Cátedra de Sistemas Distribuidos
Escuela de Ingeniería Electrónica
Departamento de Sistemas e Informática

Monografía

Redes Peer to Peer y Tecnología JXTA

Realizado por Mariano Disanzo Leg. D-1718/3

Profesor Ing. José L. Simón

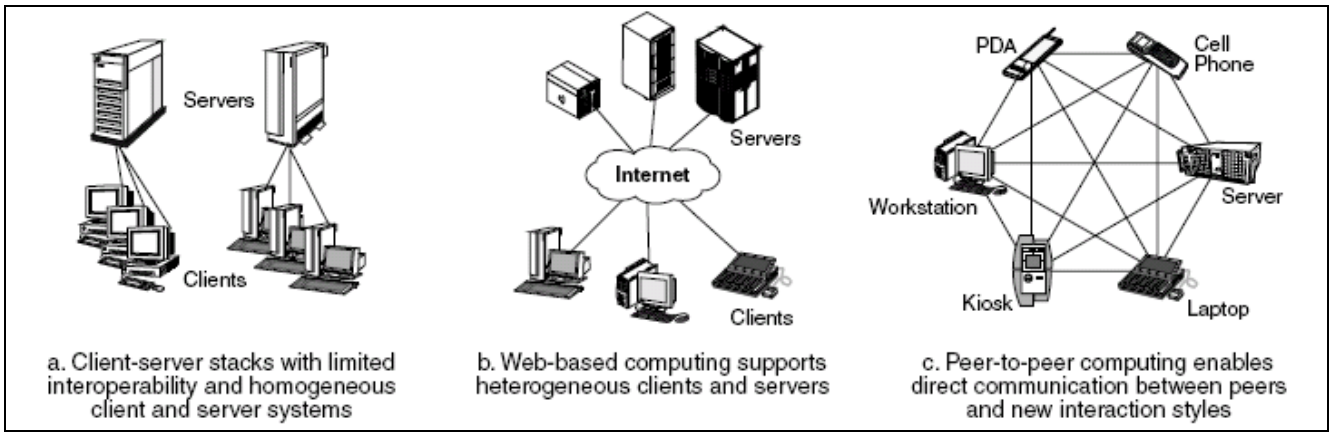
Año 2006

ÍNDICE

1. Introducción

Para millones de consumidores individuales y de empresas globales, usar redes de computadoras como una plataforma para comunicarse, colaborar y compartir es tan común como usar el teléfono o el correo postal. Los usuarios particulares utilizan mensajería instantánea y correo electrónico para mantenerse en contacto y realizar operaciones comerciales, trabajan remotamente a través de intranets y portales web y también usan servicios web para trabajar o jugar. Las redes y los distintos dispositivos usados para acceder a ellas (PCs, laptops, PDAs, celulares) son hoy parte indispensables para los negocios y la vida de la sociedad en general.

En la siguiente figura se observa la evolución de las redes de computadora desde cliente-servidor, World-Wide Web hasta peer to peer. El esquema detalla la conexión lógica, no física.



a. Client-server stacks with limited interoperability and homogeneous client and server systems

b. Web-based computing supports heterogeneous clients and servers

c. Peer-to-peer computing enables direct communication between peers and new interaction styles

Figura 1

En contraste con el diseño de Internet como interconexión plana de redes, la manera primaria en la cual las aplicaciones de red se han construido durante las dos décadas pasadas es sobre todo jerárquica, siguiendo el modelo cliente-servidor.
Al principio se utilizó en redes de área local (LAN) y las aplicaciones cliente-servidor requerían sistemas homogéneos (tanto clientes como servidores) y brindaban interoperatividad limitada o incluso nula entre las aplicaciones. (Figura 1a)

Internet hizo que el uso del modelo cliente-servidor fuese posible aceptando un cliente universal (Web Browser) que usa un protocolo de comunicación estandarizado (HTTP), que puede mostrar información descripta en un formato o lenguaje estandarizado (HTML) y que puede ejecutar aplicaciones usando Java y Tecnología Extensible Mark-up Language (XML) (Figura 1b). Cualquiera que posea un dispositivo con acceso web, desde computadoras a celulares, puede utilizar este modelo. Solo necesita conectarse a un servidor web con una ubicación y nombre conocidos.

Los programas de intercambio de archivos a través de redes peer to peer (P2P) han provisto solución a necesidades de los usuarios particulares y el uso de estas aplicaciones ha cambiado la forma en que la gente piensa respecto de como y para que se usa internet. Volviendo a las raíces para lo cual internet fue creada, los usuarios pueden interactuar con otros directamente, sin necesidad de un servidor web, un moderador de chat o un bulletin board system. De repente internet parece crecer a medida que los usuarios tienen acceso a un nuevo estilo de computación que no necesariamente requiere de un grupo de clientes y de servidores con una relación jerárquica. (Figura 1c)

El rápidamente creciente uso de aplicaciones distribuidas y servicios sugieren un modelo que complementa al de cliente-servidor (no lo reemplaza totalmente), dándole énfasis a la comunicación directa entre usuarios. En lugar de tener una relación vertical, ambos cliente y servidor coexisten como pares en la red a pesar de las distintas características de performance de cada uno. Con tanto poder de cómputo, capacidad de almacenamiento y ancho de banda afuera de los centros de datos y en las manos de los usuarios particulares por todo el mundo, la aceptación de compartir recursos personales buscando el beneficio común aceleró el uso de esta tecnología.

Dos grandes ventajas de las redes son su naturaleza dinámica y la rápida expansión de información y recursos disponibles a través de ellas, tanto en servidores web de acceso público como en intranets corporativas. El modelo peer to peer ofrece muchos beneficios para tratar con un crecimiento en número de usuarios y dispositivos conectados, ancho de banda, contenido, aplicaciones y poder de cómputo. El verdadero modelo peer to peer hace que sea mas fácil e intuitivo encontrar y compartir recursos para los nuevos usuarios brindado mas canales de conexión directa a dispositivos en lugares extremos de la red.

La existencia de múltiples pares (peers) hace que las aplicaciones P2P tengan una alta disponibilidad, ya que ellas no dependen de un único servidor central. En el caso de que alguno de los pares fallase, los restantes continúan atendiendo los requerimientos proveyendo así servicio ininterrumpido a los usuarios. Las aplicaciones y servicios P2P de hoy permiten una comunicación interactiva con casi cualquier dispositivo de internet, ayudando a entregar la información correcta y los servicios a cualquier parte donde llegue la red y dando mejor acceso a los recursos de red sin comprometer la seguridad.

P2P y en particular la tecnología JXTA brinda servicios distribuidos, servicios y redes, todo junto. Ésta tecnología JXTA se basa en un conjunto de estándares abiertos que habilita a los dispositivos y redes a operar entre ellos y provee flexibilidad ya que las aplicaciones no están sujetas al uso de un sistema propietario. Permite a los desarrolladores crear servicios P2P mas rápido ya que saca la complejidad de la red y de los ambientes operativos proveyendo una infraestructura completa. Les permitirá a los programadores concentrarse en el desarrollo de su propia aplicación mientras fácilmente crean un software para sistemas distribuidos que es flexible, interoperable y funciona en cualquier par de la red.
JXTA y las aplicaciones del tipo P2P apuestan a un estilo más socialista de computación con usuarios con capacidad de buscar y compartir en forma dinámica con su comunidad o grupo de trabajo y que puede ayudar en las comunicaciones de negocios, colaborando y compartiendo recursos a través de la red.

2. Peer to Peer

Las primeras tecnologías peer to peer surgieron hace mas de diez años atrás con el objetivo de facilitar la comunicación y la utilización de recursos dentro de las empresas. Hoy en día P2P describe el modelo general de uso de comunicación directa entre todos los dispositivos de la red. Brinda conectividad a toda la red permitiendo inclusive al dispositivo más remoto comunicarse y colaborar. Con el uso de P2P las aplicaciones pueden ser mas colaboradoras entre si, orientadas a la comunicación y la información, pueden ser mas rápidas y precisas.

P2P posibilita una serie de aplicaciones innovadoras, incluyendo entre otras:
- Compartir archivos de cualquier tipo.
- Nuevas formas de distribución y entrega de contenido.
- Mensajería instantánea y comunicación directa entre dispositivos. Los sistemas de mensajería instantánea pueden localizar usuarios rápidamente mediante búsquedas distribuidas, permitiendo una comunicación directa usuario a usuario y colaboración, independientemente del proveedor de servicio.
- Trabajos y juegos en forma cooperativa.
- Búsqueda distribuida e indexación, que permiten búsquedas más “profundas” de contenido en internet, entrega de los resultados en forma casi inmediata ya que las búsquedas P2P son distribuidas, en paralelo y asíncronas. Las búsquedas de hoy en día llevadas a cabo por los buscadores (search engines) están limitadas por el tiempo que les lleva a sus eslabones buscadores (crawlers) recorrer los sitios a indexar. Por el contrario, los resultados de búsquedas P2P son más relevantes y pueden ser más enfocados. Por ejemplo calificando el grupo de pares y calificando la búsqueda se pueden encontrar resultados más precisos sobre un espacio de búsqueda mucho más grande.
- Compartir recursos de CPU y de almacenamiento para una mayor rentabilidad del capital invertido.

Las aplicaciones que son apropiadas para este modelo son las que pueden soportar la conexión y desconexión de pares individuales. En este modelo, más probabilístico que determinístico, si no se alcanzó el resultado deseado se puede tratar de nuevo mas tarde. La información intercambiada en ambientes P2P esta acotada en tiempo y es precisa, pero aún así los resultados puede variar de un momento a otro dependiendo del grupo del que el par sea miembro.

Gracias a la existencia de múltiples pares en el grupo se consigue una alta disponibilidad haciendo que en todo momento exista un par en el grupo que esté disponible para responder a los requerimientos de un usuario. Esto se opone rotundamente a los modelos de computación tradicionales donde una alta disponibilidad se alcanza a través de sistemas de balance de cargas y esquemas de respuesta a fallas que pueden ser ambos altamente complejos.

P2P aprovecha recursos disponibles de procesamiento, almacenamiento y ancho de banda de sistemas en todo el mundo y funciona porque las personas se dan cuenta de que es bueno compartir sus propios recursos para después poder aprovechar los beneficios que dan los recursos de otros cuando se necesitan.

De muchas maneras el mundo P2P está complementando al modelo jerárquico cliente-servidor, trabajando conjuntamente con él, no reemplazándolo. De allí el termino juxtapose que en inglés significa ubicarse lado a lado. JXTA es una abreviación de juxtapose.

3. Tecnología JXTA

El proyecto JXTA estandariza un grupo de protocolos para construir redes virtuales P2P. Los protocolos JXTA definen los requerimientos de red mínimos para que los pares formen y se unan a una red virtual mediante una capa de red genérica. El proyecto JXTA posibilita a los programadores y desarrolladores de aplicaciones, no solo a los administradores de red, diseñar la topología de la red para adecuase a los requerimientos específicos de la aplicación. Se pueden crear múltiples redes virtuales Ad Hoc y dinámicamente mapearlas en una única red física.

Hasta hace poco la tecnología P2P había sido usada principalmente en aplicaciones unifuncionales, como por ejemplo mensajería instantánea. Llevando el concepto de P2P mucho más allá, el fundador de Sun Microsystem y jefe de científicos, Dr. Bill Joy concibió la idea de JXTA como un medio de integración de P2P en el mismo núcleo de la arquitectura de red.

Sun introdujo la tecnología JXTA en el 2001 con un modelo de licencia de derechos libres, código libre (open source). Hoy en día más de 16000 miembros registrados de la comunidad de desarrolladores contribuyen activamente con la evolución de esta tecnología y miles mas están investigando y trabajando con ella. Las empresas, agencias gubernamentales e instituciones educacionales están planeando adoptar esta tecnología como su principal solución P2P. Hay mas de 80 proyectos actualmente en desarrollo que van desde telecomunicaciones y servicios de mensajería a aplicaciones cooperativas en tiempo real y redes para compartir y distribuir contenido. Hay varios libros publicados sobre tecnología JXTA para ayudar a llegar a una mayor audiencia y apoyar la actividad de los desarrolladores. Es activamente apoyada por una comunidad creciente de desarrolladores P2P y ha sido ampliamente adoptada estando ya disponible como una opción comercial.

La tecnología JXTA consiste en un conjunto de simples protocolos abiertos (open source) peer to peer que permiten a los dispositivos de la red comunicarse, colaborar y compartir recursos.
En la figura 2 se observa un esquema de como los pares JXTA crean una red virtual sobre las redes existentes, ocultando así su complejidad.

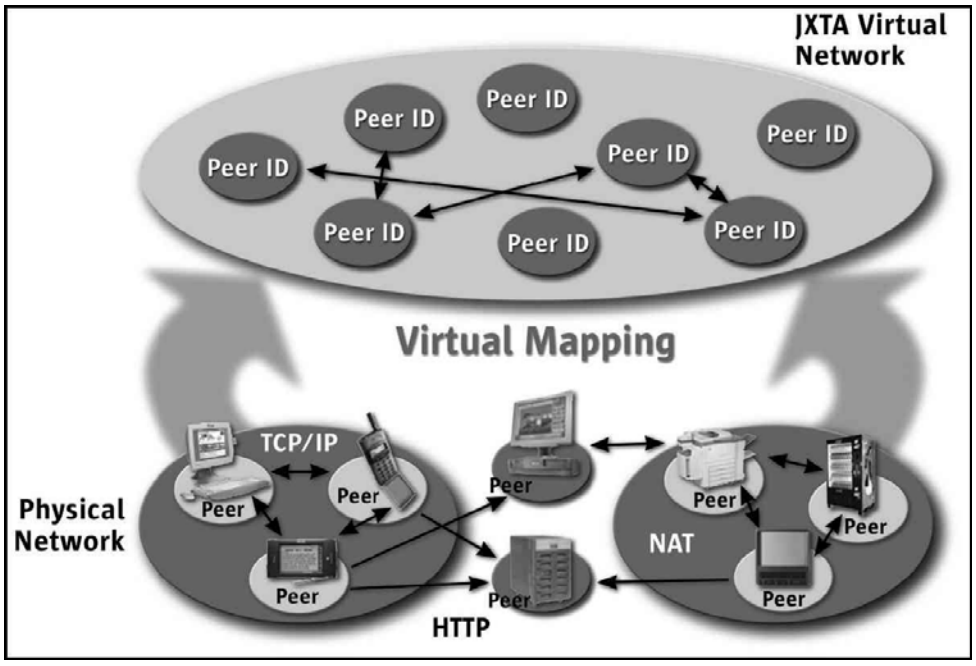


Figura 2

En esta red virtual JXTA cualquier par puede interactuar con otros pares a pesar de su ubicación, tipo de dispositivo o sistema operativo; Inclusive cuando algunos pares y recursos están ubicados detrás de un firewall o trabajan sobre un tipo de transporte de red diferente. Así el acceso a los recursos de red no está limitado por incompatibilidades de plataforma o los permisos de una arquitectura jerárquica cliente servidor. La tecnología JXTA funda sus objetivos en la ubicuidad, la independencia de plataformas, la interoperatividad y la seguridad. Puede correr sobre cualquier dispositivo que acceda a la red, incluyendo teléfonos celulares, PDAs, sensores electrónicos, computadoras de escritorio y servidores. Al estar basado en tecnologías probadas y estandarizadas tales como HTTP, TCP/IP, y XML la tecnología JXTA no depende de un lenguaje de programación en particular ni de la plataforma de red ni del sistema operativo y por eso puede trabajar con cualquier combinación de ellos.

La interoperatividad es el objetivo central y JXTA está diseñada para permitir a los pares interconectados que se ubiquen y se comuniquen entre si, participen en actividades cooperativas y ofrezcan servicios unos a otros igualmente sobre distintas plataformas y redes. Los mecanismos de seguridad integrados como Transport Layer Security (TLS), certificados digitales y autoridades certificadoras brindan seguridad y facilitan el libre flujo de comunicación.

Para los programadores y desarrolladores la tecnología JXTA provee un conjunto de bloques constructivos que forman una base sólida para aplicaciones distribuidas y soportan las funciones requeridas por cualquier sistema P2P. Las aplicaciones distribuidas requieren un gran trabajo de infraestructura y construir un framework es comúnmente una actividad que consume mucho tiempo y recursos, y en muchos casos está fuera del alcance para muchos programadores. Usando ésta tecnología las empresas se pueden concentrar en crear aplicaciones innovadoras y no tener que inventar (o re inventar) la infraestructura P2P. Dado que los programadores no tienen la tarea de inventar su propio framework, el desarrollo es más rápido y la empresa se beneficia con aplicaciones y servicios que salen al mercado en menos tiempo.

El proyecto JXTA apunta a un mundo donde cada par independientemente de la plataforma de software o hardware pueda contribuir y beneficiarse al estar conectado a millones de pares mediante la formación de múltiples redes virtuales. Este proyecto está diseñado para ser independiente del lenguaje de programación (tales como C o Java), plataforma del sistema (como sistemas operativos Windows o Unix), definición de servicios (RMI o WSDL) o protocolos de red (TCP/IP o Bluetooth). El proyecto JXTA ha sido diseñado para ser implementado en cualquier dispositivo que se conecte a la red incluyendo sensores, dispositivos electrónicos, PDAs, electrodomésticos, routers, computadoras de escritorio, servidores de centros de datos y sistemas de almacenamiento.

El proyecto JXTA tiene una serie de objetivos, derivados de los defectos de muchos sistemas peer to peer existentes o en desarrollo

- *Interoperabilidad.* La tecnología JXTA está diseñada para permitir que los pares interconectados se ubiquen entre si, se comuniquen, participen en actividades cooperativas y se ofrezcan servicios entre diferentes sistemas P2P y diferentes comunidades. Muchos sistemas P2P son construidos para proveer un único tipo de servicio. Por ejemplo, Napster brinda intercambio de archivos musicales, Gnutella brinda intercambio de archivos de cualquier tipo y AIM brinda mensajería instantánea. Dadas las diversas características de estos distintos servicios y la falta de una estructura subyacente común P2P, cada empresa desarrolladora tiende a crear sistemas incompatibles, ninguno de ellos es capaz de operar con alguno de los otros. Esto significa que cada empresa crea su propia comunidad de usuarios P2P haciendo un esfuerzo mayor al desarrollar software y bases comunes a todos los sistemas P2P. Es mas, para que un par pueda participar en múltiples comunidades organizadas por diferentes implementaciones P2P, este par debe soportar múltiples implementaciones, cada una para cada sistema o comunidad y servir como punto de agregado (Aggregated Network Access Point o ANAP). Esta situación nos recuerda a la internet pre-browser, donde para tener acceso a internet se debía suscribir a determinadas empresas (AOL, Prodigy o CompuServ). El resultado fue que el usuario estaba encerrado en una comunidad y el proveedor debía ofrecer sus servicios o contenidos de forma de adaptarse a como operaba cada comunidad. El proyecto JXTA apunta a dar al mundo P2P lo que el browser dió a internet.

- *Independencia de plataforma.* La tecnología JXTA esta diseñada para ser independiente del lenguaje de programación (tales como C o Java), plataforma del sistema (como sistemas operativos Windows o Unix), definición de servicios (RMI o WSDL) o protocolos de red (TCP/IP o Bluetooth). Muchos sistemas P2P de hoy ofrecen sus prestaciones o servicios a través de un conjunto de APIs (Application Programming Interface) que son entregadas para un determinado sistema operativo usando un determinado protocolo de red. Por ejemplo un sistema puede ofrecer un grupo de APIs en C++ sobre un sistema que corre inicialmente sólo en Windows sobre TCP/IP, mientras otro sistema ofrece una combinación de APIs en C y Java corriendo sobre un sistema UNIX / LINUX sobre TCP/IP pero que también requiere HTTP. Un programador desarrollador P2P está forzado a elegir un grupo de APIs para programar y en consecuencia a que grupo de clientes P2P apuntar. Como hay muy poca probabilidad que los dos sistemas sean interoperables, si el programador quiere ofrecer el mismo servicio a ambas comunidades deberá desarrollar el mismo servicio dos veces para dos plataformas P2P o desarrollar un puente entre ellas. Ambas posibilidades son ineficientes e impracticable dada la cantidad de plataformas P2P existentes.

- *Ubicuidad.* La tecnología JXTA esta diseñada para ser implementable en sobre cualquier dispositivo digital con acceso a la red, incluyendo teléfonos celulares, PDAs, sensores electrónicos, computadoras de escritorio y servidores. Muchos sistemas P2P, especialmente aquellos ofrecidos por las compañías mas jóvenes, tienden a elegir (no sorpresivamente) Microsoft Windows como su plataforma objetivo de desarrollo. La razón de esta elección es apuntar a la plataforma más instalada para obtener más ganancias. El resultado es que el sistema tiene muchas dependencias de características específicas de la plataforma “Winintel” (Windows + Intel). Esto es muchas veces no una decisión técnica sino una realidad ingenieril con tiempos de desarrollo ajustados y recursos limitados. Este es un punto de vista bastante pobre ya que P2P no está apuntado sólo a conexiones PC a PC. La que denominamos máquinas o PCs “Winintel” están al punto medio del espectro de hardware de

computación y es más probable que la proliferación de la tecnología P2P se dé entre los dos extremos de este espectro, grandes sistemas de empresas y pequeños sistemas orientados a los usuarios. De hecho sería un error apuntar a un único segmento con un tipo específico de plataforma.
El proyecto JXTA vislumbra un mundo donde cada par independientemente de su plataforma de software o hardware pueda contribuir y beneficiarse al estar conectado a millones de otros pares.

