SOFTWARE PARA EL MEJORAMIENTO EN LA TRANSMISIÓN DE DATOS ENTRE COMPUTADORES UTILIZANDO REDES P2P

SERGIO ANTONIO PINO GALLARDO
IRENE LIZETH MANOTAS GUTIÉRREZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2009

SOFTWARE PARA EL MEJORAMIENTO EN LA TRANSMISIÓN DE DATOS ENTRE COMPUTADORES UTILIZANDO REDES P2P

SERGIO ANTONIO PINO GALLARDO
IRENE LIZETH MANOTAS GUTIÉRREZ

Trabajo de grado para optar por el título de
Ingeniero de Sistemas

Director
MPE. HENRY ARGUELLO FUENTES
Profesor
Escuela de Ingeniería de Sistemas e informática - UIS

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2009

A Dios quien me ha dado la fortaleza, ha iluminado mi camino y ha sido mi padre, mi amigo y mi hermano.

A mi abuela Beatriz (q.e.p.d.) por haberme enseñado a tener valor para salir adelante sin importar las dificultades. Su gran fortaleza y lucha serán siempre un ejemplo para mi vida y la llevaré siempre en mi corazón.

A mi mamá, quien sacrificó su propia vida por darme la mía, la mujer paciente, noble, comprensiva y amorosa que supo entender mis decisiones, confió en mí y estuvo a mi lado apoyándome y dándome fuerzas para seguir.

A mi papá por ayudarme a emprender esta etapa en mi vida.

A Sergio, por sus ideas, por su inmensa creatividad y por su espíritu luchador que contagia. Gracias por darme todo tu amor, tu ayuda y por compartir conmigo todos tus sueños.

A Vicky, por toda su ayuda incondicional, por ser tan compresiva y apoyarnos en las dificultades, por su inmenso corazón, su gran espíritu luchador y por ser un ejemplo para Sergio y para mí.

A mi tío Manuel, por ser un apoyo incondicional y un ejemplo de perseverancia. Gracias por todos los favores recibidos. A mi tío Alberto por darme su ayuda. Y en general a toda mi familia, la cual quiero con todo mi corazón.

A mis amigas July y Paola, por ser tan espectacularmente especiales y ser un ejemplo de vida. Gracias a las dos por estar siempre ahí para escucharme y compartir los buenos y malos momentos. Las quiero mucho.

A mis compañeros de ingeniería de la UIS, por todos los bellos momentos que disfrutamos a lo largo de nuestra carrera universitaria.

Y a todas la personas que de alguna u otra forma han estado conmigo a lo largo de este camino.

Irene.

A mi madre con todo el amor y apoyo con el que me crió, por sus consejos, ejemplo y fortaleza inconmensurables.

A mi padre por mostrarme cómo pensar diferente.

A mi difunto abuelo por sus sabios consejos, su gran ejemplo y sus espléndidos cuentos.

A mi abuela por todo su amor y apoyo.

A Irene por darme fuerzas, apoyo incondicional, por la felicidad y amor que hemos evidenciado al enfrentarnos a grandes retos y sobreponernos a ellos sin soltarnos de la mano.

A Henry por su amistad , por corregirnos en el momento preciso y apoyarnos cuando más lo necesitamos.

A mis amigos, en especial a Julián Barragán, Paola Rondón y Cesar Vargas, por brindarme su amistad durante todos éstos años.

A Dios por haber puesto todas estas personas en mi camino, por darme tanto dones y por guiarme para no desperdiciarlos.

A Steve Jobs por mostrarme el valor de las ideas y a Aaron Hillegas por decirme las palabras precisas para encontrar lo que más me apasiona hacer en la vida: programar. "Yo sé que es difícil, pero tú no eres estúpido".

Sergio.

AGRADECIMIENTOS

Los autores expresan su agradecimiento:

A nuestras familias por todo su apoyo y amor brindado a lo largo de la realización de este proyecto, por haber creído en nuestras ideas y habernos guiado durante el transcurso de nuestra carrera universitaria.

Al profesor HENRY ARGUELLO FUENTES, por todas sus enseñanzas, su gran corazón, su apoyo, su ejemplo, por brindarnos su valiosa amistad y por ser nuestro director de proyecto.

A los autores: AARON HILLEGASS, BRENDON J. WILSON, JOSHEP D. GRADECKI y CAY S. HORSTMAN por sus valiosas enseñanzas sobre programación orientada a objetos, redes P2P, programación con JXTA y programación multihilo, plasmadas en sus libros Cocoa Programming, JXTA, Mastering JXTA y JAVA CORE volumen I y II, respectivamente.

A nuestros compañeros de estudio y amigos que nos acompañaron a lo largo de este camino, brindándonos su apoyo, sus pensamientos, sus conocimientos y experiencias vividas.

Al profesor PEDRO TRUJILLO por haber sido el calificador del plan de proyecto y habernos dado sus aportes respecto al tema trabajado.

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ANEXOS

RESUMEN

Titulo: SOFTWARE PARA EL MEJORAMIENTO EN LA TRANSMISIÓN DE DATOS ENTRE COMPUTADORES UTILIZANDO REDES P2P[*].

Autores: Sergio Antonio Pino Gallardo, Irene Lizeth Manotas Gutiérrez[**].

Palabras Claves: Redes P2P, framework JXTA, transferencia de archivos, protocolo de transmisión de archivos, metodología XP, Swing Application Framework, JUnit.

Descripción: Este proyecto de investigación presenta el diseño e implementación de una herramienta software basada en una red P2P la cual permite el intercambio de ficheros entre varios computadores conectados a una red P2P.

La herramienta software desarrollada se presenta como una alternativa de comunicación entre varios computadores conectados en red para compartir ficheros de tipo texto, imágenes, videos, archivos comprimidos, etc. La implementación basada en la red P2P de JXSE (Framework para el desarrollo de aplicaciones P2P de Sun Microsystems) requirió el desarrollo de un protocolo llamado U2U File Sharing Protocol o Protocolo para el intercambio de archivos U2U, para efectuar el control de la transferencia de archivos a través de la red entre varios computadores. Durante el desarrollo del presente proyecto se realizaron una serie de pruebas para verificar el correcto funcionamiento del software y poder observar y analizar el comportamiento de una red con una arquitectura P2P en comparación con otra red con arquitectura Cliente/Servidor. El resultado de las pruebas realizadas se encuentra al final del presente libro en el capítulo cuarto.

Algunos de los logros alcanzados con el presente proyecto fue la publicación del artículo “Arquitectura en capas para el desarrollo de una aplicación basada en redes P2P” en la revista GTI, volumen 7 – número 19 y la ponencia “Software para el mejoramiento en la transmisión de datos entre computadores utilizando redes P2P” realizada en el Seminario Internacional de Informática y Computación - SIIC en la ciudad de Bucaramanga en el año 2008.

Este proyecto fue patrocinado por la vicerrectoría de investigación y extensión de la Universidad Industrial de Santander con el premio adquirido en la convocatoria para apoyar proyectos promisorios 2008.

---

[*] Trabajo de investigación

[**] Facultad de Ingenierías Físico Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: Henry Arguello Fuentes.

ABSTRACT

TITLE: SOFTWARE FOR IMPROVEMENT IN DATA TRANSMISSION NETWORK USING P2P BETWEEN COMPUTERS[*].

Authors: Sergio Antonio Pino Gallardo, Irene Lizeth Manotas Gutiérrez[**].

Keywords: P2P networks, JXTA framework, file transfer, file transfer protocol, methodology XP, Swing Application Framework, JUnit.

Description: This research project presents the design and implementation of a software tool based on a P2P network that allows sharing of files among multiple networked computers.

The software tool was developed as an alternative communication between multiple networked computers to share files of type text, images, videos, archives, etc. Implementing network-based P2P JXSE (Framework for the development of P2P applications from Sun Microsystems) required the development of a protocol called U2U File Sharing Protocol or protocol for exchanging files U2U, for the monitoring of file transfers through the network between several computers. During the development of this project is a series of tests to verify the proper functioning of the software and be able to observe and analyze the behavior of a network with a P2P architecture compared to other system with Client / Server architecture.

Some of the achievements of this project was the publication of the article "layered architecture for the development of an application based on P2P networks" in GTI Magazine, Volume 7 - Number 19 and the paper "Software for the improvement in transmission data between computers using P2P networks" in the International Seminar on Computer - SIIC in the Bucaramanga city in 2008.

This project was sponsored by Vicerrectoría research and extension of the Industrial University of Santander acquired the prize in the call to support promising 2008.

---

[*] Research work

[**] Faculty of Mechanical Engineering Physics. School of Engineering and Computer Science. Director: Henry Arguello Fuentes.

INTRODUCCIÓN

Las redes P2P han sido desde su nacimiento una tecnología importante para la comunicación y la computación, sin embargo hasta hace poco se han podido difundir de manera global los beneficios que ésta tecnología ofrece a los desarrolladores de software para implementar aplicaciones.

Gracias al desarrollo del framework JXTA (Justapuesto) realizado por Sun Microsystems que ofrece un estándar para la creación de aplicaciones P2P, se pueden desarrollar aplicaciones que interactúen entre sí y sirvan como solución a problemas relacionados con las telecomunicaciones y la computación distribuida.

El grupo de investigación en Ingeniería Biomédica – GIIB de la Universidad Industrial de Santander - UIS ha querido realizar con este trabajo una investigación sobre el funcionamiento, estructura y utilización de las redes P2P a través de la creación de una herramienta P2P que sirva como una alternativa para la transmisión de datos entre computadores conectados en red con la ayuda del framework para el desarrollo de aplicaciones P2P JXSE (*protocolos JXTA para la plataforma Java Standard Edition*).

El trabajo presentado en este libro es el primero relacionado con la investigación sobre redes Peer To Peer realizado en el grupo de investigación en Ingeniería Biomédica – GIIB, pero no el primer trabajo de investigación en este tipo de redes en la Universidad Industrial de Santander.[1] Sin embargo, el desarrollo de herramientas basadas en redes P2P es muy escaso en nuestro país o hasta ahora está comenzando a surgir. Por este motivo, el Grupo de Investigación en Ingeniería Biomédica - GIIB de la UIS incursiona y se puede considerar como pionero a nivel nacional en el desarrollo de una herramienta P2P para el intercambio de archivos distribuidos basada sobre JXSE.

La herramienta desarrollada consiste en una aplicación software basada en una red P2P que permite al usuario interactuar a través de su computador como un igual dentro de la red P2P, para poder buscar y compartir archivos de tipo texto e imágenes que sean de interés para los iguales conectados en la red. Adicionalmente, la herramienta permite la configuración del igual dentro de la red P2P lo que le permite brindar diferentes funcionalidades a los otros iguales en la red de acuerdo a la configuración establecida. Otras características de la herramienta desarrollada incluyen la posibilidad de revisar los iguales o nodos que se encuentran actualmente conectados a la red y la posibilidad de observar

[1] Suárez Osorio Maria Victoria, “Peer To Peer Como Alternativa a las limitaciones de la Arquitectura Cliente/Servidor”. Director de Proyecto PHD. Oscar Gualdrón, Universidad Industrial de Santander.

algunos parámetros relacionados con las transferencias realizadas a través de la red P2P por el igual configurado.

Se espera que ésta herramienta sea de gran ayuda para el análisis de futuras soluciones que involucren la tecnología P2P dentro de su arquitectura y que los estudiantes se interesen por la investigación en ésta tecnología que puede ayudar a resolver inconvenientes a nivel regional y nacional en otras áreas de investigación como la computación distribuida, la distribución de contenidos y las telecomunicaciones.

Además, se espera que la implementación de ésta herramienta software se pueda implantar en ambientes donde se necesiten transferir archivos de tipo texto e imágenes de interés para un grupo de usuarios conectados a través de una red intranet, extranet o Internet como una alternativa a otras aplicaciones de transferencia de archivos de arquitecturas centralizadas.

El desarrollo de ésta herramienta software, al igual que los frameworks y APIs utilizados para su construcción, los algoritmos desarrollados y la estrategia de seguridad implementada, serán explicados a lo largo de éste libro. En el primer capítulo se presenta la descripción del proyecto, el objetivo general, los objetivos específicos, justificación, antecedentes y alcances del proyecto. El capitulo 2 muestra los conceptos básicos sobre las redes P2P, características, componentes, clasificación y ventajas de éste tipo de redes. Al mismo tiempo se explica el framework de desarrollo JXTA, características, cómo está conformada la tecnología JXTA, arquitectura, componentes de JXTA y algunos conceptos básicos de JXTA. Para finalizar el segundo capítulo, se muestra el Shell de JXTA y los comandos manejados por éste aplicativo. El tercer Capítulo expone la interfaz gráfica de usuario diseñada para el software U2U 1.0 y los diseños y desarrollos que fueron realizados durante el desarrollo del proyecto. El capítulo 4 presenta las pruebas realizadas a la herramienta software y los capítulos 5, 6 y 7 presentan las conclusiones, las recomendaciones y las referencias respectivamente.

Para obtener información detallada de la herramienta software se pude recurrir a los anexos que presentan los diagramas de clases diseñados, la descripción y detalle de la base de datos y el manual de usuario de la herramienta de software desarrollada U2U.

En los anexos G y H se pueden observar algunos de los avances obtenidos con el desarrollo del presente proyecto.

Capitulo 1

DESCRIPCIÓN DEL PROYECTO

1. INTRODUCCIÓN

Internet ofrece una oportunidad única, especial y decisiva a organizaciones de cualquier tamaño transformando los canales de intercambio de información haciendo de la inmediatez, la eliminación de las barreras espaciales y temporales, y la accesibilidad, sus principales características.

Actualmente la arquitectura de red más utilizada por todos los usuarios de Internet, es la arquitectura cliente-servidor. Esta arquitectura se ha visto mayormente afectada por su dependencia de servidores centrales, con sus problemas de embotellamiento y fallas periódicas. Como alternativa a dichos inconvenientes se ha venido presentando el modelo P2P, Peer To Peer o Entre iguales.

Una red informática P2P se refiere a una red que no tiene clientes ni servidores fijos, sino una serie de nodos que se comportan a la vez como clientes y como servidores de los demás nodos de la red, obteniendo como beneficios un tráfico mejor distribuido y permitiendo a los usuarios organizarse en comunidades para compartir recursos sin complicadas y costosas modificaciones técnicas de red.

Peer To Peer, está conformado por tres tecnologías: El intercambio de archivos y la colaboración, la computación distribuida, y la comunicación instantánea. Cada una de éstas tecnologías permite el desarrollo de aplicaciones que resuelven diferentes necesidades y problemas a la humanidad en general. Hoy en día existen varios desarrollos que permiten conocer los beneficios brindados por las redes P2P como el proyecto SETI@HOME que con ayuda de las redes P2P busca inteligencia etraterrestre[1], o el proyecto OMEMO que brinda un gran disco virtual P2P para almacenamiento de contenidos [2].

Una de las características del uso de programas y redes P2P es que brindan un modo sencillo de reducir costos de comunicación ya que no es necesario establecer servidores, protocolos y dependencias. Hoy en día organizaciones como Ford, Fortune, el Pentágono (STRICOM) o la serie de la NBC Dog Eat Dog, entre otros, ya emplean la tecnología P2P para reducir sus costes y aumentar la flexibilidad de sus equipos [3].

JXTA (Yuxtapuesto) es una plataforma Peer To Peer de código abierto creada por Sun Microsystems en el año 2001 [4]. Esta plataforma está definida como un conjunto de protocolos que permiten a dispositivos conectados a una red

intercambiar mensajes entre si para comunicarse. JXTA es el framework P2P más estable que actualmente existe, además fue diseñado para permitir que un amplio rango de dispositivos (computadoras, teléfonos móviles, PDAs) se comunique de forma descentralizada.

Como JXTA está basado en una serie de protocolos abiertos, puede ser portado a cualquier lenguaje moderno de computación. Actualmente, la implementación de JXTA de Java es la más avanzada y es la base del desarrollo de este proyecto de investigación.

1.2 Objetivo General

Diseñar e implementar un software basado en el Framework JXTA de J2SE, diseñado para el manejo de redes P2P (Peer to Peer), con el objetivo de mejorar el rendimiento en la transferencia de datos entre varios computadores y la seguridad presente en estas transferencias.

1.3 Objetivos Específicos

- Diseñar e implementar algoritmos que permitan lograr un adecuado manejo de la transferencia de archivos entre diferentes computadores, teniendo en cuenta los recursos disponibles en la red P2P.

- Realizar el análisis, diseño e implementación de un prototipo de software basado en el Framework JXTA que permita la transferencia de archivos de tipo texto e imágenes entre varias computadoras utilizando una red peer To peer.

- Implementar un mecanismo de protección ("seguridad") que cumpla los cuatro requisitos básicos de un sistema informático: confidencialidad, integridad, disponibilidad y autenticación.

- Verificar el desempeño del software desarrollado en cuanto al rendimiento en la transferencia de archivos en las redes P2P por medio de los tiempos de respuesta, la velocidad y la disponibilidad de los recursos en comparación con otras transferencias realizadas por medio de una red tradicional.

1.4 Justificación

La congestión del tráfico ha sido un problema desde el nacimiento del paradigma de Cliente/Servidor (C/S). Cuando una gran cantidad de clientes envían peticiones al mismo servidor y al mismo tiempo, el servidor puede verse afectado y fallar. Entre más clientes en un determinado momento estén haciendo peticiones, es mayor la probabilidad que el servidor colapse. Cuando el servidor colapsa, las peticiones de los clientes no pueden ser satisfechas provocando una insatisfacción por parte de los usuarios y una pérdida financiera considerable.

La seguridad de un esquema Cliente/Servidor es otra preocupación importante. Por ejemplo, se deben hacer verificaciones tanto en el cliente como en el servidor para que en los datos que se están transmitiendo de un computador a otro no se presenten violaciones de seguridad.

Con las situaciones mencionadas anteriormente, se puede observar que desde hace tiempo atrás ha existido la necesidad de diseñar e implementar soluciones que mejoren todas estas falencias de disponibilidad y seguridad de la información que se presentan en la actualidad en los esquemas cliente/servidor.

Las redes P2P tienen una serie de ventajas inherentes que las hacen más atractivas (en la mayoría de casos) que las redes tradicionales, como son: menor vulnerabilidad, mayor escalabilidad, menores recursos necesarios para su instalación y mantenimiento, mayor disponibilidad, la posibilidad de efectuar búsquedas (distribuidas, paralelas y asíncronas) por toda la red, la conjunción de herramientas de colaboración diversas (Como Chat, mensajería instantánea, transferencia de archivos e incluso audio y/o videoconferencia) ejecutándose sobre una misma arquitectura, y con mayor facilidad de uso [5].

Con el manejo del Framework JXTA de Sun Microsystems que actualmente es de código abierto, independiente del lenguaje de programación y de los protocolos de transporte, se abre una ventana para el diseño de una red que se maneje con la tecnología peer to peer (P2P) netamente software, que permita conectar dos ordenadores directamente entre sí. De esta forma se pueden brindar varias ventajas a la red: intercambio de archivos y colaboración dentro de los nodos que conforman la red, computación distribuida aprovechando los ciclos de procesador sin usar, comunicación instantánea, entre otras. Además, con este Framework se puede ofrecer a la red una alta seguridad basada en confiabilidad, autenticación, autorización e Integración de los Datos.

Este proyecto pretende presentar una propuesta para la transmisión de archivos entre computadoras que por medio de una red P2P descentralizada, de área

local, permita visualizar uno de los potenciales de la tecnología P2P, la transferencia de archivos, la cual se maneja por medio de nodos conectados a la red que se comportan simultáneamente como clientes y como servidores que comparten archivos y que generan redundancia de los datos, permitiendo a los nodos encontrar la información sin hacer peticiones a ningún servidor centralizado y mejorando el servicio de transmisión de archivos entre diferentes nodos.

Con el diseño de éste software basado en P2P que permita la transmisión de datos entre computadores y que implemente mecanismos de seguridad para estas transferencias, se hace posible establecer un punto de partida para que en un futuro cercano se haga uso de las tecnologías basadas en P2P como soporte en transferencia de archivos, computación distribuida y mensajerita instantánea, para el desarrollo de investigaciones dentro de la Universidad Industrial de Santander.

1.5 Antecedentes

La historia de los sistemas P2P se remonta a finales de la década de 1970. Las redes Usenet[2] (1979) y Fidonet (1984) son consideradas las primeras redes de este tipo que se crearon. En un principio, fueron desarrolladas como redes de intercambio de noticias entre varios campus universitarios de los E.U.A. Desde principios de los años 90 muchas grandes corporaciones internacionales como Intel y Boeing[3] empezaron a usar redes P2P para realizar operaciones con un gran volumen de cálculos, utilizando miles de ordenadores de todo el mundo al mismo tiempo. Este tipo de iniciativas se extendió a proyectos científicos que operaban con gran cantidad de datos, lo que les permitió prescindir de los engorrosos y costosos superordenadores.

En mayo de 1999 el uso de las redes P2P se hizo masivo gracias al proyecto Napster [6] creado por los estudiantes Shawn Fanning y Sean Parker, de la Northeastern University (Boston, E.U.A.). El objetivo de Napster era el intercambio de archivos musicales. La red se valía de un programa cliente que se podía descargar desde cualquier parte del mundo a través de la World Wide Web. Con Napster, cualquier usuario podía conectarse y descargar el último disco de su artista favorito antes, incluso, de que éste saliera al mercado en su país. También ofrecía la posibilidad de encontrar canciones inéditas de artistas

[2] Para mayor información sobre la red usenet dirijase a la página en Internet: http://en.wikipedia.org/wiki/USENET

[3] Boeing Selects Sun's JXTA Technology For U.S. Army Future Combat Systems, Disponible en internet en: http://www.sun.com/smi/Press/sunflash/2005-06/sunflash.20050613.1.xml

poco conocidos, versiones raras o limitadas de determinados LPs o música que no se podía conseguir sin ir específicamente a un determinado país.

Debido al libre intercambio de archivos por esta red, Napster tuvo que enfrentar grandes problemas que denunciaban el desafío que estos programas suponían, tanto a los derechos de producción como a los derechos de autor[4].

Desde el nacimiento de Napster, varios proyectos se han desarrollado en torno a las redes P2P enfocados a la transferencia de archivos, la colaboración, la computación distribuida y la comunicación instantánea. Solo por ver algunos ejemplos de desarrollos hechos con tecnología P2P tenemos: SubEthaEdit, aplicación que se basa en la colaboración de un grupo en la edición de documentos en red local. Este aplicativo fomenta la colaboración de varias personas entre si, al mismo tiempo y en el mismo documento, editado en tiempo real y en red [7]. En aplicaciones para transferencia de archivos hay varios desarrollos que actualmente son muy usados por los usuarios de Internet para descargar archivos de música y video. Un ejemplo de estas aplicaciones es BitTorrent, que es un protocolo diseñado para el intercambio de archivos entre iguales (peer to peer o P2P) y un programa cliente creados por el programador estadounidense Bram Cohen y que se estrenó en la Codecon 2002. El programa está escrito en el lenguaje Python y actualmente se distribuye bajo la licencia MIT [8]. Ejemplo de desarrollo en computación distribuida con P2P es el programa SETI@home[5] para la búsqueda de inteligencia extraterrestre en el que todos los usuarios que descargan un salvapantallas colaboran con el proyecto donando una parte de la capacidad de proceso de su máquina. En esta misma línea se estudia el plegamiento de las proteínas y la búsqueda de números primos grandes, entre otras iniciativas. Por último, un ejemplo de desarrollo basado en P2P y comunicación instantánea es la aplicación P2P para la ESPOL (Escuela Superior Politécnica del Litoral, Ecuador) que fue desarrollada por estudiantes de esta entidad usando tecnología JXTA, esta herramienta de colaboración permite mantener un chat con múltiples usuarios de todos los puntos de la institución como son oficinas administrativas, facultades, laboratorios, etc. facilitando las actividades colaborativas entre los miembros de la comunidad politécnica, e incidiendo en un incremento de la calidad educativa e investigativa de la misma [9].

Actualmente Sun Microsystems ha liberado a la comunidad Open Source, junto con el Framework JXTA para el desarrollo de aplicaciones P2P con java, una especialización para dispositivos móviles llamada JXME. Este Framework ha

[4] En la actualidad Napster fue comprado por la compañía especializada en la venta de productos electróinicos dentro de los Estados Unidos, Best Buy.

[5] Mayor información del proyecto SETI@HOME disponible en Internet en: http://seti.astroseti.org/setiathome/

sido utilizado en Colombia por el Grupo de Interés en el desarrollo de aplicaciones móviles, W@PColombia, perteneciente al Grupo de Ingeniería Telemática de la Universidad del Cauca, para la realización de un trabajo acerca del desarrollo de aplicaciones P2P para dispositivos móviles con JXME. A través de un prototipo de validación conocido como Punto de Encuentro Virtual Móvil se logró comprobar la viabilidad de este tipo de aplicaciones en la redes de 2.5G y se realizaron aportes importantes al trabajo desarrollado por la comunidad JXTA [10].

Otros trabajos con redes P2P en Colombia incluyen propuestas de métodos de búsqueda avanzado dentro de redes P2P y sistemas de búsqueda de componentes con redes P2P. Ejemplo de estos trabajos son la “propuesta de caché sistemático en un sistema de interrogación P2P” Universidad de los Andes, con el apoyo del laboratorio de Informática de la Universidad de Grenoble de Francia, y el “Sistema para indexación y búsqueda de componentes software, PASTEUR”, realizado también en la Universidad de los Andes [11].

1.6 Alcances Del Proyecto

El presente proyecto pretende realizar la apropiación tecnológica de las redes Peer To Peer por medio de una aplicación basada en ésta tecnología de comunicaciones.

Se pretende realizar la traducción de inglés a español de textos especializados en el framework JXTA como lo son la guía de programación de la versión 2.5 de JXSE y el libro JXTA de Brendon Wilson.

La herramienta debe realizar la creación y configuración de un igual dentro de una red Peer To Peer y permitirle al igual comunicarse con otros iguales que se encuentren conectados a la red P2P.

La herramienta debe permitir compartir archivos de tipo texto e imágenes dentro de la red Peer To Peer.

Por medio del desarrollo de ésta herramienta se pretende impulsar el desarrollo de aplicaciones que implementen las tecnologías de las redes Peer To Peer que permitan aprovechar las características ofrecidas por ésta tecnología en ambientes educativos y empresariales.

Se espera que la herramienta sea de gran ayuda para futuras investigaciones en el área de ciencias de la computación, las telecomunicaciones, la computación distribuida y la ingeniería del software.

Capitulo 2

MARCO TEÓRICO

2.1 REDES PEER TO PEER (P2P)

2.1.1 Introducción

Una red P2P se caracteriza por no tener clientes ni servidores fijos, sino una serie de nodos que se comportan simultáneamente como clientes y como servidores de los demás nodos de la red. Las redes de computadores P2P son redes que aprovechan, administran y optimizan el uso de banda ancha que acumulan de los demás usuarios en una red por medio de la conectividad entre los mismos usuarios participantes de la red, obteniendo como resultado, mucho más rendimiento en las conexiones y transferencias que con algunos métodos centralizados convencionales donde una cantidad relativamente pequeña de servidores provee el total de banda ancha y recursos compartidos para un servicio o aplicación (*ver* **Figura 1.**).

"En general P2P describe un entorno en donde computadores (*hosts*) conectados entre si en un entorno distribuido no utiliza un punto de control centralizado para enrutar o conectar trafico de datos" [6].

P2P es una tecnología que permite a cualquier dispositivo de red proveer servicios a otro dispositivo, dando así acceso a cualquier tipo de recurso que tiene a su disposición, ya sean documentos, capacidad de almacenamiento, poder de computación, o incluso su propio operador humano, evidenciando así la no dependencia de un servidor centralizado para facilitar el acceso a los servicios.

[6] Bill Yeager, Proyect JXTA Chief Technology Officer

***Figura 1**. Evolución de la computación*

Fuente: pdf "P2P: JXTA en acción, JAVA EXPO 04"

2.1.2 Características

Las redes P2P poseen seis características importantes (*ver* **Tabla 1**):

- Escalabilidad: Cuantos más nodos estén conectados a una red P2P mejor será su funcionamiento. Así, cuando los nodos llegan y comparten sus propios recursos, los recursos totales del sistema aumentan, es decir, existe una relación directamente proporcional entre número de nodos y los recursos de la red.

- Robustez: La naturaleza distribuida de las redes P2P también incrementa la robustez en caso de haber fallos en la réplica excesiva de los datos hacia múltiples destinos, y permitiendo a los iguales encontrar la información sin hacer peticiones a ningún servidor centralizado.

- Descentralización: La arquitectura P2P por definición es descentralizada y todos los nodos son iguales, es decir, no existen nodos con funciones especiales, y por tanto ningún nodo es imprescindible para el funcionamiento de la red.

- Equilibrio de carga entre los nodos: Se comparten o donan recursos a cambio de recursos. Según la aplicación de la red, los recursos pueden ser archivos, ancho de banda, ciclos de proceso o almacenamiento de disco.

- Anonimato: Es deseable que en estas redes quede anónimo el autor de un contenido, el editor, el lector, el servidor que lo alberga y la petición para encontrarlo siempre que así lo necesiten los usuarios.

- Seguridad: Es una de las características deseables de las redes P2P menos implementada. Los objetivos de una aplicación P2P segura serían identificar y evitar los nodos maliciosos, evitar el contenido infectado, evitar el espionaje de las comunicaciones entre nodos, creación de grupos seguros de nodos dentro de la red, protección de los recursos de la red, etc.

