UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA

Sistemas para la Colaboración

JXTA - Aplicaciones Peer to Peer basadas en Java

Juan Andrada Romero
Jose Domingo López López

6 de mayo de 2010

Índice

1. Introducción

A día de hoy, las aplicaciones P2P son de los tipos de aplicaciones más utilizadas en Internet. Esto es debido que el ancho de banda y los recursos de almacenamiento y cómputo son caros en Internet, y gracias al paradigma P2P es posible distribuir dichos recursos entre los distintos computadores que forman parte de la red. Algunos ejemplos de aplicación de las redes P2P son los siguientes:

- Intercambio y búsqueda de ficheros, como Napster y BitTorrent.
- Sistemas de telefonía por Internet, como Skype.
- Sistemas de mensajería instantánea.

Cuando se habla de una red P2P, también llamada *red de pares*, se refiere a una red de computadores en la que todos o algunos aspectos de ésta funcionan sin clientes ni servidores fijos, sino una serie de nodos que actúan simultáneamente como clientes y servidores respecto a los demás nodos de la red [9].

JXTA es una plataforma *open source*, desarrollada por Sun Microsystems en el año 2001, para crear aplicaciones *peer-to-peer* (P2P) basadas en Java. Además, la tecnología JXTA [5] posee un conjunto de protocolos abiertos que permiten que cualquier dispositivo conectado a la red (teléfonos móviles, PC, PDA, servidores, etc.) pueda comunicarse y colaborar siguiendo el paradigma P2P. Los pares JXTA crean una red virtual donde cualquier par puede interaccionar directamente con otros pares y recursos, aunque estén detrás de algún *firewall*, *NAT (Network Address Translation)* o en una red con un protocolo de transporte distinto.

Este documento servirá al lector como una guía de iniciación a la tecnología JXTA. En él se profundizará sobre qué es JXTA y por qué es una buena solución para el desarrollo de aplicaciones P2P, se explicarán conceptos básicos que se utilizan en la comunicación de aplicaciones, se desarrollarán los distintos protocolos que implementa esta tecnología y sus propósitos y, por último, se realizará una pequeña aplicación de ejemplo utilizando Java como lenguaje de programación.

1.1. ¿Qué es JXTA?

JXTA es un proyecto de investigación colaborativa para construir aplicaciones peer-to-peer (P2P). *Proyecto JXTA* (o simplemente JXTA) son las siglas de *Project Juxtapose*.

Las aplicaciones P2P tienen las siguientes características:

- Son espontáneas por naturaleza.
- Son más eficaces cuando la arquitectura de soporte implica muchos proveedores de servicio en lugar de hacerlo en un servidor único y centralizado.
- Los usuarios de la aplicación pueden ser los proveedores o los consumidores del servicio.
- Los usuarios de la aplicación vienen y van y por lo tanto pueden no estar disponibles en un punto específico en el tiempo.
- Los usuarios de la aplicación pueden estar usando cualquier dispositivo en cualquier momento y en cualquier ubicación.

Por este motivo, para satisfacer aplicaciones con estos requisitos, que son difíciles de implementar con la tradicional arquitectura cliente/servidor, se desarrolla el proyecto JXTA. Ésta define un conjunto pequeño de protocolos comunes que permiten la comunicación en aplicaciones P2P, donde cada protocolo es fácil de implementar e integrar en los sistemas existentes. Además, JXTA es independiente de la plataforma e independiente de la red: los protocolos JXTA pueden transmitirse por TCP/IP, HTTP, Bluetooth, redes domésticas tales como Home-PNA, etc. Los pares que se encuentren en redes muy diferentes y lejanas pueden comunicarse fácilmente utilizando los protocolos estándares de JXTA.

1.2. ¿Por qué JXTA?

Uno de los motivos por los que utilizar JXTA es que sus protocolos JXTA permiten a los desarrolladores construir y desplegar aplicaciones y servicios interoperables de P2P. Debido a que los protocolos son independientes del lenguaje de programación y de los protocolos de transporte, diferentes dispositivos heterogéneos con arquitecturas de software completamente diferentes pueden interoperar el uno con el otro.

Otra de las razones para usar JXTA es que proporciona una plataforma común en la cual desarrollar aplicaciones P2P, que tan populares son a día de hoy, ya que cuentan con las siguientes características:

- **Escalabilidad y comportamiento dinámico**: un par puede descubrir servicios de manera automática y dinámica, por lo que no es necesario un servidor central que pueda colapsarse por el gran número de servicios que los pares ofrezcan, ya que estos servicios se publican y utilizan de manera dinámica por cada par de la red, sin tener un registro central. Por otra parte, el número de servicios ofrecidos en la red aumenta, ya que cada par publica su servicio y el resto lo pueden utilizar, por ello es una red escalable.
- **Robustez**: gracias a la arquitectura que se utiliza en una red P2P y a que los servicios están publicados por diferentes pares, este tipo de red es más robusta ante fallos, pues aunque se caigan algunos pares, el resto de la red puede seguir funcionando, ya que aún existirán otros pares que puedan publicar servicios y enrutar mensajes.
- Auto-organización: debido a que no se utiliza un registro central, las comunidades P2P son auto-organizadas. Los pares pueden auto-organizarse en un grupo de pares que están interesados en los servicios de ese grupo.

Tradicionalmente, estas aplicaciones P2P se han desarrollado desde el principio por separado, haciendo que sea incompatibles unas con otras. Sin embargo, JXTA busca proporcionar una plataforma en la que se puedan construir fácilmente aplicaciones de este tipo y que puedan interoperar entre ellas, abstrayendo los diferentes dispositivos donde se ejecutan las aplicaciones P2P, las diferentes tecnologías utilizadas y los protocolos de comunicación involucrados.

2. Conceptos de JXTA

2.1. Par (*Peer*)

Un par *(peer)* es cualquier entidad de red que implementa uno o más protocolos utilizados en JXTA. Así, un par puede residir en una PDA, un teléfono móvil, un PC, etc, trabajando de manera independiente y asíncrona con respecto a otros pares.

Un par debe publicarse en la red para que pueda ser descubierto por otros pares y puedan establecer una comunicación. Por tanto, cada par va a tener un identificador único y una dirección de red, conocida esta última como *peer endpoint* y utilizada para establecer una comunicación punto a punto entre dos pares.

Normalmente, los pares se configuran para ser descubiertos espontáneamente por otros pares de la red, a partir del concepto de *grupo de pares*, que puede ser, por naturaleza, persistente o transitorio.

Los pares en JXTA pueden clasificarse en tres categorías:

- ***Minimal-Edge Peers***: pares que implementan sólo los servicios principales JXTA, utilizando otros pares para que actúen como su proxy a la hora de utilizar otros servicios que no pertenecen al *core*de JXTA. Pares que pertenezcan a esta categoría pueden ser sensores.
- ***Full-Edge Peers***: pares que implementan todos los servicios de JXTA y pueden participar en todos sus protocolos. Estos pares conforman la mayoría de pares en una red JXTA y pueden ser teléfonos, PC's, servidores, etc.
- ***Super Peer***: pares que implementan y proveen recursos para soportar el despliegue y operación de una red JXTA. Hay tres funciones principales que debe tener un *Super-Peer*:
  - **Retransmisión**: usada para almacenar y reenviar mensajes entre pares que no tienen conectividad directa debido a cortafuegos o redes privadas.
  - **Citas**: se ocupa de mantener los anuncios globales sobre donde están publicados otros pares y ayuda a los pares que actúan como proxy para buscar esos anuncios.
  - **Proxy**: utilizado por los *minimal-edge peers* para tener acceso a todas las funcionalidades de la red JXTA. El par proxy traduce las peticiones, responde a las consultas y proporciona soporte para la funcionalidad de los *minimal-edge peers*.

2.2. Grupo de pares (*Peer Group*)

Un grupo de pares es una colección de pares que comparten un conjunto común de servicios o intereses. Los pares se auto-organizan en grupos, donde cada uno de los grupos se identifica por un identificador único. Cada grupo establece su propia política de pertenencia al grupo, que puede variar desde una política abierta (cualquier par puede asociarse al grupo) hasta la muy segura y protegida (que requiere credenciales para poder ser miembro).

Los pares pueden pertenecer a más de un grupo simultáneamente. Por defecto, el primer grupo que se instancia es el *Network Peer Group*, y, posteriormente, todos los pares pueden optar por unirse a otros grupos en cualquier momento.

