Instituto Tecnológico y de Estudios Superiores de Monterrey

CAMPUS MONTERREY

PROGRAMA DE GRADUADOS EN TECNOLOGÍAS DE INFORMACIÓN Y ELECTRÓNICA

TESIS

DISEÑO E IMPLEMENTACIÓN DE UN MODELO PUNTO-A-PUNTO PARA LA INTEROPERABILIDAD ENTRE SERVIDORES DE DATOS DE UNA BIBLIOTECA DIGITAL

PRESENTADA COMO REQUISITO PARCIAL PARA OBTENER EL GRADO ACADÉMICO DE:

MAESTRO EN CIENCIAS EN TECNOLOGÍA INFORMÁTICA

POR

EUGENIO FLORES CENTENO

MONTERREY, N.L. MAYO 2006

Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Monterrey
Programa de Graduados en Tecnologías de Información y Electrónica

Los miembros del comité de tesis recomendamos que la presente sea aceptada como requisito parcial para obtener el grado de **Maestro en Ciencias de Tecnología Informática**.

Comité de Tesis

Dr. Juan Carlos Lavariega Jarquín
Asesor Principal

Dr. David A. Garza Salazar
Sinodal

Dr. José Raúl Pérez Cázares
Sinodal

Dr. David A. Garza Salazar
Director del Programa de Graduados en Tecnologías de Información y Electrónica
Mayo de 2006

DISEÑO E IMPLEMENTACIÓN DE UN MODELO PUNTO-A-PUNTO PARA LA INTEROPERABILIDAD ENTRE SERVIDORES DE DATOS DE UNA BIBLIOTECA DIGITAL

POR

EUGENIO FLORES CENTENO

TESIS

Presentada a la División de Graduados en Tecnologías de Información y Electrónica

Este trabajo es requisito parcial para obtener el grado de Maestro en Ciencias en Tecnología Informática

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

MAYO 2006

Agradecimientos

Agradezco al doctor David Garza por haberme incluido en el grupo de estudiantes que conforman el proyecto PDLib, lo cual significó un importante apoyo económico para asumir los costos de mis estudios de maestría, pero principalmente valoro esta oportunidad por la experiencia adquirida en la realización de las tareas que involucran un proyecto de investigación.

También agradezco al doctor Raúl Pérez, cuyos certeros comentarios, me sieron de gran utilidad.

Por último, agradezco a todos los compañeros del proyecto, con quienes pude, en innumerables oportunidades, solventar dificultades técnicas, y compartir tiempo de esparcimiento. Y de manera muy particular a los compañeros que tenían asignadas tareas en el servidor de datos, con quiénes consulté muchas de las decisiones que involucraban el desarrollo del modelo expuesto en esta tesis. Ellos son:

Francisco Álvarez Cavazos,
Eduardo Álvarez Cavazos, y
Alejandro Morán Zea.

Dedicatoria

Dedico las horas invertidas en este trabajo a mi padre Carlos Flores, a mi madre Yadira Centeno, a mis hermanas Lucía y Carolina Flores, porque con ellos he comprendido el significado de la solidaridad.

También dedico este tiempo a María Guadalupe Wallace, por ser una persona como pocas.

Resumen

Como resultado de la digitalización de la información, y la necesidad que tienen los individuos de contar con ella sin importar horarios ni lugar, las bibliotecas digitales han surgido como sistemas de ordenamiento y acceso a dicha información.

El Centro de investigación en Informática (CII) del Instituto Tecnológico de Monterrey, Campus Monterrey, se encuentra desarrollando un sistema que aporta una posible solución a la necesidad de acceso universal a la información: *The Personal Digital Library with Universal Access* (PDLib), que al español se puede traducir como Biblioteca Personal de Acceso Universal. Este sistema provee a cada usuario con un repositorio general de documentos, el cual es accesible desde cualquier lugar y a cualquier momento desde cualquier dispositivo conectado a Internet, incluidos los dispositivos móviles.

A la fecha se cuenta con una versión del sistema que permite la creación y administración de repositorios con las características antes descritas, sin embargo, el almacenamiento físico y las operaciones administrativas recaen en un único servidor de datos. Y debido a la naturaleza móvil de los usuarios, al crecimiento esperado tanto de ellos como de la cantidad de documentos digitales a administrar; el soporte con un único servidor significa adoptar algunas de las desventajas de los sistemas centralizados, tales como la imposibilidad de escalar en almacenamiento más allá de la capacidad de una sola computadora, contar con un único punto de fracaso y la incapacidad de compartir recursos de cómputo.

Por tanto, es en el ámbito del cómputo punto-a-punto (P2P), donde se explora para proponer una solución a las limitantes previamente descritas.

El presente trabajo de tesis propone un modelo para comunicar instancias de una misma aplicación instaladas en diferentes computadoras, con el fin de compartir información y distribuir documentos digitales. Particularmente, el modelo se implementa en el módulo núcleo del sistema PDLib, bautizado con el nombre de *Data Server* (Servidor de Datos), el cual contiene la lógica de negocio del sistema y es el proveedor de servicios a los módulos clientes.

Por consiguiente, es a través de este modelo de interoperabilidad que PDLib se convierte en una aplicación con almacenamiento escalable, con capacidad para realizar búsqueda de documentos entre distintas instancias de su módulo principal, y por último, provee la capacidad de replicar información, lo que significa, copiar de un servidor a otro, los elementos que los usuarios estimen necesarios, eliminando de esta manera las desventajas inherentes de la arquitectura cliente-servidor previamente adoptada en PDLib.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

No se sabe con exactitud en que momento de la historia de la humanidad, los hombres tuvieron conciencia de la capacidad intrínseca de comunicarse con sus semejantes. Pero lo cierto es que a partir de esa noción, los seres humanos no han dejado de perseverar en la búsqueda de formas de comunicación que se impongan a los obstáculos de ubicación física, que pueden existir entre dos o más personas con aspiraciones, deseos o necesidad de comunicarse.

No son ajenas a la memoria, las épocas en donde el hombre buscaba vencer las distancias utilizando señales de humo, o recurriendo a aves mensajeras, para establecer vías de información entre poblaciones distantes. Y es a partir de la invención del telégrafo, cientos de siglos después de la utilización de estos primeros mecanismos, en que la materialización de los sueños de invención y de masificación de medios de comunicación ha sido completamente descollante.

Hoy en día, gracias a este progreso vertiginoso en cuyos logros fundamentales se encuentran la creación de la red mundial digital Internet y la telefonía móvil; la información tiene un valor vital para el desempeño productivo de los individuos en sus organizaciones, y la necesidad de contar con ella sin tomar en cuenta horarios ni distancias, ha propiciado la investigación tanto en el área de informática como de telecomunicaciones.

El Centro de Investigación en Informática (CII) del Instituto Tecnológico de Monterrey se encuentra desarrollando un sistema que aporta una posible solución a la necesidad de acceso universal a la información: *The Personal*

Digital Library with Universal Access (PDLib), que al español se puede traducir como Biblioteca Personal de Acceso Universal[14]. Este sistema provee a cada usuario con un repositorio general de documentos, el cual es accesible desde cualquier lugar y a cualquier momento desde cualquier dispositivo conectado a Internet, incluidos los teléfonos móviles y los PDA.

El término biblioteca digital se adopta porque la funcionalidad del sistema es una abstracción de la forma en como opera una biblioteca de textos impresos. PDLib, por tanto, tiene la capacidad de almacenar bancos de documentos con distintos tipos de formatos digitales, tales como texto, video, música, imágenes, etc., a la vez que cubre las tareas administrativas de clasificación, ordenamiento, búsquedas y acceso. Por ejemplo, la forma de localizar documentos sucede de igual manera como ocurre en una biblioteca tradicional, es decir, que se realiza por medio de los atributos de éstos, como nombre de autor, título, año, tema, contenido, etc., lo que se conoce como búsqueda a través de metadatos [8].

A la fecha se cuenta con una versión del sistema que permite la creación y administración de repositorios con las características antes descritas, sin embargo, el almacenamiento físico y las operaciones administrativas recaen en un único servidor de datos. Y debido a la naturaleza móvil de los usuarios, al crecimiento esperado tanto de ellos como de la cantidad de documentos digitales a administrar; el soporte con un único servidor significa adoptar algunas de las desventajas de los sistemas centralizados, tales como la imposibilidad de escalar en almacenamiento más allá de la capacidad de una sola computadora, contar con un único punto de fracaso, lo que se conoce en inglés como *single point of failure*, y la incapacidad de compartir recursos de cómputo.

Por tanto, es en el ámbito del cómputo distribuido, específicamente, dentro del cómputo punto-a-punto (P2P), donde se explora para proponer una solución a las limitantes previamente descritas. Es relevante, sin embargo, hacer ver que los sitemas cliente-servidor proponen mecanismos y estrategias para solventar las dificultades que esta arquitectura enfrenta, pero que están intimamente orientadas a aumentar las capacidades de la entidad servidor.

El presente trabajo de tesis propone un modelo para comunicar instancias de una misma aplicación instaladas en diferentes computadoras, con el fin de compartir información y distribuir documentos digitales. Particularmente, el modelo se implementa en el módulo núcleo del sistema PDLib, bautizado con

el nombre de *Data Server* (Servidor de Datos), el cual contiene la lógica de negocio del sistema y es el proveedor de servicios a los módulos clientes.

Por consiguiente, es a través de este modelo de interoperabilidad que PDLib se convierte en una aplicación con almacenamiento escalable, con capacidad para realizar búsqueda de documentos entre distintas instancias de su módulo principal, y por último, provee la capacidad de replicar información, lo que significa, copiar de un servidor a otro, los elementos que los usuarios estimen necesarios, reduciendo de esta manera, el riesgo de contar con un único punto de fracaso.

A estas alturas de la lectura, es relevante aclarar el significado del término principal que precisa el funcionamiento del modelo recién abordado, y que será utilizado a lo largo de las siguientes secciones y capítulos. De tal modo, que definiremos interoperabilidad, como la capacidad que tienen dos o más sistemas independientes de comunicarse entre sí y de forma transparente para los usuarios de los mismos.

Por otro lado, una de las ambiciones del proyecto PDLib es poder masificar la aplicación, llegando, por ejemplo, a brindar sus servicios a todos los alumnos del Campus Monterrey, de tal modo que remediar las limitantes inherentes del cómputo centralizado es necesario para poder conseguir este objetivo, y en consecuencia este trabajo de tesis cobra relevancia e importancia, a la vez que se pretende generar un aporte a la formulación de soluciones punto-a-punto en proyectos similares en el ámbito de bibliotecas digitales.

A lo largo de los siguientes capítulos se expondrán en detalle los elementos y conceptos expuestos en esta introducción y sobre los cuales gira la presente investigación:

- Bibliotecas Digitales
- Sistema PDLib y sus componentes
- Tecnología P2P
- Modelo de Interoperabilidad P2P PDLib.

1.1. Planteamiento del problema

A pesar de la reducción de los precios de las computadoras, los servidores de aplicaciones masivas, principalmente en Internet, todavía son muy costosos. Esto se debe a que mientras aumenta la cantidad de usuarios, debe de aumentar la capacidad de procesamiento y almacenamiento. Dado que la velocidad del CPU aumenta al doble cada dos años [10], ha proliferado el uso de *clusters* de servidores para poder obtener de múltiples CPU's la capacidad de procesamiento que permita satisfacer la demanda de centenas de miles de usuarios, y aún así hay ejemplos de servicios que no pueden sobrellevar la demanda del total de sus usuarios. Tomemos por ejemplo AOL-Time Warner[10], una compañía de servicios de Internet y comunicaciones de Estados Unidos, la cual cuenta con más de 30 millones de clientes, sin embargo sólo 2 millones pueden estar en línea simultáneamente debido a las limitaciones de hardware.

Pero más allá de esas limitaciones, el problema radica en la proliferación de la arquitectura de cómputo cliente-servidor, en la cual, las aplicaciones clientes son entidades que se limitan a solicitar información procesada por el servidor. Lo que ocasiona que el peso del procesamiento recaiga en una de las entidades que componen el total del sistema. El escenario ideal sería, que la carga del procesamiento de las aplicaciones sea asumida por un conjunto de computadoras autónomas comunicadas a través de Internet o cualquier tipo de red; de tal manera, que los recursos de procesamiento, y almacenamiento no dependan de una entidad aislada y costosa, y sí, de un conglomerado de computadoras aportando cada una su capacidad cómputo.

Otros problemas, ya mencionados, que surgen en la centralización de recursos son los que se identifican a continuación:

- Único punto de fracaso, que significa, no ser tolerante a fallas.
- Escalabilidad de almacenamiento limitada a la capacidad del servidor.
- Detrimento del desempeño a medida que aumenta la demanda de usuarios.

Y en el caso particular de la implementación PDLib, cuyo nivel de avance hace impostergable la instauración de una estrategia que resuelva

los problemas del almacenamiento (escalabilidad) de grandes volúmenes de objetos digitales; del transporte o migración de información según la ubicación física de los usuarios, es decir, que éstos cuenten con la facilidad de colocar su información en un servidor de datos próximo a ellos, consiguiendo de este modo reducir los tiempos de respuestas (desempeño) de solicitudes remotas, a la vez que esta migración puede ser empleada como una estrategia para mantener esa información disponible en el eventual caso de que un servidor de datos salga de línea (tolerancia a fallas). Y por último, se busca ampliar la gama resultados y opciones de uso de los servicios ofrecidos por un solo servidor de datos, como por ejemplo, los servicios de búsqueda de documentos y/o colecciones.

Mientras no se implemente un modelo competente que solucionen estas dificultades, el ahnelo de masificación del sistema no será posible, y se mantendrá como una aplicación para el uso reducido de un pequeño número de usuarios, en un área física limitada.

1.2. Organización de la tesis

Este documento está organizado en cinco capítulos. En el capítulo dos se presentan los antecedentes, la exposición de los conceptos sobre los cuales se basa esta investigación, las tecnologías empleadas en la solución de problemas similares, y la tecnología utilizada en el modelo de distribución. En el capítulo 3 se explica a detalle el modelo de interoperabilidad entre Data Servers, posteriormente en el capítulo 4 se expondrá el funcionamiento del modelo en un entorno operativo real; y por último, en el capítulo 5 se encuentran las conclusiones y el trabajo futuro.

Capítulo 2

Antecedentes

En este capítulo se definirán a detalle los conceptos y se describirán ejemplos de los temas que sirven de base para el desarrollo de esta investigación, y que fueron mencionados brevemente en el capítulo introductorio: Bibliotecas digitales, Proyecto PDLib y Tecnología punto-a-punto.

2.1. Bibliotecas digitales

La palabra biblioteca tiene su origen en dos voces griegas: biblio, que significa libro, y tekes, que quiere decir, caja. Por consiguiente, una biblioteca, partiendo de sus voces de origen, es un espacio donde se guardan textos. Tradicionalmente se interpreta como una colección privada o pública de libros, pero generalmente se relaciona con inmensas colecciones de textos administradas por instituciones privadas o públicas.

Con el devenir de la computación, la información creada en medios electrónicos se fue guardando en archivos con diferentes formatos digitales, que a su vez son depositados en algún dispositivo de almacenamiento accesible a través de interfaces de software.

Una biblioteca digital, en resumidas cuentas, puede ser vista como una abstracción del concepto original de biblioteca en el ámbito de información electrónica. Conceptualmente, es una colección organizada de documentos digitales que ofrece diversos servicios a los usuarios, como envío, clasificación,

búsqueda, recuperación y administración. Facilitando actividades de estudio e investigación colaborativa entre usuarios distribuidos geográficamente [17].

La descripción de los servicios tradicionales de una biblioteca digital se muestra a continuación [8]:

Almacenamiento:

Capacidad de acopiar información digital en una gran variedad de formatos digitales. Los documentos de texto con formato como ASCII, LaTex, HTML, SGML, y PostScript, son los más fáciles de almacenar, en cambio los documentos de multimedia, como audio y video, son los que presentan mayor dificultad de almacenamiento, debido a que demandan mayor espacio y procesamiento.

Las bibliotecas digitales pueden utilizar tanto el sistema operativo, o algún manejador de base de datos, como mecanismos de acopio de archivos.

También se auxilian de técnicas de compresión de datos para la optimización del uso del espacio de almacenamiento.

Interfaz de usuario:

Tal vez el componente más importante de una biblioteca digital. Debe de incorporar una gran variedad de técnicas que permitan una interacción ideal entre los usuarios y la información que les interesa encontrar.

La interfaz de usuario de una biblioteca digital debe desplegar grandes volúmenes de información de forma efectiva.

Clasificación e indexamiento:

Esquemas de clasificación e indexamiento son utilizados para recuperar contenido relacionado en grupos, de tal forma que esta presentación sea intuitiva para el usuario

En particular, el indexamieno permite la obtención de información rápidamente luego que una consulta es ejecutada.

Los sistemas de bibliotecas digitales clasifican sus recursos a través de metadata. Este término tiene su origen del griego meta (su propia

categoría) y del latín data (información). Literalmente significa "información sobre información", y no es más que una estrategia para clasificar recursos empleados especialmente en catálogos de bibliotecas.

Recuperación de información:

En el dominio de bibliotecas digitales, existen varias técnicas para recuperar información. Ejemplos de éstas son: búsqueda a través de metadatos [8], búsqueda completa en el texto del documento, y búsqueda de contenido para otros tipos de datos.

La medida del éxito de una solicitud para recuperar cierta información es el porcentaje de información útil que es regresada. Este porcentaje es medido exclusivamente por los usuarios de la biblioteca digital, dado que son ellos los que valoran que información les es beneficiosa.

Entrega de contenido:

Se refiere a la manera en que un objeto de interés es entregado al usuario. Si es pequeño, como un archivo de texto de 100 páginas, o una imagen de 50 Kbytes, éste puede ser transferido por el mismo medio en que se solicitó el documento. Pero si el contenido es muy grande, como archivos de video, el traspaso se debe de realizar auxiliándose de otros medios, como por ejemplo, a través de la infraestructura de comunicación (anchos de bandas) disponible en los sistemas de video conferencia.

Presentación:

La presentación del contenido es dependiente del poder de cómputo del dispositivo desde donde se accede a la biblioteca digital. En la sección 2.2.1 se presenta una clasificación de las aplicaciones clientes de acuerdo a la capacidad de cómputo de los dispositivo donde residen, y el medio de ingreso al servidor de datos asignado para proveer servicios de biblioteca digital.

Administración:

Las tareas fundamentales que un sistema de biblioteca digital debe ejecutar son:

- Administración de permisos de acceso a los documentos, o secciones de la biblioteca digital.
- Protección de la identidad de los usuarios.

Lo que implica, que la administración de un sistema digital aborda las tareas relacionadas con la seguridad de acceso a los recursos y al mantenimiento de versiones de estos recursos.

2.2. PDlLib: Biblioteca digital personal con acceso universal

El Centro de Investigación en Informática del ITESM propone un sistema para construir bibliotecas digitales, que además de proporcionar todos los servicios encontrados en los sistemas tradicionales de esta índole (descritos en la sección anterior), adiciona los conceptos de acceso universal y biblioteca personal a la arquitectura del sistema.

El adjetivo personal, agregado al concepto de biblioteca digital, expresa que a cada usuario se le proveerá con un repositorio de propósito general (una biblioteca digital). Y es de acceso universal, porque la manipulación del repositorio podrá acontecer desde cualquier dispositivo electrónico conectado a Internet, incluyendo teléfonos móviles, PDA y computadoras portátiles, de tal forma que la información puede ser obtenida "desde cualquier lugar, a cualquier hora"[14].

El resultado de esta formulación es el sistema PDLib, acrónimo del inglés *Personal Digital Library with Universal Access*, que en español significa Biblioteca Personal de Acceso Universal.

Para poder cumplir con los conceptos inherentes al nombre del sistema (Biblioteca digital, de índole personal, y con acceso universal) los siguientes requerimientos fueron formulados [14]:

- **Manejo Flexible de colecciones y *medatata***. Las colecciones son un mecanismo de clasificación de documentos. A los usuarios se les debe otorgar la facilidad de definir el conjunto de metadatos que serán utilizados para describir el contenido de cada colección.

- **Adición de documentos digitales**. Los usuarios deben tener la capacidad de agregar todo tipo de objetos digitales a su biblioteca personal.
- **Búsquedas y obtención**. Los mecanismos de búsquedas y obtención deben ser adaptados al esquema de clasificación definido por el usuario del sistema.
- **Acceso universal**. El usuario debe tener la facultad de acceder a su información al momento deseado y desde el lugar deseado, utilizando cualquier tipo dispositivo electrónico conectado a Internet.
- **Administración y control de ingreso**. El dueño de una biblioteca digital debe tener acceso irrestricto a su contenido, asimismo podrá compartirlo con otros usuarios a través de la expedición de diferentes niveles de admisión.
- **Interoperabilidad**. Comunicación con otros usuarios de la biblioteca personal y con otros sistemas de bibliotecas digitales haciendo uso de protocolos bien conocidos.

2.2.1. Arquitectura del sistema PDLib

Para poder cumplir con los requerimientos recién numerados se concibió una arquitectura de tres capas, tal y como lo refleja la figura 2.1, y la descripción que continúa.

- **Capa Cliente**. Incluye el ingreso desde cualquier tipo de dispositivo desde donde un usuario puede interactuar con el sistema.
- **Capa Servidor**. Corresponde a la infraestructura de servidores que provee de servicios a los clientes: *Data Server* (Servidor de Datos), *Mobile Connection Middleware* (MCM) y *Web Front-end*.
- **Capa de Interoperabilidad**. Incluye a otros servidores de datos y a otros sistemas de bibliotecas digitales compatibles con el protocolo OAI-MHP[13].

