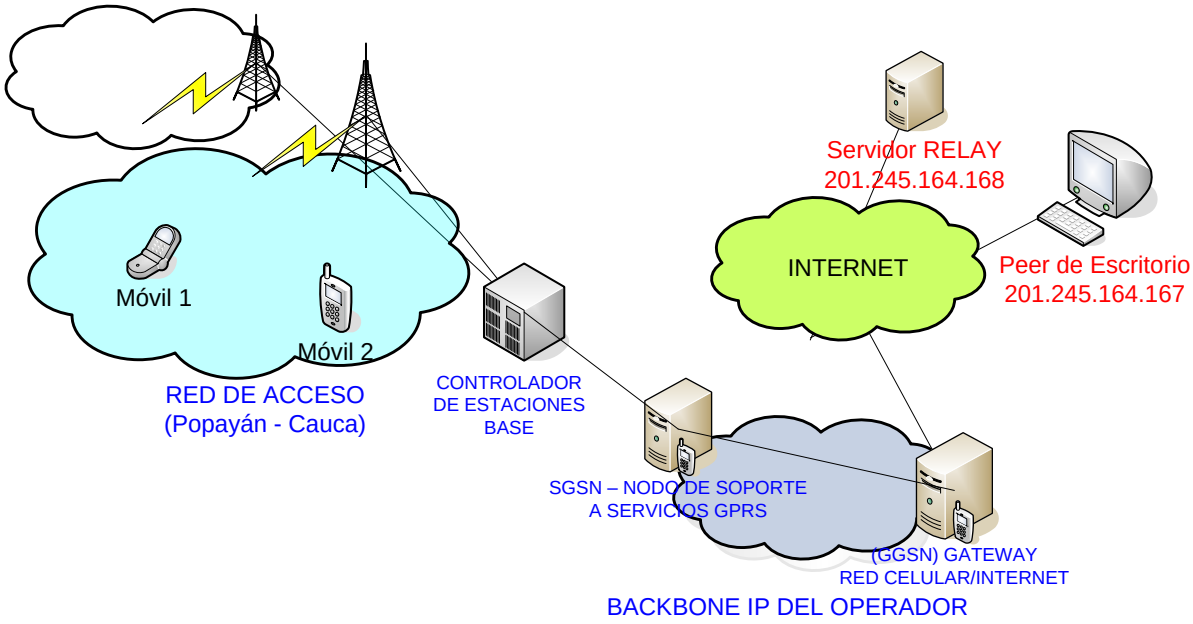


Figura 5.10 Modelo General de Red para la prueba de Aplicaciones P2P móviles.

La prueba realizada para determinar tiempos de respuesta, volúmenes de datos y costos de operación se llevó a cabo teniendo en cuenta la arquitectura de la Figura 5.13 y las siguientes condiciones:

- **Teléfonos Móviles (Móvil 1-2):**

  - Móvil1 y Móvil2: Teléfono celular Nokia 3595, serie 40, con capacidades GPRS y máximo tamaño de MIDlet de 64KB.

  - El intervalo de sondeo resulta ser un factor muy importante al momento de pensar en tiempos de respuesta y cantidad de tráfico cursado entre el móvil y la red. Para efectos de experimentación se escogió el valor mínimo "1", de tal manera que los resultados generados representaran una situación extrema en la que gran cantidad de datos fueran generados y los tiempos de respuesta fueran los menores.

- **La red del operador:**

  En este caso corresponde a la red GSM/GPRS de Colombia Móvil - OLA que opera en la banda de los 1900MHz. Y ofrece un ancho de banda uplink de 14Kbps máximo y downlink de entre 25Kbps y 56Kbps máximo, estas capacidades dependen de la carga de la red y de las condiciones en cuanto a cobertura y niveles de señal en la interfaz aérea en el momento de la conexión.