Los protocolos JXTA describen como los pares pueden publicarse, descubrirse, unirse, y monitorear grupos de pares; sin embargo, dichos protocolos no indican cuándo o por qué se crean los grupos de pares. Por otra parte, unirse a un grupo consiste simplemente en instanciar todos los servicios definidos por ese grupo.

Hay varias motivaciones para la creación de grupos de pares:

- Para crear un entorno seguro, ya que se puede aplicar una política de seguridad. Ésta puede ser tan simple como un intercambio de texto plano con nombre de usuario y contraseña, o tan sofisticada como criptografía de llave publica. Así, los miembros del grupo pueden publicar y acceder a contenido protegido.
- Para delimitar el alcance, ya que los grupos que permiten la creación de un dominio local específico. Por ejemplo, los pares pueden agruparse para implementar una red de documentos compartidos o una red de CPUs compartidas. Los grupos de pares sirven para subdividir la red en regiones abstractas proporcionando un mecanismo implícito de alcance. De este modo, las fronteras del grupo definen el alcance de la búsqueda cuando se busca algo dentro del grupo.
- Para crear un entorno de monitorización, ya que permiten monitorear un conjunto de pares para un fin especial.

2.3. Servicios de red (*Network Services*)

Los pares cooperan y se comunican para publicar, descubrir, e invocar servicios de red. Además, los pares pueden publicar múltiples servicios los cuáles, a su vez, son descubiertos a través del *Peer Discovery Protocol*. De este modo, los protocolos JXTA reconocen dos niveles de servicios de red:

- **Servicios de par**: un servicio de este tipo es solamente accesible por el par que está publicando ese servicio. Si ese par fallara, el servicio también fallaría.
- **Servicios de grupo de pares**: un servicio de grupo de pares está compuesto de una colección de instancias del servicio, cooperando entre sí y funcionando en múltiples miembros del grupo. Sí falla cualquiera de los pares, el servicio colectivo del grupo no se verá afectado siempre y cuando el servicio este aún disponible desde al menos un par miembro.

2.4. Servicios de grupos de pares (*Peer Group Services*)

JXTA define un conjunto básico de servicios de grupos de pares, que todos los pares miembros del grupo deben implementar, al igual que los pares que se unan a dicho grupo. Así, para que dos pares puedan interactuar a través de un servicio, deben ser ambos miembros del mismo grupo. Los servicios de grupo que cada uno de los pares debe implementar con el fin de participar en una red JXTA son:

- **Servicio de Destino (*Endpoint Service*)**: el servicio de destino o de *punto final* es utilizado para enviar y recibir mensajes entre pares. Este servicio normalmente implementa el protocolo de encaminamiento al destino (ERP), aunque implementaciones mínimas proporcionan sólo una pequeña parte del total de este protocolo.
- **Servicio de Resolución (*Resolver Service*)**: el servicio de resolución es utilizado para enviar solicitudes de consulta genérica a otros pares. Los pares pueden definir e intercambiar consultas para encontrar cualquier información que pueda ser necesaria (por ejemplo, encontrar otros pares, determinar el estado de un servicio o la disponibilidad de una tubería, etc.).

Además de los servicios básicos anteriores, se pueden definir otros servicios estándares en los grupos, aunque éstos no tienen por qué ser implementados por todos los grupos, sino sólo por aquellos que los necesiten. Los servicios estándares que se encuentran en la mayoría de grupos de pares son:

- **Servicio de Descubrimiento (*Discovery Service*)**: el servicio de descubrimiento es usado por los pares miembros para buscar recursos en el grupo, tales como otros pares, otros grupos, servicios y tuberías.
- **Servicio de afiliación a un grupo(*Membership Service*)**: es utilizado por los pares para identificarse con seguridad dentro de un grupo. La identidad de un par permite a las aplicaciones y los servicios determinar quién está solicitando una operación y decidir si la operación debe permitirse o no. Las aplicaciones pueden realizar su propio control de acceso o pueden utilizar el servicio de acceso.
- **Servicio de Acceso (*Access Service*)**: el servicio de acceso es usado para validar las peticiones hechas por un par a otro. El par que recibe la petición verifica las credenciales del par que la solicita para determinar si se le permite el acceso. Sin embargo, no todas las acciones dentro del grupo necesitan comprobarse con el servicio de acceso; sólo aquellas acciones que son limitadas a algunos pares necesitan ser comprobadas.
- **Servicio de Tuberías (*Pipe Service*)**: el servicio de tubería es utilizado para crear y manejar conexiones de tubería entre los miembros del grupo.
- **Servicio de Monitoreo (*Monitoring Service*)**: el servicio de monitoreo se utiliza para permitir a un par monitorizar a otros miembros del mismo grupo de pares.

2.5. Módulos

Los módulos de JXTA son una abstracción de JXTA de bajo nivel usada para representar cualquier pieza de ”código” y el API que proporciona ese código para su uso. Los módulos son utilizados para implementar en JXTA servicios, transporte de mensajes, etc.

Los módulos proveen una abstracción genérica para permitir a un par instanciar una función o servicio como, por ejemplo, cuando se une a un grupo, ya que necesitan implementar los servicios de ese grupo. Los módulos permiten la representación y la publicación de funcionalidades independientes de la plataforma, y permite a los pares describir e instanciar cualquier tipo de implementación de una funcionalidad. Por ejemplo, un par puede instanciar una implementación en Java o C de una funcionalidad específica.

La posibilidad para describir y publicar (o anunciar) una funcionalidad independiente de la plataforma es esencial para soportar el desarrollo de nuevos servicios de grupo. De hecho, JXTA utiliza los anuncios de módulos para auto-describir sus propios servicios.

Un módulo incluye los siguientes elementos:

- **Clases**: las clases de un módulo se utilizan principalmente para anunciar la existencia de una funcionalidad. Cada clase de un módulo es identificada por un único ID, el *ModuleClassID*.
- **Especificación**: la especificación de un módulo se utiliza para acceder a un módulo, ya que contiene toda la información necesaria para acceder o invocar dicho módulo. Por tanto, la especificación de un módulo es un método para proporcionar la funcionalidad que implica ese

módulo, pudiendo existir diversas especificaciones para un determinada clase. Cada especificación se identifica de manera única, a través del *ModuleSpecID*.

- **Implementación**: la implementación de un módulo es la implementación de una determinada especificación, pudiendo existir múltiples implementaciones para una determinada especificación del módulo. Cada implementación del módulo contiene el *ModuleSpecID* de la especificación asociada que implementa.

Así, los servicios JXTA pueden usar los módulos para identificar la existencia del servicio (las clases del módulo), la especificación del servicio (una especificación del módulo), o una implementación del servicio (una implementación del módulo). Cada uno de estos componentes tiene un anuncio asociado el cuál puede ser publicado y descubierto por otro par JXTA.

Por ejemplo, el servicio de descubrimiento de JXTA (*Discovery Service*) tiene un único *ModuleClassID*, identificándolo como un servicio de descubrimiento, que es su funcionalidad abstracta. Además, puede tener múltiples especificaciones del servicio, utilizando diferentes estrategias para realizar el descubrimiento, adaptadas al tamaño del grupo y su dispersión a través de la red. Para cada especificación, puede haber múltiples implementaciones, cada una de las cuales contiene el mismo *ModuleSpecID*.

2.6. Mensajes

Los servicios y aplicaciones JXTA se comunican usando mensajes JXTA, que son las unidades básicas de intercambio de datos entre pares. Cada protocolo JXTA está definido como un conjunto de mensajes que intercambian los pares participantes. Los mensajes son enviados entre los pares utilizando el servicio de destino (*Endpoint Service*), el servicio de tubería (*Pipe Service*), así como un socket y otras vías de acceso. La mayoría de las aplicaciones no necesitan utilizar directamente tuberías unidireccionales o el servicio de destino, sino que utilizan comúnmente *sockets* de comunicación y tuberías bidimensionales.

Los protocolos de JXTA están especificados como un conjunto de mensajes intercambiados entre pares. El uso de mensajes de XML para definir los protocolos permite a muchos tipos diferentes de pares utilizar un protocolo determinado. Debido a que los datos están etiquetados, cada par es libre para implementar el protocolo de un modo más adecuado a sus capacidades y rol. Por ejemplo, un par que esté muy limitado y no tenga capacidad suficiente para procesar la totalidad de un mensaje, puede usar las etiquetas de datos para extraer las partes que pueda procesar e ignorar las restantes.