En el nivel más alto de abstracción la tecnología JXTA es un conjunto de protocolos. Cada protocolo está definido por uno o más mensajes intercambiados entre los participantes de la comunicación. Cada mensaje tiene un formato predefinido y puede incluir varios campos de datos. En este aspecto está relacionado con TCP/IP. Mientras TCP/IP vincula nodos de internet, la tecnología JXTA conecta pares nodos entre ellos. TCP/IP es independiente de la plataforma gracias a que es un conjunto de protocolos, al igual que JXTA. Es mas, JXTA es independiente del transporte y puede utilizar tanto TCP/IP como cualquier otro estándar de transporte.

Están definidos los siguientes protocolos:
• Peer Discovery Protocol (Protocolo de Descubrimiento)
• Peer Resolver Protocol (Protocolo de Resolución)
• Peer Information Protocol (Protocolo de Información)
• Rendezvous Protocol (Protocolo de puntos encuentro) (Rendezvous es un lugar donde la gente se encuentra)
• Pipe Binding Protocol (Protocolo de unión de tuberías)
• Endpoint Routing Protocol (Protocolo de Ruteo de Extremos)

Se usará la terminología en inglés ya que es la que se aplica en la práctica, la traducción es meramente informativa. Además los nombres mismos de los protocolos no son obvios y si se traducen pueden llegar a confundir aún más. Por ejemplo el Peer Discovery Protocol es el que usa un par para llevar a cabo la acción de descubrir. El nombre puede sugerir que el protocolo se usa para descubrir pares solamente pero puede ser usado para descubrir pares, grupos de pares y cualquier otro tipo de anuncio.
Para dar soporte a estos protocolos la tecnología JXTA define una serie de conceptos que incluyen al peer (par), peer group (grupo de pares), advertisement (anuncio), message (mensaje), pipe (tubería) y demás.
Los conceptos parecen muy simples, esto es deliberadamente así con el objetivo de mantener las especificaciones reducidas simples. Más importante aún, hay muchas áreas donde no hay un modo correcto de hacer las cosas o donde lo que se deba hacer depende de la naturaleza y el contexto de la aplicación. Este fenómeno se da intensamente en el área seguridad donde cada aplicación P2P puede elegir un esquema de autentificación diferente, un camino diferente para dar seguridad a la comunicación, un algoritmo de encriptación diferente para asegurar los datos, un esquema de firmas diferente para autentificación y diferentes políticas de control de acceso. Para estas áreas se tiende a subespecificar concentrándose en los mecanismos en vez de las políticas, de forma que los desarrolladores de aplicaciones puedan tener la máxima libertad para innovar y ofrecer soluciones competitivas. Dada esta subespecificación es importante distinguir lo que está definido por la tecnología JXTA y lo que la primera implementación (la versión 1.0) hace. Muchas veces la implementación elige hacer algo en una forma particular, pero esto no significa que JXTA está definido para comportarse de esa manera.

Identificadores. JXTA usa UUID (Universally Unique Identifier), datos de 128 bits para referirse a una entidad (ya sea un par, un anuncio, un servicio, etc). Es fácil de garantizar que cada entidad tendrá un único UUID dentro de un ambiente de ejecución local, pero dado que no se asume un estado global, no hay manera de garantizar absolutamente la unicidad en una comunidad entera que puede consistir en millones de pares. Esto no es un problema grave ya que UUID se usa como un identificador interno. Se hace importante solo después de estar ligado a otra información tal como nombre y dirección de red.

Anuncio. Un anuncio es un documento XML estructurado que nombra, describe y publica la existencia de un recurso tal como un par, un grupo de pares, un pipe o un servicio. La tecnología JXTA define un conjunto básico de anuncios. A partir de los tipos básico se pueden formar mas subtipos de anuncios usando esquemas XML.

Pares. Un par es una entidad que puede hablar los protocolos requeridos por un par. Esto se relaciona con Internet donde un nodo es una entidad que puede hablar los protocolos IP. De tal forma un par se puede manifestar en la forma de un procesador, un proceso, una máquina o un usuario. Es importante destacar que un par no debe necesariamente hablar los seis protocolos a los que se refirió anteriormente, un par puede trabajar a un nivel reducido si no soporta un determinado protocolo.

Mensajes. Los mensajes están diseñados para ser usados sobre un transporte asíncrono, no confiable y unidireccional. Por ello un mensaje es diseñado como un datagrama, contiene una envolvente con encabezado y cuerpo. La envolvente contiene un encabezado, un resumen del mensaje (digest), la fuente (opcionalmente) y el destino. El destino es un destino lógico dado en forma de URL sobre cualquier transporte de red capaz de enviar y recibir mensajes tipo datagrama. Los destinos son normalmente mapeados a direcciones físicas por una capa de mensajes. El formato de los mensajes está diseñado para soportar múltiples estándares de

transporte. El cuerpo contiene un número variable de bytes y una o más credenciales usadas para identificar al remitente ante el destinatario. El formato exacto y el contenido de las credenciales no son especificados. Por ejemplo una credencial puede ser una firma que dé prueba de la integridad del mensaje y/u origen. Otro ejemplo, el cuerpo de un mensaje puede ser encriptado y la credencial provee información de como desencriptar el contenido.

Grupos de Pares. Un grupo de pares es una entidad virtual que habla el conjunto de protocolos de grupo de pares. Comúnmente un grupo de pares es una colección de pares cooperativos que brindan un conjunto común de servicios. La especificación no indica cuando, donde o porque crear un grupo de pares, o el tipo de grupo, o la membresía del grupo. Ni siquiera define como crear un grupo. De hecho la relación entre un par y un grupo de pares puede ser de alguna manera metafísica. JXTA no se preocupa por la serie de eventos que dieron existencia a un par o a un grupo de pares. Más aún, no limita a cuantos grupos puede pertenecer un par o si se pueden formar grupos anidados. Si se define como descubrir grupos de pares usando el Protocolo de Descubrimiento de Pares (Peer Discover Protocol).
Hay un grupo especial llamado World Peer Group que incluye a todos los pares JXTA. Esto no significa que los pares pertenecientes a este grupo especial puedan descubrirse y comunicarse entre ellos siempre. Por ejemplo ellos pueden estar separados por una partición de red. Se participa del World Peer Group por defecto.

Pipes. Pipes (o tuberías) son canales de comunicación para enviar y recibir mensajes y son asincrónicos. También son unidireccionales, así que hay input pipes (tuberías de entrada) y output pipes (tuberías de salida). Los pipes también son virtuales ya que la salida de uno de ellos puede ser enlazada a uno o más pares. Un pipe es comúnmente unido dinámicamente a un par, en tiempo de ejecución, a través del Pipe Binding Protocol (Protocolo de unión de tuberías). Esto significa que un pipe puede ser movido y unido a diferentes pares en distintos instantes. Esto es útil por ejemplo cuando un conjunto de pares proveen un alto nivel de tolerancia a fallas, donde un par caído es reemplazado con un nuevo par ubicado en otro lugar y éste toma el pipe existente para mantener la comunicación.
Un pipe punto a punto conecta exactamente dos pares. El pipes es un output pipe para el emisor y un input pipe para el receptor, con tráfico fluyendo en una única dirección, de emisor al receptor. Un pipe propagado (propagate pipe) conecta múltiples pares desde un output pipe a uno o mas input pipes. El resultado es que cualquier mensaje enviado al output pipe es enviado a todos los input pipes.
JXTA no define como funciona el interior de un pipe. Se pueden usar cualquier tipo y número de protocolos y algoritmos unicast, multicast y sus combinaciones. De hecho un pipe puede ser una cadena donde cada eslabón usa un protocolo de transporte completamente diferente.
Los pipes están diseñados para ser asíncronos, unidireccionales y no confiables porque esta es la base de todas las formas de transporte y conlleva al mínimo gasto. Se pueden mejorar los pipes añadiendo funciones de confiabilidad, seguridad y calidad de servicio pero dado que JXTA puede correr sobre transportes que ya las tienen implementadas es fácil que una aplicación las utilice. Por ejemplo cuando dos pares se comunican entre si y ambos soportan TCP/IP una aplicación puede crear pipes bidireccionales fácilmente.

Con estos conceptos se pueden definir y especificar los protocolos antes mencionados

• Peer Discovery Protocol
Este protocolo le permite a un par encontrar lo que otros pares publican o anuncian y puede ser usado para encontrar otro par, grupo de pares o anuncios. Este protocolo es el protocolo de descubrimiento por defecto para todos los grupos de pares, incluyendo al World Peer Group. Se puede considerar que alguien quiera desarrollar un mecanismo de descubrimiento extra que pueda o no elegir usar el mecanismo de descubrimiento por defecto. Pero al incluir el protocolo por defecto se consigue que todos los pares puedan entenderse a un nivel básico.
El descubrimiento de pares puede hacerse especificando o no un nombre de un par o de un grupo a buscar. Si no se especifica un nombre entonces se recibirán todos los anunciados.

• Peer Resolver Protocol
Este protocolo le permite a un par enviar y recibir peticiones genéricas para encontrar o buscar otros pares, grupos de pares, pipes y otra información. Comúnmente este protocolo es implementado sólo por aquellos pares que poseen acceso a bancos de datos y ofrecen posibilidades de búsquedas avanzadas.

• Peer Information Protocol
Este protocolo le permite a un par aprender sobre las capacidades y el estado de otros pares. Por ejemplo uno puede envía un mensaje de *ping* para saber si el otro está vivo. Se puede requerir las propiedades de un par, donde cada propiedad posee un nombre y un valor (un string numérico).

• Rendezvous Protocol
Este protocolo le permite a un par propagar un mensaje en el ámbito de un grupo de pares.

• Pipe Binding Protocol
Este protocolo le permite a un par unir un pipe anunciado al extremo de otro pipe, indicando así adonde van realmente los mensajes por el pipe. De alguna manera un pipe puede ser visto como una cola de mensajes abstracta identificada por un nombre, que soporta un número de operaciones abstractas tales como crear, abrir, cerrar, borrar, enviar y recibir. La unión ocurre durante la operación de abrir (open) y la separación en la operación cerrar (close).

• Endpoint Routing Protocol
Este protocolo le permite a un par preguntarle a un par ruteador por las rutas disponibles para enviar un mensaje a un par receptor. Muchas veces dos pares que se quieren comunicar no pueden ser conectados directamente. Por ejemplo dos pares que no están usando el mismo protocolo de transporte o pares separados por firewalls o NAT. Los pares ruteadores responden a los requerimientos con información de las rutas disponibles, que consisten en una lista de gateways a lo largo de la ruta. Cualquier par puede elegir convertirse en un par ruteador implementado el Endpoint Routing Protocol.

El proyecto JXTA está creando sistemas P2P identificando un pequeño conjunto de funciones básicas para dar soporte a las aplicaciones P2P y brindándolas en forma de bloques constructivos para funciones de más alto nivel (figura 3). En el núcleo o core, debe existir la capacidad de crear y borrar grupos, publicarlos a potenciales miembros para que los puedan encontrar y unírseles o abandonarlos. En la siguiente capa, las capacidades del núcleo pueden ser usadas para crear un conjunto de servicios para pares (peer services) que incluyen indexado, búsqueda y compartir archivos. Las aplicaciones pueden ser construidas usando éstos últimos. Además se crearon un peer command y un shell para crear una ventana dentro de la red basada en tecnología JXTA.

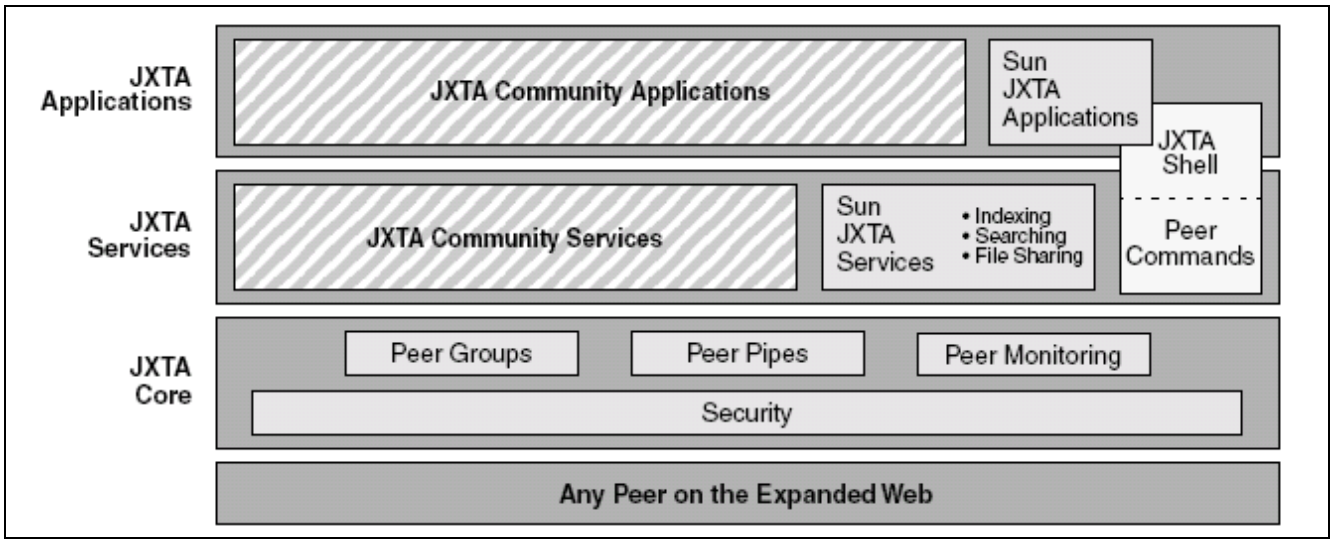


Figura 3

Núcleo JXTA

El núcleo JXTA o *core* provee el soporte fundamental para los servicios peer to peer y las aplicaciones. Se proveen los mecanismos de *peer groups, peer pipes* y *peers monitoring* en un ambiente de ejecución seguro multiplataforma.
- *Peer groups* establece un grupo de pares y los identifica nombrándolos dentro de un grupo de pares, con mecanismos para crear políticas para la creación y borrado, membresía, publicidad y descubrimiento de otros grupos de pares y nodos, comunicación, seguridad e intercambio de contenido.

- *Peer pipes* provee canales de comunicación entre los pares. Los mensajes enviados en *peer pipes* son estructurados con XML y soportan intercambio de datos, contenido y código en una forma independiente del protocolo permitiendo un amplio rango de opciones de seguridad, integridad y privacidad.

- *Peers monitoring* permite el control del comportamiento y las actividades de los pares en un grupo (*peer group*) y puede ser utilizado para implementar funciones de administración de pares incluyendo control de acceso, seteos de prioridad, medidas de tráfico y balance de ancho de banda.

La capa núcleo (*Core*) soporta opciones como usuarios anónimos vs. registrados, encriptados vs. contenido de texto libre, sin imponer políticas específicas sobre los desarrolladores. Las opciones de políticas se hacen, o cuando son necesarias, se implementan en las capas de servicio y aplicación. Por ejemplo los servicios de administración, tales como aceptar o denegar la membresía de un par a un grupo pueden ser implementados usando funciones en la capa núcleo.

Servicios JXTA

Un servicio es un conjunto de funciones que brinda un proveedor. Un par puede ofrecer un servicio por si mismo o en cooperación con otros pares. Un par proveedor de servicios publicita un anuncio del servicio, otros pares pueden descubrir éste servicio y hacer uso de él. Cada servicio tiene una ID única y un nombre que consiste en un string nombre y una serie de claves descriptivas que identifican unívocamente al servicio.

Así como las múltiples librerías en sistemas operativos UNIX / LINUX soportan funciones de mas alto nivel que el kernel, los servicios JXTA expanden las propiedades del núcleo y facilitan el desarrollo de aplicaciones. En esta capa se proveen mecanismos de búsqueda, intercambio, indexado y obtención de código y contenido para habilitar puentes entre aplicaciones y traducido de archivos.
Las capacidades de búsqueda incluyen búsquedas distribuidas, en paralelo entre los grupos que son llevadas a cabo comparando una representación XML de una petición a ser procesada con representaciones de las respuestas provistas por cada par. Estas funciones pueden ser usadas para búsquedas simples, como buscar un par, o para búsquedas complejas de contenido generado dinámicamente que es inalcanzable por motores de búsqueda convencionales. Las búsquedas P2P pueden hacerse a través de una Intranet de una compañía, ubicando rápidamente la información relevante dentro de un ambiente seguro. Si una compañía ejerce ajustados controles sobre la membresía del grupo e implementa una comunicación encriptada entre los pares puede extender sus capacidades a una Extranet que incluyen compañías con las que negocia, consultores y proveedores como sus pares. El mismo mecanismo que facilita las búsquedas a través del grupo puede ser usado como puente para incorporar resultados de la búsqueda en internet e incluir datos fuera de los propios de los pares.

La capa de servicios de pares puede ser usada para dar soporte a otras funciones particulares de aplicaciones específicas, por ejemplo un sistema de mensajería seguro entre pares puede ser construido para permitir autorías anónimas y un almacenamiento de mensajes persistente. La capa de servicio provee los mecanismos para crear tales herramientas seguras. Las políticas de una herramienta específica son determinadas por los mismos desarrolladores de la aplicación.
A veces un servicio está bien definido y ampliamente disponible de forma tal que un par puede simplemente hacer uso de él. Otras veces se puede requerir de código específico para realmente acceder al servicio. Por ejemplo, la forma de interactuar con el proveedor del servicio puede estar codificada en una pieza de software. En este caso es más conveniente si el par puede ubicar una implementación que se adapte a su ambiente de ejecución específico. Se ve así que si hay múltiples implementaciones disponibles del mismo servicio entonces los pares alojados en sistemas con entorno de ejecución Java pueden usar la implementación programada en lenguaje java y los pares nativos pueden usar la implementación de código nativo.
La implementación de servicios puede ser preinstalada en un par nodo o cargada desde la red. El proceso de encontrar, bajar e instalar un servicio desde la red es similar a realizar una búsqueda de una determinada página web en internet, recuperando el contenido de la página y luego instalando el plug-in necesario para trabajar con la página. Una vez que el servicio está instalado y activado se pueden usar *pipes* para comunicarse con él.

Nos referimos a un servicio de par (*peer service*) al que se ejecuta en un único par como un servicio y nos referiremos como servicio de grupo de pares (*peer group service*) a un servicio que está compuesto por un conjunto de instancias cooperativas del servicio que se ejecutan en múltiples pares. Un servicio de grupo de pares (*peer group service*) puede emplear algoritmos de tolerancia a fallas para darle al servicio un alto grado de disponibilidad, mucho mas de lo que un servicio de par (*peer service*) puede brindar.
Un grupo formado usando plataforma JXTA requiere de un conjunto mínimo de servicios para dar soporte a la comunicación del grupo. La primera implementación JXTA tiene incorporada un conjunto de servicios de grupo de pares predeterminados, tales como descubrimiento de pares y una serie de servicios configurables como ruteo. Esto no significa que cualquier par debe tener estos servicios. Por ejemplo se puede pensar en un par teléfono celular que esté preconfigurado con suficiente información como para contactar a un determinado servidor provisto por la compañía de telecomunicaciones. Esto es suficiente para conectar al par teléfono celular sin necesitar que este tenga servicios adicionales cargados.

Shell JXTA

Entre los límites de los servicios y las aplicaciones está el *JXTA shell*, que le permite tanto a desarrolladores como a los usuarios experimentar con las capacidades de la tecnología JXTA o con aplicaciones prototipo y controlar el ambiente o entorno de los pares. El JXTA Shell brinda acceso interactivo a la plataforma JXTA por vía de una simple interface de tipo línea de comandos, como en UNIX donde el acceso a la plataforma se hace a través de un shell interprete de líneas de comando que permite a los usuarios manipular y administrar los archivos y procesos. Se puede aprender mucho sobre el funcionamiento interno de UNIX usando el UNIX shell y también llevar a cabo muchas tareas escribiendo shell scripts. Lo mismo se dá para el JXTA Shell. Sin embargo la mayoría de comandos del shell UNIX están diseñados para ejecutarse en la máquina local y los del shell JXTA en un ambiente de red. Lo que sucede detrás de esto es que un comando de un usuario genera una secuencia de intercambio de mensajes entre un grupo de pares, una serie de procesamiento en los nodos remotos y luego retorna la respuesta al usuario.