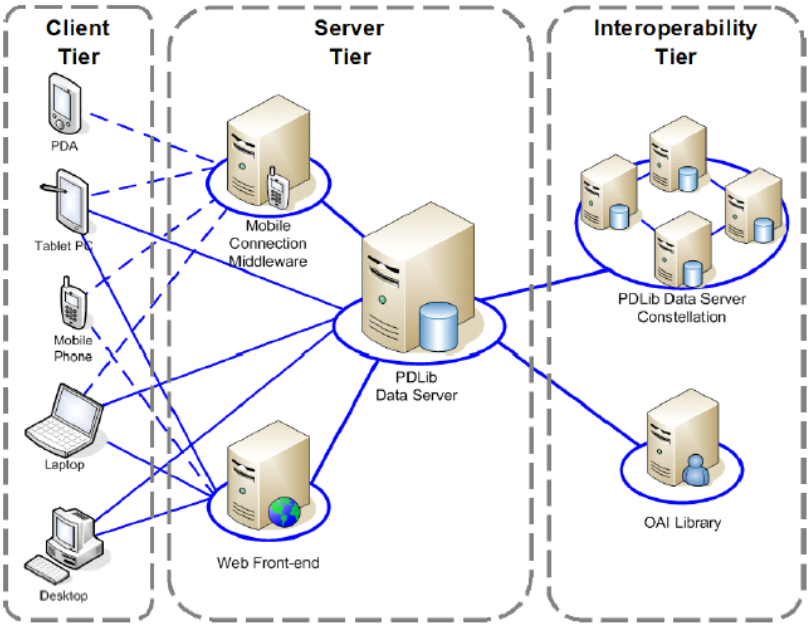

Figura 2.1: Arquitectura PDLib

Es precisamente en esta última capa donde radica el desarrollo del presente trabajo de tesis, el cual consiste en proponer e implementar un modelo de interacción entre varias instancias del servidor de datos.

La arquitectura también contempla el establecimiento de los medios de ingreso en concordancia con las características de los dispositivos clientes. Estos medios son descritos a continuación:

- **Ingreso a través del *middleware***. Soporta dispositivos móviles, especialmente aquellos con recursos limitados, como lo son los teléfonos celulares o las agendas personales tales como las PDA o las Pocket PC. Esta interacción se lleva a cabo por medio del paso de mensajes que proporciona el protocolo de comunicación XmlRpc[20].
- **Ingreso Web**. Se realiza a través del protocolo HTPP desde cualquier dispositivo que cuente con un navegador Web.
- **Ingreso directo**. Ejecutado desde aplicaciones residentes en clientes pesados [18], capaces de tener una interacción directa con el servidor

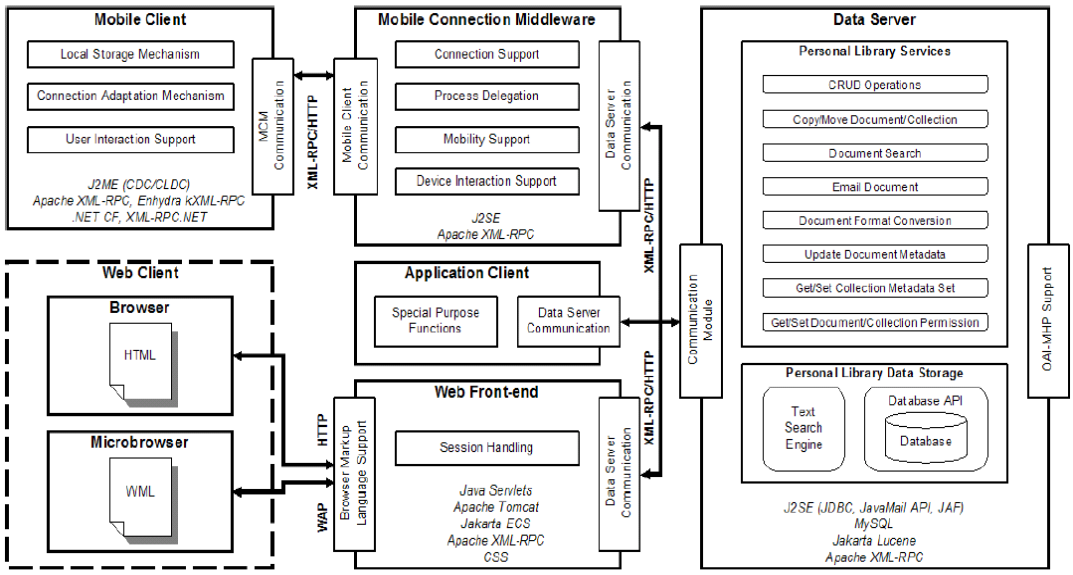

Figura 2.2: Arquitectura Interna del Sistema PDLib

de datos mediante XmlRpc.

La figura 2.2 muestra a detalle la composición de cada una de las capas que constituyen el sistema. La presentación minuciosa de los elementos que las forman será el objetivo de las secciones posteriores.

2.2.2. Componentes de la Capa Cliente

Del tipo de acceso a los servicios definidos en la sección anterior surge la clasificación de las aplicaciones clientes.

- **Clientes Pesados (*Thick clients*)**. Aplicaciones que corren en dispositivos con suficiente capacidad de cómputo, como las computadoras personales de escritorio (PC), o computadores personales portátiles. Por tanto, no necesitan valerse de un intermediario para interactuar con el servidor de datos. Lo que conlleva a que las aplicaciones clientes residentes en estos dispositivos mantengan una conexión directa con él.

- **Clientes Web (*Thin Client*)**. Los dispositivos que cuentan con un navegador, o micro navegador Web tendrán la facilidad de conectarse a un servidor Web, que servirá de intermediario entre el servidor de datos y el usuario.
- **Clientes Móviles**. Son aplicaciones desarrolladas para ambientes móviles, de tal modo que afrontan las limitaciones de recursos de este tipo de dispositivos, por lo cual se auxilian del MCM para lograr comunicarse con el servidor de datos.

Esta división contrasta las características particulares de cada tipo de aplicación. Hace notar que los clientes pesados cuentan con los recursos necesarios para realizar tareas demandantes, como presentar el total de documentos de una colección numerosa, o descargar archivos de varios MB de tamaño. En cambio, los clientes Web están diseñados para acceder a los recursos de PDLib sin la necesidad de tener instalada una aplicación del tipo cliente pesado.

Por otro lado, los Clientes Móviles requieren de mecanicismos de auxilio que les permitan brindar los servicios del servidor de datos, sobrellevando las limitaciones de recursos de cómputo inherentes. El objetivo es presentarle al usuario una abstracción de su biblioteca personal y sus servicios.

Los mecanismos utilizados para lograr este propósito son los siguientes:

- **Almacenamiento local**. Permite acumular objetos de la biblioteca personal obtenidos desde el servidor de datos en la memoria del dispositivo.
- **Cálculo automático del tamaño de página**. Realiza la predicción del estado de la red para la adaptación de la información a comunicar.
- **Interfaz de usuario**. Interfaz gráfica que permite a un usuario interactuar con los servicios ofrecidos desde el servidor de datos.

2.2.3. Componentes de la Capa Servidor

***Data Server* (Servidor de Datos)**

Es el núcleo de sistema PDLib. Proporciona los servicios de una biblioteca personal digital, como el almacenamiento de objetos característicos (colecciones, docuementos y *metadata*). Soporta interoperabilidad con otras aplicaciones de bibliotecas digitales similares a través del protocolo OAI-MHP. Y a partir del trabajo desarrollado en la presente investigación es capaz de interactuar con otras instancias de servidores de datos instaladas en una misma máquina, o en diferentes computadoras.

El servidor de datos presenta la siguiente funcionalidad:

- **Servicios de una biblioteca personal digital**. Ofrece operaciones para crear, actualizar, descargar, y eliminar objetos de la biblioteca (colecciones, documentos y metadatos). También otorga servicios para copiar y mover documentos o colecciones, realizar búsquedas de documentos o colecciones, enviar documentos a la biblioteca personal de otros usuario, mandar documentos vía correo electrónico, convertir el formato de los documentos, definir el cúmulo de metadatos de las colecciones y los documentos, y otorgar o remover permisos sobre los diferentes objetos de la biblioteca.
- **Almacenamiento personal de datos**. Guarda los datos de la biblioteca personal digital en forma de objetos de datos. Para indexar el contenido de estos objetos se utiliza una herramienta de obtención de información en texto. El almacenamiento se realiza en una base datos, debido a se requiere un modelo estructurado para representar objetos personales y la relación entre ellos.

Web Front-end

Es el punto de acceso Web a los servicios expuestos en el servidor de datos. Está diseñado para dispositivos que cuenten con un navegador Web, y es capaz de entregar WML, o HTML según sea el dispositivo de donde se origine la petición.

El Web Front-end suministra la siguiente funcionalidad:

- **Manejo de sesiones**. Es utilizada para la interacción con el *Web Front-end* más allá de una simple solicitud HTTP.

- **Distinción de lenguajes de presentación de texto**. Verifica que tipo de lenguaje de presentación soporta el navegador que ejecuta las peticiones, y en dependencia del resultado, muestra las respuestas con HTML o WML.

Mobile Connection Middleware (MCM)

El servidor de datos fue diseñado para hacerle frente a las solicitudes provenientes de aplicaciones residentes en una gran variedad de dispositivos. Sin embargo, dadas las diferencias en recursos computacionales que se presentan entre los tipos de clientes expuestos en la sección 2.2.2, resulta complicado idear un diseño que se adapte a las necesidades de cada grupo de clientes. Por consiguiente, un *middleware* se propone como mediador entre los servicios ofrecidos por el servidor de datos y las solicitudes elaboradas desde los dispositivos de escasas capacidades.

Mobile Connection Middelware (MCM) es el nombre para el intermediario ideado como solución al problema planteado al inicio de esta sección, y tiene bajo su cargo las siguientes responsabilidades:

- **Soporte a la conexión**. Es el acervo de funciones que ayudan al cliente móvil a adaptarse al entorno variable y limitado de las conexiones inalámbricas.
- **Delegación de procesos**. Ejecuta procesos que demandan una excesiva (y posiblemente no disponible) cantidad de recursos del dispositivo móvil. Así como el soporte a la interacción fuera de línea.
- **Soporte a la interacción con el usurario**. Realizar la adaptación del contenido dependiendo de las características del dispositivo en el que se desea mostrar la información

2.2.4. Componentes de la Capa de Interoperabilidad

Si bien es cierto que los componentes de esta capa físicamente se encuentran desarrollados dentro del servidor de datos, los recursos a los que se accede a través de los mecanismos de interoperabilidad se encuentran en otros sistemas de bibliotecas digitales. De tal modo que son estos sistemas

los que forman la capa definida como Interoperabilidad. A través de ella se logra:

- **Soporte del protocolo OAI-MHP [13]**. El servidor de datos expone los metadatos de los documentos de la biblioteca personal digital por medio del protocolo OAI-MHP, a la vez que es capaz de acceder y hacer disponibles los servicios ofrecidos por aplicaciones remotas e independientes que sean compatibles con en este protocolo.
- **Interacción con otros Servidores de Datos PDLib**. Representa la habilidad de formar grupos de servidores de datos, con el ánimo de compartir servicios y recursos, teniendo como propósito los siguientes objetivos.

  1. Ampliar la gama de resultados y opciones de uso de los servicios ofrecidos por un solo servidor de datos.
  2. Tener acceso a un inmenso volumen de recursos (objetos de una biblioteca personal) distribuidos en innumerables servidores de datos conectados a Internet.
  3. Escalar en el almacenamiento de objetos digitales, más allá de la capacidad de una computadora.
  4. Replicar información, de tal forma que la información se mantenga disponible en el eventual caso de que un servidor de datos salga de línea (evitar contar con un único punto de fracaso).
  5. Migrar información para ubicarla estrategicamente según sea la ubicación física y el ámbito de trabajo de cada usuario. Mejorando, de este modo, la calidada en el desempeño del sistema.

  Es en la creación de un modelo punto-a-punto, que lleve a la realidad estas pretensiones, en lo que se centra esta investigación. La presentación y el resultado de la implementación de este modelo son los objetivos de los 2 subsiguientes capítulos, mientras que el fundamento teórico y la explicación del porque de la adopción de una arquitectura punto-a-punto para crear estas facultades, son los temas que se abordarán en las restante secciones del capítulo.

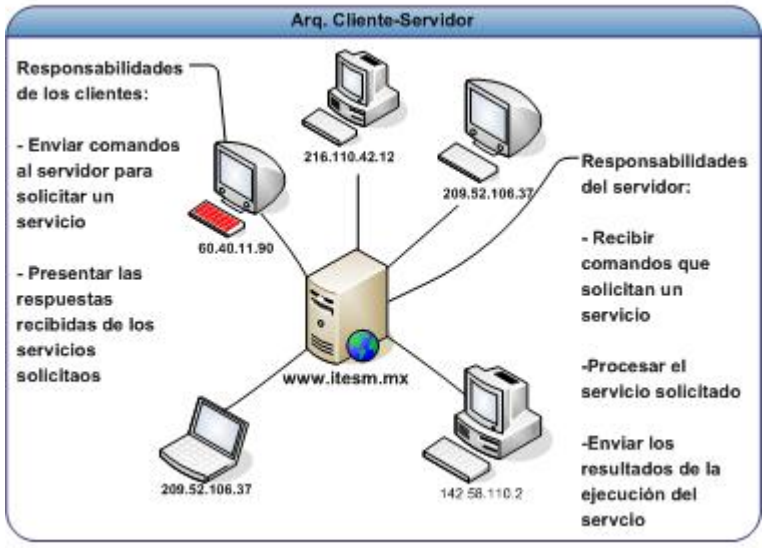

Figura 2.3: Arquitectura Cliente-Servidor

2.3. Modelo de computación Punto-a-Punto (P2P) *(Peer-to-Peer Paradigm)*

El término punto-a-punto se acuñó a mediados del año 2000 [16] como una calificación a un estilo de computación paralelo al modelo cliente-servidor (c-s), el cual ha sido el estándar de desarrollo de sistemas expuestos en Internet.

Básicamente, el modelo cliente-servidor, ilustrado en la figura 2.3, se compone de una entidad que inicia, centraliza y controla todos los servicios ofrecidos por una aplicación, y otra que simplemente se limita a solicitar la ejecución estos servicios.

Por otro lado, el modelo punto-apunto, o de usuario-a-usuario, gráficamente apreciable en la figura 2.4 , implica que las partes involucradas tienen responsabilidades idénticas. Extrapolándolo al modelo cliente-servidor se podría decir entonces, que en cualquier momento un cliente puede convertirse en servidor, y un servidor asumir el rol de cliente.

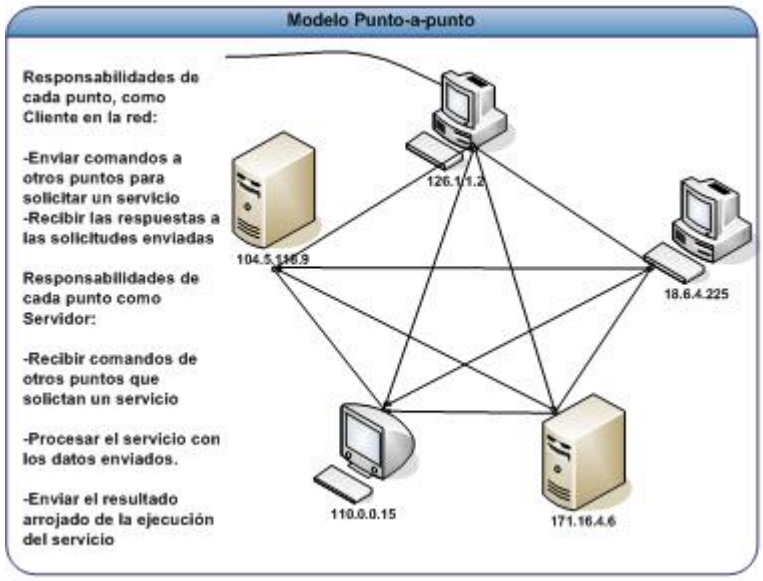

Figura 2.4: Arquitectura Punto-a-punto

2.3.1. ¿Por qué surge el modelo punto-a-punto?

A pesar de que el modelo cliente-servidor ha sido adoptado como el paradigma de facto para el diseño y desarrollo de sistemas distribuidos, estos sistemas se encuentran constreñidos por las limitaciones inherentes al modelo:

- **Dependencia del ancho de banda**. La cantidad límite de clientes posibles que un servidor puede atender depende del ancho de banda disponible.
- **Dependencia de los recursos de hardware de la computadora donde reside el servidor**. Mientras más grande es el número de clientes mayor es el consumo de recursos computacionales, tales como operaciones de CPU, RAM, y almacenamiento secundario.
- **Único punto de fracaso**. El total de las aplicaciones clientes dependen de la disponibilidad del servidor. Al momento en que falle, ningún cliente podrá obtener servicios del servidor.

Hoy en día, las industrias y los negocios no pueden darse el lujo de operar sus sistemas con estas limitaciones. No se tolera, por ejemplo, que un sistema

bancario o un sistema de reservación aéreo no se encuentren disponibles para atender la demanda del total de los clientes que requieren del servicio.

Por tanto, han surgido estrategias para sobrellevar las debilidades del modelo cliente-servidor. Algunas de ellas han apostado por el aumento de la capacidad de cómputo de las computadoras servidoras y el mejoramiento de la infraestructura de comunicación.

Pero mientras el número de usuarios en Internet siga creciendo, la capacidad de los servidores deberá seguir creciendo. Debido a que la velocidad del CPU es duplicada cada dos años [10], ha proliferado la investigación y el desarrollo de cluster de servidores que obtienen de múltiples CPUs el procesamiento que permite satisfacer la demanda de miles de usuarios.

También han surgido diferentes modelos de respaldo que hacen frente a una eventual caída del nodo servidor, un riesgo siempre latente. Una solución exitosa en este tipo de situaciones es contar con un servidor de respaldo en línea, el cual mantiene una copia simultánea de los datos y los servicios del servidor principal. Al momento que ocurra una falla en el servidor principal, el de respaldo asume la responsabilidad.

Estas estrategias, sin embargo, resultan muy costosas, y en realidad no eliminan las limitaciones que el modelo cliente-servidor padece. Lo que se logra realmente es un aumento en el número techo de clientes que los servidores pueden asumir, y un incremento en el nivel de disponibilidad de los servicios. Porque a pesar de que un servidor cuente con un respaldo en línea, eventualmente puede ocurrir que este respaldo también falle.

El modelo punto-a-punto, surge entonces, como una alternativa de desarrollo de aplicaciones distribuidas libres de las limitaciones y los costos económicos de las soluciones del modelo cliente-servidor.

2.3.2. Descripción de los sistemas punto-a-punto

Actualmente es difícil encontrar una PC con menos de 800MHZ de CPU, 128MB de RAM, y 20GB de disco duro. Las computadoras cada vez son más rápidas, tiene mayor capacidad de almacenamiento, y se vuelven más baratas. Las PCs de hoy en día son más poderosas que los servidores utilizados tan sólo unos años atrás.

Las aplicaciones clientes en el modelo c-s no necesitan consumir grandes

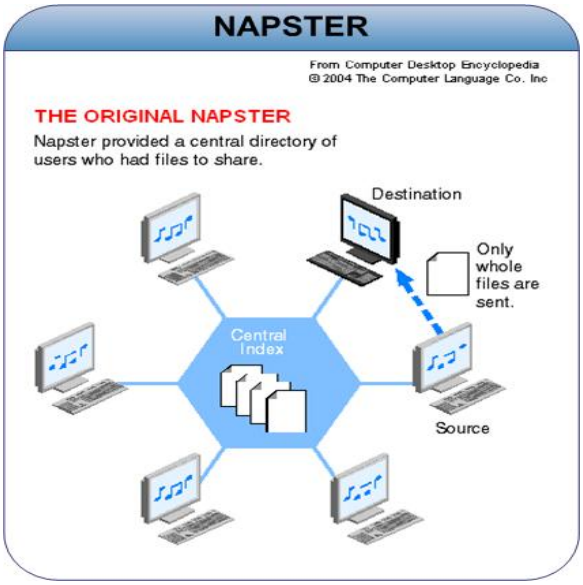

Figura 2.5: Operación del Sistema Napster

cantidades de recursos, porque, como ya se ha mencionado, todo el peso del procesamiento recae sobre los hombros del servidor. Esto significa que las computadoras clientes son subutilizadas por estos sistemas.

Las aplicaciones y los protocolos P2P utilizan los recursos de los nodos individuales que colectivamente forman el sistema P2P. De tal manera que el total de recursos de cómputo es la suma de los recursos de cada punto. Dependiendo de la cantidad de nodos involucrados en el sistema, la suma del total de recursos puede sobrepasar fácilmente las capacidades de los más potentes servidores. Esto da como resultado sistemas cien por ciento escalables.

Los sistemas P2P pueden ser diseñados con diferentes estrategias de distribución. Por ejemplo Napster [7](Figura 2.5), un servicio punto-a-punto ampliamente utilizado para compartir archivos de multimedia, indexa en un nodo central la información que identifica a los puntos y sus recursos, mientras que la transferencia de archivos ocurre sin la intervención de este punto principal, como se refleja en la figura 2.5. Los usuarios buscan archivos en el nodo central, el cual regresa los puntos desde donde éstos pueden ser transferidos. La ausencia de la entidad central limita la operación del sistema.

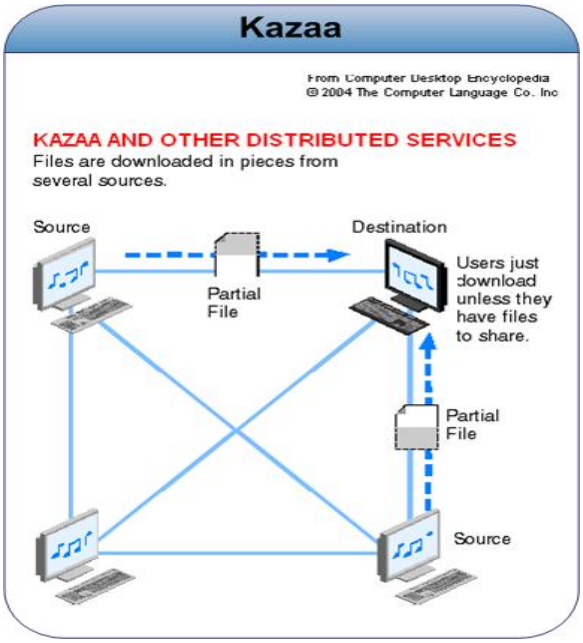

Figura 2.6: Operación del Sistema Kazaa

Por otro lado, Kazaa [5], que es otro sistema punto-a-punto destinado para intercambiar archivos multimedia, mantiene distribuido el total de sus recursos y servicios tal y como se puede observar en la figura 2.6. Ningún punto, de ser removido, afectará significativamente el desempeño del sistema.