2.7. Tuberías (*Pipes*)

Los pares de JXTA utilizan tuberías para enviar mensajes a otro par. Por tanto, las tuberías son un mecanismo de transferencia de mensajes asíncrono, unidireccional y no confiable (con la excepción de las tuberías punto a punto segurass) usados para la comunicación y transferencia de datos. Las tuberías son canales de comunicación virtuales y pueden conectar pares que no tienen un vinculo físico directo, lo que se traduce en una conexión lógica. Las tuberías pueden ser usadas para enviar cualquier tipo de datos incluyendo texto XML y HTML, imágenes, música, código binario, cadenas de datos y objetos Java.

La tuberías de destino o de ”punto final” (*endpoints*) están dinámicamente vinculadas con los pares de destino (correspondientes a interfaces de red, como un puerto TCP) cuando la tubería se abre. Toda la resolución y comunicación de tuberías se realiza dentro del alcance de un grupo de pares. es decir, las tuberías de salida y entrada deberán pertenecer al mismo grupo de pares.

Las tuberías ofrecen dos modos de comunicación: punto a punto (*unicast*) y propagación, como se puede observar en la Figura 2.1.

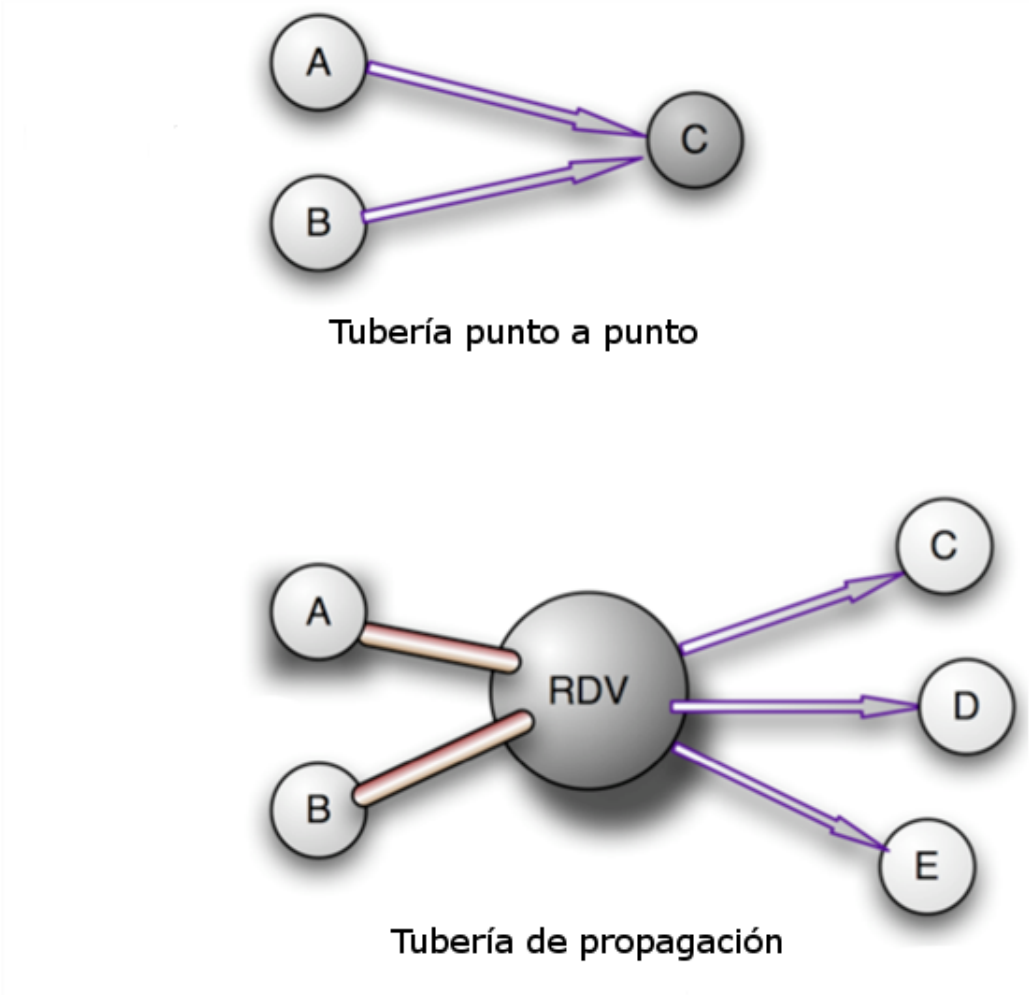

Figura 2.1: Tipos de tuberías

- **Tuberías punto a punto**: una tubería punto a punto conecta exactamente dos tuberías de punto final. La tubería de entrada en un par recibe mensajes enviados desde la tubería de salida del otro par. También es posible para varios pares unirse a una sola tubería de entrada.
- **Tuberías de propagación**: este tipo de tuberías conecta una tubería de salida a varias tuberías de entrada. Los mensajes fluyen desde la tubería de salida, que es la fuente de propagación, hacia las tuberías de entrada.
- **Tuberías punto a punto seguras**: es un tipo de tubería punto a punto que provee un canal de comunicación seguro y fiable.

2.8. Anuncios (*Advertisements*)

Todos los recursos de la red JXTA (servicios, pares, grupos de pares y tuberías) son representados como anuncios. Los anuncios son un lenguaje de estructuras de meta-datos representados como documentos XML (ver Figura 2.2). Los protocolos JXTA usan anuncios para describir y publicar la existencia de recursos de un par. Del mismo modo, los pares descubren recursos mediante la búsqueda de sus correspondientes anuncios y pueden guardar en caché cualquier anuncio local descubierto.

Cada anuncio es publicado con un tiempo de vida que especifica la disponibilidad de sus recursos asociados. El tiempo de vida permite la supresión de los recursos obsoletos sin la necesidad de un control centralizado, de modo que un anuncio puede ser publicado de nuevo (antes de que el anuncio original expire) para extender la vida útil de un recurso.

Los protocolos JXTA definen los siguientes tipos de anuncios:

- **Anuncios de pares (*Peer Group Advertisement*)**: describen los recursos de un par. El principal uso de estos anuncios es para mantener información específica sobre el par, como su nombre, su identificador, destinos disponibles, y cualquier atributo que se desee publicar en tiempo de ejecución.
- **Anuncios de grupos de pares (*Peer Group Advertisement*)**: describe los recursos de un grupo específico de pares, como son el nombre, el identificador del grupo, descripción, etc.
- **Anuncios de tubería (*Pipe Advertisement*)**: describe un canal de comunicación de una tubería, y es usado por el servicio de tubería para crear las tuberías asociadas a una entrada y a una salida. Cada anuncio de tubería contiene un identificador simbólico y un tipo de tubería (punto a punto, propagada, segura).
- **Anuncio de clases de un módulo (*Module Class Advertisement*)**: describe una clase de un módulo. Su objetivo principal es documentar formalmente la existencia de una clase del módulo. Incluye un nombre, descripción, y un único identificador (*ModuleClassID*).
- **Anuncio de especificación de un módulo (*Module Specification Advertisement*)**: describe una especificación de un módulo. Su principal objetivo es proveer referencias a la documentación necesaria con el fin de crear implementaciones conforme con la especificación. Un uso secundario es hacer correr instancias que se puedan utilizar de manera remota mediante la publicación de información. Este anuncio incluye nombre, descripción, identificador único de la especificación del módulo (*ModuleSpecID*), anuncio de tubería y un campo que contiene parámetros arbitrarios a ser interpretados por cada aplicación.
- **Anuncios de implementación de un módulo (*Module Implementation Advertisement*)**: define una implementación de una especificación de un módulo. Incluye un nombre, el *ModuleSpecID* asociado, así como el código, paquete y campos de parámetros que permiten a un par recuperar los datos necesarios para ejecutar la implementación.
- **Anuncio de citas (*Rendezvous Advertisement*)**: describe como se establecen citas entre pares de un grupo.
- **Anuncio de información de un par (*Peer Info Advertisement*)**: describe los recursos de información de un par. El uso principal de este anuncio es contener información especifica acerca del estado actual de un par, tales como el tiempo de actividad, cuenta de mensajes de entrada y salida, el tiempo que ha pasado desde el ultimo mensaje recibido y el tiempo que ha pasado desde el ultimo mensaje enviado.

```
<?xml version="1.0"?>
<!DOCTYPE jxta:PipeAdvertisement>
<jxta:PipeAdvertisement xmlns:jxta="http://jxta.org">
 <Id>
 urn:jxta:uuid-596162616461626|14E504720503250338E3E786229EA460DADC1A176B69B731504
 </Id>
 <Type>
  JxtaUnicast
 </Type>
 <Name>
  TestPipe
 </Name>
 </jxta:PipeAdvertisement>
```

Figura 2.2: Ejemplo de un anuncio de una tubería

2.9. Seguridad

Redes P2P dinámicas, como la red JXTA, necesitan soportar diferentes niveles de acceso a los recursos. En JXTA, los pares operan en un modelo basado en roles, en el cual un par individual actúa bajo la autoridad de otro par de confianza para realizar una tarea.

