3 JXTA

3.1 INTRODUCCIÓN

3.1.1 QUÉ ES JXTA

JXTA es un conjunto de protocolos P2P abiertos y de propósito general, que permiten a cualquier dispositivo conectado a la red comunicarse y actuar como *peers*. Los protocolos JXTA son independientes del lenguaje de programación, y existen múltiples implementaciones para distintos entornos. Los protocolos son también independientes de los protocolos de transporte usados, por lo que pueden ser implementados sobre TCP/IP, HTTP, Bluetooth, etc.

Estos protocolos estandarizan la forma en que los *peers*:

- Se descubren unos a otros.
- Se auto-organizan en grupos.
- Publican y descubren servicios.
- Se comunican con otros.
- Se monitorizan unos a otros.

El Proyecto JXTA proporciona una implementación de referencia de estos protocolos en código abierto (*open source*), disponible en la web del proyecto www.jxta.org. El término JXTA es una abreviatura de "*juxtaposed*" (yuxtapuesto).

3.1.2 OBJETIVOS DE JXTA

El objetivo principal de JXTA es proporcionar una plataforma con las funciones básicas y necesarias para una red P2P. Además, JXTA busca:

- **Interoperabilidad**: la tecnología JXTA está diseñada para que *peers* muy heterogéneos puedan interactuar con otros mediante servicios también muy heterogéneos.
- **Independencia de la plataforma**: JXTA no depende del lenguaje de programación usado, ni del protocolo de transporte, ni de la plataforma.
- **Ubicuidad**: la tecnología JXTA está diseñada para que cualquier tipo de dispositivo digital pueda implementar JXTA.

JXTA crea una red virtual donde los *peers* interactúan entre ellos directamente, aunque estén detrás de barreras como *firewalls*: JXTA proporciona mecanismos para sortear estos obstáculos. Por eso, se dice que JXTA proporciona una red *overlay*, como si estuviera encima de todo el confuso y desordenado Internet.

3.1.3 HISTORIA DE JXTA

JXTA nació en el seno de Sun Microsystems como un proyecto de investigación de la tecnología P2P. Desde el principio, se formó una comunidad de desarrollo interesada en la tecnología. En abril de 2001, se abre la web del proyecto JXTA, incluyendo las especificaciones de los protocolos y animando a desarrolladores de todo el mundo a unirse en los esfuerzos por implementar nuevos servicios y plataformas con la nueva tecnología. La comunidad JXTA es ahora la que tiene la responsabilidad de desarrollar el proyecto.

JXTA está fuertemente apoyado por Sun, y se han unido al apoyo numerosas universidades y empresas privadas de software.

Actualmente, multitud de proyectos están siendo llevados a cabo dentro del ámbito de la comunidad JXTA, desde las implementaciones en lenguajes como Python hasta motores de búsqueda en Internet.

3.2 ARQUITECTURA DE JXTA

3.2.1 MODELO DEL SOFTWARE JXTA

La arquitectura *software* del proyecto JXTA está dividido en 3 capas, mostrado en la figura siguiente:

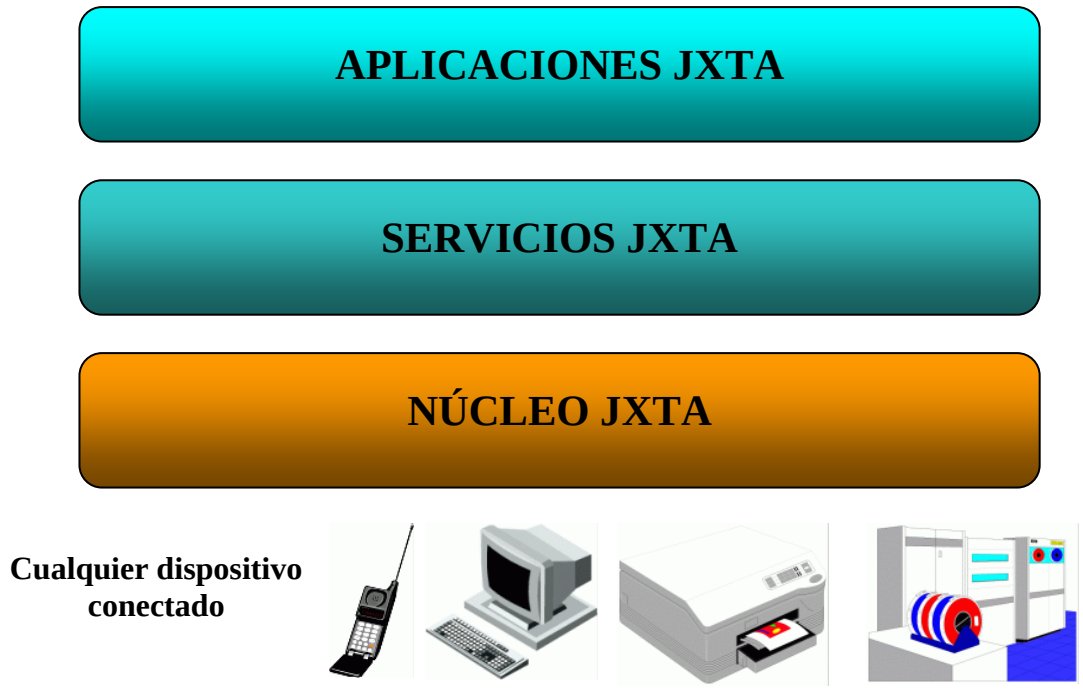

Figura 3-1: Arquitectura SW de JXTA

- **Núcleo JXTA**: también conocida como la capa plataforma, encapsula las primitivas esenciales en una red P2P. Proporciona los bloques básicos para la

construcción de aplicaciones P2P, como las primitivas de transporte, creación de *peers* y grupos, seguridad, descubrimiento.