- **Servidor Relay:**

  Es un computador con 256MB de memoria RAM, 40GB de disco duro, un procesador Intel Pentium4 de ~1.8GHz de frecuencia de reloj, una interfaz de red y conectado a una red LAN Fast Ethernet con acceso a Internet mediante un enlace identificado con la dirección IP oficial 201.245.164.168.

- **Peer de Escritorio:**

  Es un computador con 256MB de memoria RAM, 40GB de disco duro, un procesador Intel Pentium4 de ~1.8GHz de frecuencia de reloj, una interfaz de red y conectado a una red LAN FastEthernet con acceso a Internet mediante un enlace identificado con la dirección IP pública 201.245.164.167.

Las pruebas fueron realizadas bajo los siguientes parámetros:

- Hora: entre las 3:00PM y las 5:15PM
- Día: jueves
- Ciudad: Popayán - Cauca
- Condiciones climáticas y disposición física: Día soleado en una ciudad húmeda, con los dispositivos móviles a una distancia media de la estación base más cercana del punto de pruebas. Sin obstáculos que afecten la señal radioeléctrica significativamente (Puertas metálicas grandes o estructuras que reflejen la señal de radio afectando la calidad de la comunicación).

5.2.3.2 Registro y análisis de resultados obtenidos

	Mensajería Instantánea (Punto de Encuentro)	Juego (Ajedrez P2P)
Tiempo Máximo de establecimiento de Conexión (Seg)	50	45
Retardo máximo entre el envío y recepción de un mensaje/jugada (Seg)	15	12
Retardo promedio entre el envío y recepción de un mensaje/jugada (Seg)	7	8
Volumen de datos promedio en 6 minutos (KBytes)	Enviado: 45,806KB Recibido: 28,235KB Total: 74,041KB	Enviado: 32,615KB Recibido: 25,645KB Total: 58,260KB
Costo de una Sesión de 30min.(Pesos $)	$18 por 1KB -> $6660(*1) $32 por 1KB->$11840(*2)	$18 por 1KB ->$5220(*1) $32 por 1KB ->$9280(*2)

Tabla 5.2 Resultados de tráfico en pruebas de aplicaciones JXME sobre una red GSM/GPRS

Nota1: (*1) Valor estándar de los planes de datos de Comcel, a Enero de 2005.
(*2) Valor estándar de los planes de datos de OLA, a Enero de 2005.
Estos valores pueden variar de acuerdo al plan de datos que tenga el usuario con su operador. Por ejemplo, existen planes de tipo corporativo en los cuales valor por KiloByte transferido se reduce a $8 y hasta $5 pesos[11].

Nota2: La cantidad de tráfico registrada en cada sesión de 6 minutos fue tomada gracias al contador de datos GPRS que incluye cada uno de los teléfonos Nokia 3595 en su Menú 2-7.

De los datos registrados en la tabla 5.2, se puede ver que la cantidad de datos enviados es mayor que los datos recibidos, para ambas aplicaciones, debido a que el mecanismo

[11] Ver planes de datos en el sitio web de cada uno de los operadores:
Comcel: http://comcel.com.co/aplicaciones3gsm/planes_datos.php
Colombia Móvil - OLA: http://www.ola.com.co/

de sondeo es un evento periódico, en tanto que el envío de un mensaje en el caso del punto de encuentro, o de una jugada en el ajedrez, ambos corresponden a eventos asíncronos no periódicos que pueden ser más o menos en un intervalo de tiempo.