Se proveen cinco requisitos básicos de seguridad en JXTA:

- **Confidencialidad**: garantiza que el contenido de un mensaje no se divulgará a pares no autorizados.
- **Autenticación**: garantiza que el emisor de un mensaje es quién dice ser.
- **Autorización**: garantiza que el emisor está autorizado para enviar un mensaje.
- **Integridad de los datos**: garantiza que el mensaje no ha sido modificado accidental o deliberadamente mientras se transmitía.
- **Refutabilidad**: garantiza que el mensaje fue transmitido por un emisor identificado y que no es una respuesta de un mensaje previamente transmitido.

Los mensajes XML permiten añadir nuevos meta-datos como credenciales, certificados y claves públicas a los mensajes de JXTA, permitiendo que se cumplan los requisitos de seguridad anteriores. Además, los mensaje se pueden encriptar y firmar.

Un credencial es un token que se utiliza para identificar un emisor y para comprobar que tiene autorización para enviar el mensaje. Además, este token está presenta en cada mensaje que ese emisor envía, comprobándose con la identidad del emisor cuando se recibe el mensaje. Cabe destacar que la implementación de las credenciales se realiza como una configuración de un plug-in, por lo que pueden exisitr múltiples configuraciones de credenciales dentro de la misma red. Con esto se intenta que JXTA sea compatible con otros mecanismos de seguridad como SSL (*Secure Socket Layer)* e IPSec (*Internet Protocol Secure*).

2.10. Identificadores (*IDs*)

Los pares, grupos de pares, tuberías y otros recursos de JXTA necesitan tener un identificador único. Un ID de JXTA únicamente identifica un recurso y sirve como una forma canónica de referirse a ese recurso.

Este ID se expresa en forma de URN (*Uniform Resource Name*), que son una forma de URI (*Uniform Resource Identifier)* destinados a servir como identificadores de recursos de manera persistente e independiente de la ubicación. En la Figura 2.3 se puede observar un ejemplo de un ID de par JXTA.

```
urn:jxta:uuid-59616261646162614A78746150325033F3BC76FF13C2414CBC0AB663666DA53903
```

Figura 2.3: Ejemplo de un ID de un par JXTA

3. Arquitectura Software de JXTA

En los siguientes apartados se va a detallar la arquitectura software de JXTA y los componentes que en ella intervienen.

3.1. Descripción

La arquitectura software de JXTA está dividida en tres capas, como se muestra en la Figura 3.1.

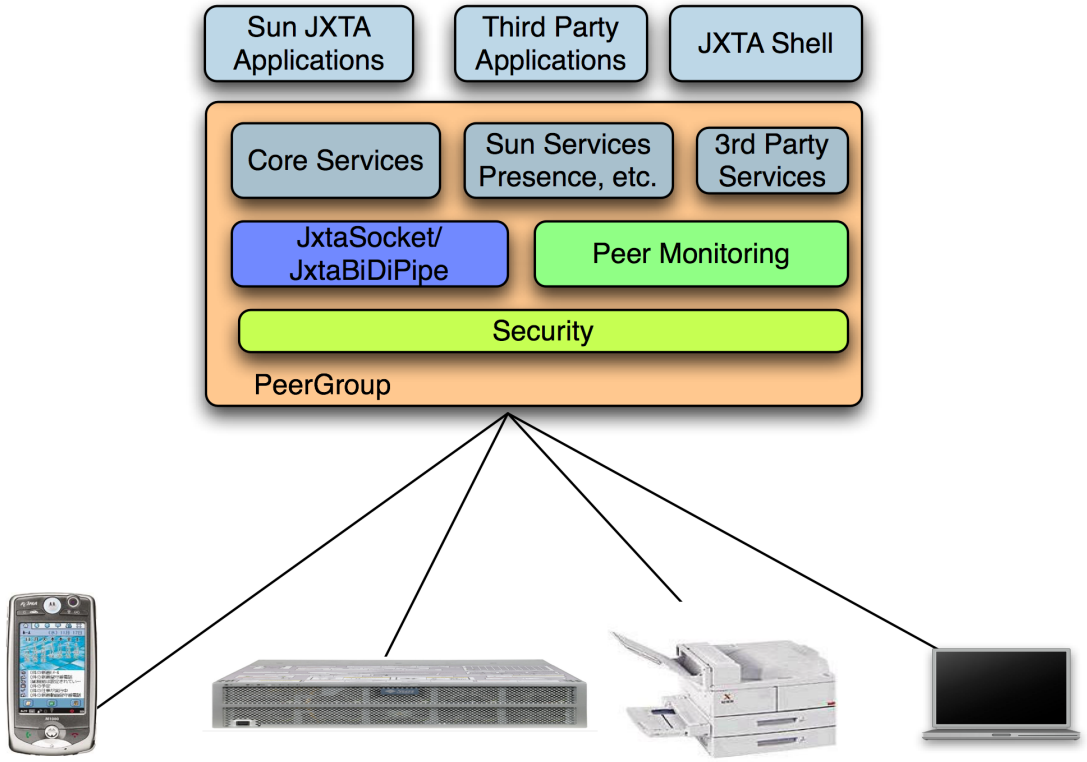

Figura 3.1: Arquitectura de JXTA

- **Núcleo (*core*) de JXTA**: el núcleo de JXTA encapsula las primitivas mínimas y esenciales que se utilizan comúnmente para la comunicación P2P. Esto incluye componentes básicos para permitir las funcionalidades clave para las aplicaciones P2P, incluyendo descubrimiento, la creación de pares y grupos de pares y primitivas asociadas con seguridad.
- **Capa de servicios**: la capa de servicios incluye los servicios de red que podrían no ser absolutamente necesarios para que una red P2P opere, pero que son comunes o deseables. Los ejemplos de servicios de red incluyen búsqueda e indexación, directorios, sistemas de almacenamiento, uso compartido de archivos, sistemas de archivos distribuidos, agregación y petición de recursos, autenticación y servicios de PKI (*Public Key Infrastructure*).
- **Capa de aplicaciones**: la capa de aplicaciones incluye la implementación de aplicaciones integradas, como aplicaciones P2P de mensajería instantánea, distribución de documentos y de recursos, sistemas de correo electrónico P2P, sistemas de subasta distribuido, etc.

Sin embargo, la frontera entre aplicaciones y servicios no es rígida, ya que una aplicación de un cliente puede considerarse como un servicio para otro cliente. El sistema JXTA está diseñado para ser modular, permitiendo a los desarrolladores seleccionar y escoger una colección de servicios y aplicaciones que se adapten a sus necesidades.

3.2. Componentes de JXTA

Una red JXTA consiste en una serie de nodos (pares) interconectados mediante cualquier protocolo de interconexión adecuado, incluyendo TCP/IP, HTTP, Bluetooth, GSM, etc. Además, múltiples pares pueden correr sobre un único dispositivo físico y varios dispositivos físicos podrían cooperar para actuar como un solo par.

Cada par ofrece un conjunto de servicios y recursos que están a disposición de otros pares. Así, existen dos tipos de servicios comunes en las redes JXTA, que son los servicios de pares y servicios de grupo. De este modo, todos los pares JXTA implementan un pequeño número de servicios básicos necesarios y también proporcionan varios servicios estándares adicionales. Cada grupo de pares, por su parte, incluye el conjunto de servicios de grupo que cada par debe ejecutar para participar en ese grupo.

Los recursos de un par son normalmente contenido estático (no-interactivo) los cuáles él controla, posee o simplemente tiene una copia. Los recursos pueden incluir archivos, documentos, índices, pero también pueden ser recursos del mundo real tales como switches, sensores e impresoras. Dichos servicios y recursos son publicados como un *anuncio*.