- **Servicios JXTA**: Son los servicios que no tienen porqué ser totalmente necesarios para el correcto funcionamiento de la red P2P, pero sí son útiles. Ejemplos de servicios JXTA son: búsqueda, directorio, almacenamiento, compartición de ficheros, sistemas de fichero distribuidos, autentificación, traducción de protocolos, servicios de clave pública, etc.
- **Aplicaciones JXTA**: son las implementaciones de las aplicaciones en sí, como por ejemplo: mensajería instantánea, sistemas de correo electrónico, subastas distribuidas, compartición de documentos, etc.

3.2.2 COMPONENTES JXTA

La arquitectura de JXTA es muy sencilla. Se puede resumir en:

- La red JXTA Consiste en una serie de nodos interconectados, llamados *peers*, que proporcionan una serie de servicios.
- Los *peers* JXTA publican sus servicios en documentos llamados anuncios, que describen los recursos y la forma de interactuar con los servicios. Los otros *peers* podrán descubrir esos anuncios y, por tanto, podrán usar el servicio.
- Los *peers* JXTA usan sockets y tuberías (pipes) para enviar mensajes a otros *peers*. Los sockets son conexiones seguras bi-direccionales usadas por las aplicaciones para comunicaciones seguras. Los sockets están construidos sobre las tuberías. Las tuberías son mecanismos de transporte asíncronos y unidireccionales de mensajes. Las tuberías están ligadas a puntos extremos específicos, como una dirección IP o un puerto TCP.
- Los anuncios y mensajes usados en JXTA son documentos XML sencillos.

Los aspectos clave de la arquitectura JXTA son:

- El uso de documentos XML para describir recursos y mensajes.
- Abstracción de sockets y tuberías en *peers*, y de *peers* en puntos extremos, sin recurrir a ningún mecanismo de nombrado y direccionamiento centralizado (como por ejemplo DNS).
- Un esquema uniforme de direccionamiento de los *peers* mediante identificadores de *peer*.

3.3 TERMINOLOGÍA DE JXTA

3.3.1 PEERS

Un *peer* es cualquier dispositivo conectado (desde ordenadores hasta sensores) que implementa uno o más protocolos JXTA. Cada *peer* opera independiente y asíncronamente del resto y está únicamente identificado por un identificador de *peer*.

No es necesaria una conexión punto a punto entre *peers*, sino que *peers* intermediarios específicos pueden ser usados para el encaminamiento.

Los *peers* publican anuncios que describen sus servicios, identificados por un punto extremo de *peer*, y usados para establecer conexiones punto a punto entre dos *peers*.

3.3.2 GRUPOS DE PEERS

Un grupo de *peers* es un conjunto de *peers* que se han puesto de acuerdo para prestarse un conjunto de servicios comunes. Están identificados por un identificador de grupo único. Los grupos en JXTA admiten relaciones de jerarquía, pero con un sólo grupo padre.

Un *peer* puede pertenecer a más de un grupo simultáneamente. El grupo por defecto al que todos están conectados es el *Net Peer Group*, a través del cual los *peers* se unen a los otros grupos. Un grupo puede tener políticas de acceso, desde abierto hasta protegido.

Los motivos para crear un grupo son varios:

- **Para crear un entorno seguro**: gracias a que los grupos pueden implementar políticas de membresía, desde las sencillas usuario/clave hasta criptografía de clave pública. De esta manera, sólo los miembros pueden publicar y acceder a los servicios.
- **Para crear un entorno especializado**: así, los *peers* con ciertos intereses pueden compartirlos sin ser interferidos. Por ejemplo, un conjunto de *peers* que implementen un servicio de compartición de archivos.
- **Para crear un entorno de monitorización**: de esta forma se pueden controlar ciertos aspectos del grupo como el volumen de tráfico, estadísticas, etc.

Un grupo proporciona un conjunto de servicios de grupo. JXTA define un conjunto denominado núcleo de servicios de grupo, aparte de los que se puedan implementar. Para buscar y usar un servicio de un grupo, el *peer* debe pertenecer a dicho grupo. El núcleo de servicios de grupo definidos por JXTA son:

- **Servicio de Descubrimiento**: es usado por los *peers* para descubrir y publicar recursos en el grupo, como *peers*, grupos, tuberías, etc.
- **Servicio de Membresía**: es usado para aceptar o rechazar solicitudes de admisión en el grupo. Un *peer* que quiera entrar en el grupo debe localizar a un *peer* perteneciente a ese grupo y pedir la admisión, y éste decidirá si es admitido o no. Incluso se pueden generar votaciones o representantes.
- **Servicio de Acceso**: se usa para validar las peticiones hechas de un *peer* a otro. En una petición viajan los credenciales del *peer* que realiza la petición para que el extremo pueda verificar si tiene los permisos adecuados para ejecutar la operación.