En cuanto al ancho de banda, la cantidad total de datos generada por la aplicación no representa una carga significativa para la red GSM/GPRS sobre la que se experimentó.
Ya que en promedio, por usuario, se tiene:

Para el Punto de Encuentro:

~74KB/6min X 1min/60seg X 8bits/1Byte = 1,6444Kbps

Para el Ajedrez:

~58KB/6min X 1min/60seg X 8bits/1byte = 1,2888Kbps

Por otra parte, se ratifica el análisis de la latencia presentada por la red durante la ejecución de la aplicación, realizado en el apartado 5.2.1. La cual se le atribuye principalmente al retardo introducido por la interfaz aérea y al tiempo de procesamiento por parte del terminal móvil. El retardo introducido en la interfaz aérea es ocasionado por las condiciones climáticas y ambientales presentes al momento de realizar las pruebas, se tenía un clima húmedo lluvioso y un terreno de construcciones medianamente altas con algunos obstáculos naturales (Árboles, montañas…), por otra parte no hay que dejar de lado el retardo ocasionado por el procesamiento de la aplicación en el terminal móvil. Hay que tener en cuenta que java para su ejecución requiere de una máquina virtual que traduce los *bytecodes* [12] en instrucciones entendibles para un dispositivo o plataforma en particular, a esto se le puede asignar parte en la latencia global presentada durante la ejecución de las pruebas reales.

Existen otras condiciones que afectan el desempeño de las aplicaciones, como la disponibilidad del servicio de datos y los recursos destinados para ello, actualmente (Enero de 2005) solo se tiene destinado para comunicaciones de datos GPRS un 10% del total de la infraestructura disponible en la red de Colombia Móvil - OLA.
Finalmente, en cuanto al costo generado por el uso de la aplicación resulta un poco alto para usuarios particulares promedio. Esto es relativo, ya que depende del nivel de ingresos de cada usuario, quien finalmente determinará si resulta costoso o no.

[12] Los bytecodes son un conjunto de códigos compilados que en realidad son instrucciones para la máquina virtual de java instalada en el dispositivo, quien interpretará dichos códigos. Esto permite escribir código fuente una sola vez y ejecutarlo en múltiples dispositivos y plataformas sin realizar cambios sobre dicho código.

Sin embargo, si se realiza un cálculo en términos generales de lo que una persona (Entre los 10 y los 25 años de edad) gasta en video juegos (Promedio 1 hora = $1200, 30Min = $800), navegando en Internet especialmente en salas de chat (Promedio 1hora = $2000), alquilando consolas de juego (Promedio 24Horas = $10000) y otras actividades similares, el costo promedio del uso de aplicaciones P2P en Colombia resulta relativamente alto aún, ya que las tarifas de datos para el soporte a este tipo de aplicaciones siguen siendo aún elevados. Se estima que dentro de algunos meses el precio por utilizar aplicaciones de datos en Colombia disminuya como consecuencia de la competencia que libran los operadores de telefonía móvil en el mercado de las telecomunicaciones tratando de ofrecer más y mejores servicios a los clientes.

VI. CONCLUSIONES

- El Punto de Encuentro virtual surge como una alternativa a las aplicaciones de mensajería para dispositivos móviles existentes en la actualidad, las cuales se basan en WML y/o protocolos propietarios, como *MSN Messenger, Yahoo Messenger o Jabber;* con la ventaja de tener a disposición la robustez de un lenguaje como Java que permite al programador extender sus horizontes a aplicaciones aún más potentes en otros campos, como e-commerce, seguridad y monitoreo de bienes (carros, automóviles, …), turismo, servicios de información, entre otros.
- La arquitectura y las funciones del Punto de Encuentro Virtual son reutilizables casi por completo en otros entornos de aplicación usando los conceptos de peer, peer groups e intercambio de archivos. Muestra de esto son las aplicaciones presentadas en el apartado 5.1.
- La contribución más grande del proyecto JXME a la comunidad en Internet es el aporte de un conjunto de protocolos para el networking P2P con dispositivos móviles que facilita a los programadores una plataforma completa de herramientas de comunicación de tal manera que no deben preocuparse de generar sus propios protocolos de networking sino solamente ocuparse de la aplicación, estandarizando de esta manera los mecanismos de comunicación para entornos P2P. Por otra parte, la naturaleza Open Source de JXTA/JXME permite que los protocolos estén en constante evolución y mejoramiento, ya que toda la comunidad JXTA en Internet contribuye con sus observaciones para esto.
- Antes de prometer el desarrollo de una aplicación con alguna tecnología en especial, debe verificarse que la tecnología soporta las características que se pretenden alcanzar con el desarrollo del proyecto. Por ejemplo, inicialmente se quiso construir un sistema de intercambio de archivos P2P pero Java solo presenta la opción de examinar la galería de fotos, juegos y la lista de contactos de los dispositivos móviles mediante un paquete opcional (FCOP – File Connection Optional Package / JSR -75), el cual era soportado por unos pocos dispositivos en el mundo, por ser un paquete opcional no estaba muy difundido. Y contradecía la intención inicial de construir una aplicación que se pudiera ejecutar en gran parte de los dispositivos móviles en Colombia.
- Tecnologías como SMS (Servicios de Mensajería Corta), MMS (Servicios de Mensajería Multimedia) o aplicaciones desarrolladas para navegadores XHTML en arquitecturas Cliente/Servidor se presentan en la actualidad como las opciones