Usando el shell JXTA se puede jugar con el núcleo de la plataforma JXTA creando bloques tales como pares, grupos de pares, pipes y codats. Codats son unidades de almacenaje que pueden contener tanto código como datos. Un usuario puede publicar, buscar y ejecutar codats, descubrir pares o grupos de pares, crear pipes para conectar dos pares y enviar y recibir mensajes.
El intérprete en el shell JXTA trabaja en un simple lazo: Acepta un comando, lo interpreta, lo ejecuta y queda a la espera del próximo comando. El shell muestra un prompt tipo "*JXTA>*" para informar a los usuarios que está listo para aceptar un nuevo comando. Así como hay múltiples versiones del shell de UNIX (el original Bourne Shell, el Shell C, BASH, el Korn Shell, TCSH) no sería raro si se diera la misma situación para el shell JXTA.
El shell de los pares contiene funciones incorporadas que facilitan el acceso a funciones del núcleo a través de una interfase de línea de comandos y comandos externos que pueden ser ensamblados usando pipes para cumplir tareas más complejas, igual que en los sistemas operativos UNIX pero extendido a un entorno P2P.

Por ejemplo un comando del shell "peers" lista los pares accesibles del grupo. Otras funciones incluyen el acceso por línea de comando a descubrir pares, unirse y abandonar un grupo y enviar mensajes entre pares.
Deliberadamente se eligieron los nombres de los comandos del shell JXTA basados en los del shell UNIX, tales como "ls", "cat" con el objetivo que sea más amigable a los usuarios. En esta implementación basada en tecnología Java la mayoría de los comandos del shell son sólo programas escritos en lenguaje java y son dinámicamente cargados e inicializados por el framework shell cuando los correspondientes comandos son tipeados. Por ello agregar nuevos comandos de shell consiste sólo en escribir un programa en lenguaje java.
El operador "pipe" ("|") para encadenar comandos y *stdin*, *stdout* y *stderr* son fundamentales el la programación de shell UNIX. El shell JXTA provee una capacidad de "pipe" para redireccionar la salida de un comando a la entrada de otro. Todos los comandos shell JXTA poseen pipes de entrada, salida y error que un usuario puede conectar desconectar y reconectar a otros comandos del shell.

Operación dinámica de pipes. En los sistemas operativos UNIX el comando de shell C 'cat myfile | grep "jxta"' se tiene que completar o se puede terminar con Ctrl-C. El usuario no puede modificar la redirección del pipe una vez que el comando se lanzó. En JXTA dado que los pipes son más permanentes, el usuario puede dinámicamente desconectar y conectar pipes.

Aplicaciones JXTA

Las aplicaciones JXTA son construidas usando tanto los servicios de pares como la capa núcleo. La filosofía del proyecto es dar soporte a los niveles fundamentales ampliamente y confiar en la comunidad de desarrolladores P2P para proveer servicios y aplicaciones adicionales.
Las aplicaciones de los pares le permiten a las capas núcleo y servicios incluir subastas P2P que vinculan vendedores y compradores directamente. Los compradores son capaces de programar su estrategia de oferta escribiendo un simple script. Las aplicaciones que comparten recursos como SETI@home pueden ser construidas en una forma más rápida y fácil con soporte para grupos de pares heterogéneos distribuidos por todo el mundo desde el primer día de lanzamiento. Se facilita la comunicación y colaboración dentro de los grupos de pares mediante servicios de mensajería instantánea y calendario que son seguros e independientes de la necesidad de alojamiento en un proveedor de servicio.

Consideraciones de Seguridad

Los requerimientos de seguridad de un sistema P2P son muy similares a los de cualquier otro sistema de computación. Los tres requerimientos predominantes son confidenciabilidad, integridad y disponibilidad. Esto se traduce en requerimientos de funcionalidad específicos que incluyen autentificación, control de acceso, auditoría, encriptación, comunicación segura y no repudio. Tales requerimientos son comúnmente satisfechos con un modelo o arquitectura de seguridad apropiado, el cual está normalmente expresado en términos de sujetos, objetos y acciones que los sujetos pueden hacer sobre los objetos. Por ejemplo el sistema operativo UNIX tiene un modelo de seguridad simple. Los usuarios son sujetos, los archivos son objetos. El hecho de que un sujeto pueda leer, escribir o ejecutar un objeto depende de los permisos que tenga ese sujeto sobre ese objeto que tiene modos específicos de permisos. Sin embargo a bajos niveles del sistema el modelo de seguridad se expresa con enteros, en términos de uid (User ID), gid (Group ID) y modo de permisos. Aquí los mecanismos de sistema de bajo nivel no necesita entender el concepto de usuario y no necesita involucrarse en como un usuario es autentificado y que uid y gid se le asignan.
Dado que JXTA está definido sobre conceptos de pares y grupos de pares, se puede vislumbrar una arquitectura de seguridad en la cual las IDs de los pares y las IDs de los grupos son tratadas como sujetos de bajo nivel (como uid y gid), codats son tratados como objetos (como los archivos) y las acciones son las operaciones entre pares, grupos de pares y codats. Sin embargo la realidad es más complicada. Por ejemplo dado que los codats pueden tener formas y propiedades arbitrarias no está claro que serie de acciones deben ser definidas para ellos. Es mas probable que los codats contengan definiciones de como deben ser accedidos. Tales codats son análogos a objetos los cuales definen métodos de acceso a ellos mismos que otros pueden invocar.

Al considerar una arquitectura de seguridad es importante notar que los requerimientos de seguridad para JXTA serán afectados en un futuro por algunas características particulares.

• La tecnología JXTA es una plataforma concentrada en mecanismos y no políticas. Por ejemplo se usan UUIDs (Universally Unique ID), pero ellas por si misma no tienen significado externo. Si no tienen servicios adicionales de nombre (namimg) y enlace (binding), las UUIDs son sólo números que no se corresponden con nada. Por ello a diferencia de otros sistemas, JXTA no define un modelo de seguridad de alto nivel, tal como flujo de información, Bell-LaPadula o Muralla China. Cuando los UUIDs son ligados o vinculados a nombres o entidades externas para formar security principals (una entidad que puede ser positivamente identificada y verificada a través de una técnica de autentificación) se puede asegurar la autenticidad del vínculo colocando en el campo de datos atributos de seguridad, por ejemplo firmas digitales que testifiquen la confiabilidad del vínculo. Una vez que el vínculo está establecido se puede llevar a cabo la autenticación del principal, control de acceso basado en el principal y en las políticas de seguridad prevalecientes y otras funciones tales como revisión del uso de recursos.

• La tecnología JXTA es neutral con respecto a los esquemas criptográficos o algoritmos de seguridad. No especifica una solución de seguridad específica, provee un framework donde se pueden conectar diferentes soluciones de seguridad. Por ejemplo, todo mensaje tiene un campo credencial designado que puede ser usado para colocar información referente a la seguridad. Sin embargo la forma exacta de implementar tal información está mas allá del alcance de la especificación, eso se deja a los servicios y aplicaciones.

• La tecnología JXTA puede satisfacer los requerimientos de seguridad a diferentes niveles del sistema. No se fuerza a los desarrolladores a una implementación en particular para permitir la máxima flexibilidad y evitar redundancia.

Como ejemplo se examinarán dos requerimientos: seguridad en la comunicación y anonimato.

Seguridad en la comunicación. Ya se ha definido que los pares se comunican a través de pipes. Supongamos que se requiere confidenciabilidad e integridad en el canal de comunicación. Una solución es utilizar VPNs (Redes Privadas Virtuales) para conducir todo el tráfico de red. Otra solución es crear una versión segura del pipe, similar a un túnel protegido, de forma tal que cada mensaje que pasa por el pipe es automáticamente asegurado. Una tercera solución es utilizar mecanismos comunes de comunicación pero dejar que el desarrollador o programador proteja específicamente los datos que considera útiles con técnicas de encriptación y firmas digitales.

Anonimato. Anonimato no significa ausencia de identidad. Ciertamente en algunas ocasiones no se puede evitar un cierto grado de identificación. Por ejemplo el número de un teléfono celular o de una tarjeta SIM no puede permanecer anónimo ya la compañía de teléfonos lo necesita para autorizar y efectuar las llamadas. Como otro ejemplo el número IP de una PC no puede ocultarse de su gateway o router más cercano si la PC quiere enviar y recibir datos.
El anonimato puede ser implementado sobre la identidad pero no al revés y en muchas ocasiones hay múltiples maneras de asegurar el anonimato. En los ejemplos anteriores se hace mucho más difícil relacionar una tarjeta SIM pre paga, que se puede revender y pagar en efectivo, con el usuario actual del teléfono. Asimismo un router o gateway puede ayudar a ocultar el verdadero IP de las PCs del mundo exterior usando relay de mensajes o NAT.
Supongamos que un servicio de nombres basado en tecnología JXTA puede relacionar a un par con un ser humano. El anonimato del usuario se asegura a través del servicio de nombre, o el servicio de autentificación, o un servicio de proxy o una combinación de ellos. JXTA es independiente de la solución elegida para cada aplicación en particular.

4. Implementación

La implementación inicial de referencia de los protocolos JXTA 1.0 fue publicada en abril del 2001. A continuación se describirá la implementación del Proyecto JXTA 2.0. Se tratará de mantener la mayor cantidad de términos posibles en su idioma original ya que en este punto se trata la implementación en si misma. Sólo se traducirán los términos al español cuando no den lugar a confusión, por ejemplo peers se traducirá como pares (cosa que se viene haciendo durante todo el texto).

4.1. Introducción

La referencia de implementación del proyecto JXTA 2.0 introduce una serie de nuevas características para mejorar la performance y escalabilidad de la red JXTA. La versión 2.0 hace una fuerte diferenciación en la forma en que los súper-pares tales como relay y rendezvous se comportan e interactúan con los pares en los extremos de la red. Esta versión JXTA 2.0 introduce el concepto de rendezvous peer view (RPV) para propagar pedidos y shared resource distributed index (SRDI o índice distribuido de recursos compartidos) para indexar anuncios en el rendezvous peer view para operaciones de búsquedas mas eficientes de dichos anuncios.
También se introduce el concepto de pluggable walkers para propagar pedidos dentro de la red RVP. Esta nueva implementación JXTA 2.0 agrega una mejor administración de recursos (threads y message queues) e

implementa límites de uso para los recursos con el objetivo de ser mas justo asignando recursos entre servicios. Crea thread pools y límites de threads para controlar el uso de threads para los servicios. Cuando se envían mensajes optimiza la reducción de un excesivo almacenado y copiado de los mismos. El endpoint service minimiza el número de threads de conexiones activas requeridos para conectar un gran número de pares a un único par. Los pares establecen la conexión, obtiene sus mensajes direccionados y luego permanecen inactivos hasta que otro mensaje esté disponible o se cierre la conexión. El JXTA 2.0 agrega soporte para relays TCP/IP para permitir NAT transversales eficientemente. Finalmente provee respuesta a fallas dinámica de los súper-pares rendezvous y relay para recuperarse de las posibles fallas de dichos super-pares y roam para mejorar la conectividad y funcionalidad.

4.2. Red Virtual JXTA

Los protocolos del proyecto JXTA crean una red virtual sobre la ya existente infraestructura de la red física, que le permite a los pares intercambiar mensajes con otros pares independientemente de la ubicación dentro de la red (detrás de un firewall, NAT). Los mensajes son ruteados en forma transparente, si es necesario a través de firewalls o NATs y usando diferentes protocolos de transporte/transferencia (TCP/IP, HTTP) hasta alcanzar al par receptor (como se ve en la figura 4 que es análoga a la figura 1). La red JXTA le permite a los pares comunicarse sin necesidad de manejar o entender topologías de redes complejas y cambiantes, permitiendo a los pares móviles desplazarse en forma transparente desde una ubicación a otra. La red virtual del proyecto JXTA estandariza la manera en que los pares se descubren unos a otros, se auto organizan en grupos, descubren los recursos de los pares y se comunican entre si.

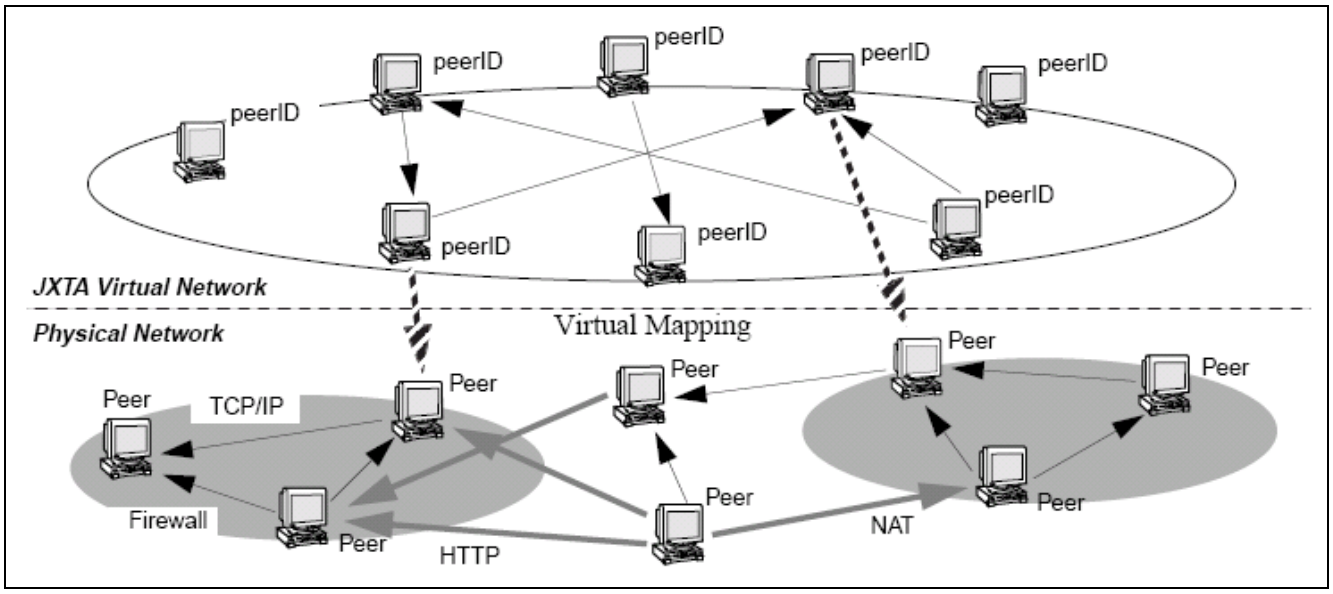


Figura 4

La implementación del Proyecto JXTA 2.0 se construye sobre las cinco abstracciones de redes virtuales introducidas en la versión JXTA 1.0.
Primero un modelo de direccionamiento lógico (logical peer addressing) que une completamente la red JXTA. Segundo, grupos de pares (peer groups) que permite a los pares auto organizarse dinámicamente en dominios virtuales protegidos. Tercero, anuncios para publicar los recursos (pares, grupos de pares, endpoint, servicios, contenido). Cuarto, un mecanismo de enlace universal (binding), llamado resolver, para llevar a cabo todos los enlaces o vínculos requeridos en un sistema distribuido. Y quinto, pipes como un canal de comunicación virtual que permite a las aplicaciones comunicarse entre si. En las siguientes secciones se abordarán estas abstracciones y mejoras hechas a JXTA 2.0.

4.2.1. JXTA IDs

El modelo de direccionamiento del proyecto JXTA es del tipo lógico y es uniforme e independiente de la ubicación. Todo recurso de red (par, pipe, dato, grupo de pares, etc.) tiene asignado un único ID JXTA. Estos IDs JXTA son objetos abstractos y permiten representaciones de ID múltiples (IPv6, MAC) para poder coexistir dentro de la misma red JXTA. La implementación de referencia está usando UUIDs aleatorias de 128 bits permitiendo a cada par auto generar su propio ID. Un par en la red JXTA está unívocamente identificado por su ID de par (peer ID) permitiendo que el par sea direccionado independientemente de su dirección física. Por ejemplo una laptop que bootea y usa DHCP (Dynamic Host Configuration Protocol) Protocolo de configuración dinámica de servidores, tendrá asignado diferentes IP en diferentes momentos según vuelva a bootear pero siempre poseerá el mismo peer ID. De igual manera un par que soporta múltiples interfases de red (Ethernet, Wi-Fi, etc) será direccionado como un único par independientemente de la interfaz usada. La abstracción de ID de par (peer ID) le permite a un par encapsular, no solo transportes físicos, sino también protocolos de transporte “lógicos” tales como HTTP o TLS. El modelo de direccionamiento lógico de JXTA introduce una

referencia fundamental, separando la identificación de un recurso y su ubicación, permitiendo usar una gran variedad de mapeos virtuales para determinar la ubicación física del recurso.
Con cada par se asocia un endpoint, encapsulando todas las direcciones de red físicas para un par. Los endpoints se parecen a una tarjeta de presentación usada para los negocios ya que tienen una lista de las posibles formas de contactar a una persona (teléfono, fax, dirección de mail, etc.). Cuando se recibe el anuncio del endpoint de un par, otro par puede seleccionar la forma mas eficiente de comunicarse con el primero dada la lista de direcciones del endpoint que estén disponibles (usando TCP/IP directamente cuando es posible o HTTP sobre TCP/IP si uno debe atravesar un firewall).

4.2.2. Anuncios

Todos los recursos de red en el Proyecto JXTA tales como pares, grupos de pares, pipes y servicios son representados por anuncios (advertisements). Los anuncios son estructuras de metadata escritas en un lenguaje neutral, son descriptores de recursos representados como documentos XML.
El proyecto JXTA estandariza los anuncios para los siguientes recursos del núcleo JXTA: Pares, grupos de pares, pipes, servicios, medición, ruteo, contenido, rendezvous, endpoint y transporte.
Los desarrolladores pueden crear subcategorías de estos anuncios para crear sus propios subtipos con el fin de agregar una cantidad ilimitada de información metadata adicional, más rica, a cada descripción de recurso. Por ejemplo un anuncio de servicio Web contendrá el documento WSDL (Web Services Description Language) asociado con el servicio. Los anuncios pueden ser usados para describir virtualmente cualquier cosa: código fuente, scripts, binarios, clases, código compilado JIT (Just In Time ó Producción Justo a Tiempo), objetos Java, EJB (Enterprise JavaBeans), contenedores J2EE (Java 2 Platform, Enterprise Edition).

A continuación se muestra un ejemplo (figura 5) de Anuncio de grupo de pares PeerGroup Advertisement (jxta:PGA) que describe a un grupo de pares mediante un único ID de grupo (peergroup ID, GID), un módulo de especificación ID (module specification ID, MSID) que apunta a un anuncio que describe todos los servicios disponibles en este grupo, un nombre de grupo (peergroup name, Name) y una descripción del grupo (Desc).

```
<?xml version="1.0"?>
<!DOCTYPE jxta:PGA>
<jxta:PGA xmlns:jxta="http://jxta.org">
<GID> urn:jxta:jxta-NetGroup</GID>
<MSID>urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE000000010206</MSID>
<Name>NetPeerGroup</Name>
<Desc>NetPeerGroup by default</Desc>
</jxta:PGA>
```

Figura 5

El cache del par publica e intercambia anuncios para descubrir y encontrar recursos de red disponibles. Los pares descubren recursos buscando sus anuncios asociados. Todos los anuncios son publicados con un tiempo de vida (lifetime) que especifica la duración de ese anuncio en la red. Un anuncio puede ser re-publicado en cualquier momento para extender su tiempo de vida antes de que expire. Los pares intercambian anuncios pero mantienen sus fechas de expiración durante dichos intercambios. Mediante el uso de los tiempos de vida se pueden purgar anuncios que expiraron sin necesidad de una administración centralizada.

4.3. Rendezvous Super-Peers

La red JXTA usa un mecanismo de enlace universal llamado Resolver para llevar a cabo las operaciones de resolución encontradas en los tradicionales sistemas distribuidos, tales como resolver el nombre de un par en a una dirección IP (DNS), enlazar un socket a un puerto, ubicar un servicio a través del Servicio de Directorio (LDAP) o buscar contenido en un sistema de archivos distribuido (NFS). En JXTA todas las operaciones de resolución están unificadas bajo el simple descubrimiento de uno o más anuncios. Los protocolos del proyecto JXTA no especifican como se lleva a cabo la búsqueda de anuncios sino que provee un framework genérico resolvedor de protocolo (resolver) con una política por defecto que puede ser sobre escrita. Los desarrolladores pueden modificar a su medida sus implementaciones de resolver para usar soluciones descentralizadas, centralizadas o híbridas según lo requiera su aplicación específica. La infraestructura central del resolver provee la capacidad de enviar y propagar requerimientos, y recibir respuestas. El resolver lleva a cabo la autenticación y verificación de las credenciales y descarta mensajes inválidos.

La red JXTA provee una política resolvedora por defecto basada en Rendezvous super-peers. Rendezvous son pares que tienen agregado al cache índices de anuncios (Por ejemplo punteros a pares en los extremos de la red que almacenan los correspondientes anuncios). Rendezvous corresponden conceptualmente a ubicaciones conocidas usadas para indexar y localizar anuncios. La política rendezvous por defecto provee la mínima infraestructura para desarrollar eficientemente políticas de anuncios de búsquedas de alto nivel.