- **Servicio de Tubería**: se usa para crear y gestionar conexiones a través de tuberías entre los miembros de un grupo.
- **Servicio de Resolución**: es usado para enviar peticiones de propósito general a otros *peers*.
- **Servicio de Monitorización**: usado para que un *peer* pueda monitorizar al resto del grupo.

No todos los servicios tienen que ser implementados por un grupo: se pueden usar los servicios del *Net Peer Group* para tareas menos críticas.

3.3.3 SERVICIOS DE RED

Los servicios de red son publicados, descubiertos y usados por los *peers*.

Se dividen en dos grupos:

- **Servicios de peer**: sólo están disponibles en el *peer* que publica el servicio. Si el *peer* falla, el servicio falla también. Puede haber múltiples *peers* ejecutando el mismo servicio, pero cada uno publica su propio anuncio.
- **Servicios de grupo**: está compuesto por un conjunto de instancias del servicio ejecutándose en varios miembros de un grupo. Si un *peer* falla, el servicio sigue estando disponible (siempre que quede algún *peer*). Las instancias pueden cooperar entre ellas. Los servicios de grupo se publican en el anuncio de grupo.

Los servicios de red pueden ser:

- Implementados directamente en el código de la aplicación.
- Descargados desde la red, como si fuera un *plug-in* para una página web.

3.3.4 MÓDULOS

Son abstracciones usadas para representar comportamientos implementados en código. Un servicio de red es un ejemplo de comportamiento que puede ser instanciado en un *peer*. El módulo no especifica qué código es: puede ser una clase Java, o librerías DLL, etc. Los módulos pueden ser usados para representar distintas implementaciones de un servicio de red en distintas plataformas.

Existen 3 tipos de módulos:

- **Clase de Módulo**: anuncia la existencia de un comportamiento simplemente. Está identificado por un identificador único.
- **Especificación de módulo**: se usa para acceder a un módulo. Contiene toda la información necesaria para la invocación del módulo. Por ejemplo, en el caso de un servicio, contendría el anuncio de tubería que se usará para comunicarse con el servicio.

  Está determinado por un único identificador, aunque pueden existir distintas especificaciones de módulo para una misma clase de módulo.

Se usa para asegurar la compatibilidad de red: todas las implementaciones de una especificación de módulo deben cumplir con los mismos protocolos y ser compatibles, aunque estén escritos en distintos lenguajes.

- **Implementación de Módulo**: es la implementación en sí de una especificación de módulo determinada. Puede haber varias implementaciones para una misma especificación de módulo. Esta determinado por un identificador único.

Los módulos pueden ser usados por servicios de grupo o de *peer*. Los servicios JXTA pueden usar un módulo para identificar la existencia del servicio (mediante la clase de módulo), la especificación del servicio (mediante la especificación de módulo) y/o la implementación del servicio (mediante la implementación de módulo). Cada uno de estos componentes tiene asociado un anuncio, que será publicado y descubierto por los *peers* del grupo.

3.3.5 TUBERÍAS (PIPES)

Los *peers* en JXTA usan tuberías para enviar mensajes a otros *peers*. Es un mecanismo asíncrono y unidireccional, y soportan cualquier tipo de objeto en el transporte, como cadenas de texto, binarias, objetos Java, etc.

Los puntos extremos de la tubería son llamados tubería de entrada (*Input pipe*) y tubería de salida (*Output pipe*), y son asociadas dinámicamente a los puntos extremos del *peer* (que serán direcciones IP y puertos TCP por ejemplo) en tiempo de ejecución.

Las tuberías son canales de comunicación virtuales, por lo que pueden unir *peers* que no tienen conexión física directa. Entonces, puntos extremos intermediarios se usarán para encaminar los mensajes entre los puntos extremos de la tubería.

Existen 2 tipos de tuberías, más una tercera añadida por el núcleo de JXTA:

- **Tuberías punto a punto**: conecta dos puntos extremos, de forma que un *peer* recibe por la tubería de entrada lo que otro envía por la de salida.
- **Tuberías de propagación**: conecta una tubería de salida con múltiples tuberías de entrada. La propagación se realiza sólo dentro del grupo.
- **Tuberías unidireccionales seguras**: es una tubería punto a punto que proporciona un canal de comunicación seguro. Está definido en el núcleo de JXTA.

Tuberías Unidireccionales son abstracciones de comunicaciones a muy bajo nivel de JXTA. Es recomendado para la mayoría de desarrolladores usar abstracciones de comunicaciones a un alto nivel. Esto es proporcionado por JxtaSocket y JxtaBidiPipe descrito en la siguiente sección.