Características	Cluster	Grid	Webservices	P2P
Población	Grandes Servidores en granjas	Estaciones gama alta	Grandes Servidores	Cualquier dispositivo o PC
Propiedad	Simple	Múltiple	Simple / múltiple	Múltiple
Búsqueda	NA	Índice centralizado	Servicio búsqueda	Descentralizado
Gestión de usuarios	Centralizada	Descentralizado	NA	Descentralizado
Gestión de recursos	Centralizada	Distribuida	Distribuida	Distribuida
Organización de trabajo	Centralizada	Descentralizado	Descentralizado	Descentralizado
Interoperabilidad	¿Propietaria?	No estándar	Estándar	No estándar
Escalabilidad	100s	¿1000?	¿1000?	¿Millones?
Capacidad	Garantizada	Alta, pero no garantizada	Garantizada	Varia
Prestaciones	Media	Alta	Alta	Muy alta

Tabla 1. *Características de las redes P2P en comparación con otras arquitecturas de red*

Fuente: Documento pdf "P2P: JXTA en acción, JAVA EXPO 04"

2.1.3 Componentes de una red P2P

A continuación se describen los principales componentes que conforman una red P2P.

2.1.3.1 Iguales

En una red P2P las entidades denominadas iguales (*peers*), son los mismos nodos de la red y representan la unidad fundamental de procesamiento de cualquier solución P2P. “Un igual se define como cualquier entidad capaz de ejecutar algún trabajo útil y comunicar los resultados de ese trabajo a otra

entidad sobre una red, directa o indirectamente"[7]. Es decir, se puede tener como igual un dispositivo o una aplicación dentro de una red P2P, o una persona dentro de una red social.

2.1.3.2 Grupos de Iguales

En una red P2P, el concepto de grupo de iguales permite subdividir el espacio de la red. Al crear un grupo de iguales se crea un espacio en la red sobre el cual se proveen servicios a los iguales miembro, servicios que no son accesibles por otros iguales en la red P2P.

Los objetivos de un grupo de iguales se pueden generalizar en tres objetivos principales:

- Ofrecer una aplicación en la que los iguales puedan colaborar como un grupo: Un grupo de iguales es formado para intercambiar servicios que los miembros no quieren tener disponibles para la red P2P entera, debido por ejemplo a la naturaleza privada de los datos.
- Implementar requerimientos de seguridad a los iguales involucrados: Un grupo de iguales puede implementar mecanismos de autenticación para restringir el acceso a solo los iguales que puedan unirse al grupo y utilizar los servicios ofrecidos por este.
- Brindar información de estado de los miembros del grupo: Los miembros de un grupo de iguales podrían monitorizar otros miembros del mismo grupo, con el fin de mantener un nivel mínimo de servicios para la(s) aplicación(es) del grupo de iguales.

2.1.3.3 Servicios de red

Los servicios proveen a los iguales la posibilidad de ejecutar "trabajo útil" como una transferencia de archivos, monitoreo de dispositivos, cálculos, etc. en un igual remoto.

Los servicios se dividen en dos categorías:

- **Servicios de igual:** Son funcionalidades ofrecidas por un igual en la red a otros iguales, estas estarán disponibles solo cuando el igual esté conectado a la red.

[7] BRENDON, J. Wilson., "JXTA", New Riders. 2002. Págs. 3-37.

- **Servicios de Grupo de Iguales:** funcionalidades ofrecidas por un grupo de iguales a los miembros del grupo. Estas funcionalidades pueden ser proporcionados por varios miembros del grupo para proporcionar acceso redundante al servicio.

2.1.4 Clasificación

De la idea principal nacen diversas implementaciones de P2P que se asumen como la clasificación de este tipo de redes: Clasificación por arquitectura y clasificación por estructura, entre otras [12].

2.1.4.1 Clasificación por arquitectura

- **Redes P2P centralizadas**

Este tipo de red P2P se basa en una arquitectura monolítica donde todas las transacciones se hacen a través de un único servidor que sirve de punto de enlace entre dos nodos, y que a la vez almacena y distribuye los nodos donde se almacenan los contenidos (*ver* **Figura 2**.). Poseen una administración muy dinámica y una disposición más permanente de contenido, sin embargo, está muy limitada en la privacidad de los usuarios y en la falta de escalabilidad de un sólo servidor, además de ofrecer problemas en puntos únicos de fallo y enormes costos en el mantenimiento así como el consumo de ancho de banda.

Una red de este tipo reúne las siguientes características:

- Se rige bajo un único servidor que sirve como punto de enlace entre nodos y como servidor de acceso al contenido, el cual distribuye a petición de los nodos.
- Todas las comunicaciones (como las peticiones y encaminamientos entre nodos) dependen exclusivamente de la existencia del servidor.

Algunos ejemplos de este tipo de redes son Napster y Audiogalaxy.

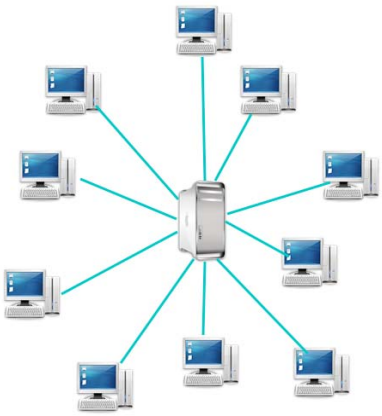


Figura 2. *Red P2P Centralizada*

Fuente: Autores del proyecto. Basada en información de Wikipedia: Peer To Peer.

- **Redes P2P "puras" o totalmente descentralizadas**

Las redes P2P de este tipo no requieren de un gestionamiento central de ningún tipo, lo que permite una reducción de la necesidad de usar un servidor central, por lo que se opta por los mismos usuarios como nodos de esas conexiones y también como almacenistas de esa información.Todas las comunicaciones son directamente de usuario a usuario con ayuda de un nodo quien permite enlazar esas comunicaciones. (*Ver* **Figura 3.**)

Las redes de este tipo tienen las siguientes características:

- Los nodos actúan como cliente y servidor.
- No existe un servidor central que maneje las conexiones de red.
- No hay un enrutador central que sirva como nodo y administre direcciones.

Ejemplos de una red P2P pura son Gnutella, AresGalaxy y Freenet.

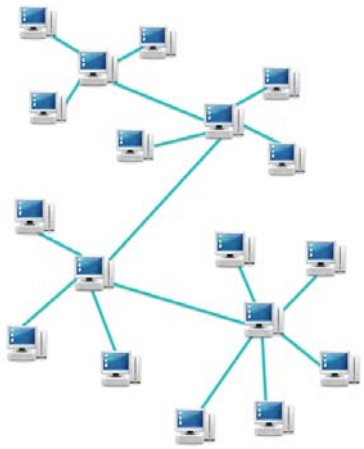


Figura 3. *Red P2P Descentralizada*

Fuente: Autores del proyecto. Basada en información de Wikipedia: Peer To Peer.

- **Redes P2P híbridas, semi-centralizadas o mixtas:**

En este tipo de red, se puede observar la interacción entre un servidor central que sirve como hub[8] y administra los recursos de banda ancha, enrutamientos y comunicación entre nodos pero sin saber la identidad de cada nodo y sin almacenar información alguna, por lo que el servidor no comparte archivos de ningún tipo a ningún nodo. Puede incorporar más de un servidor que gestione los recursos compartidos, pero también en caso de que el o los servidores que gestionan fallen, el grupo de nodos sigue en contacto a través de una conexión directa entre ellos mismos con lo que es posible seguir compartiendo y descargando más información en ausencia de los servidores. (*Ver* **Figura 4**.)

Este tipo de P2P tiene las siguientes características:

- Posee un servidor central que guarda información en espera y responde a peticiones para esa información.
- Los nodos son responsables de hospedar la información (debido a que el servidor central no almacena la información), que permite al servidor central reconocer los recursos que se desean compartir, y para poder descargar esos recursos compartidos a los iguales que lo solicitan.
- Las terminales de enrutamiento son direcciones usadas por el servidor, que son administradas por un sistema de índices para obtener una dirección absoluta.

[8] Hub: Equipo de redes que permite conectar entre sí otros equipos y retransmite los paquetes que recibe desde cualquiera de ellos a todos los demás.

Algunos ejemplos de una red P2P híbrida son Bittorrent, eDonkey2000 y Direct Connect.

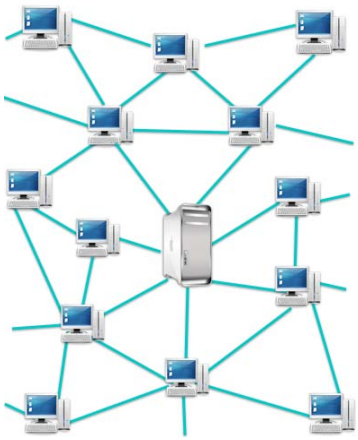


Figura 4. *Red P2P Híbrida*

Fuente: Autores del proyecto. Basada en información de Wikipedia: Peer To Peer.

2.1.4.2 Clasificación por estructura

La red de sobrecapa de P2P consiste en todos los iguales que participan como nodos de red, estos nodos reciben comúnmente el nombre "*Edge Peers*" o "*Iguales Borde*". Hay enlaces entre dos nodos cualesquiera que se conozcan: es decir si un igual participante conoce la localización de otro igual en la red del P2P, entonces hay un borde dirigido del nodo anterior al último nodo en la red de sobrecapa. En base a cómo los nodos en la red de sobrecapa se enlazan el uno al otro, podemos clasificar las redes del P2P como no estructuradas o estructuradas.

- **Redes P2P sin estructura**

Una red P2P no estructurada se forma cuando los enlaces de la sobrecapa se establecen arbitrariamente. Tales redes pueden ser construidas tan fácilmente como un igual que desea unirse a la red puede copiar enlaces existentes de otro nodo y después formar sus propios enlaces en un cierto plazo.

En una red P2P no estructurada, si un igual desea encontrar un pedazo deseado de datos en la red, la petición tiene que recorrer toda la red para encontrar tantos iguales como sea posible, para conseguir a alguien que comparta los datos.

Un contenido popular es muy probable que esté disponible en varios iguales y cualquier igual que busca ese contenido popular, es muy probable que encontrase lo mismo, sin embargo si un igual está buscando datos específicos o no tan populares compartidos por solamente algunos otros iguales, es altamente probable que la búsqueda no sea acertada. Puesto que no hay correlación entre un igual y el contenido compartido por él, no hay garantía que la petición encontrará al igual que tiene los datos deseados.

- **Redes P2P estructuradas**

Las redes P2P estructuradas superan las limitaciones de redes no estructuradas manteniendo una tabla de hash distribuida (DHT) y permitiendo que cada igual sea responsable de una parte específica del contenido en la red. Estas redes utilizan funciones de hash distribuido y asignan valores a cada contenido y a cada igual en la red. Después siguen un protocolo global en la determinación de qué igual es responsable de qué contenido. De esta manera, siempre que un igual desee buscar ciertos datos, utiliza el protocolo global para determinar el o los iguales responsables de los datos y después dirige la búsqueda hacia el o los iguales responsables.

2.1.5 Ventajas de las redes P2P

La principal ventaja de las redes P2P es la distribución de la responsabilidad de la prestación de servicios entre todos los iguales en la red, lo que elimina las interrupciones de los servicios debido a un único punto de error y ofrece una solución más escalable para ofrecer estos servicios.

P2P permite la comunicación a través de una variedad de rutas de red, reduciendo así la congestión de la red.

Todos los clientes proporcionan recursos, incluyendo ancho de banda, espacio de almacenamiento, y poder de computo. Por lo tanto, al agregar más nodos y la demanda en el sistema aumente, la capacidad total del sistema también aumenta. Esto no es cierto en el caso de una arquitectura cliente-servidor con un conjunto fijo de servidores, en el que la incorporación de más clientes podría significar una tasa menor de transferencia de datos para todos los usuarios.

P2P tiene la capacidad de servir los recursos con una alta disponibilidad a un tiempo mucho menor al tiempo que maximiza la utilización de recursos de todos los iguales conectados a la red P2P. Considerando que las soluciones cliente / servidor se basan en adiciones costosas de ancho de banda, equipo, e instalaciones (espacio físico) para mantener una solución robusta, P2P puede ofrecer un nivel similar de la robustez mediante la distribución de las demandas de red y recursos a través de la red P2P.

2.2 JXTA

2.2.1 Introducción

Para convertir a P2P en una tecnología estable para la construcción de soluciones se requiere de Interfaces de Programación de Aplicaciones (APIs) unificadas que faciliten el trabajo a los desarrolladores y un lenguaje común o protocolos que les permita a los participantes (*nodos*) comunicarse y realizar diversas tareas sobre la red P2P.

Teniendo en cuenta estas necesidades, Sun Microsystems creó el Proyecto JXTA con el ánimo de crear un común denominador en el ambiente de las redes P2P. En su interior, JXTA es un conjunto de especificaciones protocolares sobre las cuales se pretende soportar una extensa gama de aplicaciones; de esta forma, quien desee producir una nueva aplicación se ahorra la dificultad de diseñar sus propios protocolos para manejar el núcleo de funciones de una comunicación P2P.

El equipo de desarrolladores de Sun reconoció que las soluciones P2P siempre existirían junto a las actuales soluciones cliente/servidor en lugar de reemplazarlas completamente [13]. El Proyecto JXTA no ha sido diseñado para sustituir las tecnologías y los modelos de computación actuales, sino para ampliarlos con el fin de satisfacer un nuevo conjunto de necesidades.

Los protocolos JXTA están diseñados para ser independientes del lenguaje de programación y de los protocolos de transporte. Estos pueden ser implementados en Java, C/C++, Perl, y en muchos otros lenguajes de programación. Dichos protocolos se pueden basar en TCP/IP, HTTP, Bluetooth, Home PNA u otros protocolos de transporte.

JXTA crea una red virtual que permite a los peers o iguales interactuar entre si, aún cuando algunos de ellos estén detrás de firewalls, NATs o usen distintos transportes de red. Además, cada igual es identificado por un ID único, un URN SHA-1 de 160 bits en la implementación de Java, permitiendo que los iguales puedan cambiar su dirección pero conservar su número de identificación [14].

2.2.2 Características de JXTA

Un principio primario de diseño de JXTA es proveer una plataforma que contenga las funciones básicas de las redes P2P. Las principales características del proyecto JXTA son:

• ***Interoperabilidad*:** La tecnología JXTA está diseñada para habilitar la provisión de servicios P2P al igual para localizar y comunicarse el uno con otro independientemente de direcciones de red y protocolos físicos.

• ***Independiente de la Plataforma*:** La tecnología JXTA está diseñada para ser independiente del lenguaje de programación, protocolos de transporte de red, y plataformas de despliegue.

• ***Presente en cualquier parte (Ubiquity):*** La tecnología JXTA está diseñada para ser accesible por cualquier dispositivo con pulso digital (*digital heartbeat*), no solo PCs o una plataforma de despliegue específica.

Una característica común de los iguales en una red P2P es que ellos frecuentemente existen en el extremo de las redes regulares, este extremo frecuentemente se representa con dispositivos conectados ocasionalmente y que tienen asignadas direcciones no estáticas (*ej. DHCP*). Debido a que están sujetos a conectividad imprevisible con direccionamientos de red potencialmente variables, estos están fuera del alcance estándar del DNS. JXTA equipa a los iguales en el extremo de la red con el aprovisionamiento de un esquema de dirección global único de igual (*globally unique peer addressing scheme*) que es independiente del tradicional nombre de servicio [15].

2.2.3 Tecnología JXTA

JXTA es una plataforma de computación de red abierta diseñada para computación P2P proveyendo los bloques de construcción básica y servicios requeridos para permitir en cualquier momento o lugar conectividad de la aplicación.

El nombre “JXTA” no es un acrónimo. Es una contracción para yuxtapuesto, colocado junto a algo o en posición inmediata a algo. Es un reconocimiento que P2P es yuxtapuesto a cliente-servidor o computación basada en Web, los cuales son modelos tradicionales de computación distribuida hoy en día.

JXTA provee un conjunto común de protocolos abiertos que cuentan con implementaciones de referencia de código abierto para el desarrollo de aplicaciones Peer to Peer. Los protocolos JXTA estandarizan la forma en que sus iguales:

- Se descubren.
- Se auto organizan en grupos de iguales.
- Publican y descubren recursos en la red.
- Se comunican entre sí.
- Se supervisan.

Los protocolos JXTA son diseñados para ser independientes de lenguajes de programación y protocolos de transporte. Estos protocolos son: Peer Discovery Protocol, Peer Resolver Protocol, Rendezvous Protocol, Peer Information Protocol, Pipe Binding Protocol y Endpoint Routing Protocol (*ver* **Figura 5**).

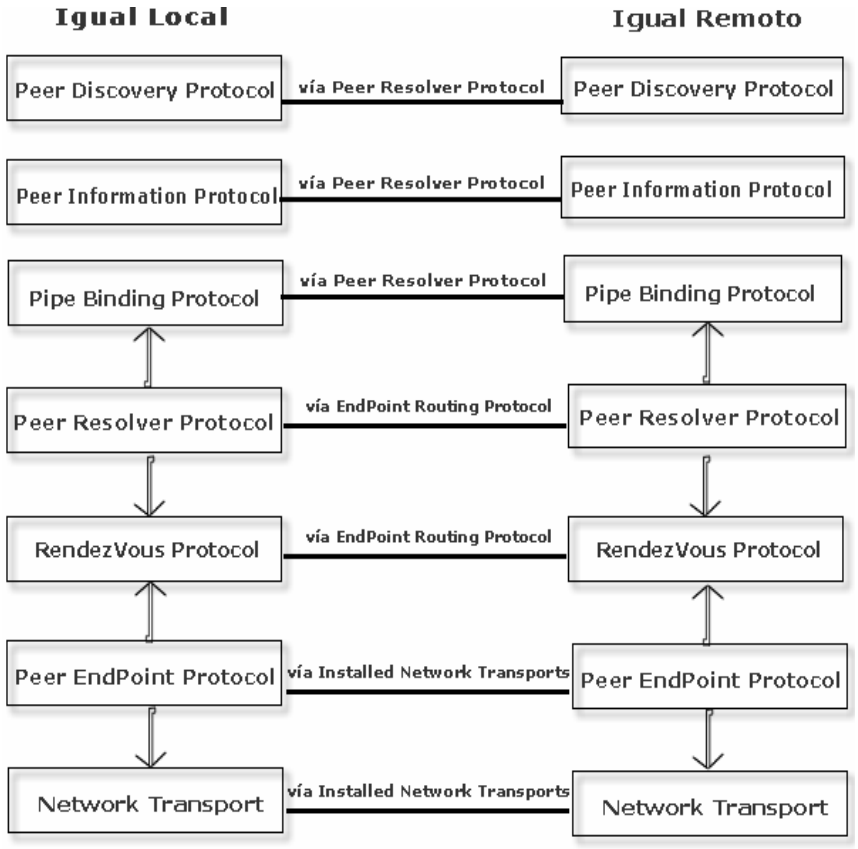


Figura 5. *Conjunto de Protocolos JXTA*

Fuente: Autores del proyecto. Basada en información del libro JXTA de Brendon Wilson.

2.2.4 Arquitectura JXTA

La arquitectura software de JXTA está dividida en tres capas: El corazón de JXTA, la capa de servicios y la capa de aplicaciones, como se muestra en la **Figura 6**.

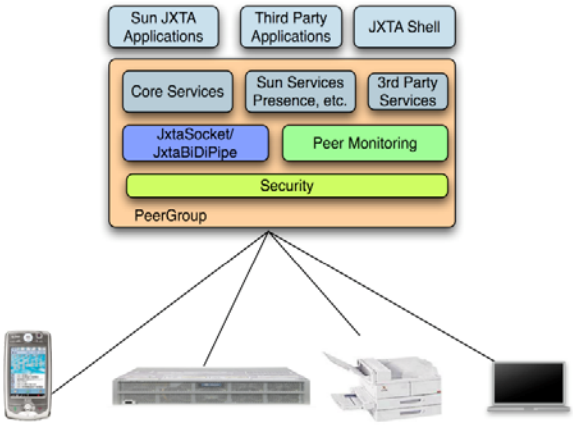


Figura 6. *Arquitectura de software JXTA*

Fuente: Pdf P2P: JXTA en Acción, JAVA EXPO 04.

- **Corazón de JXTA**

El corazón de JXTA encapsula las primitivas mínimas y esenciales que son comunes para interconexión P2P. Esto incluye componentes básicos para permitir mecanismos claves para las aplicaciones P2P, incluyendo descubrimiento, transportes de comunicación (incluyendo firewall, cortafuegos y NAT traversal), la creación de iguales y grupos de iguales, y primitivas asociadas con seguridad.

- **Capa de Servicios**

La capa de servicios incluye los servicios de red que podrían no ser absolutamente necesarios para que una red P2P opere, pero son comunes o deseables en un ambiente P2P.

- **Capa de Aplicaciones**

La capa de aplicaciones incluye implementación de aplicaciones integradas, P2P como mensajería instantánea, distribución de documentos y de recursos, gestión y entrega de contenidos de entretenimiento, sistemas de correo electrónico P2P, sistemas de subasta distribuido, y muchos otros.

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

2.2.5 Componentes JXTA

La red JXTA consiste de una serie de nodos interconectados, o iguales (*peers*). Un igual puede ser cualquier tipo de dispositivo desde un sensor hasta un supercomputador o incluso un proceso virtual. Múltiples iguales pueden correr sobre un único dispositivo físico, y potencialmente, varios dispositivos físicos podrían cooperar para actuar como un solo igual. Los iguales pueden ser conectados por cualquier protocolo de interconexión adecuado incluyendo TCP/IP, http, Bluetooth, GSM, etc.

Cada igual ofrece un conjunto de servicios y recursos que están a disposición de otros iguales. Todos los iguales JXTA implementan un pequeño número de los servicios básicos requeridos y frecuentemente también proporcionan varios servicios estándar adicionales. Cada grupo de iguales incluye como parte de su definición el conjunto de servicios de grupo que cada igual debe ejecutar para participar en el grupo de iguales.

Los recursos de un igual son normalmente contenido estático (no-interactivo) los cuales el igual controla, posee o simplemente tiene una copia. Los recursos pueden incluir archivos, documentos, multimedia, avisos, índices, pero también pueden ser recursos del mundo real tales como switches, sensores e impresoras.

Los iguales JXTA publican sus servicios y recursos usando documentos XML llamados avisos (*advertisement*). Los avisos habilitan a los iguales sobre la red para descubrir recursos y servicios y para determinar conectarse e interactuar con estos servicios.

Para enviar mensajes los iguales JXTA utilizan sockets y pipes. Los sockets de JXTA son conexiones bidireccionales usadas por las aplicaciones para comunicarse confiablemente. Los pipes son un mecanismo de transferencia de mensajes asíncrono y unidireccional usado para la comunicación del servicio. Los mensajes son simples documentos XML cuya cubierta contiene información de ruta, resumen y credencial. Los pipes están obligados a especificar puntos finales (*endpoints*), tales como puerto TCP y una dirección IP asociada.

2.2.6 Conceptos JXTA

2.2.6.1 Iguales

En JXTA un igual es cualquier entidad de red que implemente uno o más de los protocolos de JXTA. Cada igual opera independientemente y asincronamente de todos los otros iguales y es identificado únicamente por un ID de igual.

Los iguales publican una o más direcciones de red para usarlas con los protocolos JXTA. Cada dirección publicada se anuncia como un punto final (*endpoint*) del igual, que identifica la dirección de red. Los puntos finales del igual son usados por los iguales para estabilizar directamente conexiones punto-a-punto entre dos iguales.

Las conexiones de red punto-a-punto directas no siempre están disponibles entre iguales. Iguales intermediarios pueden ser utilizados para enrutar mensajes a los iguales que están separados debido a los límites físicos de la red. Los límites de la red pueden ser límites naturales por ejemplo entre redes de Ethernet y de Bluetooth o creadas artificialmente debido a la configuración de red. Las barreras artificiales pueden incluir NAT, cortafuegos y servidores proxy.

Los iguales JXTA pueden ser categorizados dentro de tres tipos principales [16] (*ver* **Figura 7**):

- ***Minimal-Edge peers*:** Iguales que implementan solo los servicios requeridos del corazón de JXTA y pueden contar con otros iguales para que actúen como su proxy para otros servicios y participar completamente en una red JXTA. Los iguales proxy actúan como un proxy para los servicios que no son del corazón de JXTA. Los iguales *Minimal-Edge* incluyen típicamente sensores y dispositivos de automatización de casas.

- ***Full-Edge peers:*** Iguales que implementan todos los servicios del corazón y estándar de JXTA y pueden participar en todos los protocolos JXTA. Estos iguales conforman la mayoría de iguales en una red JXTA y pueden incluir teléfonos, PC's, servidores, etc.

- ***Super-Peer*:** Iguales que implementan y proveen recursos para soportar el despliegue y operación de una red JXTA. Hay tres funciones principales en un *Super-Peer*. Un igual simple puede implementar una o más de estas funciones:

- ***Función Relay (receive, reinforce, and retransmit*):** Usada para almacenar y reenviar mensajes entre iguales que no tienen conectividad directa debido a cortafuegos o NATs. Solamente los iguales que no están en capacidad de recibir conexiones de otros iguales requieren un igual funcionando como Relay.

- ***Función Rendezvous*:** Mantiene los índices globales de publicidad y ayuda a iguales extremos (o iguales Edge) y proxys con los anuncio de búsquedas. También se ocupa de la radiodifusión de mensajes.

- ***Función Proxy*:** Utilizada por los *Minimal-Edge peers* para tener acceso a todas las funcionalidades de la red JXTA. El igual proxy, traduce y resume las peticiones, responde a las consultas y proporciona soporte para la funcionalidad de los *Minimal-Edge peers.*

Figura 7. *Protocolos JXTA utilizados por los iguales según la categoría*

Fuente: Autores del proyecto. Basada en información del pdf: Project JXTA "An Open P2P Application Platform", Juan Carlos Soto. 2002.

2.2.6.2 Grupos de Iguales

En JXTA, un grupo de iguales es una colección de iguales que se han puesto de acuerdo en un conjunto común de servicios o intereses. Los iguales se auto-organizan en grupos de iguales, cada uno de los cuales se identifica unívocamente por un ID de grupo de iguales.

Cada grupo de iguales establece su propia política de membresía incluyendo abierta (cualquier igual puede afiliarse) hasta la muy segura y protegida (que requiere credenciales para poder ser miembro).

Los iguales pueden pertenecer a más de un grupo de iguales simultáneamente. Por defecto, el primer grupo que se instancia es el *Network Peer Group*. Todos los iguales pertenecientes al *Network Peer Group*, pueden optar por unirse a otros grupos de iguales en cualquier momento (*ver* **Figura 8**).

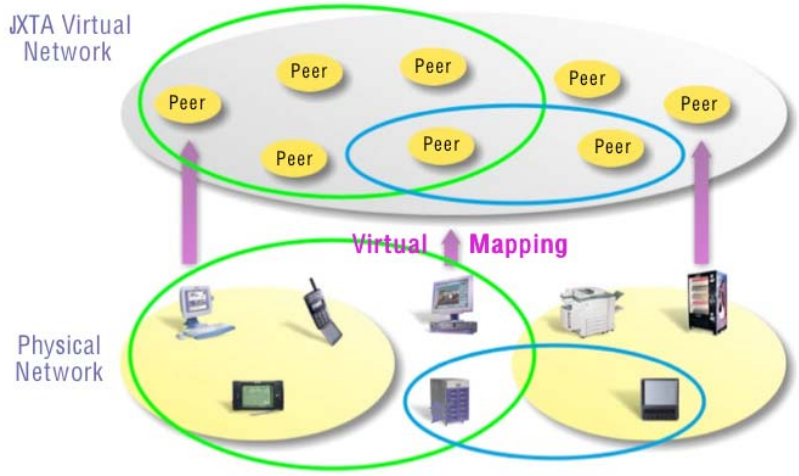


Figura 8*. Grupos de iguales con JXTA*

Fuente: Presentación Project JXTA: An Open P2P Applications Platform Introduction and Update. Disponible en internet en: http://www.cs.rpi.edu/academics/courses/fall02/netprog/notes/jxta/jxta.pdf

Los protocolos JXTA describen como los iguales pueden publicar, descubrir, unirse, y monitorear grupo de iguales, ellos no dictan cuando o por qué los grupos de iguales son creados. Unirse a un grupo es simplemente instanciar todos los servicios de grupo de iguales definidos por el grupo de iguales. Hay varias motivaciones para la creación de grupos de iguales:

- *Para crear un entorno seguro*: Aplicando una política de seguridad específica como un intercambio de texto plano con nombre de usuario y contraseña, o tan sofisticada como criptografía de llave publica creando límites que permiten acceder y publicar contenido protegido.

- *Para crear un entorno de alcance*: Los grupos permiten la creación de un dominio local de especialización. Por ejemplo, los iguales pueden agruparse para implementar una red de documentos compartidos o una red de CPU compartida. Las fronteras del grupo de iguales definen el alcance de la búsqueda en un contenido del grupo.

- *Para crear un ambiente de vigilancia*: Los grupos de iguales permiten a un igual monitorear un conjunto de iguales para un fin especial.

2.2.6.3 Servicios

Los iguales cooperan y se comunican para publicar, descubrir, e invocar *servicios de red*. Los iguales pueden publicar múltiples servicios los cuales, a su vez, son descubiertos a través del Peer Discovery Protocol.

Los protocolos JXTA reconocen dos niveles de servicios de red: Los servicios de iguales y los servicios de grupos de iguales.

Los servicios de iguales son accesibles solamente por el igual que está publicando el servicio y los servicios de grupos de iguales son accesibles por cualquier igual que sea parte del grupo de iguales proporcionando el servicio.

Los servicios pueden ser tanto pre-instalados en un igual o cargados desde la red. Con el fin de ejecutar un servicio, un igual puede tener que localizar una aplicación adecuada para el ambiente de ejecución del igual.

JXTA define un conjunto básico de servicios de grupo de iguales. Estos servicios de grupo de iguales forman la firma de servicio del grupo de iguales, cada uno de los iguales que se unen al grupo tienen que implementar estos servicios.