Por ejemplo vía rendezvous un par puede encontrar suficientes pares como para desarrollar una política de búsqueda más avanzada. Se pretende que los servicios de búsqueda de alto nivel brinden mecanismos de búsqueda más eficientes porque tienen conocimiento más amplio de la topología de distribución del contenido. La infraestructura por defecto de los pares rendezvous provee un mecanismo de bajo nivel para descubrir anuncios pero también provee mecanismos para permitir a los servicios de descubrimiento de alto nivel participar en el proceso de búsqueda.
Los pares en los extremos de la red mantienen una relación especial con sus pares rendezvous. Cualquier par se puede convertir en un par rendezvous asumiendo que posea las credenciales correctas. Los grupos de pares seguros pueden restringir que par en el extremo de la red puede actuar como rendezvous. Un cambio fundamental hecho en JXTA 2.0 es que rendezvous no captura los anuncios de pares en los extremos de la red. Rendezvous sólo mantiene un índice de anuncios publicados por sus pares extremos. El hecho de no capturar anuncios hace a la arquitectura rendezvous más escalable y reduce el problema de capturar anuncios que ya expiraron.
El servicio índice de recursos distribuidos compartidos (Shared-Resource Distributed Index, SRDI) es usado por los pares en los extremos de la red para indexar sus anuncios en rendezvous. Estos pares extremos utilizan SRDI para cargar los índices de anuncios en sus rendezvous cuando publican nuevos anuncios.
Los índices se cargan sincrónicamente cuando un nuevo anuncio es publicado o asincrónicamente por el demonio SRDI que se ejecuta a intervalos de tiempo fijo.

Por ejemplo en la figura 6 ambos pares el Par A y el Par B están cargando índices de sus anuncios en sus respectivos rendezvous Rdv1 y Rdv2.

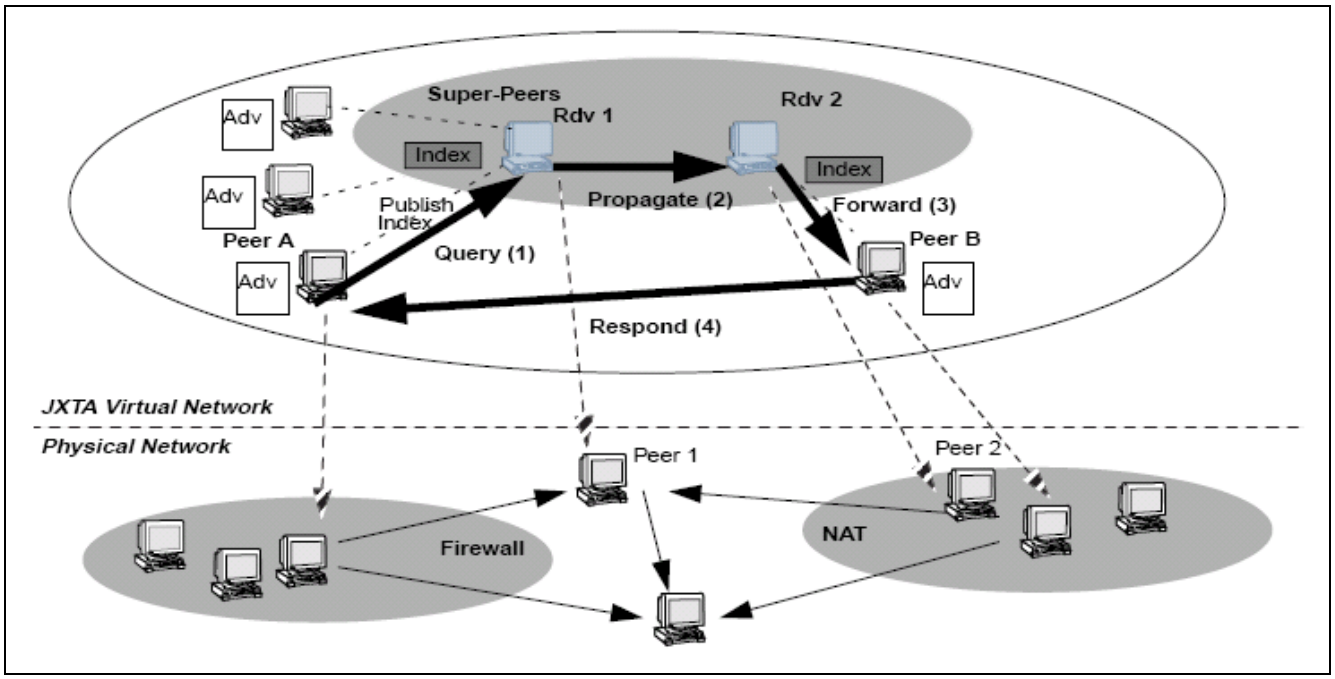


Figura 6: Súper-pares Rendezvous

Cuando el Par A peticiona un anuncio alojado en el Par B, el pedido se envía al rendezvous Rdv1 (1), luego Rdv1 se fija si tiene indexado ese anuncio. Si no lo tiene propaga la petición (2) al próximo rendezvous Rdv2. Cuando la petición alcanza a Rdv2 éste encuentra indexado el anuncio y direcciona la petición al Par B (3). Esto se hace para asegurar que lo que se envía al Par A sea la última copia del anuncio del Par B. Cuando el Par B recibe la petición envía el anuncio al Par A (4). Es importante destacar que cualquier par independientemente de su ubicación física puede actuar como rendezvous. En la figura 6 el Par 2 está ubicado físicamente detrás de una red que utiliza NAT y actúa como rendezvous.
Otra mejora significativa del JXTA 2.0 es que las peticiones de anuncios no son más propagadas a los pares extremos. Sólo los pares rendezvous están involucrados en la propagación de peticiones de anuncios. Una petición sólo se envía a un par extremo cuando el índice buscado ha sido encontrado. Esto reduce significativamente el tráfico al realizar búsquedas de anuncios. En la siguiente sección se verá como las peticiones se propagan entre los rendezvous super-peers.

4.3.1. Una red Rendezvous Consistente - Flexible

Los súper pares rendezvous se organizan a ellos mismos en redes no rígidamente acopladas, flexibles. Esto se hace debido al relativamente alto índice de abandono predicho por las redes Ad Hoc JXTA. En una red fluctuante es difícil mantener una vista consistente de todos los pares en un grupo, sin asumir que el grupo será pequeño o que se tiene una estructura de membresía centralizada y altamente acoplada que mantiene la lista de los pares. Mantener una vista consistente se vuelve aun mas difícil si el tamaño del grupo y el número de

pares se incrementa a un número extremadamente grande (un millón de pares). Quizás rendezvous en redes P2P de empresas muestre una mayor estabilidad y en redes de usuarios (PDA, celulares) aparente ser menos estable y muestre una tasa de pares desconectados mas alta. JXTA difiere de Distributed Hash Table (DHT) tales como Brocade, Chord o CAN que asumen una infraestructura de pares relativamente estable, donde se puede mantener una visión distribuida consistente con un mínimo gasto de recursos. Al desplegar una red P2P no todas las entidades estarán de acuerdo en cubrir el costo extra de utilizar una infraestructura de pares dedicados (súper pares) para dar soporte a su red. Este factor económico y la necesidad de permitir redes P2P auto suficientes hizo que JXTA tome un enfoque totalmente Ad Hoc, no estructurado que permita a cualquier par convertirse en un súper par. Es importante destacar que esta política tan libre, flexible y desestructurada es sólo la política por defecto. Los creadores de los grupos de pares pueden modificar la política por defecto y definir sus propias políticas.
En un ambiente altamente fluctuante e impredecible el costo de mantenimiento de un índice distribuido consistente es tan alto que contrarresta las ventajas de usar dicho índice. Se perderá mas tiempo actualizando índices que llevando a cabo las búsquedas. Se pueden separar los costos de Distributed Hash Table (DHT) en mantenimiento de índice y búsqueda indexada. DHT provee el mecanismo más eficiente de búsqueda indexada denominado log(N) con N igual al número de pares. Sin embargo hay un costo de mantenimiento que normalmente crece exponencialmente al incrementarse el abandono de pares. Por otro lado si se piensa no usar DHT se necesitarán realizar búsquedas sumamente exhaustivas (y por lo tanto caras) implicando aun más tráfico en la red.
El Proyecto JXTA2.0 propone una solución híbrida que combina el uso de DHT flexible-consistente con un walker rendezvous de rango limitado, para recolectar y eliminar los índices ya inservibles. No se requiere que los pares mantengan un índice de hash distribuido, por ello el término DHT flexible-consistente. Si la tasa de desconexión o de abandono de los pares aparenta ser muy baja entonces el RPV se mantiene estable, se sincroniza el DHT flexible-consistente y alcanza la performance de búsqueda óptima.

4.3.2. Rendezvous Peer View (RPV)

Cada rendezvous mantiene su propia vista de pares, esto es una lista ordenada de rendezvous conocidos dentro del grupo por si ID de par. No se usan mecanismos fuertes o consistentes para hacer que se mantenga la integridad del RPV a través de todos los rendezvous, éstos pueden tener RPVs temporalmente o permanentemente inconsistentes. Se usa un algoritmo especial para hacer converger los RPVs a través de todos los rendezvous. Los rendezvous periódicamente seleccionan un número aleatorio de rendezvous de su RPV local y les envían una lista aleatoria de sus rendezvous conocidos. También envían un heartbeat a sus rendezvous vecinos (posición +1 y -1 RPV). Un heartbeat es un mensaje (corto) de estado enviado una máquina determinada de la red, a intervalos regulares de tiempo con el propósito de monitorear el estado de salud. Los rendezvous actualizan y purgan los rendezvous que no responden de su RPV.
Puede darse que también recupere información de un conjunto predeterminado de rendezvous que actúan como fuente. Cada grupo de pares tiene la habilidad de definir su propio conjunto de rendezvous fuente.
Todo rendezvous puede actuar como rendezvous fuente (y recordemos que todo par puede convertirse en rendezvous). Estos rendezvous fuentes (seeding rendezvous) permiten acelerar el proceso de convergencia RPV ya que todos los rendezvous son preconfigurados con la lista de rendezvous fuente. Los rendezvous fuente son usados como el último recurso cuando un rendezvous no puede encontrar ningún otro rendezvous y para incorporar inicialmente los rendezvous a la red. Esto se hace para limitar la dependencia de los rendezvous fuente y descentralizar el manejo de RPV. Luego de la iniciación y de que un rendezvous ha aprendido sobre otros rendezvous, los rendezvous fuente son tratados como cualquier otro rendezvous. Si la red de rendezvous es relativamente estable, los RPVs convergen rápidamente a través de todos los rendezvous permitiendo una distribución de índice consistente.

4.3.3. Conexión de un par extremo con un rendezvous

En esta sección se verá como un par extremo (en el extremo de la red) se conecta con un rendezvous disponible. Los pares extremos primero chequean en su cache local si hay algún anuncio de rendezvous que apunte a un rendezvous alcanzable. Los anuncios de rendezvous persisten a través de las conexiones de pares extremos. El par extremo ordena a los rendezvous candidatos en grupos de a cinco y prueba los cinco a la vez hasta que se establece una conexión con un rendezvous. Los pares extremos sólo mantienen una conexión.
Si no se establece la conexión con un rendezvous después de un período de tiempo configurable (por ej. 30 seg.), el par extremo intentará buscar un rendezvous vía una solicitud sobre sus transportes disponibles (IP multicast), o sobre un pipe de propagación específico del grupo (si el par se ha unido a un grupo). Los rendezvous existentes están escuchando y responderán a esa solicitud. Si después de un cierto período (30 seg.) no se encontró ningún rendezvous, el par extremo consultará a uno de los rendezvous fuente. Finalmente si ninguno de los rendezvous fuente es alcanzado después de un período de tiempo configurable (por ej. 5 min.), el par extremo tratará de volverse un rendezvous (si posee las credenciales adecuadas). El par extremo puede volver al modo de par extremo en cuanto se encuentre otro rendezvous. El par continuará buscando rendezvous en segundo plano.
Si un conjunto de rendezvous no están conectados se puede particionar el RPV, sin embargo dichas particiones de RPV comenzarán a fundirse a medida que cada partición alcance uno de los rendezvous fuente o a un rendezvous común existente en ambas particiones. El mecanismo flexible-consistente RPV es lo

suficientemente robusto para permitir la fundición de particiones incluso en el caso de que fallen los pares fuente. Las posibles inconsistencias se resuelven con el tiempo, a medida que los rendezvous continúan intercambiando información aleatoriamente. Esta política de rendezvous es una solución de compromiso que toma ventaja de algunos rendezvous fuentes y reduce la dependencia a estos pares. Después de que un par se inicializó a si mismo a través de los rendezvous fuente se espera que aprenda sobre suficiente rendezvous de forma tal que no tenga que volver mas al rendezvous fuente.

4.3.4. Propagación Rendezvous

Esta sección discute como se propagan las solicitudes dentro del RPV del rendezvous.

La figura 7.1) muestra como un nuevo anuncio publicado se indexa en el RPV del rendezvous. El par P1 publica un nuevo anuncio en su rendezvous R2 vía el servicio SRDI (shared resource distributed index). Cada anuncio es indexado por SRDI usando un número predefinido de claves, tales como nombre o ID. R2 usa la función DHT (Distributed Hash Table) (*H(adv1)*) para mapear el índice a un rendezvous en su RPV local. El RPV en R2 contiene los rendezvous de R1 a R6. Supongamos que la función DHT retorne R5, entonces R2 cargará el índice en R5. Para aumentar la probabilidad de recuperar el índice en las cercanías de R5 y direccionar la potencial desaparición de R5 se puede replicar el índice a los vecinos RPV de R5 (+1 y -1 en la lista RPV ordenada). En este ejemplo el índice se replica en R4 y R6. Es importante notar que la proximidad de índices en RPV no necesariamente implica que físicamente las redes están cerca. R5 y R4 pueden estar en hemisferios diferentes. Replicar el índice alrededor de R5 significa que se está creando una región lógica en el RPV donde ese índice puede ser ubicado.

Asumamos que el par extremo P2 está buscando anuncios (advertisements) indicado como adv1 en la figura 7.2.a). P2 hará una petición a su rendezvous R3. El servicio SRDI en R3 calculará la función (H(adv1)) usando el RPV local de R3. Si el RPV en R2 y en R3 son los mismos, la función DHT entregará el mismo rendezvous R5. R3 puede pasar esta petición a R5 que se la pasará a P1 para responderle a P2. Esto funciona siempre y cuando el RPV sea igual en R2 y en R3.

Supongamos que R5 se cae y R3 actualiza su RPV, para reflejar el hecho de que R5 ha desaparecido de su RPV, figura 7.2.b). R3 se dará cuenta de que R5 está caído ya sea por la actualización del mapeo RPV o cuando trate de enviar un mensaje a R5. En este escenario R3 tiene un nuevo RPV que contiene a los rendezvous desde R1 a R5. Pero ahora R5 apunta al rendezvous correspondiente al R6 del mapa RVP anterior. Un rendezvous faltante significa que el RVP se corre una posición. Luego de recibir la petición del resolver de P2, R3 calculará la función DHT que retornará el nuevo rendezvous R5. Dado que también habíamos publicado el índice en R6 como parte de la estrategia de replicado, se encontrará el índice en el nuevo R5. Este ejemplo demuestra como aún con un RPV inconsistente se puede usar satisfactoriamente DHT. Mientras el desplazamiento del RVP esté dentro de la distancia replicada de la DHT (+1, -1) se puede garantizar que el índice será encontrado. Para incrementar la capacidad de recuperar índice, aún con un RPV desplazado, se replica el índice en las proximidades del rendezvous objetivo inicial. Es importante destacar que aquí no se requiere un mantenimiento distribuido del RVP. La distancia de replicación puede incrementarse (+2 o +3) para mapas RVP mas grandes.

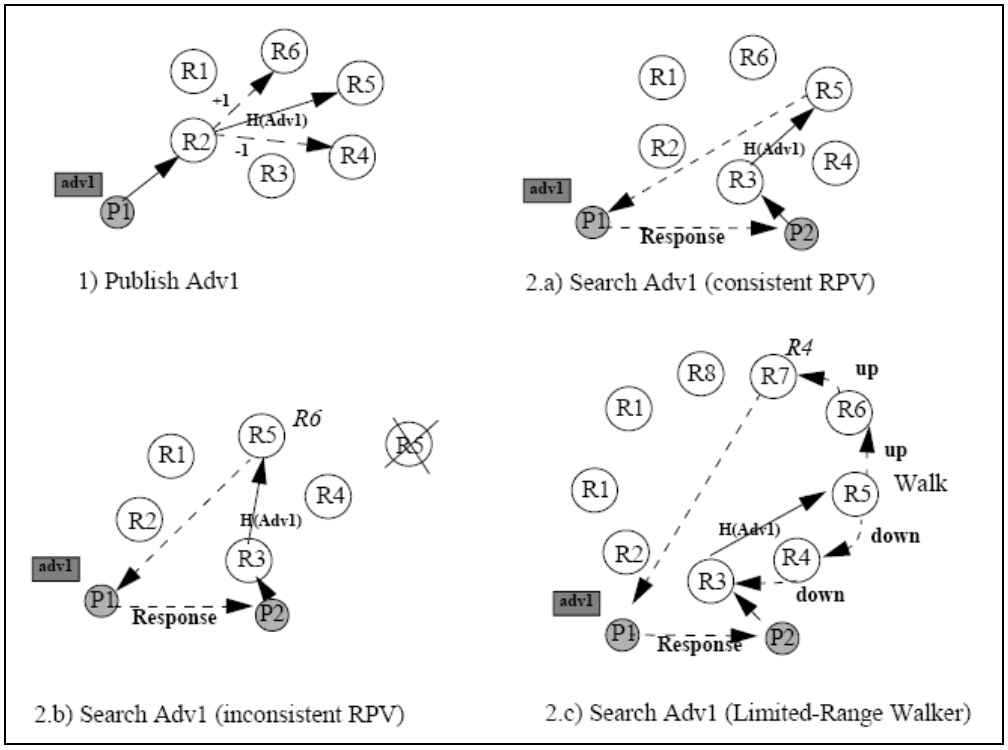


Figura 7: Rendezvous DHT

Imaginemos un escenario más caótico donde el RVP experimenta cambios importantes, figura 7.2.c). El RPV en R3 se compone de ocho rendezvous, R1 a R8. R7 corresponde al R4 anterior. Cuando R3 recibe la petición de P2 computará la función DHT que mapea el índice a R5. Dado que el RVP ha cambiado drásticamente el índice no se encuentra en R5. En este caso se utiliza un mecanismo alternativo para "caminar" por el RVP y continuar la búsqueda. Se utiliza un walker de rango limitado (limited-range walker) para recorrer el rendezvous desde su DHT objetivo inicial. El walker procederá en ambas direcciones, hacia arriba y hacia abajo, figura 7.2.c). El recorrido hacia arriba irá hasta R6 y el hacia abajo hasta R4. La intención del walker de rango limitado es buscar en las proximidades del rendezvous objetivo inicial a un rendezvous que posea el índice. Este walker aprovecha el hecho de que la probabilidad de encontrar el índice en esa región aumenta debido al esquema de replicación DHT. En este ejemplo R5 transmite la solicitud tanto a R4 como a R6. Un contador de saltos (hop counter) se utiliza para especificar la cantidad máxima de veces que la solicitud puede ser transmitida. Si R4 no tiene el índice, retransmitirá la petición al próximo par hacia abajo, que es R3. De igual manera R6 retransmitirá la petición al próximo hacia arriba, R7. Cuando se encuentra el índice en R7 la petición se transmite a P1 y el recorrido hacia arriba se detiene. La recorrida en la dirección hacia abajo continuará hasta que se alcance el contador de saltos o hasta que no haya más rendezvous en esa dirección. En lugar de continuar el recorrido hasta que la cuenta de saltos llegue a su límite, se puede enviar un requerimiento que pregunte a P2 si se debe continuar la búsqueda después de haber alcanzado una determinada parte del recorrido (continue walk request). Quizás P2 ya haya recibido la respuesta de la otra porción del camino recorrido. Entonces P2 puede decidir si continuar o detener el recorrido.

Para minimizar los efectos de crecimiento o disminución del RVP o su temporal inconsistencia se usa una función hash que intenta compensar los cambios del RVP. Dicha función consiste en dividir el espacio hash en partes iguales, tantas como la cantidad de rendezvous en el RVP. A cada rendezvous se le asigna un segmento del hash en función de su ranking en el RVP. Si el número de hash cae en el medio del espacio hash entonces el rendezvous seleccionado estará en el medio del RVP. La función de hash permite escalar el espacio hash al tamaño del RVP. Inclusive si el RVP cambia drásticamente entre el instante que se ingresó el índice y el instante donde se encuentra, hay probabilidades de que el RVP haya cambiado lo mismo hacia la izquierda así como a la derecha de un determinado rendezvous. La posición relativa de un rendezvous tiende a mantenerse a medida que el RVP evoluciona. El uso de ID de pares para ordenar el RVP ayuda a asegurar una distribución de cambio uniforme en el RVP. Por más que el RVP se agrande o se achique un rendezvous ubicado en el medio del RVP tenderá a quedarse cerca del medio. Al reducir las deformaciones del índice se ayuda a reducir el número de pasos requeridos durante un recorrido para encontrar un anuncio. Recorriendo el RVP en ambas direcciones reduce la latencia de las solicitudes e incrementa la capacidad de recuperación de la "forma original" del RVP.