Otro componente necesario en una red JXTA son *sockets* y tuberías, que se utilizan para enviar mensajes entre pares. Como ya se ha comentado en la sección 2, los *sockets* de JXTA son conexiones bidireccionales confiables usadas por las aplicaciones para comunicarse., y las tuberías son un mecanismo de transferencia de mensajes asíncrono y unidireccional usado para la comunicación del servicio. Los mensajes son simples documentos XML que contienen información acerca de la ruta a seguir, un resumen y credenciales.

Para terminar, se destacan cuatro aspectos esenciales de la arquitectura JXTA que la distinguen de otros modelos de red distribuida.

1. El uso de documentos XML (*anuncios*) para describir recursos de red.
2. La abstracción de tuberías a pares y de pares a "puntos finales" (*endpoints*), sin depender de una autoridad central de nombramiento/direccionamiento, tal como DNS.
3. Un esquema de direccionamiento uniforme de pares.
4. Una infraestructura de búsqueda descentralizada basada sobre tablas Hash distribuidas (*Distributed Hash Table - DHT*) para la indexación de recursos.

4. Arquitectura de red de JXTA

4.1. Organización de la red

La red JXTA es una red *Ad-Hoc*, multi-salto y adaptativa, compuesta de pares conectados. Las conexiones en la red pueden ser temporales y, como resultado, el enrutamiento de mensajes entre pares es no determinista, ya que los pares pueden unirse o dejar la red en cualquier momento, lo que resulta siempre en un cambio de información de enrutamiento.

Normalmente se utilizan cuatro tipos de pares en una red JXTA, como se observa en la Figura 4.1:

- ***Minimal-Edge Peer***: puede enviar y recibir mensajes, pero no guardar anuncios en caché o enrutar mensajes para otros pares.
- ***Full-featured Peer***: puede enviar y recibir mensajes y típicamente guardara en caché los anuncios. Este tipo de par simple contesta a peticiones de descubrimiento con información encontrada en su caché de anuncios, pero no reenvía ninguna petición de descubrimiento.
- ***Rendezvous peer***: es un par de infraestructura que ayuda a otros pares con la propagación de mensajes, descubrimiento de anuncios y rutas y mantiene un mapa topológico de otros pares de infraestructura, que se utilizan para controlar la propagación y el mantenimiento de la tabla Hash distribuida. Cada grupo de pares mantiene su propio conjunto de pares *rendezvous* y puede tener tantos como sea necesario.
- ***Relay peer***: es un par de infraestructura que ayuda a otros pares no direccionables con la retransmisión del mensaje. Un par puede solicitar una sección del mensaje a este tipo de par para facilitar la retransmisión del mensaje cada vez que sea necesario.

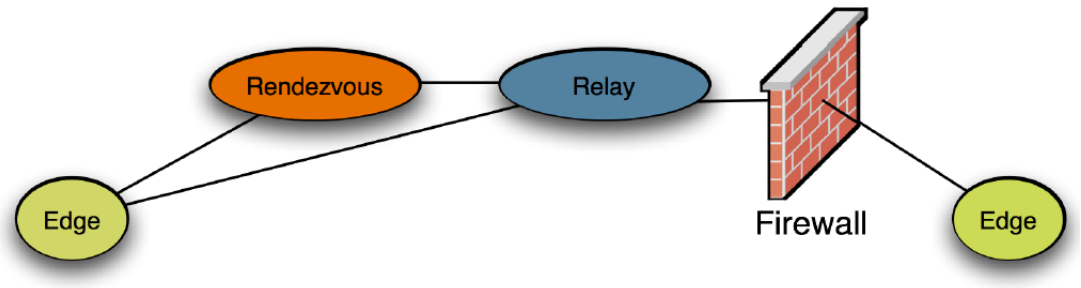

Figura 4.1: Arquitectura de red de JXTA

4.2. Índice Distribuido de Recursos Compartidos (*Shared Resource Distributed Index - SRDI*)

JXTA implementa un SRDI para conseguir un mecanismo eficiente de propagación de peticiones en la red JXTA. Los pares de tipo *rendezvous* mantienen un índice de anuncios publicados por los pares de tipo *edge*, de tal modo que cuándo uno de estos pares publica un anuncio, el par *rendezvous* lo indexa en su SRDI, utilizando como clave el ID del anuncio.

Cada par de tipo *rendezvous* conoce a otros pares de este tipo que se encuentren en su grupo, de tal modo que cuando se indexa un nuevo recurso, el índice utilizado se envía al resto de pares *rendezvous*, para que también lo almacenen en su SRDI.

A continuación, se va a comentar un ejemplo de uso del SRDI cuando un par solicita un recurso, que es el mostrado en la Figura 4.2:

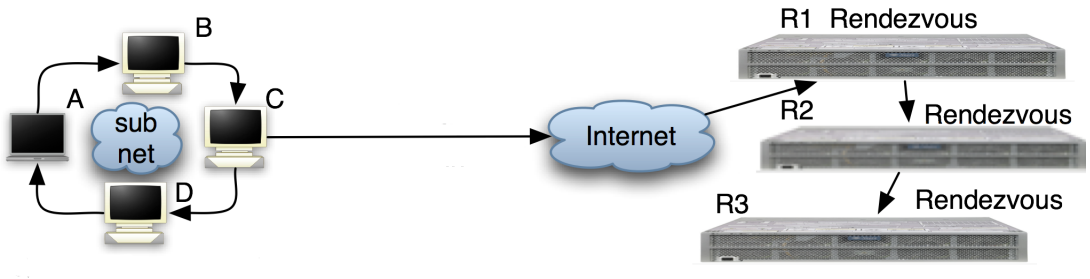

Figura 4.2: Propagación de una petición en una red JXTA usando SRDI

- El par A de tipo *edge* solicita un recurso a su par R1 de tipo *rendezvous* y al resto de la red, usando difusión (*multicast*).
- Si los pares de la red de A tienen el recurso en su caché, se lo envían.
- Del mismo modo, el par R1 usa el índice del anuncio de petición para buscarlo en su SRDI. Si lo encuentra, se lo envía al par A. Si no lo encuentra, propaga la petición a los pares *rendezvous* que se encuentran en su grupo, para que éstos realicen la búsqueda o repitan este proceso si no tienen ese recurso disponible.

4.3. Firewalls y NAT

Un par detrás de un cortafuegos (*firewall*) puede enviar un mensaje directamente a un par que esté fuera del cortafuegos, pero no es posible establecer una comunicación directa en el otro sentido. Lo mismo ocurre para los pares que se encuentren tras un dispositivo de NAT (*Network Address Translation*).

Para poder realizar la conexión con pares que están tras cortafuegos o usan NAT, se deben cumplir las siguientes condiciones:

- Al menos un par del grupo que se encuentra en el cortafuego debe poder comunicarse con un par tras el cortafuegos.
- Estos pares deben soportar un protocolo común de transporte, como TCP o HTTP.
- El cortafuegos debe permitir tráfico HTTP o TCP.

5. Protocolos utilizados en JXTA

JXTA define una serie de mensajes XML, o protocolos, para la comunicación entre pares. Los pares utilizan estos protocolos para descubrir uno a otro, anunciar y descubrir los recursos de red.

Existen seis tipos de protocolos en JXTA, que son asíncronos y se basan en un modelo de consultas y respuestas. Un par JXTA utiliza uno de los protocolos para enviar una consulta a uno o más pares de su grupo. Es posible que reciba cero, una o más respuestas a su consulta. Por ejemplo, un par puede utilizar el protocolo PDP para enviar una consulta de descubrimiento, solicitando todos los pares registrados en el grupo por defecto (*Net Peer Group*). En este caso, varios pares responderán con sus respuestas de descubrimiento.

Los pares JXTA no necesitan implementar todos los protocolos, sólo necesitan implementar los protocolos que vayan a utilizar.

En los siguientes apartados se detallan cada uno de los protocolos que existen en JXTA.

5.1. Protocolo de Descubrimiento de Pares (*PDP - Peer Discovery Protocol*)

El protocolo de descubrimiento de pares (PDP) se utiliza para descubrir cualquier recurso publicado por otros pares en forma de anuncio. Un recurso puede ser un par, un grupo de pares, una tubería, un servicio o cualquier otro recurso que tenga un anuncio.

PDP es el protocolo de descubrimiento por defecto para todos los grupos de pares definidos por el usuario y para el el grupo de pares por defecto. De este modo, si se definen servicios de descubrimiento personalizado, éstos se pueden aprovechar del PDP, redefiniendo las partes que sean necesarias.