A los sistemas que tienen prorrateados los recursos y servicios se les puede considerar P2P puros. En cambio los que no son totalmente distribuidos, los calificaremos como híbridos. Entonces, resulta observable que sólo a los de estilo P2P puro se les puede considerar sistemas altamente disponibles.

Los sistemas punto-a-punto incluyen la mayoría de las siguientes características [10]:

- Cada punto tiene conciencia de la disponibilidad de otros puntos.
- Los puntos crean redes virtuales a pesar de las complejidades físicas de interconexión que hay entre ellos. La red es creada a pesar de la existencia de corta fuegos *(firewalls)*, y del lugar donde físicamente residan.

- Cada punto puede actuar tanto como de cliente, como de servidor.
- Los puntos forman comunidades de trabajo que pueden ser descritas como grupos de puntos.
- El desempeño del sistema tiende a aumentar a medida que más nodos o puntos son agregados.
- El direccionamiento ocurre en tiempo real.

Si bien es cierto que el desempeño en los sistemas P2P tiende a incrementar a medida que se adicionan más puntos, también lo es que el desempeño es dependiente de la topología de red. La topología de red es delineada por el total de nodos comunicados y sus respectivos anchos de bandas [10].

Pueden ocurrir situaciones donde algunos nodos, envíen una exagerada cantidad de mensajes, que terminen saturando una porción importante de la red. Una muestra simple de cómo esta situación se puede presentar, es una aplicación de búsqueda. Si se propaga (se envía a todos los puntos) una solicitud por la red, este mensaje puede ser enviado a los mismos nodos en múltiples ocasiones, causando una disminución en el ancho de banda.

Para prevenir este problema varias técnicas son sugeridas [10]:

- Los mensajes deben ser enviados un número fijo de veces.
- Guardar localmente los datos que son solicitados y encontrados, de tal manera que no sea necesario buscar remotamente la misma información.
- Establecer en los mensajes un tiempo límite de vida con el fin de que no puedan navegar ilimitadamente en la red.
- Utilizar eficientemente las capacidades de los nodos. Lo que reduce la carga en nodos con ancho de banda limitado o escaso poder de procesamiento.

Varias tecnologías para la construcción de aplicaciones P2P han sido desarrolladas. En la sección que continúa se presentan algunas de ellas.

2.4. Tecnologías para el desarrollo de aplicaciones P2P

Todos los componentes del sistema PDLib fueron elaborados en ambientes de desarrollo Java. Para poder dotarlos con las competencias de los sistemas P2P, se advierte entonces, que se requieren tecnologías compatibles con Java.

En el mercado existen dos [9] tecnologías para producir sistemas P2P en ambientes Java: JRRA y JXTA. Cada tecnología, también definida como plataforma o *framework*, se constituye por un conjunto de clases que pueden ser importadas en códigos que serán compilados y ejecutados por la máquina virtual de Java. Las aplicaciones, resultan entonces, de adherir las funcionalidades de, una u otra plataforma, al grupo de clases de Java.

JRRA está sumamente orientado hacia dispositivos móviles. Permite la subsistencia de aplicaciones en poca RAM y con pocos ciclos de CPU. También hace frente a las debilidades intrínsecas de las redes donde estos dispositivos pueden residir, tales como latencia, cambios impredecibles en la topología, y pérdida de paquetes.

Por otra parte, JXTA es más genérico porque está dirigido a abarcar un mayor ámbito de operación. Está orientado a cubrir una vasta variedad de plataformas de hardware, redes, y lenguajes de programación.

El servidor de datos, como ha sido explicado precedentemente, es una aplicación diseñada y desarrollada para residir en computadoras personales o servidores de gran poder de cómputo. Por esta circunstancia fundamental, JXTA fue la plataforma escogida para implementar la interoperabilidad entre instancias independientes del servidor de datos.

A continuación presentaremos brevemente las particularidades generales de JRRA, y nos adentraremos a mayor profundidad en JXTA, debido que es necesario precisar cuales son los componentes de esta plataforma para poder comprender la manera en que fue implementado el modelo propuesto a lo largo del capítulo 3.

2.4.1. JRRA *(Rocky Road)*

JRRA es un protocolo que otorga servicios para la elaboración de aplicaciones P2P. Estas pueden ser de tipo estándar, como las que corren PCs, pero principalmente microediciones, que son aquellas que se ejecutan en PDAs y en los teléfonos celulares.

Ofrece los siguientes servicios [9]:

- **Búsqueda e indexación**. Realiza la tarea de buscar en indexar recursos en la red. Crea una jerarquía basada en la información personal de cada nodo de la red.
- **Direccionamiento y seguridad**. Provee mecanismos de autenticación a través de la asignación de direcciones y llaves.

2.4.2. JXTA

La principal fuente de información sobre la cual nos basamos para explicar los conceptos y componentes de esta tecnología es [19], de tal manera que se recomienda consultar esta fuente en caso de que se desee ampliar los temas aquí expuestos.

JXTA es el acrónimo de la voz inglesa juxtapose, que en español equivale al verbo yuxtaponer, definido como poner algo junto a otra cosa o inmediata a ella. El término es un reconocimiento que P2P está yuxtapuesto al modelo cliente-servidor, el cual, como ya se ha señalado, es el modelo distribuido tradicional.

JXTA está formado por un conglomerado de protocolos que permiten a cualquier dispositivo conectado en una red (desde teléfonos celulares y PDAs, hasta PCs y servidores) comunicarse y colaborar como un punto (*peer*). Su objetivo es ofrecer servicios que posibiliten la creación de aplicaciones innovadoras en medios punto-a-punto.

Los protocolos estandarizan la manera bajo la cual los puntos:

- Se descubren unos a otros.
- Se organizan como grupos de puntos *(peers)*

- Exponen sus propios servicios.
- Se comunican entre ellos.
- Se monitorean los unos a los otros.

Los protocolos son diseñados para ser independientes de los entornos de programación, y de los protocolos de transporte, por consiguiente, pueden ser utilizados en códigos Java, C/C++, Perl, y de otros lenguajes. Asimismo, aprovechan la infraestructura de transporte de datos establecidas en protocolos como TCP/IP, HTPP, Bluetooth, HomePNA.

JXTA consta de 6 protocolos:

- **Protocolo de Descubrimiento de Puntos *(Peer Discovery Protocol)***. Usado por los nodos (*peers*) para publicar y encontrar recursos. Cada recurso es descrito y publicado usando un *advertisement* (definido en una sección posterior).
- **Protocolo de Información de Puntos *(Peer Information Protocol)***. Proporciona datos de estatus, como tráfico reciente, o estatus de disponibilidad de un punto.
- **Protocolo de Resolución de Puntos *(Peer Resolver Protocol)***. Permite el envío de solicitudes (*queries*) de cualquier índole. A diferencia de los protocolos anteriores, los cuales son empleados para requerir información predefinida, este protocolo permite a los servicios definir e intercambiar cualquier tipo de información.
- **Protocolo de Enlace *Pipe (Pipe Binding Protocol)***. Utilizado para establecer canales de comunicación virtual entre los nodos.
- **Protocolo de Enrutamiento de Puntos Finales *(Endpoint Rounting Protocol)***. Tiene la finalidad de proporcionarle a los puntos (*peers*) los puertos de destino de otros puntos.
- **Protocolo de Encuentro *(Rendezvous Protocol)***. Mecanismo bajo el cual los pares (*peers*) se pueden suscribir para adquirir el servicio de propagación. Es usado especialmente para la comunicación entre nodos que no se encuentran físicamente en una misma red.

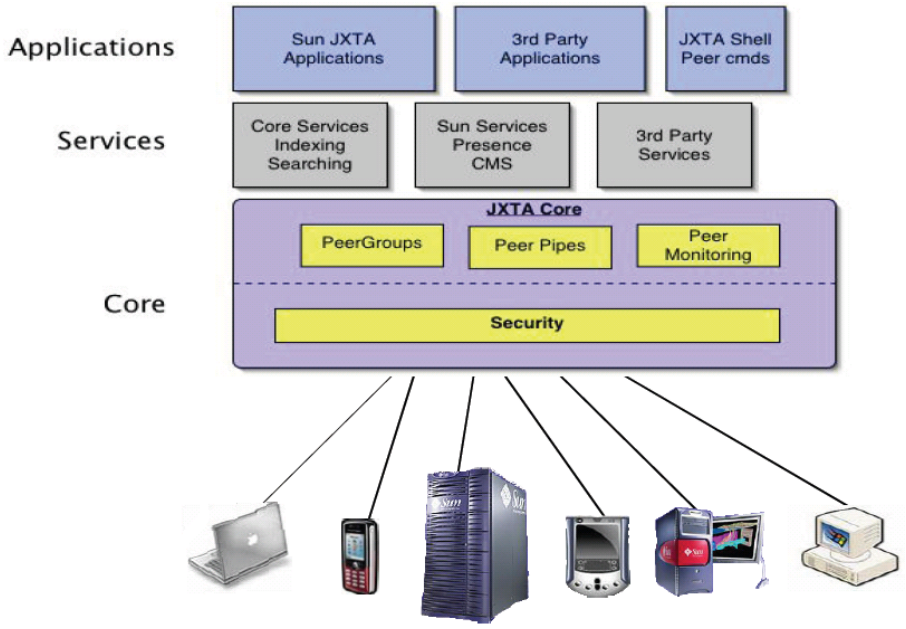

Figura 2.7: Arquitectura de la plataforma JXTA, tomando de [19]

Arquitectura de software de JXTA

Las aplicaciones P2P implementadas con los protocolos de JXTA emplean los servicios proporcionados por éstos. Cada aplicación hace uso de los servicios conforme a sus necesidades particulares. Las aplicaciones junto con los servicios y los protocolos forman la arquitectura de software de JXTA, la cual es conformada por la interrelación de tres capa, como se ilustra en la figura 2.7.

- **Capa de plataforma (JXTA *Core*)**. La capa de plataforma, también conocida como el núcleo de JXTA, encapsula las funciones esenciales que son comunes para aplicaciones P2P. Incluye los mecanismos de descubrimiento, transporte, creación de puntos, grupos, y elementos básicos de seguridad.
- **Capa de Servicio**. Establecida por servicios que no son absolutamente necesarios para que una aplicación P2P opere, pero que son utilizados por la mayoría de ellos. Algunos de éstos son los servicios de

búsqueda e indexamiento, almacenamiento y distribución de archivos y autenticación.

- **Capa de Aplicación**. Abarca las aplicaciones integradas por la utilización de los servicios de las capas precedentes. Los sistemas de mensajería instantánea, de distribución y recuperación de archivos multimedia, son casos típicos de aplicaciones ubicadas en esta capa.

Cabe señalar, sin embargo, que los límites entre la segunda y la tercera capa no son rígidos. Una aplicación para un cliente, puede ser vista como un servicio para otro nodo. La arquitectura completa está diseñada para ser modular, para conceder a los desarrolladores la facilidad de seleccionar la colección de servicios y aplicaciones que mejor llenen sus necesidades.

Como compendio, podemos afirmar que una aplicación JXTA consiste de una serie puntos *(peers)*. Estos puntos se pueden organizar en grupos facultados a ofrecer un servio determinado, como localizar y recuperar documentos digitales.

En JXTA todos los servicios y recursos que se ofrecen son hechos públicos por medio de documentos XML llamados *advertisements*. A través de ellos, otros puntos en la red aprenden como utilizar el servicio. JXTA utiliza un mecanismo unidireccional y asíncrono para trasmitir mensajes entres los puntos llamados *pipes*. Los mensajes pueden ser documentos XML, Bytes u objetos de datos.

Estos conceptos sumados al resto de componentes JXTA son descritos a mayor detalle en la sección que prosigue.

Conceptos de JXTA

Punto *(Peer)*

Es un dispositivo de red que consume uno o más de los servicios ofrecidos en los protocolos de JXTA. Cada punto opera independientemente y asincrónicamente de los otros puntos, y es reconocido por un identificador único (*Peer* ID).

Cada punto publica una o varias interfaces de red parar comunicarse con sus semejantes. Estas interfaces se vuelven disponibles en tiempo de

ejecución, los *sockets* de JXTA, abordados más adelante, son un prototipo de una interfaz de red.

Los puntos generalmente son programados para que espontáneamente se descubran unos a otros con el objetivo de formar relaciones transitorias o permanentes llamadas grupos de puntos *(peer groups)*.

Grupos de puntos *(Peer groups)*

Es una colección de puntos que han acordado colaborar entre sí. Un punto puede pertenecer simultáneamente a varios grupos. Por defecto, el primer grupo del cual se crea una instancia es el *Net Peer Group*. Todos los puntos, sin excepción, forma parte de este grupo.

Los grupos, al igual que los puntos, se registran por medio de un identificador único. Los protocolos de JXTA describen como los puntos pueden publicar, descubrir, unirse, y monitorear grupos de puntos; ellos no dictan cuando y por qué los grupos son creados.

Existen varias motivaciones para crear grupos de puntos:

- **Generar un ambiente seguro**. Los grupos forman un dominio de trabajo con determinada política de ingreso, que puede ser abierta (cualquiera se puede unir), o segura, donde se necesita el ingreso de un nombre y una clave para poder ingresar al grupo. Los grupos forman regiones lógicas cuyas fronteras limitan la obtención de recursos disponibles en los puntos.
- **Instaurar un ámbito de trabajo**. Con los grupos se logra establecer un nivel de especialización. Por ejemplo, algunos puntos pueden formar un grupo para implementar una red para compartir archivos. Los grupos subdividen el ámbito de una red en regiones abstractas con su propio y reducido ámbito de participación. Los límites de un grupo definen el ámbito de búsqueda cuando se solicitan recursos en los puntos.
- **Crear un ambiente de monitoreo**. En los grupos los puntos pueden monitorearse entre sí, con el fin de obtener información de sus semejantes (conocer si se encuentran en línea, o reconocer el nivel de demanda de un punto).

- **Disponibilidad de servicios**. Un servicio es accesible únicamente en el punto que lo implementa. Si el punto deja de funcionar, obviamente, el servicio deja de existir. Por otra parte, un servicio grupal es ofrecido como una colección de instancias de ese servicio disponible en puntos que cooperan entre sí. Si uno de ellos sufre una falla, el colectivo de instancias no es afectado.

Un grupo provee una serie de funciones llamadas servicios de grupos. Para que dos puntos puedan interactuar a través de un servicio, deben de ser parte del mismo grupo. En los protocolos de JXTA se definen las operaciones básicas para que grupos puedan ser creado y habilitados para operar. Ellas son:

- **Servicio de Descubrimiento *(Discovery Service)***: Es utilizado por los miembros de un grupo para buscar recursos como, pares (*peers*), grupos, *pipes* y servicios.
- **Servicio de Membresía *(Membership Service)***: Contiene las funciones para establecer políticas de ingreso al grupo. Un punto que desee se parte de un grupo primero debe de localizar un miembro actual y luego solicitar su incorporación. La petición, bien puede ser aceptada o rechazada en dependencia de las políticas implantadas.
- **Servio de *Pipe***: Es usado para formar conexiones *pipe* entre miembros de un grupo, de tal modo que se puedan comunicar entre sí.
- **Servicio de Resolución *(Resolver Service)***: Es destinado para enviar mensajes genéricos entre los puntos, los cuales tienen la finalidad de encontrar información necesaria, como el estatus de un servicio o el estado de una conexión *pipe*.

No todos estos servicios deben ser empleados por un grupo. El desarrollador es libre de aprovechar los que considere necesarios para su aplicación.

Módulos

Los módulos son una abstracción ocupada para representar alguna pieza de código o comportamiento útil en el entorno de JXTA, el cual permite a los puntos instanciar ese comportamiento. Los servicios son ejemplos comunes de comportamientos que pueden ser instanciados en un punto. Un módulo no especifica qué es este código: puede ser una clase de Java, un DLL *(Dynamic library)*, un conjunto de archivos XML, o un escrito *(script)*. La implementación del módulo es responsabilidad del desarrollador del mismo.

Un modulo se compone de una clase, de una especificación y de una implementación:

- **Clase**. Es usada para publicar la existencia de un comportamiento. La definición de la clase representa un compartimiento esperado. Cada clase es identificada por un identificador irrepetible (ModuleClassID).
- **Especificación**. Contiene la información necesaria para invocar un módulo.
- **Implementación**. Es en sí el comportamiento a desempeñar localizable gracias a un identificador (ModuleSpecID).

A nivel de codificación, por ejemplo, el Net Peer Group contiene todos los servicios básicos de grupo (creación, búsqueda, etc) en un módulo, entonces, para que otro grupo pueda instanciar este comportamiento tiene que ingresar este módulo como parámetro en el constructor del grupo.

Pipes

Son un mecanismo asíncrono, unidireccional y no confiable de transmisión de mensajes, utilizado para comunicar y transferir datos entre los puntos. El mecanismo es indiscriminado, soporta el traslado de cualquier tipo de objeto, incluyendo código binario, cadena de datos, y objetos basados en tecnología Java.

Cuando se establece una comunicación, los *pipes* involucrados se califican como *input pipes* (donde se reciben los datos) u *output pipes* (donde se envían los datos). Los *pipes* se ligan a un puerto TCP y a una dirección IP asociada en tiempo de ejecución.

Los *pipes* son un canal de comunicación virtual, que puede ser establecido entre puntos aún cuando no estén ligados físicamente por un canal directo. En estos casos uno o más puntos debe servir como intermediario entre los nodos que desean transferir mensajes entre si.

Los *pipes* ofrecen dos modos de comunicación: directa *(point-to-point)* y propagada *(propagate)*. También proveen una variante de la comunicación directa llamada transmisión directa segura *(secure unicast pipes)*. Cada estratigia de conexión se explica en seguida:

- **Transmisión directa *(Point-to-point)***. Conecta exactamente a dos puntos, uno de recepción *(input)* y otro de envío *(output)*.
- **Propagación *(propagate)***. Un punto de envío se conecta múltiples puntos de recepción. La propagación ocurre dentro del ámbito de un grupo.
- **Transmisión directa segura *(secure unicast pipes)***. Provee un canal de comunicación confiable entre dos puntos comunicados directamente.

Canales Bidireccionales Confiables

Dado que los *pipes* proporcionan comunicación unidireccional y no confiable, se hace necesaria la implementación de canales bidireccionales y confiables. Las aplicaciones, según sea el nivel de calidad del servicio requerido, se pueden beneficiar de los siguientes insumos suministrados por la plataforma:

- JxtaSocket, JxtaServerSocket:
  - Construidos sobre pipes y librerías confiables.
  - Sub-clases de java.net.Socket, y de java.net.ServerSocket respectivamente.
  - Proveen comunicación bidireccional y confiable.
  - Suministran configuración interna del tamaño temporal de memoria (*buffer*).

- JxtaBiDiPipe, JxtaServerPipe:
  - Construidos sobre *pipes* y librerías confiables.
  - Proveen comunicación bidireccional y confiable.
  - Exponen una interfaz basada en mensajes
  - Las aplicaciones necesitan asegurarse que el tamaño del mensaje no exceda los 64K permitido.

Tanto JxtaServerSocket y JxtaServerPipe inicializan un *pipe* de recepción *(input pipe)* para procesar solicitudes de conexión. Mientras que los JxtaSockets y los JxtaBiDiPipe se ligan a sus respectivos pares de recepción para formar líneas de comunicación dedicadas y privadas.

Advertisement

En el entorno de JXTA, todos los recursos (puntos, grupos, *pipes* y servicios) son representados por un *advertisement*. Ellos son estructuras de metadatos representados por documentos XML. Los protocolos de JXTA usan los *advertisiments* para describir y publicar la existencia de los recursos. Los puntos *(peers)* descubren los recursos por medio de la búsqueda de sus *advertisements* correspondientes.

Ejemplos de *advertisements* son:

- ***Peer* (punto) *advertisement***. Describe a un punto por medio de sus datos de identificación, como nombre, identificación (ID), y otros atributos asignados a la hora de su creación. La figura 2.8 exhibe un ejemplo de la estructura y los datos del *advertisements* de un par.
- ***Peer group* (grupo de punto) *advertisement***. Describe al grupo a través de sus identificadores, como nombre, ID, descripción, y atributos asignados a la hora de su creación.
- ***Pipe advertisiment***. Detalla un canal de comunicación, el cual es usado por el servicio de pipe para crear los puntos de envío *(input pipe)* y recepción *(out pipe)*. Cada *advertisement* contiene el ID del *pipe* y el tipo de *pipe* que representa (directo, o de propagación).