3.3.6 JXTASOCKET AND JXTABIDIPIPE

(Canales de comunicación Seguros Bidireccional)

Las tuberías básicas JXTA proporcionan canales de comunicación unidireccionales e inseguros. Para hacer las tuberías más útiles a las aplicaciones, es necesario implementar canales bidireccionales y seguros sobre las primitivas tuberías (pipes). JXSE proporciona la funcionalidad requerida para la calidad del nivel del servicio requerido para la mayoría de las aplicaciones:

- Fiabilidad
- Asegurar la secuenciación del mensaje
- Asegurar la entrega
- Expone interfaces para mensajes y streams
- Seguridad

➢ **JxtaSocket y JxtaServerSocket:**

- Sub-class java.net.Socket y java.net.ServerSocket respectivamente.
- Están construidos sobre tuberías (pipes), puntos extremos (endpoint messenger), y biblioteca segura.
- Proporciona canales de comunicaciones bidireccionales, fiables y seguros.
- Expone interface basada en Streams
- Proporciona “buffering” interno configurable y segmentación del mensaje.
- No implementa el algoritmo de Nagle, además los streams deben ser vaciados cuando sea necesario.

➢ **JxtaBiDiPipe y JxtaServerPipe:**

- Están construidos sobre tuberías (pipes), puntos extremos (endpoint messenger), y biblioteca segura.
- Proporciona canales de comunicaciones bidireccionales, fiables y seguros.
- Expone interface basada en mensajes
- No proporciona segmentación del mensaje. Las aplicaciones necesitan asegurar que el tamaño del mensaje no excede la limitación del tamaño estándar de 64K.

JxtaServerSocket y JxtaServerPipe concede una tubería de entrada a los procesos de petición de conexión y negocia los parámetros de la comunicación. JxtaSocket y JxtaBiDiPipe, por otro lado, están ligados a sus respectivas tuberías privadas dedicadas independientes de la tubería de petición de conexión.

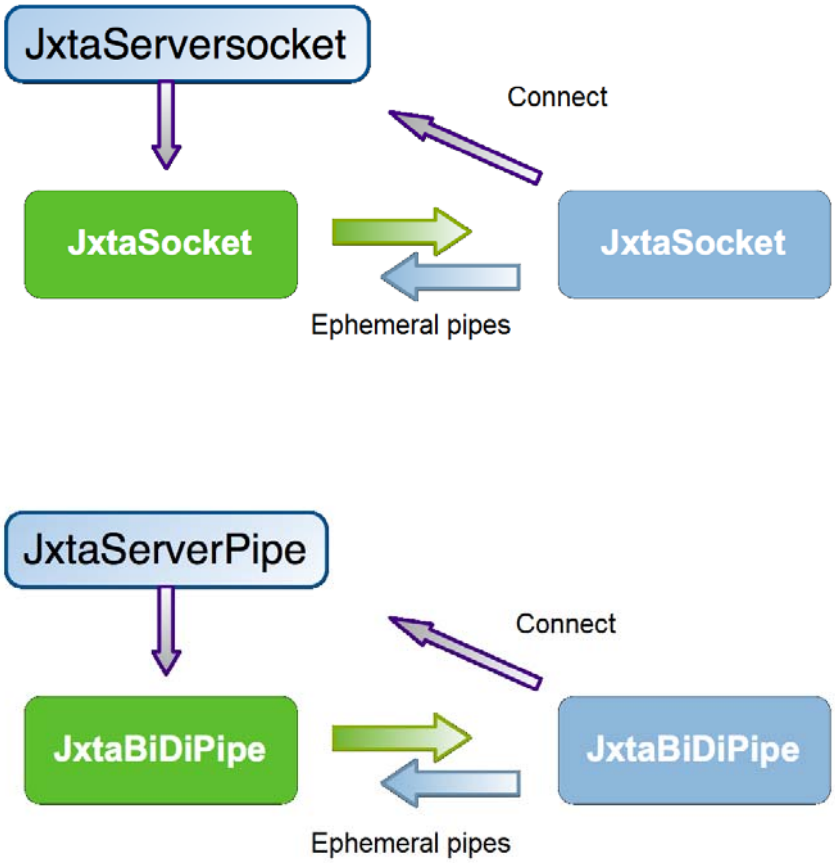

Figura 3-2: Conexiones Pipes

3.3.7 MENSAJES

Es un objeto enviado entre *peers* JXTA, y representa la unidad básica de intercambio de datos entre *peers*. Son enviados por los servicios de Tubería y de punto extremo. Un mensaje consiste en una secuencia de contenidos nombrados y tipados, denominados elementos de mensaje. El contenido puede ser de tipo arbitrario.

Los mensajes pueden estar representados en XML o de forma binaria. La representación en XML facilita la interacción de todos los *peers*, y la cómoda identificación de partes de mensaje.

Los protocolos JXTA están especificados como un conjunto de mensajes que se intercambiarán los *peers*.

3.3.8 ANUNCIOS

Todo recurso en una red JXTA (*peers*, grupos, tuberías y servicios) está representado mediante un anuncio. Están expresados en XML, por lo que son independientes del lenguaje. Para anunciar en la red la existencia de un recurso

determinado, hay que publicar su anuncio, y puede usarse un cierto recurso si se descubre su anuncio asociado. Los anuncios descubiertos pueden guardarse en un caché.