En JXTA pueden construirse servicios adicionales para proveer funcionalidades específicas. La única restricción para utilizar un servicio entre dos iguales es que ambos iguales deben ser miembros del mismo grupo de iguales.

Los servicios de grupo de iguales básicos que cada uno de los iguales debe implementar con el fin de participar en la red JXTA son:

- ***Endpoint Service:*** El servicio de punto final (*endpoint)* es utilizado para enviar y recibir Mensajes entre iguales. Este servicio implementa el Endpoint Routing Protocol.

- ***Resolver Service:*** El servicio de resolución (*resolver)* es utilizado para enviar solicitudes de consulta genérica a otros iguales. Gracias a éste servicio los iguales pueden definir e intercambiar consultas para encontrar cualquier información necesaria para realizar un trabajo.

Además de los servicios de grupo de iguales básicos exigidos a cada igual, varios servicios estándar adicionales son comúnmente definidos para cada grupo de iguales

Los servicios de grupo de iguales estándar que son encontrados en la mayoría de los grupos de iguales son:

- ***Discovery Service:*** El servicio de descubrimiento es usado por los iguales miembros para buscar recursos de grupo de iguales. Tales como iguales, grupos de iguales, tuberías y servicios.

- ***Membership Service:*** El servicio de membresía es utilizado por los iguales para establecer con seguridad la identidad y la confianza dentro de un grupo de iguales.

- ***Access Service:*** El servicio de Acceso es usado para validar las peticiones hechas por un igual a otro. Sólo aquellas acciones que son limitadas a algunos iguales necesitan ser comprobadas.

- ***Pipe Service:*** El servicio de tubería es utilizado para crear y manejar conexiones de tubería (*pipe connections*) entre los miembros del grupo de iguales.

- ***Monitoring Service:*** El servicio de monitoreo se utiliza para permitir a un igual monitorear a otros miembros del mismo grupo de iguales.

2.2.6.4 Mensajes

Lo servicios y aplicaciones JXTA se comunican usando mensajes JXTA. Los mensajes JXTA son las unidades básicas de intercambio de datos entre iguales. Cada protocolo JXTA está definido como un conjunto de mensajes que intercambian los iguales participantes. Los mensajes son enviados entre los iguales utilizando el servicio de "punto final" (*Endpoint Service*), ó el servicio de tubería (*Pipe Service*), ó JxtaSocket y otros enfoques. La mayoría de las

aplicaciones no necesitan utilizar directamente tuberías unidireccionales (*unidirectional pipe*) o el servicio de punto final de JXTA. En lugar de ello, aplicaciones y servicios utilizan comúnmente los canales de comunicación Socket JXTA y JxtaBiDiPipe para enviar y recibir mensajes.

Los protocolos JXTA definen dos representaciones de mensajes: XML y binaria. Estas representaciones son el formato de dato utilizado para la transmisión del mensaje entre iguales. JXSE utiliza el formato de mensaje binario.

2.2.6.5 Tuberías

Los iguales JXTA utilizan tuberías (*pipes*) para enviar mensajes hacia otros iguales. Las tuberías son un mecanismo de transferencia de mensajes asíncrono, unidireccional y no confiable (con la excepción de tuberías de tipo *unicast secure*) usados para comunicación y transferencia de datos. Las tuberías son canales de comunicación virtual y pueden conectar iguales que no tienen un vínculo físico directo, lo que resulta en una conexión lógica. Las tuberías pueden ser usadas para enviar cualquier tipo de datos incluyendo texto XML y HTML, imágenes, música, código binario, cadenas de datos y objetos Java.

La tuberías punto Final (*endpoints*) son referenciadas para recibir una tubería de entrada *(input pipe*) y enviar una tubería de salida *(output pipe)*. Las tuberías punto final están dinámicamente vinculadas con los iguales punto final (*endpoint*) cuando la tubería se abre. Los iguales punto final corresponden para habilitar una interfaz de red entre iguales, por ejemplo si se tiene un puerto TCP y una dirección IP asociada se pueden utilizar para enviar y recibir mensajes. Las tuberías JXTA pueden tener puntos finales que estén conectados a diferentes iguales en tiempos diferentes, o podrían no estar conectados a todos. Toda la resolución y comunicación de tuberías se realiza dentro del alcance de un grupo de iguales. Es decir, la salida y la entrada de las tuberías deberán pertenecer al mismo grupo de iguales.

Las tuberías ofrecen dos modos de comunicación: punto a punto y propagación (*Ver* **Figura 9**). JXSE provee también una tubería unicast segura, una variante segura de la tubería punto a punto.

- ***Tuberías Punto a Punto:*** Una tubería punto a punto conecta exactamente dos tuberías punto final en conjunto, una tubería de entrada en un igual recibe mensajes enviados desde la tubería de salida de otro igual.

- ***Tuberías Propagadas:*** Una tubería propagada conecta una tubería de salida a varias tuberías de entrada. Los mensajes fluyen desde la tubería de salida, la fuente de propagación, hacia la tubería de entrada.

- ***Tuberías Unicast Seguras:*** Una tubería unicast segura es un tipo de tubería punto a punto que provee un canal de comunicación seguro y fiable.

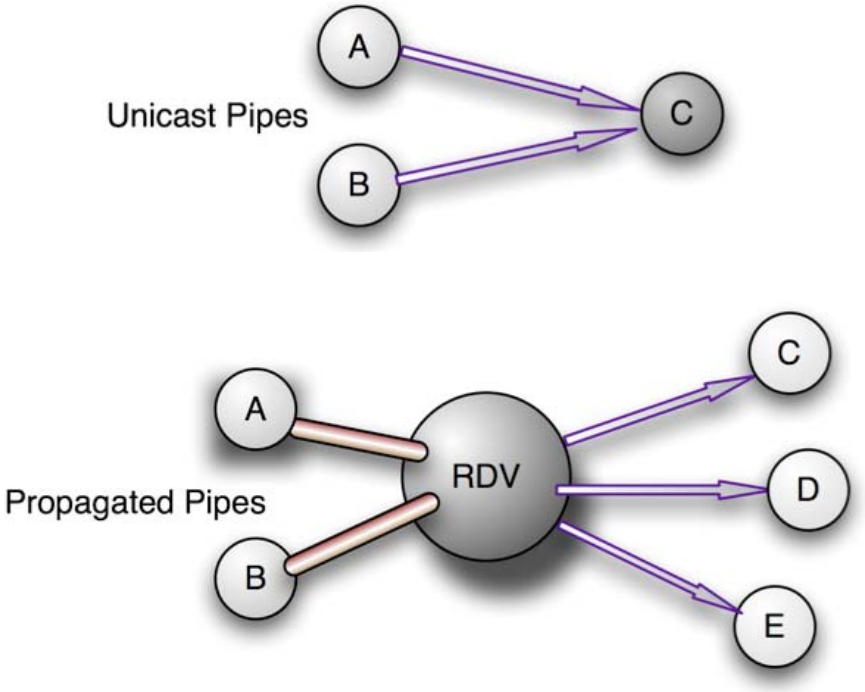


Figura 9. *Tuberías punto a punto y tuberías propagadas*

Fuente: Guía de Programación de JXTA v. 2.5

2.2.6.5.1. Canales de comunicación bidireccionales fiables JxtaSocket y JxtaBiDipipe

Las tuberías básicas de JXTA proveen un canal de comunicación unidireccional poco fiable. Con el fin de hacer más útil las tuberías para servicios y aplicaciones, es necesario implementar canales de comunicación bidireccionales fiables por encima de las tuberías primitivas. JXSE provee funcionalidades para satisfacer el nivel de calidad de los servicios requeridos por la mayoría de las aplicaciones por medio de los sockets JXTA y los Pipes Bidireccionales JXTA (*Ver* **Figura 10**), los cuales proporcionan:

- Fiabilidad
- Aseguran la secuenciación del mensaje
- Garantizan la entrega
- Exponen mensajes y flujo de interfases
- Implementan seguridad

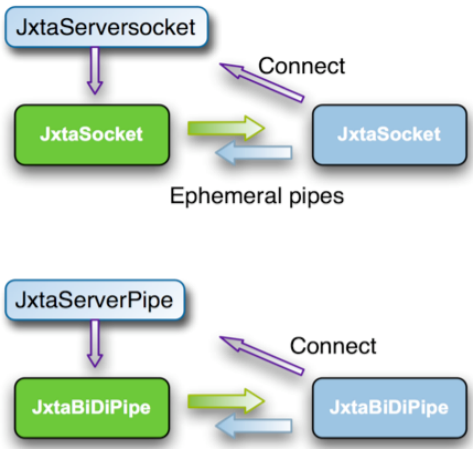


Figura 10. *Comunicaciones Bidireccionales fiables.*
Arriba: Comunicación entre Sockets JXTA
Abajo: Comunicación entre Pipes JXTA

Fuente: Pdf P2P: JXTA en Acción, JAVA EXPO 04

2.2.6.6 Anuncios

Un anuncio en P2P se define como una representación estructurada de una entidad, servicio o recurso hecho disponible por un igual o un grupo de iguales como una parte de una red P2P [9], es decir, los iguales, grupos de iguales, pipes, endpoints, servicios y hasta un contenido, pueden ser representados como anuncios.

Para proporcionar un formato generalizado el equipo de JXTA decidió implementar los anuncios utilizando el formato XML.

Algunos de los principales anuncios manejados dentro de la especificación de JXTA son los siguientes:

- Peer Advertisement
- Peer Group Advertisement
- Pipe Advertisement
- Module Class Advertisement
- Module Spec Advertisement
- Module Impl Advertisement
- Rendezvous Advertisement
- Peer Info Advertisement

[9] BRENDON, J. Wilson., "JXTA", New Riders. 2002. Págs. 3-37.

2.3 Shell JXTA

2.3.1 Introducción

El Shell de JXTA es una aplicación que demuestra los principales conceptos de JXTA JXSE. El Shell de JXTA habilita a los usuarios para interactuar con la plataforma JXTA a través de un intérprete de línea de comandos. Similar al Shell de UNIX®, el Shell de JXTA es útil para acceder y manejar los objetos fundamentales de la plataforma (iguales, grupo, pipes), realizar depuración de problemas de comunicación, verificar el estado de los iguales o de los grupos de iguales, y comunicarse con otros servicios y aplicaciones JXTA.

El Shell de JXTA no hace parte del corazón de JXTA. Ésta es una aplicación diseñada para interactuar con la plataforma, monitorear ciertas actividades de una aplicación de igual y dar resultados que se puedan utilizar para otros propósitos.

Los comandos del Shell JXTA son cargados e iniciados dinámicamente por el framework del Shell JXTA [17] cuando éstos son invocados. Esto permite al Shell JXTA ser fácilmente extendido por nuevos comandos sin tener que cambiar el código fuente del Shell y los comandos existentes.

2.3.2 Comandos del Shell JXTA

El Shell de JXTA proporciona un cierto número de comandos que permiten al usuario explorar funcionalidades básicas del framework JXTA.

Los comandos proporcionados por el Shell de JXTA pueden ser consultados en el ANEXO A.

Capítulo 3

SOFTWARE PARA EL MEJORAMIENTO EN LA TRANSMISIÓN DE DATOS UTILIZANDO REDES P2P, U2U

En este capítulo se mostrarán los desarrollos que se llevaron a cabo para la realización de la herramienta software basada en una red P2P, U2U versión 1.0 y el resultado de las investigaciones realizadas entorno al proyecto.

Los diseños e implementaciones aquí mostrados son obra de los autores del proyecto y en algunos casos adaptaciones tecnológicas de otros desarrollos referentes a la tecnología distribuida de las redes P2P.

En la primera parte de este capítulo se enseñarán el diseño y clasificación de la arquitectura de la red P2P. Después se mostrarán la Interfaz Gráfica de Usuario o GUI de la aplicación software U2U. Luego se explicará el diseño e implementación de la arquitectura Software, la creación del servicio para compartir archivos sobre la red P2P, el diseño e implementación del protocolo para el manejo de la transferencia de archivos y trozos de archivos entre iguales y, la creación e implementación de un comando para el manejo del servicio sobre el Shell U2U. Para terminar se indicará la forma como fue implementado el mecanismo de seguridad para el software U2U 1.0, y las pruebas de transferencia de archivos realizadas con el software desarrollado.

3.1 Arquitectura y Estructura de la Red P2P

La arquitectura de la red P2P diseñada comprende la forma como deben configurarse los computadores dentro de la red P2P para que puedan llevarse a cabo las tareas de compartir, descubrir y descargar archivos entre los nodos de la red.

3.1.1. Arquitectura de la Red P2P

La arquitectura de red diseñada se clasifica dentro de las redes P2P como una red pura o totalmente descentralizada.

Este tipo de red no requiere de un gestionamiento central de ningún tipo, es decir, no hay ningún servidor central que administre los recursos de la red, por

lo que se opta por los mismos usuarios como nodos de esas conexiones y también como almacenistas de la información. Todas las comunicaciones son directamente de usuario a usuario.

En la arquitectura planteada siempre debe existir a lo menos un igual en la red que se comporte como un igual *Super Peer* de tipo *Rendezvous* (*Ver* **Figura 11**).

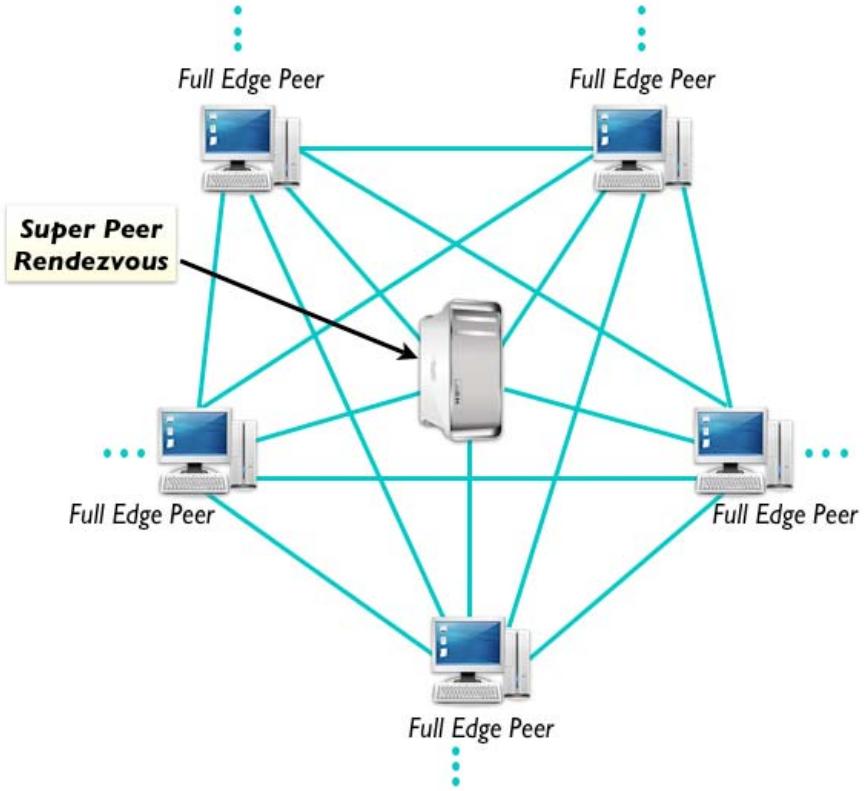


Figura 11. *Primera configuración de la red P2P: Un igual Rendezvous y varios iguales Full Edge*

Fuente: Autores del proyecto.

El igual configurado como *Rendezvous* será el encargado de dar soporte a la red P2P ya que es el principal recurso para que los iguales configurados como *Full Edge* puedan conocer otros iguales que se encuentran en la red. Además, el igual *Rendezvous* servirá como soporte para las búsquedas de archivos dentro de la red P2P.

Si se quieren realizar conexiones con iguales *Full Edge* que se encuentren en el extremo de Internet o detrás de firewalls o NATs, debe configurarse dentro de la red un igual *Relay* que tenga una dirección válida en Internet. (*Ver* **Figura 12**).

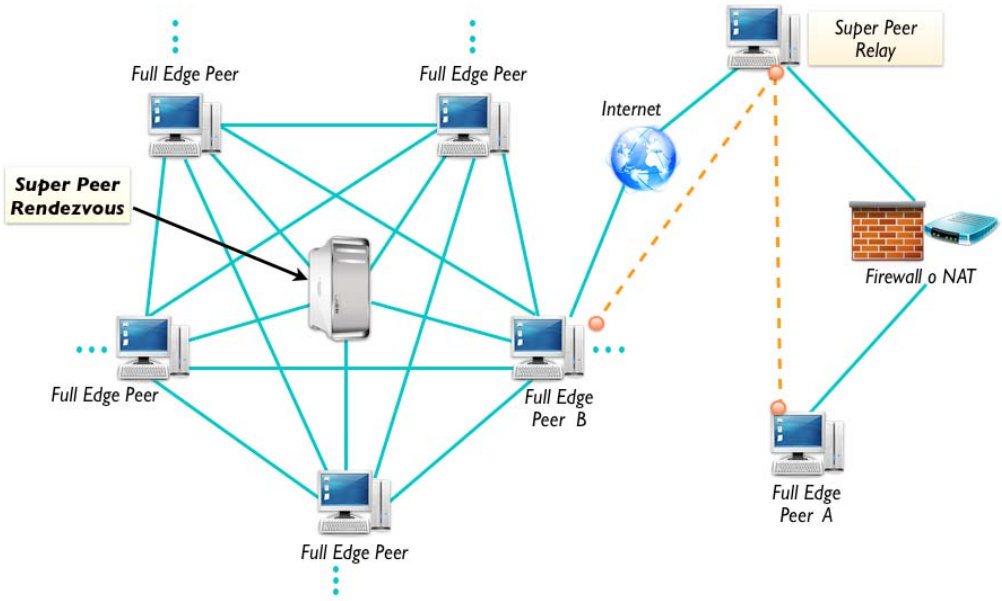


Figura 12. *Segunda configuración de la red: Un igual Rendezvous, un igual Relay y varios iguales Full Edge.*

Fuente: Autores del proyecto.

En la **Figura 12** se puede observar la segunda configuración de la red, la cual incluye un igual *Relay* el cual debe tener una dirección válida en Internet para poder realizar la comunicación con iguales *Full Edge* que se encuentren en el borde de Internet, como el igual *Full Edge A*. En esta figura se puede observar como el igual *Full Edge A* que se encuentra en el borde de Internet detrás de un firewall o de un NAT, puede establecer una comunicación con el igual *Full Edge B* gracias a la existencia del igual *Relay*.

La arquitectura propuesta incluye los elementos mostrados en la **Tabla 2**, según la configuración escogida los elementos de la red varían. Los iguales *Rendezvous* e iguales *Relay* de ambas configuraciones son establecidos como iguales privados, es decir, son iguales configurados para una red privada y no para ser compartidos con otras redes públicas[10].

[10] Para revisar los requisitos de una configuración de iguales *Rendezvous* y *Relay* Públicos puede dirigirse a: http://wiki.java.net/bin/view/Jxta/RunningPublic

Tipo de Red	Configuración	Elementos
LAN	Primera configuración	Un igual *Rendezvous* y n iguales *Full Edge*
MAN o WAN	Segunda configuración	Un igual *Relay*, Un igual *Rendezvous* y n iguales *Full Edge*

Tabla 2. *Elementos de la red P2P según la configuración escogida*.

Las características de los iguales que conforman la red son explicadas en la **Tabla 3.**

Tipo de igual (Peer)	Características
Igual Full Edge	Cualquier computador personal que tenga instalada la maquina virtual de Java versión 5.0 o superior.
Igual Rendezvous	Cualquier computador personal que tenga instalada la maquina virtual de Java versión 5.0 o superior. y tener una dirección IP válida en Internet
Igual Relay	Cualquier computador personal que tenga instalada la maquina virtual de Java versión 5.0 o superior y tener una dirección IP válida en Internet

Tabla 3. *Características de los iguales que conforman la red P2P*[*]

3.1.2 Clasificación de la Red P2P según la estructura

La red P2P diseñada se clasifica como una red estructurada debido a la utilización del framework JXTA.

JXTA maneja un servicio llamado Índice Distribuido de Recursos compartidos o *Shared Resource Distributed Index* (SRDI) [18] que permite a los iguales configurados como *Rendezvous* mantener un índice de los anuncios publicados por los iguales que se encuentran en la red. De esta manera cada vez que un igual del tipo *Full Edge* realiza una búsqueda por un archivo dentro de la red P2P su solicitud se comunica a sus iguales conocidos y al igual *Rendezvous,* el cual busca dentro de su SRDI por iguales que contengan el mismo anuncio solicitado y así poder facilitar la búsqueda del archivo dentro de la red P2P.

[*] Para un mejor funcionamiento de la aplicación debe instalarse la maquina virtual de java Hot Spot suministrada por Sun Microsystems.

3.2 Arquitectura del Software U2U

La arquitectura software es el diseño de más alto nivel de la estructura del sistema en general y fue la base para alcanzar los objetivos de la herramienta software.

A continuación se muestra la arquitectura en capas que se diseñó para el desarrollo de la herramienta software U2U.

3.2.1 Arquitectura en capas para el Software U2U

En el diseño de una arquitectura de aplicación influyen varios factores como son la definición de los módulos principales de la aplicación y sus responsabilidades, la interacción que existe entre estos módulos, el control y flujo de los datos, los protocolos de interacción y comunicación, y la ubicación del hardware. Para el desarrollo de la arquitectura de aplicación aquí expuesta se manejaron las siguientes herramientas: el paradigma orientado a objetos (POO), la metodología de desarrollo rápido eXtreme Programming (XP) y el ingenio de los desarrolladores. El POO se utiliza básicamente como modelo mental, con el cual se analiza la solución en términos de los actores que intervienen en el sistema y como estos interactúan para dar solución al problema. De la metodología de desarrollo, eXtreme Programming, se utilizan los planteamientos más globales, centrándose en la simplicidad y la característica OAOO “Once and only once¨ [19].

Para determinar la arquitectura software del proyecto, U2U, se tuvo en cuenta la segunda meta de diseño del sistema operativo UNIX, la cual tiene como propósito el desarrollo de utilerías simples que realizan tareas específicas de la mejor manera posible y que al combinarlas en una línea de comandos se puede obtener casi cualquier resultado posible [20] (*Ver* **Figura 13**).

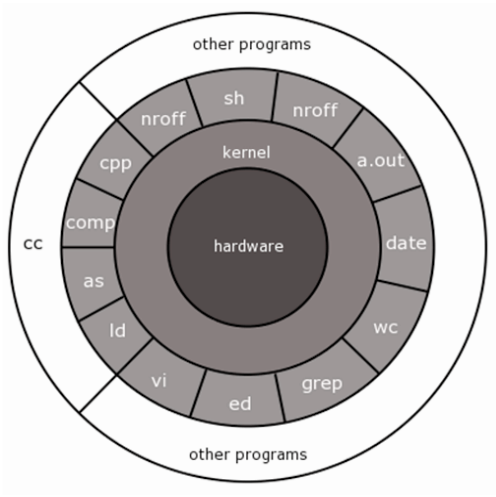


Figura 13. *Arquitectura en capas de UNIX*

Fuente: http://www.monografias.com/trabajos59/sistemasoperativos/sistemasoperativos2.shtml

La arquitectura UNIX propone en su tercera capa el utilitario del sistema llamado SHELL.

El SHELL de UNIX es un intérprete de comandos. Su tarea es tomar los comandos enviados por el usuario, interpretarlos y llamar a las rutinas correspondientes.

Cuando el SHELL lee una línea de comandos, extrae la primera palabra, asume que éste es el nombre de un programa ejecutable, lo busca y lo ejecuta. El SHELL suspende su ejecución hasta que el programa termina, tras lo cual intenta leer la siguiente línea de órdenes.

Basándose en la meta de diseño de UNIX, se planteó el propósito principal de la arquitectura base para éste proyecto: realizar una arquitectura basada en un aplicativo SHELL que de soporte a utilerías simples pero fundamentales en el manejo de una red P2P (*Ver* **Figura 14**).

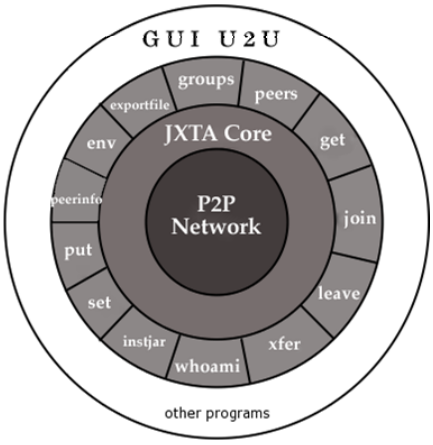


Figura 14. *Arquitectura software creada para el proyecto U2U sobre redes P2P*

Fuente: Autores del proyecto.

La arquitectura diseñada para el proyecto U2U está compuesta de cuatro capas:

Primera Capa: Red Peer To Peer
Segunda Capa: Servicios fundamentales de JXTA y servicio U2UFSS
Tercera Capa: Aplicativo SHELL U2U
Cuarta Capa: Interfaz Gráfica de Usuario U2U

En la primera capa de la arquitectura se encuentra la red P2P configurada según lo visto en el apartado 3.1, es decir, la definición de los iguales *Edge*, iguales *Rendezvous* e iguales *Relay*.

En la segunda capa se encuentran los servicios fundamentales de JXTA y otros servicios implementados sobre la red. Los servicios de JXTA proporcionan la base de comunicación para la red P2P y el servicio U2UFSS – U2U File Sharing Service, es el servicio diseñado e implementado sobre la red de JXTA para permitir a los iguales compartir, transferir y descargar archivos de tipo texto e imágenes sobre la red P2P.

En la tercera capa se encuentra el aplicativo SHELL U2U, que está basado sobre el SHELL de JXTA [17]. Los comandos del SHELL de JXTA se pueden ver en el ANEXO A. La principal diferencia entre el SHELL de JXTA y el SHELL U2U es la implementación del comando *u2ufs* el cual fue creado para llevar a cabo las tareas relacionadas con el servicio para compartir archivos U2UFSS. Los detalles del servicio U2UFSS se explicarán más adelante.

El Shell U2U no se visualiza dentro de un intérprete de comandos como el Shell de JXTA, sino que se ejecuta embebido dentro del aplicativo U2U. La

posibilidad de visualizar el Shell U2U dentro de un intérprete de comandos fue eliminada ya que el usuario no manipulará directamente los comandos del Shell sino la interfaz grafica proporcionada por el aplicativo. Adicionalmente, el Shell U2U no implementa todos los comandos proporcionados por el Shell de JXTA por que no se consideran todos necesarios para el propósito de la aplicación. En la **Tabla 4** se muestran los comandos implementados por el SHELL U2U.

Comando Del Shell	Descripción
man	Ayuda principal del sistema para el Shell de JXTA. Si se escribe el comando man sin parámetros lista todos los comandos del Shell JXTA acompañados de una pequeña explicación.
Cat	Muestra el contenido de una variable de entorno del Shell JXTA.
Chpgrp	Cambia el grupo de iguales por defecto por el especificado.
Env	Muestra en la pantalla todas las variables de entorno definidas en el Shell de JXTA.
Exit	Termina la ejecución del Shell JXTA
exportfile	Exporta una variable del Shell dentro de un archivo externo
flush	Borra un anuncio JXTA
groups	Descubre y lista los grupos de iguales
importfile	Importa un archivo externo dentro de una variable del Shell JXTA
info	Muestra información de un anuncio
instjar	Instala archivos jar conteniendo comandos del Shell adicionales
join	Instancia e ingresa a un grupo de iguales
leave	Abandona un grupo de iguales
login	Autentica con el Membership Service del grupo
mem	Muestra información de la memoria
mkadv	Crea un nuevo anuncio desde un documento almacenado en una variable de entorno del Shell JXTA
newpgrp	Crea un nuevo anuncio de grupo de iguales
newmodulespec	Crea un nuevo anuncio Module Class
newmoduleclass	Crea un nuevo anuncio Module Class
peerconfig	Forza reconfigurar el igual
peerinfo	Obtiene información sobre iguales
peers	Descubre y lista los iguales en la red
pse.certs	Muestra los certificados contenidos en el Membership PSE actual del Igual
pse.createkey	Crea una clave en el almacén de claves del PSE
pse.dumpcred	Elimina un certificado
pse.dupkey	Crea una clave en el almacén de claves del PSE
pse.erase	Borra una clave o un certificado del almacén de claves PSE
pse.importcert	Importa una cadena de un certificado probado
pse.keys	Muestra las claves contenidas en el PSE Membership del actual grupo de iguales
pse.newcsr	Genera un certificado firmado del documento solicitado
pse.signcsr	Firma un certificado solicitando firma
pse.status	Muestra información de estado del PSE Membership del grupo
publish	Publica un anuncio JXTA
put	Coloca datos dentro de un mensaje
rdvcontrol	Controla el comportamiento del servicio Rendezvous
rdvstatus	Muestra información sobre el servicio Rendezvous
relaystatus	Muestra una lista de Relays y clientes conectados a éste igual
remotepublish	Publica remotamente un anuncio JXTA

route	Muestra información de ruta de un igual
search	Descubre anuncios JXTA
set	Modifica una variable de entorno
slepp	Duerme por una cantidad de milisegundos especificada
storehome	Muestra la localización de la carpeta de jxta
talk	Habla con otro igual
transports	Muestra información sobre los transportes de mensajes disponibles en el grupo actual
uninstjar	Desinstala archivos jar previamente instalados con el comando instjar
unset	Remueve una variable de entorno
u2ufs	**Ejecuta el servicio U2UFSS para compartir y descargar archivos entre iguales**
version	Muestra la versión del Shell de ésta instancia del Shell
who	Muestra información de credencial
whoami	Muestra información sobre este igual o el actual grupo de iguales

Tabla 4. *Comandos del SHELL U2U. Incluye el comando u2ufss, creado para ejecutar el servicio para compartir archivos dentro de la red P2P.*

Dentro de la capa del Shell U2U se encuentra la clase Shell que también está implementada en el Shell de JXTA. La clase Shell es la clase encargada de interpretar los comandos ingresados en el intérprete de comandos del Shell. En la **Figura 15** se encuentra la estructura del método ***executeCmd*** el cual fue creado para poder ejecutar comandos desde la interfaz gráfica a través de una instancia del Shell.

```
public boolean executeCmd(String statement)
{
    return (this.processCmd(statement) == ShellApp.appNoError ? true : false );
}
```

Figura 15. Estructura del método ***executeCmd***

Fuente: Autores del proyecto.