Dado que el RVP está ordenado por ID, cada rendezvous es consultado solamente una vez, inclusive si los RVPs son inconsistentes. Los recorridos de rango limitado le proveen un mecanismo de recolección de basura al iniciar la DHT. Si la infraestructura de rendezvous es estable se tomará plena ventaja de DHT y raramente se usarán los recorridos de rango limitado que son más costosos.

4.4. Relay Super-Peers

La red JXTA es del tipo Ad Hoc, multi-salto y adaptativa. Las conexiones pueden ser transitorias. El ruteo de mensajes no es determinístico. Las rutas pueden ser unidireccionales y pueden cambiar rápidamente.

La red JXTA introduce el concepto de súper pares llamados relay (relay peers) para conectar pares que no tienen una conectividad física directa (NAT, Firewalls). Cualquier par puede convertirse en un relay si posee el nivel de credenciales apropiado y la capacidad en lo que respecta a ancho de banda, baja tasa de desconexión y conectividad directa. Los pares relay proveen la capacidad de enviar mensajes a pares en los extremos de la red, que son inaccesibles o que están disponibles temporalmente.

Por ejemplo en la figura 8 el par A quiere enviarle un mensaje al par B. Como el par B está usando NAT, el par A no puede acceder a la dirección del par B, entonces el par A no puede enviarle un mensaje directamente. El par B usa el par relay D para volverse accesible. Los pares relay cumplen la función de "puntos de referencia" permitiendo el ruteo de mensajes entre pares alejados e inaccesibles.

Los pares anuncian una serie de relays preferidos para ayudar a la resolución del ruteo. Siempre tratan de conectarse directamente con otros pares antes de usar un relay. Los mensajes van a través de uno o más relays a causa de los firewalls o NATs.

La resolución de rutas mediante los pares relay se hace en forma transparente y dinámica por la red virtual JXTA. Las aplicaciones direccionan a los pares mediante su ID de par, no necesitan preocuparse de la infraestructura de los súper pares relay de la red JXTA. Los pares extremos mantienen una conexión "arrendada" a un relay preferido y recuperan los mensajes desde su cola de mensajes. Los pares pueden enviar mensajes a través de cualquier relay disponible, no sólo del preferido. De tanto en tanto los pares extremos pasan de un relay a otro para optimizar su visibilidad o mejorar su calidad de conectividad a la red.

Es importante destacar que la asociación par relay / par extremo es transitoria por naturaleza. Los relays sólo mantienen el estado para sus pares extremos durante un período convenido por leasing. Los pares extremos se pueden reconectar en cualquier momento a un relay diferente. Así como los rendezvous, los relays mantienen una vista rígida-flexible entre ellos mismos para mantener una lista de relays disponibles.

Se usan relays fuente para descubrir relays disponibles. Los pares extremos almacenan los anuncios de relays para poder recordar la lista de relays disponibles después de un reinicio. Los relays fuente se usan sólo cuando no hay otro relay disponible.

En el caso de que un relay falle, los pares extremos se conectan en forma transparente a otro relay conocido.

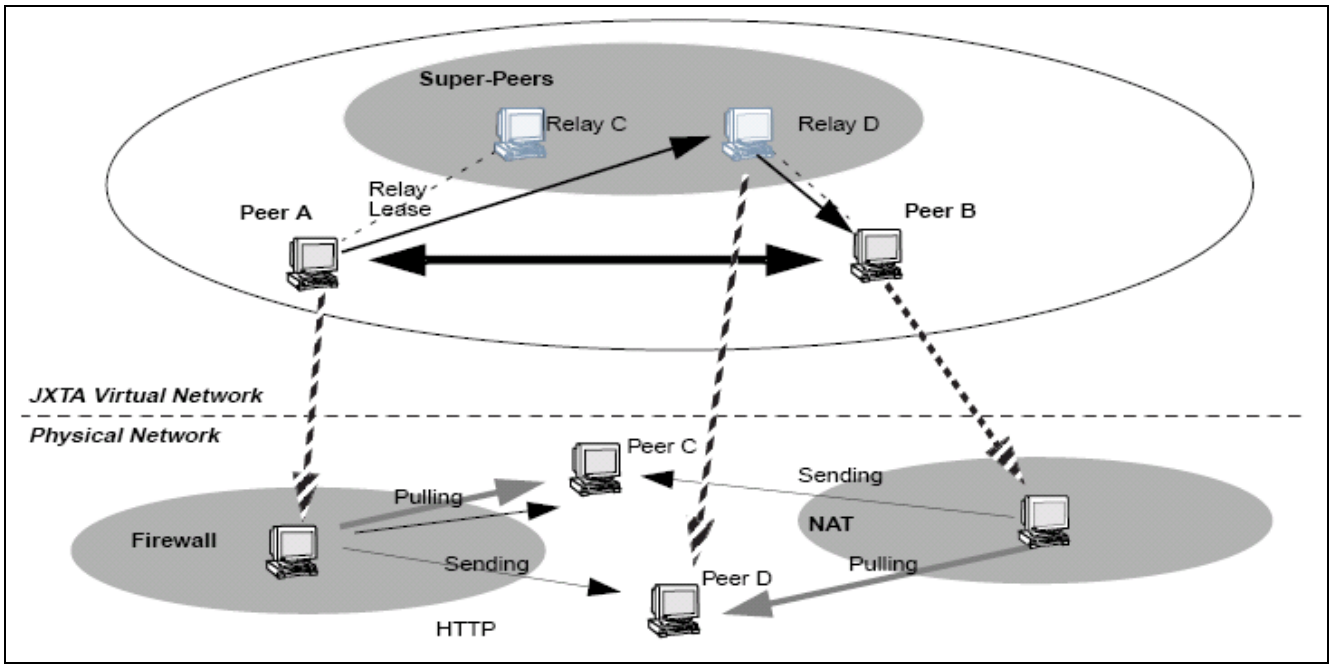


Figura 8: Relay Super-Peers

JXTA utiliza un ruteo adaptativo basado en la fuente. Las rutas son construidas inicialmente por el emisor. Esto se hace para descentralizar la administración del ruteo hacia los pares extremos y reducir la dependencia de tablas de ruteo centralizadas. Las rutas, como cualquier otro recurso, son representadas por anuncios de ruta (route advertisements). Los pares extremos capturan los anuncios de rutas y hacen solicitudes para descubrir un anuncio de ruta para un destino dado. Un anuncio de ruta describe el camino para alcanzar a un par como una secuencia ordenada de saltos. Cada salto está definido por una ID de par con un conjunto opcional de direcciones endpoint. Las direcciones endpoint se proveen como una ayuda, así el emisor no tiene que resolver IDs a direcciones endpoint. Generalmente los saltos son relay peers. En una red móvil Ad Hoc los pares extremos pueden también actuar como un salto para rutear mensajes debido a la falta de conectividad directa entre pares. En implementaciones comunes de internet se necesita sólo un relay para que dos pares extremos puedan comunicarse entre si y rutear mensajes a través de un dominio con NAT o un firewall.
Es importante señalar que la definición de rutas en un anuncio de ruta no depende de la ubicación del emisor. La descripción del ruteo en un anuncio de ruta puede ser usada por varios emisores diferentes. Los emisores seleccionarán la porción de la ruta que es importante para ellos. Por ejemplo, si el anuncio de ruta para el Par A contiene la siguiente lista de saltos <Par B, Par C, Par D> y el Par F sabe como comunicarse con el par C entonces sólo usará una porción de la ruta <Par B, Par C > para llegar al Par A. La intención es que la información de ruteo crezca y se consolide con el paso del tiempo a medida que se encuentran más saltos alternativos o atajos. Así como cualquier anuncio, los anuncios de rutas tienen un tiempo de vida seteado en 15 minutos.
Los mensajes JXTA también contienen información de ruteo como parte de su información útil. Cada vez que un mensaje pasa a través de un salto la información de ruteo se actualiza agregando ese mismo salto. Cuando un par recibe un mensaje, puede usar la información de ruteo del mismo como una ayuda para rutear un mensaje de respuesta al emisor. La información de ruteo inverso toma en cuenta enlaces unidireccionales y potenciales atajos. Por ejemplo un par detrás de un firewall puede enviar un mensaje directamente a un par afuera de un firewall pero la operación inversa no puede realizarse debido a consideraciones de NAT o de firewall. Debido a la falta de confiabilidad de los pares, el proyecto JXTA tomó la decisión de confiar totalmente en los relay peers para el ruteo. Cada mensaje es autosuficiente ya que contiene su propia información de ruteo. Cuando un mensaje contiene información de ruteo obsoleta debido a una falla del relay peer, se descubre una nueva ruta dinámicamente usando otros relay peers conocidos. La información de ruteo del mensaje es también usada para la detección de lazo.

4.4.1. Mensajes y Credenciales

Los mensajes son la unidad básica de intercambio de datos entre pares. Los pares interactúan enviando y recibiendo mensajes. El proyecto JXTA utiliza un formato binary wire para representar mensajes y permitir un transporte eficiente de virtualmente cualquier tipo de información. Se puede enviar tanto información XML como binaria. Cada transporte JXTA puede usar el formato más apropiado para transmitir datos. Un mensaje es una secuencia ordenada de contenidos, con nombre y tipo, llamados elementos. El elemento que se agregó mas recientemente aparece al final del mensaje. A medida que un mensaje se mueve hacia abajo en las capas de protocolos (aplicaciones, servicios y transportes) cada nivel puede agregar uno o mas elementos (con nombre)

al mensaje. A medida que un mensaje se mueve de vuelta hacia las capas de arriba, los protocolos extraerán aquellos elementos.
La necesidad de soportar diferentes niveles de autentificación y acceso a recursos en la red Ad Hoc JXTA conlleva a un modelo de confianza basado en el "rol", en el cual un par actuará comúnmente bajo la autoridad que le fue garantizada por otro par confiable. La relación entre pares puede cambiar rápidamente y las políticas que gobiernan el control de acceso necesitan ser flexibles para permitir o denegar el acceso. El formato de mensajes del proyecto JXTA permite agregar una serie de información metadata al mensaje, como credenciales, resúmenes, certificados, claves públicas, etc. Todo mensaje contiene una credencial. Una credencial es un token (o posta) que cuando se presenta en el cuerpo del mensaje permite identificar al remitente y puede ser usado para verificar los derechos del remitente para enviar mensajes. La credencial es un token opaco que es validado por el receptor. Cada implementación de credenciales se especifica como la configuración de un plug-in, permitiendo la coexistencia de múltiples configuraciones de autentificación en la misma red.
El resumen de mensajes garantiza la integridad de los datos del mensaje. Los mensajes también pueden ser encriptados y firmados para tener confidenciabilidad, integridad e irrefutabilidad. La intención de JXTA es ser compatible con la amplia gama de mecanismos de seguridad de la capa de transporte, ya aceptados para arquitecturas basadas en mensajes, tales como Secure Sockets Layer (SSL), Transport Layer Security (TLS) e Internet Protocol Security (IPSec). La implementación JXTA 2.0 minimiza el número de mensajes copiados y almacenados (buffereados) cuando se envían y reciben mensajes.

4.5. Grupos de pares (PeerGroups)

Los pares en el Proyecto JXTA se auto organizan en grupos de pares. Un grupo de pares representa un conjunto dinámico de pares que tienen intereses comunes y han aceptado una serie de políticas (membresía, intercambio de contenido, etc.). Cada grupo de pares es unívocamente identificado por un ID de grupo. El proyecto JXTA no indica cuando, donde o porque se crean los grupos, sólo describe como se crea un grupo, como se publica y se descubre. Los usuarios, desarrolladores de servicios y administradores de red pueden crear dinámicamente grupos para tener interacción entre los pares y emparejar las demandas de sus aplicaciones.

En la figura 9 el Grupo A (PeerGroup A) está dentro de un dominio físico que está detrás de un firewall. El Grupo de pares B (PeerGroup B) atraviesa múltiples dominios físicos. El grupo C (PeerGroup C) abarca hasta los límites de un dominio NAT. Un par puede pertenecer a varios Grupos de Pares al mismo tiempo.

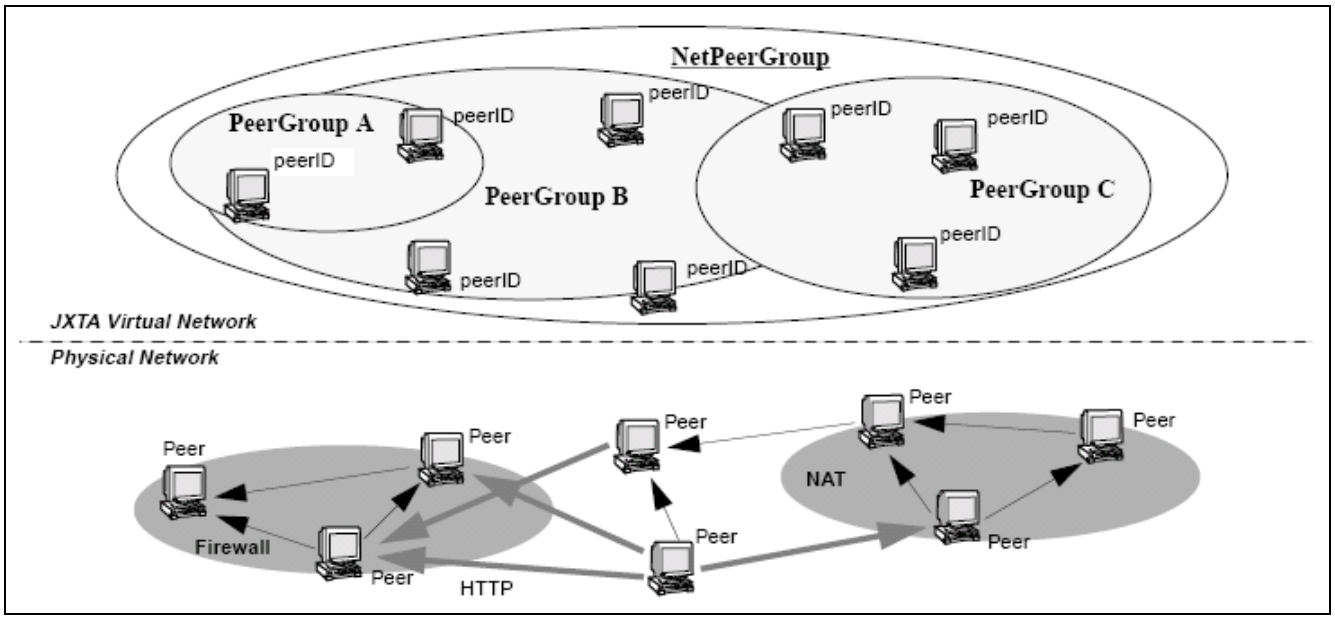


Figura 9

En el proyecto JXTA se reconocen tres motivos principales para crear grupos:
• Crear dominios seguros para intercambiar contenido seguro. Los grupos de pares forman regiones lógicas cuyas fronteras limitan el acceso a los que no son miembros. Un Grupo de Pares no necesariamente refleja los límites subyacentes de la red física, tales como aquellos impuestos por un router o un firewall. Los Grupos virtualizan el concepto de router y firewall subdividiendo la red en regiones seguras sin respetar los límites de la red física.

• Crear un radio de alcance. Los Grupos de pares son normalmente formados y auto organizados basados en los intereses mutuos de los pares. No se impone ninguna regla en particular a la manera en que se forman los pares, pero los pares con los mismos intereses tenderán a unirse a los mismos grupos. Los grupos sirven para

subdividir la red en regiones abstractas, brindando un mecanismo implícito de alcance para restringir la propagación de los requerimientos de búsqueda y descubrimiento.

• Crear un ambiente monitoreado. Los Grupos permiten el monitoreo (inspección de tráfico, contabilidad, traza) de cualquier par para cualquier propósito

Al inicializar todo par se une al NetPeerGroup. Este actúa como grupo raíz al que todos los pares pertenecen inicialmente. Los Grupos de Pares normalmente publican un conjunto de servicios llamados servicios de grupo (PeerGroup Services). El proyecto JXTA especifica un conjunto estándar de servicios básicos de grupo de pares con sus protocolos asociados (discovery, resolver, pipe, peer info, y rendezvous). Si un protocolo no es lo suficientemente adecuado para otro grupo más demandante, puede ser reemplazado con un protocolo modificado, para cumplir los requisitos particulares del nuevo grupo. Los anuncios de los grupos contienen la lista de todos los servicios que implementa el protocolo utilizado en dicho grupo. Los servicios básicos de grupo del Proyecto JXTA proveen la funcionalidad básica para dar soporte a la existencia de un grupo (publicar y descubrir recursos e intercambiar mensajes). Los creadores de los grupos pueden modificar los servicios de su grupo para cumplir con sus requerimientos particulares (centralizado o descentralizado, determinístico o no determinístico, etc.). Los servicios de grupo se componen de una colección de instancias del servicio ejecutándose en múltiples miembros del grupo. Cada instancia puede trabajar autónomamente como una réplica o en forma cooperativa con otros. Si algún par fallase, el servicio de grupo no se ve afectado porque es muy probable que el servicio siga disponible desde otro par miembro del grupo.

4.6. Pipes

Las tuberías o pipes son canales de comunicación virtual usados para enviar y recibir mensajes entre servicios y aplicaciones. Estos pipes proveen una abstracción virtual sobre los endpoints de los pares para proveer la ilusión de casillas de correo entrantes y salientes, que no están físicamente ligadas a la ubicación física de los pares. Los pipes pueden conectar uno o más endpoints. Se asume que se posee software en cada endpoint para enviar, recibir o administrar colas de mensajes. Los extremos de los pipes se denominan como “input pipe” al extremo final y “output pipe” al extremo transmisor (ver figura 10).
Los pipes son publicados y descubiertos utilizando los anuncios de pipe (pipe advertisements) y son unívocamente identificados utilizando un ID de pipe. Los extremos de un pipe son unidos dinámicamente en tiempo de ejecución por el resolver. Usando la abstracción de pipes, las aplicaciones y servicios pueden implementar tolerancia a fallas transparente desde un endpoint físico de un par a otro, con el objetivo de solucionar una falla de un par o un servicio, o para acceder a una nueva instancia publicada de un servicio. El proceso de enlazado de pipes consiste en buscar y conectar uno o mas extremos de un pipe. Cuando se envía un mensaje a un pipe, el mensaje es enviado por el pipe de salida local hacia el pipe de entrada de destino (que está escuchando a éste pipe).

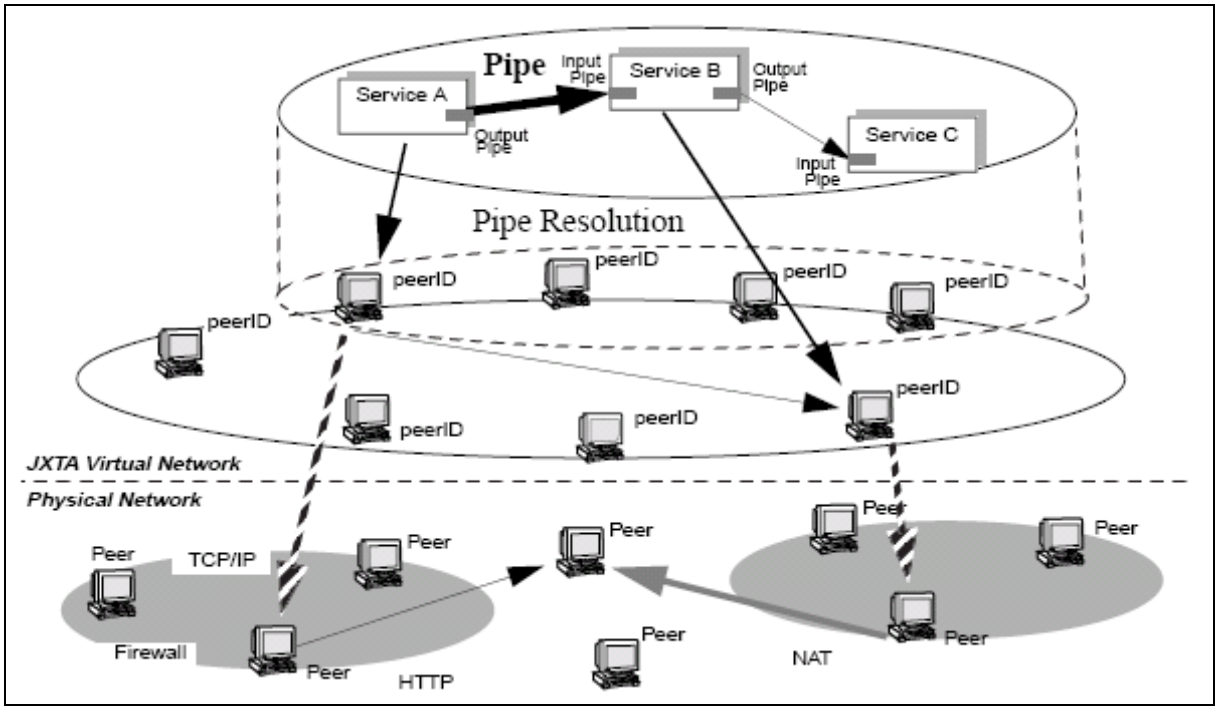


Figura 10: Pipes

Los pipes admiten dos modos de operación:
• Un pipe punto a punto conecta exactamente dos extremos de pipes por medio de una canal unidireccional y asíncrono, un extremo de entrada que recibe mensajes enviados por un extremo de salida. No hay

implementación de operaciones de respuesta o de reconocimiento (reply o de acknowledgement). Se puede requerir información adicional en el mensaje, como ID, para enlazar secuencias de mensajes. La información de los mensajes puede contener un anuncio de pipe que puede ser usado para abrir un nuevo pipe para responder al transmisor (enviar una respuesta).

