Consideraciones iniciales para implementar funciones de búsqueda federada por medio de una red tipo igual-a-igual (P2P) de repositorios basados en Web Semántica.

Alejandro Gilberto Martínez Romero.

Introducción

Las bibliotecas digitales son espacios virtuales que facilitan el acceso, uso y generación de conocimiento. La Corporación Universitaria para el Desarrollo de Internet (CUDI) reúne a varias instituciones que propician un avance acelerado en la construcción de bibliotecas digitales en México y su participación en la comunidad internacional.

Como parte de este objetivo la CUDI financió el proyecto *Red Abierta de Bibliotecas Digitales* (RABiD); esta red a su vez ha apoyado la elaboración del sistema Pescador, el cual es un software para servidores Web cuya función principal es generar las páginas de colecciones digitales en línea. En su funcionamiento interno el sistema almacena los catálogos de las colecciones en una representación de conocimiento, de la cual extrae los datos “al vuelo” para construir las fichas que se soliciten. Se espera que este mecanismo permita a mediano y largo plazo la implementación de funciones especializadas para la consulta, análisis, gestión y catalogación de los fondos digitales [3].

El proyecto RABiD y el desarrollo del Pescador apoyarán los esfuerzos que se realizan en las instituciones que integran CUDI para construir una red abierta de bibliotecas digitales, la cual permitirá el acceso federado a colecciones y servicios digitales disponibles y facilitará la participación de nuevas bibliotecas digitales.

Como parte de una instrumentación para lograr representaciones de conocimiento cada vez más completas y relacionar colecciones de datos de diferentes procedencias, cualquier servidor que ejecute el Pescador podrá estar en comunicación con otros servidores Pescador, cada uno teniendo su propia representación de las cosas, como a su vez sus propios datos.

Esto nos lleva a preguntarnos: si realizamos una consulta en un servidor, y otro servidor cuenta con datos que son relevantes para nuestra consulta, entonces, ¿cómo poder acceder a estos datos? Como posible solución a este problema, pensamos en un sistema de búsqueda y manejo de resultados, dentro de una red de servidores que ofrecen cada uno, una perspectiva posiblemente diferente a la de los demás Pescadores.

Búsqueda de datos en la red de Pescadores.

El sistema de búsqueda del Pescador realiza sus operaciones localmente, por lo cual está restringido a un muy limitado número repositorios de conocimiento; además de que si varios Pescadores se encuentran funcionando en máquinas diferentes, sus repositorios quedarán aislados unos de otros impidiendo lograr una relación eficaz de sus datos.

Una solución interesante para la compartición de los datos y formar una búsqueda global con los datos de cada Pescador es la de integrar los sistemas en una red de Pescadores de igual a igual, es decir una red P2P (acrónimo en inglés de *peer-to-peer*). Algo que es importante mencionar con relación al lenguaje de este trabajo es que, como las redes hacen referencia a los nodos como sus elementos y las gráficas de la Web Semántica también lo hacen. Entonces cuando el significado del término “nodo” no queda claro a partir del contexto en el cual se usa, se procura especificar si se trata de un nodo de la red o un nodo de una gráfica WS

Las redes P2P nos permiten tratar con las consecuencias primarias de una red, ellas son principalmente proveer recursos y compartir ancho de banda, poder de cómputo, almacenamiento y datos. De esta manera cada Pescador servirá de cliente y servidor a la vez en la compartición de los resultados de búsqueda que cada uno llevará independientemente en su repositorio de conocimiento.

Las redes P2P se pueden clasificar de acuerdo a si es:

- Centralizada
- Descentralizada
    - Estructurada
    - Des estructurada
    - Híbrida

Centralizada.

Una red P2P centralizada se refiere a que un servidor principal está gestionando los recursos y los datos que en la red se están compartiendo, así, si cualquier Pescador se quiere incorporar a la red, necesitaría hacer una solicitud con el servidor central para poder transmitir sus datos a los demás Pescadores.

Ventajas

- Seguridad en el control del acceso de la información.

Desventajas

- La red es muy susceptible a fallos, ya sea por carga de datos en el servidor central o la caída de la red que conecta a este servidor.

Descentralizada.

Una red P2P descentralizada es básicamente que todos los Pescadores que quieran integrarse a la red sólo necesitan saber con quién conectarse, y sus transacciones son transmitidas a los demás sin necesitad de un control central.