Existen varias formas para descubrir información distribuida. La implementación actual de JXTA utiliza una combinación de propagación (*multicast*) IP a la subred local y el uso de una red de citas, que es una técnica basada en citas mantenido la DHT (tabla hash distribuido). Los nodos de la red de tipo *rendezvous* proporcionan el mecanismo necesario para dirigir las solicitudes y poder descubrir de forma dinámica la información. Además, un nodo puede configurarse para que, al iniciarse, localice dinámicamente nodos o recursos de su red local a través de mensajes de multidifusión.

5.2. Protocolo de Información de Pares (*PIP - Peer Information Protocol*)

Este protocolo es utilizado por los pares para obtener información de estado (el tiempo de actividad, estado, reciente de tráfico, etc.) de otros pares. Así, una vez que se encuentra un nodo utilizando el protocolo anterior, se pueden consultar sus capacidades y su estado.

El protocolo de información de pares proporciona un conjunto de mensajes para obtener información del estado de un par. Esta información puede utilizarse para la implementación comercial o interna de aplicaciones JXTA. Por ejemplo, en las implementaciones comerciales, la información

puede utilizarse para determinar el uso de un servicio de pares y facturar a los consumidores por el uso de ese servicio. En una implementación interna de una empresa, la información puede utilizarse por el departamento de tecnologías de la información (TI) para supervisar el tráfico de red para mejorar el rendimiento general.

Entre los mensajes que permite utilizar este protocolo, destacan el mensaje de *ping* y el mensaje de *PeerInfo*. El primero de ellos se envía a un para comprobar está vivo y para obtener información acerca de él. El mensaje de ping especifica si se devuelve una respuesta completa (anuncio de pares) o un mensaje de notificación (si está vivo o no y tiempo de actividad).

Por otra parte, el mensaje de *PeerInfo* se utiliza para enviar un mensaje en respuesta a un mensaje de *ping*. Contiene las credenciales del remitente, el ID de los pares de origen y destino, el tiempo de actividad y el anuncio de pares.

5.3. Protocolo de Resolución de Pares (*PRP - Peer Resolver Protocol*)

El protocolo de resolución de pares (PRP) permite a un par enviar solicitudes de consultas genéricas a otros pares, e identificar respuestas coincidente a esas consultas. Las solicitudes de consulta pueden ser enviadas a un punto específico o pueden ser propagadas a través de los servicios de cita a otros pares en el ámbito de un grupo de pares. De este modo, el PRP utiliza el servicio de citas para difundir una consulta a varios pares y utiliza mensajes *unicast* para enviar consultas a pares específicos.

El PRP es un protocolo base para apoyar las solicitudes de consultas genéricas. Tanto el PIP como el PDP se construyen usando PRP y proporcionan consultas y peticiones específicas: el PIP se utiliza para consultar la información de estado y el PDP se utiliza para descubrir recursos de pares. Para cualquier consulta genérica que puede ser necesaria en una aplicación, se puede utilizar el PRP.

El mensaje de respuesta a la consulta (mensaje de resolución de la consulta) contiene la credencial del par que envía la petición, un controlador de servicio específico, un ID único de la consulta y el remitente. Cada servicio puede registrar un controlador en el grupo de pares para procesar las solicitudes de consulta y generar las respuestas.

5.4. Protocolo de Enlace a Tuberías (*PBP - Pipe Binding Protocol*)

El protocolo de enlace de tubería (es utilizado por miembros del grupo de pares para enlazar un anuncio de tubería a un par que se encuentra en el extremo de la tubería. De este modo, una tubería puede ser vista como una cola de mensajes con nombre abstracto, que soporta operaciones de creación, apertura, cierre y borrado de la tubería, y emisión y recepción de mensajes. Las implementaciones reales de una tubería pueden diferir, pero todas las implementaciones usan PBP para enlazar la tubería a un extremo.

Los mensajes de consulta de este protocolo se envían por un extremo de la tubería para encontrar a un par en el extremo de la tubería enlazado al mismo anuncio de la tubería. El mensaje de consulta podrá solicitar información para obtener información acerca de los pares enlazados

a la tubería. El mensaje de consulta también puede contener, opcionalmente, un ID que, si está presente, indica que sólo el par especificado debe responder a la consulta.

En cuánto a los mensajes de respuesta, éstos se envían por cada par asociado a la tubería al par solicitante, es decir, a aquel que realizó la consulta.

5.5. Protocolo de Encaminamiento al Destino (*ERP - Endpoint Routing Protocol*)

El protocolo de encaminamiento al destino permite a los pares enviar mensajes a pares remotos sin tener una conexión directa con ellos. El mensaje se pasará a través de pares intermediarios para llegar a su destino final. ERP define un protocolo de consulta y respuesta que se utiliza para descubrir información de enrutamiento de pares, adjuntando a los mensajes utilizados en JXTA un mensaje especial que describe la ruta por la que debe viajar la información desde un par (el origen) a otro (el destino).

Para enviar un mensaje a otro par, el origen primero busca en su memoria caché para determinar si tiene una ruta hacia el par destino. Si no tiene una ruta hacia el destino, envía a una consulta de solicitud de ruta pidiendo información de la ruta a seguir para el par de destino. Cualquier par que reciba esta consulta de ruta comprobará si conoce alguna una ruta para el par solicitante. Si conoce una ruta válida, responderá a la consulta de ruta con la información de la ruta que él posee para ese destino. De este modo, cualquier par puede realizar consultas sobre rutas y cualquier par dentro de un grupo de pares puede ofrecer información de ruta o distribuir mensajes destinados a otros pares.

Las solicitudes de consulta de ruta son enviadas por un par origen para solicitar la información de ruta hacia un par de destino. Las respuestas de la ruta incluyen el ID del par de origen, el ID del par de destino de la ruta y una secuencia semi-ordenada de IDs de pares. Esta secuencia representa los pares que van a formar la ruta para transmitir el mensaje desde el par origen al par de destino, es decir, son los diferentes saltos o nodos que va a seguir el mensaje. Cada par a lo largo de la ruta indicada en la secuencia puede mejorar y optimizar la ruta basándose en su propia información. Por ejemplo, si un par recibe un mensaje que contenga una ruta con diez saltos pero ese par está directamente conectado al destino, puede reenviar el mensaje directamente, en lugar de enviarlo a lo largo de la ruta.

El procedimiento de enrutamiento de mensajes utilizado por este protocolo sigue aproximadamente estos pasos:

1. El par de origen comprueba si está conectado al destino. Si no tiene una ruta hacia el destino, realiza una consulta de ruta y espera una respuesta, renunciando a la comunicación si no la recibe en un tiempo determinado.

2. El mensaje se difunde por los pares del grupo, estableciendo la secuencia de pares que pueden transmitir el mensaje hacia el destino pasando por otros pares. Así, se crea la ruta entre el origen y el destino.

3. Si es un par intermedio en la ruta y no tiene una ruta hacia el destino o no se puede reenviar el mensaje a otro par que se haya establecido en la ruta encontrada, se manda un mensaje de fallo al origen, indicando que es imposible enviar su mensaje al destino.

4. Si no ocurre la situación anterior, el par intermedio lo reenvía al siguiente par en la ruta, o al destino si está conectado directamente con éste.

5.6. Protocolo de Citas (*RVP - Rendezvous Protocol*)

El protocolo de citas se utiliza para la propagación de mensajes dentro de un grupo de pares. El RVP proporciona mecanismos para habilitar la propagación de mensajes, realizándose de un modo controlado y eficiente. Este protocolo se divide en tres partes:

- El protocolo utilizado por los pares de tipo *Rendezvous* para organizarse, también conocido como protocolo ***PeerView***.
- El protocolo utilizado por pares cliente para registrar su interés de recibir mensajes de propagación, conocido como protocolo simple de arrendamiento.
- El protocolo utilizado para la propagación de mensajes a los pares que han registrado su interés para recibir mensajes. Este protocolo es el único que todos los participantes deben implementar.

6. Iniciación a la programación de aplicaciones con JXTA

El objetivo de esta sección es que sirva al lector como guía de referencia en su iniciación a la programación de aplicaciones basadas en JXTA. Para ello se comenzará configurando el entorno de Java y se darán unas nociones básicas de configuración del entorno de JXTA. Acto seguido, se indicará como ejecutar el *shell* de JXTA y, por último, se hará un pequeño ejemplo tipo *HelloWorld*.