más viables en el mercado tecnológico en Colombia si se pensara en un despliegue comercial, ya que tienen mayor presencia entre los usuarios, los terminales que responden a estas tecnologías superan a los que soportan acceso a Internet Móvil y mucho más a los que soportan aplicaciones J2ME. Por esta razón, las aplicaciones móviles basadas en J2ME serían viables actualmente en los países europeos o en los países norteamericanos donde los servicios de datos se ofrecen a costos mucho más bajos que en Latinoamérica y Colombia, donde hasta ahora se están desplegando servicios de Internet Móvil u otras aplicaciones de datos para empresas y servicios corporativos, pero no se ha promovido en masa la utilización de dichas aplicaciones y servicios. Es necesario esperar a que evolucione la telefonía celular en Colombia, se rebajen los costos por el uso del celular para navegar por Internet, se popularicen los terminales con soporte a aplicaciones Java y los desarrolladores generen aplicaciones interesantes e innovadoras.

- A través del desarrollo del Punto del Encuentro, se detectaron fallos en cuanto al manejo de pipes y a la gestión grupos. Por una parte, una pipe creada no puede ser eliminada por un peer, la única manera de cerrarla o destruirla es esperar a que caduque su tiempo de vida (1 año). Para superar este problema se implementó un sistema de reconocimiento y actualización de información de presencia, en el cual cada peer informa de su salida o llegada a un peer group enviando un mensaje al grupo.

  Otro fallo que se encontró estuvo relacionado con el tipo de pipe, JXTA ha definido supuestamente tres tipos de pipe, a saber: Propagate, Unicast y Secure Unicast, a las cuales se les asignan diferentes conceptos, como puede verse en el apartado 2.4.5. Sin embargo, todas presentan la misma funcionalidad, lo único que cambia es el nombre.

  Un último fallo que se detectó fue la imposibilidad de gestionar grupos desde un peer JXME, ya que a pesar de unirse a un grupo, no se recibía ninguna respuesta por parte de este. Esto se debía a que el servicio Proxy no se estaba activando para cada grupo, por lo cual fue necesario realizar modificaciones correspondientes en el servidor relay.
- En cuanto a los factores que más afectan el desempeño de las aplicaciones P2P basadas en JXME sobre redes celulares, se puede concluir que el ancho de banda de una red GSM/GPRS en Colombia es suficiente para satisfacer los volúmenes de tráfico generados por la aplicación, por lo cual no resulta ser un factor tan crítico en el rendimiento como lo es la latencia. La latencia, como se mencionó en

el apartado 5.2.2. está presente en cualquier red de comunicaciones, y es incrementada por cada uno de los componentes de la red, el procesamiento en la estación móvil, entre otros, en las redes celulares es afectada en gran parte por la interfaz aérea, la cual presenta condiciones variables de acuerdo al clima, la disposición de las antenas y de los clientes móviles en el momento de la ejecución de la aplicación. Por otra parte, la arquitectura P2P presenta una desventaja frente a la arquitectura Cliente/Servidor en este sentido, como se pudo observar en el apartado 5.2.1, la cual se aumenta por la distancia que debe recorrer la señal durante la ejecución de la aplicación.