• Un pipe de propagación (propagate peer) conecta la salida de un pipe a varias entradas de pipes. Los mensajes fluyen desde el extremo de salida del pipe (fuente de propagación) a los extremos de entrada de los pipes. El mensaje propagado se envía a todos los extremos de entrada de los pipes que están escuchando en el grupo. Este proceso puede crear múltiples copias del mensaje. En TCP/IP, cuando la propagación mapea una subred física de manera uno a uno, se puede usar IP multicast como una implementación para propagar pipes. La propagación de pipes se puede implementar usando comunicaciones punto a punto sobre transportes que no proveen multicast (por ejemplo HTTP).

Se han implementado servicios bidireccionables y confiables sobre los servicios de pipes del núcleo. Los pipes punto a punto asincrónicos pueden tener extremos que se conectan a diferentes pares en distintos momentos o no se conectan. Los anuncios de pipes pueden contener esquemas de tipos de datos XML únicos, para describir el conjunto válido de mensajes que pueden ser enviados o recibidos a través de un pipe. La implementación JXTA 2.0 introduce una API de sockets al servicio de los pipes para facilitar la programación.

4.7. Seguridad

Los modelos basados en certificados de autoridad que proveen una infraestructura de clave pública, para dar seguridad a las transacciones realizadas en la Web, son complejos, costosos de implementar y de mantener. JXTA provee un modelo base que no insume ningún costo y que puede ser generalizado fácilmente para soportar los modelos de confianza existentes, usados en internet. Este modelo es apropiado para chat rooms, compartir contenido y transacciones financieras seguras. El modelo de confianza de JXTA le permite a los pares ser ellos mismos su propia autoridad certificadora. Provee algoritmos cifrados muy fuertes para proteger los datos locales (todos los datos locales se protegen con passwords), los datos en tránsito por la red virtual JXTA y datos almacenados remotamente.
La Internet Engineering Task Force's (IETF) Transport Layer Security (TLS) es la continuación IETF de SSL.V3, y se usa para proveer seguridad a las comunicaciones entre computadoras. JXTA implementa un transporte virtual basado en TLS para proveer comunicaciones seguras entre los pares. El cifrado por defecto es RSA1024 con 3DES (“triple DES”, Data Encryption Standard, donde se usan 3 claves sucesivas) y SHA-1 (Secure Hash Algorithm que implemta una función hash que mapea objetos de longitud menor a 264 bits a valores hash de un largo exacto de 160 bits).

Cuando se crea un pipe seguro y se resuelven los extremos del par asociado se inicializa un transporte TLS virtual. Todos los datos transmitidos a través de pipes seguros son luego multiplexados en esta única instancia de la TLS virtual. El transporte es asegurado bidireccionalmente extremo a extremo con TLS independientemente de los relays JXTA y de los transportes de la capa física subyacente. Los pares pueden crear pipes que se comportarán como múltiples canales seguros sobre un único transporte TLS. La implementación TLS del Proyecto JXTA minimiza la necesidad de recursos de red, amortizando un handshake TLS sobre múltiples pipes de datos y haciendo un uso conservador del ancho de banda de la red física.

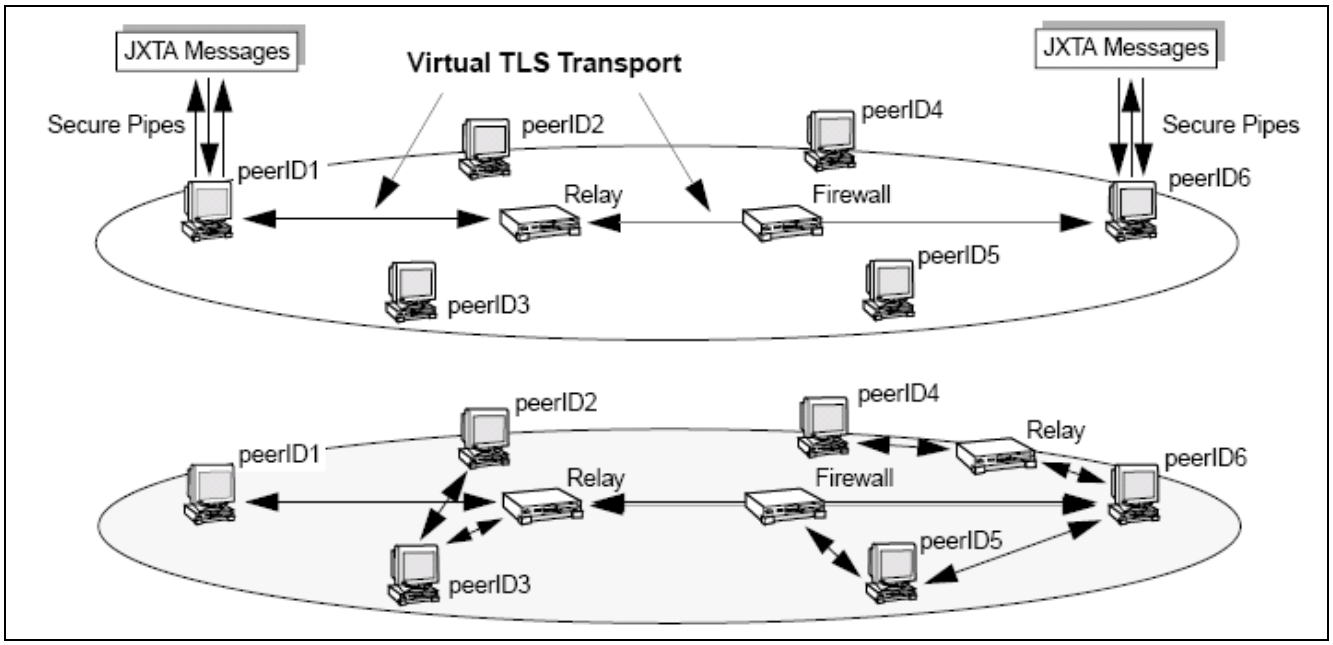


Figura 11: Transporte TLS virtual

Como el transporte virtual TLS es bidireccional, se requiere autentificación del cliente y del servidor porque los pares pueden ser ambos clientes o servidores. Los pares deben poseer los certificados de raíz X509.V3 de cualquier par con el cual deseen comunicarse en forma segura. Los certificados de raíz contienen la clave pública RSA de 1024 bits usada para verificar la firma de clave privada RSA de los certificados del servicio X509.V3 del par. Los certificados del servicio se usan para autentificar los extremos de los pares comunicantes por TLS, después de que las firmas de los certificados de raíz son localmente verificadas.
Es posible implementar autentificación de grupo basada en certificados X509.V3. Cuando uno se une a un grupo de pares recibirá el certificado de raíz del creador del grupo a través de una conexión TLS protegida y luego usara un Certificate Service Request (CSR) para adquirir un certificado X509.V3 de membresía al grupo, firmado con la clave privada del certificado de raíz del creador del grupo. Toda esta información se guarda localmente en el par, protegida por una password. Cuando un par contacta a otro miembro del grupo, el primero puede ser autentificado por el segundo usando el certificado de pedido/respuesta del handshake TLS y verificando el certificado.

4.8. Protocolos JXTA

Los protocolos del Proyecto JXTA se componen de una serie de seis protocolos (figura 12) divididos en dos categorías:
• Core Specification Protocols (Protocolos de especificación del núcleo)
• Standard Service Protocols (Protocolos de servicios estándares)

4.8.1. Core Specification protocols

Los protocolos JXTA han sido diseñados para ser implementados en sistemas muy pequeños, con sólo unos pocos requerimientos de componentes y comportamientos. La funcionalidad requerida por todas las implementaciones está definida por los Core Specification Protocols (Protocolos de especificación del núcleo). Las implementaciones que deseen ser compatibles con JXTA deben implementar todos los Core Specification Protocols. La implementación del Core Specification no garantiza, ni siquiera provee interoperabilidad con otras implementaciones JXTA. Hay una serie de comportamientos que quizás necesiten ser provistos por la implementación JXTA, que no son parte del Core Specification. Las implementaciones existentes de estos componentes se describen en forma separada en la especificación de los Standard Services, que se verá en la sección siguiente.

La Core Specification define dos protocolos:

• El Endpoint Routing Protocol (ERP). Es el protocolo por el cual un par puede descubrir una ruta (secuencia de saltos) usada para enviar un mensaje a otro par. Si el par A quiere enviarle un mensaje al par C, y no hay ninguna ruta directa entre A y C, entonces el par A necesita encontrar el o los pares intermediarios para rutear el mensaje a C.
ERP se usa para administrar y determinar la información de ruteo. Si la topología de red ha cambiado de forma tal que la ruta a C ya no puede ser usada, el par puede usar ERP para encontrar rutas conocidas por otros pares para construir una nueva ruta a C.

• El Peer Resolver Protocol (PRP) es el protocolo por el cual un par puede enviar una petición genérica a uno o más pares y recibir una respuesta (o múltiples respuestas). El protocolo PRP permite la diseminación de peticiones genéricas a uno o mas manejadores dentro del grupo y también fijarse si las peticiones coinciden (match) con respuestas. Cada petición es direccionada a un manejador específico. Este manejador (handler name) define la semántica particular del requerimiento y de su respuesta, pero no está asociado con algún par específico. Una petición dada puede ser recibida por cualquier número de pares dentro del grupo, posiblemente todos, y procesada de acuerdo al nombre de manejador si tal manejador está definido en alguno de esos pares.

4.8.2. Standard Service Protocols

La Core Specification define los componentes y comportamientos requeridos para todas las implementaciones JXTA. Para crear una implementación JXTA completa hay algunos componentes adicionales que todas las implementaciones deberían brindar. Los Standard Service Protocols (Protocolos de servicios estándares) son protocolos y comportamientos opcionales. No se requiere que las implementaciones implementen estos servicios, pero es altamente recomendado que lo hagan. Al implementar estos servicios se proveerá mayor interoperabilidad con otras implementaciones y más amplia funcionalidad.

La especificación de los Standard Service Protocols define cuatro protocolos:
• El Rendezvous Protocol (RVP) es el protocolo por el cual los pares pueden suscribir o ser un suscriptor de un servicio de propagación. Dentro de un grupo, los pares pueden ser rendezvous o pares que escuchan a los rendezvous. RVP permite enviar mensajes a todos los que escuchan un servicio. RVP es usado por el Peer Resolver Protocol (PRP) para propagar los mensajes.

• El Peer Discovery Protocol (PDP) es el protocolo por el cual un par publica sus propios anuncios y descubre anuncios de otros pares (pares, grupos de pares, módulos, pipes y contenido). PDP usa el Peer Resolver Protocol (PRP) para enviar y propagar peticiones de descubrimiento de anuncios.

• El Peer Information Protocol (PIP) es el protocolo por el cual un par puede obtener información del estado de otros pares, tales como estado, tiempo actividad, carga de tráfico, capacidades, etc. PIP usa el Peer Resolver Protocol (PRP) para enviar y propagar peticiones de información.

• El Pipe Binding Protocol (PBP) es el protocolo por el cual un par puede establecer un canal de comunicación virtual (o pipe) entre uno o más pares. El PBP es usado por un par para enlazar los dos (o más) extremos de conexión de un pipe (input y output pipe) con un extremo físico de un par. PBP usa el Peer Resolver Protocol (PRP) para enviar y propagar peticiones de enlazado de pipes.

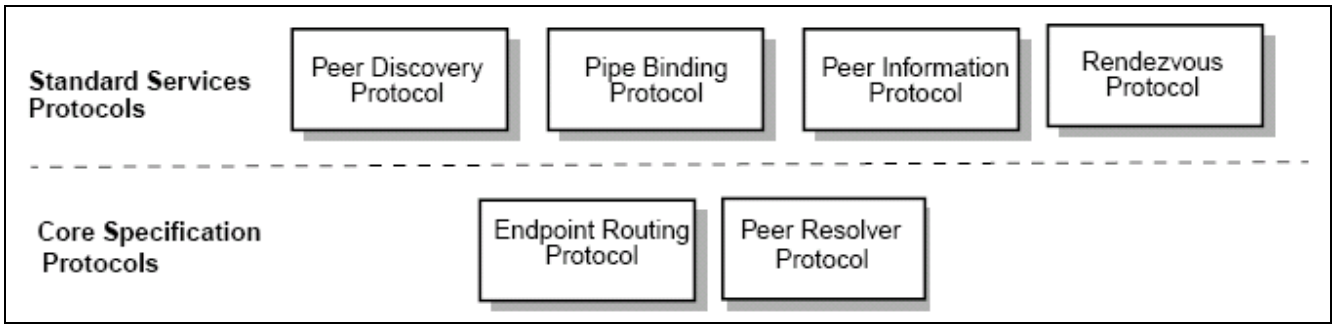


Figura 12: Protocolos JXTA

Los protocolos del proyecto JXTA han sido diseñados para ser implementados fácilmente en enlaces unidireccionales y transportes asimétricos. En particular muchas formas de redes inalámbricas y móviles no proveen a los dispositivos iguales capacidades de enviar y recibir. JXTA permite usar cualquier enlace unidireccional cuando sea necesario, mejorando así la conectividad de toda la red en el sistema. La intención es que los protocolos JXTA sean lo mas permanentes posibles y fáciles de implementar sobre cualquier transporte. Las implementaciones sobre transportes confiables y bidireccionales tales como TCP/IP o HTTP permiten alcanzar comunicaciones bidireccionales eficientes.
Un par solo necesita implementar los protocolos que él requiere. Por ejemplo, un dispositivo puede tener todos sus anuncios precargados en memoria y no necesita implementar el Peer Discovery Protocol (PDP). Un par puede usar un conjunto preconfigurado de relays para rutear todos sus mensajes, por lo tanto no requiere implementar el Endpoint Routing Protocol (ERP), solo envía los mensajes a los relays para que ellos los enruten. Quizás un par no necesite obtener o no quiera proveer información de su estado a otros pares, por ello no necesita implementar el Peer Information Protocol (PIP).
El diseño de los protocolos JXTA busca crear un conjunto de protocolos que sean de bajo costo, asuma muy poco del transporte de red de la subred física e imponga pocos requerimientos al ambiente de pares, y aún así sean posibles de usar para desarrollar una amplia variedad de aplicaciones P2P y servicios en un ambiente de red sumamente no confiable y cambiante.

4.9. Referencia de la Arquitectura de Implementación del Proyecto JXTA 2.0

Esta sección presenta una descripción de la referencia de implementación del JXTA 2.0 Java J2SE. La referencia de implementación provee un plano de diseño para que los desarrolladores puedan implementar los protocolos JXTA, los seis protocolos antes mencionados y algunos otros servicios típicos.
La implementación se descompone en un número de componentes interdependientes (figura 13). Cada componente define sus propias APIs para interactuar con otros componentes.
El objetivo es crear una implementación modular que permite reemplazar fácilmente o extender cada componente.

Administración de Anuncios

Los siguientes componentes proveen la infraestructura necesaria para administrar anuncios:

4.9.1. ID

Dentro de los protocolos JXTA hay un número de entidades que necesitan ser unívocamente identificadas (pares, grupos de pares, pipes, etc.). Una ID JXTA identifica unívocamente a cada una de estas entidades y sirve como un medio canónico de referirse a esa entidad. Este componente implementa IDs JXTA como UUIDs de 128 bits. Estas UUIDs son auto generadas localmente usando un generador aleatorio. Una ID JXTA es un estándar URN (Uniform Resource Name - Nombre de recurso uniforme, otro estándar de URL) en el espacio de nombres de ID JXTA. Los URNs de los JXTA IDs son identificados por el identificador del espacio de nombres URN jxta (por ejemplo urn:jxta:id12345).

4.9.2. Cache Manager (CM)

El Cache Manager o administrador de cache se usa para capturar, indexar y almacenar anuncios persistentes. Este componente permite el almacenamiento, indexado y recuperado de anuncios en forma eficiente. La implementación JXTA 2.0 usa una versión reducida a escala de la base de datos XML Xindice de Apache, que es open source, para almacenar y recuperar anuncios JXTA. La parte de indexado Xpath de Xindice no se usa. El cache manager permite especificar en cual campo se debe indexar un anuncio. Los anuncios de objetos Java son serializados y deserializados como los documentos XML. El cache manager provee almacenamiento persistente de anuncios después de los reinicios, también implementa un mecanismo de indexado eficiente para la recuperación de anuncios, optimizando el tiempo de recuperación para los anuncios más comunes (pares, grupos de pares, módulos, pipes, rutas, etc.) y controla la disminución gradual de anuncios en el cache. Cada anuncio se publica en el cache con un tiempo de vida asociado y se borra del cache cuando expira.

4.9.3. XML Parser

Un parser transforma los datos de entrada en una estructura de datos que puede ser procesada más fácilmente por ejemplo para corregir la semántica, generar código o simplemente para facilitar la comprensión de los datos de entrada. Un XML Parser lee un documento XML y determina la estructura y propiedades de los datos.
El parser permite la serialización y deserialización de objetos Java en flujos de caracteres XML de entrada y de salida. Se implementó un parser XML liviano que implementa la funcionalidad básica XML DOM. DOM o modelo de objeto de documento, define un conjunto estándar de comandos que los parsers devuelven para facilitar el acceso al contenido de los documentos XML desde sus programas. Un analizador de XML compatible con DOM toma los datos de un documento XML y los expone mediante un conjunto de objetos que se pueden programar. Los protocolos JXTA usan un subconjunto mínimo de XML (espacio de nombres limitado y no validación). Los dispositivos pequeños no requieren un parser completo ya que los mensajes pueden ser pre generados.

4.9.4. Advertisements (Anuncios)

El módulo de advertisements o anuncios implementa los anuncios principales usados en los protocolos JXTA: pares, grupos de pares, endpoints, rendezvous, transporte, pipes, módulos y ruteo. El módulo de anuncios le permite a un objeto Java ser serializado y deserializado en un documento XML que represente el anuncio.

Administración Transporte Físico y Mapping entre Virtual Messenger y Conexiones Físicas

Los componentes siguientes administran el transporte físico y mantienen el mapeo entre mensajeros virtuales (virtual messenger) y conexiones físicas:

4.9.5. Transporte HTTP

El transporte HTTP implementa el enlace de transporte HTTP que se especifica en las especificaciones del protocolo de enlace (Binding) JXTA. El transporte HTTP se implementa como servlets usando el servidor Jetty. Los servlets son objetos que corren dentro del contexto de un servidor de aplicaciones y extienden su funcionalidad, permitiendo a los usuarios ejecutar código Java en el servidor y enviar páginas HTML a un browser. Jetty es un servidor HTTP java y contenedor de servlets, así que no es necesario correr otro servidor web (como Apache) para poder usar servlets. El transporte HTTP se ejecuta en el mismo proceso que la plataforma JXTA y provee la posibilidad de iniciar la conexión entre dos pares. La conexión primero se usa para determinar las identidades lógicas de los pares participantes.
Se utiliza la petición HTTP GET para determinar la identidad lógica del lado del servidor HTTP y para interrogar en busca de mensajes. Se utiliza la petición HTTP POST para enviar y recibir mensajes JXTA.
El transporte HTTP soporta atravesar firewalls a través de proxys.

4.9.6. Transporte TCP/IP

El transporte TCP/IP implementa un enlace de transporte TCP/IP como se especifica en las especificaciones del protocolo de enlace (Binding) JXTA. La implementación JXTA aprovecha las conexiones bidireccionales, así un único socket puede ser usado para enviar y recibir datos. Las conexiones inactivas son recicladas para permitir manejar un gran número de conexiones en relays y rendezvous. Se puede establecer un número limitado de conexiones físicas (actualmente tres) a un mismo destino para transferir datos simultáneamente. El transporte TCP/IP puede ser configurado para aceptar mensajes de difusión broadcast a través de Multicast IP.

4.9.7. Transporte Virtual TLS

El transporte TLS virtual (virtual Transport Layer Security) implementa un transporte extremo a extremo, confiable y seguro sobe HTTP y TCP/IP. La implementación del transporte virtual TLS divide los mensajes en registros TLS. Un protocolo de mensajes confiable se usa para garantizar que los mensajes que contienen estos registros lleguen a su par de destino en el mismo orden en el que han sido transmitidos. El transporte TLS lleva a cabo el intercambio de claves y la negociación de una clave de sesión para encriptar los datos.