En la cuarta capa de la arquitectura se encuentra la Interfaz Gráfica de Usuario – GUI. Esta es la última capa en la arquitectura propuesta y es la capa que se comunica con el SHELL U2U. De esta manera, al ejecutarse un evento en la interfaz gráfica que involucre la ejecución de alguna funcionalidad sobre la red P2P, el evento se comunicará directamente con el SHELL U2U y éste último realizará las operaciones necesarias para llevar a cabo la tarea propuesta.

Algunos de los eventos generados por la GUI ejecutan comandos en el Shell que devuelven respuestas asíncronas. Para manejar las respuestas asíncronas de un evento se implementaron oyentes (*listeners*) [21]. La teoría general de los oyentes establece que existe un objeto al que se le llama disparador que realiza alguna acción sobre otro objeto reactivo. El objeto entonces informa a varios oyentes de que algo ha ocurrido en el disparador y así, cada uno de los oyentes realiza una acción en base al evento que ocurrió. La **Figura 16** muestra

cómo se manejan oyentes entre las capas de la arquitectura de la aplicación U2U.

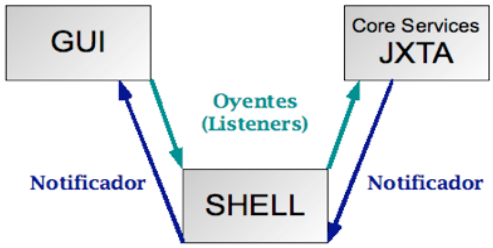


Figura 16. Oyentes creados entre las capas GUI, Shell y JXTA Core

Fuente: Autores del proyecto.

En la implementación de los eventos de la GUI se manejo el Swing Application Framework de Java [22], para realizar un desarrollo más ordenado que permitiera reducir la redundancia de código y además poder manejar un ciclo de vida de la aplicación.

En la **Figura 17** se puede observar como se llevan a cabo las interacciones entre las diferentes capas que conforman la arquitectura software creada para el aplicativo U2U al realizar una búsqueda de un archivo.

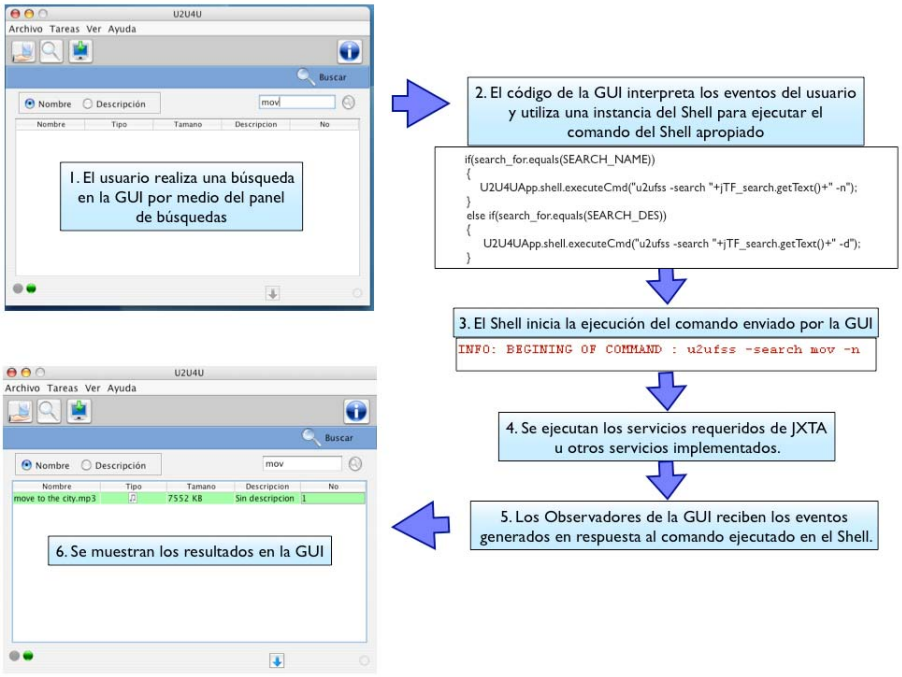


Figura 17. Ejemplo de Interacciones entre las capas de la arquitectura software creada.

Fuente: Autores del proyecto.

3.3 Interfaz Gráfica de Usuario U2U

La interfaz gráfica de usuario (GUI) fue desarrollada en el IDE de programación NetBeans y el lenguaje Java 2 Standard Edition.

La GUI de la herramienta U2U está compuesta por un panel de autenticación y un marco principal donde se alojan los paneles para compartir, buscar y descargar archivos en la red P2P. Adicionalmente existen los paneles de preferencias y de búsqueda de iguales en la red P2P.

En esta sección sólo se mostrarán los componentes principales de la interfaz gráfica de usuario más no los algoritmos y demás implementaciones que se desarrollaron para que la aplicación funcionara correctamente. En el apartado 3.4 se explicarán los diseños e implementaciones que permitieron alcanzar los objetivos propuestos.

La metodología utilizada para el desarrollo del proyecto U2U puede ser vista en el ANEXO B.

Para mayor información de cómo utilizar el aplicativo U2U diríjase al ANEXO C donde se enseña el manual de usuario de U2U.

3.3.1 Panel de Autenticación de Usuario

El panel de autenticación fué diseñado para que el usuario se autentique ingresando el nombre de usuario y la contraseña. El nombre de usuario ingresado por el usuario es también utilizado para identificar el igual en la red P2P.

Los valores por defecto son: Para el nombre de usuario “Peer” y para la contraseña “12345678”. Para cambiar estos valores el usuario puede reconfigurarlos en el panel de Configuración que se explicará más adelante.

La contraseña definida por el usuario es encriptada con el algoritmo simétrico Advanced Encryption Standard AES [23] y es almacenada en un archivo oculto. En la **Figura 18** se puede ver el panel de ingreso para la aplicación U2U.

Figura 18. Panel de ingreso de la aplicación U2U 1.0

Fuente: Autores del proyecto.

3.3.2 Ventana principal

El marco principal del aplicativo U2U es la ventana que se muestra al usuario una vez se ha autenticado en el sistema por medio del panel de autenticación.

La ventana principal de U2U está conformada por la barra de menús, la barra de iconos, la zona de paneles y la barra de estado (*Ver* **Figura 19**).

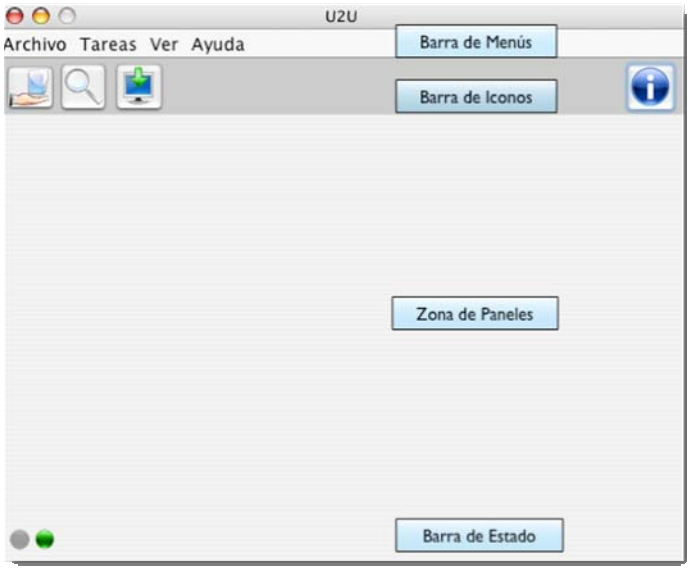


Figura 19. Ventana principal de la aplicación U2U 1.0

Fuente: Autores del proyecto.

3.3.2.1 Panel Compartir Archivos

El panel para compartir archivos permite al usuario escoger un archivo de cualquier tipo y agregarlo a la lista de archivos compartidos para publicarlo y que otros usuarios (iguales) en la red P2P lo puedan descargar. Este panel está localizado dentro del marco principal de U2U y se puede observar en la **Figura 20.**

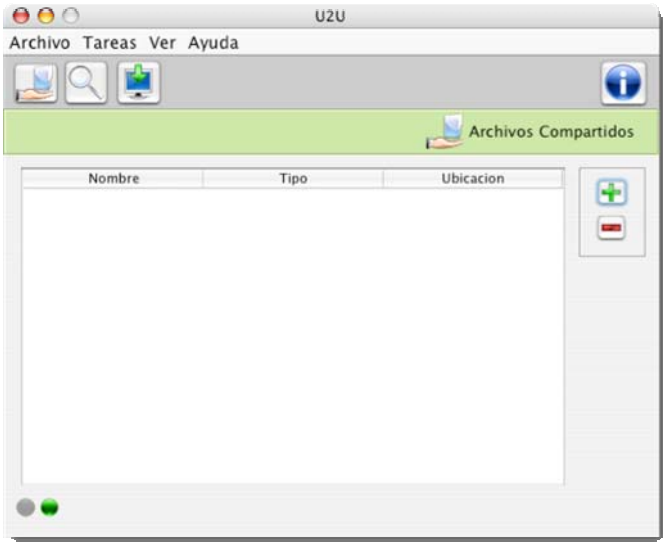


Figura 20. Panel para compartir archivos de la aplicación U2U 1.0

Fuente: Autores del proyecto.

3.3.2.2 Panel Búsquedas

El panel de búsquedas permite al usuario ejecutar una consulta por un archivo sobre la red P2P. El usuario puede buscar por el nombre del archivo o por la descripción con la que haya sido publicado el archivo. Los resultados de una búsqueda se ven reflejados en la tabla de resultados ubicada en la parte posterior del panel.

Una vez que una búsqueda ha arrojado algún resultado sobre la tabla de resultados, el usuario puede elegir descargar o no el archivo encontrado. El panel de búsquedas se muestra en la **Figura 21**.

Figura 21. Panel para buscar archivos de la aplicación U2U 1.0

Fuente: Autores del proyecto.

3.3.2.3 Panel Descargas

El panel de Descargas es el encargado de mostrar las descargas que el usuario ha solicitado a través del panel de búsquedas.

El panel de descargas se muestra en la **Figura 22**.

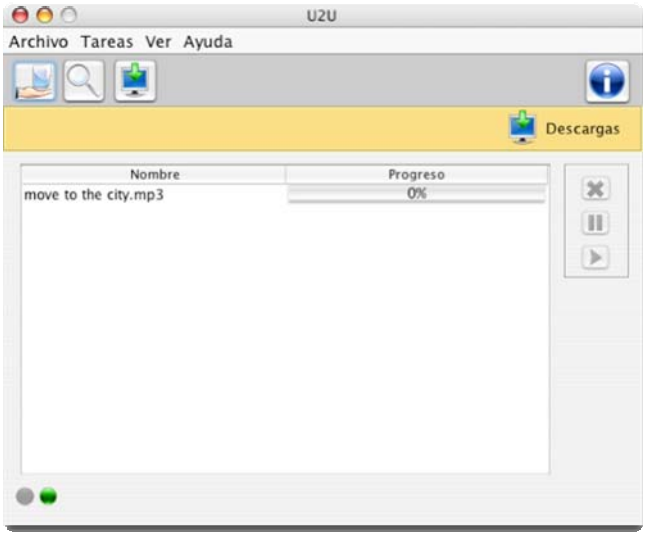


Figura 22. Panel de descargas de la aplicación U2U 1.0

Fuente: Autores del proyecto.

3.3.3 Panel Preferencias

El panel de preferencias permite al usuario configurar la aplicación de acuerdo al rol que el igual desempeñe en la red (*igual Edge, Rendezvous o Relay*), además de permitirle modificar el nombre del igual y la contraseña.

El panel de preferencias se puede ver en la **Figura 23**. Este panel está conformado por tres fichas: General, Red y Actividades.

Figura 23. Panel preferencias de la aplicación U2U 1.0

Fuente: Autores del proyecto.

3.3.4 Panel Buscar Iguales

El panel para buscar iguales, permite al usuario visualizar los iguales de tipo *edge*, *rendezvous* o *relay* que se encuentren conectados en la red P2P.

Este panel está conformado por una tabla de resultados y los botones Buscar y Salir, como se muestra en la **Figura 24**.

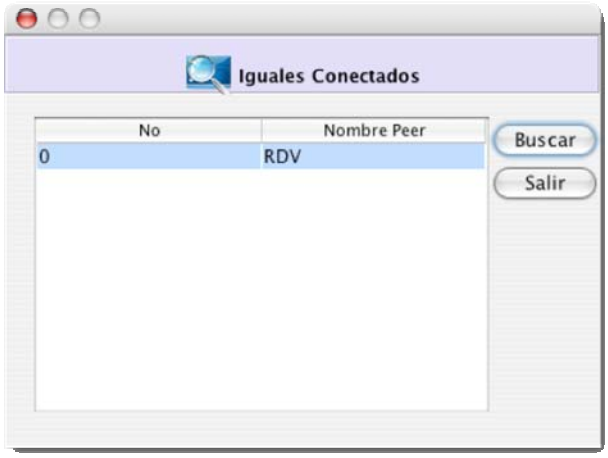


Figura 24. Panel para buscar Iguales conectados, aplicación U2U 1.0.

Fuente: Autores del proyecto.

3.4 Servicio para compartir archivos sobre la red P2P

Para compartir archivos dentro de la red P2P se realizó el diseño y la implementación de un servicio sobre JXTA para permitir compartir, transferir y descargar archivos en la red P2P. El servicio implementado se denominó Servicio para compartir archivos U2U ó U2U File Sharing Service – U2UFSS.

Para diseñar el Servicio se tomó como base los módulos descritos en el artículo *"The Algorithm of Sharing Incomplete Data in Decentralized P2P"* de la IJCSNS [24]. El servicio maneja los dos posibles comportamientos de un igual *Full Edge*: cliente y servidor.

En la **Figura 25** se puede apreciar el diseño general del servicio U2UFSS, el cual está conformado por dos partes: La parte Servidor del igual (*Upload Peer*) y la parte cliente del igual (*Download Peer*).

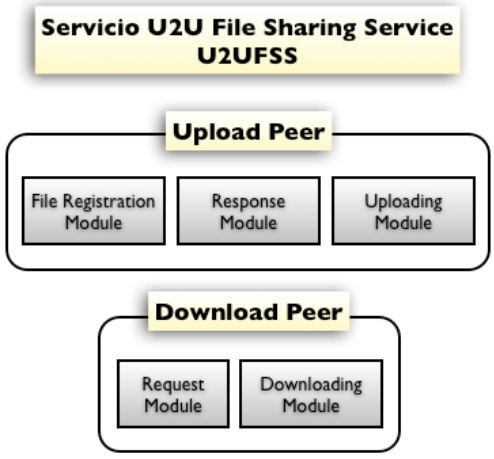


Figura 25. *Diseño del U2UFSS. Servicio para compartir archivos*
Arriba: Upload Peer. Parte Servidor del igual Full Edge
Abajo: Download Peer. Parte cliente del igual Full Edge

Fuente: Autores del proyecto.

Para la implementación de cada uno de los módulos anteriormente mencionados se diseñaron clases que implementan los algoritmos necesarios para la ejecución de las tareas que cada uno de estos módulos debe ejecutar.

El diagrama de clases para la implementación realizada para cada uno de los módulos que componen el servicio U2UFSS se muestra en el ANEXO D.

3.4.1 Igual de Carga (Igual modo Servidor)

El igual registra los archivos que va a compartir en la red P2P, responde a las solicitudes de descarga de archivos de otros iguales y carga archivos, por medio de los módulos de registro de archivos, respuesta y carga respectivamente, cuando se comporta como un servidor dentro de la red P2P.

3.4.1.1 Módulo de Registro de Archivos

El módulo de registro de archivos ó *File Register Module,* es la parte del igual que se creó para que el igual en su faceta de servidor registre , calcule los resúmenes de mensaje o huellas digitales y publique los archivos que quiere compartir a otros iguales en la red P2P.

Este módulo utiliza una base de datos llamada U2UClient para registrar los archivos compartidos por el igual. Antes de registrar un archivo en la base de datos U2UClient, el archivo debe pasar por tres procesos:

1. Cálculo del resumen de mensaje del archivo completo
2. Cálculo del resumen de mensaje de cada uno de los trozos del archivo
3. Publicación del archivo en la red P2P

El cálculo del resumen de mensaje del archivo se realiza por razones de seguridad. Un resumen de mensaje es una huella dactilar de un bloque de datos [25] que permite identificar su contenido, cualquier cambio en tan solo un bit de información en el archivo hará que cambie su resumen de mensaje completamente. Esto permite que, al descargar un archivo, se busquen en la red P2P archivos que sean exactamente iguales al que se desea descargar, ya que estarán identificados con el mismo resumen de mensaje del archivo buscado y no se tendrán en cuenta archivos que contengan el mismo nombre pero que hayan sido alterados, ya que su resumen de mensaje será diferente.

El algoritmo de *hash* o de resumen utilizado para autenticar los archivos compartidos es el algoritmo *Secure Hash Algorithm* o SHA1 [26]. Para la implementación del algoritmo de hash SHA1 se utilizó el API Bouncy Castle [27] y el paquete de seguridad del lenguaje de programación Java [28].

Los archivos compartidos en la red P2P son divididos en trozos de 256KB para permitir descargar los archivos por partes desde varios iguales. Estos trozos del archivo también deben ser autenticados y es por esta razón es que se calculan los resúmenes de mensaje de cada uno de los trozos que conforman el archivo.

Luego de terminar de calcular los resúmenes de mensaje del archivo y sus trozos, el archivo debe ser publicado en la red P2P para que otros iguales lo puedan encontrar y descargar.

Para publicar un archivo éste debe representarse por un anuncio en formato XML [29]. Para representar los archivos de tipo texto, video, imágenes, etc. fue necesaria la implementación de un nuevo tipo de anuncio denominado U2UContentAdvertisementImpl, este anuncio está representado por una clase del mismo nombre la cual es una implementación de la clase abstracta ContentAdvertisementImpl del API Share de JXTA [30].

Luego de crear el anuncio, éste es publicado en la red por medio del Servicio de descubrimiento o *Discovery Service* de JXTA y después el archivo es almacenado en la base de datos U2UClient.

La base de datos U2UClient es una base de datos Derby embebida [31] dentro de la aplicación U2U.

El diseño de la base de datos U2UClient creada para almacenar la información de los archivos compartidos y descargados por el igual a través de la red P2P se muestra en la **Figura 26**.

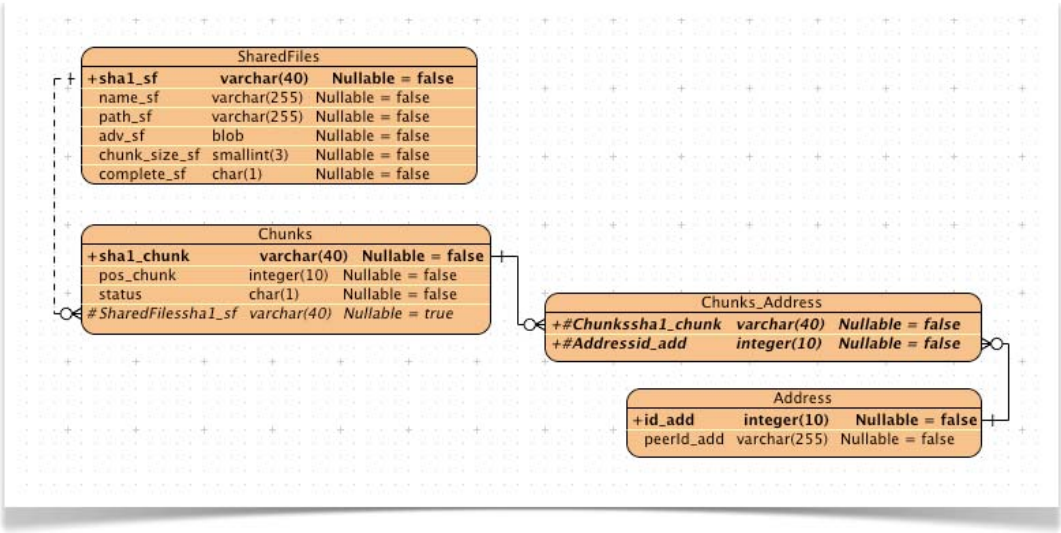


Figura 26. *Diseño de la base de datos creada para registrar los archivos compartidos.*

Fuente: Autores del proyecto.

3.4.1.2 Módulo de Respuesta

Los iguales descubren recursos enviando una solicitud hacia otros iguales *Full Edge* y hacia su igual *Rendezvous* y recibiendo respuestas contenidas en anuncios que describen los recursos disponibles en la red P2P.

El módulo de respuesta es representado por el Discovery Service que es uno de los servicios fundamentales de JXTA. El *Discovery Service* fue utilizado para que un igual responda a los iguales que solicitan información de un archivo.

Cada igual posee una caché local donde se almacenan todos los anuncios publicados y descubiertos por un igual. Cada vez que un anuncio es descubierto por el servicio de descubrimiento *Discovery Service,* el anuncio se añade automáticamente al caché local del igual. Cuando el servicio de descubrimiento busca dentro de su caché local por un anuncio él debe realizar un análisis del anuncio según el tipo y convertirlo en un objeto de la clase asociada al tipo de anuncio. Los anuncios definidos en JXTA se describieron en el apartado 2.2.6.6. Para el manejo de los archivos de tipo imagen y texto que se comparten dentro de la red P2P se manejan los anuncios creados de tipo U2UContentAdvertisementImpl.

Para lograr que el servicio de descubrimiento reconozca éste tipo de anuncio en una consulta de descubrimiento realizada por otro igual, fue necesario incluir un

registro del nuevo tipo de anuncio U2UContentAdvertisementImpl en la clase AdvertisementFactory del framework JXTA. En la **Figura 27** se muestra el fragmento de código utilizado dentro del Servicio U2UFSS para registrar el nuevo anuncio. Al registrar el anuncio de tipo U2UContentImpl, el servicio de descubrimiento puede analizar el XML que representa el anuncio y convertirlo en un objeto de la clase U2UContentAdvertisementImpl correspondiente.

```
//registering the type of advertisement into AdvertisementFactory
static {
    AdvertisementFactory.registerAdvertisementInstance(
            U2UContentAdvertisementImpl.getAdvertisementType(),
            new U2UContentAdvertisementImpl.Instantiator());
}
```

Figura 27. *Registro del nuevo anuncio de tipo U2UContentAdvertisementImpl creado para representar los archivos compartidos en la red P2P*

Fuente: Autores del proyecto.

3.4.1.3 Módulo de Carga de archivos

El módulo de carga de archivos es el encargado de manejar la carga de un archivo solicitada por otros iguales en la red. Este módulo está encapsulado dentro de una clase llamada U2UUploadingManager. Cada instancia de ésta clase es creada para representar la carga de un archivo dentro de la red P2P, es decir, un archivo compartido dentro de la red por un igual hacia otros iguales.

La clase U2UUploadingManager es responsable de:

1. Manejar los conexiones con otros iguales a través de las instancias del protocolo para la transferencia de archivos – U2U File Sharing Protocol (U2UFSP).
2. Decidir si atiende a un igual cliente o no.
3. Manejar los eventos generados por el protocolo U2UFSP.
4. Leer los trozos solicitados de un archivo desde el sistema de archivos.
5. Manejar la tabla que representa los trozos de un archivo en la base de datos U2UClient.

Cada instancia de la clase U2UUploadingManager maneja un conjunto de instancias fijas de la clase U2UFileSharingProtocol, las cuales representan conexiones con iguales remotos por donde se transmiten los datos de archivos siendo compartidos.

El protocolo U2UFileSharingProtocol será explicado con más detalle en el apartado 3.5.

El diseño de la clase U2UUploadingManager incluye la interacción con la clase U2UUMThinker. La clase U2UUMThinker fue creada para recibir notificaciones de eventos generados por el protocolo U2UFileSharingProtocol. Además, una instancia de la clase U2UUMThinker se encarga de recibir eventos y convertirlos en problemas representados por la clase U2UUMThinkerProblem, creada para representar los tipos de problemas, e inmediatamente agregarlos en una cola con bloqueo[11] (LinkedBlockingQueue). Más adelante los problemas son analizados por el U2UUMThinker y convertidos en tareas, representadas por la clase U2UUMExecutorTask, creada para representar los tipos de tareas asignadas al protocolo. Las tareas posteriormente son ingresadas en una cola con bloqueo para que luego la instancia del U2UUploadingManager ejecute esas tareas con la ayuda de instancias del protocolo U2UFSP. (*Ver* **Figura 28**)

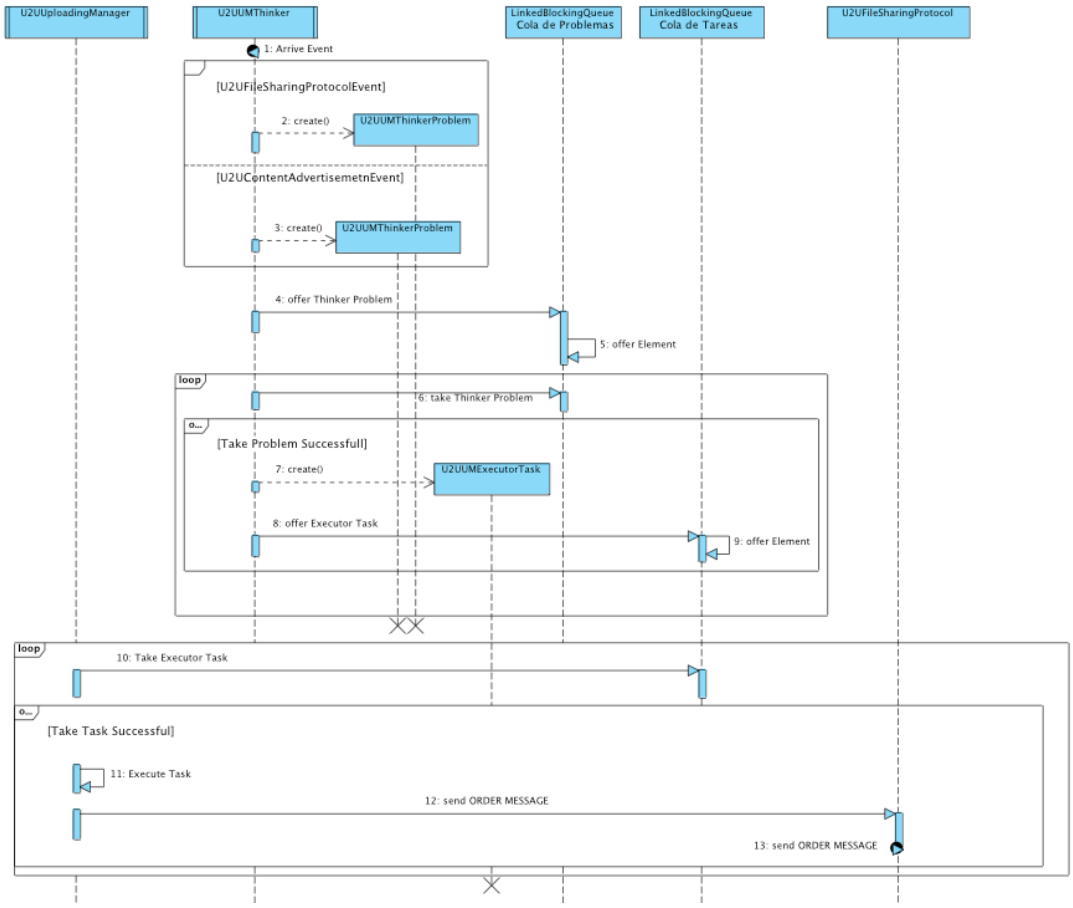


Figura 28. Diagrama de secuencia para el proceso de carga de un archivo dentro del aplicativo U2U 1.0.

Fuente: Autores del proyecto.

[11] Las colas con bloqueo permiten mantener bloqueado a un hilo hasta que hayan elementos en la cola.

Las instancias de la clase U2UUploadingManager son almacenadas dentro de un grupo de hilos de tamaño fijo según el número de archivos que se puedan compartir activamente. Lo mismo pasa con los protocolos U2UFSP, son almacenados dentro de una flota de hilos que se mantiene constante de acuerdo a la configuración realizada por el usuario. La **Figura 29** muestra los grupos de hilos para las cargas y los protocolos, y los mapas que almacenan las referencias a los protocolos activos, comunicándose con otros iguales en la red, y los protocolos en cola, aquellas instancias del protocolo esperando volver a conectarse a un igual remoto, debido a un mensaje de respuesta de tipo 401 (Los mensajes de respuesta se verán con más detalle en el apartado 3.5.1.1).