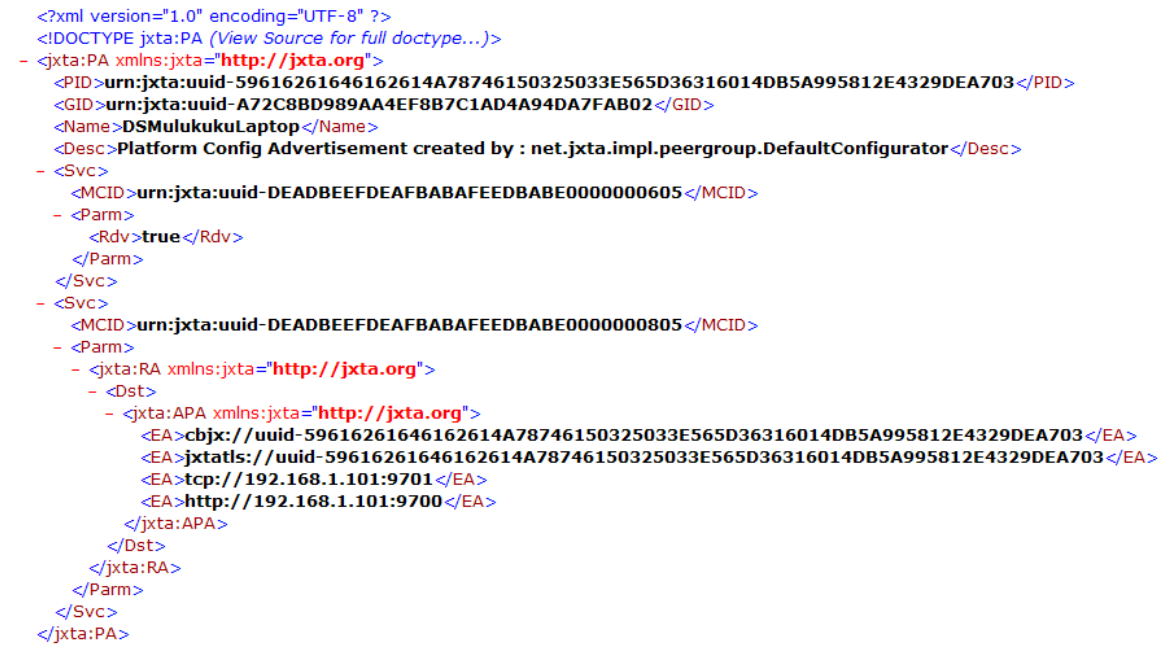

Figura 2.8: Ejemplo del *advertisement* de un punto

Identificadores (IDs)

En JXTA todos sus elementos deben ser individualmente identificados. Los IDs son un medio canónico para referir recursos.

Los URNs son empleados para expresar los IDs en JXTA. Los URN son una forma de URI que tienen por objetivo servir como identificadores independientes de recursos. Igual que otras formas de URI, los IDs en JXTA son representados en texto.

Un ejemplo del identificador de un punto (*peer*) es: urn:jxta:uuid-59616261646162614A78746150325033F3BC76FF13C2414CBC0 AB663666DA

Los IDs son generados aleatoriamente en el ambiente de programación donde son construidas. Existen dos IDs reservadas: el ID NULL, y el ID del *Net Peer Group*. Los identificadores nulos son ocupados para solicitar servicios en cualquier punto que acepte la petición. Cuándo un servicio es requerido de un punto en particular, el ID de dicho punto es necesario para ejecutar la petición.

2.5. Trabajo relacionado

En el capítulo introductorio se señaló que el presente trabajo de investigación consiste en proponer un modelo de interoperabilidad punto-a-punto para sistemas de bibliotecas digitales.

Existen innumerables ejemplos de aplicaciones que adoptan el modelo de distribución punto-a-punto, de igual manera es posible encontrar un sinnúmero de sistemas de bibliotecas digitales. Por consiguiente, en esta sección sólo expondremos trabajos que, al igual que el nuestro, se encuentran en el ámbito donde se aplica el paradigma punto-a-punto en sistemas de repositorios de documentos digitales.

2.5.1. Gnutella

Es una aplicación formada por una red de puntos (*peers*). Cada par está conectado a un grupo en particular, y ninguno de ellos es capaz de detener el funcionamiento de la red. Cada punto posee la capacidad de transferir archivos a cualquier otro nodo en la red.

Originalmente fue creado como una herramienta para buscar y transferir información digital, pero las últimas versiones proveen librerías para crear aplicaciones P2P, sin embargo, el propósito original de su creación sigue siendo la razón más popular de su uso [3].

2.5.2. Freenet

Es un sistema para el almacenamiento, indexamiento, y recuperación de documentos, usado en el entorno de un grupo de trabajo (*work-group*). Se compone de una interfaz Web (*web-based front-end*) conectada a un servidor HTTP local. Permite a los usuarios crear grupos bajo los cuales podrán compartir documentos etiquetados (XML o HTML), y ejecutar búsquedas en el contexto de un grupo [2].

2.5.3. Jibe

Es un sistema para compartir contenido almacenado en base de datos. Cada cliente es un punto (*peer*) que cuenta con un sistema de almacenamiento que puede ser accesible mediante una interfaz JDBC u ODBC.

Jibe proporciona un servlet de java que interpreta XML para cada taxonomía, y muestra los resultados de búsqueda en una forma Web, la cual puede ser utilizada por múltiples usuarios. Los usuarios pueden, simplemente escribir una cadena de datos en el espacio de búsqueda del servlet, y deja que Jibe haga el trabajo de búsqueda, presentación, y ordenamiento de resultados [4].

2.5.4. MusicBrain Metadata

Es una iniciativa formada para servir como medio de intercambio de metadatos de archivos musicales transferibles en Internet. La aplicación resultante, es un servicio Web gratuito basado en el empleo de metadatos Dublin Core [11] y servicios Web para crear información descriptiva (nombres de canciones, disco, año, etc) de los recursos musicales a ser compartidos.

Los archivos musicales se encuentran distribuidos en pares (*peers*) conectados a Internet, desde donde son transferibles, gracias a la información generada por medio del servicio Web [6].

2.5.5. P2PDLib

Es una aplicación punto-a-punto que ofrece servicios de bibliotecas digitales personales, como envío y recepción de documentos, transferencia de metadatos de colecciones y documentos, entre otros.

Cada dispositivo móvil contiene un esqueleto de la estructura de una biblioteca personal residente en un servidor de datos y basada en ella opera de forma independiente, mediante la transferencia de metadatos obtenidos del esqueleto, y el envío de documentos entre los puntos, sin intervención del servidor de datos.

El objetivo de esta aplicación es liberar al servidor de datos de asumir peticiones de servicios que pueden ser ejecutados sin su intervención,

liberándolo, por tanto, de una considerable carga de trabajo [9].

2.6. Resumen del capítulo

Los antecedentes sobre los que se sustenta el modelo de interoperabilidad entre instancias independientes del servidor de datos fue el tema central de las secciones recién expuestas.

El capítulo inició introduciendo los conceptos que definen un sistema de biblioteca digital, dado que son éstos los que constituyen las bases para comprender el funcionamiento y el objetivo de la solución propuesta en el Centro de Investigación en Informática del ITESM para el acceso universal a información digital almacenada en repositorios, la cual fue bautizada con el nombre de PDLib.

La arquitectura y los conceptos de este sistema fueron tratados en la segunda sección del capítulo, con el fin de vislumbrar los alcances pretendidos y los logros obtenidos con este diseño; a la vez que se consigue introducir el ámbito particular de la arquitectura en donde se aplicará el trabajo realizado en la presente investigación, y el conjuntos de sus objetivos.

Posteriormente se trató el paradigma de distribución punto-a-punto, sobre el cual se implementó el modelo de interoperabilidad propuesto. En esta sección se expusieron las particularidades de este paradigma y las razones que llevaron a la elaboración de esta alternativa yuxtapuesta al modelo cliente-servidor, que es el modelo de facto para construir sistemas distribuidos.

Una vez presentada la arquitectura punto-a-punto, se abordaron las tecnologías de desarrollo para la implementación de este modelo a un problema particular. También se dieron a conocer las razones que llevaron a tomar la decisión de escoger JXTA como la plataforma de trabajo en que se apoyó el modelo de interoperabilidad.

Por último, se ofreció información sobre aplicaciones que encajan en el marco de trabajo de esta investigación, dado que es relevante conocer las similitudes y diferencias de las aplicaciones que están disponibles en el mercado.

En el siguiente capítulo se expondrá el modelo de interoperabilidad entre instancias del servidor de datos, que es el núcleo del presente trabajo.

Capítulo 3

Modelo Punto-a-Punto para la Interoperabilidad entre Instancias Independientes del Servidor de Datos de PDLib

3.1. Introducción

Como se explicó en el capítulo anterior, el servidor de datos es el corazón de PDLib, contiene todas las funciones que hacen posible la operación del sistema, y es por medio de él que los distintos tipos de clientes tienen la facultad de ofrecer a los usuarios todos los servicios disponibles en PDLib.

La interacción entre los diferentes tipos de clientes y el servidor de datos, más allá de las particularidades de acceso vistas en la sección 2.2.1 (Arquitectura del sistema PDLib), está basada en la arquitectura cliente-servidor. Aún cuando los clientes móviles o los clientes Web, necesitan de un intermediario para acceder a los servicios ofrecidos por el servidor de datos, este factor no afecta el hecho de que existe una entidad proveedora de servicios y otras que se limitan a solicitarlos y a presentar los resultados amablemente. Esta realidad implica que PDLib engloba las limitantes de la arquitectura cliente-servidor expuestas en la sección 2.3.1 (¿Por qué surge el modelo punto-a-punto).

Por tanto, se exploró dentro del modelo de computación punto-a-punto con el objetivo de remediar estas dificultades y hacer de PDLib un sistema con almacenamiento escalable, es decir, que pueda crecer más allá de las capacidades de almacenamiento de una sola computadora; a la vez que sea un sistema seguro, sin tener que recurrir al establecimiento un servidor de respaldo con determinadas políticas de está índole, y por último, convertirlo en una red virtual global para compartir información y transferir colecciones de archivos con cualquier tipo de formato.

La presentación de la implementación de un modelo punto-a-punto aplicado en el servidor de datos a través del uso de la plataforma JXTA (sección 2.4.2, página 25) es el objetivo de este capítulo, el cual se compone de dos secciones: en la primera de ellas se expondrá a detalle la arquitectura del servidor de datos y la forma en como se adhiere el modelo P2P a él; y la segunda parte estará dedicada completamente a detallar cada uno de los componentes del modelo y la interrelación entre ellos.

3.2. Servidor de Datos de PDLib

En la sección 2.2.3 explicamos como el servidor de datos se relacionaba con los otros componentes del sistema, a la vez que expusimos los servicios que éste ofrece. En esta sección presentaremos toda su funcionalidad y abordaremos los detalles de su implementación, para comprender la forma en que opera y como el modelo P2P introducido encaja en su arquitectura.

Las responsabilidades del servidor de datos introducidas se dividen en los siguientes apartados:

- **Biblioteca digital personal**. Realiza operaciones CRUD, acrónimo del inglés que engloba las operaciones de creación, navegación, búsqueda, actualización y borrado de los objetos almacenados en la biblioteca digital personal (colecciones, documentos y metadatos). También permite copiar documentos y colecciones, envíar documentos por correo electrónico, y la conversión de formatos de archivos.
- **Almacenamiento**. Provee la infraestructura necesaria para el acopio de los documentos de los usuarios. Estos documentos se encuentran físicamente almacenados en la base de datos de PDLib.

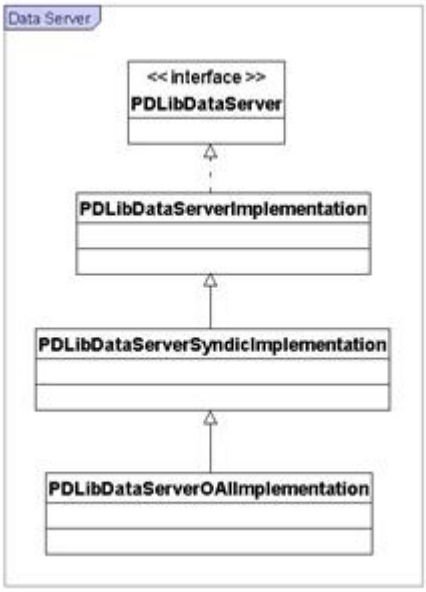

Figura 3.1: Jerarquía de herencias del servidor de datos

- **Interoperabilidad y Comunicación**. A través del protocolo OAI-PMH, PDLib muestra los metadatos de los documentos almacenados en otros servidores OAI-PMH.

  Por medio del protocolo XML-RPC expone los métodos operativos del servidor de datos para que puedan ser utilizados por la gama de clientes soportados. De igual manera permite la interacción con otros servidores de datos PDLib a través del modelo P2P expuesto en el apartado 3.3 de este capítulo.

3.3. Estrategia de implementación del servidor de datos de PDLib

El servidor de datos implementa una jerarquía de clases que inicia con la interfaz que contiene la declaración de todos los métodos que harán posible las operaciones de **Almacenamiento** y de **Biblioteca digital personal**. Posteriormente le sobreviene la clase que implementará los

métodos declarados en dicha interfaz, a la cual le prosiguen las clases que sobrescribirán o utilizarán éstos métodos heredados, con el fin de ocuparlos según las normas del protocolo de distribución a manejar. Lo que significa, que se cuenta con un diseño que habilita la comunicación con el MCM a través del protocolo XML-RPC, y otra, que interactúa con servidores exógenos por medio del protocolo OAI, tal y como se muestra en la figura 3.1.

El orden de la jerarquía de herencia a partir de la primera implementación de los métodos de la interfaz, se define según la conveniencia de las clases inferiores, es decir, que la implementación de OAI hereda de la implementación de XML-RPC porque podría precisar de algunos métodos de esta clase.

Esta arquitectura fue concebida con la finalidad de obtener independencia entre los mecanismos de comunicación existentes. De tal manera, que se podría desarrollar la comunicación a través de RMI o CORBA, sin tocar del todo las clases ya elaboradas. Por otra parte, las nuevas implementaciones no afectan el funcionamiento de las clases previamente elaboradas, lo único que logran es ampliar los servicios ofrecidos por el servidor de datos. Por ejemplo, si se eliminara la implementación de OAI, el servidor de datos seguiría siendo funcional, simplemente dejaría de proporcionar los métodos de la clase excluida.

Por tal razón, la introducción del soporte para la interoperabilidad entre servidores de datos se define como una clase independiente que se incluye en la jerarquía de herencia del servidor datos, gráficamente ilustrado en la figura 3.2

El servidor de datos, también fue concebido para ser independiente de cualquier manejador relacional de base de datos (RDBMS), de tal forma que PDLib pueda operar bajo Oracle, SQL Server, MySQL, o cualquier otro RDMBS.

Esta capacidad se logra gracias al uso de un API (*Aplication program interface*) de base datos, bautizado con el nombre de DBAPI [15], el cual ofrece comandos estándares y propios de consultas a base datos, los que posteriormente, son traducidos al manejador empleado. La figura 3.3 describe esta situación.

En el capítulo 2 describimos el funcionamiento de PDLib, y en esta sección profundizamos en el diseño del servidor de datos, lo que nos sirve de base

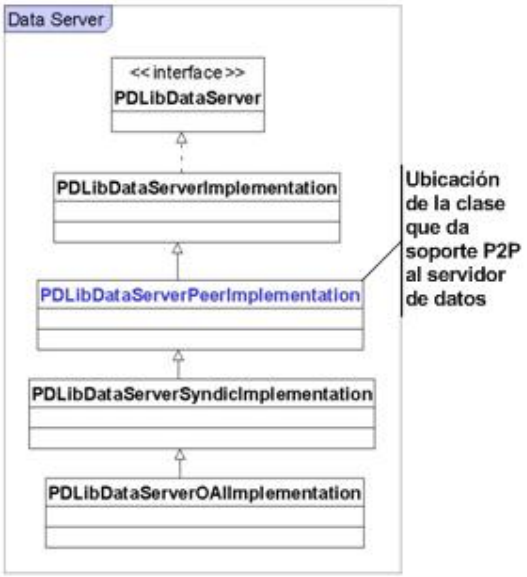

Figura 3.2: Introducción de la clase de implementación del soporte P2P a la Jerarquía de herencias del servidor de datos

para comprender como una arquitectura P2P es introducida en el sistema, objetivo del apartado que prosigue.

3.4. Modelo de interoperabilidad entre servidores de datos de PDLib

En la arquitectura de PDLib (figura 2.2) se visualiza, dentro de la capa de interoperabilidad, la interacción entre distintas instancias del servidor de datos, lo que se denomina constelación de servidores de datos. Es importante definir que cada instancia es completamente independiente entre sí; cada una puede estar instalada en una computadora diferente, y sin el soporte de interoperabilidad, no es posible la interrelación entre los diferentes servidores instalados.

Esta constelación fue pensada para operar como una red virtual P2P, cuyos nodos componentes no deben ser obligados a formar parte de un mismo enlace físico, por lo tanto, cualquier servidor de datos con conexión a

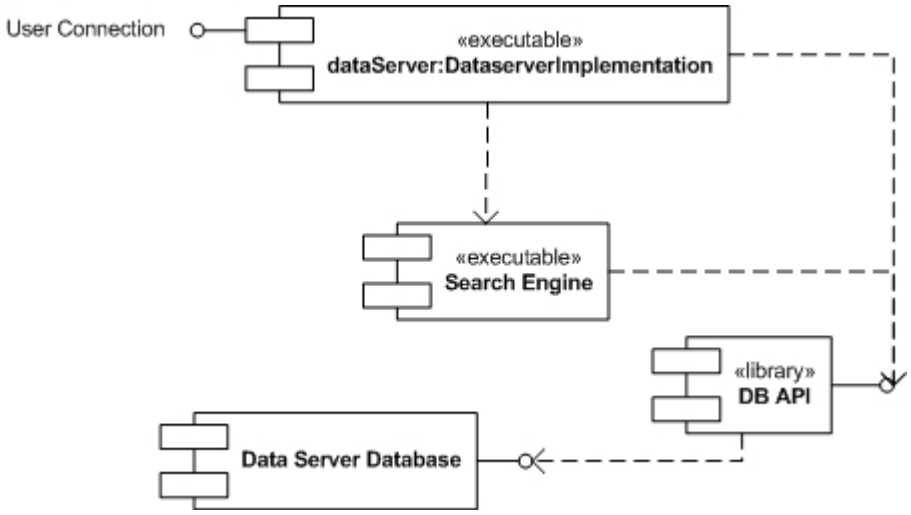

Figura 3.3: Componentes del servidor de datos

Internet debe ser capaz de formar parte de la constelación. Para cumplir con estos requerimientos, el diseño del modelo de interoperabilidad se basó en los protocolos P2P de JXTA, tal y como se justifica en la sección 2.4. El diseño propuesto e implementado se presenta en la figura 3.4

3.5. Componentes del modelo de interoperabilidad

Para poder formar la red virtual descrita, cada servidor de datos se convierte en un punto (*peer*) de la red global de puntos de JXTA, la cual se denomina como grupo global de JXTA (JXTA World Group). Por defecto, todo punto generado con los protocolos de JXTA pertenece a este grupo, lo que permite la comunicación global entre los nodos conectados a Internet en el mismo momento. La interrelación entre los puntos en el grupo global es limitada; fundamentalmente se aplican los protocolos JXTA de búsqueda y adhesión a grupo para localizar nodos o grupos de interés, que una vez encontrados, son empleados para un fin en particular; o en el caso de los grupos, para formar parte de ellos. Un nodo debe tener implementado alguna funcionalidad, más allá de los servicios estándares proporcionados por JXTA,

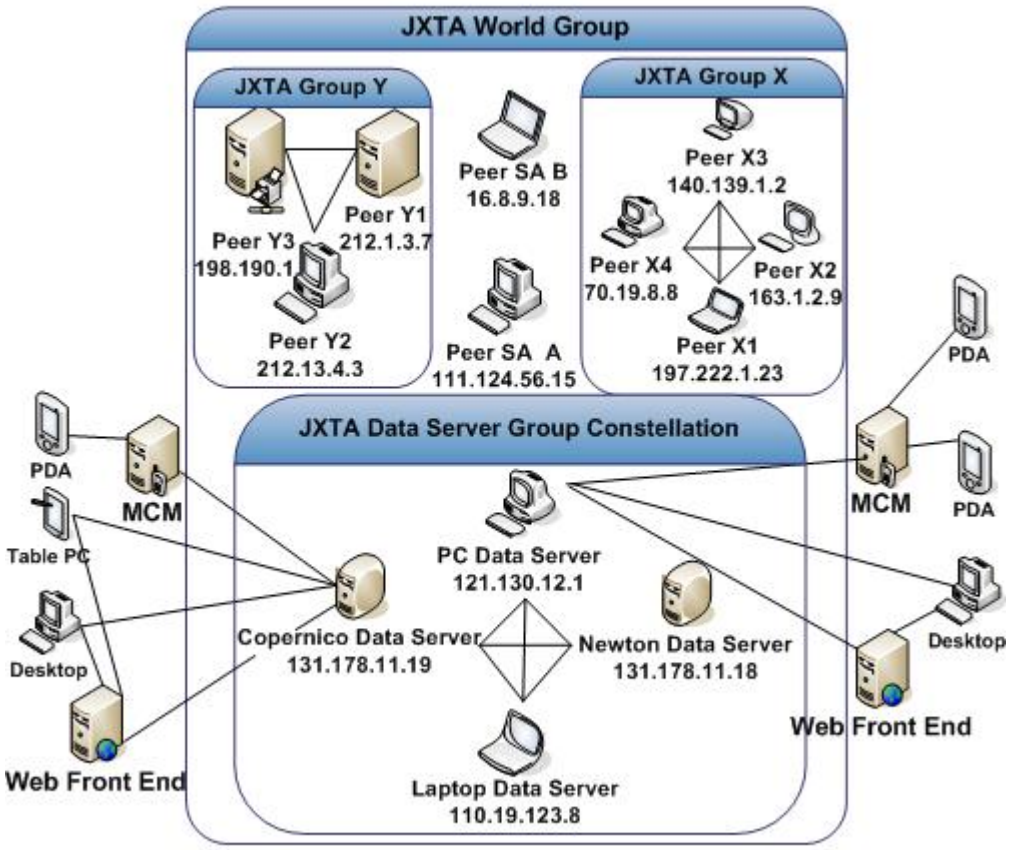

Figura 3.4: Modelo P2P implementado

de lo contrario, su valor será limitado para alguna aplicación en particular. Por ejemplo, si se desarrolla un sistema para mensajería instantánea, cada par que desee formar parte de él, debe contener las operaciones necesarias que posibiliten el intercambio de mensajes con los distintos nodos que conforman la aplicación.

En la sección 2.4.2 (JXTA página 25) enumeramos las motivaciones para formar grupos de puntos, por tales razones la constelación de servidores de datos se genera en el ámbito de un grupo, el cual puede ser percibido como una subred de la red global de JXTA. La figura 3.4 ilustra de forma adecuada esta percepción.