Ventajas

- La red es muy robusta, y asegura que los datos permanecerán accesibles sin importar que la mayoría de los “Pescadores” estén desconectados de la red.
- Posiblemente mayor facilidad para el manejo descentralizado de la información.

Desventajas

- No hay un control central el cual permita establecer qué Pescadores pueden incorporarse a la red y cuáles no.

Desestructurada.

Consiste en que todos los participantes son tratados como nodos de una red los cuales pueden estar conectados a un número arbitrario de nodos. Tales redes pueden ser construidas fácilmente sin necesidad de preocuparse de las relaciones entre los nodos. Pero su principal desventaja es la de no asegurar o de tener una pobre resolución de las peticiones de búsqueda de los datos en cada nodo determinado, sobre todo si son datos raros o poco conocidos por los demás nodos,

además causa un elevado tráfico en la red en su afán de buscar datos en varios nodos.

Estructurada.

Las redes P2P estructuradas emplean un protocolo de consistencia global para asegurar que cualquier nodo puede eficientemente direccionar la búsqueda a un dato determinado, incluso si este dato es extremadamente raro, garantizando que se puede encontrar cualquier dato almacenado en la red.

Híbrida.

Las redes P2P híbridas pueden ser redes centralizadas o descentralizadas, esto nos ofrece el poder de ambos paradigmas.

Justificación

Como se vio las redes P2P tienes distintas estructuras, cada una de ellas buenas de acuerdo a la utilidad para las que se les puede dar empleo. En nuestro caso para la redes de Pescadores es vital tener una red P2P estructurada con la opción a escoger si esta puede ser híbrida o sólo descentralizada.

La opción de una red híbrida puede resultar interesante, ya que ofrecería simultáneamente las ventajas de una red centralizara (que permitiría un control mínimo cuando “nuevos” servidores Pescador se van agregando a la red) y descentralizada (siempre se tendría la posibilidad de encontrar algún servidor Pescador en la red, y se minimizaría la carga de datos que pasen por la red distribuida, lo cual evitaría posibles aglomeraciones o retrasos de datos).

Propuesta para la arquitectura

Una propuesta básica para la arquitectura del “Pescador” la cual agrega la capacidad para el manejo de las transacciones de la búsqueda en una red P2P es la siguiente:

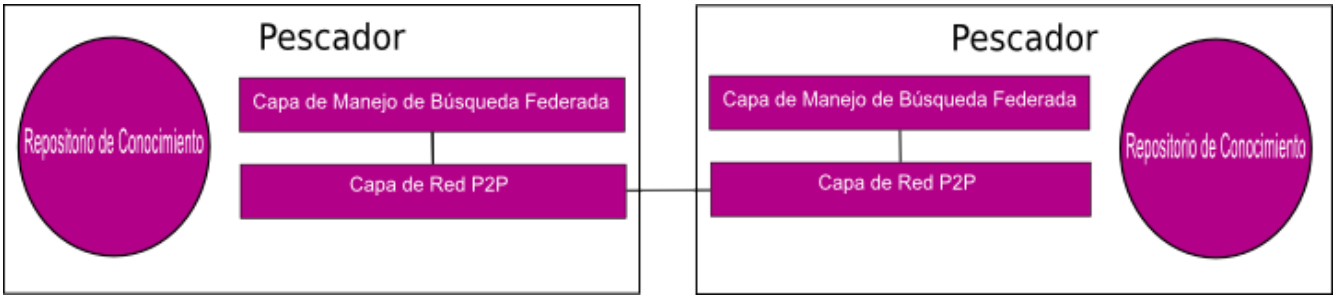

Figura 1

La capa de red P2P maneja las conexiones a los nodos y los datos transmitidos a través de la red y su direccionamiento, así como también las peticiones para la distribución de los objetos de la búsqueda.

La capa de manejo de búsqueda federada, reúne y manipula las peticiones y respuestas de la búsqueda en los demás “Pescadores” conectados a la red, como a su vez en el mismo “Pescador” local, para luego pasarlos a la vista del usuario que realizó la búsqueda.

Propuesta Inicial de Servicios y Algoritmos

La propuesta inicial para poder realizar la búsqueda de datos a través de la red P2P es ofrecer servicios a partir de las operaciones básicas de las redes antes mencionadas, y establecer mecanismos para la equiparación de los nodos equivalentes en la gráfica WS.