VII. RECOMENDACIONES Y TRABAJOS FUTUROS

- A través del proceso de desarrollo de aplicaciones para dispositivos móviles es muy importante tener en cuenta las limitaciones y capacidades de los dispositivos a los cuales va dirigida la aplicación y tener a disposición dispositivos reales sobre los cuales probar los prototipos generados, ya que el despliegue de interfaces y las características de rendimiento no son completamente visibles en la emulación realizada por las herramientas de desarrollo. El no tener en cuenta esta recomendación podría prolongar el proceso de desarrollo al tener que hacer modificaciones considerables en el código de la aplicación para que se ajuste a las características técnicas de los dispositivos reales.
- Uno de los objetivos al desarrollar el Punto de Encuentro era generar una aplicación que pudiera ser ejecutada por la mayor parte de dispositivos con soporte J2ME, esto conllevó a diseñar una aplicación apropiada para celulares de gama baja, específicamente los que satisfacen el MIDP 1.0. Estos dispositivos generalmente no presentan características avanzadas como captura de imágenes, videos cortos y sonidos, como sí las presentan los de gama alta, que satisfacen el MIDP2.0. Hecho que abre la posibilidad de extender la capacidad del Punto de Encuentro a un subsistema de intercambio de archivos multimedia (Fotografías, cortos videos, sonidos…), especial para dispositivos que cumplen el MIDP2.0, extendiendo el alcance del sistema a más móviles en el mundo y haciendo más interesante la experiencia de los usuarios en el Punto de Encuentro.
- El Punto de Encuentro no tiene un sistema de intercambio de fotografías, esta característica mejoraría notablemente la experiencia de los visitantes del Punto de Encuentro, sin embargo esto se podría implementar reutilizando el sistema de transferencia de datos que se utiliza para consultar el perfil de un usuario en el Punto de Encuentro.
- Como un futuro trabajo en este sentido, puede construirse un sistema P2P de intercambio de archivos (File Sharing) para móviles Symbian, los cuales a través del tiempo han cobrado gran popularidad. Como el desarrollo de aplicaciones Symbian se basa en C++, habría necesidad de implementar el protocolo JXTA en este lenguaje, sin embargo en la actualidad se cuenta con adelantos en esta parte en el proyecto "*Jxta-C"* el cual ha implementado los protocolos JXTA en lenguaje C, lo cual sería un punto de partida para el desarrollo de dicho proyecto.

IX. REFERENCIAS

- Rendón Gallón Álvaro. "*EL LENGUAJE UNIFICADO DE MODELADO (UML)"*. Facultad de Ingeniería Electrónica Y Telecomunicaciones. Universidad del Cauca. Popayán. Mayo de 2000.

- Grupo de Ingeniería Telemática. "*MODELO PARA CONSTRUCCIÓN DE SOLUCIONES"*. Universidad del Cauca, 2001.

- Gálvez Sergio, Ortega Lucas. "*JAVA A TOPE (J2ME)"*. Universidad de Málaga. España. Diciembre de 2003

- Muchow John W. "*CORE J2ME Technology and MIDP"*. Editorial Prentice Hall, Diciembre de 2001.

- Wilson Brendon. "*JXTA"*, http://www.brendonwilson.com/projects/jxta/pdf/, Junio 6 de 2002.

- Sun Microsystems Inc. & The Internet Society, "*JXTA PROTOCOLS*", http://spec.jxta.org/v1.0/docbook/JXTAProtocols.html#id2797939, 2004.