La comunicación interna entre los miembros de la constelación de servidores de datos se realiza ocupando los servicios comunes que proporciona JXTA a través del protocolo de búsqueda, especialmente para el descubrimiento de pares miembros. Y hace uso de Sockets de JXTA para el establecimiento de los servicios propios de la constelación.

Se podría afirmar, entonces, que los componentes principales del modelo de interoperabilidad son: cada servidor de datos convertido en punto, el grupo de puntos que conforman la constelación, y por último, los servicios prestados en cada punto del grupo.

Cada uno de estos elementos es descrito a continuación:

3.5.1. Punto (*Peer*)

Se define como punto a toda instancia del servidor de datos capaz de operar con plena independencia de cualquier otra instancia instalada. Sin embargo, debe de contar con la facultad de intercambiar servicios con otros nodo con el objetivo de ampliar el conjunto de resultados de las funciones previamente establecidas, y de incrementar la oferta de prestaciones a los clientes.

Todo punto contiene exactamente el mismo código de implementación, son instancias idénticas, pero reconocibles por un nombre y un identificador (ID) único e irrepetible. Los IDs son establecidos por la plataforma JXTA, en cambio los nombres son cadenas (*Strings*) de caracteres tradicionales introducidos por el administrador del servidor de datos.

3.5.2. Grupo

Para que las instancias de los servidores intercambien servicios propios de PDLib, deben de pertenecer a un grupo denominado PDLib, en caso contrario, sólo podrán comunicarse a través de los servicios estándares de JXTA, como lo harían con cualquier otro punto del grupo global.

Al igual, que los puntos, todo grupo debe ser reconocido en el mundo de JXTA por medio de un ID único e irrepetible, y generado por esta misma plataforma.

A nivel de implementación del modelo, todos los puntos servidores de datos nacen como miembros del grupo PDLib, lo que les facilita la búsqueda de otros pares pertenecientes a este grupo. En el diagrama de actividad de la figura 3.5 se explica la operación completa del modelo de interoperabilidad, en él se podrá apreciar el funcionamiento que posee cada nodo servidor.

Cada servidor de datos lo primero que hace al comenzar a operar, es iniciar la plataforma de JXTA; si es la primera vez que está siendo ejecutado, entonces se adhiere al grupo PDLib, en caso contrario, esta operación no será necesaria, dado que el proceso de ingreso se lleva a cabo solamente en una sola ocasión, es decir, que una vez que acontece, el nodo mantendrá por siempre la membresía del grupo.

En el siguiente capítulo se expondrá detalladamente, el funcionamiento real de este modelo en un ambiente operacional.

3.5.3. Comunicación

Tal y como se mencionó recientemente, los puntos se valen de dos mecanismos de comunicación. Las tareas particulares que se ejecutan a través de estos medios se dividen de la siguiente manera:

- **A través de servicios comunes de JXTA**.

    1. **Búsqueda remota de puntos pertenecientes al grupo PDLib.** Esta tarea es empleada para ubicar otros servidores de datos que se encuentren operando en un momento determinado, y es ejecutada gracias al servicio de descubrimiento de JXTA (*DiscoveryService*) ofrecido por el protocolo de descubrimientos

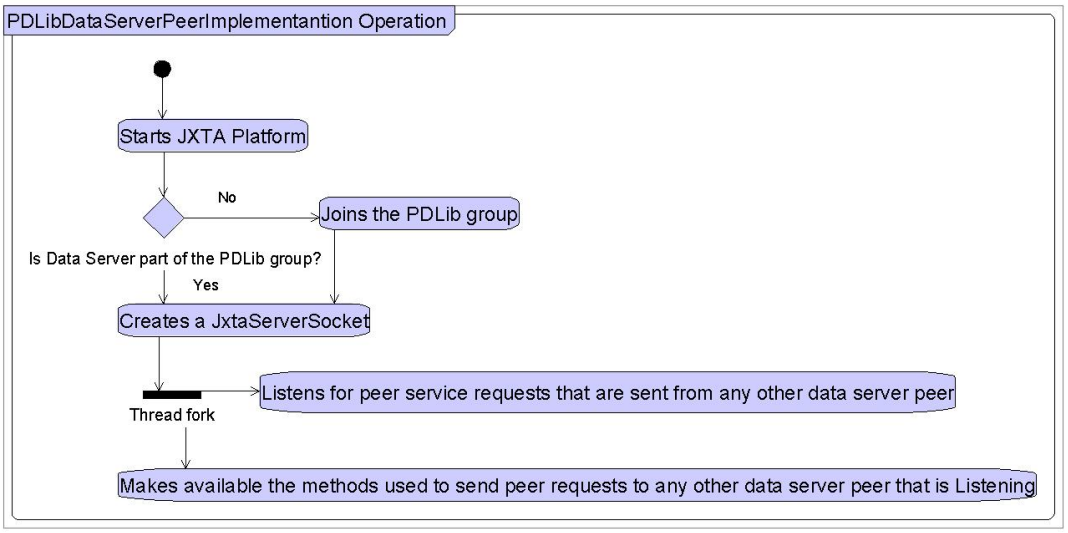

Figura 3.5: Operación de la clase que implementa el soporte P2P en el servidor de datos

de puntos (*Peer Discovery Protocol*). Básicamente, la localización remota se lleva a cabo propagando (*multicast*) peticiones de búsqueda que brinda el servicio.

2. **Búsqueda local de puntos pertenecientes al grupo PDLib.** Como se relató en la sección 2.4.2 todos los recursos de JXTA son representados por *advertisements*. Una vez que un punto es ubicado, su *advertisement* es guardado temporalmente en la memoria local cache del par que ejecutó la búsqueda. El tiempo de almacenamiento local establecido por defecto en JXTA es de dos horas. Es muy importante puntualizar que el concepto de memoria cache local de un punto, no es la memoria RAM de la computadora donde reside, más bien es el archivo *advertisement* del nodo donde se introducen los *advertisements* de los nodos encontrados. Una vez concluido el tiempo de almacenamiento temporal, las secciones donde se escribió la descripción (*advertisements*) de los puntos encontrados son borradas. La eliminación de esta información es ejecutada transparentemente por JXTA. También es importante señalar que JXTA efectúa esta exclusión porque los recursos remotos no siempre se encuentran disponibles, por tanto, es lógico

que antes de trabajar con ellos, primero hay que cerciorarse de la disponibilidad de éstos.

Para que un servidor de datos pueda establecer una línea directa de comunicación con otra instancia, es imperioso que conozca el identificador (ID) de éste, el cual se obtiene buscando localmente dentro de los *advertisements* guardados en el cache.

- **Por medio de Sockets propios de JXTA**.

  Tal y como se indicó en la sección 2.4.2, JXTA utiliza *pipes* para generar vías de comunicación directa entre los puntos. Entre las opciones de *pipes* ofrecidas, escogimos los JxtaSockets y los JxtaServerSockets debido a que son canales de comunicación bidireccional confiables, en donde se pueden enviar y recibir objetos Java en espacios de memoria (*buffers*) de tamaño configurable. Y dado que los servidores de datos intercambiaran información y objetos propios de una biblioteca digital (colecciones, documentos y objetos de búsqueda), este tipo de *pipes* resulta ser el mecanismo adecuado para las necesidades de PDLib.

  Un objeto JxtaSocket es destinado para enviar peticiones de servicios que deberán ser atendidas remotamente por un objeto JxtaServerSocket, que tiene por misión escuchar permanentemente por requerimientos de esta naturaleza, tal y como sucede con los sockets de la clase Java.net. La diferencia, sin embargo, radica en que, tanto los objetos JxtaSockets como los JxtaServerSockets establecen el medio de comunicación teniendo como parámetros, el ID de grupo, y el ID de punto, y no direcciones IP y puerto, como lo hacen los de la clase Java.net. Por lo demás, la transferencia de objetos y tipos de datos Java ocurre de la misma manera en ambos tipos de sockets, es decir, ocupando objetos OutputStream e InputStream, para enviar y recibir, respectivamente.

  Resumiendo, se pueden enumerar los pasos para crear una vía de comunicación recurriendo a objetos JxtaServerSockets y JxtaSockets de la siguiente manera.

  - **JxtaServerSocket**:

    Construir el objeto registrando un grupo disponible en el entorno de JXTA. Bloquearse hasta que un cliente JxtaSocket se conecte.

Recibir y enviar datos en objetos OutputStream e InputStream. Y finalmente cerrar la conexión.

- **JxtaSocket**:
  Construir el objeto proporcionando un grupo disponible y un ID del punto donde reside el JxtaServerSocket. Enviar y recibir datos en objetos OutputStream e InputStream. Cerrar conexión.

El siguiente apartado abordará los servicios P2P propios de PDLib, los cuales son establecidos gracias al uso de los sockets recién descritos; y se podrán apreciar gráficas que muestran la lógica de comunicación cuando uno de estos servicios se efectúa.

3.5.4. Servicios

Los servicios P2P particulares de PDLib son las operaciones que hacen posible la erradicación de las desventajas inherentes de los sistemas con arquitectura cliente-servidor. Para el alcance de esta tesis, se diseñaron e implementaron los siguientes servicios: búsqueda remota de objetos característicos de la biblioteca personal (documentos y metadatos), y la copia remota de colecciones de documentos.

Estas acciones se desarrollaron sirviéndose de los métodos originalmente definidos en el servidor de datos, lo que significa que los servicios P2P incorporados, no son más que la operación remota y no local, de los métodos contenidos en un servidor de datos. Por consiguiente, todas, absolutamente todas las funciones locales de un servidor de datos pueden ser expandidas a un versión remota, aprovechando la infraestructura de comunicación P2P construida para este modelo.

Los objetivos y la lógica del funcionamiento de los servicios se especifican en las secciones que prosiguen:

Búsqueda remota (Peer search)

Con esta funcionalidad se persigue obtener recursos de otros sistemas PDLib localizables, de tal modo, que el conjunto de resultados presentados a los usuarios sea mayor. Sin este servicio, los clientes estarán limitados

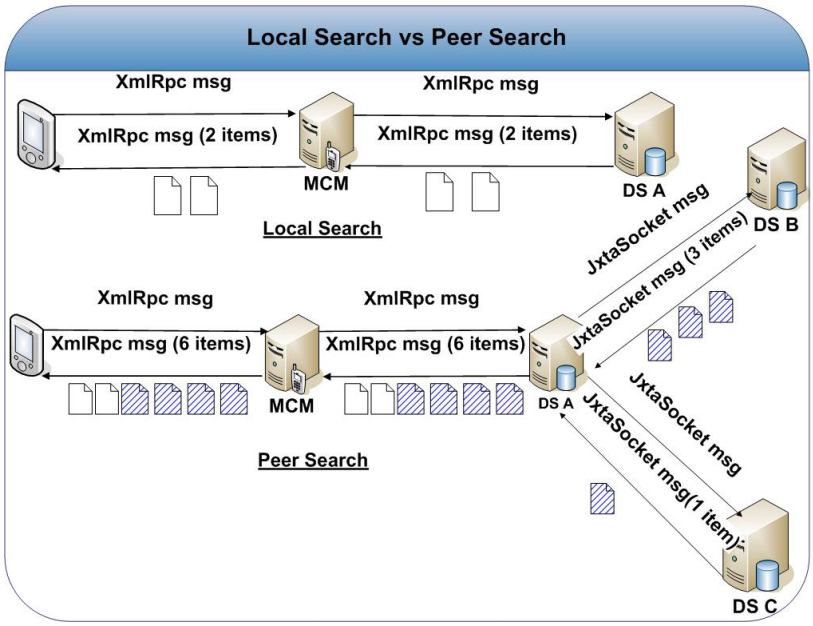

Figura 3.6: Contraposición del funcionamiento de los dos tipos de búsqueda

a hacer solicitudes de objetos ubicados en el servidor de datos que tiene la responsabilidad de atenderlos. Imaginemos, entonces, el caso de un usuario que necesita encontrar artículos digitales de Gabriel García Márquez, supongamos además, que la constelación de servidores de datos se compone de 3 instancias que contiene en total 6 artículos del mismo autor, sin embargo, el nodo encargado de atender al usuario, sólo registra 2 artículos del escritor pretendido. Con la función de búsqueda original (local) que provee todo servidor de datos, el usuario solamente obtendrá 2 artículos de su interés, pero gracias a la búsqueda remota, adquirirá información de 4 documentos más. La figura 3.6 refleja estos dos escenarios.

La búsqueda remota, al igual que el resto de servicios remotos, se codificó utilizando la infraestructura de comunicación P2P entre servidores de datos. Básicamente, se consigue por el establecimiento de un dialogo sencillo entre transmisor y receptor: el transmisor le indica al receptor, a través de un mensaje único, cual son los parámetros de búsqueda que solicita, mientras que el receptor recibe la petición, la procesa, y responde con el objeto que contiene los resultados de la indagación. Si el transmisor no recibe respuesta alguna en un lapso de tiempo definible por el usuario, se interpretará que el

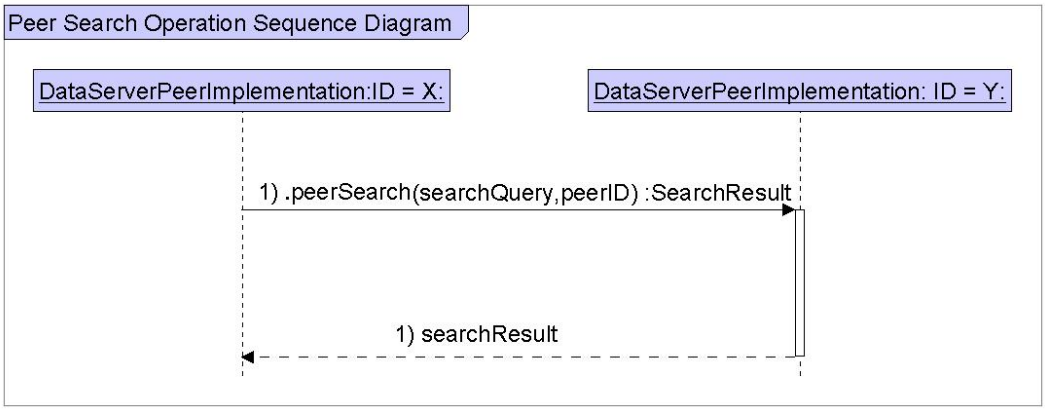

Figura 3.7: Diagrama de secuencia de la operación de la Búsqueda Remota

receptor está caído. Debemos, sin embargo, definir como respuesta al objeto que contiene el resultado de la aplicación de la búsqueda, el cual pudiera llegar vacío, dada la eventualidad de que no se encuentre, en el nodo destinatario, ninguna correspondencia con los parámetros transmitidos. El diagrama de secuencia 3.7 resalta los pasos de este procedimiento, teniendo como punto emisor el servidor de datos con ID X, y sirviendo como correlativo el punto cuyo identificador está representado por la letra Y. Cabe resaltar, como se ha venido insistiendo, que un servidor de datos tiene las facultades para asumir los dos roles involucrados en la comunicación.

Resulta pertinente hacer ver que este servicio involucra solamente a dos instancias, por lo tanto, si se deseara efectuar una búsqueda en todos los miembros de la constelación, el servidor de datos solicitante deberá de repetir la búsqueda remota tantas veces como entidades existan en el grupo, cambiando el identificador (ID) del servidor receptor en cada llamada del método. Visualmente, la figura 3.8 exterioriza la secuencia de invocaciones a la función de búsqueda remota de parte de un servidor de datos (ID X) fuente a los 2 restantes puntos, que hipotéticamente forman la constelación en este ejemplo.

JXTA proporciona sockets UDP (MulticastSockets) con los cuales se propagan (*multicast*) mensajes en el contexto de un grupo. Un servicio como la búsqueda remota podría implementarse aprovechando este tipo de transmisión, a tal grado, que cuando se necesite buscar en todos los miembros de la constelación, la fuente mandaría un mensaje *multicast* del cual esperaría

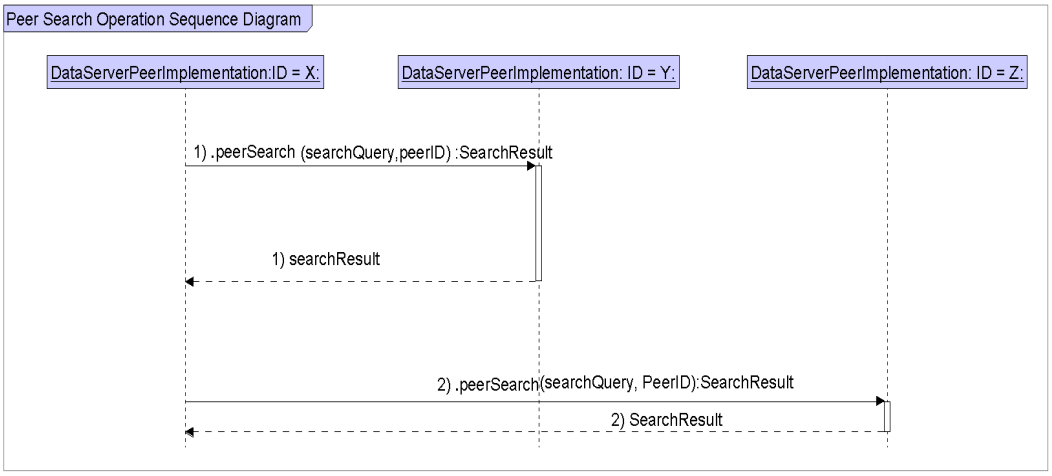

Figura 3.8: Diagrama de secuencia de la operación de la Búsqueda Remota en dos nodos

recibir respuestas de todos los destinatarios, de igual forma como se ilustra en la figura 3.8.

Pero esta solución no fue implementada en el modelo por las desventajas que abarca:

- **Congestión de tráfico en el nodo receptor**. Supongamos que existan varias decenas de servidores de datos en la constelación, cada uno de ellos enviaría mensajes con objetos pesados que ahogaría a un sólo destinatario.
- **Confiabilidad limitada**. Los mensajes UDP no garantizan entrega, trabajan bajo el esquema de mejor esfuerzo para hacer llegar los paquetes, lo que implica que el usuario no tiene garantías de obtener toda la gama de objetos que realmente concuerdan con los parámetros de búsqueda.
- **Escaso nivel de control**. Los mensajes *multicast* se envían a todos los miembros de un grupo, por tal motivo,el usuario no puede decidir a que miembros de la constelación dirigir su solicitud. En cambio, los JxtaSockets, como se ha venido mencionado, son canales confiables, en donde se transmiten mensajes *unicasts*, lo que faculta a los usuarios a seleccionar las instancias en donde se desea buscar objetos.

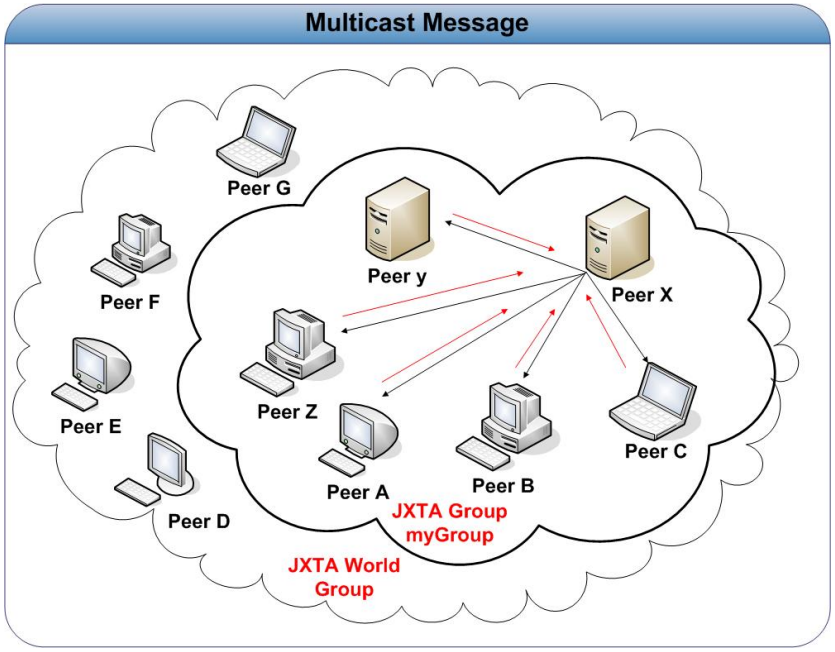

Figura 3.9: Comunicación a través del uso de JXTA MulticastSockets

Copia remota de colecciones (*Peer copy collection*)

Copiar colecciones implica trasladar la estructura (nombre, identificador de colección, idetificador de librería a la que pertenece, permisos e identificadores del tipo de metadatos bajo el cual se rige) de una colección junto con los documentos que contiene. La motivación para copiar una colección reside en mantener las replicas que el usuario estime conveniente, de tal manera que pueda mantener disponible su información en caso de que el servidor de datos asignado salga de línea. Para que un usuario pueda disponer de su información en varios servidores de datos, es necesario que en cada uno de ellos posea una cuenta de acceso.

Existen otras 2 consideraciones a tomar en cuenta a la hora de copiar una colección:

1. Los identificadores (ID) de librería, tanto de la entidad transmisora como del receptor deben ser iguales. De no cumplirse esta condición, no será posible copiar colecciones entre los puntos (*peers*) involucrados en esta acción. Por defecto, el ID de la primera librería que forma

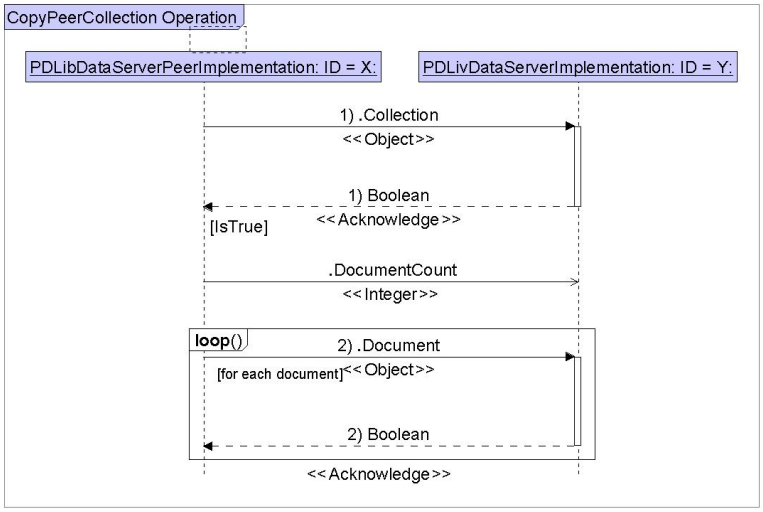

Figura 3.10: Diagrama de secuencia de la operación que copia una colección entre dos punto(*peers*)

un servidor de datos es siempre el mismo, y dada esta circunstancia, el movimiento de colecciones entre estas librerías no adolece de esta restricción. La existencia de este cortapisa deviene del diseño e implementación del modelo de datos de PDLib, el cual estipula que las colecciones forman parte de las librerías. Por lo tanto, no es lógico pensar que la entidad destinataria creará una colección sin una librería que la contenga.