Los nodos de la WS son elementos almacenados en computadora por el sistema Pescador. Estos nodos son representaciones de objetos en el mundo real; por ello puede existir el caso en que dos Pescadores tengan representaciones del mismo objeto del mundo real pero cada Pescador con leves variantes en la forma en que se estructuran las gráficas WS que se refieren a dicho objeto (varias cosas pueden referirse a lo mismo pero tener nombres diferentes).

Por ejemplo, supongamos que en el servidor A tenemos un nodo de WS con un nombre único, el cual es *photo_01*, y en el servidor B otro nodo de WS con nombre *photo_08*. Pero ambos representan a la misma fotografía en el mundo real. A eso nos referimos cuando hablamos de que dos nodo en servidores distintos se refieren a un mismo objeto en el mundo real. Habrá que establecer

criterios para determinar cuándo esto es el caso.

Parece sencillo en un principio pero se necesita ser cuidadoso en la parte de decisión. Un problema importante que se debe resolver es cómo procesar los resultados semejantes y evitar repeticiones de datos, para que al momento de desplegar los resultados al usuario, estos se muestren en una manera concisa y clara.

Para comenzar con la propuesta de manejo de datos de los Pescadores, los servicios pueden ser:

a) A partir de una red P2P formada por sistemas “Pescador”, se crearán grupos que determinen el espacio de búsqueda, lo que evitará retrasos en las operaciones en la red.

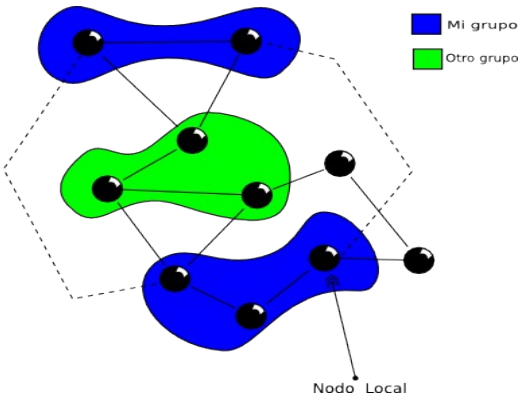

Figura 2

En la imagen se encuentra una red con sistemas Pescador, esta a su vez tiene dos grupos (verde y azul), y nuestra máquina se encuentra en el llamado nodo local el cual es la fuente de las peticiones para la búsqueda. Como se muestra, los elementos del grupo no necesariamente se encuentran conectados directamente, sino pueden estar en otra parte de la red, así mediante una conexión virtual la cual utiliza los nodos internos de la red para comunicarse con los demás nodos, hace parecer a esos Pescadores parte del grupo.

b) Determinados los grupos de búsqueda, se pueden hacer dos cosas para la transmisión de datos hacia los nodos del grupo.
El primero es un *multicast* en el cual el nodo local transmite las peticiones a todos los nodos del grupo. Este tiene la desventaja de incrementar

demasiado el trafico en la red del nodo local por la continua comunicación hacia el grupo, lo cual con un grupo demasiado grande la red se saturaría. El segundo es por medio de propagación, este método transmite los datos a un nodo determinado, el cual a su vez transmite los datos a otro nodo y así sucesivamente hasta que todos los nodos tengan una copia de la petición, cada uno la atiende y regresa la información resultante a los nodos que le enviaron la petición.

c) La equiparación de nodos de la gráfica WS sería el paso más complejo en la etapa de búsqueda, ya que es aquí donde una vez reunidos los datos necesitamos saber cómo relacionarlos y cuándo estamos teniendo datos semejantes para realizar la unificación de los resultados. Esto lo detallaremos en la siguiente sección. Por ahora los requerimientos para el procesamiento de los resultados de la búsqueda serían: especificar el tipo de dato en que la función de búsqueda regrese sus resultados (Cadena de texto, subgráfica WS, etcétera). La capa de manejo de búsqueda federada debe de preparar los datos para enviarlos a la capa inferior, especificando cuantos resultados por Pescador se pueden aceptar, coordinar los resultados que van llegando de los nodos restantes del grupo, como a su vez el de equipararlos, y finalmente pasarlos a capas superiores para su muestra al usuario.

Mecanismos para la equiparación de nodos

Aquí el reto radica en definir de manera adecuada o suficientemente coherente la equiparación de nodos, como para que los resultados de todos los nodos del grupo sean consistentes con lo especificado en la búsqueda del usuario.