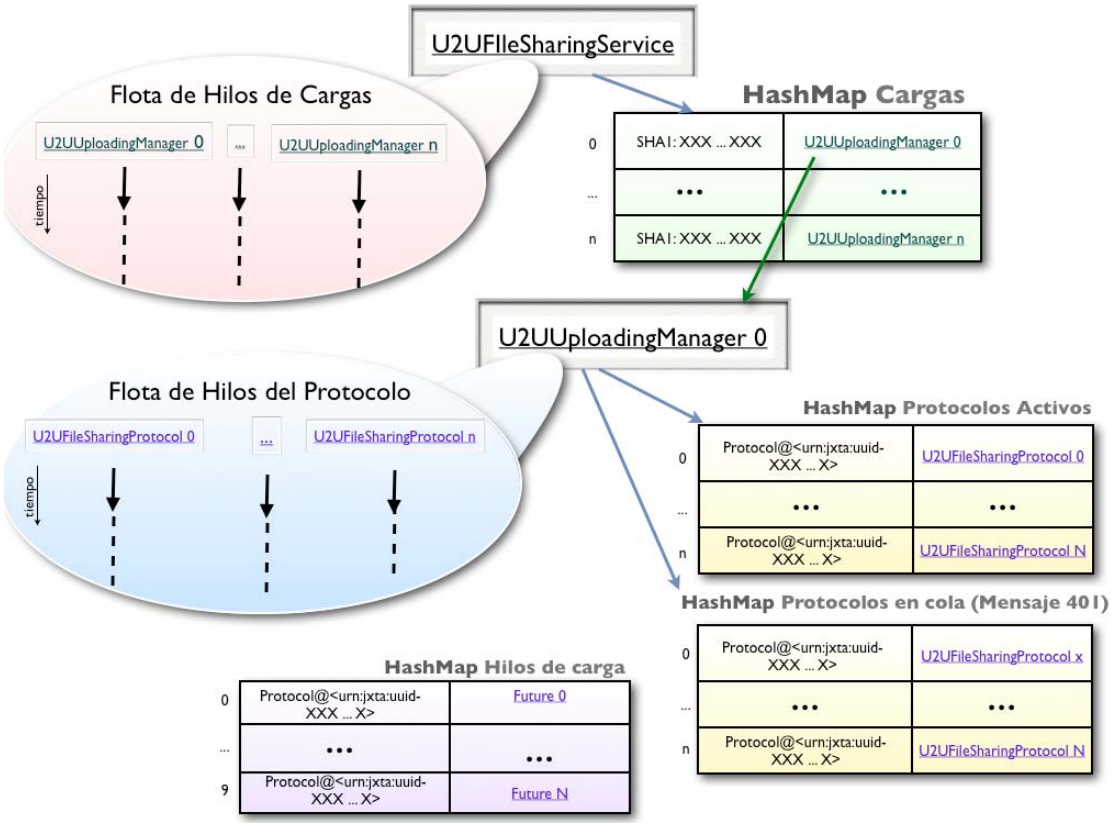


Figura 29. Grupos de hilos creados para manejar las cargas de archivos y las instancias del protocolo utilizadas por estas cargas (Instancias de la clase U2UUploadingManager y U2UFileSharingProtocol respectivamente)

Fuente: Autores del proyecto.

Una instancia del módulo de carga de archivos puede compartir el mismo archivo a tantos iguales según como esté configurado el igual. Para establecer el número de iguales a los cuales se puede compartir un mismo archivo, el

usuario debe dirigirse al panel de **Preferencias** en el menú **Archivo** de U2U (*Ver* **Figura 30**). En el panel de **Preferencias** específicamente en la ficha **Actividades** el usuario puede establecer en la caja de texto **Número de conexiones máximas por carga**, a cuántos iguales puede compartirse un mismo archivo.

Figura 30. Ejemplo de configuración de 5 conexiones máximas por archivo compartido. Panel de Configuración

Fuente: Autores del proyecto.

El número de archivos compartidos por un igual dentro de la red P2P está limitado por las capacidades de la máquina donde corre la aplicación U2U, es decir, la capacidad de almacenamiento del computador. A diferencia de los archivos compartidos pasivamente, es decir, los que se encuentran disponibles en la red para que otros iguales los descarguen pero que no se están descargando, se encuentran los archivos compartidos activamente, aquellos que están disponibles para compartirlos con otros iguales en la red P2P y que *están siendo descargados por otros iguales en la red, los archivos compartidos activamente si tienen un límite. El número de instancias del módulo de carga de archivos representa el número de archivos activos compartidos por un igual al mismo tiempo en un determinado instante. En el panel de **Preferencias** específicamente en la ficha **Actividades** el usuario puede establecer cuántos archivos compartir al mismo tiempo a otros iguales en la red modificando el valor de la caja de texto **Número de archivos compartidos**.

3.4.2 Igual de Descarga (Igual modo Cliente)

Un igual busca y descarga archivos por medio de los módulos de solicitud y descarga respectivamente, cuando se comporta como un cliente dentro de la red P2P.

3.4.2.1 Módulo de Solicitud

El módulo de solicitud se creó para buscar los iguales que contengan un archivo con el mismo nombre del archivo buscado por un usuario.

Para realizar la búsqueda de un archivo, el módulo de solicitud utiliza el servicio de descubrimiento *Discovery Service* para encontrar el anuncio que representa el archivo sobre la red P2P.

3.4.2.2 Módulo de Descarga

El módulo de descarga de archivos fue creado para manejar las descargas de archivos.

La clase que representa el módulo de descarga se llama U2UDownloadingManager. Cada instancia de la clase U2UDonwloadingManager representa una descarga que está siendo ejecutada por el igual.

La clase U2UDownloadingManager se diseñó para ser responsable de:

1. Crear y eliminar instancias del protocolo U2UFSP e ingresarlas a la flota de hilos.
2. Enviar órdenes a cada una de las instancias del U2UFSP.
3. Manejar los eventos generados por las instancias U2UFSP.
4. Serializar los trozos que lleguen de un archivo desde otro igual.
5. Manejar la tabla responsable de almacenar los trozos de un archivo en la base de datos U2UClient.
6. Guardar el estado de una descarga.
7. Preguntar acerca de los trozos de un archivo disponibles en la red P2P.

El diseño hecho para el comportamiento de la clase U2UDownloadingManager se asemeja al de una relación jefe - trabajadores. El jefe está representado por una instancia de la clase U2UDownloadingManager y cada trabajador está representado por una instancia de la clase U2UFileSharingProtocol, donde el jefe, está encargado de recibir notificaciones de eventos y convertirlos en tareas para enviárselas a los trabajadores, los cuales finalmente las ejecutan.

La clase U2UDownaloadingManager con la clase U2UDMThinker. La clase U2UDMThinker fue diseñada para recibir notificaciones de eventos generados por búsquedas a través del módulo de U2URequestManager y por el protocolo U2UFileSharingProtocol. Una instancia de la clase U2UDMThinker se encarga de recibir un evento y convertirlo en un problema, representado por la clase U2UDMThinkerProblem, e inmediatamente agregarlo en una cola con bloqueo (LinkedBlockingQueue). Más adelante los problemas son analizados por el U2UDMThinker y convertidos en tareas, representadas por la clase U2UDMExecutorTask. Las tareas posteriormente son ingresadas en una cola con bloqueo para que luego la instancia del U2UDownloadingManager ejecute esas tareas con la ayuda de instancias del protocolo U2UFSP. En la **Figura 31** se puede apreciar el diagrama de secuencia [32] de las instancias de la clase U2UDownloadingManager y su interacción con la clase U2UDMThinker.

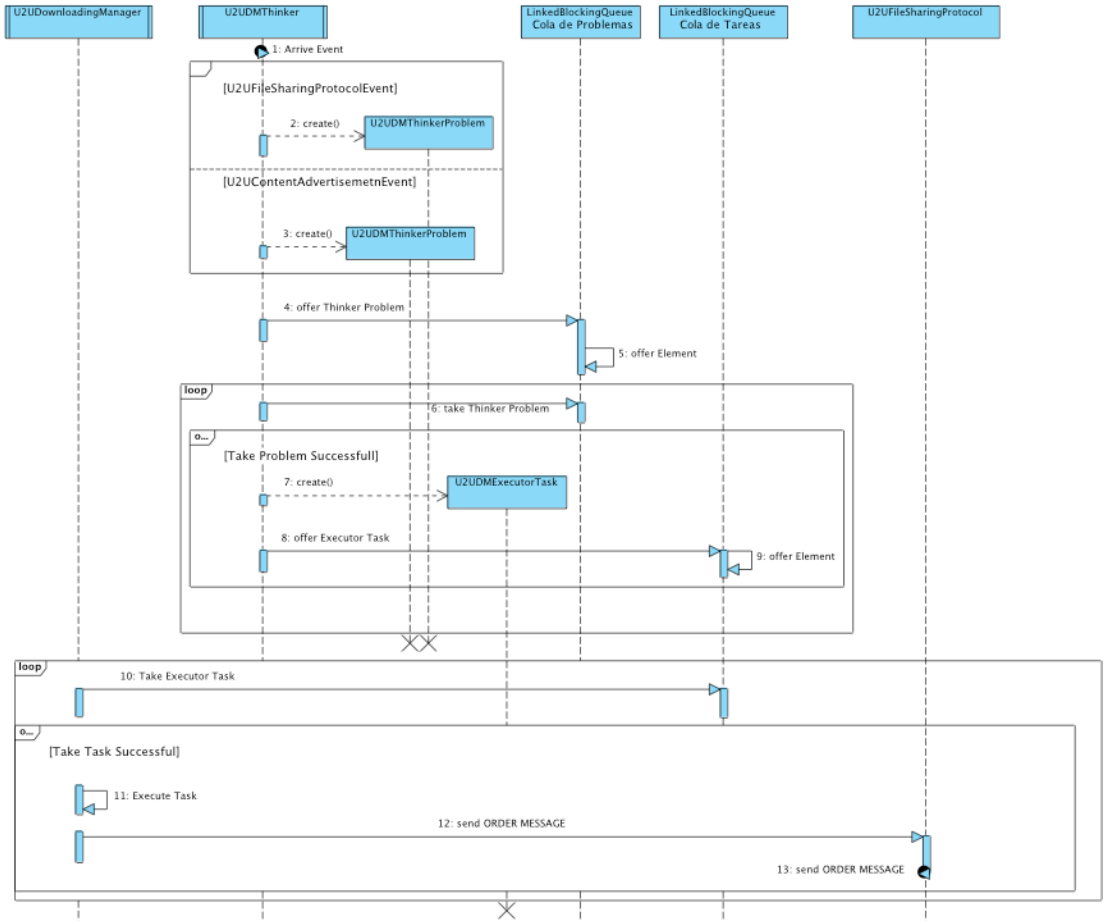


Figura 31. Diagrama de secuencias para el proceso de descarga de un archivo dentro del aplicativo U2U 1.0.

Fuente: Autores del proyecto.

El número máximo de descargas que un igual puede realizar al mismo tiempo es configurado por el usuario en el panel de configuración de la herramienta U2U. El usuario también debe configurar el número de iguales a los que se puede conectar un igual para descargar un archivo.

Cada descarga se ejecuta en un hilo separado del hilo principal de la aplicación. Cada instancia de la clase U2UDownloadingManager maneja instancias del protocolo U2UFileSharingProtocol ó U2UFSP, como se muestra en la **Figura 32**.

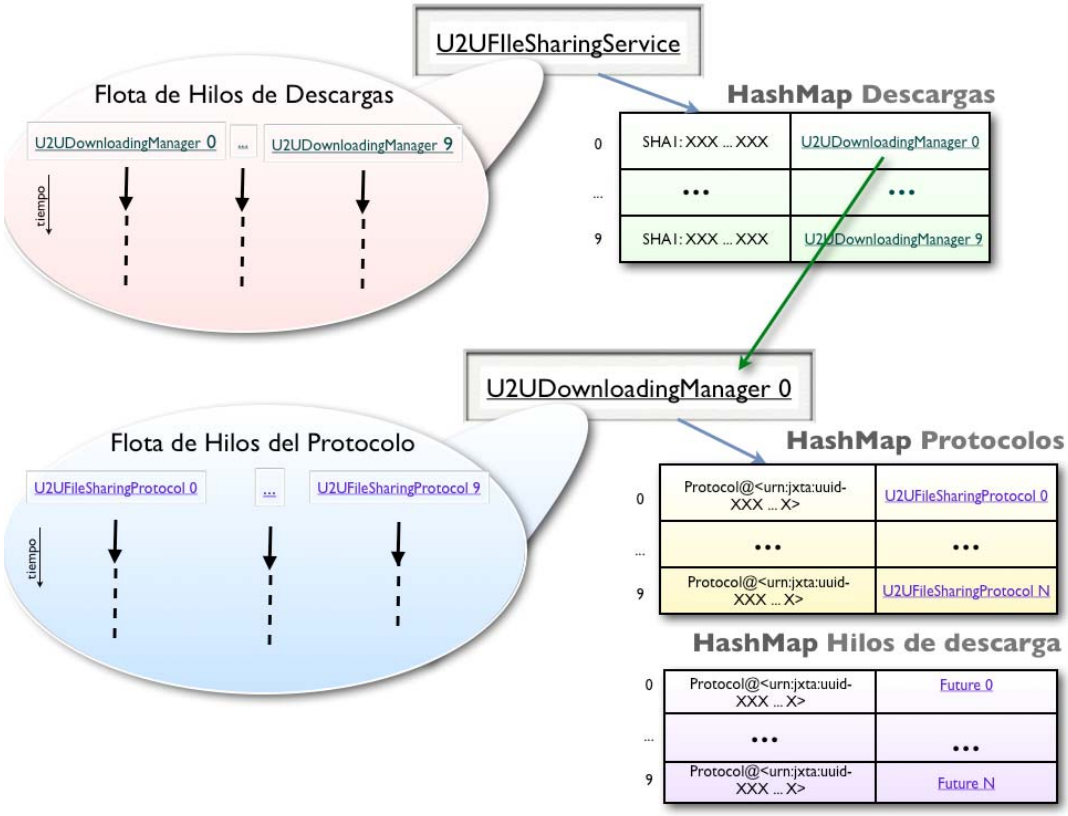


Figura 32. Grupos de hilos creados para manejar las descargas y las instancias del protocolo utilizadas por éstas descargas (instancias de la clase U2UDownloadingManager y U2UFileSharingProtocol respectivamente)

Fuente: Autores del proyecto.

La arquitectura de hilos que se utilizó para la ejecución de los protocolos es la arquitectura de hilos por conexión [33]. Cada hilo del protocolo representa una conexión con otro igual en la red P2P que posee el mismo anuncio (con igual SHA1) del archivo que el usuario desea descargar.

Las referencias a las instancias que representan descargas, protocolos e hilos del protocolo, son almacenadas dentro de un mapa dispersivo (HashMap[12]) que almacena parejas clave-valor.

Los hilos de descargas y del protocolo son creados dentro de un grupo de hilos o ejecutores [26]. Las grupos de hilos son utilizados para controlar el número de hilos concurrentes y así no afectar el rendimiento de la máquina virtual de Java y del sistema operativo (S.O) del computador. Además, los ejecutores permiten la reutilización de hilos mitigando el impacto de la gestión de hilos que hace la máquina virtual con el S.O.

Cuando una instancia del protocolo es enviada hacia los ejecutores como una tarea, éstos retornan un objeto de la clase Future, la cual permite consultar el estado de la tarea (ejecución de la instancia del protocolo).

El protocolo U2UFSP es utilizado para controlar la comunicación con otros iguales en la red y será explicado en el próximo apartado.

3.5 Protocolo

Por definición un protocolo es un conjunto de estándares que controlan la secuencia de mensajes que ocurren durante una comunicación entre entidades que forman una red[13].

Para poder controlar la comunicación entre dos iguales que están compartiendo recursos, en este caso archivos de tipo texto e imágenes, fue necesaria la creación de un protocolo al cual se le dio el nombre de Protocolo para la transferencia de archivos U2U o U2U File Sharing Protocol – U2UFSP.

3.5.1 U2U File Sharing Protocol

El protocolo U2U File Sharing Protocol ó U2UFSP creado se encuentra ubicado por encima de la pila de protocolos de JXTA (*Ver* **Figura 33**).

[12] Más información sobre la estructura de un mapa dispersivo en Java, disponible en Internet en: http://java.sun.com/j2se/1.4.2/docs/api/java/util/HashMap.html
[13] Más información sobre protocolo. Disponible en: http://es.wikipedia.org/wiki/Protocolo

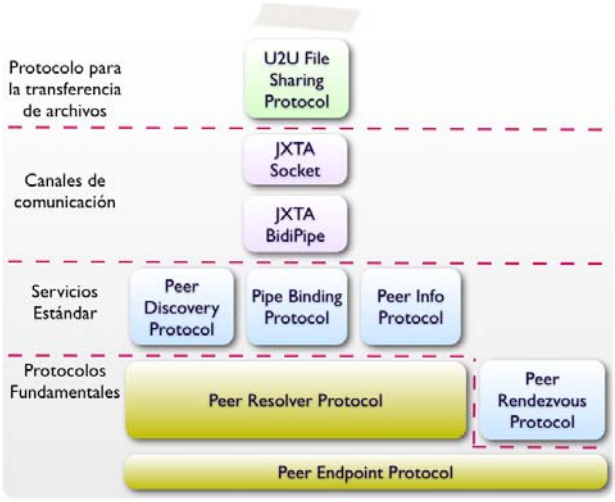


Figura 33. Localización del protocolo U2UFSP sobre la pila de protocolos de JXTA.

Fuente: Autores del proyecto.

El protocolo U2UFSP fue diseñado para controlar los mensajes enviados desde un igual actuando como cliente hacia otro igual remoto que actúa como servidor. Los mensajes enviados entre los iguales permiten que éstos puedan realizar las siguientes acciones:

- Conectarse a través de JXTASockets
- Pedir información sobre un archivo
- Solicitar la transferencia del trozo (*chunk*) de un archivo
- Enviar un mensaje de asentimiento
- Desconectarse de un igual remoto

El servicio U2UFSS visto en el apartado 3.4 hace uso del protocolo U2UFSP por medio de los módulos que componen la parte cliente y parte servidor de un igual.

3.5.1.1 Especificación del protocolo U2UFSP

El protocolo U2UFSP se implementó para enviar órdenes que identifican las solicitudes realizadas por un igual para descargar un archivo compartido por uno o varios iguales remotos dentro de la red P2P. Al mismo tiempo, el protocolo maneja una serie de respuestas a los mensajes de órdenes dados por el protocolo desde otro igual en la red (*Ver* **Figura 34**).

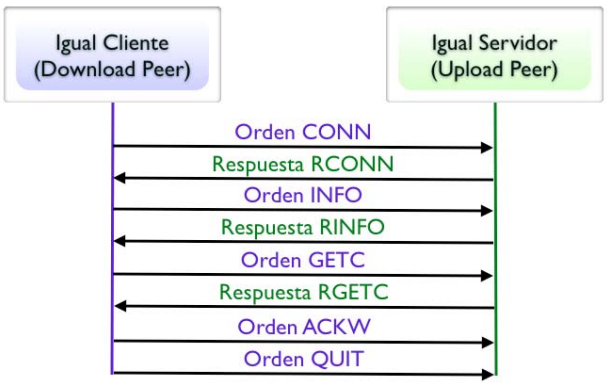


Figura 34. Mensajes de órdenes y respuestas creados para ser manejados por el protocolo U2UFSP entre iguales.

Fuente: Autores del proyecto.

Los mensajes de tipo orden manejados por el protocolo son los encargados de enviar peticiones de conexión, solicitar información de un archivo, pedir la descarga de un trozo (*chunk*) de un archivo, mandar confirmación de una transferencia y de una finalización de una conexión.

Los mensajes de respuesta manejados por el protocolo son los encargados de responder a las solicitudes realizadas por un mensaje de orden. Los mensajes de respuesta fueron clasificados dentro de 3 tipos:

1. Respuestas de finalización afirmativa: Son aquellas respuestas creadas para ser enviadas a otro igual cuando se ha recibido un mensaje de orden y éste ha finalizado exitosamente.

2. Respuestas de finalización negativa transitoria: Respuestas creadas para enviarse cuando la petición solicitada por un mensaje de orden no se ha podido realizar debido a que el igual receptor de la orden no se encuentra disponible porque no puede manejar más cargas en ese momento, pero existe la posibilidad de efectuar la orden en otro intento mientras el igual receptor vuelve a estar disponible.

3. Las respuestas de finalización negativa permanente: Son respuestas que se envían cuando el igual receptor del mensaje de orden no puede ejecutar la petición realizada debido a que el igual está desconectado, o por otras razones desconocidas no puede definitivamente atender al igual emisor de la orden.

Los mensajes manejados por el protocolo tienen un formato general para su estructura. A continuación se puede observar en la **Tabla 5** el formato para los mensajes de orden y de respuesta.

Tipo De Mensaje	Formato
Orden	<length prefix><order ID><payload> 4 Bytes 1Byte n Bytes
Respuesta	<length prefix><response ID><response number><payload> 4 Bytes 1Byte 2 Bytes (SHORT) n Bytes

Tabla 5. Formato general diseñado para los mensajes del protocolo U2UFSP

La etiqueta <*lenght Prefix*> de los mensajes de orden y de respuesta contiene el tamaño del mensaje siendo enviado o recibido. Esto permite saber cuántos Bytes en total se van a leer por mensaje cuando llegue a su destinatario.

Las etiquetas <*order ID*> y <*response ID*> contienen los identificadores del mensaje de orden y respuesta respectivamente. Los mensajes de respuesta deben tener el valor de la etiqueta <response ID> igual al de la etiqueta <order ID> del mensaje de orden que generó la respuesta. Esto se hace para verificar que el igual remoto responde al mensaje de orden correcto enviado por el igual emisor.

La etiqueta <*response number*> almacena el número de respuesta siendo enviada hacia un igual remoto. Los números que identifican los mensajes de respuestas se muestran en la **Tabla 6**.

Número	Tipo de Mensaje de Respuesta
2	Respuestas de finalización afirmativa
201	Solicitud de transmisión aceptada
211	Trozo solicitado disponible
212	Trozo solicitado No disponible
221	Información de archivo completo
222	Información de archivo incompleto
260	Finalización correcta
4	Respuestas de finalización negativa transitoria
401	Servicio de transmisión no disponible
411	Solicitud de info no ejecutada
421	Acción solicitada cancelada
5	Respuestas de finalización negativa permanente
501	Error de sintaxis, orden no conocida
511	Error de sintaxis, parámetros incorrectos

Tabla 6. Tipo de mensajes de respuesta creados para que los iguales respondan a través del protocolo U2UFSP a otros iguales en la red P2P.

La etiqueta <*payload*> corresponde a la carga útil, es decir, los datos que se envían a través del mensaje de orden o de respuesta. Dependiendo del tipo de mensaje ésta etiqueta puede estar representada por una o más etiquetas que

Mensaje	Tipo	Estructura
CONN	Orden	<1+54+n><CONN><Protocol Versión><File's SHA1><PeerID><(Optional)Certificate> Payload 1Byte 1Byte 20 Bytes 33 Bytes n Bytes
RCONN	Respuesta	<1+2+34><RCONN><Response Number><Protocol Versión><PeerID> Payload 1Byte 2 Byte 1 Bytes 33 Bytes
INFO	Orden	<1+21><INFO><Query Type><File's SHA1> Payload 1Byte 1 Byte 20 Bytes
RINFO	Respuesta	<1+2+(1+n)><RINFO><Response Number><Query Type><List> Payload 1Byte 2 Bytes 1 Bytes n Bytes
GETC	Orden	<1+2><GETC><Index> Payload 1 Byte 2 Bytes
RGETC	Respuesta	<1+2+n><RGETC><Response Number><chunk Bytes> Payload 1Byte 2 Bytes n Bytes
ACKW	Orden	<1+1><ACKW><successful transfer> Payload 1 Byte 1 Byte
QUIT	Orden	<1><QUIT> 1 Byte

contienen los datos útiles siendo enviados. En la **Tabla 7** se ilustra la estructura de cada uno de los mensajes manejados por el protocolo U2UFSP.

Tabla 7. Estructura de los mensajes de orden y respuesta manejados por el protocolo U2UFSP

El mensaje de orden INFO puede solicitar al igual remoto dos tipos de información: Consultar cuáles son los trozos del archivo que contiene el igual remoto ó cuál es la lista de los mensajes de resumen del archivo. En la **Tabla 8** se muestran los tipos de consulta hechos por un mensaje de orden INFO y la estructura de la respuesta a éste mensaje.

Tipo de Orden INFO	Significado	Formato de Respuesta RINFO
0	Trozos del archivo que contiene el igual remoto	<T> | <F> (1X 2X ... nX) 1Byte 1Byte 2 Bytes 2 Bytes n Bytes 1Byte
1	Lista de mensajes de resumen (SHA1) de todos los trozos del archivo	H (1XXXX 2XXXX ... nXXXX) 1Byte 1Byte 22 Bytes 22 Bytes 22 Bytes 1Byte

Tabla 8. Tipos de mensaje de orden INFO creados para consultar el número de trozos de un archivo o la lista de mensajes de resumen del archivo que tiene un igual remoto.

La **Tabla 9** muestra la funcionalidad ofrecida por cada uno de los mensajes enviados y recibidos por el protocolo U2UFSP.

Mensaje	Función
CONN	Establece un canal de comunicación con un igual remoto. Este mensaje puede contener un certificado que identifique al igual emisor.
RCONN	Respuesta al mensaje CONN que retorna el identificador del igual remoto.
INFO	Consulta los datos de un archivo contenidos en un igual remoto a través del mensaje (SHA1) de resumen del archivo.
RINFO	Respuesta al mensaje INFO que devuelve la información del archivo consultado dentro de una lista que especifica si el igual remoto contiene el archivo completo o partes del archivo y cuáles partes contiene.
GETC	Pide la transferencia de un trozo de un archivo a través de la posición del trozo.
RGETC	Respuesta al mensaje GETC que trae consigo los datos del trozo pedido.
ACKW	Mensaje de asentimiento enviado para confirmarle al igual remoto que se han recibido los datos pedidos.
QUIT	Cierra el canal de comunicación establecido con el igual remoto.

Tabla 9. Función de cada uno de los mensajes creados para ser manejados por el protocolo U2UFSP

El protocolo U2UFSP se considera un protocolo fiable ya que maneja JXTASockets para establecer canales de comunicación con otros iguales en la red P2P. Como se expuso en el apartado 2.2.6.5.1, los JXTASockets proporcionan: Fiabilidad, aseguran la secuenciación del mensaje, y garantizan la entrega del mensaje.

En el ANEXO E muestra el procedimiento para enviar un mensaje de orden CONN desde un igual actuando como cliente hasta un igual actuando como servidor.

3.6 Comando sobre el SHELL U2U para compartir archivos

El Shell U2U proporciona una serie de comandos, heredados del Shell de JXTA, que permiten ejecutar acciones básicas sobre la red P2P: buscar iguales conectados, ver el estado de un igual *Rendezvous*, ver información sobre los anuncios encontrados en una consulta, entre otras.

Para acceder a las funcionalidades ofrecidas por el servicio U2U File Sharing Service ó U2UFSS , y debido a la arquitectura en capas propuesta, hubo la necesidad de crear un comando en el Shell que proporcionara el acceso al servicio U2UFSS de tal manera que éste comando fuese un canal de comunicación entre las utilidades ofrecidas por el servicio de archivos compartidos U2U (U2UFSS) y la interfaz gráfica de usuario. El comando creado en el Shell U2U para acceder al servicio U2UFSS fue denominado: **comando u2ufss.**

Cada uno de las opciones del comando u2ufss fueron creadas para acceder a las utilidades brindadas por el servicio U2UFSS. En la **Tabla 10** se muestran las diferentes opciones del comando u2ufss.

Opción del comando	Función
u2ufss -init	Inicia el servicio U2U File Sharing Service.
u2ufss -stop	Detiene la ejecución del servicio U2U File Sharing Service.
u2ufss -share -p <File's path> [-d]	Comparte un archivo especificando la localización y la descripción del archivo.
u2ufss -unshare <File's name>	Deja de compartir un archivo especificando el nombre del archivo.
u2ufss -search <File's name or description> -n | -d	Busca un archivo en la red por su nombre o descripción.
u2ufss -download <Enviroment's variable>	Descarga un archivo representado en una variable de entorno del Shell.
u2ufss -stopdownload <Enviroment's variable>	Detiene y elimina la descarga de un archivo representado en una variable de entorno del Shell.
u2ufss -pausedownload <Enviroment's variable>	Detiene la descarga de un archivo representado en una variable de entorno del Shell.
u2ufss -restartdownload <Enviroment's variable>	Reanuda la descarga de un archivo representado en una variable de entorno del Shell.
u2ufss -progress	Consulta el progreso de las descargas activas.
u2ufss -register	Registra el oyente para las búsquedas realizadas por éste igual
u2ufss -addlistener	Agrega y registra un oyente para eventos del Servicio U2UFSS
u2ufss -removelistener	Elimina un oyente para eventos del Servicio U2UFSS

Tabla 10. Opciones brindadas por el comando u2ufss creado en el Shell U2U para controlar el servicio para compartir archivos U2UFSS

Para iniciar la ejecución del servicio U2UFSS se utiliza la opción –init del comando u2ufss. Una vez que el servicio sea inicializado se pueden utilizar las demás opciones del comando que permiten, en general, compartir un archivo y descargar archivos compartidos desde otros iguales. Cuando el servicio U2UFSS es detenido a través del comando u2ufss y la opción –stop, todos los archivos compartidos por el igual no podrán ser accedidos por otros iguales en la red, y las descargas que estaban ejecutándose se detendrán.

El usuario del aplicativo U2U no tiene acceso directo al Shell U2U, ni tampoco al comando u2ufss. La capa de la GUI es la encargada de utilizar el Shell U2U y, por lo tanto, utiliza el comando u2ufss.

Cada una de las funcionalidades del comando u2ufss acceden al servicio U2U File Sharing Service y a los módulos que éste servicio ofrece, permitiendo ejecutar y poner en funcionamiento el servicio

3.7 Mecanismo de Seguridad

A continuación se indican las estrategias utilizadas para cumplir con los requisitos básicos de seguridad del aplicativo U2U:

- **Integridad:** La integridad de la información hace referencia a que la información permanezca inalterada a menos que sea modificada por personal autorizado.