2. Si la colección existe en la librería destino, entonces sólo se trasladarán los documentos contenidos en la colección emisora. Puede suceder que copias de esos documentos ya existan en el receptor, pero esto no evitará que la transferencia se lleve a cabo, porque el modelo de datos permite tener copias de documentos en una colección.

A diferencia de la búsqueda remota, en donde el trámite de comunicación consta del envío de dos mensajes, uno de ida y otro de regreso, en la copia remota de colecciones el diálogo entre los involucrados es mucho más elaborado. En él se recurre a acuses de recibos para sincronizar el envío y recepción de objetos. El diagrama de secuencia de la figura 3.10 ilustra el diálogo sostenido entre las partes involucradas.

3.6. Diferencias del modelo de interoperabilidad con otros ejemplos de bibliotecas digitales

Existen múltiples proyectos desarrollados de bibliotecas digitales, ejemplos de éstos son el sistema Phronesis [12], los ya mencionados repositorios OAI, o los repositorios DSpace [1].

Todos estos sistemas comparten una característica en común, y es que pueden ser considerados sistemas punto-a-punto, dado que los repositorios de estos ejemplos tienen la capacidad de ofrecer servicios a otras entidades similares, y a la vez cuentan con la facultad de solicitar estos servicios a estas mismas entidades; es decir, tienen la habilidad para asumir funciones, tanto de servidores, como de clientes.

En los casos particulares de los repositorios OAI y DSpace, éstos desarrollaron protocolos de comunicación propios para la interoperabilidad entre sus instancias, lo que significa que es necesario regirse por las normas de éstos para que la interacción pueda ocurrir. Estos protocolos no tienen la capacidad de formar comunidades en donde las instancias espontáneamente se puedan localizar, por consiguiente, la interoperabilidad ocurre por la intervención de un administrador que introduce las direcciones de la ubicación de los repositorios disponibles en un ámbito de red determinado.

El sistema Phronesis, por otro lado, si bien no desarrolla protocolos para la interacción entre sus nodos, depende, al igual, que los dos sistemas anteriores, de un usuario administrador que conozca y e introduzca las direcciones donde se ubican otros repositorios Phronesis con los que se desee entablar comunicación.

Por tanto, la principal diferencia con estos sistemas, es la espontaneidad que tienen las instancias para conocer a los miembros pertenecientes a la comunidad de servidores de datos instalados en un entorno de red. También es importante hacer notar que PDLib no desarrolla protocolos ni estándares para la comunicación entre sus nodos, éste se basa en la plataforma JXTA, lo cual permite centrar el desarrollo del sistema en los servicios que se quieren proporcionar y no en la infraestructura de comunicación, como lo hacen los sistemas OAI y DSpace.

3.7. Resumen del capítulo

La aplicación práctica de una arquitectura P2P en el sistema PDLib es el tema central de esta investigación. El modelo propuesto e implementado en el sistema fue el contenido de este capítulo. Se describieron meticulosamente cada uno de los elementos que conforman el modelo, de igual forma se justificaron las decisiones de diseño para la codificación de estos componentes, y como la estructura planificada del servidor de datos facilitó la adhesión del soporte P2P.

Del mismo modo, se exteriorizaron las ventajas que los servicios P2P desarrollados le añaden al funcionamiento de PDLib.

En el siguiente capítulo se expondrá el funcionamiento del modelo en un ambiente operativo real, del tal modo que se puendan reafirmar, o terminar de comprender, los conceptos explicados recientemente.

Capítulo 4

Operación y Evaluación del Modelo de Interoperabilidad del Servidor de Datos de PDLib

4.1. Introducción

El objetivo del presente capítulo es presentar la ejecución del modelo de interoperabilidad en un entorno funcional, de tal manera que se logre apreciar y extrapolar las explicaciones ofrecidas sobre el diseño, operación e interrelación de sus componentes, aplicadas en un ambiente operativo real.

Iniciaremos exponiendo la organización de red que tiene la plataforma JXTA, para advertir la forma en como se transmiten los mensajes entre los puntos. Esta descripción es vital para comprender el entorno de operación del modelo de interoperabilidad implementado. Posteriormente describiremos el ambiente de prueba, enumerando y detallando cada uno de sus componentes. Y finalmente, explicaremos, principalmente a través de figuras, la operación típica de los servicios ofrecidos en el modelo de interoperabilidad desarrollado. En esta última etapa nos auxiliaremos de herramientas gratuitas utilizadas para monitorear elementos (grupos, puntos, *advertisements*, etc) de la plataforma JXTA.

Y en la penúltima sección del capítulo expondremos los resultados arrojados en la realización de las pruebas de los servicios remotos

implementados en el modelo, y finalmente concluiremos con el resumen del capítulo.

4.2. Organización de la red JXTA

Basándonos en [19] podemos afirmar que la red de JXTA es *ad-hoc*, adaptable, con múltiples dispositivos conectados como puntos (*peers*). Las conexiones pueden ser transitorias, y el enrutamiento de mensajes es no-determinista, es decir, que la ruta de los mensajes no puede ser predicha, debido a que existen múltiples caminos posibles. Los puntos pueden ingresar y salir de la red de JXTA en cualquier momento.

Para que dos puntos puedan comunicarse, ambos deben hacerlo a través de los protocolos de JXTA. En la práctica existen dos tipos de puntos:

- **Punto circundante con todas las capacidades (*Full-featured edge peer*)**.Tiene la capacidad de enviar y recibir mensajes, y típicamente guarda localmente los *advertisements* localizados en la red. Un punto circundante contesta a solicitudes de búsquedas (*discovery requests*) con información que es localizable en el conjunto de *advertisements* almacenados localmente, sin embargo, no retransmiten (*forward*) mensajes a otros pares. La mayoría de los nodos que conforma la red o grupo mundial de JXTA (*Net peer group*, p.28) caen en la calificación que estamos describiendo.

- **Punto de Encuentro (*Rendezvous peer*)**. Son nodos especiales que tienen por objetivo retransmitir (*forward*) mensajes con el objetivo de comunicar puntos ubicados en subredes diferentes. Un punto de encuentro puede ser visto como un encaminador (*router*) de mensajes. En Internet, Jxta provee de forma transparente estos tipos de nodos, lo que permite que un punto circundante al activar la plataforma de JXTA es capaz de comunicarse con otros pares por medio de esta infraestructura.

  Los puntos de encuentro, naturalmente, permiten construir topologías controladas (cuando un punto se establece como tal, una lista de otros puntos de encuentro debe ser especificada) para manipular eficientemente las comunicaciones necesarias del sistema propuesto.

Una solicitud de búsqueda se realiza por medio de mensajes de multidifusión (*multicast*) que se propagan por toda la subred física donde el punto de emisión reside. Si esta subred cuenta con un punto de encuentro, éste retransmitirá las peticiones a otros puntos de encuentro con los que mantiene comunicación, y así sucesivamente hasta que el destinatario es alcanzado; quien responderá directamente al punto que originó la búsqueda. La plataforma define un límite de 7 saltos en puntos de encuentros como tiempo de vida del mensaje, si el destinatario no es encontrado dentro de este margen se interpretará como punto caído.

La figura 4.1 ejemplifica la transmisión de una petición de búsqueda desde el punto B hacia los puntos que forman un grupo, los cuales residen en diferentes lugares de Internet. El punto B transmite vía *multicast* la solicitud a los miembros de su subred, y deja que el punto encuentro asignado (RDV 1) se encargue de retransmitir el mensaje a los otros puntos de encuentros remotos, quienes a su vez, reenviarán el mensaje original en el ámbito de su subred. Las dos saetas de ida y vuelta de las líneas comunicantes, significan que por la misma vía retornará la respuesta a la solicitud.

4.3. Entorno operativo

Las pruebas aquí presentadas se llevaron a cabo utilizando la infraestructura de comunicación institucional del Campus Monterrey, en particular las subredes, tanto alámbrica como inalámbrica asignadas al Centro de Investigación en Informática. Es importante señalar que la red interna del Campus Monterrey implementa un *firewall* de seguridad, lo implica que los equipos conectados no pueden contar con conectividad del tipo *end-to-end*, ocasionando la imposibilidad de hacer uso de puertos y protocolos necesarios para establecer comunicación con otros nodos alojados en Internet. JXTA sobrelleva este problema por medio de un servidor *proxy* de la red local, sin embargo, el puerto y el nombre del que dispone el campus, es información a la cual no tenemos acceso.

Por consiguiente, construiremos una topología de comunicación sirviéndonos de un punto de encuentro (*rendezvous peer*).

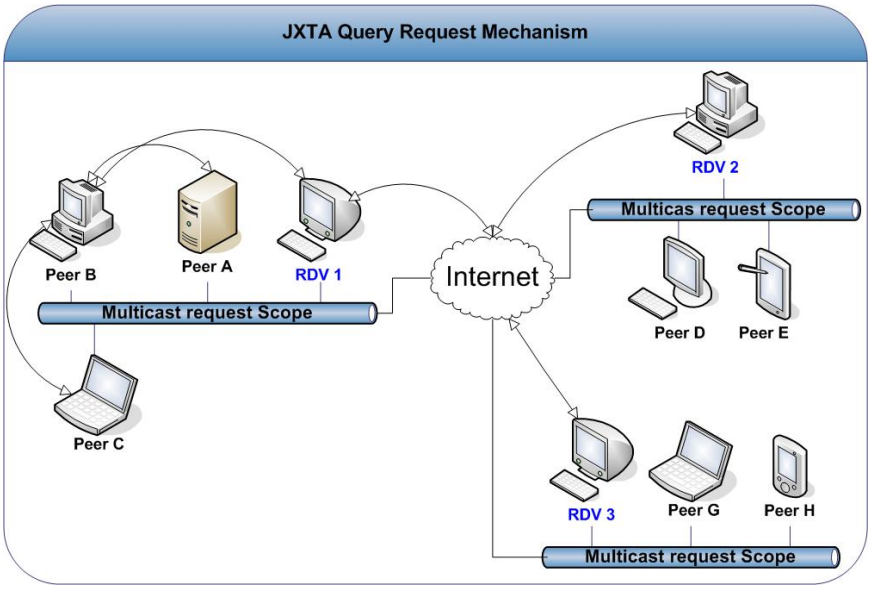

Figura 4.1: Mecanismo de envío y recepción de solicitudes de búsqueda en JXTA

Para la ejecución de estas pruebas se instalaron 4 servidores de datos en dos computadoras (dos instancias por equipo) con las siguientes características:

Computadora personal (PC) con procesador Pentium 4 de 3 Ghz, con 0.99 GB de Ram, y sistema operativo *Microsoft Windows XP Professional*.

Computadora personal portátil (*Laptop computer*) con procesador Pentium 4 de 2.8 Ghz y 488 MB de Ram, y sistema operativo *Microsoft Windows XP Home Edition*.

Las computadoras se encuentran situadas en subredes diferentes. La computadora portátil forma parte de la subred inalámbrica, en cambio la Computadora personal es miembro de la subred local alámbrica, tal y como se ilustra en la figura 4.2.

El modelo de interoperabilidad fue desarrollado con la versión JXTA Java 2.3.5 ("Shebakia" *released*).

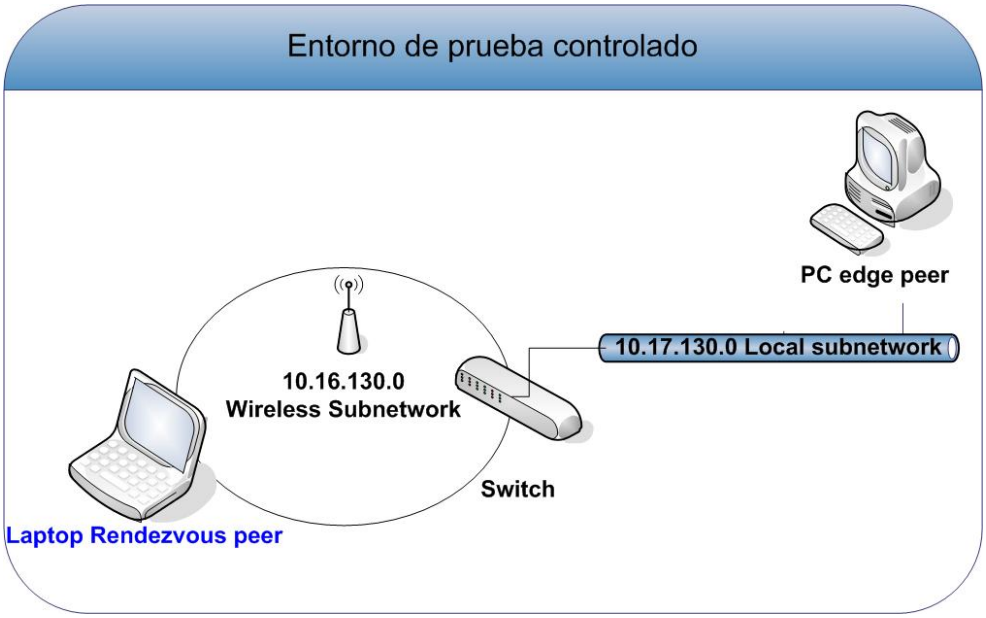

Figura 4.2: Entorno de prueba controlado

4.4. Funcionamiento del modelo de interoperabilidad

Todo servidor de datos lo primero que hace al comenzar a operar, es iniciar la plataforma de JXTA; si es la primera vez que está siendo ejecutado, entonces se adhiere al grupo PDLib, en caso contrario, esta operación no será necesaria, dado que el proceso de ingreso se lleva a cabo solamente en una sola ocasión, esto conlleva, que una vez que acontece esta operación, el nodo mantendrá por siempre la membresía del grupo.

El procedimiento de incorporación no es más que la creación de un identificador de grupo que sirve como atributo del par. Por tanto, todos los puntos que cuenten con esta misma propiedad, se verán como miembros del mismo grupo. Una vez concluida esta tarea, un objeto JxtaServerSocket es creado para escuchar permanentemente por solicitudes de servicios provenientes de otros pares. Por último, se genera una bifurcación del hilo del programa principal; por un lado, un hilo tendrá la función de iniciar y mantener el proceso de prestar oídos a los servicios valiéndose del

JxtaServerSocket recién establecido, mientras que el otro (el hilo original), pone a disposición los métodos y funciones que permiten enviar solicitudes a otros nodos. Es importante tener presente que esta división es la que hace posible que un servidor de datos, sea a su vez proveedor de servicios (*server side*), y demandante de éstos en otros nodos (*client side*). En el capítulo anterior presentamos el diagrama de actividad (figura 3.5, página 47) que ilustra este proceso.

4.4.1. Configuración de la plataforma JXTA

Iniciar la plataforma de JXTA implica contar con la facultad que tiene un punto de comunicarse con otros similares por medio de los protocolos y servicios que ésta provee. La primera vez que una aplicación comienza a operar, una ventana (JXTA *Configurator*) de configuración es desplegada para definir el tipo de punto que se generará, y los medios a los que recurrirá para comunicarse en el entorno de red. Como bien se ha dicho, un nodo puede ser circundante (*edge peer*) o de encuentro (*edge peer*), y en nuestro caso, que vamos a establecer una topología controlada, nos valemos de esta herramienta para asignar a cada punto el rol que le corresponde en el ambiente de red implantado.

La figuras 4.3, 4.4, y 4.5 muestran los 3 paneles de parámetros que constituyen este instrumento.

El panel principal (4.3) permite bautizar al nodo, también es posible ingresar una clave de seguridad, espacio que no aparece en la figura debido a que es información codificada en la implementación de la clase **PDLibDataServerPeerImplementation**. Los elementos de seguridad en JXTA son requeridos para casos en que políticas de esta índole son establecidas, como por ejemplo, cuando dos o más nodos requieran conocer las claves de sus contrapartes para poder intercambiar servicios entre si, algo que no es necesario en nuestro modelo.

El segundo panel (4.4) permite atribuir el rol bajo el cual el punto operará en el entorno. Y el último (4.5), presenta los campos disponibles para ingresar las direcciones (IP, protocolo y puerto) donde se localizan los puntos de encuentros que permitirán comunicar al nodo con pares de otras redes o subredes.

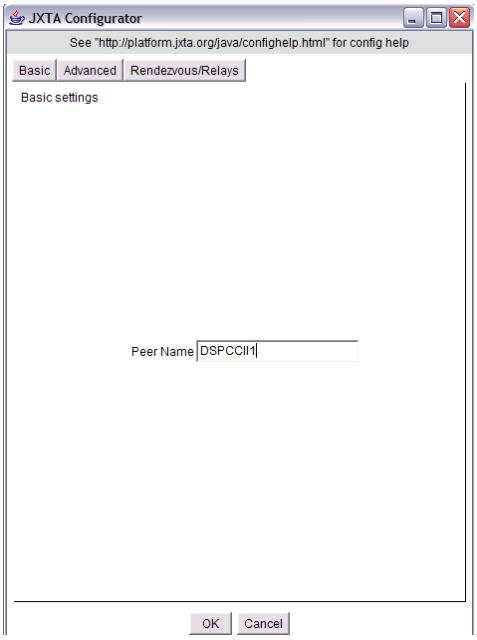

Figura 4.3: Panel 1 de configuración de la plataforma JXTA

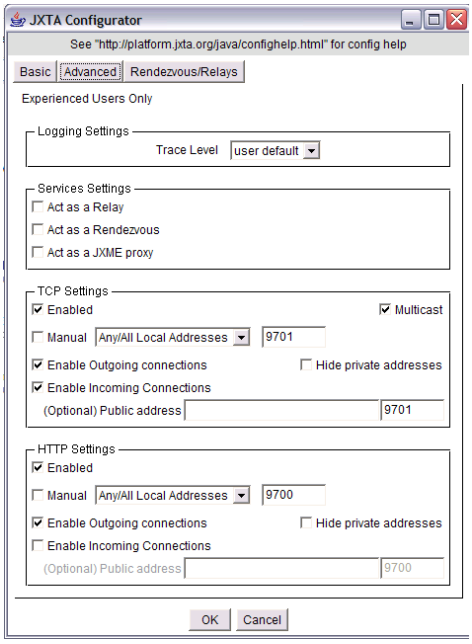

Figura 4.4: Panel 2 de configuración de la plataforma JXTA

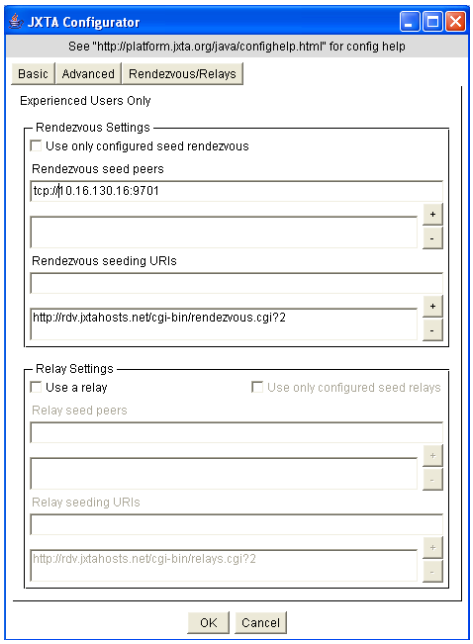

Figura 4.5: Panel 3 de configuración de la plataforma JXTA

4.4.2. Configuración y verificación del ambiente de prueba

Nuestro ambiente de pruebas, como se indicó la sección 4.3, lo constituyen 4 servidores de datos, nombrados de la siguiente manera:

Los que residen en la computadora portátil:

1. **DSManaguaLaptop**
2. **DSMulukukuLaptop**

Y los que se localizan en la computadora personal:

1. **DSPCCII1**
2. **DSPCCII2**

Al servidor de datos DSMulukukuLaptop se le asignó, por medio de la herramienta de configuración recién descrita, la responsabilidad de ser el punto de encuentro (*rendezvous peer*) del grupo PDLib. Las 2 instancias instaladas en la computadora personal utilizan la dirección física (IP, puerto, protocolo) de este nodo para definirlo como punto de encuentro con los puntos localizados en la red inalámbrica, así como lo demuestra la figura 4.5. Los pares residentes en una misma subred, tal y como se ha venido remarcando, no requieren los servicios de un punto de encuentro para comunicarse entre si, por tanto, en el nodo DSManaguaLaptop no se establece ninguno.

A nivel de código, para que un grupo pueda valerse de un punto de encuentro, el nodo que tiene asignada esta responsabilidad inicia este servicio en el ámbito del grupo al que pertenece, mientras que lo pares miembros situados en otras subredes, deben obtener los servicios de este nodo mediante una petición remota. Dado que todos los servidores de datos se componen del mismo código de programación, un parámetro de entrada es ocupado para indicar cual de estas tareas, conforme a su rol, el punto debe de realizar.

Existen aplicaciones desarrolladas para visualizar y monitorear un entorno de red de puntos generados con la plataforma JXTA. Estos instrumentos son sumamente útiles para detectar problemas o incongruencias en el comportamiento de pares que componen un sistema de instancias que cooperan entre sí.