Supongamos que tenemos dos servidores, uno es el servidor A y otro el servidor B, ambos tienen en común la ontología “**prueba**” y estamos buscando las fotografías que estén relacionadas con la persona Juan Pérez. Como se ve en la figura 3, el servidor A tiene una foto relacionada con Juan Pérez, con el *uri photo_01* y cuyo título es “Vista Nocturna”, mientras que el servidor B tiene el *uri photo_08* (foto que en el servidor A esta con el uri photo_01); esto a fin de cuentas para la unificación de los datos de la búsqueda es un problema, ya que si

ambos tuvieran el mismo *uri*, la búsqueda daría sólo un resultado, además este resultado tendría los libros que se publican con esa foto, cosa que el servidor A no tiene esto en su repositorio, pero sí lo tiene el servidor B. En el caso de estos *uris* (que son distintos), se tendrían dos resultados, uno el del servidor A y otro el del servidor B.

Entonces la equiparación de nodos en la capa de manejo de búsqueda federada, puede estar basada en alguna de tres opciones:

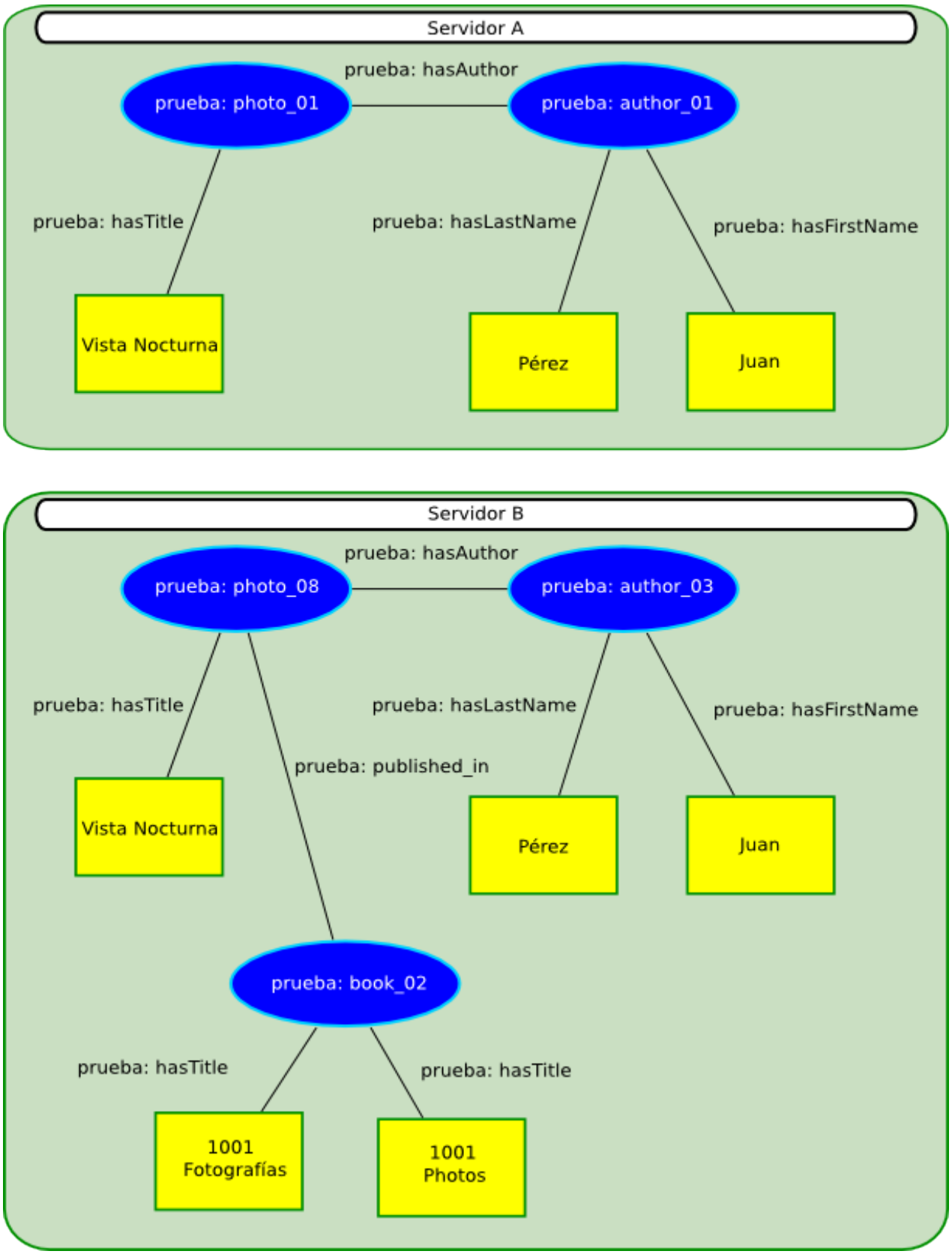
Servidor A
prueba: hasAuthor
prueba: photo_01
prueba: author_01
prueba: hasTitle
prueba: hasLastName
prueba: hasFirstName
Vista Nocturna
Pérez
Juan
Servidor B
prueba: hasAuthor
prueba: photo_08
prueba: author_03
prueba: hasTitle
prueba: hasLastName
prueba: hasFirstName
prueba: published_in
Vista Nocturna
Pérez
Juan
prueba: book_02
prueba: hasTitle
prueba: hasTitle
1001
Fotografías
1001
Photos