6.1. Requisitos del sistema

La implementación actual de JXTA requiere que el sistema operativo tenga instalado una versión igual o posterior a la 5.0 del entorno de ejecución de Java (JRE o *Run-Time Environtment*) o del entorno de desarrollo (SDK o *Software Development Kit*). Tanto el JDK como el JRE están disponibles para sistemas basados en Solaris, Microsoft Windows, Linux, Mac OS X, etcétera.

6.2. Instalación y configuración de los entornos

6.2.1. Java

Para comprobar si ya dispone de una instalación del entorno de ejecución de Java en el equipo y que está configurado correctamente, abra una ventana de comandos *(Inicio ->Ejecutar ->cmd)* y escriba «java -version». Debe obtener un resultado similar al de la figura 6.1:

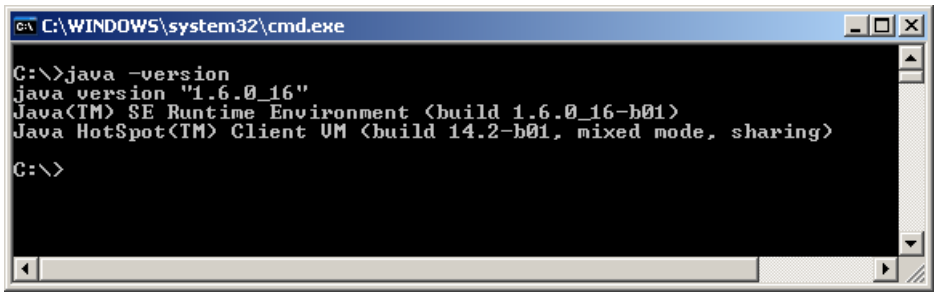

Figura 6.1: Cómo consultar la versión de Java instalada

Si aparece un mensaje de error, pruebe lo siguiente:

- Compruebe si tiene instalado Java. Para ello compruebe que exista el directorio Java en *C:\Archivos de programa* (típicamente). Si lo está, configúrelo como se indica más abajo.
- Si no existe, vaya a la siguiente dirección de Internet y descargue la última versión del Java SE Development Kit (JDK) [2]

Una vez descargado, ejecute el archivo, elija el directorio de instalación y siga los pasos para completar la instalación. Tras instalarlo, debe configurarlo. Para ello vaya a *Inicio* y haga clic derecho en *Mi PC*. En las opciones, elija la pestaña de *Opciones Avanzadas* y haga clic en *Variables de entorno*. En la nueva ventana que aparece, busque la variable *Path* y añada al final `;C:\Archivos`

`de programa\Java\jdk1.x.x_xx\bin`, donde «x» hace referencia a la versión que tiene instalada de Java. Ese es el directorio de instalación por defecto de Java. Si en la instalación eligió otro diferente, debe introducir el que eligió al instalarlo.

Una vez realizados los pasos descritos, pruebe a ejecutar de nuevo la sentencia «java -versión» en una ventana de comandos.

6.2.2. JXTA

Para comenzar, debe hacerse de una versión funcional de la librería de JXTA. Se proponen dos métodos (ambos requieren que visite la página de descargas de JXTA [3]):

- Descargue el fichero *jxse-src-2.5.zip*. Descomprímalo y compílelo con *ant* o *make*. Acto seguido se generará el fichero *jxta.jar* en el directorio *./jxse-src-2.5/dist/*.
- Descargue el fichero *jxse-lib-2.5.zip*. En su interior se encuentra el fichero *jxta.jar* ya compilado.

También puede descargar la documentación de JXTA (fichero *jxse-doc-2.5.zip*) o la guía de programación de JXTA para Java SE/EE (JXSE) 2.5 [6].

En versiones anteriores era necesario realizar una instalación de dicho motor, pero hoy en día basta con incluir la librería de JXTA en la aplicación y realizar una configuración del *peer*, bien sea en el código fuente de la aplicación o mediante un *configurator* gráfico externo (ver sección 6.3).

6.3. Cómo ejecutar aplicaciones JXTA

Para ejecutar una aplicación JXTA es necesario indicar donde se encuentran las librerías, bien incluyéndolas en el *classpath* del sistema operativo o añadiéndolas al proyecto, en el entorno de programación.

JXTA incluye una interfaz gráfica (ver Figura 6.2) que nos permite configurar el *peer*, aunque normalmente este paso se suele evitar ya que cada aplicación lo configura por código utilizando la clase *NetworkConigurator* o el *NetkworkManager*, que abstrae la configuración a una serie de configuracións predefinidas:

- **Ad-Hoc**: un nodo típicamente desplegado en una red ad-hoc, es decir, una red en la que no hay un nodo central y todos los dispositivos están en igualdad de condiciones. El *par* no utilizará una infraestructura de *pares* (servicios *rendezvous* o *relay*).
- **Edge**: este tipo de nodos se pueden unir a una infraestructura de *pares* (servicios *rendezvous* o *relay*).
- **Rendezvous**: proporiona servicios auto-suficientes (descubrimiento, etc.).
- **Relay**: proporciona servicios de mensajería, permitiendo atravesar corta-fuegos.
- **Proxy**: proporciona servicios de proxy.
- **Super**: proporciona la funcionalidad de un nodo *rendezvous*, *relay* o *proxy*.

Una vez que el *peer* ha sido configurado por alguno de los métodos mencionados, se creará automáticamente un directorio llamado *.jxta* en el directorio de la aplicación. Si se inspecciona dicho directorio, podrá encontrar un fichero llamado *PlatformConfig*, que es el fichero de configuración del par. Si desea volver a configurar el *par*, se recomienda que borre dicho directorio.

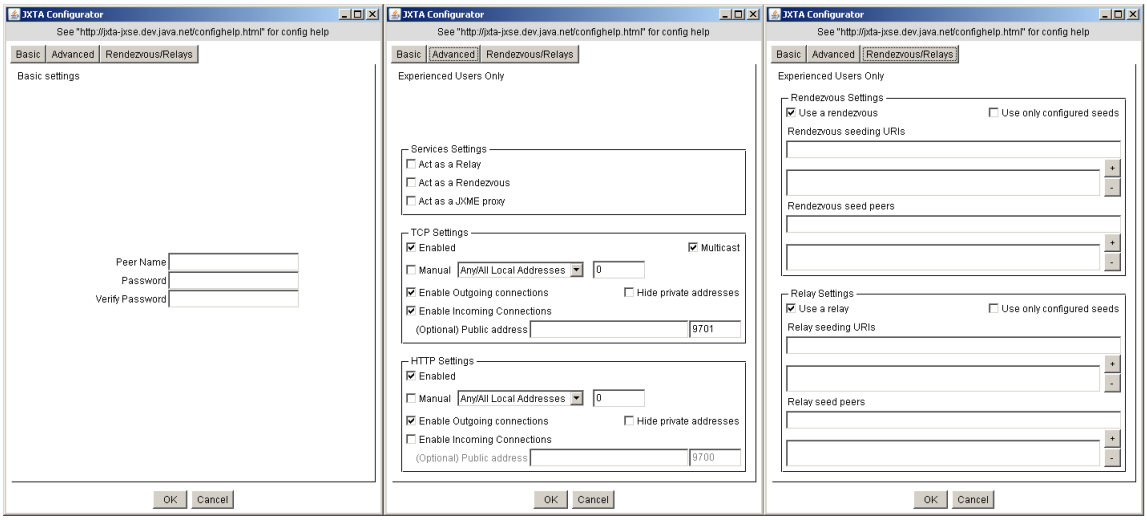

Figura 6.2: JXTA Configurator

6.4. Cómo utilizar el *Shell* de JXTA

Para familiarizarse con JXTA, se recomienda practicar primero con el *shell* que incorpora JXTA. De este modo se puede comprender el funcionamiento del sistema antes de adentrarse en el mundo de la programación. A continuación se explicará un sencillo ejemplo de cómo descubrir los pares del grupo por defecto, pero primero debe descargarse el shell de su página Web [4].

Una vez que haya descargado el shell, descomprímalo y ejecute el script *run-shell.sh* (Linux/MacOS) o *run-shell.bat* (Windows). Estos scripts incluyen automáticamente en su classpath la ubicación de las librerías de JXTA. Obtendrá una ventana como la que se muestra en la Figura 6.3. Si introduce la órden *'man'* obtendrá una lista con todas las órdenes que puede ejecutar en el *shell* junto a una breve descripción de cada una de ellas.