4.9.8. Mensajes

El servicio de mensajes implementa el formato wire binary JXTA usado cuando los mensajes se envían en la red virtual JXTA. El proyecto JXTA usa formatos binarios para permitir una transferencia eficiente tanto en datos binarios como en XML. Todos los protocolos de mensajes JXTA son representados como datos XML. Los mensajes se forman como una secuencia ordenada de elementos. Cada elemento tiene un nombre único, longitud y tipo MIME (Multipurpose Internet Mail Extensions, permite especificar el tipo de archivo enviado y el método que debe ser usado para volverlo a su forma original). Es importante destacar que la representación del wire format JXTA es independiente de la representación de los datos. Cualquier tipo de datos puede ser enviado. Cada servicio, cuando procesa un mensaje, puede añadir o eliminar sus propios elementos del mensaje. Los elementos del mensaje permiten aislar las diferentes partes del mensaje, permitiendo aumentar la protección. Por ejemplo, el servicio de Ruteo está autorizado a manipular los elementos de ruteo del mensaje, pero no puede tocar ningún otro elemento.

4.9.9. Mensajería Virtual (Virtual Messenger)

El servicio de mensajería virtual abstrae todos los transportes JXTA a través de una interfase común para el servicio de extremos (endpoint service). El virtual messenger normaliza el comportamiento de transportes de mensajería sincrónicas versus asincrónicas, con comportamientos bien definidos con respecto al sincronismo para enviar y recibir mensajes.

4.9.10. Servicio de Extremos (Endpoint Service)

El endpoint service implementa la abstracción de extremo JXTA, que es usada para encapsular múltiples transportes de mensajería en un único extremo (endpoint) virtual. Los extremos (endpoints) proveen la abstracción de red virtual usada por los pares para comunicarse independientemente de la topología de la red subyacente (firewalls o NATs) y de los transportes físicos. El servicio de endpoint provee demultiplexado uniforme de los mensajes entrantes y provee administración de los recursos asociados. El servicio de endpoint delega la propagación de la red y el establecimiento de la conexión al transporte de mensajería apropiado. También provee almacenamiento temporal (buffering) de los mensajes salientes, captura de mensajeros salientes y un comportamiento de mensajería uniforme.

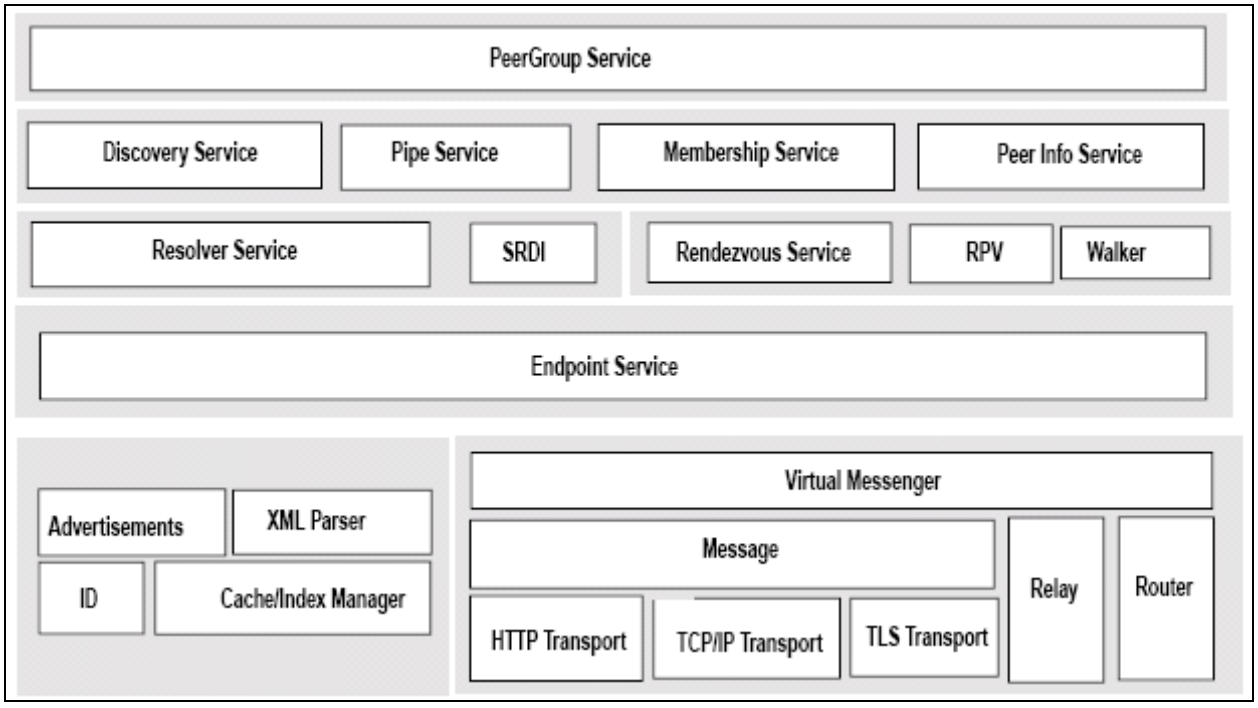


Figura 13: descripción de la referencia de implementación del JXTA 2.0 Java J2SE

4.9.11. Ruteador (Router)

El ruteador (router) implementa el servicio de ruteo usado por el endpoint de un par para descubrir y mantener la información de la ruta a otros pares. Puede que algunos pares no tengan una conexión directa entre si, por lo que los mensajes se deberán rutear a través de uno o mas pares hasta alcanzar su destino final. El router implementa el Endpoint Routing Protocol (ERP) permitiéndole a un par requerir y descubrir información de ruteo. El router mantiene las tablas de rutas descubiertas en memoria. Todo mensaje enviado y recibido contiene un elemento de ruteo usado para actualizar la información de rutas. El router provee información de ruteo al servicio de Endpoint para entregar mensajes.

4.9.12. Relay
El servicio de relay provee un mecanismo para que un par que no es alcanzable directamente pueda usar un par relay para almacenar mensajes hasta que puedan ser entregados. El servicio de relay es usado para atravesar firewalls, reverse proxys y NATs. Este servicio un mecanismo de cuota y leasing para administrar colas de mensajes para pares extremos.

Servicios Opcionales
Los siguientes componentes definen servicios opcionales para soportar grupos de pares JXTA:

4.9.13. Servicio Rendezvous
El servicio de rendezvous es usado para propagar mensajes en el entorno de un grupo de pares. Este servicio implementa el RendezVous Protocol (RVP). El servicio rendezvous le permite a los pares extremos de la red obtener un leasing para enviar y recibir mensajes propagados. El servicio rendezvous usa el servicio de endpoint (Endpoint Service) para propagar mensajes.

4.9.14. Rendezvous Peer View (RPV)
El servicio RVP administra una lista de rendezvous disponibles en un grupo. El RVP mantiene una visión débilmente acoplada y descentralizada de todos los rendezvous, que converge con el paso del tiempo. Cada rendezvous tiene su propia lista RVP.

4.9.15. Walker
El servicio de walker provee un mecanismo para recorrer o “caminar” por el RVP del rendezvous para propagar peticiones. El walker implementa una política por defecto de recorrer el rendezvous desde el Shared-Resource Distributed Index (SRDI) rendezvous objetivo inicial en dirección hacia arriba y hacia abajo.

4.9.16. Servicio de Resolver
El servicio de resolver se usa para enviar peticiones de búsqueda/respuesta genéricas dentro del alcance del grupo de pares. El servicio de resolver implementa el Peer Resolver Protocol (PRP). El resolver usa el servicio de endpoint y el de rendezvous para unicast y para propagar peticiones dentro del grupo

4.9.17. SRDI
El servicio SRDI distribuye índices de recursos compartidos dentro de la red de rendezvous. Estos índices pueden ser usados para orientar las peticiones en la dirección donde es más probable que sean resueltas, o repropagar mensajes a pares interesados en estos mensajes propagados.

4.9.18. Servicio de descubrimiento
El servicio de descubrimiento se usa para descubrir y publicar cualquier tipo de anuncio (pares, grupos de pares, pipes, etc.) dentro de un grupo. Este servicio implementa el Peer Discovery Protocol (PDP). El servicio de descubrimiento usa el servicio de resolver para enviar y recibir peticiones de descubrimiento. Este servicio también usa el cache manager para capturar y almacenar anuncios.

4.9.19. Servicio de Pipe
El servicio de pipe se usa para crear y enlazar extremos de pipes (input pipes y output pipes) a los endpoints de los pares en el alcance del grupo. Se implementan tres tipos de pipes, unicast (uno a uno), seguro y propagado (uno a N). El servicio de pipe usa un pipe resolver para enlazar dinámicamente el extremo de un pipe al endpoint de un par. El servicio de pipe resolver implementa el Pipe Binding Protocol (PBP).

4.9.20. Servicio de Membresía
El servicio de membresía se utiliza para administrar la membresía a un grupo y emitir credenciales de membresía. Se requiere que los pares nuevos sean autentificados antes de que puedan unirse al grupo. El servicio de membresía provee un framework de autentificación para soportar diferentes mecanismos de autentificación (JAAS, LDAP).

4.9.21. Servicio de Información de Par
El servicio de información de par provee un framework para medir y monitorear pares. Los monitores de medición pueden ser asociados con cualquier servicio de grupo para recolectar información sobre ese servicio. El servicio de información de par provee la capacidad de monitoreo remoto para recolectar datos sobre pares remotos. El servicio de información de par implementa el Peer Information Protocol (PIP).

4.9.22. Servicio de Grupo de Pares
El servicio de grupo de pares implementa la membresía por defecto al NetPeerGroup y la navegación del grupo. El servicio expone todos los servicios básicos del NetPeerGroup (discovery, resolver, pipe, rendezvous, etc.) a las aplicaciones que se han unido al NetPeerGroup. El servicio de grupo de pares le permite a un par

crear un grupo, anunciar y unirse a nuevos grupos. Los grupos de pares exportan un conjunto de servicios de grupo que se representan como módulos. El servicio de grupo de pares administra la infraestructura subyacente para anunciar y cargar módulos. El servicio de grupo de pares también administra el archivo de configuración PlatformConfig.

4.10. Estado y Directivas Futuras

Los protocolos de la red JXTA establecen una red virtual sobre las redes existentes, ocultan su topología física subyacente. El proyecto JXTA le permite a los desarrolladores de aplicaciones, no sólo a los administradores de red a diseñar topologías de red que se ajusten a los requerimientos de cada aplicación en particular. Se pueden crear múltiples redes virtuales ad hoc y mapearlas dinámicamente en una red física alcanzando un mundo de redes virtuales multidimensionales. El proyecto JXTA vislumbra un mundo donde billones de servicios de red, todos direccionables en la red serán capaces de descubrirse e interactuar entre si en una manera Ad Hoc y descentralizada, mediante la formación de una multitud de redes virtuales. La red virtual JXTA estandariza la manera en la cual los pares se descubren, se auto organizan en grupos, descubren recursos de red, se comunican y se monitorean entre si.
El proyecto JXTA se construye a partir de varias abstracciones (direccionamiento uniforme de ID de pares, grupos, anuncios, resolver y pipes) que proveen una infraestructura genérica para desplegar servicios y aplicaciones P2P. Al proveer estos mecanismos base y no dictar políticas, el Proyecto JXTA posibilita desarrollar una gran diversidad de aplicaciones P2P.
En este trabajo se ha descrito la implementación del Proyecto JXTA 2.0 que introduce una serie de nuevas características para mejorar la funcionalidad, performance y escalabilidad e la red JXTA. Hace una fuerte diferenciación en la manera en que los super-peers (los relay y los rendezvous) se comportan e interactúan con los pares extremos. La implementación JXTA 2.0 introduce el concepto de rendezvous peer view para interconectar a los rendezvous dentro de un grupo. Las peticiones de resolver no se propagan mas a los pares extremos como en la versión JXTA 1.0 reduciendo así el tráfico de red.
Los pares extremos usan un hash index distribuído flexible para indexar anuncios en el rendezvous peer view para búsquedas de requerimientos mas eficientes. La implementación JXTA 2.0 provee una mejor administración de recursos (threads y queues) e implementa límites para el uso de recursos, con el objetivo de asignar recursos mas equitativamente entre servicios. La referencia de implementación completa de los protocolos JXTA 2.0 está disponible en *platform.jxta.org.*
Se continúa mejorando la escalabilidad de implementación, añadiendo mas infraestructura de seguridad y mejores herramientas de monitoreo. También se busca proveer mejor integración y soporte para servicios web (SOAP, UDDI, WSDL).

5. Conclusión

Se ha desarrollado una plataforma de programación de red específicamente diseñada para ser la base de sistemas peer to peer.

Sus objetivos son:

• Interoperabilidad: todos los sistema P2P construidos con JXTA puede dialogar entre si.
• Independencia de plataforma: JXTA puede ser implementado con cualquier lenguaje de programación y ejecutarse en cualquier plataforma de software y hardware.
• Ubicuidad: JXTA puede ser implementado en cualquier dispositivo con un heartbeat digital. Un heartbeat es un mensaje (corto) de estado enviado una máquina administradora de la red (o a una máquina determinada), a intervalos regulares de tiempo con el propósito de monitorear el estado de salud.

JXTA se define como un conjunto de protocolos que usan mensajes XML encodeados. Se mantiene fuera de las APIs y es independiente del lenguaje de programación, de forma tal que puede ser implementado en C/C++, Java, Perl o en otros lenguajes. Esto significa que dispositivos heterogéneos con software completamente diferente pueden interoperar a través de los protocolos JXTA. Puede ser implementado sobre TCP/IP, HTTP, Bluetooth, HomePNA y muchos otros protocolos. Esto significa que un sistema construido sobre JXTA funciona en la misma manera cuando el sistema se expande a un nuevo entorno de red o a un nuevo tipo de dispositivo, siempre que haya un manejador de protocolo de transporte correcto para el nuevo protocolo de red.
Los beneficios del proyecto JXTA pueden ser ilustrados con algunas aplicaciones o ejemplos de uso. Por ejemplo supongamos que hay una comunidad P2P ofreciendo capacidades de búsqueda para sus miembros, donde un miembro puede enviar una solicitud y otros miembros pueden escucharla y enviar una respuesta. Se puede pensar que un miembro es usuario de Napster e implementa una función que cuando una búsqueda contenga una petición de un archivo MP3, éste miembro buscará en su propio Napster y luego responderá a la petición con la información devuelta por el sistema Napster. Aquí un miembro sin ningún conocimiento de Napster se puede beneficiar porque otro miembro implementó un puente para conectar su sistema P2P a

Napster. Este tipo de puente es muy útil, pero cuando el número de servicios es muy grande se vuelve más difícil y es indeseable. JXTA pretende ser el puente de plataformas que conecta varios sistemas P2P.
Otro ejemplo podría ser suponer que un grupo de ingeniería requiere una capacidad de almacenamiento escalable pero con redundancia para proteger los datos de posibles pérdidas. Una solución común es comprar un sistema de almacenamiento con una gran capacidad y discos espejados. Otro grupo de ingeniería en otro sector de la empresa decide comprar el mismo sistema. Ambos grupos terminan teniendo una gran capacidad extra y tienen que pagar altos precios por la capacidad extra de espejado. Usando tecnología JXTA los grupos pueden crear un sistema que requerirá que sólo compren un único sistema de almacenamiento más simple sin espejado. En este sistema los discos se pueden descubrir entre si automáticamente y formar un grupo de pares de almacenamiento, espejando sus datos haciendo uso de su propio espacio libre.
Como un tercer ejemplo muchos dispositivos (teléfonos celulares, pagers, dispositivos de mail inalámbricos, PDAs y PCs) poseen información de agenda y calendario. Actualmente el proceso de sincronización entre ellos es en algunos casos tedioso o difícil. Muchas veces la PC se vuelve el punto central de sincronización, donde todos los dispositivos se deben conectar a la PC usando un driver único para cada dispositivo. Con tecnología JXTA todos estos dispositivos pueden interactuar entre si sin la necesidad de interfases de red extras, excepto aquellas necesarias por el mismo dispositivo.
JXTA podría ser la capa común de comunicación e intercambio de datos.

6. Bibliografía

Toda la bibliografía utilizada fue obtenida del sitio oficial del Proyecto JXTA, http://www.jxta.org

- "JXTA™ Technology: Creating Connected Communities"
Murray Stein.
Enero de 2004.

- "Project JXTA: An Open, Innovative Collaboration"
Sun Microsystems Inc. posts.
25 Abril de 2001.

- "Project JXTA 2.0 Super-Peer Virtual Network"
Bernard Traversat, Ahkil Arora, Mohamed Abdelaziz, Mike Duigou, Carl Haywood, Jean-Christophe Hugly, Eric Pouyoul, Bill Yeager.
25 de Mayo de 2003.

- "Project JXTA: A Technology Overview"
Li Gong.
29 de Octubre de 2002

Apéndice

Usando el shell JXTA.

Se habló anteriormente del shell JXTA y de su importancia, sobre todo en la etapa de aprendizaje. Este shell JXTA es una aplicación que demuestra algunos conceptos claves de la plataforma de desarrollo JXTA. Con el shell se pueden crear pares, grupos, pipes, etc. Un usuario puede publicar anuncios, descubrir pares, grupos de pares, unirse a ellos, crear pipes para conectar dos pares y enviar y recibir mensajes.
Se dispone de una LAN con dos PCs de escritorio corriendo el shell JXTA que se descargó de jxta.shell.org. Esto es un código prototipo que demuestra algunas de las características del shell. Cada PC será un par.
Se procederá a realizar una serie de operaciones didácticas para familiarizarse con los comandos del shell JXTA.
Al inicializar el shell se visualiza:

```
==========================================
=======<[ Welcome to the JXTA Shell ]>=======
==========================================

The JXTA Shell provides an interactive environment to the JXTA
platform. The Shell provides basic commands to discover peers and
peergroups, to join and resign from peergroups, to create pipes
between peers, and to send pipe messages.

The Shell provides environment  variables that permit binding
symbolic names to JXTA platform objects. Environment variables
allow Shell commands to exchange data. The command 'env' displays
all defined environment variables in the current Shell session.

The Shell creates a JXTA InputPipe (stdin) for reading input from
the keyboard, and a JXTA OutputPipe (stdout) to display
information on the Shell console. All commands executed by the
Shell have their initial 'stdin' and 'stdout' set up to the
Shell's stdin and stdout pipes. The Shell also creates the
environment variable 'stdgroup' that contains the current JXTA
PeerGroup in which the Shell and commands are executed.

The 'man' command is available to list the commands available.
Type 'man <command>' to get help about a particular command. To
exit the Shell, use the 'exit' command.
JXTA>
```

A partir de allí se pueden utilizar los siguientes comandos de shell.