Figura 3

1. En base a un conjunto ya homogéneo de servidores, cada vez que otro servidor se integre a la red se examinan sus datos, existendos opciones más:
   - Mantenemos un índice o índices distribuidos de *uris* equivalentes, para un rápido acceso a los *uris* que ya están catalogados como equivalentes de acuerdo a resultados anteriores y añadimos los nuevos *uris* equivalentes al índice.
   - Se verifica qué datos son semejantes a otros ya existentes y se hacen homogéneos sus *uris*, para que así todos los participantes de la red puedan tener resultados de búsqueda unificados. La creación de nodos en una grafo WS existente es necesario para añadir los nodos de los *uris* renombrados.

   Sería importante añadir herramientas en la interfaz de usuario para que la persona encargada del catálogo pueda decidir cuáles nodos en otros pescadores son o no equivalentes a los que hay en la red.
2. Mientras se hace el procedimiento de búsqueda se examinan las rutas resultantes, y al momento de la unificación se analizan sus propiedades y literales, si estos son semejantes, entonces, se puede manejar que ambos resultados son iguales y se renombra los *uris,* eligiendo alguno de los dos *uris* comparados y se hacen los cambios si es que existe más datos con el identificador elegido. Las desventajas de esta opción son que: pueden existir nodos con relaciones equivalentes y sus literales tengan valores diferentes, y estén tratando de cosas semejantes. Por ejemplo en el servidor A tenemos photo_01, con titulo “Vista nocturna” y el fotógrafo sea Juan Pérez Pérez, mientras que en el servidor B tenga la photo_02, con titulo “Valle de noche” y el fotógrafo sea Juan Pérez y sean la misma foto con títulos diferentes y el fotógrafo sin un apellido.

Revisión de protocolos que manejan la red de igual a igual.

Existen actualmente una gran variedad de bibliotecas que implementan las redes P2P, en sus diferentes características como lo mencionamos en la introducción. Una de ellas por ejemplo, es la estructura con la que se construyen. Cada protocolo ofrece diferentes algoritmos para la creación y mantenimiento de la red de acuerdo a la teoría en la que se basa.

De las bibliotecas que existen nos enfocaremos por ahora a dos (JXTA, Chord), las cuales ofrecen cosas adecuadas para un buen funcionamiento.

JXTA

Basándonos en la guía para el programador de JXTA[1], mencionaremos brevemente los aspectos que nos interesan de esta biblioteca.

JXTA es un conjunto de protocolos P2P abiertos y generalizados que permiten a todo dispositivo conectado a la red (sensores, teléfonos celulares, PDAs, laptops, estaciones de trabajo, servidores y supercomputadoras) comunicarse y colaborar mutuamente como elementos iguales de la red. Los protocolos del JXTA son independientes del lenguaje de programación, y múltiples implementaciones, también conocidos como *bindings*, existen para diferentes ambientes, y por lo tanto el uso común de los protocolos significa que ellos son completamente operantes.

JXTA provee un conjunto común de protocolos abiertos con referencias a implementaciones de código abierto para el desarrollo de aplicaciones P2P. Estos protocolos estandarizan la manera en que los elementos de la red:

- Se encuentran los unos con los otros.
- Se auto organizan en grupos.
- Notifican y descubren recurso en la red.
- Se comunican los unos con los otros.
- Se monitorizan los unos con los otros.