Un usuario primero debe ingresar con su nombre y contraseña al aplicativo P2P para poder compartir, dejar de compartir o descargar archivos en la red P2P. Si los archivos que se estaban compartiendo por este usuario son borrados del sistema de archivos, el aplicativo U2U elimina estos archivos de la base de datos y no permite que se compartan anuncios de archivos inexistentes. Si un usuario U1 comparte un archivo en la red P2P éste archivo es publicado con el mensaje de resumen del archivo, SHA1[14], de tal manera que si otro usuario U2 está interesado en descargar éste archivo lo pueda identificar por medio del mensaje de resumen (SHA1) del archivo compartido por el usuario U1. Si otro usuario U3 comparte un archivo con el mismo nombre del archivo publicado por el usuario U1, pero la información es diferente o ha sido modificado el archivo original publicado por U1, el archivo compartido por el usuario U3 será publicado con un mensaje de resumen (SHA1) diferente, lo que hará que el archivo compartido por U3 pueda ser diferenciado del archivo original publicado por U1, y otros usuarios en la red que quieran descargar el archivo publicado por U1 no descarguen archivos corruptos, sino copias del archivo original que contengan el mismo mensaje de resumen, manteniendo así la integridad de los archivos compartidos en la red.

Por otra parte debido a que la transferencia de los datos de un archivo que se está descargando se realiza mediante la transmisión de trozos(chunks) de esté desde los iguales que poseen el archivo completo o partes de él, el sistema reutiliza el concepto anteriormente plasmado y realiza una verificación de la no corrupción de los trozos descargados, mediante un proceso de cálculo del SHA1 del trozo entrante y haciendo la respectiva comparación con los datos almacenados en la base de datos que relacionan la posición de un trozo en el

[14] Para mayor información sobre el algoritmo de reducción criptográfica SHA-1, Disponible en Internet en http://es.wikipedia.org/wiki/SHA

archivo y su respectivo SHA1, mecanismo que evita la corrupción de la descarga total del archivo debido al envío de unos cuantos trozos corruptos.

Con relación a la transmisión de los datos, el protocolo desarrollado en esta investigación proporciona al sistema una comunicación fiable que minimiza al máximo los errores en la transmisión, asegurando que los datos se recibirán exactamente como los envío el igual emisor, lo cual induce que si llegan trozos corruptos es debido a que el emisor dispone de información corrupta.

- **Disponibilidad**: La disponibilidad de la información se refiere a que la información pueda ser recuperada en el momento que se necesite.

Un usuario puede descargar un archivo no solo desde un igual sino desde varios iguales dentro de la red P2P. En el instante en el que un igual comparte un archivo y otros iguales empiezan a descargarlo, existen más posibilidades de acceder a este, debido a que no solo se puede descargar desde iguales que posean una copia completa del archivo sino de aquellos que disponen de algunos trozos de este, además una vez un igual empieza la descarga del archivo se convierte inmediatamente en servidor de este, compartiendo la información de SHA1 de los trozos que lo componen y los trozos que haya descargado, significando para el sistema que entre más iguales estén descargando el archivo, más iguales lo estarán compartiendo, creciendo los recursos de la red directamente con el número de usuarios en esta, evitando así un único punto de fallo, ya que no existe un servidor encargado del alojamiento centralizado de los archivos, sino que todos los iguales se comportan como cliente y como servidores de los archivos compartidos.

- **Autenticación:** La autenticación del sistema significa que el sistema debe ser capaz de verificar la identidad de los usuarios.

El usuario es autenticado por medio de los datos almacenados en dos archivos: el archivo de configuración y el archivo de contraseña el cual almacena encriptada la contraseña ingresada por el usuario con el algoritmo AES (Advanced Encryption Standard) [23]. El archivo de configuración proporciona, entre otras cosas, el nombre del usuario. El aplicativo U2U verifica que los datos nombre de usuario y contraseña ingresados en el panel de autenticación sean iguales a la información almacenada en los archivos anteriormente mencionados. Si la información no es la correcta el usuario no puede ingresar al aplicativo U2U.

El mecanismo de seguridad anteriormente descrito es un mecanismo básico de seguridad para el tratamiento que debe dársele a un sistema distribuido P2P. En el apartado de las recomendaciones se mencionan algunos puntos

importantes que deben tenerse en cuenta para trabajos posteriores que quieran profundizar sobre la seguridad en redes P2P.

Capítulo 4

RESULTADOS DE LAS PRUEBAS REALIZADAS A LA HERRAMIENTA SOFTWARE

Uno de los objetivos de la realización de éste proyecto fue llevar a cabo una serie de pruebas a la herramienta software desarrollada comparándola con un software para transferencia de archivos de la arquitectura cliente/servidor, de tal manera que se pudiera demostrar el potencial de las redes P2P para la transferencia de archivos entre computadores en un entorno local como una LAN o una Intranet.

Para la realización de las pruebas se utilizaron computadores personales de características homogéneas, pertenecientes a una misma subred. Se utilizó la primera configuración propuesta de la red P2P: Un igual *Rendezvous* y varios iguales *Full Edge* con el software U2U y para simular el comportamiento de la arquitectura Cliente – Servidor se utilizó el servicio FTP por medio de la consola de comandos. Se configuró un computador como servidor del archivo y los demás computadores como clientes. El servicio FTP para computadores con sistema operativo Windows XP Professional, que fue el sistema operativo utilizado en las pruebas, define un máximo de 10 conexiones cliente concurrentes para el servidor FTP.

El archivo escogido para las pruebas contiene los tipos de archivos, PDF, DOC, ODT, ODP, XLS, PPT, MP3, ZIP, TAR.GZ y EXE. Las características de los equipos utilizados en las pruebas de transmisión y las características del archivo que fue transferido se pueden consultar en el ANEXO F.

Los resultados de la realización de esta prueba indican el comportamiento de las arquitecturas P2P y Cliente – Servidor, a medida que aumenta el número de nodos conectados a la red queriendo ejecutar un mismo servicio: Transferir un archivo desde un computador servidor, que posee el archivo, hasta un computador cliente que desea tenerlo. A continuación se muestran los resultados de las pruebas realizadas a la herramienta software U2U acompañados de los resultados obtenidos con FTP.

4.1 Resultados: Tiempos de respuesta

El tiempo de respuesta es el tiempo que pasa desde que se envía una comunicación y se recibe la respuesta. En este caso se midió el tiempo que gasta un igual, actuando como cliente, desde que envía una solicitud de

descarga de un archivo hasta que recibe todos los trozos del archivo (compartiendo el archivo en la red P2P hacia otros iguales desde el momento en que descarga el primer trozo), reconstruye el archivo con los trozos recibidos y almacena en la base de datos que el archivo está completo. En el caso de C/S el tiempo que gasta un cliente ftp en ejecutar el comando get.

En la **Tabla 11** y en la **Figura 35** se pueden observar los resultados obtenidos por los iguales *EDGE* dentro de la red P2P de acuerdo al número de nodos que se encontraban en la red compartiendo y descargando el mismo archivo, y los resultados obtenidos por los clientes FTP dentro de la arquitectura Cliente - Servidor.

CLIENTES TOTALES	ID CLIENTE	BYTES RECIBIDOS [Bytes] P2P	TIEMPO DE RESPUESTA [seg] P2P	BYTES RECIBIDOS [Bytes] C/S	TIEMPO DE RESPUESTA [seg] C/S
1	130	434.376.768	770,407	436.083.731	35,06
3	130	434.376.768	715,863	436.083.731	112,09
	131	434.376.768	705,64	436.083.731	103,17
	132	434.376.768	735,421	436.083.731	111,55
6	126	434.376.768	523,922	436.083.731	223,39
	127	434.376.768	534,124	436.083.731	221,88
	128	434.376.768	558,114	436.083.731	219,88
	130	434.376.768	532,172	436.083.731	223,09
	131	434.376.768	583,406	436.083.731	223,31
	132	434.376.768	513,563	436.083.731	223,38
10	122	434.376.768	516,172	436.083.731	371,65
	123	434.376.768	481,125	436.083.731	367,05
	124	434.376.768	492,258	436.083.731	371,47
	126	434.376.768	501,872	436.083.731	369,73
	127	434.376.768	468,053	436.083.731	370,98
	128	434.376.768	486,344	436.083.731	368,53
	129	434.376.768	484,71	436.083.731	369,25
	130	434.376.768	463,828	436.083.731	371,19
	131	434.376.768	471,854	436.083.731	369,89
	132	434.376.768	521,47	436.083.731	368,31

Tabla 11. Resultados: Tiempo de respuesta en arquitecturas P2P y Cliente – Servidor [*]

Como se puede apreciar en la **Tabla 11** y en la **Figura 35**, y tomando como referencia el hecho que el mejor tiempo de respuesta que se puede obtener con

[*] Los identificadores (ID) de los iguales clientes fueron tomados como el último número de la dirección IP del igual.

el hardware utilizado es el valor que arroja ftp cuando solo existe un cliente concurrente, es decir, 35.06 segundos entre la petición de descarga del archivo

(414,25MB) y el fin de la descarga, el tiempo de respuesta obtenido con P2P con un solo cliente es 21.97 veces el tiempo de respuesta referencia, esto debido a la sobrecarga que se asume por el uso de los Sockets de JXTA, el procesamiento que deben realizar las instancias del U2UFileSharingProtocol en el igual cliente e igual servidor, la instancia de U2UUploadingManager en el servidor, la instancia de U2UDownloadingManager en el cliente y las latencia debido al acceso a la base de datos.

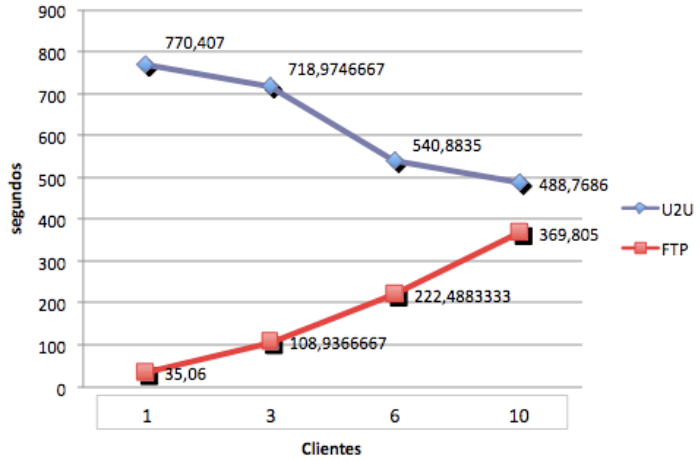


Figura 35. Comparación del promedio de tiempos de respuesta obtenidos con el software U2U (Arquitectura P2P) y con FTP (Arquitectura C/S).

Conforme aumenta el número de clientes concurrentes en la arquitectura Cliente – Servidor (C/S), los tiempos de respuesta obtenidos van aumentando siendo con tres , seis y diez clientes de 3.11, 6.34 y 10.54 veces el tiempo de respuesta referencia respectivamente, lo cual nos muestra una evidente perdida de eficiencia en la arquitectura C/S conforme aumenta el número de clientes; mientras que con U2U 1.0 los tiempos de respuesta a pesar de la sobrecarga impuesta van disminuyendo a medida que aumenta el número de clientes concurrentes, siendo con tres, seis y diez clientes de 20.50, 15.42 y 13.94 veces el tiempo de respuesta referencia respectivamente, lo cual percibe una mejora en la eficiencia de U2U 1.0; pudiendo deducir que a medida que más clientes ingresen a la red U2U con el objetivo de descargar el archivo, los tiempos de respuesta mejoran para el conjunto de clientes debido a la naturaleza distribuida de la red y a la estrategia de descarga de la misma, mientras que en C/S si no existiera la restricción impuesta en el SO para el número de clientes concurrentes de ftp los tiempos de respuesta se

degenerarían hasta llegar al punto en que se es imposible atender a la demanda de los clientes.

Se puede decir que FTP se comporta de manera satisfactoria cuando el número de clientes es pequeño, pero se ve muy afectado cuando éste número aumenta, contrario a lo que sucede con el software U2U, que da a sus clientes mejores tiempos de respuesta a medida que aumentan los nodos en la red P2P.

4.2 Resultados: Velocidad de transferencia

La velocidad de transferencia es el promedio del número de bits, caracteres o bloques, que se transfieren entre dos computadores, por unidad de tiempo y se mide en KiloBytes por segundo (KB/s). En la **Tabla 12** se pueden apreciar los resultados obtenidos para la velocidad de transferencia de un archivo cuando existen en la red un número total de 2, 4, 7 y 11 nodos.

CLIENTES Totales	ID Cliente	BYTES RECIBIDOS [Bytes] P2P	Velocidad [KB/s] P2P	BYTES RECIBIDOS [Bytes] C/S	Velocidad [KB/s] C/S
1	130	434.376.768	550,612939	436.083.731	11860,02
3	130	434.376.768	592,5659833	436.083.731	3890,34
	131	434.376.768	601,150817	436.083.731	4226,76
	132	434.376.768	576,8071112	436.083.731	3909,42
6	126	434.376.768	809,6549916	436.083.731	1952,12
	127	434.376.768	794,1902302	436.083.731	1965,45
	128	434.376.768	760,0527177	436.083.731	1983,33
	130	434.376.768	797,1033096	436.083.731	1954,71
	131	434.376.768	727,1026738	436.083.731	1952,8
	132	434.376.768	825,9864174	436.083.731	1952,25
10	122	434.376.768	821,8114553	436.083.731	1307,97
	123	434.376.768	881,6753702	436.083.731	1304,79
	124	434.376.768	861,7352334	436.083.731	1313,69
	126	434.376.768	845,2275929	436.083.731	1305,28
	127	434.376.768	906,2992065	436.083.731	1310,24
	128	434.376.768	872,2140347	436.083.731	1310,85
	129	434.376.768	875,1543449	436.083.731	1307,47
	130	434.376.768	914,5546679	436.083.731	1313,01
	131	434.376.768	898,9985515	436.083.731	1305,21
	132	434.376.768	813,462064	436.083.731	1173,5

Tabla 12. Resultados: Velocidad de transferencia en arquitecturas P2P y Cliente - Servidor[*]

[*] Ibid.

Con los datos que nos presenta la **Tabla 12** y la **Figura 36**, y tomando como velocidad de descarga promedio de referencia la obtenida por ftp cuando solo un cliente concurrente descarga el archivo de 414,25MB se obtiene la máxima velocidad de transferencia: 11860.02 KB/seg (100% del canal de comunicación de acuerdo al hardware utilizado en las pruebas). En la arquitectura P2P utilizando la aplicación desarrollada U2U 1.0 se obtuvo para solo un cliente concurrente que la velocidad de descarga promedio es el 4.64% de la velocidad referencia, esta perdida en el uso del canal de comunicación es debida principalmente a la perdida en el rendimiento impuesta por el uso de lo Sockets de Jxta fiables y a la sobrecarga impuesta por el servicio desarrollado sobre estos.

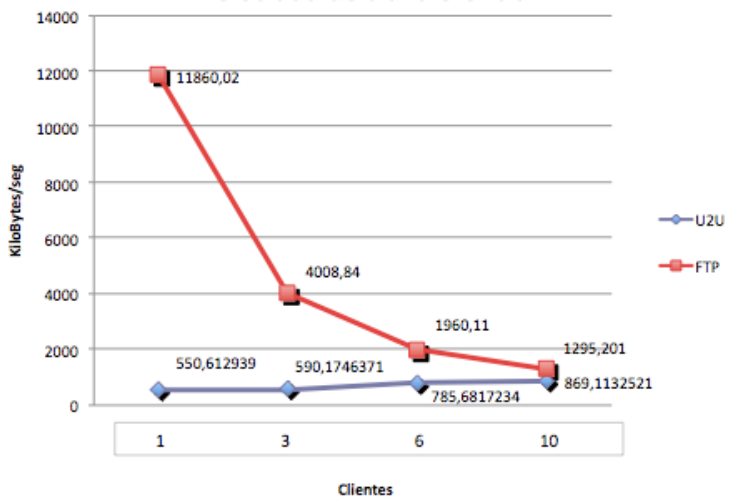


Figura 36. Comparación del promedio de velocidades de transferencia obtenidos con el software U2U (Arquitectura P2P) y con FTP (Arquitectura C/S).

A medida que aumenta el número de clientes concurrentes en la arquitectura Cliente – Servidor (C/S), las velocidades de descarga promedio obtenidas van disminuyendo siendo con tres, seis y diez clientes de 33.80%, 16.52% y 10.92% de la velocidad de descarga referencia respectivamente, lo cual nos muestra una evidente disminución en el uso del canal en la arquitectura C/S conforme aumenta el número de clientes; mientras que con U2U 1.0 las velocidades de descarga a pesar de la sobrecarga impuesta van aumentando a medida que el número de clientes concurrentes aumenta , siendo con tres, seis y diez clientes de 4.97%, 6.62% y 7.32% de la velocidad de descarga referencia respectivamente, mostrando una mejora en la velocidad de descarga de U2U 1.0; pudiendo deducir que a medida que más clientes ingresen a la red U2U con el objetivo de descargar el archivo, las velocidades de descarga mejorarán (a pesar de las perdidas en el rendimiento de los Sockets utilizados) para el conjunto de clientes debido a que cada igual aportara una porción de su ancho

de banda para soportar la carga de la red, es decir, cargara trozos del archivo a iguales remotos con el fin de que estos completen la descarga de este.

Capitulo 5

CONCLUSIONES

- Este trabajo de investigación permitió experimentar la escalabilidad ofrecida por las redes P2P, la cual se pudo alcanzar gracias a la implementación del servicio para compartir archivos U2UFSS, el cual le permite a los iguales conectados a la red P2P explotar sus facetas como cliente y como servidor de un archivo o de los trozos (*chunks*) de un archivo, dando como resultado un óptimo desempeño del sistema a medida que un gran número de nodos conectados comparten intereses iguales ofreciendo el mismo servicio y colaborándose mutuamente, ya que todos se comportan como clientes y como servidores al mismo tiempo.

- El servicio para compartir archivos, U2U File Sharing Service, fue creado para brindar la característica de robustez de las redes P2P al sistema distribuido, ya que permite que un igual descargando un archivo encuentre rápidamente otros iguales en la red actuando como servidores del mismo recurso. Esto debido a que se realizó la implementación necesaria para que apenas un igual reciba un trozo de un archivo, éste igual publique el archivo siendo descargado en la red P2P, y así lograr que otros iguales interesados en la descarga del archivo en común puedan descargar del igual los trozos que éste ya tenga disponibles. Esto permite que un menor tiempo se ofrezca sobre la red varios iguales actuando como servidores de donde se puedan descargar información de un mismo recurso, evitando caer en un único punto de fallo.

- La descentralización de los recursos, característica de los sistemas P2P, se soporta en el modo de operación del módulo de descargas del servicio U2U File Sharing Service, ya que ningún igual dentro de la red P2P se hace totalmente imprescindible, debido a que la descarga de los trozos de un archivo se hace de manera aleatoria en cada uno de los iguales que se encuentra descargando el mismo archivo, garantizando en un tiempo t, menor al tiempo total de descarga, los iguales tendrán trozos del archivo diferentes, lo que permite que en caso de que el igual que compartió el archivo completo por primera vez se caiga, pueda ser virtualmente reemplazado por la sumatoria de los trozos del archivo

descargados aleatoriamente por otros iguales, la cual puede ser menor o igual que el total de trozos del archivo, pero siempre va a ser mayor que los trozos de archivo contenido por cada uno de los iguales.

- El equilibrio de carga en la red P2P, se logró gracias a la implementación de una política de dar y recibir en el servicio U2U File Sharing Service, que permite a todos lo iguales pertenecientes a la red ser parte de los recursos disponibles para esta, haciendo responsable a cada igual de brindar una porción de su banda ancha para la carga de archivos a otros iguales y equilibrando de ésta manera la carga total de la red, entre todos los nodos y no sólo en uno de ellos.

- El desarrollo de este proyecto permitió la apropiación tecnológica de las redes P2P y la implementación de la tecnología para uno de sus posibles usos: compartición de archivos; posibilitando a los investigadores el desarrollo de soluciones basadas en esta arquitectura para problemáticas a nivel regional y nacional, que contemplen costos de inversión (HW, SW, ancho-de-banda), medio ambiente, comunicaciones y aprovechamiento de la infraestructura computacional existente en el entorno.

- El uso del framework JXTA fue fundamental en el desarrollo del proyecto ya que ofreció una alternativa multiplataforma, independiente del transporte de red y robusta, que permitió a los investigadores de este proyecto construir una aplicación P2P teniendo una base de servicios y protocolos que solucionaron algunos de los problemas básicos de conexión, búsqueda y nombramiento de los recursos de la red, y centrándo a los investigadores en el análisis, diseño y desarrollo de una solución para la transferencia de archivos en un ambiente distribuido.

- Con el diseño e implementación de la arquitectura en capas propuesta se logró que la funcionalidad del sistema (el servicio para compartir archivos U2U File Sharing Service) fuese independiente totalmente de la interfaz de usuario, llevando en este caso a los investigadores del proyecto a construir la solución desde la capa más baja hacia la más alta, realizando un proceso de análisis, diseño y desarrollo de la lógica del servicio logrando de éste modo reutilizar la funcionalidad desarrollada en el proyecto para que en un futuro pueda ser portada a interfaces de usuario más complejas o incluso integrándola con otros tipos de funcionalidades como computación distribuida o mensajería instantánea en una misma solución.

- El uso de la base de datos embebida DERBY ayudó a incrementar el rendimiento de la organización y manipulación de la información en cada uno de los iguales de la red, ya que se utilizan los beneficios de las bases de datos relacionales exclusivamente en el entorno de la aplicación.

- La característica estructurada de la red P2P desarrollada en esta investigación permite garantizar que los recursos que se encuentren disponibles en la red (iguales, archivos, etc) sean encontrados por otros iguales a través del servicio de descubrimiento utilizado y con la ayuda de la implantación de un igual *Rendezvous*.

- El protocolo U2U File Sharing protocol desarrollado en esta investigación brindó una manera de realizar las transacciones necesarias entre un igual actuando como cliente y un igual actuando como servidor en el proceso de compartir un archivo, para soportar el correcto funcionamiento de las transferencias, además de ofrecer un canal de transmisión de datos fiable gracias a que el protocolo se construyó sobre la implementación de JxtaSockets, que proporciona la característica de fiabilidad adquirida. Por otro lado, la implementación del protocolo realizada se hizo para que éste utilizara el mínimo de Bytes necesarios para una transacción entre dos iguales y optimizando así el uso de los recursos en la red P2P.

- Por medio de las pruebas realizadas en las que se compararon la herramienta desarrollada en esta investigación (U2U 1.0) versus el servicio FTP, se pudo comprobar que en la arquitectura cliente – servidor los recursos son inversamente proporcionales al número de clientes, a diferencia de la arquitectura P2P en la cual los recursos van aumentando a medida que más nodos se van añadiendo a la red, a pesar de la sobrecarga que se asume sobre la red por la complejidad del servicio desarrollado y la interacción necesaria entre los nodos de la red utilizándolo.

- Por medio de los resultados obtenidos en las pruebas de comparación de las arquitecturas P2P y cliente – servidor, se puede deducir que con un número de nodos igual o mayor que 10, se puede conseguir un mejor desempeño de la red P2P gracias al sistema de compartición de archivos incompletos implementada y a la distribución de la carga sobre los iguales que conforman la red, contrario a lo que pasa con la arquitectura cliente – servidor, la cual no permite más de 10 conexiones a un servidor FTP común y en donde el recurso de banda de ancha suministrado por el servidor se divide exactamente entre los clientes realizando una petición hacia éste.

- La implementación del mecanismo de integridad utilizando mensajes de resumen SHA-1 permitió que la aplicación U2U 1.0 desarrollada fuese tolerante al envío de datos corruptos por parte de iguales remotos siendo capaz de comprobar que los trozos de un archivo que se está descargando sobre la red contengan los datos exactos que componen dicho archivo, descartando de este modo aquellos pedazos que llegan corruptos con el objetivo de garantizarle al usuario de la aplicación que si la descarga ha finalizado es debido a que se recibieron con éxito los datos necesarios para la reconstrucción de una replica exacta del archivo original. Además, el resumen SHA-1 es el mecanismo con el cual la red encuentra recursos redundantes de un archivo sin importar si el nombre o la descripción de estos son completamente diferentes.

- La característica multihilo del servicio U2U File Sharing Service desarrollado permitió la ejecución concurrente de aquellas tareas que por definición siempre deberían estar disponibles y que en su ejecución no dependieran de otras tareas (manejadores de carga, descarga y el protocolo), evitando bloqueos entre hilos ejecutando diferentes tareas y mejorando los tiempos de respuesta para cada una de las responsabilidades de cada tarea.

- Gracias a la metodología ágil de desarrollo extreme Programming, específicamente al uso estricto de la técnica “Test First Programming”, se aseguró que durante todo el desarrollo de la solución se corroborara el cumplimiento de los objetivos iniciales y el buen funcionamiento de la aplicación desarrollada.

Capitulo 6

RECOMENDACIONES

- La creación de una red P2P implica construir un sistema de comunicación distribuido que requiere tener en cuenta la interacción de varios computadores comportándose al mismo tiempo como clientes y como servidores. Para sacar provecho a ésta ventaja ofrecida por la tecnología y evitar caer en errores de comunicación entre los computadores, deben realizarse varias pruebas que involucren la interacción de varios computadores (*más de 3*) comportándose como clientes y como servidores cuando se ofrece un servicio sobre la red.

- La implementación del protocolo para la transferencia de archivos U2U File Sharing Protocol, incluyó la definición de un parámetro, dentro del mensaje de orden CONN, que indica que los datos transferidos desde un igual actuando como servidor hacia un igual actuando como cliente, se enviarían encripados. Este parámetro está incluido dentro de la especificación del mensaje de orden CONN, pero no fue utilizado en esta versión de la aplicación U2U 1.0. De tal manera que en próximas versiones del aplicativo, es recomendable hacer uso de ésta opción para que el usuario pueda escoger, según el ambiente donde se encuentre si quiere enviar o no cifrados los datos que está compartiendo o descargando.

- El desarrollo de ésta herramienta software brinda una base para la construcción de un aplicativo integrado que incluya los tres ámbitos en los que la arquitectura P2P puede trabajar: Computación distribuida, mensajería instantánea y compartir de archivos.

- Es importante crear una línea de investigación en la cual se realicen trabajos sobre temas relacionados con mensajería instantánea, computación distribuida, nuevas técnicas de intercambio de archivos en la arquitectura P2P y mejoras en los protocolos de JXTA, tecnología en la que se puede incursionar realizando avances significativos que

mejoren las comunicaciones, las búsquedas y la seguridad para las aplicaciones basadas en redes P2P.

- Con el fin de promover el desarrollo de las investigaciones en el campo de las redes P2P, es necesario crear una materia electiva que enseñe fundamentos de programación en este tipo de redes utilizando JXTA y los estudiantes se sientan atraídos por este tipo de tecnologías y propongan sus trabajos de grado en la línea de investigación relacionada.

- Con el objetivo de crear soluciones a problemas relativos al entorno, los investigadores de este proyecto proponen un esquema de licenciamiento libre que elimine los problemas operativos relacionados con el pago de licencias, promoviendo desarrollos en conjunto con otras universidades y empresas a nivel regional y nacional. Una alternativa viable podría ser la licencia Creative Commons reglamentada en el territorio nacional.

Capítulo 6

REFERENCIAS

[1] Proyecto SETI@HOME. Disponible en Internet en: http://setiathome.berkeley.edu/sah_about.php. Fecha de consulta: Mayo 2008.

[2] Proyecto OMEMO. Disponible en Internet en: http://www.omemo.com/es/. Fecha de consulta: Mayo 2008.

[3] CORTELL, Jorge. “En Defensa de las redes P2P”, Disponible en Internet en: http://www.faq-mac.com/mt/archives/007519.php. Marzo de 2004.

[4] Proyecto JXTA. Sun Microsystems. Disponible en Internet en: https://jxta.dev.java.net/. Fecha de consulta: Diciembre de 2007.

[5] CORTELL, Jorge. Ponencia: “Futuro de las redes P2P en entornos corporativos”, Pág.2. Disponible en Internet en: http://homepage.mac.com/jorgecortell/docs/P2Pcorp.pdf. Fecha de consulta: Abril de 2008.

[6] NAPSTER. WIkipedia. Disponible en Internet en: http://es.wikipedia.org/wiki/Napster. Fecha de consulta: Marzo de 2008.

[7] Proyecto Subethaedit. Disponible en internet en: http://codingmonkeys.de/subethaedit/. Fecha de consulta: Marzo de 2008.

[8] BitTorrent. Wikipedia. Disponible en Internet en: http://es.wikipedia.org/wiki/BitTorrent. Fecha de consulta: Agosto de 2008.

[9] CALLE PEÑA, Xavier , CADEÑO MIELES, Vanessa, ABAD ROBALINO, Cristina. “Diseño y desarrollo de una aplicación P2P de mensajería para la ESPOL usando tecnología JXTA”. Disponible en Internet en: http://www.rte.espol.edu.ec/archivos/Revista_2006/99Final.pdf. Octubre de 2006.

[10] MARTÍNEZ, Francisco. CAMACHO, Ricardo. GARCÍA, Luis. CAICEDO, Oscar. HURTADO, Javier. “JXME: Una plataforma robusta para el desarrollo de aplicaciones P2P en dispositivos móviles”. Revista de la División de Ingenierías de la Universidad del Norte, Edición No. 20. Disponible en Internet en: http://dialnet.unirioja.es/servlet/articulo?-codigo=2506178. Julio-Diciembre de 2006.

[11] POMARES QUIMBAYA, Alexandra y MORALES CHAVARRO, Javier Mauricio. “PASTEUR: Un sistema para indexación y búsqueda de componentes de software”. Memorias de la Conferencia Latinoamericana de Alto Rendimiento, CLCAR 2007, Santa Marta, Colombia. Agosto 2007

[12] P2P. Wikipedia. Disponible en Internet en: http://es.wikipedia.org/wiki/Caracteristicas_p2p. Febrero de 2008.