Los anuncios son publicados con un tiempo de vida determinado, evitando así el uso de un control centralizado de publicación/borrado de recursos.

Existen varios tipos de anuncio:

- **Anuncio de peer**: contiene información básica del *peer* como su nombre, identificador, puntos extremos disponibles, etc.
- **Anuncio de grupo**: contiene información básica del grupo como su nombre, descripción, identificador, especificación y parámetros del servicio.
- **Anuncio de tubería**: describe una tubería, con el nombre, identificador y tipo. Es usado por el servicio de tubería para crear las tuberías de entrada y salida.
- **Anuncio de clase de módulo**: describe una clase de módulo, con un nombre, identificador, etc.
- **Anuncio de especificación de módulo**: describe una especificación de módulo, con el nombre, identificador, etc. También puede contener un anuncio de tubería.
- **Anuncio de implementación de módulo**: describe una implementación de módulo, con el nombre, identificador, código, paquetes, etc.
- **Anuncio de *rendezvous***: describe a un *peer* que realiza labores de *rendezvous*.
- **Anuncio de información de peer**: contiene información más específica sobre un *peer*.

3.3.9 IDENTIFICADORES

Identifican unívocamente a una entidad, como *peers*, grupos, tuberías, contenidos, clases de módulos y especificaciones de módulos.

Una URN (*Uniform Resource Name*) es una forma de expresión de un identificador JXTA. Es una forma de URI presentada en texto, de esta manera:

```
urn:jxta:uuid-537469238742374234 87E65JK8686978J88U98986704
```

3.3.10 SEGURIDAD

Se deben proporcionar 5 requerimientos de seguridad:

- **Confidencialidad**
- **Autenticación**
- **Autorización**
- **Integridad de datos**
- **Refutabilidad**

Los mensajes, al ser XML, permiten la inserción de credenciales, certificados, claves públicas, etc. por lo que JXTA puede cumplir los cinco requerimientos antes mencionados: los credenciales proporcionan autorización y autenticación, la encriptación proporciona confidencialidad y refutabilidad, y la redundancia proporciona la integridad de los datos.

3.4 ASPECTOS DE RED

3.4.1 ORGANIZACIÓN DE LA RED JXTA

La red JXTA es una red ad-hoc, multisalto y adaptativa compuesta por *peers* conectados. Las conexiones en la red son generalmente transitorias, y las rutas entre *peers* no deterministas. En cualquier momento, un *peer* puede unirse o dejar la red, por lo que las rutas cambian frecuentemente. Además, un *peer* puede asumir en cualquier momento algunas funciones extra y necesarias para el buen funcionamiento de la red.

Aunque la organización de la red no está impuesta por el marco de desarrollo de JXTA, en la práctica se usan 4 tipos de *peers*:

- **Peer *mínimo (edge peer)***: puede enviar y recibir mensajes, pero no almacena ningún anuncio descubierto ni encamina hacia otros *peers*. Son *peers* con limitados recursos, como una PDA o teléfono móvil.
- **Peer *completo (edge peer)***: puede enviar y recibir mensajes, almacenar los anuncios descubiertos en una memoria caché y responder a peticiones de descubrimiento con la información almacenada. Sin embargo, no propaga ninguna petición. La mayoría de los *peers* en una red JXTA son de este tipo.
- **Peer *rendezvous***: funciona como un *peer* completo, pero sí propaga además las peticiones de búsqueda (descubrimiento) para ayudar a otros *peers* a encontrar ciertos recursos en la red. Cuando un *peer* entra en un grupo, automáticamente busca a un *peer rendezvous*. Si no encuentra ninguno, automáticamente el *peer* asume ese papel dentro de ese grupo. Un *peer rendezvous* mantiene 2 listas:
    - Otros *peers rendezvous* conocidos.
    - Los *peers* del grupo que están usando a este *peer* como *peer rendezvous*.

  Los *peers* "normales" envían peticiones de búsqueda a los *peers rendezvous*, y éstos propagan el mensaje a otros *peers rendezvous* si no tienen la información para responder. El proceso continúa hasta que un *peer* tiene la información pedida o expira el TTL del mensaje, que por defecto son 7 saltos.
- **Peer *relay***: mantiene información de encaminamiento hacia otros *peers*, y encamina mensajes. Cuando un *peer* envía un mensaje, mira en su caché la ruta, si no la tiene pregunta a su *peer relay*. También propagan mensajes en nombre de *peers* que no pueden acceder a otros, uniendo redes físicamente y/o lógicamente diferentes.

Un *peer* puede actuar como *rendezvous*, *relay* o incluso los dos a la vez.