Comando	***Descripción***
cat	Muestra en el stdout el contenido de un objeto jxta guardado en una variable, actualmente limitado a tipos jxta. jxta> mkadv -p apipe jxta> cat env1 jxta> cat -p env1
chpgrp	El comando chpgrp se usa para cambiar la variable del shell stdgroup, grupo por defecto, a otro grupo al que se unió previamente con el comando join. El comando join se usa para unirse a un grupo. Después de cambiar de grupo, la variable de shell stdgroup toma el valor del nuevo grupo al que se unió.
env	Env muestra todas las variables definidas en la sesión de shell. Las siguientes variables se definen por defecto: consin = Default Console InputPipe consout = Default Console OutputPipe stdout = Default OutputPipe stdin = Default InputPipe Shell = Root Shell stdgroup = Default peer group rootgroup = Default NetPeerGroup Las variables de shell se definen como resultado de ejecutar comandos de shell. El operador `=´ puede ser usado para asignar valores a una variable en particular. Por ejemplo *myenv = mkmsg* le asignará un nuevo objeto mensaje a la variable *myenv*.

exit	Sale del shell.
exportfile	Exporta el contenido de una variable de shell a un archivo externo. El objeto exportado se guarda en el nombre de archivo suministrado como argumento. Si no se da ningún nombre de variable, se usa el pipe stdin para leer datos y salvarlos en el archivo. exportfile es la operación inversa de import file. La variable de shell no se borra después de crear el archivo. jxta> exportfile -f /home/userdir/filename peer0 jxta> importfile -f /home/userdir/filename envpeer
get	get devuelve el tag body (la etiqueta cuerpo) de un mensaje. Los mensajes JXTA se componen de una serie de tag body, cada una identificada con un nombre de tag único. Se da el nombre de tag de un mensaje para saber específicamente cual tag body extraer. jxta> pipeadv = mkadv -p mypipe jxta> inpipe = mkpipe -i pipeadv jxta> Shell -s en la nueva ventana de shell jxta> opipe = mkpipe -o pipeadv jxta> mymsg=mkmsg jxta> importfile -f PlatformPeerGroup file1 jxta> put mymsg mytag file1 jxta> send opipe mymsg volviendo a la primer ventana jxta> msg = recv inpipe En este punto se puede hacer cat msg, que mostrará el tag y el msg o hacer lo siguiente, que retornará msg inserto en la variable data jxta> data = get msg mytag jxta> cat -p data
groups	Descubre grupos de pares por propagación o en una ubicación específica. Por defecto groups lista todos los grupos de pares conocidos por el par local. La opción –r se usa para enviar una solicitud propagada para encontrar nuevos grupos. groups guarda los resultados en el cache local e inserta anuncios en el entorno, usando el nombramiento por defecto: groupX donde X es un entero creciente. jxta> groups jxta> groups -r jxta> groups jxta> mkpgrp soccer en otro par jxta > groups -r -aName -vsoccer jxta> groups (debería mostrar el grupo soccer)
help	Muestra la ayuda del shell y los comandos disponibles
importfile	Importa un documento externo estructurado al ambiente de shell jxta> importfile -f /home/userdir/afile envfile jxta> cat envfile
join	El comando join se usa para unirse a un grupo que fue creado a través del comando mkpgrp o usando un anuncio que fue previamente descubierto. Para unirse a un grupo, el grupo necesita ser primero descubierto o creado. Si no se le dan argumentos, join lista todos los grupos existentes y su estado (join, unjoined) y el grupo actual al que pertenece el par. Después de que el par se unió a un grupo, se crea la variable 'PG@<group name>' que contiene la información del grupo. Cuando se hace un join a un nuevo grupo, este es anunciado en el NetPeerGroup. No hay soporte para ningún tipo de jerarquía de grupos. crear un grupo de pares: jxta> mkpgrp volleyball jxta> join volleyball identity: nobody Cuando se hace join a un nuevo grupo se puede requerir una identidad, dependiendo del tipo de membresía al grupo. La identidad puede requerir un password. En el caso anterior se usa un autentificador null con la identidad 'nobody' Abandonar el grupo y unirse a otro jxta> leave jxta> groups -r -a Name -v soccer jxta> groups jxta> join -d group#

leave	El comando leave se usa para abandonar un grupo al que se unió previamente con el comando join. Después de abandonar un grupo la variable de shell stdgroup se resetea al valor por defecto del rootgroup (que es el NetPeerGroup). Antes que un usuario pueda usar el grupo de vuelta, el usuario deberá re-unirse al grupo con el comando join. jxta>mygroupadv = mkadv -g mygroup jxta>mkpgrp -d mygroupadv mygroup jxta>join mygroup jxta>leave
mkadv	Crea un anuncio de grupo o de pipe a partir de un documento guardado en una variable de shell. Se puede especificar un tipo particular de pipe (TBD). Se puede asociar un nombre simbólico con el pipe. El nombre puede ser usado para buscar este mismo anuncio. Cuando se crea un nuevo anuncio de grupo se debe proveer un nombre, que será el nombre del grupo. -p para un anuncio de pipe -g para un anuncio de grupo -d para especificar un documento a usar para crear el objeto anuncio jxta> agroupadv = mkadv -g -d fileimported jxta> mkpgrp -d agroupadv mygroup
mkmsg	Crea el objeto msg para usar enviar o recibir mensajes de pipes. Ver el ejemplo de get.
mkpgrp	Crea un nuevo grupo usando el anuncio de grupo suministrado. Si no se provee ningún anuncio, el comando creará un clon del grupo NetPeerGroup con el nombre especificado. El comando mkadv –g se usa para crear un anuncio de grupo. La variable *PG#<group name>* se crea para guardar la información del nuevo grupo. El nuevo grupo es anunciado en el NetPeerGroup. Todos los grupos son creados en el NetpeerGroup, que actúa como World Peer group. Todo grupo de pares puede ser encontrado en este grupo. jxta> mkpgrp soccer jxta> group_from_a_imported_file= mkpgrp -d file1
mkpipe	Crea un input pipe o un output pipe a partir de un documento de anuncio de pipe dado. Para que los pipes se comuniquen se debe crear un input pipe y un out pipe con el mismo anuncio de pipe. Los anuncios de pipes son documentos estructurados que contienen al menos la ID de pipe, que identifica unívocamente al pipe en el mundo JXTA. Los pipes no son ubicados o enlazados a un par físico. La conexión de pipes se establece buscando un anuncio de pipe y resolviendo dinámicamente la ubicación de un objeto input pipe enlazado con ese anuncio. Un input pipe puede ser enlazado al mismo anuncio de pipe en múltiples pares en forma transparente para el output pipe. El output pipe no necesita saber en que par físico se encuentra el input pipe. Para poder comunicarse con el pipe, el output pipe necesita buscar el input pipe enlazado a ese anuncio. Ver el ejemplo de get.
more	Pagina a través de un objeto de shell.
peerconfig	Se usa para reconfigurar un par. El comando fuerza a la plataforma a mostrar un cuadro de diálogo para configuración, la próxima vez que se inicia. Después de ejecutar este comando se necesita reinicializar la plataforma.
peerInfo	Se usa para obtener información sobre otros pares dentro del grupo o en una ubicación específica. La opción por defecto lista sólo los anuncios de información de pares conocidos por el par. La opción –r se usa para enviar una solicitud de búsqueda propagada para encontrar información sobre nuevos pares. El comando peerinfo guarda los resultados en el cache local e inserta anuncios en el entorno usando la denominación por defecto: peerinfoX donde X es un entero creciente. jxta> peerinfo -r jxta> peerinfo jxta> cat peerinfo1
peers	Se usa para descubrir a otros pares dentro del grupo o en una ubicación específica. Si se ejecuta sin opciones, se lista sólo los pares ya conocidos por el par. La opción –r se usa para enviar una solicitud de búsqueda propagada para encontrar nuevos pares. El comando peers guarda los resultados en el cache local e inserta anuncios en el entorno usando la denominación por defecto: peerX donde X es un entero creciente. jxta> peers jxta> peers -r jxta> peers
put	Guarda un documento en el cuerpo del mensaje. Los mensajes JXTA se componen de una serie de tag body, cada una identificada con un nombre de tag único. Se da el nombre de tag de un mensaje para saber especificamente cual nombre tag usar para guardar el documento. En el extremo receptor el documento puede ser recuperado con el comando get.

	jxta> importfile -f /home/username/myfile mydata jxta> msg = mkmsg jxta> put msg mytag mydata
recv	Recibe un mensaje de un input pipe. El input pipe necesita haber sido previamente creado. Puede darse un timeout en segundos. Si no se especifica un timeout, la llamada se bloqueará hasta que se reciba un mensaje. Un timeout de cero corresponde a una llamada no bloqueada. Ver el ejemplo de get.
search	Busca un documento de anuncio que tiene un campo igual (valor de tag). Los documentos se buscan en el cache del par local o remotamente en los miembros del grupo. La búsqueda remota se hace en forma asíncrona. No hay garantía de que un anuncio publicado sea encontrado. Se hace el mejor esfuerzo, pero el par que ha publicado el anuncio puede no estar disponible en el momento que se realiza la búsqueda. La capacidad de búsqueda es rudimentaria, sólo maneja tags de primer nivel en los documentos. Por defecto se buscan documentos localmente. jxta> search -r name identity jxta> search name indentity
send	Envía un mensaje a un pipe. Ver el ejemplo de get.
setenv	Setéa una variable de entorno.
share	Comparte un documento anunciado en el grupo actual. El documento se hace visible a todos los miembros del grupo. Los anuncios son documentos XML que pueden representar objetos JXTA anunciados. Los documentos se buscan en el cache local del par o remotamente con el comando search. jxta> share mydoc
Shell	Inicia un nuevo shell jxta> Shell -s (inicia un nuevo shell en una ventana nueva) jxta> Shell -s -f /home/username/batchfile (inicia un nuevo shell en una ventana nueva y ejecuta batchfile)
talk	Este comando implementa una mensajería simple de línea de comando, para que dos usuarios en dos pares remotos puedan intercambiar mensajes. Estos mensajes se visualizan en la consola del shell. Para usar talk los usuarios deben estar registrados e implementar un pipe para escuchar, ésto se hace siguiendo una serie de pasos: 1. Registrarse utilizando el comando: talk –register <nombre de usuario> Este comando crea un anuncio de talk para ese usuario. Esto debe hacerse únicamente una vez, la primera vez que el usuario se registra con talk en un grupo específico. Se puede agregar la opción –secure para establecer una sesión de talk segura y –propagate para establecer una sesión de talk al estilo chat room. 2. Loguearse utilizando el comando: talk –login <nombre de usuario> Este comando loguea al usuario e inicializa un demonio que escucha. Este comando debe usarse cada vez que el par se reinicia. 3. Iniciar la conversación con otro usuario destino usando el comando: talk –u <mi nombre de usuario> <nombre de usuario destino> Luego de usar este comando se puede empezar a enviar mensajes. jxta> talk -register user1 jxta> talk -login user1 en otra ventana de shell o en otro par jxta> talk -register user2 jxta> talk -login user2 jxta> talk user1 Para dejar de recibir más mensajes, el usuario puede detener el demonio que escucha conversaciones (talk) con el comando: talk -logout <nombre de usuario>
wc	Cuenta las líneas nuevas, palabras separadas con espacios en blanco y caracteres en el objeto de shell especificado o en el input pipe si no se especifica un objeto. Escribe una línea de cuentas en el output pipe. Las cuentas se escriben en orden: líneas, palabras, caracteres. Por defecto wc escribe las tres cuentas. Se puede especificar que se escriban determinadas cuentas con otras opciones. Las opciones no deshacen otras dadas previamente, así wc –c –l escribe las cuentas de caracteres y líneas. jxta>peers | wc
whoami	Muestra información sobre un par o un grupo de pares. Sin opciones retorna información sobre en par local. Con la opción –g retorna información sobre el grupo al que se unió.
version	Muestra la versión actual del shell, build y la fecha.

Se puede obtener ayuda e información detallada del uso de los comandos utilizando `'man <command>'`

Se utilizarán dos PCs conectadas en una LAN. Una de ellas actúa como gateway, de forma que ambas tienen conexión a internet. En una PC que será el Par A se registra el usuario “mariano”.
Si usamos el comando peers sin argumentos listará los pares que se encuentren en el grupo. Con la opción –r se propaga una búsqueda de otros pares.

```
JXTA>peers
peer0: name = JXTA.ORG 123:9701/9700
peer1: name = mariano
```

Inicialmente se pertenece al NetPeerGroup por defecto. Si no se le pasan argumentos join muestra todos los grupos existentes y el grupo actual.

```
JXTA>join
rootgroup     "NetPeerGroup"     (current)
stdgroup      "NetPeerGroup"     (current)
worldgroup    "World PeerGroup"
```

El Par A es un par extremo (EDGE), se verifica con el comando rdvstatus.

```
JXTA>rdvstatus

Rendezvous Status:
__________________

Current configuration : EDGE
     "mariano" A  p  t  [45/26]

Peer View :
     [None]

Rendezvous Connections :
     [None]

Rendezvous Disconnections :
     JXTA.ORG 123:9701/9700
```

El Par A crea un nuevo grupo, con el comando newpgrp, llamado “grupodeprueba” y se une al mismo actuando como rendezvous. Para ello se usa el join con la opción –r. Se pide que se ingrese una Identidad que será la que tendrá ese usuario en el grupo. Una identidad se usa para asignar credenciales a los usuarios cuando acceden a los recursos del grupo.

```
JXTA>mygroupadv = newpgrp -n grupodeprueba
JXTA>join -r -d mygroupadv
# join - Starting rdv
mariano - Enter the identity you want to use for group
'grupodeprueba' :
Identity : Par_A
```

Ahora se utilizan nuevamente los comandos rdvstatus, join y peers. Estos indican que el Par A “mariano”, se está comportando como rendezvous y pertenece al grupo “grupodeprueba”

```
JXTA>rdvstatus

Rendezvous Status:
__________________

Current configuration : RENDEZVOUS
     "mariano" A  p  t  [25/25]

Peer View :
     [None]

Rendezvous Client Connections :
     [None]
```

```
JXTA>join
stdgroup       "grupodeprueba"     (current)
worldgroup     "World PeerGroup"
rootgroup      "NetPeerGroup"
env1           "grupodeprueba"     (current)
JXTA>peers
peer0: name = mariano
```

Se puede usar el comando whoami para obtener información.
Sin parámetros whoami muestra información del par.
Se observa el nombre del par, una descripción, la ID del par y las direcciones IP de las interfaces de red instaladas en el Par A, que recordemos actúa como gateway dentro de una LAN y por eso su IP es 192.168.0.1 y por otro lado la dirección IP asignada por el ISP para salir a internet es 201.231.39.32.

```
JXTA>whoami
<Peer>mariano</Peer>
<Description>Platform Config Advertisement created by :
net.jxta.impl.peergroup.AutomaticConfigurator</Description>
<PeerId>urn:jxta:uuid-
59616261646162614A78746150325033E43F59A9702547CFB85589E5A3CDF6FF03
</PeerId>
<TransportAddress>cbjx://uuid-
59616261646162614A78746150325033E43F59A9702547CFB85589E5A3CDF6FF03
</TransportAddress>
<TransportAddress>jxtatls://uuid-
59616261646162614A78746150325033E43F59A9702547CFB85589E5A3CDF6FF03
</TransportAddress>
<TransportAddress>relay://uuid-
59616261646162614A78746150325033E43F59A9702547CFB85589E5A3CDF6FF03
</TransportAddress>
<TransportAddress>tcp://192.168.0.1:2595</TransportAddress>
<TransportAddress>tcp://201.231.39.32:2595</TransportAddress>
```

Con el parámetro –g se muestra información del grupo. Nombre del grupo, una descripción y la ID de grupo.

```
JXTA>whoami -g
<PeerGroup>grupodeprueba</PeerGroup>
<Description>created by newpgrp</Description>
<PeerGroupId>urn:jxta:uuid-
114E3791016445A0831F95DCA9B54F7902</PeerGroupId>
```

Por otro lado en la segunda PC, que será el Par B, se registra el usuario “tito”.
Con el comando rdvstatus vemos que es un par extremo, con peers y join verificamos que el par “tito” pertenece al NetPeerGroup.

```
JXTA>peers
peer0: name = tito
peer1: name = JXTA.ORG 120:9711/9710
JXTA>join
rootgroup      "NetPeerGroup"     (current)
stdgroup       "NetPeerGroup"     (current)
worldgroup     "World PeerGroup"
JXTA>rdvstatus

Rendezvous Status:
__________________

Current configuration : EDGE
"tito" A  p  t  [52/41]

Peer View :
"JXTA.ORG 120:9711/9710" A  P  t  [36/36]

Rendezvous Connections :
JXTA.ORG 120:9711/9710 C : 81540 / -8460

Rendezvous Disconnections :
[None]
```

Se usa el comando whoami para obtener información.
Se observa el nombre del par, una descripción, la ID del par y la dirección IP de la interface de red instalada en el Par B 192.168.0.2.

```
JXTA>whoami
<Peer>tito</Peer>
<Description>Platform Config Advertisement created by :
net.jxta.impl.peergroup.AutomaticConfigurator</Description>
<PeerId>urn:jxta:uuid-
59616261646162614A787461503250335BE7F06A1B854AFFBD0DEAFC7E91EC9B03
</PeerId>
<TransportAddress>cbjx://uuid-
59616261646162614A787461503250335BE7F06A1B854AFFBD0DEAFC7E91EC9B03
</TransportAddress>
<TransportAddress>jxtatls://uuid-
59616261646162614A787461503250335BE7F06A1B854AFFBD0DEAFC7E91EC9B03
</TransportAddress>
<TransportAddress>relay://uuid-
59616261646162614A787461503250335BE7F06A1B854AFFBD0DEAFC7E91EC9B03
</TransportAddress>
<TransportAddress>tcp://192.168.0.2:1026</TransportAddress>
```

Usando el comando groups –r se envía un solicitud de búsqueda de grupos. Luego de unos instantes con el comando groups sin argumentos se listan los grupos encontrados.

```
JXTA>groups -r
# groups - Discovery message sent.
JXTA>groups
group0: name = NetPeerGroup
group1: name = SysVol100
group2: name = grupodeprueba
group3: name = MyJXTA
group4: name = com.javadojo.xptnl.Activator
group5: name = snscms
group6: name = AcornGroup
group7: name = snsfs
group8: name = mygroup10
group9: name =
$GROUP$lindy7981\u7684\u7B2C\u4E00\u4E2A\u793E\u7F51$/GROUP$
group10: name = $GROUP$SNS\u8BBA\u575B$/GROUP$
```

Se identifica que el grupo 2 de la lista es el grupo creado por el Par A, llamado “grupodeprueba”, entonces el Par B se une a dicho grupo con el comando join y el modificador –d.

```
JXTA>join -d group2
tito - Enter the identity you want to use for group
'grupodeprueba' :
Identity : Par_B
```

Si ahora usamos el comando peers se verán los dos pares que pertenecen al grupo.

```
JXTA>peers
peer0: name = tito
peer1: name = mariano
```

Recordemos que el Par A “mariano” estaba actuando como rendezvous. Entonces si ejecutamos el comando rdvstatus en el par B, se ve como dicho Par B “tito”, utiliza al Par A “mariano” como rendezvous.

```
JXTA>rdvstatus

Rendezvous Status:
__________________

Current configuration : EDGE
     "tito" A  p  t  [391/382]

Peer View :
     "mariano" A  P  t  [380/22]

Rendezvous Connections :
     mariano C : 820790 / 520790

Rendezvous Disconnections :
     [None]
```

Una vez que los pares están en el mismo grupo pueden hablar entre ellos con el comando talk. Este comando implementa una mensajería simple de línea de comando, para que dos usuarios en dos pares remotos puedan intercambiar mensajes. Estos mensajes se visualizan en la consola del shell.
Para usar talk, los usuarios deben estar registrados e implementar un pipe para escuchar, ésto se hace siguiendo una serie de pasos:

1. Registrarse utilizando el comando: talk –register <nombre de usuario>
Este comando crea un anuncio de talk para ese usuario. Esto debe hacerse únicamente una vez, la primera vez que el usuario se registra con talk en un grupo específico. Se puede agregar la opción –secure para establecer una sesión de talk segura y –propagate para establecer una sesión de talk al estilo chat room.

2. Loguearse utilizando el comando: talk –login <nombre de usuario>
Este comando loguea al usuario e inicializa un demonio que escucha. Este comando debe usarse cada vez que el par se reinicia.

3. Iniciar la conversación con otro usuario destino usando el comando:
talk –u <mi nombre de usuario> <nombre de usuario destino>
Luego de usar este comando se puede empezar a enviar mensajes.

En el Par A “mariano” se procede a registrarse, creando un pipe unicast y luego a loguearse

```
JXTA>talk -register mariano
# talk - Creating pipe named :mariano of type :JxtaUnicast
.....................
# talk - User 'mariano' is now registered
JXTA>talk -login mariano
```

Si se usa el comando talk con la opción –search se busca usuarios para hablar.

```
JXTA>talk -search
.....................
# talk - Found the following talk registrations:
mariano
```

Como solamente está registrado el usuario “mariano” no podrá inicializarse una conversación con “tito” porque éste último no está registrado.

```
JXTA>talk -u mariano tito
.....................
# talk - User 'tito' is not a registered.
```

En el Par B el usuario “tito” procede a registrarse, creando un pipe unicast y luego a loguearse.

```
JXTA>talk -register tito
# talk - Creating pipe named :tito of type :JxtaUnicast
.....................
# talk - User 'tito' is now registered
```

```
JXTA>talk -login tito
```

Si ahora el Par A “mariano” hace nuevamente una búsqueda de talk encontrará al Par B “tito”.

```
JXTA>talk -search
.....................
# talk - Found the following talk registrations:
mariano
tito
```

Ya es posible iniciar la conversación, para ello se utiliza el comando talk –u seguido del nombre de usuario del par que inicia la conversación, seguido del par de destino. Luego se ve como se encuentra el anuncio de talk del Par B y se implementa la conexión. A continuación ya se puede ingresar el mensaje y enviarlo con enter. Observar que el prompt JXTA> desaparece.

```
JXTA>talk -u mariano tito
# talk - Found advertisement for 'tito'. Attempting to connect
# talk - Connection established to user tito
# talk - Type your message. To exit, type "." at beginning of line
Hola tito, como estas!
```

En el Par B se recibirá el mensaje y se visualizará como una línea en el shell JXTA de la siguiente manera:

```
# talk - mariano to tito> Hola tito, como estas!
```

El usuario del Par B puede contestarle. De esta manera se estable la comunicación en ambos sentidos. Del Par A al Par B y del Par B al Par A.

```
JXTA>talk -u tito mariano
.# talk - Found advertisement for 'mariano'. Attempting to connect
# talk - Connection established to user mariano
# talk - Type your message. To exit, type "." at beginning of line
muy bien, y vos?
```

Para dejar de enviar mensajes se sale escribiendo un punto “.” al principio de la línea.
Para dejar de recibir más mensajes, el usuario puede detener el demonio que escucha conversaciones (talk) con el comando: talk -logout <nombre de usuario>

Para dejar un grupo se usa el comando leave y para salir del shell se usa el comando exit.

De la misma manera que se uso talk para enviar mensajes entre pares se puede usar sftp para enviar archivos.

```
JXTA>sftp -register mariano
......
User : mariano is now registered
JXTA>sftp -login mariano
JXTA>sftp -search
......
found the following registrations:
JxtaSftpUserName.mariano
JxtaSftpUserName.tito
JXTA>sftp -s mariano tito /temp/imagen.jpg
found user's advertisement attempting to connect
Please be patient ...
sftp is connected to user tito
Sending file /temp/imagen.jpg, size = 36303 bytes in 3 chunks
!!!
Sent: 36303 bytes in 0.01 secs[3630Kbytes/sec]
```