[13] SING LI. “Making P2P interoperable: Creating JXTA systems”. Disponible en Internet en: http://www.ibm.com/developerworks/java/library/j-p2pint3/. 2002.

[14] JXTA. Wikipedia. Disponible en Internet en: http://en.wikipedia.org/wiki/JXTA. Fecha de consulta: Febrero de 2008.

[15] Sun Microsystems, “JXTA JAVA™ STANDAR EDITION V.2.5. Programmers Guide”, Septiembre 10 de 2007.

[16] Sun Microsystems. JXTA Technical Training. Diciembre de 2002.

[17] Proyecto Shell JXTA. Sun Microsystems. Disponible en Internet en: https://jxse-shell.dev.java.net/. Fecha de consulta: Diciembre de 2007.

[18] Sun Microsystems. “Shared Resource Distributed Index (SRDI)”, JXTA JavaTM Standard Edition v2.5 Programmers Guide, September 10th, Página. 22. 2007.

[19] CORTIZO, Jose. eXtreme Programming. Disponible en Internet en: http://www.esp.uem.es/jccortizo/xp.pdf. Fecha de consulta: Marzo de 2008.

[20] STALLINGS, William. Sistemas Operativos, Editorial Prentice Hall. 2003

[21] Sun Microsystems. Tutorial “Writen Event Listeners”. Disponible en Internet en: http://java.sun.com/docs/books/tutorial/uiswing/events/-index.html. Fecha de consulta: Agosto de 2008.

[22] Sun Microsystems. Swing Application Framework. Disponible en internet en: https://appframework.dev.java.net/. Fecha de consulta: Diciembre de 2007.

[23] Advanced Encryption Standard. Wikipedia. Disponible en Internet en: http://es.wikipedia.org/wiki/Advanced_Encryption_Standard. 2008

[24] SEO, Jin-Wook. KIM, Dong-Kyun. KIM, Hyun-Chul. CHUNG, Jin-Wook. “The Algorithm of Sharing Incomplete Data in Decentralized P2P”. IJCSNS Internacional Journal of Computer Science and Network Security, VOL 7 No. 8, Agosto de 2007. Disponible en Internet en: http://paper.ijcsns.org/07_book/200708/20070821.pdf.

[25] WILLIAM STALLINGS. “Autenticación de mensajes y funciones de mezcla (HASH)”. Comunicaciones y redes de computadores. Quinta Edición. 1997.

[26] CAY S. HORSTMANN, GARY CORNELL”. Core Java 2 Características Avnzadas. Séptima Edición. Editorial Sun Microsystems. 2006

[27] Bouncy Castle Crypto APIs. Disponible en Internet en: http://www.bouncycastle.org/. Fecha de consulta: Diciembre de 2007.

[28] Sun Microsystems. Documentación Framework de seguridad de Java. Disponible en internet en: http://java.sun.com/j2se/-1.3/docs/api/java/security/package-summary.html. Fecha de consulta Junio de 2008.

[29] HUNTER, David. RAFTER, Jeff. FAWCETT, Joe. VAN DER VLIST, Eric, AYERS, Danny, DUCKETT,Jon. WATT, Andrew. MCKINNON, Linda. “XML Schemas”. Beginning XML. Cuarta Edición. Editorial Wrox. 2007

[30] Sun Microsystems. Documentación del API Share de JXTA. Disponible en internet en: https://jxta-cad.dev.java.net/api/overview-tree.html. Fecha de consulta: Julio de 2008.

[31] ZIKOPOULOS, Paul. SCOTT, Dan. BAKLARZ, George. Apache Derby – Off to the races. Editorial IBM PRESS. 2005

[32] BOOCH, Grady, RUMBAUGH, James, JACOBSON, Ivar. El Lenguaje Unificado de Modelado. Editorial PEARSON Addison Wesley. 2005.

[33] COULOURIS, George. DOLLIMORE, Jean. DORMIDO, Sebastián KINDBERG, Tim. Sistemas Distribuidos. Tercera Edición. Editorial PEARSON Addison Wesley. Pag. 205. 2001.

ANEXO A.
COMANDOS DEL SHELL JXTA

Comando	Descripción
man	Ayuda principal del sistema para el Shell de JXTA. Si se escribe el comando man sin parámetros lista todos los comandos del Shell JXTA acompañados de una pequeña explicación.
cat	Muestra el contenido de una variable de entorno del Shell JXTA.
chpgrp	Cambia el grupo de iguales por defecto por el especificado.
clear	Limpia la pantalla del Shell JXTA.
env	Muestra en la pantalla todas las variables de entorno definidas en el Shell de JXTA.
exit	Termina la ejecución del Shell JXTA
exportfile	Exporta una variable del Shell dentro de un archivo externo
flush	Borra un anuncio JXTA
get	Obtiene datos desde un mensaje de pipe
grep	Busca patrones iguales
groups	Descubre y lista los grupos de iguales
help	Ayuda sobre los comandos del Shell JXTA
importfile	Importa un archivo externo dentro de una variable del Shell JXTA
info	Muestra información de un anuncio
instjar	Instala archivos jar conteniendo comandos del Shell adicionales
join	Instancia e ingresa a un grupo de iguales
leave	Abandona un grupo de iguales
login	Autentica con el Membership Service del grupo
mem	Muestra información de la memoria
mkadv	Crea un nuevo anuncio desde un documento almacenado en una variable de entorno del Shell JXTA
mkmsg	Crea un nuevo mensaje de pipe
mkpipe	Crea una nueva pipe
newpgrp	Crea un nuevo anuncio de grupo de iguales
newmodulespec	Crea un nuevo anuncio Module Class
newmoduleclass	Crea un nuevo anuncio Module Class
newpipe	Crea un nuevo anuncio de pipe
peerconfig	Forza reconfigurar el igual
peerinfo	Obtiene información sobre iguales
peers	Descubre y lista los iguales en la red
pse.certs	Muestra los certificados contenidos en el Membership PSE actual del Igual

pse.createkey	Crea una clave en el almacén de claves del PSE
pse.dumpcred	Elimina un certificado
pse.dupkey	Crea una clave en el almacén de claves del PSE
pse.erase	Borra una clave o un certificado del almacén de claves PSE
pse.importcert	Importa una cadena de un certificado verdadero
pse.keys	Muestra las claves contenidas en el PSE Membership del actual grupo de iguales
pse.newcsr	Genera un certificado firmado del documento solicitado
pse.signcsr	Firma un certificado solicitando firma
pse.status	Muestra información de estado del PSE Membership del grupo
publish	Publica un anuncio JXTA
put	Coloca datos dentro de un mensaje
rdvcontrol	Controla el comportamiento del servicio Rendezvous
rdvstatus	Muestra información sobre el servicio Rendezvous
recv	Recibe un mensaje desde un pipe
relaystatus	Muestra una lista de Relays y clientes conectados a éste igual
remotepublish	Publica remotamente un anuncio JXTA
route	Muestra información de ruta de un igual
rsh	Conecta a un Shell JXTA remoto
rshd	Demonio del Shell JXTA remoto
search	Descubre anuncios JXTA
send	Envía un mensaje dentro de un pipe
set	Modifica una variable de entorno
sftp	Envía un archivo a otro igual
share	Comparte un anuncio
slepp	Duerme por una cantidad de milisegundos especificada
storehome	Muestra la localización de la carpeta de jxta
talk	Habla con otro igual
transports	Muestra información sobre los transportes de mensajes disponibles en el actual grupo
uninstjar	Desinstala archivos jar previamente instalados con el comando instjar
unset	Remueve una variable de entorno
version	Muestra la versión del Shell de ésta instancia del Shell
wc	Cuenta el numero de líneas, palabras y caracteres de un objeto
who	Muestra información de credencial
whoami	Muestra información sobre este igual o el actual grupo de iguales
xfer	Envía un archivo hacia otro igual

ANEXO B.

METODOLOGIA DE DESARROLLO EXTREME PROGRAMMING.

La metodología que se utilizó en el desarrollo de la herramienta software (U2U 1.0) es la metodología ligera de desarrollo de software eXtreme Programming o XP creada por Kent Beck[15] , autor del primer libro sobre la materia, Extreme Programming Explained: Embrace Change (1999).

La Programación Extrema es una metodología ligera de desarrollo de software que se basa en la simplicidad, la comunicación y la realimentación o reutilización del código desarrollado. La metodología XP se basa en:

1. Pruebas Unitarias: Las pruebas realizadas a los principales procesos, de tal manera que adelantándose en algo hacia el futuro, se pueda hacer pruebas de las fallas que pudieran ocurrir.

Para la realización de las pruebas unitarias se utilizó el framework para pruebas JUnit[16]. Gracias a éste framework, se pudo llevar a cabo las pruebas de aceptación de cada una de las clases implementadas en el proyecto, y así cumplir con uno de los objetivos de la fase de planeación de la metodología XP.

En la siguiente tabla se listan algunas de las clases de prueba creadas para verificar un funcionamiento de la base de datos U2Uclient, el Shell U2U, el servicio U2UFSS o del protocolo U2UFSP.

Clase de prueba	Prueba
ShellEnvTest	Verificar el acceso a las variables de entorno del Shell, la creación de nuevas variables y la obtenciónde las variables creadas
ShellTest	Verifica el correcto funcionamiento del Shell: inicialización, ejecución de comandos, resultados y finalización.
RequestModuleTest	Comprueba que el módulo de solicitud del servicio U2UFSS realice adecuadamente las solicitudes de archivos a través del shell y el servicio DiscoveryService de JXTA.
U2UContentManagerTest	Comprueba que la solicitud de publicación de un anuncio en la red P2P se haga de manera satisfactoria, y que los archivos que sean descargados y compartidos se almacenen correctamente en la base de datos U2Uclient.
U2UDownloadingManagerTest	Verifica que las instancias creadas para manejar las descargas tengan el comportamiento adecuado: Manejen la descarga de un archivo específico, asignen tareas al protocolo y mantengan la comunicación con otros iguales a través de los hilos de conexión y el protocolo U2UFSP.

[15] http://www.extremeprogramming.org/

[16] http://www.junit.org/home

Las pruebas de aceptación con JUnit se implementan un Test JUnit que por defecto está conformado por los métodos *setUpClass()*, *tearDownClass()*, *setUp()* y *tearDownn()* y las anotaciones[17] *@BeforeClass, @AfterClass, @After y @Before*. Estos cuatro métodos y anotaciones básicas se ejecutan antes de iniciar y después de finalizar el la clase que compone el Test JUnit , y antes y después de cada método de prueba. Los métodos de prueba son creados por los desarrolladores de software para crear pequeñas pruebas que comprueben que los resultados de una tarea son los deseados. Los métodos de prueba con JUnit se identifican con anotaciones de tipo *@Test* que se ubican por encima del método de prueba.

Dentro de los métodos de prueba se utilizan las afirmaciones[18] (*assertions*) del lenguaje de programación Java. Las afirmaciones se utilizan para verificar el resultado de una operación y de un Test. Algunas de las afirmaciones más utilizadas en JUnit son: *assertEquals, assertTrue, assertFalse, assertNotNull* y *fail().* Cuando todas las afirmaciones dentro de un test se cumplen, la prueba es exitosa, de lo contrario, la prueba ha fallado. En la **Figura A** se puede ver la estructura básica de una clase de prueba para una clase existente.

```
public class u2ufssTest {

    public u2ufssTest() {
    }

    @BeforeClass
    public static void setUpClass() throws Exception {
    }

    @AfterClass
    public static void tearDownClass() throws Exception {
    }

    @Before
    public void setUp() {
    }

    @After
    public void tearDown() {
    }

    /**
     * Test of startApp method, of class u2ufss.
     */
    @Test
    public void testStartApp() {...}

    /**
     * Test of help method, of class u2ufss.
     */
    @Test
    public void testHelp() {...}

    /**
     * Test of getDescription method, of class u2ufss.
     */
    @Test
    public void testGetDescription() {...}
```

Figura A. *Clase de prueba creada por un Test JUnit*

2. Refabricación: se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.

[17] http://today.java.net/pub/a/today/2006/12/07/junit-reloaded.html
[18] http://java.sun.com/j2se/1.4.2/docs/guide/lang/assert.html

La refrabricación se utilizó para hacer el código fuente más especializado, fácil de entender y reutilizar.

El IDE utilizado para desarrollar el proyecto fue NetBeans 6.5. Este IDE proporciona las herramientas necesarias para efectuar técnicas de refrabricación o *refactoring*[19].

Las técnicas de *refactoring* empleadas incluyeron especialización de clases, conversión de bloques de código en métodos, especialización de métodos, creación de campos de clase, entre otras.

En la **Figura B** se puede observar una imagen del menú ofrecido por el IDE Netbeans para realizar técnicas de refactorización sobre el código fuente.

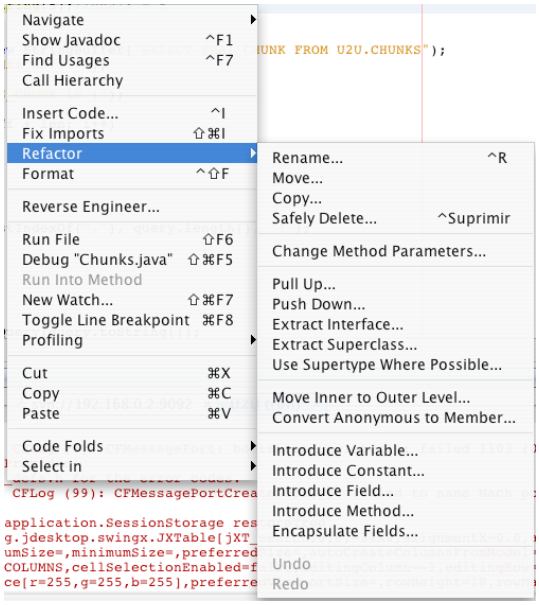


Figura B. Menú REFACTOR del IDE Netbeans, que facilita las técnicas de refactorización

3. Programación en pares: una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento.

La programación en pares se llevo a cabo en la fase de desarrollo y la fase de pruebas de cada una de las iteraciones implementadas. Se pudo observar un avance significativo al interactuar los dos programadores (*Ver* **Tabla 1**) en un

[19] http://wiki.netbeans.org/Refactoring

mismo computador intercambiando ideas para el desarrollo de la herramienta P2P. Esto no sólo consiguió desarrollar el aplicativo software sino también compartir pensamientos, ideas, aclarar conceptos, conseguir una realimentación mutua y sobretodo, aprender a trabajar en grupo confiando en las capacidades del compañero y creciendo juntos.

Paquete org.u2u.filesharing		
CLASE	**LDC Aprox.**	**TIEMPO Aprox.**
U2UConnectionHandler	237	1 día
U2UContentAdvertisementImpl	410	2 días
CLASE	**LDC Aprox.**	**TIEMPO Aprox.**
U2UContentIdImpl	263	2 días
U2UFileContentImpl	74	½ día
U2UFileSharingService	759	30 días
U2UFileSharingServiceEvent	45	½ día
U2UFileSharingServiceListener	54	½ día
TOTAL	**1842**	**35 días**
Paquete org.u2u.filesharing.downloadpeer		
CLASE	**LDC Aprox.**	**TIEMPO Aprox.**
U2UContentDiscoveryEvent	68	½ día
U2UDMExecutorsTask	139	2 días
U2UDMThinker	770	7 días
U2UDMThinkersProblem	102	1 día
U2UDownloadingManager	1294	15 días
U2URequestManagerImpl	233	2 días
U2USearchListener	54	½ día
TOTAL	**2660**	**28 días**
Paquete org.u2u.filesharing.uploadpeer		
CLASE	**LDC Aprox.**	**TIEMPO Aprox.**
U2UContentManagerImpl	424	5 días
U2UUMExecutorsTask	96	1 días
U2UUMThinker	319	3 días
U2UUMThinkersProblem	75	1 día
U2UUploadingManager	780	15 días
TOTAL	**1694**	**25 días**
Paquete org.u2u.filesharing.fsprotocol		
CLASE	**LDC Aprox.**	**TIEMPO Aprox.**
U2UFSPOrderConnection	96	½ día
U2UFSPOrderFileInfo	85	½ día
U2UFSPOrderGetChunk	79	½ día
U2UFSPOrderQuit	52	½ día
U2UFSPResponseFileInfo	483	5 días
U2UFSPResponseGetChunk	133	2 días
U2UFSProtocolOrder	511	5 días
U2UFSProtocolResponse	656	6 días
U2UFileSharingProtocol	1356	15 días
U2UFileSharingProtocolEvent	87	1 día
U2UFileSharingProtocolListener	52	1 día
TOTAL	**3590**	**37 días**
Paquete net.jxta.impl.shell.bin.u2ufss		

CLASE	LDC Aprox.	TIEMPO Aprox.
U2ufss	766	6 días
Paquete org.u2u.common.db		
CLASE	**LDC Aprox.**	**TIEMPO Aprox.**
Address	220	2 días
Chunks	1007	10 días
Chunks_Add	232	2 días
SharedFiles	619	6 días
ConnectTo	478	5 días
TOTAL	**2556**	**25 días**

Tabla 1. Líneas De Código – Programación En Pares
LDC Totales: 12.345. En un tiempo apróximado de 5 meses [*]

HERRAMIENTAS UTILIZADAS PARA EL DESARROLLO DE XP

Tres herramientas fueron principalmente las utilizadas en conjunto con el desarrollo de la metodología XP :

1. NETBEANS

El IDE (*Integrated Developer Enviroment*) NetBeans[20] es una herramienta que permite el desarrollo de aplicaciones java para cualquiera de las arquitecturas software para las que el lenguaje está diseñado: J2EE, J2SE y J2ME. En este caso, se utilizó NetBeans versión 6.1 y 6.5, para el desarrollo de una herramienta J2SE basada en redes P2P.

Éste IDE permitió la creación de un diseño totalmente orientados a objetos compuesto por clases, métodos y objetos para implementar todas las funciones ofrecidas por el Shell U2U, el servicio para la transferencia de archivos U2UFSS, el protocolo U2UFSP y la interfaz gráfica del usuario. Además, NetBeans permite el acceso directo al sistema de control de versiones SUBVERSION utilizando un módulo *(plugin*) especializado para ésta funcionalidad, permite el acceso directo a la base de datos embebida de DERBY, habilita al desarrollador de un mecanismo para monitorear una aplicación a través de la función PROFILE permitiendo observar el consumo de CPU, la actividad de los hilos, la cantidad de memoria y el tiempo de ejecución de los métodos y facilita las técnicas de refactorización, entre otras funciones.
Otra característica importante de NetBeans es multiplataforma, lo que permitió a los desarrolladores utilizar esta herramienta en diferentes sistemas operativos

[*] Los datos presentados en la tabla 1 no contemplan las LDC creadas para la GUI del software U2U.
[20] http://www.netbeans.org/

para depurar (*debugging*) el software desarrollado cuando las circunstancias lo requirieron[21].

MANTIS

El sistema para seguimiento de errores (*bugtraking*) MANTIS[22], es un proyecto basado en Web que está diseñado para mantener un control de los errores que se producen durante el desarrollo de un software.

En el desarrollo del proyecto la herramienta MANTIS no fue solo utilizada para registrar errores que eventualmente sucedían sobre el software, sino que fue utilizada principalmente para registrar las historias de usuario (HU) que representaban los requerimientos del software y que son la base del desarrollo con la metodología XP.

MANTIS permitió ingresar cada una de las HU de la herramienta software y adicionalmente proporcionó una manera fácil y rápida de llevar un control sobre el desarrollo de los requisitos representados en cada HU. Además, ésta herramienta permite la asignación de responsabilidades entre los integrantes del proyecto, agregar notas a una HU, mirar el avance de una HU, cuántas HU han sido terminadas, cuántas necesitan ser revisadas o no han sido revisadas por el respectivo integrante, etc.

MANTIS fue configurado para que sólo los integrantes del proyecto pudieran ingresar a la herramienta y actualizaran o publicaran HU o errores relacionados con el desarrollo del proyecto.

En la **Figura C** se puede observar un imagen de la herramienta MANTIS que muestra algunas de las HU ingresadas junto con el identificador, el modulo del proyecto al que pertenece, el estado, la fecha de actualización, el responsable y la descripción de la HU.

[21] Las pruebas de funcionamiento del aplicativo U2U fueron realizadas en salas de computo con sistemas operativos: FEDORA y WINDOWS XP, y el software fue desarrollado en computadores con sistema operativo MAC OS X TIGER y LEOPARD.
[22] http://www.mantisbt.org/

			0000054		[Shell and Services] App	trivial	asignada (irenelizeth)	**2009-01-13**	Implementar la política de seguridad para la red P2P
			0000050		[Shell and Services] Algoritmos	funcionalidad	resuelta (irenelizeth)	**2009-01-13**	Definir los mensajes de respuesta y orden del protocolo U2UFSP
			0000053		[AgathaP2P] GUI	**mayor**	asignada (irenelizeth)	**2009-01-13**	Diseño para crear, unirse y abandonar un grupo de iguales
			0000052		[GUI] GUI	trivial	asignada (irenelizeth)	**2009-01-13**	Diseño e implementación de la GUI para ver las descargas realizadas
			0000051		[GUI] GUI	trivial	asignada (irenelizeth)	**2009-01-13**	Realizar el diseño del panel para compartir archivos dentro de la red P2P
			0000046	1	[GUI] GUI	funcionalidad	se necesitan más datos (irenelizeth)	**2009-01-13**	Interfaz Gráfica para la configuración del igual dentro de la red
			0000037	1	[Shell and Services] GUI	menor	se necesitan más datos (irenelizeth)	**2009-01-13**	Mostrar la informacioión de los iguales que se encuentran dentro de la red U2U4U
			0000049		[Shell and Services] App	trivial	nueva	2008-12-31	Realizar la implementación de la parte del Servidor para el Protocolo U2UFSP (Parte de carga de archivos de un igual)
			0000047		[Shell and Services] App	trivial	nueva	2008-12-31	Realizar la implementación de la parte del cliente para el Protocolo U2UFSP (Parte de descarga de archivos de un igual)
			0000039	4	[Shell and Services] App	funcionalidad	asignada (spino327)	2008-12-31	UploadPeer: Realizar la implementación del código necesario para PUBLICAR los archivos comparidos por un igual
			0000045	2	[Shell and Services] App	funcionalidad	asignada (spino327)	2008-12-30	Diseño y Desarrollo del U2UFileSharing Protocol
			0000034	2	[Shell and Services] Shell	trivial	aceptada	2008-11-28	Realizar la implementación del servicio para manejar los archivos compartidos por la red U2U4U, facetas: UploadPeer,DownloadPeer
			0000044		[Test] App	menor	resuelta (spino327)	2008-11-28	Codificación ejemplo JxtaSocket

Figura C. Historias de Usuario del proyecto ingresadas en MANTIS

SUBVERSION

El sistema para control de versiones SUBVERSIÓN fue utilizado principalmente para almacenar continuamente los cambios que se realizarán sobre el software desarrollado (versiones), tener un respaldo de seguridad del proyecto y actualizar los cambios realizados por los programadores desde diferentes estaciones de trabajo sobre el mismo proyecto, permitiendo tener siempre actualizado el código fuente.

SUBVERSION es un proyecto OpenSource y pudo ser integrado al IDE Netbeans a través de un módulo para facilitar su utilización.

ANEXO C.
MANUAL DE USUARIO

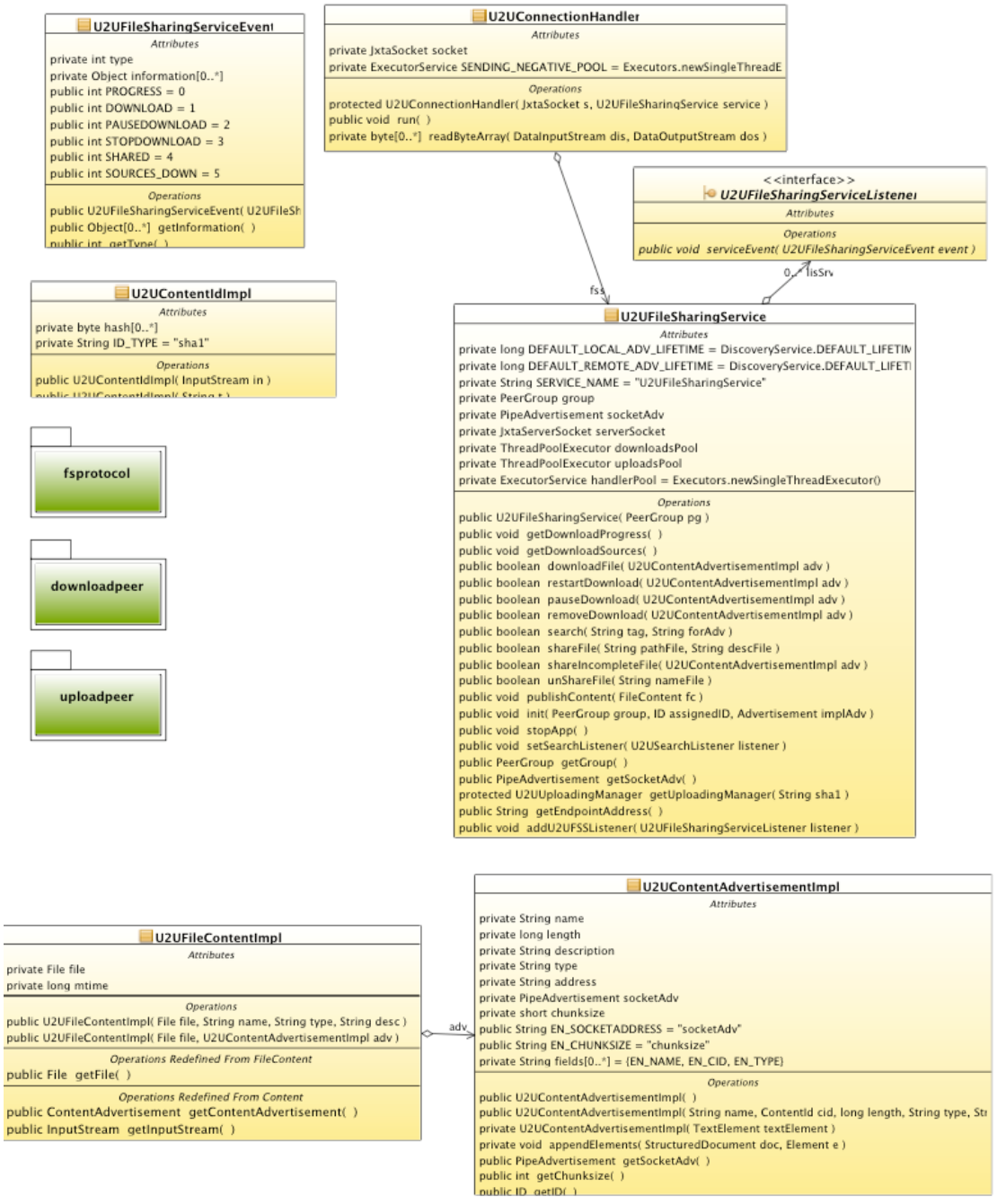
Ver CD.

ANEXO D.