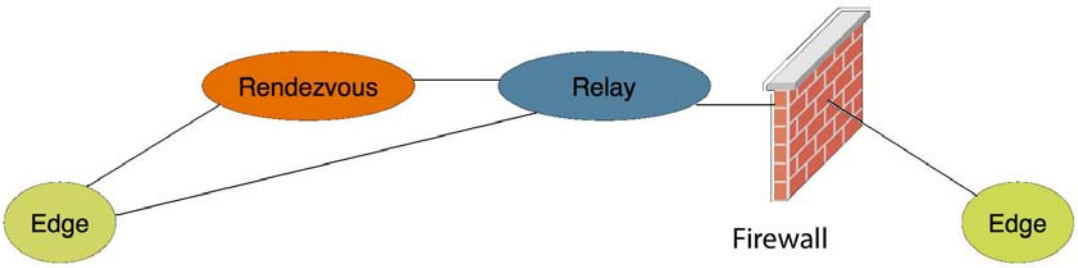

Figura 3-3: Organización de la red

3.4.2 FIREWALLS, NAT, PROXIES

Un *peer* situado detrás de un *firewall* puede enviar un mensaje fuera del *firewall*, pero un *peer* exterior no podría establecer una conexión con uno interior. Entonces, la red *peer*-to-*peer* no funcionaría más que en el entorno de red local. Sin embargo, la especificación de protocolos JXTA proporciona mecanismos para poder realizar la conexión a través de *firewall*, pero deben darse las 3 condiciones siguientes:

- Al menos un *peer* del grupo situado dentro del *firewall* debe conocer la existencia de un *peer* situado fuera del *firewall*.
- Los *peers* situados dentro y fuera del *firewall* que se quieran comunicar deben conocer la existencia del otro y soportar HTTP.
- El *firewall* debe admitir transferencias HTTP (no tiene porqué estar limitado al puerto 80).

Cumplidas estas condiciones, JXTA define la forma de sortear el dispositivo barrera: un *peer* actuará como intermediario (será un *peer relay*), con el que el *peer* interior a la red se conectará vía HTTP, estableciendo posteriormente una conexión TCP/IP con el *peer* exterior a la red. De esta forma, se pueden sortear barreras que de otro modo son infranqueables, y que impedirían la comunicación entre los *peers* de un grupo en JXTA. En la siguiente figura se muestra el esquema de conexiones:

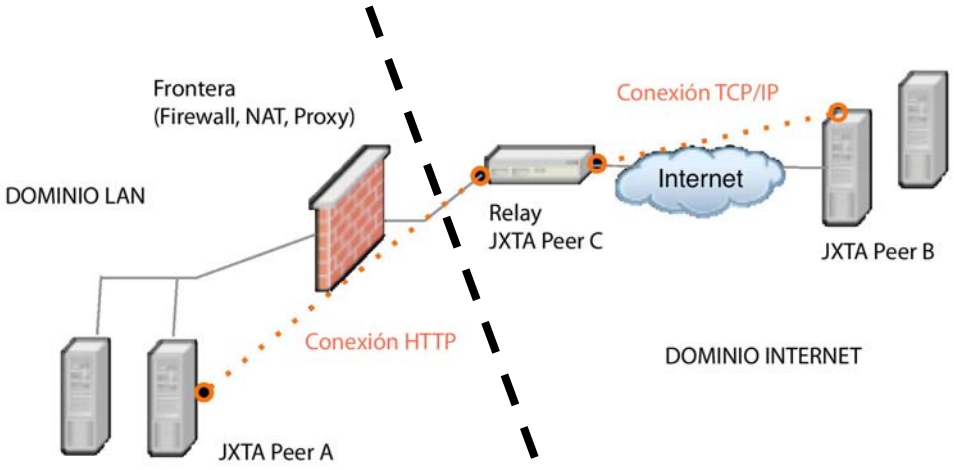

Figura 3-4: Ejemplo de participación de un *peer relay* ante una barrera y encaminamiento de mensajes a través de un firewall.

De esta manera, mediante una conexión HTTP se puede penetrar el *firewall*, y se usa un *peer* intermediario para realizar la comunicación.

B y A quieren comunicarse, pero el *firewall* no lo permite. A establece una conexión HTTP con un *peer* C conocido, por lo que el *firewall* es traspasado, y C conecta con B mediante una conexión TCP/IP por ejemplo. Se ha establecido una conexión virtual entre A y B.

En caso de un dispositivo NAT (*Network Address Translator*), la situación es muy parecida: como JXTA maneja dentro de los mensajes direcciones y puertos, los mensajes recibidos dentro de la red protegida por NAT no contendrían información verídica sobre los verdaderos puertos. Además, algunos NAT actúan como *firewall* en un solo sentido, complicando aún más la comunicación entre el interior y el exterior de la subred. Pero gracias al mecanismo de la conexión a un tercero (un *peer relay*) mediante HTTP, esta situación se resuelve sin problemas.

El mismo problema se presenta con un servidor *proxy*. En la mayoría de los casos, se puede atravesar el proxy sin problemas, pero en otros no se garantiza que JXTA sea capaz de atravesarlo sin previa autorización del administrador de la red.

3.5 PROTOCOLOS JXTA

JXTA define un conjunto de formatos de mensajes, denominados protocolos, para la comunicación entre *peers*. Los *peers* usan estos protocolos para todas las tareas relacionadas con la plataforma *peer*-to-*peer*.