Una de esas herramientas es el JXTA *Shell*, el cual permite a los usuarios interactuar con la plataforma a través de un interpretador de línea de comandos. Muy similar a un *Shell* de UNIX, este instrumento provee los servicios básicos y comunes existentes en la plataforma, como búsqueda de grupos, puntos, unión a grupos, entre otros.

Un *Shell* es un punto de la plataforma de JXTA al cual se le debe asignar un nombre. La figura 4.6 exhibe los comandos y los resultados que permiten verificar la existencia del grupo PDLib y de sus 4 miembros, desde un Shell bautizado como PCCocibolcaCIIShell, y ubicado en la computadora personal donde residen dos servidores de datos de tipo circundante (*edge peers*).

Para comprobar la presencia de los elementos de nuestro modelo, primero se busca remotamente los grupos existentes en el ambiente, luego se verifica la existencia de dichos grupos. Posteriormente el punto representado por el *Shell* se une al grupo PDLib y en el ámbito de este grupo, se buscan los nodos miembros.

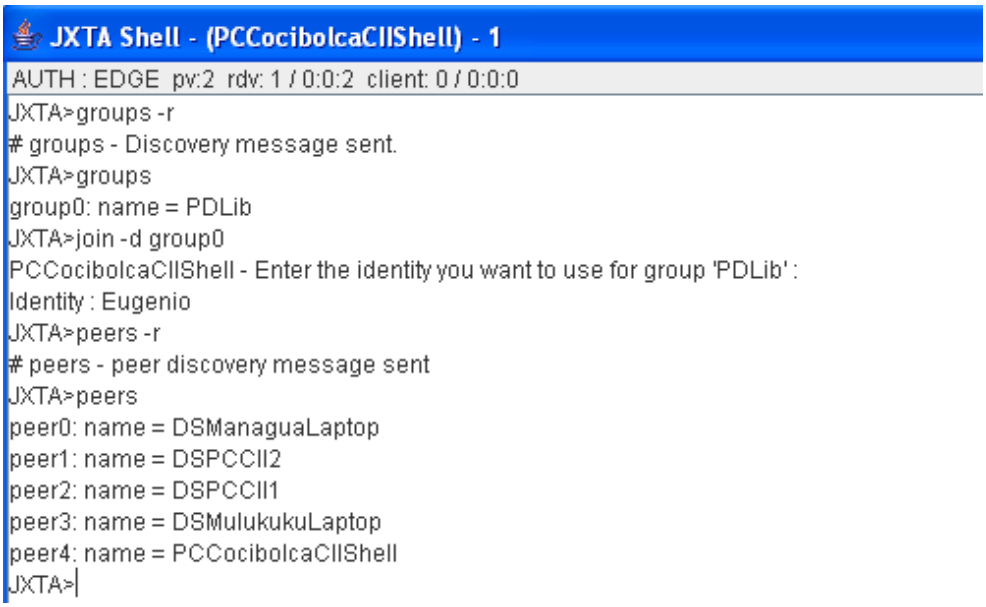

Figura 4.6: Comandos utilizados para comprobar la existencia de los puntos miembros del group PDLib

En la figura 4.7 se puede apreciar como el nodo DSMulukukuLaptop es el único punto de encuentro configurado para el grupo PDLib.

Otra aplicación conveniente para observar los puntos que están corriendo en un momento determinado es JXTNetMap, una utilidad visual que se auxilia de un punto de encuentro especial (*iViewRendezvous*) para representar gráficamente un entorno de red de JXTA.

El *iViewRendezvous* de JXNetMap es un punto de encuentro global, y no de grupo, común a todos los nodos de un entorno, de tal forma que visualmente sólo se aprecian los pares dados de altas y no la organización de éstos en grupos. La figura 4.8 presenta los 4 servidores de datos, más los otros pares que al momento de tomar imagen, estaban funcionando. El par DSMulukukuLaptop se distingue del resto por ser un punto de encuentro.

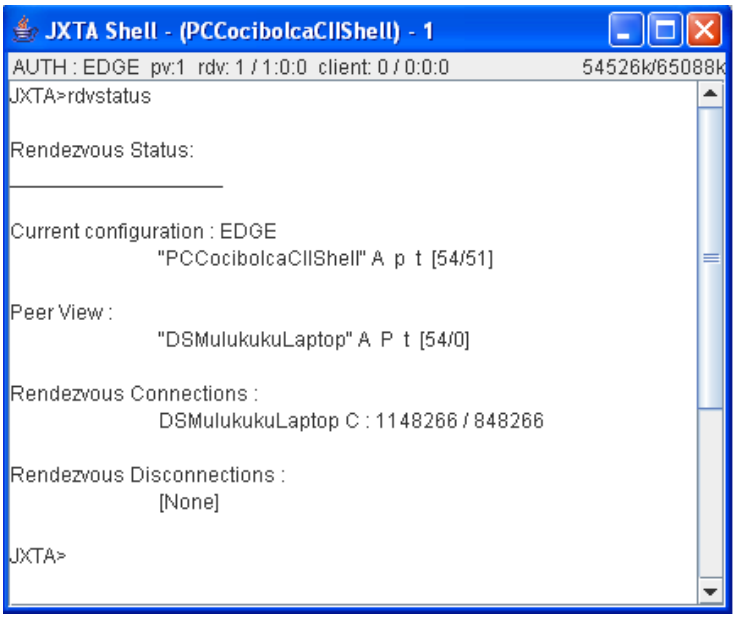

Figura 4.7: Verificación del punto de encuentro (*rendezvous*) utilizado

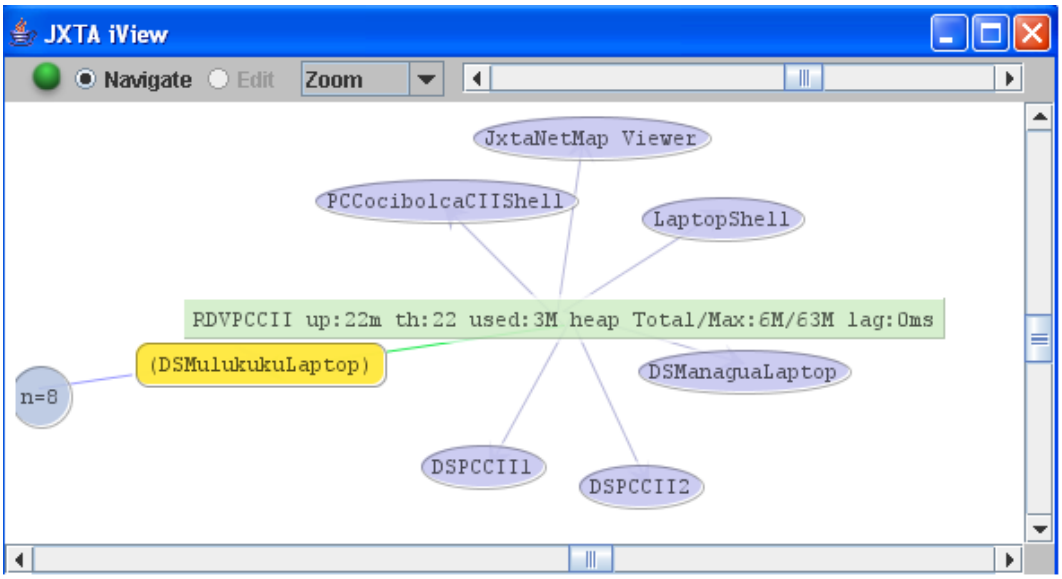

Figura 4.8:

4.4.3. Funcionamiento de los servicios remotos del sistema PDLib

Una vez concluidas las tareas que tienen por objetivo habilitar los medios de comunicación, el servidor de datos estará listo para escuchar por solicitudes de servicios proveniente de otras instancias, al mismo tiempo que habilitará los métodos que permiten realizar peticiones remotas a otros servidores.

Los servicios propios del sistema PDLib, descritos en el capítulo 3 y desarrollados para el alcance de esta investigación, son la búsqueda remota de documentos, y la copia remota de colecciones, los cuales fueron diseñados para ser llamados desde los módulos clientes asignados a un servidor de datos y desde él, transmitir las solicitudes a otras instancias, de la misma manera que se ilustra en la figura 3.6 .

Como se precisó en la sección 3.4 titulada Servicios, las peticiones de los clientes deben de ir dirigidas al nodo o nodos desde donde se desee obtener los servicios. Por tanto, los clientes deben de conocer cuales son los servidores de datos que se encuentran disponibles para recibir estas peticiones. Para acceder a esta información, que a nivel de implementación se compone por el nombre del servidor y su identificador, el servidor de datos provee un método (*searchForPeers*) que investiga en el ámbito del grupo PDLib cuales son los servidores de datos que se encuentran trabajando al momento en que se ejecuta este método, lo que arroja como resultado el almacenamiento local (en el servidor de datos que ejecuta la búsqueda) de los *advertisements* de los nodos encontrados. Pero debido a que esta indagación se lleva a cabo enviando asincrónicamente solicitudes de búsquedas de JXTA, se deben esperar varios segundos, mientras otras solicitudes se envían, antes de recibir una respuesta. El método recibe como parámetro de entrada la cantidad de peticiones que se deseen enviar en el proceso de búsqueda. En entornos controlados, similares al nuestro, dos peticiones son más que suficientes para encontrar los pares miembros de la constelación.

Una vez que el procedimiento de búsqueda concluye, es el momento de obtener, de los *advertisements* almacenados, los nombres e identificadores (ID) de los puntos encontrados, operación que se realiza a través de una función (*peersInLocalCache*) que regresa la información deseada en una matriz de dos columnas, la primera para el nombre del servidor de datos,

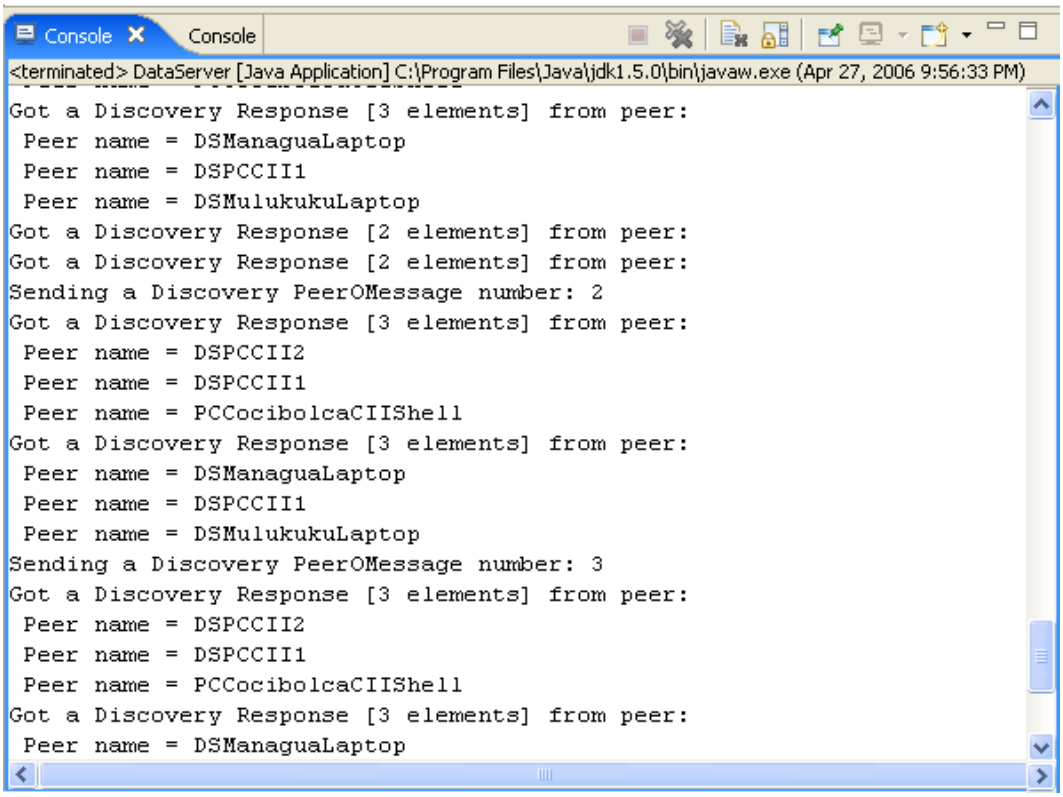

Figura 4.9: Consola de un servidor de datos al momento en que se ejecuta la búsqueda remota de puntos miembros del grupo PDLib

y las segunda para su identificador. Esta matriz se implementa por medio de la estructura de datos *Hashtable* de Java, y es esta tabla la que se regresa a los clientes. El usuario entonces, podrá decidir a cual o cuales puntos dirigir sus peticiones.

En la consola del servidor de datos que busca, por orden de un cliente, los pares dados de alta, se imprime el nombre de los que van siendo encontrados, tal y como se muestra en la figura 4.9. Y en la figura 4.10 se expone, una imagen del *Hashtable* que se retorna a los clientes, captada utilizando las herramientas de rastreo del ambiente de programación Eclipse. En ambas figuras se pueden apreciar los 4 servidores de datos empleados en el ambiente de prueba.

Los dos servicios remotos propios de PDLib, al momento de ser invocados

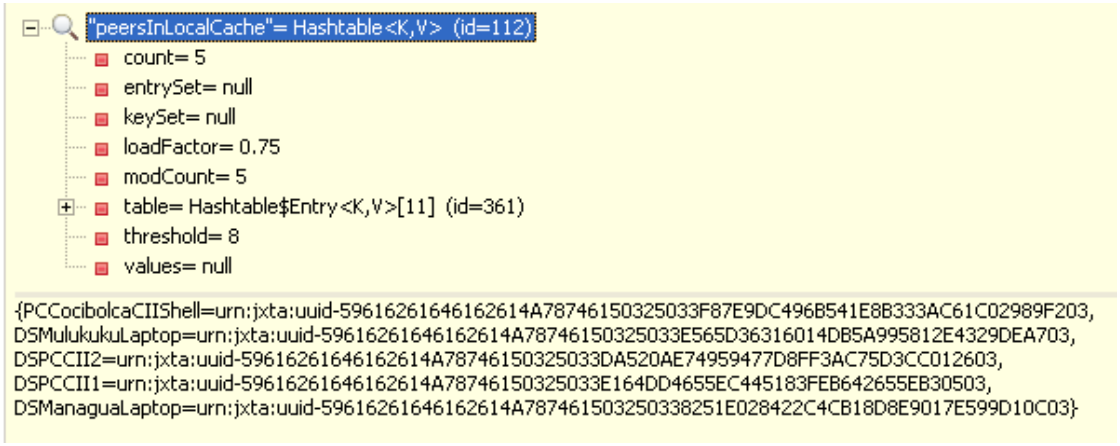

Figura 4.10: Vista en modo rastreo (*debug*)del *Hashtable*

crean un JxtaSocket, que en conjunto con el JxtaServerSocket del punto hacia donde se dirigirá la solicitud de servicio, formaran el canal de comunicación necesario para el traspaso de mensajes. Cuando el diálogo entre ambas instancias termine, los JxtaSockets se cerrarán y los JxtaServerSockets estarán listos para atender otras llamadas.

4.5. Evaluación de los servicios remotos implementados

En esta sección presentaremos la evaluación de las pruebas realizadas al servicio de búsqueda remota en función del tiempo de ejecución en contraposición al tiempo invertido en el servicio de búsqueda local.

Y expondremos las estadísticas del tiempo invertido en la ejecución del servicio de copia remota el última fase de esta sección.

4.5.1. Comparación del tiempo de ejecución del servicio de búsqueda local vs servicio de búsqueda remota

Las pruebas se efectuaron bajo el entorno de red exhibido en la sección 4.3 de la siguiente manera:

- Se utilizó un cliente de tipo pesado (*Thick clien*t) residente en la computadora portátil, desde donde se solicitaron, tanto los servicios de búsqueda remota, como local.
- Para las pruebas de los servicios remotos, el cliente pesado tenía asignado uno de los dos servidores de datos residente en la computadora portátil, el cual para fines de pruebas se definirá como Laptop1 (Mulukuku).
- La computadora personal tendrá, entonces, instalados dos servidores de datos, al igual que se definió en el entorno de prueba, más el cliente pesado; y la computadora portátil seguirá albergando otros dos servidores de datos.
- Las llamadas al servicio de búsqueda local se ejecutaron desde el mismo cliente pesado, pero cambiando la dirección IP del servidor que debía atender el servicio, tal y como se muestra en la figura 4.11.
- Las llamadas al servicio de búsqueda remota se llevan a cabo a través del servidor de datos intermediario asignado al cliente pesado. En este nodo también se ejecutará el servicio de búsqueda remota, a pesar de que sea el servidor asignado al cliente. En este caso la llamada remota será así mismo.Conceptualmente las llamadas se realizan tal y como se muestra en la figura 4.12.
- En las tablas y en las gráficas que se mostrarán a continuación se compara el tiempo invertido del servicio de búsqueda remota realizado desde el servidor de datos asignado al cliente, con el servicio de búsqueda local propio de cada servidor de datos.
- En cada evaluación se llevaron a cabo 4 pruebas, una en cada servidor de datos. En el caso del servidor de datos asignado al cliente pesado, de igual forma se ejecutaron tanto la búsqueda local como remota.

- El tiempo presentado en las estadísticas es el tiempo que le toma al servidor de datos ejecutar la operación, y no el tiempo desde que el cliente hace la llamada remota y recibe una respuesta. Lo que nos interesa medir es el tiempo que invierte el servidor de datos en realizar cada operación y no los tiempos en que se tranportan los objetos a través del protocolo XmlRpc.

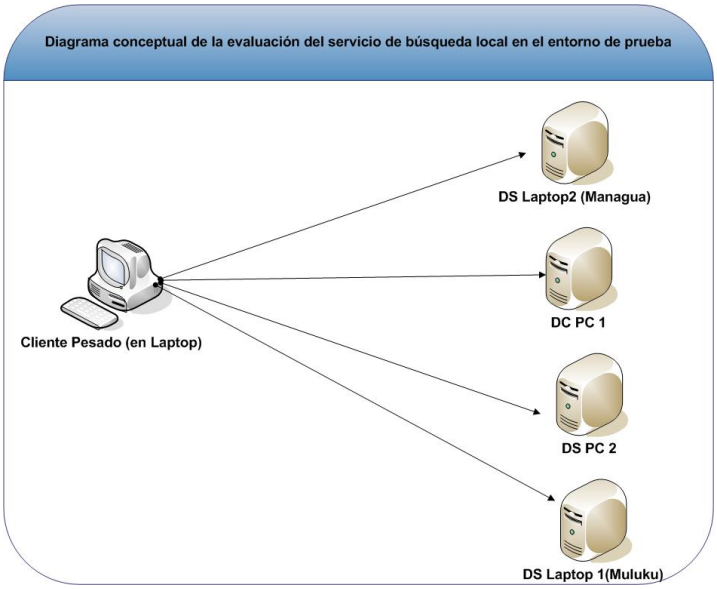

Figura 4.11: Diagram conceptual de la evaluación del servicio de búsqueda local en el entorno de prueba

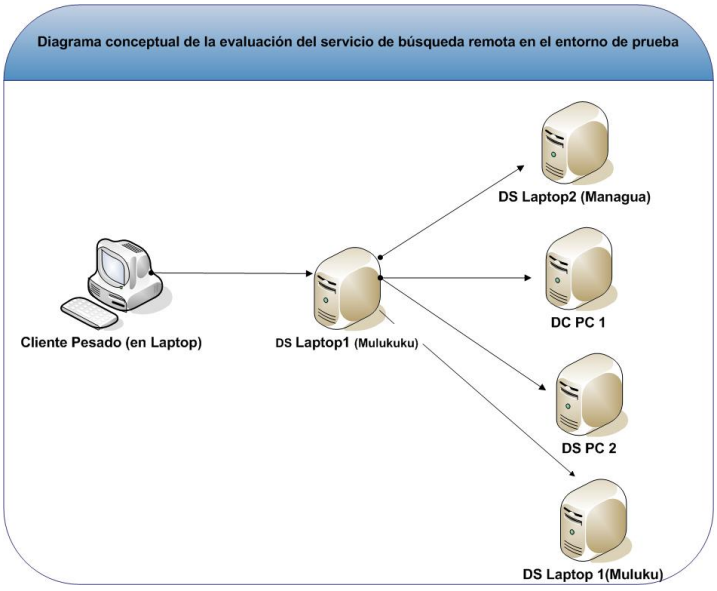

Figura 4.12: Diagram conceptual de la evaluación del servicio de búsqueda remota en el entorno de prueba

Servidor	Cantidad de documentos	PeerSearch	LocalSearch	Overhead
Laptop1	1	110	31	79
Laptop2	1	657	78	579
PC1	1	141	141	0
PC2	1	375	32	343
Totales	4	1283	282	1001

Cuadro 4.1: Evaluación 1

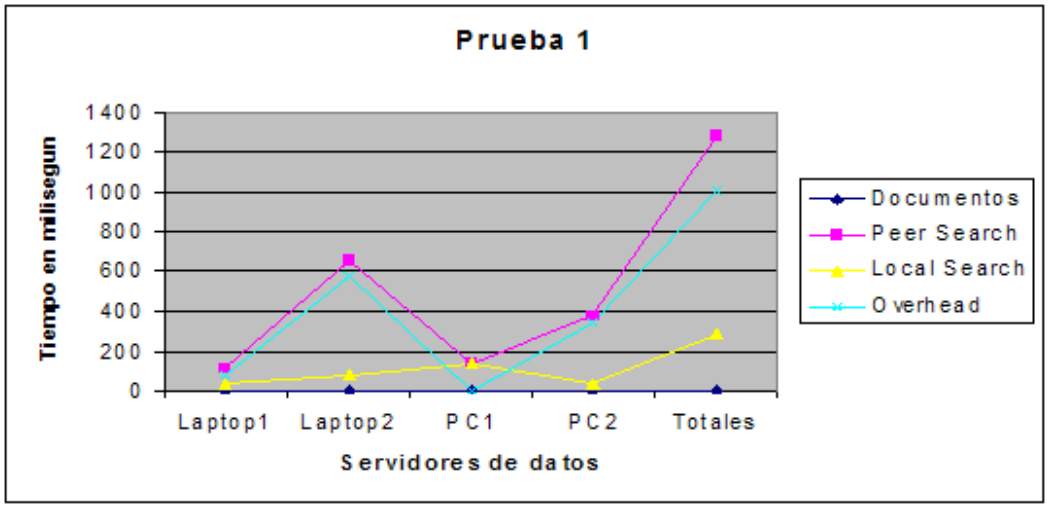

Figura 4.13: Representación gráfica de los tiempos de ejecución de los servicios de búsqueda (Evaluación 1)