Usando la tecnología JXTA, los desarrolladores pueden escribir aplicaciones funcionales para redes con las características:

- Encontrar nodos en la red con descubrimiento dinámico a través de *firewalls* y NATs.
- Fácilmente comparten recursos con cualquiera a través de la red.
- Encuentran el contenido que esté disponible en los sitios de la red.
- Crean un grupo de nodos que proveen un servicio.
- Monitorizan las actividades de los nodos remotamente.
- Comunicación segura con otros nodos en la red.

Como parte de las redes P2P actuales utiliza DHT (Tablas Hash Distribuidas) para la indexación de los recursos a través de la red.

Finalmente el JXTA utiliza los Mensajes JXTA, los cuales la red necesita para comunicar servicios y aplicaciones. Los Mensajes JXTA son la unidad básica del intercambio de datos entre los participantes de la red JXTA. Un Mensaje JXTA es un contenedor para el protocolo de información y contenido. Cada mensaje está compuesto de una serie de elementos del mensaje, y estos elementos del mensaje contienen una porción del mensaje. El propósito y contenido de cada elemento es definido por la implementación del protocolo del mensaje. Estos elementos del mensaje pueden contener cualquier tipo de dato requerido por el mensaje, incluso objetos Java serializados.

Chord[2]

Un problema fundamental que enfrentan las aplicaciones en redes P2P es localizar eficientemente el nodo que almacena un elemento en particular. Por su parte Chord que es un protocolo para redes P2P trata este problema.

Chord provee soporte para una sola operación: dada una llave este la mapea a un nodo. La localización de los datos puede ser fácilmente implementada encima de Chord. A su vez Chord adapta eficientemente como los nodos se unen, abandonan el sistema y puede responder peticiones inclusos si el sistema esta continuamente cambiando. Resultados de análisis teóricos, simulaciones y experimentos muestran que Chord es escalable, con costo de comunicación y mantenimiento del estado por cada nodo escalando logarítmicamente con el numero de nodos en la red Chord.

Una implementación de Chord se encuentra en la dirección http://www.uni-bamberg.de/en/fakultaeten/wiai/faecher/informatik/lspi/bereich/research/software_projects/openchord/, la cual utilizamos para hacer el análisis de su funcionamiento y cómo adaptarlo a nuestra necesidades.
Este provee las siguientes características:

- Guarda cualquier objeto java serializado dentro de un DHT.
- Provee posibilidad de agregar llaves personalizadas dentro del DHT implementando una interfaz del API de OpenCHord

- Facilita la replicación de elementos en la DHT.
- Actualmente provee dos protocolos para la comunicación entre nodos de Chord:
    1. Llamadas locales a métodos: Este protocolo puede servir para crear un DHT dentro del JVM, para propósitos de pruebas y visualización. Una interfaz de línea de comandos es provista para ejecutar una red Chord.
    1. Java Sockets: Este protocolo crea un DHT distribuido sobre diferentes nodos. Para propósitos de pruebas los nodos de Chord utilizan este protocolo.
    1. Protocolos Personalizados: Estos protocolos pueden ser provistos por otros desarrolladores.

Chord a comparación con JXTA es muy básico y no ofrece tantas facilidades de personalizarlo y extenderlo de acuerdo a las necesidades que se requieran como JXTA, y como consecuencia el desarrollo de aplicaciones tomaría un plazo mayor de tiempo. JXTA tiene casi todo para comenzar la implementación de nuestra aplicación sólo sería adecuado ver a mayor detalle la comunicación entre nodos y los servicios que pueden prestar para intercambiar datos. Sin embargo Chord sería una buena alternativa, ya que ofrece ser muy robusto a fallos y asegura encontrar cualquier recurso almacenado en la red.

Bibliografía

[1] **Sun Microsystems, Inc,** JXTA JavaTM Standard Edition v2.5 Programmers Guide, 2007, pag. 7-17

[2] **Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, Hari Balakrishnan** , *Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications, pag. 1.* (http://citeseer.ist.psu.edu/442155.html)

[3] **Andrew Russell Green**, *Metadatos Transformados. Archivos Digitales, La Web Semántica y el nuevo Paradigma de la Catalogación, 2006, pag. 11*. (http://durito.nongnu.org/docs/metadatos_transformados_green.pdf)