Los protocolos JXTA son asíncronos y están basados en modelos petición/respuesta. Un *peer* JXTA usa un protocolo para enviar una petición a uno o más *peers* de su grupo, y puede recibir cero, una o más respuestas a su petición. No es obligatorio que un *peer* implemente todos los protocolos, sino sólo los que vayan a usar.

JXTA define 6 protocolos:

- ***Peer Discovery Protocol (PDP)***: se usa para descubrir o anunciar recursos asociados a un *peer*, mediante el descubrimiento o publicación de un anuncio que describe el recurso.
- ***Peer Information Protocol (PIP)***: se usa para obtener información de otros *peers*.
- ***Peer Resolver Protocol (PRP)***: se usa para enviar peticiones genéricas a uno o más *peers*, y recibir una o más respuestas a esa petición.
- ***Pipe Binding Protocol (PBP)***: se usa para establecer un canal virtual de comunicación entre uno o más *peers*.
- ***Endpoint Routing Protocol (ERP)***: se usa para encontrar rutas a los puertos de destino de otros *peers*.

- ***Rendezvous Protocol (RVP)***: es un mecanismo para el registro de *peers* a un servicio de propagación.

PRP y ERP están definidos como protocolos del núcleo de JXTA. Los restantes son protocolos estándar.

3.5.1 PEER DISCOVERY PROTOCOL (PDP)

Los recursos en JXTA como un *peer*, un grupo, un servicio, tubería, etc. se representan mediante anuncios, que son documentos expresados en XML con información que describe el recurso. Estos anuncios deben ser publicados en la red JXTA para su descubrimiento por otros *peers*: de esta tarea se encarga el protocolo PDP. Es el protocolo por defecto de todos los grupos definidos por usuarios y por el grupo principal de JXTA (*Net Peer Group*). Si un grupo que no sea el principal no tiene implementado ningún protocolo de descubrimiento propio, se usa PDP para descubrir los anuncios de otros *peers*. El PDP permite la búsqueda y publicación de anuncios sólo en el grupo donde esté implementado.

PDP se usa para proveer una infraestructura básica de descubrimiento para los servicios JXTA de más nivel, de forma que estos niveles más altos, incluido el nivel de aplicación, puedan participar directamente en el proceso de búsqueda. Los servicios JXTA deberían mejorar el servicio básico de búsqueda PDP, por ejemplo, filtrando los anuncios descubiertos que no son de utilidad para el servicio.

PDP usa el protocolo PRP para encaminar peticiones y respuestas.

3.5.2 PEER INFORMATION PROTOCOL (PIP)

Una vez que un *peer* esté localizado, puede ser útil obtener sus capacidades y estado. El protocolo PIP proporciona un conjunto de mensajes para obtener información de un *peer* concreto, aunque no es obligatorio que el *peer* responda a peticiones PIP, ya que PIP es un protocolo JXTA opcional. Está montado sobre el protocolo PRP.

El mensaje de petición de PIP se envía para comprobar si un *peer* está conectado y para obtener información sobre el *peer*. El mensaje especifica si se desea recibir una respuesta completa (un anuncio de *peer*) o asentimiento simple (si está conectado y el tiempo transcurrido).

Este protocolo es útil para la obtención de información sobre tareas de administración y monitorización de *peers*, servicios comerciales (destinados a facturar por tiempo de conexión por ejemplo), etc.

3.5.3 PEER RESOLVER PROTOCOL (PRP)

Este protocolo permite el envío de peticiones genéricas a otros *peers* e identificar las respuestas concordantes a la petición enviada. Las peticiones pueden ser enviadas a un *peer* específico, o propagadas dentro de los límites de un grupo de *peers* mediante el uso del servicio *rendezvous*. PRP usa el servicio *rendezvous* para diseminar una petición destinada a múltiples *peers*, y usa mensajes *unicast* para enviar una petición a un solo *peer*.

Los mensajes de petición y respuesta PRP contienen varios campos: el credencial del *peer* que envía el mensaje, un identificador único de mensaje, un manejador de servicio específico, y el cuerpo del mensaje en sí (petición o respuesta). Cada servicio puede registrar un manejador en el servicio de resolución del grupo para procesar peticiones PRP y generar las respuestas.

PRP es un protocolo clave en JXTA, debido a que soporta peticiones genéricas. Es usado por otros protocolos como PIP y PDP como base.

PRP también se usa en el Índice Distribuido de Recursos Compartidos (SRDI), que proporciona un mecanismo general donde los servicios JXTA pueden usar un índice distribuido de recursos compartidos con otros *peers* agrupados en un conjunto de *peers* más capacitados para el servicio (por ejemplo, un *peer rendezvous*). Estos índices, y a través de peticiones PRP de contenido SRDI, son utilizados para dirigir peticiones en la dirección donde es más probable que sea contestada, o redirigir hacia *peers* que están interesados en recibir dichas peticiones.