Servidor	Cantidad de documentos	PeerSearch	LocalSearch	Overhead
Laptop1	2	141	56	85
Laptop2	2	484	125	359
PC1	2	437	46	391
PC2	2	328	46	282
Totales	8	1390	273	1117

Cuadro 4.2: Evaluación 2

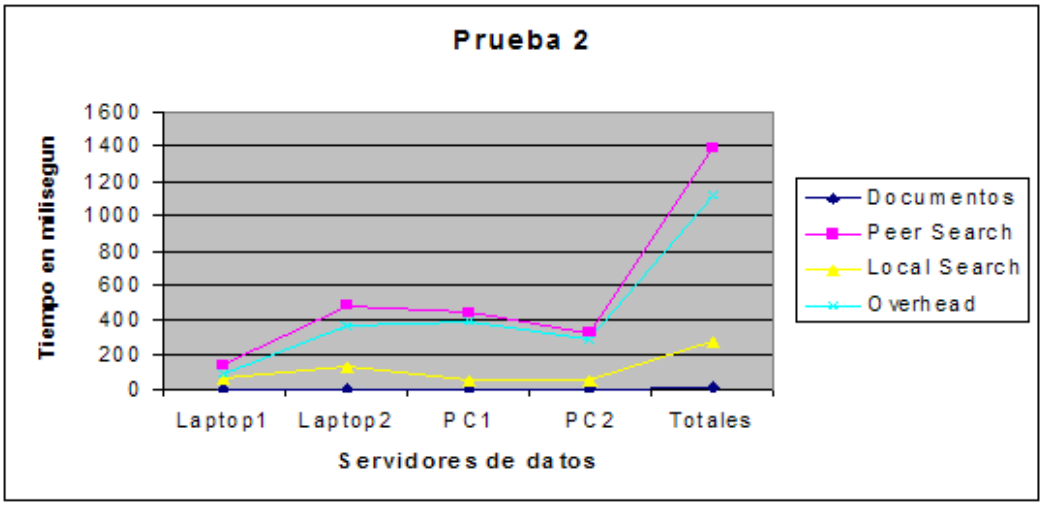

Figura 4.14: Representación gráfica de los tiempos de ejecución de los servicios de búsqueda (Evaluación 2)

Servidor	Cantidad de documentos	PeerSearch	LocalSearch	Overhead
Laptop1	3	319	62	257
Laptop2	3	204	187	17
PC1	6	281	156	125
PC2	6	219	203	16
Totales	18	1023	608	415

Cuadro 4.3: Evaluación 3

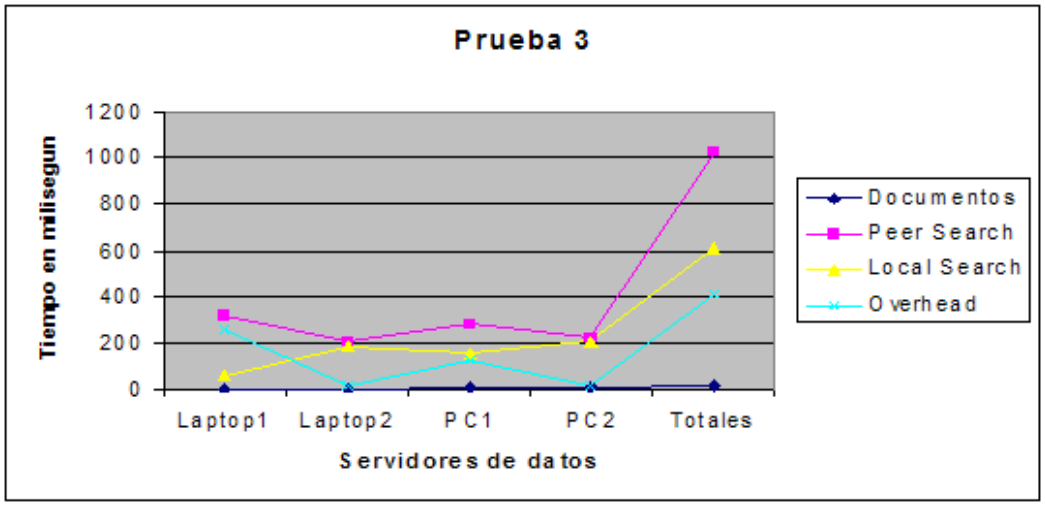

Figura 4.15: Representación gráfica de los tiempos de ejecución de los servicios de búsqueda (Evaluación 3)

Servidor	Cantidad de documentos	PeerSearch	LocalSearch	Overhead
Laptop1	4	407	63	344
Laptop2	4	375	109	266
PC1	7	390	63	327
PC2	7	266	140	126
Totales	22	1438	375	1063

Cuadro 4.4: Evaluación 4

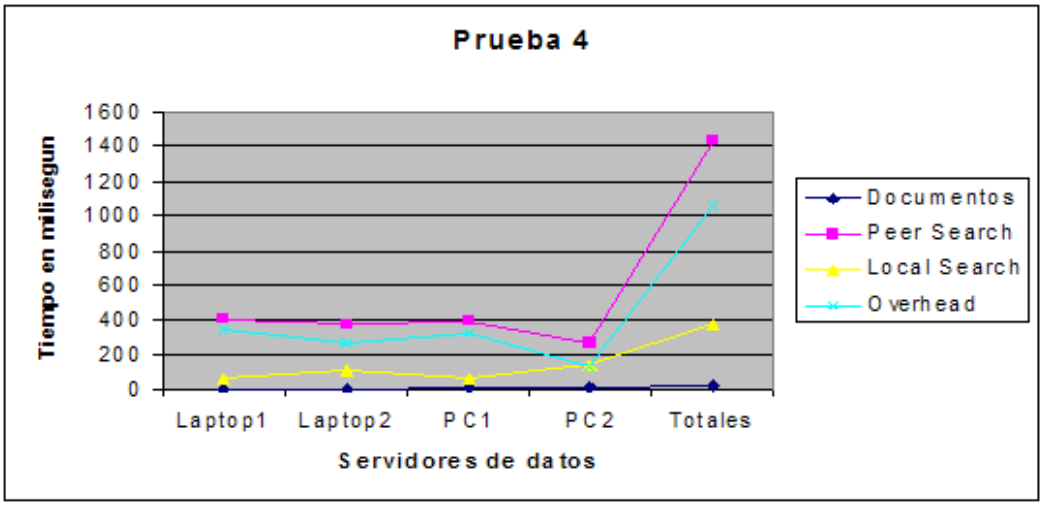

Figura 4.16: Representación gráfica de los tiempos de ejecución de los servicios de búsqueda (Evaluación 4)

Servidor	Cantidad de documentos	PeerSearch	LocalSearch	Overhead
Laptop1	5	344	109	235
Laptop2	5	297	93	204
PC1	8	250	79	171
PC2	8	359	62	297
Totales	26	1250	343	907

Cuadro 4.5: Evaluación 5

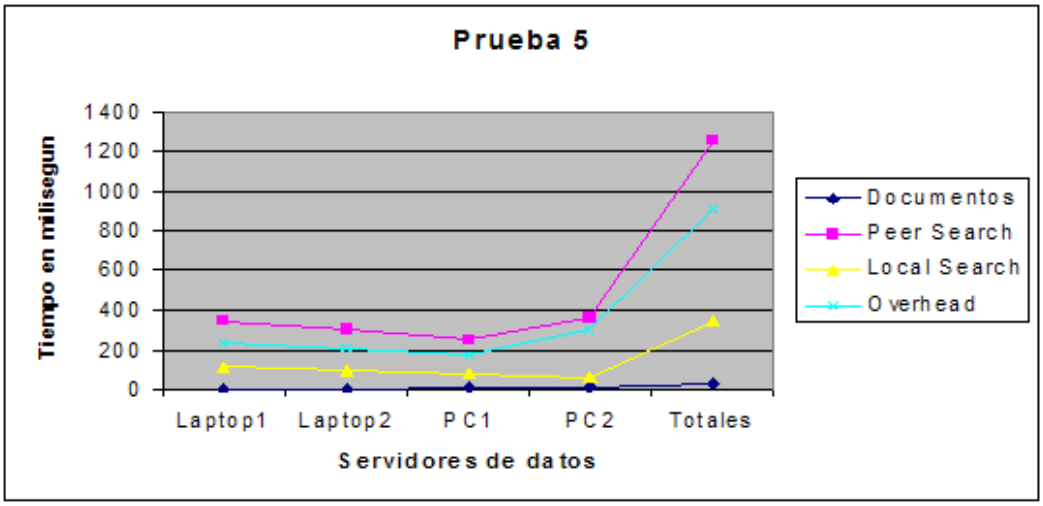

Figura 4.17: Representación gráfica de los tiempos de ejecución de los servicios de búsqueda (Evaluación 5)

4.5.2. Comparación del tiempo de ejecución del servicio de búsqueda local vs servicio de búsqueda remota

La evaluación siguiente muestra los tiempos que demora el trasalado de cada documento perteneciente a una colección. En total son 10 documentos los que contiene la colección que se copió de un servidor de datos residente en la computadora portátil, a uno instalado en la computadora personal. La figura 4.18 muestra la gráfica que compara el tiempo de demora con respecto al tamaño de cada documento.

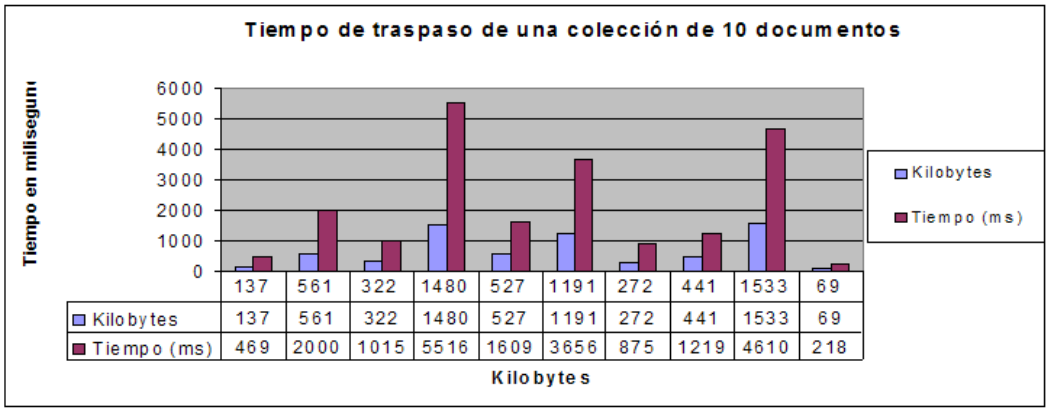

Figura 4.18: Tiempos de traslado de documentos pertenecientes a una colección

4.6. Resumen y conclusiones del capítulo

La operación del modelo de interoperabilidad entre instancias del servidor de datos, fue el contenido de este capítulo, en donde se explicó y apreció mediante un entorno de pruebas el funcionamiento y verificación de las partes contenidas en él.

Para constatar la disponibilidad de los servidores que formaron la constelación en el ambiente de prueba nos auxiliamos de dos herramientas gratuitas disponibles para lo comunidad de desarrolladores de aplicaciones basadas en la plataforma JXTA. Mientras que para comprobar los resultados

de los elementos necesarios para que los servicios remotos propios de PDLib se puedan desempeñar, recurrimos a la impresión de resultados en consola y a la facilidades de rastreo que provee el ambiente de programación Eclipse. También, se explicó como estos servicios establecen los canales de comunicación necesarios para entablar el diálogo particular de cada función con su respectivo correlativo.

Finalmente, se presentaron los resultados de varias evaluaciones hechas a los servicios remotos del modelo. Para comprender el grado de desempeño del servicio de búsqueda remota, se comparó el tiempo de ejecución de esta operación cotejándolos con los arrojados por el servicio local de búsqueda. El resultado de estas pruebas nos hace concluir que el tiempo de *overhead* entre estas funciones es en promedio de 1 segundo en el ambiente controlado; lo que nos hace ver que el desempeño en este entorno arroja valores aceptables para los usuarios de esta operación. También es importante notar que los tiempos de las operaciones de búsquedas no son dependientes de la cantidad de objetos encontrados, dado que se presentaron casos en que se reflejaron tiempos menores en solicitudes que regresaban mayor número de objetos que correspondían con los parámetros de búsqueda. Esto nos hace concluir que la función de búsqueda local no es dependiente de la cantidad de objetos localizados.

Por otro lado, se exhibieron los tiempos que toma el traslado de los documentos de una colección según su tamaño. En este caso no existe otro servicio con el cual se puedan comparar los resultados, por tanto, los datos expuestos tienen la finalidad de mostrar, a través de un ejemplo, el desempeño de este servicio.

En el capítulo siguiente se abordaran las conclusiones obtenidas en el presente trabajo de investigación, y se hablará del rumbo que a manera de trabajo futuro puede tomar.

Capítulo 5

Conclusiones y Trabajo Futuro

5.1. Conclusiones

La arquitectura cliente-servidor es el principal modelo de distribución empleado en sistemas informáticos, fundamentalmente de aquellos disponibles en Internet. Sin embargo, son varios e importantes los problemas que deben afrontar las aplicaciones construidas bajo esta estrategia, y que aspiran a ser soluciones idóneas para atender la demanda masiva de usuarios. Las principales deficiencias se muestran a continuación:

- Intolerancia a fallas.
- Escalabilidad de almacenamiento limitada a la capacidad del servidor.
- Detrimento del desempeño a medida que aumenta lademanda de usuarios.

El modelo punto-a-punto surge, entonces, como un camino alternativo para sobrellevar las limitaciones inherentes del paradigma cliente-servidor. Principalmente aboga por la distribución equitativa de los "deberes y derechos"de los componentes miembros del sistema, lo que significa que el peso de la responsabilidad de la ejecución de la tareas de una aplicación no recaen en una sola entidad, sino más bien en todos los miembros de la "comunidad".

El aporte de esta investigación radica precisamente en implementar un modelo punto-a-punto en el núcleo del sistema PDLib, el servidor de datos.

El modelo se compone de los siguientes elementos:

- **Punto**: Es una instancia independiente del servidor de datos, que a su vez, cuenta con la habilidad de establecer comunicación con otras instancias. Cada punto tiene la facultad de asumir tanto el rol de servidor, como de cliente.
- **Grupo**: Ámbito de colaboración entre las distintas instancias. Esta cooperación se materializa por medio del intercambio de los servicios implementados en el servidor de datos. Este grupo es identificado bajo el nombre de PDLib
- **Comunicación**: Son los mecanismos utilizados para la transferencia de objetos de datos entre las instancias miembro del grupo PDLib.
- **Servicios**: Son las versiones remotas de los servicios propios del servidor de datos. Cada servicio puede ser atendido por cualquier instancia de la constelación, de la misma manera, que cualquiera puede ser la entidad demandante de la operación.

Cualquier servicio implementado en un servidor de datos puede ser desarrollado a su versión remota. El presente modelo cuenta con la implantación del servicio de búsqueda remota, y copia remota de colecciones.

La búsqueda remota proporciona la capacidad de ubicar recursos residentes en servidores de datos que pueden estar instalados en cualquier lugar de Internet, proporcionando, de este modo, una ampliación de los resultados ofrecidos por la búsqueda local.

Mientras que la copia remota de colecciones es una funcionalidad que puede ser aplicada para respaldar información, evitando de este manera, contar con único punto de fracaso en el eventual caso de que uno de los servidores de datos asignados al usuario salga de línea. También, el transporte de información puede ser destinado para situar estratégicamente la información en nodos que geográficamente están más próximo al usuario, obteniendo, gracias a ello, un mejor un desempeño en el tiempo de envío y respuesta de solicitudes.

Gracias a estos servicios se resuelve el problema de escalabilidad del almacenamiento, porque los objetos de la biblioteca digital pueden residir en dos o más computadoras, y serán accesibles desde cualquier nodo mediante el traspaso de peticiones y objetos en los canales de comunicación definidos entre las instancias miembros del grupo de servidores de datos de PDLib.

La arquitecturas cliente-servidor proponen aumentar las capacidades de las entidades servidoras, y definir estrategias de respaldo que cubran el total de las responsabilidades de estos nodos, de tal forma que se pretende resguardar la entidad concentradora de la información y de los servicios. Es en esta característica donde el modelo propuesto difiere de este tipo de soluciones. El modelo de interoperabilidad distribuye en cada instancia la responsabilidad del funcionamiento del sistema en las instancias participantes, las cuales son independientes unas de otras, y no imagenes o respaldos de un nodo.

La implementación de este modelo de interoperabilidad fue realizada bajo la plataforma de computación punto-a-punto JXTA, y es a través de la infraestructura de comunicación que provee, que la colaboración entre instancias de servidores de datos es posible.

Los sistemas punto-a-punto basados en JXTA, pueden valerse de una infraestructura pública, la cual está disponible en Internet, o de un entorno controlado, que debe de ser planificado por los arquitectos del sistema.

Fue en un ámbito controlado donde se ejemplificó la operación del modelo, y cómo sus componentes interactúan entre sí.

Finalmente, para concluir con este último capítulo y con la exposición del presente trabajo de tesis, presentaremos, en el siguiente apartado, el rumbo que a manera de trabajo futuro a este proyecto se le puede dar continuidad.

5.2. Trabajo Futuro

A través del modelo de interoperabilidad implantado se pueden construir las versiones remotas de todos los servicios que ofrece un servidor de datos. De tal manera que debe de ocurrir una planificación de los servicios cuyas versiones remotas deben ser desarrolladas para agrandar el conjunto de prestaciones ofrecidas a los potenciales usuarios del sistema PDLib.

También es relevante hacer ver, que a pesar de que el modelo de interoperabilidad cuenta con la facultad de transportar colecciones de documentos, lo que resuelve varios de los problemas planteados y que fueron motivo de origen de esta investigación; no existe una estrategia de operación para la ejecución de este servicio; lo que obliga a definir las condiciones específicas bajo las cuales se hará uso de esta funcionalidad, tales como, adaptar el esquema de acceso público-privado existente en cada servidor de datos, a tal grado que existan los permisos y la congruencia entre cuentas de usuarios necesaria para que el traslado de colecciones pueda llevarse a cabo.

Por otro lado, este modelo forma parte de la capa de interoperabilidad del servidor de datos, en conjunto con el soporte OAI-PMH. La ampliación de los servicios de esta capa es un campo de trabajo fértil, que actualmente está siendo abonado por la construcción de los mecanismos que habiliten la interacción entre servidores de datos y repositorios DSpace.

Finalmente, se deben de establecer patrones para elegir automáticamente los servidores de datos donde se deben ejecutar las búsquedas, o el resto de servicios remotos, dado que actualmente es decisión de los usuarios clientes elegir los repositorios remotos desde donde se obtendrán los servicios deseados.

Bibliografía

[1] Dspace. Available at: http://dspace.org/introduction/index.html, May 2006.

[2] Freenet. Available at: http://freenet.sourceforge.net/, May 2006.

[3] Gnutella. Available at: http://www.gnutella.com/, May 2006.

[4] Jibe. Available at:http://www.jibeinc.com/, May 2006.

[5] Kazaa. Available at: http://www.kazaa.com/, May 2006.

[6] Music brain metadata. Available at:http://www.jibeinc.com/, May 2006.

[7] Napster. Available at: http://www.napster.com/, MAY 2006.

[8] N.R. Adam, R. Holowczak, M. Halem, N. Lal, and Y Yesha. Digital library task force. *Computer*, 29:89–91, August 1996.

[9] Rodrigo Ruiz Baca. Servicios de bibliotecas digitales a través de una arquitectura punto a punto en dispositivos móviles. Master's thesis, Instituto Tecnológico de Estudios Superiores de Monterrey, 2005.

[10] Daniel Brookshier, Darren Gonovi, and Navaneeth Krishman. *JXTA: Java P2P Programming*. Sams, 2002.

[11] Dublin Core. Dublin core metadata initiative. *Available at: http://dublincore.org/, last visited May 2006*, 2005.

[12] Carlos Yanuario Rivero Gomez. Esquema de interoperabilidad entre bibliotecas digitales basado en open archive y en encapsulamiento de

datos. Master's thesis, Instituto Tecnológico de Estudios Superiores de Monterrey, 2003.

[13] The Open Archives Initiative. The open archives initiative. *Available at: http://www.openarchives.org/ last visited May*, 2006.

[14] Joint International Conference on Digital Libraries. *PDLib: Personal Digital Libraries with Universal Access*, Denver CO, July 2005.

[15] Francisco Álvarez Cavazos. Distribución de datos para bases de datos distribuidas, una arquitectura basada en componentes de software. Master's thesis, Instituto Tecnológico de Estudios Superiores de Monterrey, 2003.

[16] Andy Oram. *Peer-to-peer : harnessing the benefits of a disruptive technology*. O'Reilly, 2001.

[17] David A. Garza Salazar, Lorena G. Gomez Martinez, Juan C. Lavariega Jarquin, Juan A.Ñolazco Flores, and Martha Sordia Salinas. Pdlib: The personal digital library project. Technical report, ITESM Monterrey, 2004.

[18] Roberto García Sáchez. Técnicas de adaptación a la conexión para clientes móviles que accesan servicios de biblioteca digital. Master's thesis, Instituto Tecnológico de Estudios Superiores de Monterrey, 2004.

[19] Sun Microsystems. *JXTA v2.3.x: Java(tm) Programmer's Guide*, April 2005. Available at: www.jxta.org, last visted May, 2006.

[20] XML-RPC. Simple cross-platform distributed computing, based on the standards of the internet. *Available at: http://www.xmlrpc.com/, last visited May 2006*, 2005.