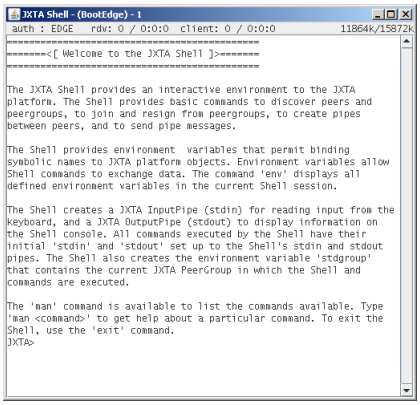

Figura 6.3: Shell de JXTA

Ejemplo práctico: Descubrimiento de pares

A continuación se realizará un sencillo ejemplo de cómo descubrir los pares que hay en el grupo. Se propone al lector que inicie un *shell* en dos ordenadores distintos, pertenecientes a una misma red, y ejecute las órdenes «peers» y «peers -r». Para obtener más información acerca de la orden «peers» puede ejecutar «man peers». Si observa la figura 6.4 para ver los resultados de este ejemplo.

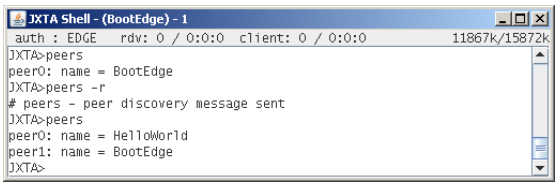

Figura 6.4: Ejemplo de descubrimiento de pares

Se propone al lector que haga uso de las órdenes *talk* y *xfer* para establecer una conversación de chat entre dos pares y para transferir ficheros, respectivamente. Utilice «man talk» y «man xfer» para obtener sus respectivas ayudas.

Por último, si no consigue que funcione alguno de los ejercicios propuestos, pruebe a deshabilitar el *firewall* de su equipo.

6.5. Ejemplo de un HelloWorld

A continuación se muestra el código Java correspondiente a un programa de tipo *Hola Mundo*. En este ejemplo se hace uso del *NetworkManager* para configurar el *par* en código, por lo que en ningún momento se mostrará el *JXTA UI Configurator*.

La sentencia `manager = new NetworkManager(ConfigMode.EDGE, "HelloWorld");` configura el *par* con una configuración por defecto: nodo de tipo *edge* y le asigna el nombre *HelloWorld*. Se podría personalizar esta configuración obteniendo el objeto *NetworkConfigurator* que encapsula el *NetworkManager* con una sentencia del tipo a `NetworkConfigurator configurator = manager.getConfigurator();`.

El cometido de la aplicación es sencillo:

1. Crea una instancia del *NetworkManager* e inicia la red JXTA.
2. Se pone a la escucha de conexiones durante 2 minutos.
3. Detiene la red JXTA.

Una vez que la ejecución de la aplicación termina, se puede comprobar el directorio de configuración que se ha creado (*.jxta*):

- **PlatformConfig**: el fichero de configuración creado por el *NetworkManager*.
- **cm**: es un directorio que contiene un subdirectorio para cada grupo que se ha descubierto. En este ejemplo, deberían aparecer los subdirectorios *jxta-NetGroup* y *jxta-WorldGroup*. Estos subdirectorios deben contener ficheros de índice (*.idx) y ficheros de anuncios (advertisements.tbl).

```
import java.text.MessageFormat;

import net.jxta.peergroup.PeerGroup;
import net.jxta.platform.NetworkManager;
import net.jxta.platform.NetworkManager.ConfigMode;

public class HelloWorld {

    public static void main(String args[]) throws Exception  {
        System.out.println("Starting JXTA ....");

        // Create, and Start the default jxta NetPeerGroup
        NetworkManager manager = new NetworkManager(ConfigMode.EDGE, "HelloWorld")
            ;

        // Connect to network (default group)
        manager.startNetwork();

        // Some info about the default group and the created peer
        PeerGroup peerGroup = manager.getNetPeerGroup();
        System.out.println("Hello from JXTA group " + peerGroup.getPeerGroupName()
            );
        System.out.println(" Group ID = " + peerGroup.getPeerGroupID().toString())
            ;
        System.out.println(" Peer name = " + peerGroup.getPeerName());
        System.out.println(" Peer ID = " + peerGroup.getPeerID().toString());

        System.out.println("Waiting for a rendezvous connection");
        boolean connected = manager.waitForRendezvousConnection(120000);
        System.out.println(MessageFormat.format("Connected :{0}", connected));

        System.out.println("Stopping JXTA");
        manager.stopNetwork();

        System.out.println("Good Bye ....");
    }
}
```

Notas de interés

1. Cada vez que se instancia el NetworkManager, se crea un nuevo identificador de *par* (JXTA PeerID), a menos que se haya cargado una configuración local.

2. Si se necesita cambiar la configuración local de un *par* (de EDGE a RENDEZVOUS, por ejemplo), se debe hacer mediante el NetworkManager utilizando el método *setMode()* **antes** de que se inicie la red.

7. Toolkits e implementaciones alternativas

Aunque la implementación de JXTA en Java es una buena herramienta para desarrollar aplicaciones P2P, no es la única alternativa de la que disponen los desarrolladores. Existen numerosas alternativas tanto de JXTA para otros lenguajes, como otra plataformas diferentes a JXTA.

7.1. IBM BabbleNet

AlphaWorks es un grupo de IBM que desarrolla y distribuye nuevas tecnologías, principalmente para desarrolladores. BabbleNet [1] es una de estas tecnologías y consiste en un programa P2P descentralizado que permite a sus usuarios construir un chat en tiempo real sin la necesidad de conectarse a un servidor. El sistema está desarrollado sobre un framekwork P2P escrito en Java que soporta comunicaciones entre nodos.

El código fuente puede examinarse una vez instalado el software. La documentación es mínima y los comentarios que hay en el código describen el sistema. Se trata de un software en fase experimental y con licencia *open-source*.

7.2. Intel

Intel también se encuentra dentro del mercado del P2P, y lo hace con su *Peer-to-Peer Acelerator Kit* [8]. Se trata de un *middleware* que se utiliza con Microsoft .NET.

7.3. Microsoft .NET

Microsoft ha desarrollado una página Web en la que se muestra cómo se pueden desarrollar sistemas P2P con la plataforma .NET [7]. Algunas de las características que destacan son la siguientes:

- Todo el código P2P se basa en la plataforma .NET.
- Los mensajes que se envían entre los *pares* son serializados como XML.
- Los *pares* pueden compartir y acceder a distintos objetos.
- También se ha implementado un servicio de descubrimiento utilizando .NET.

7.4. Peer-to-Peer Trusted Library

Peer-to-Peer Trusted Library (PtPTL) [10] es una biblioteca *open-source* cuyo objetivo es profundizar en la seguridad de los sistemas P2P. Está disponible tanto para sistemas basados en Windows como en sistemas basados en Linux. Sus principales características son las siguientes:

- Certificados digitales.
- Autenticación de pares.

- Almacenamiento seguro.
- Métodos de codificación simétricos y asimétricos.
- Firmas digitales.
- Tarjetas digitales.

Es importante apreciar que esta librería no es un *toolkit* para desarrollar sistemas P2P, si no que es una biblioteca que permite dotar de «confianza» a dichos sistemas.

Referencias

[1] Página Web de IBM BabbleNet. http://alphaworks.ibm.com/tech/babblenet?open&l=p2pt,t=gr.

[2] Página Web de descarga de Java. http://java.sun.com/javase/downloads/index.jsp.

[3] Página Web de descargas de JXTA. http://download.java.net/jxta/.

[4] Página Web de descarga del Shell de JXTA. http://download.java.net/jxta/jxta-jxse/2.5/jxse-shell/jxse-shell-2.5.zip.

[5] Página Oficial de JXTA. http://jxta.dev.java.net/.

[6] Guía de programación de JXTA para Java SE/EE (JXSE) 2.5. https://jxta-guide.dev.java.net/.

[7] Página Web de la iniciativa P2P de Microsoft .NET. http://gotdotnet.com/team/p2p.

[8] Página Web de P2P Acelerator Kit. http://cedar.intel.com/cgi-bin/ids.dll/topic.jsp?catCode=BYM.

[9] Wikipedia: Definición del término *peer-to-peer*. http://jxta.dev.java.net/.

[10] Página Web de Peer-to-Peer Trusted Library. http://sourceforge.net/projects/ptptl.