3.5.4 PIPE BINDING PROTOCOL (PBP)

Este protocolo es usado por miembros de un grupo de *peers* para enlazar un anuncio de tubería con un punto extremo de la tubería. El enlace virtual de tubería puede ser montado sobre un gran número de protocolos de transporte, como TCP/IP. Cada extremo de la tubería tiene la tarea de mantener y restablecer el enlace virtual, e incluso si es necesario, buscar y enlazar con el actual punto extremo de la tubería.

Una tubería puede ser considerada como una cola de mensajes abstracta referenciada por un nombre, que implementa operaciones de creación, enlace, cierre, eliminación de la tubería, y envío y recepción de mensajes a través de ella. Las tuberías implementadas pueden ser muy distintas, pero todas usan PBP para enlazar la tubería al punto extremo.

Una petición PBP es enviada por un punto extremo de tubería de un *peer* para encontrar otro punto extremo de tubería asignado al mismo anuncio de tubería. Es decir, la comunicación entre puntos extremos se hace a través del enlace de esos puntos extremos a un mismo anuncio de tubería. La petición puede pedir además información actualizada del *peer* que posee el punto extremo de destino.

La respuesta PBP es enviada a cada punto extremo asociado al mismo anuncio de tubería, y contiene información del *peer* destino que ha creado la tubería de entrada.

3.5.5 ENDPOINT ROUTING PROTOCOL (ERP)

Este protocolo define un conjunto de mensajes usados para obtener información de encaminamiento, necesaria para enviar mensajes de un *peer* origen a otro destino.

Cuando un *peer* quiere enviar un mensaje a un punto extremo, dado por su dirección, primero busca en el caché local para comprobar si tiene información de encaminamiento hacia el *peer* destino. Si no la tiene, envía una petición ERP a su *relay peer* pidiendo información de rutas, especificando si quiere como respuesta la información del caché del encaminador o información de una nueva búsqueda por parte de éste. Cuando el *relay peer* recibe una petición, comprueba si conoce la ruta, y

devuelve un mensaje de información de encaminamiento con una enumeración de saltos a otros relays que representan el camino hacia el destino, aunque la lista no tiene porqué estar completa. También se incluye en la respuesta un tiempo de vida (TTL) de la ruta.

Cualquier *peer* puede iniciar peticiones ERP de encaminamiento, y también convertirse en un *relay peer*, ya que generalmente sólo almacenan información de encaminamiento en caché.

3.5.6 RENDEZVOUS PROTOCOL (RVP)

Este protocolo es el responsable de propagar mensajes dentro de un grupo de *peers*, permitiendo:

- Propagar mensajes y recibir mensajes propagados.
- Controlar la propagación de mensajes, atendiendo a estos 3 aspectos:
    - Tiempo de vida (TTL).
    - Presencia de bucles.
    - Duplicados.

RVP introduce el concepto de *peers rendezvous*, que son un tipo especial de *peers* usados para propagar los mensajes que reciben. Los *peers* pueden convertirse dinámicamente en *peers rendezvous*, o conectarse también dinámicamente a uno de ellos. La conexión entre un *peer* normal y uno de tipo *rendezvous* se realiza mediante el protocolo ERP, que está escuchando en una dirección de punto extremo predeterminado para RVP.

Es un protocolo simple, y posibilita que un grupo tenga su propio protocolo de propagación de más alto nivel. Es usado por los protocolos PRP y PBP para la propagación de mensajes.

3.6 JXTA EN JAVA

JXTA es independiente del lenguaje de programación y del protocolo de transporte. Sin embargo, la implementación realizada en Java para la J2SE se ha convertido en la referencia del proyecto, debido fundamentalmente a 2 razones:

- Permite el rápido desarrollo y prueba de las ideas, resultando en una implementación robusta.
- Java está disponible para todo tipo de plataformas (Unix, Windows, MacOS X, etc.).

Otros subproyectos de JXTA están actualmente trabajando en implementaciones de los protocolos JXTA para C/C++, Python, J2ME, etc. La mayoría de subproyectos dedicados a aplicaciones y servicios JXTA, han elegido la implementación en Java como medio para el desarrollo.

Especialmente interesante es la implementación del protocolo PDP en J2SE: usa una combinación de multidifusión IP dentro de la red local y los *peers rendezvous*. Los *peers rendezvous* proporcionan el mecanismo de envío de peticiones de un *peer*

conocido a otro para descubrir información dinámicamente. Un *peer* puede ser preconfigurado con una lista de *peers rendezvous*, o puede descubrirlos en tiempo de ejecución. Otras técnicas de descubrimiento, como la implementación CAN de DHT, también pueden ser usadas. Se incluye en la versión 2.0 un algoritmo basado en índices distribuidos de recursos compartido (SRDI).

También se implementan nuevos tipos de tuberías: bidireccionales y bidireccionales confiables, extendiendo el concepto de tubería unidireccional proporcionada por el núcleo de JXTA.

El actual material de referencia disponible para usar JXTA en J2SE consta de:

- Guía del programador Java para la versión 2.0 de JXTA.
- Colección de ejemplos-guía.
- Archivos .jar de la implementación J2SE.

En la web de JXTA se encuentra todo el material relacionado con JXTA, además de todos los subproyectos relacionados y amparados por JXTA y desarrollados en Java.