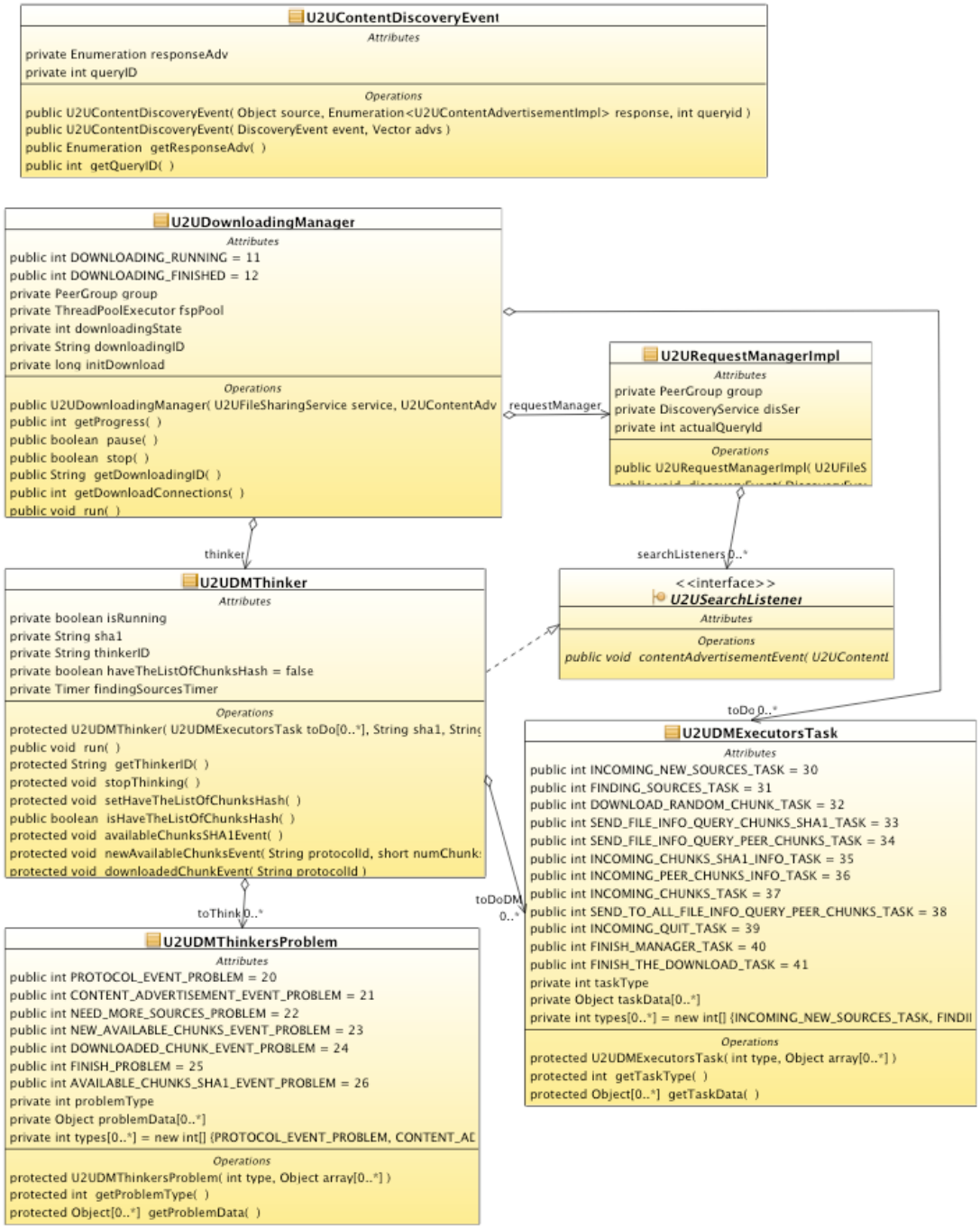
DIAGRAMAS DE CLASES

1. U2UShell

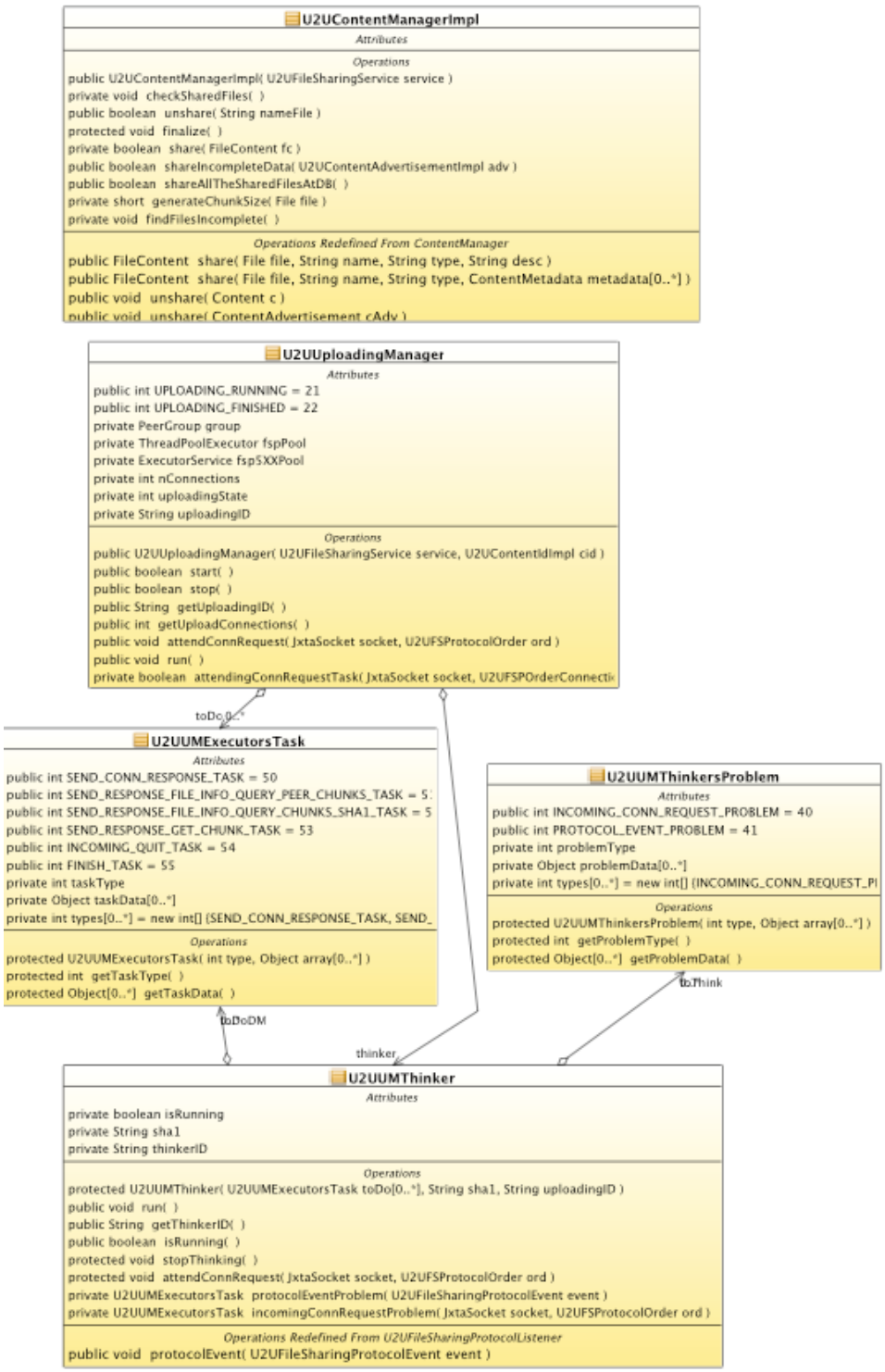
- Paquete org.u2u.filesharing



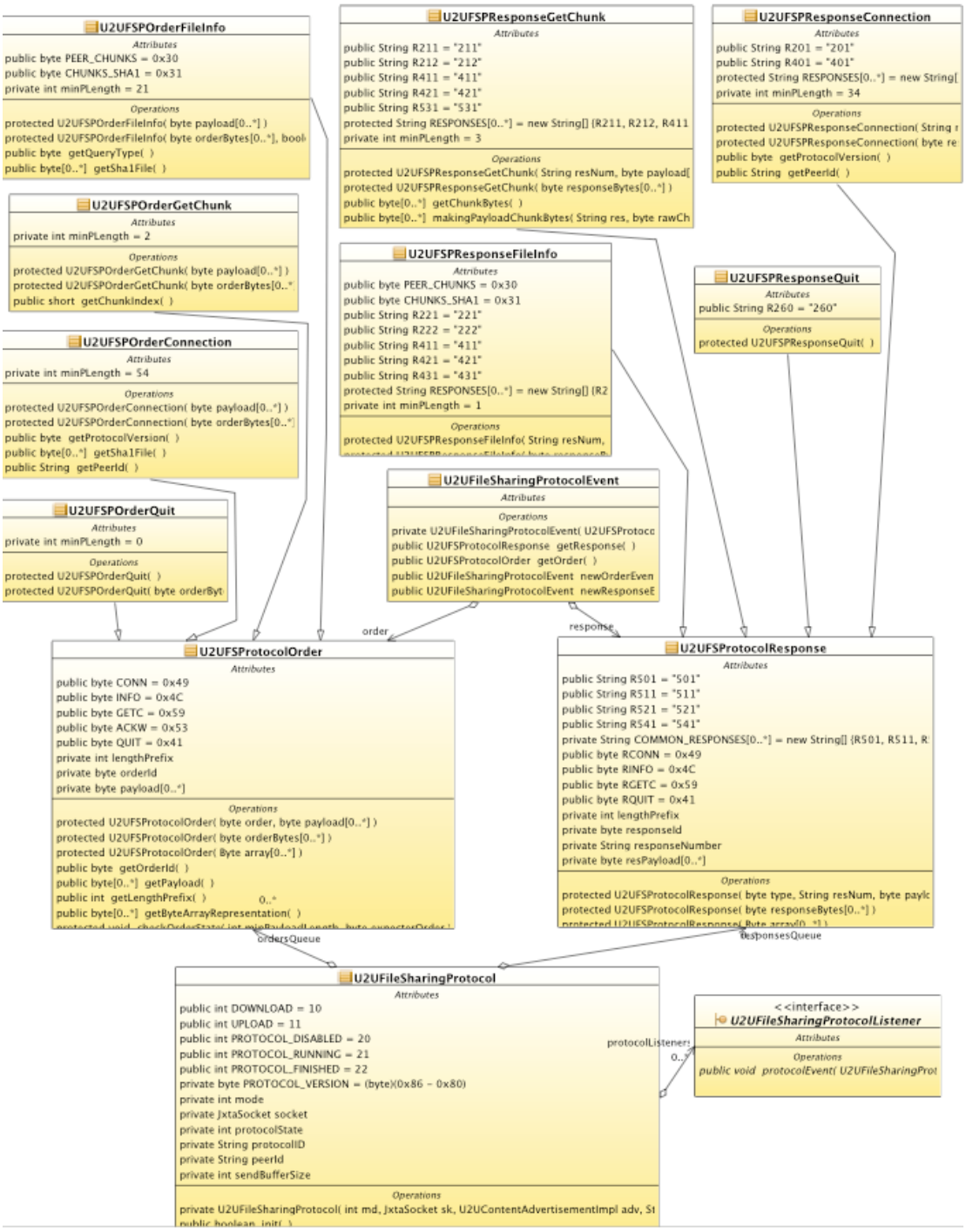
- Paquete org.u2u.filesharing.downloadpeer



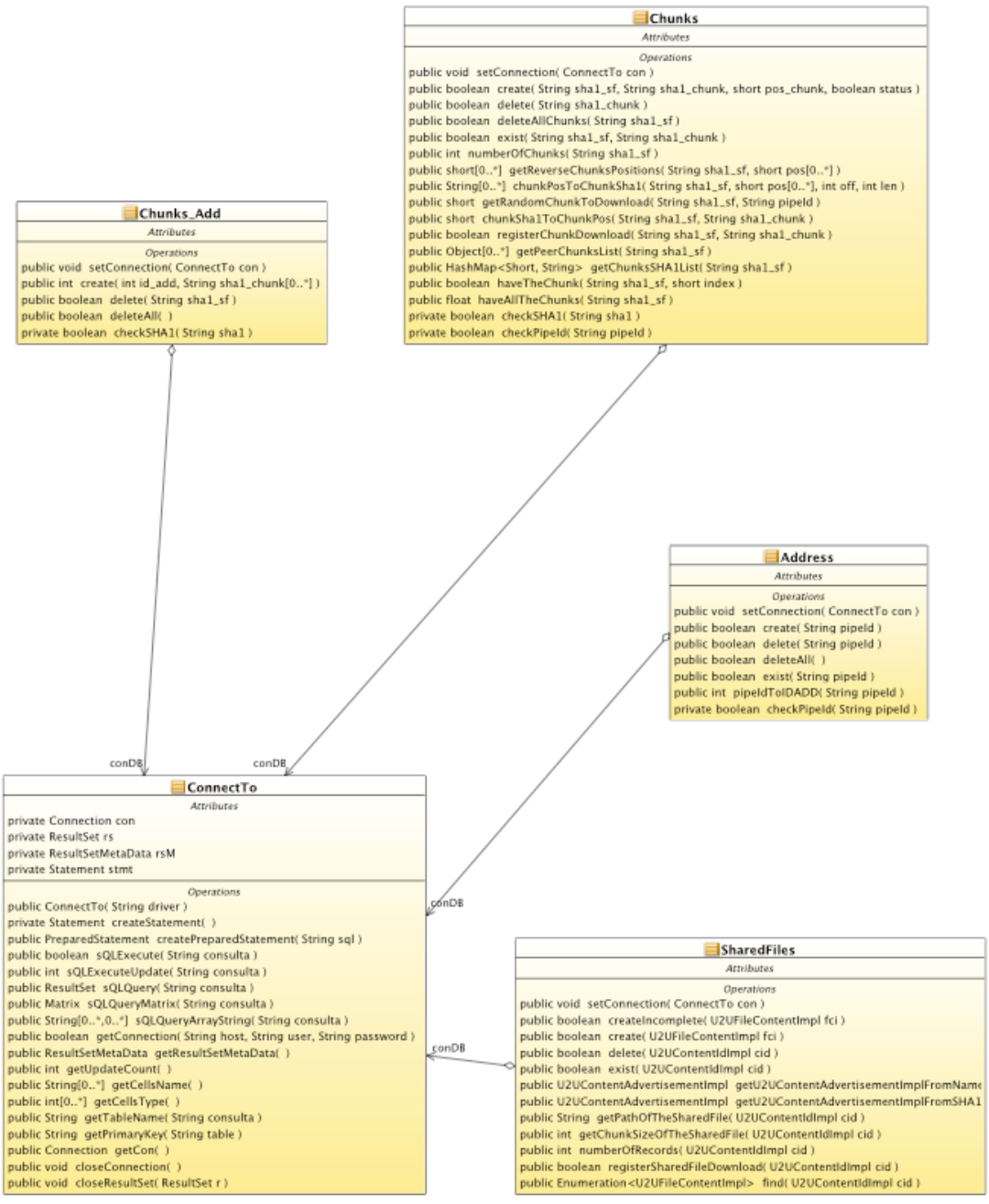
- Paquete org.u2u.filesharing.uploadpeer



- Paquete org.u2u.filesharing.fsprotocol



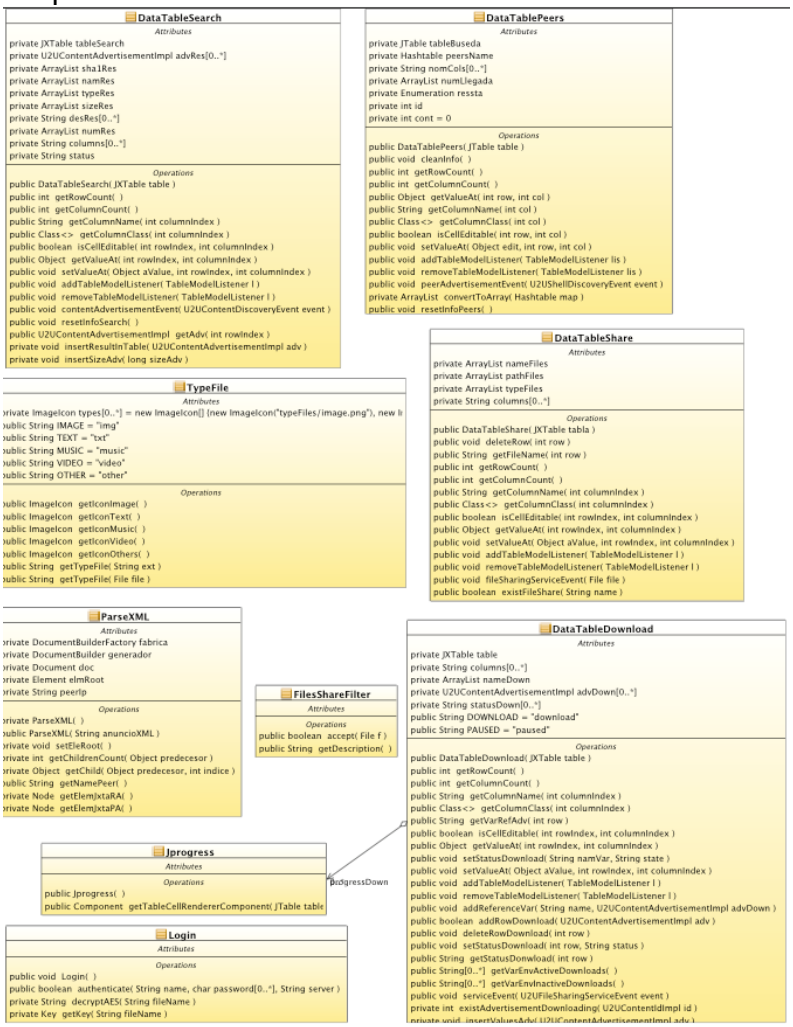
- Paquete org.u2u.commons.db



- Paquete net.jxta.impl.shell.bin.u2ufss

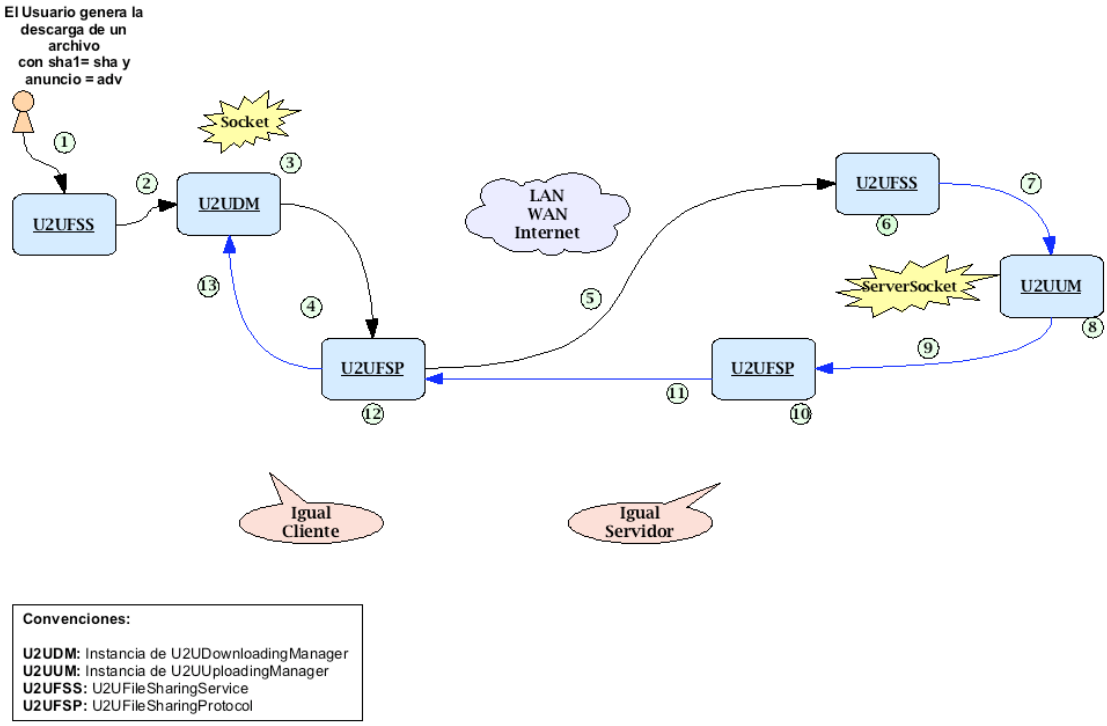
2. GUI

- Paquete data



ANEXO E.

PROCESO DE ENVÍO DE UN MENSAJE DE ORDEN CONN A TRAVÉS DEL PROTOCOLO U2U FILE SHARING PROTOCOL.



1. Se crea una instancia del servicio U2UFSS si no hay una creada.
2. Se crea una instancia del U2UDM (corre en un hilo) que será la encargada de gestionar la descarga y a la cual se le pasa el anuncio (*adv*). Si la instancia que maneja la descarga actual existe (se verifica dentro del U2UDM), se ignora la petición de descarga, de lo contrario, se genera un mensaje de aviso que informa que la descarga ya se está realizando.
3. El U2UDM crea un socket con uno de los iguales que poseen el anuncio (*adv*).
4. El U2UDM crea un U2UFSP dentro de un hilo (el hilo es generado por medio de la arquitectura de hilos por conexión) y se le pasa la información del socket creado en el paso3 y el anuncio (*adv*) para que dialogue con el U2UFSP del igual remoto (actuando como igual servidor).

5. El U2UFSP envía un mensaje de orden de tipo CONN hacia el igual remoto con el cual se creó el socket para saber si puede atender la solicitud de descarga de trozos (*chunks*) del archivo.
6. El U2UFSS recibe el mensaje de orden y comienza el análisis de la solicitud.
7. Averigua si hay alguna instancia U2UUM que esté actualmente manejando el archivo pedido y le pasa el socket y el mensaje de orden. Si no hay una instancia del U2UUM creada para éste archivo se crea una nueva instancia siempre y cuando no se encuentren creadas el número máximo de instancias U2UUM (valor definido por el usuario). Si la instancia U2UUM pudo ser creada se le pasa el socket y la orden de mensaje. Si la instancia del U2UUM no pudo ser creada, se crea una nueva instancia del protocolo para ser enviada hacia el U2UFSP del cliente a través de un mensaje de respuesta 521.
8. La instancia U2UUM toma una decisión respecto al mensaje de orden CONN según la carga que tenga ésta instancia (cuántos hilos de conexión esta atendiendo). Si la decisión es negativa debido a que el igual servidor se encuentre ocupado, el igual cliente es almacenado en una cola de espera.
9. La instancia U2UUM crea una instancia U2UFSP con el U2UFSP con la información del socket y la decisión tomada por el igual servidor para que dialogue con el U2UFSP del igual cliente.
10. Las instancia del U2UFSP crea un mensaje en base a la decisión recibida por parte del U2UUM. Los posibles mensajes de respuesta pueden ser de tipo: 201, 401 o 501.
11. El U2UFSP del servidor envía el mensaje de respuesta.
12. El U2UFSP del cliente recibe el mensaje de respuesta.
13. El mensaje de respuesta es enviado hacia el U2UDMThinker del U2UDM quién analiza que mensaje de respuesta de orden enviar y así continuar con el proceso de descarga del archivo.

ANEXO F.

CARACTERÍSTICAS DE LOS EQUIPOS Y DEL ARCHIVO UTILIZADOS PARA LAS PRUEBAS DE TRANSMISIÓN DE ARCHIVOS ENTRE COMPUTADORES.

CARACTERÍSTICAS DE LOS COMPUTADORES	
Sistema operativo:	Windows XP Professional Versión 2002. Service Pack 3
Disco Duro:	80 GB
RAM:	512 MB
Procesador:	PENTIUM 4 CPU 3,20 GHZ y 3,19 GHZ
CARACTERÍSTICAS DE LA RED	
Tarjeta de Red:	FAST ETHERNET 100Mbps
Tipo de Red:	PENTIUM 4 CPU 3,20 GHZ y 3,19 GHZ

Tabla 1. Características de los computadores, y de la red utilizada para las pruebas.

Los computadores utilizados para la realización de las pruebas fueron los ubicados en la salas 2-7, 3-1 y 3-11 del Centro de Tecnologías de Información y Comunicación (CENTIC) de la Universidad Industrial de Santander, UIS.

CARACTERÍSTICAS DEL ARCHIVO TRANSMITIDO	
Nombre:	cogitoWireless.toast
Tipo:	Imagen de disco de la aplicación TOAST TITANIUM del SO MAC OS X
Tamaño:	434376768 Bytes (414 MB)
SHA1:	fdd2aee439ec97fdd90224fd555b9701ec9289ed
Licencia de contenido:	Creative Commons
Contenido:	Información acerca de redes inalámbricas comunitarias (videos, documentos y audio)

Tabla 2. Características del archivo transmitido entre los computadores de la red LAN en las pruebas de transmisión realizadas.

ANEXO G.

PONENCIA "SOFTWARE PARA EL MEJORAMIENTO EN LA TRANSMISION DE DATOS ENTRE COMPUTADORES UTILIZANDO REDES P2P".

SOFTWARE PARA EL MEJORAMIENTO EN LA TRANSMISION DE DATOS ENTRE COMPUTADORES UTILIZANDO REDES PEER-TO-PEER

- Presentación: Oral
- Tópico: Computación de alto rendimiento
- Autores: **Sergio Antonio Pino Gallardo** spino327@gmail.com
  Irene Lizeth Manotas Gutierrez irenelizeth@gmail.com
  Henry Argüello Fuentes henarfu@uis.edu.co
- Institución: Universidad Industrial de Santander
- Teléfono: 3157829602
- Fecha: 22 de Agosto de 2008

Resumen:
A través de los años Internet se ha ido posicionando como la mejor herramienta a la que podemos acceder los humanos para transferir y recibir información, como alternativa a los modelos centralizados en donde existe una relación inversamente proporcional entre la disponibilidad de los recursos y el número de clientes accediendo a estos y en donde la escalabilidad de las soluciones representa inversiones en equipos más potentes y ancho de banda, se plantea crear una red Peer-to-Peer (P2P) con el objetivo de descentralizar los recursos y la comunicación a un bajo coste de adquisición y con la posibilidad de alcanzar una mayor disponibilidad de los recursos en la red y una transferencia óptima de los recursos disponibles.
Con el presente trabajo se pretende crear un software que soportado bajo la infraestructura de una red P2P, presente una alternativa que logre mejorar las transferencias de archivos entre los nodos que conformen la red por medio de algoritmos computacionales que tengan en cuenta las características de la red P2P y aprovechen los recursos disponibles logrando una transferencia eficiente de los archivos.
El desarrollo e implementación de este proyecto utiliza como principal característica una red User To User , donde los nodos que acceden al sistema son parte de una comunidad que comparte los mismos intereses y que se une para compartir con confianza. Además, esta característica permite a los miembros de una misma comunidad crear y prestar servicios a sus miembros de manera redundante, ya que cada participante puede actuar como consumidor o proveedor de un servicio.
Para la realización del presente trabajo se utiliza el Framework JXTA de Sun Microsystems como soporte para el desarrollo de la red P2P, el cual ha sido utilizado en compañías como BOEING para la realización de proyectos basados en P2P.


SIIC 2008
SEMINARIO INTERNACIONAL DE
INFORMÁTICA
&COMPUTACIÓN
UNIVERSIDAD PONTIFICIA BOLIVARIANA
Universidad
Pontificia
Bolivariana
SECCIONAL BUCARAMANGA
Bucaramanga, 16 al 18 de Septiembre
SW PARA EL MEJORAMIENTO EN LA TRANSMISIÓN DE DATOS ENTRE COMPUTADORES UTILIZANDO REDES P2P.
SERGIO ANTONIO PINO GALLARDO
BUCARAMANGA, SEPTIEMBRE DE 2008
Silverlight

ANEXO H.

ARQUITECTURA EN CAPAS PARA EL DESARROLLO DE UNA APLICACIÓN BASADA EN REDES PEER-TO-PEER

Presentado en la revista GTI volumen 7 No. 19

SERGIO ANTONIO PINO GALLARDO
10° Nivel de Ingeniería de Sistemas
miembro del Grupo de Investigación en Ingeniería Biomédica GIIB
Universidad Industrial de Santander
Spino327@gmail.com
COLOMBIA

IRENE LIZETH MANOTAS GUTIÉRREZ
10° Nivel de Ingeniería de Sistemas
miembro del Grupo de Investigación en Ingeniería Biomédica GIIB
Universidad Industrial de Santander
irenelizeth@gmail.com

MPE. HENRY ARGUELLO FUENTES
COLOMBIA
Profesor titular de la Escuela de Ingeniería de Sistemas
Director grupo GIIB
Universidad Industrial de Santander
henarfu@uis.edu.co
COLOMBIA

Fecha de recepción: 15 de Septiembre de 2008
Artículo tipo 2.

RESUMEN

Hoy en día las redes Peer-To-Peer están cobrando gran importancia no sólo por su capacidad para compartir recursos entre varios nodos de una red, sino también por sus capacidades sobre la comunicación instantánea y la computación distribuida. Un aplicación basada en una red Peer-To-Peer debe tener características especiales que le permitan manejar el comportamiento de la red en general, así como controlar las comunicaciones entre los nodos y servicios que se prestan entre éstos nodos. Para alcanzar los resultados esperados en el desarrollo de cualquier solución software es fundamental disponer de una arquitectura que facilite el desarrollo, elimine al máximo los puntos de error evitando la redundancia de código, y esté acorde con el problema. Este artículo propone una arquitectura software en capas para aplicaciones basadas en redes Peer-to-Peer que permita alcanzar las metas de diseño y funcionalidad de una manera eficiente, de fácil entendimiento y que sobretodo, pueda ser una solución escalable para que el desarrollo de otras funcionalidades sobre ésta arquitectura sea posible. Dentro de la arquitectura propuesta en este artículo se toma como base el Framework JXTA basado en J2SE de Sun Microsystems para el desarrollo de un software que utiliza una red P2P.

PALABRAS CLAVE: Redes P2P, Arquitectura de aplicación, Framework JXTA, Swing Application Framework.

ABSTRACT

Today networks Peer-To-Peer are gaining great importance not only for his ability to share resources among multiple nodes of a network, but also by their abilities on instant communication and distributed computing. An application based on a network Peer-To-Peer must have special features that enable it to handle behavior of the network in general, as well as monitor communications between nodes and services provided between these nodes. We know that to achieve the expected results in the development of any software solution is essential to have an architecture that facilitates development, remove the most points of error avoiding redundancy code, and is commensurate with the problem. This article proposes a layered software architecture for network-based applications Peer-to-Peer to achieve the goals of design and functionality of an efficient, easy understanding and above all, can be a scalable solution for the development of other features on this architecture possible. Within the architecture proposed in this paper is taken as the basis Framework JXTA based on J2SE Sun Microsystems to develop software that uses a P2P network.

KEY WORDS: P2P Networks, architecture application, Framewok JXTA, Swing Application Framework.

1. INTRODUCCIÓN

En el proceso de desarrollo de una solución software de cualquier índole es necesario disponer de un diseño con el que se puedan alcanzar los resultados esperados, que no contenga código redundante y aproveche al máximo las capacidades de la máquina[1]. El diseño de una aplicación basada en redes Peer-to-Peer(P2P) para compartir archivos, realizar computación distribuida y mensajería instantánea no escapan a esta necesidad. Este articulo propone una arquitectura basada en capas para el desarrollo de un proyecto software basado en una red P2P, en donde los objetivos fundamentales de la arquitectura son evitar al máximo el código redundante y separar las funcionalidades inherentes a las redes P2P de la Interfaz Gráfica (GUI) de la aplicación.

Este artículo está dividido en cuatro partes: La primera parte hace referencia a la definición y estructura de las redes P2P. La segunda parte comprende el Framework JXTA, la razón de por qué se escogió como base para el desarrollo y su estructura de protocolos para el manejo de redes P2P. Luego, la tercera parte indica cómo se realizó el planteamiento de la arquitectura basada en capas para el desarrollo de la aplicación software. Y para terminar, la cuarta parte está comprendida por las conclusiones.

2. PEER TO PEER (P2P)

En pocas palabras una red informática entre iguales, Peer to Peer o P2P, se refiere a una red que no tiene clientes ni servidores fijos, sino una serie de nodos que se comportan simultáneamente como clientes y como servidores de los demás nodos de la red.

P2P es la solución a una sencilla pregunta: ¿Cómo puedo conectar un conjunto de dispositivos de modo que puedan compartir información, recursos y servicios (colaborar, comunicarse y compartir)?

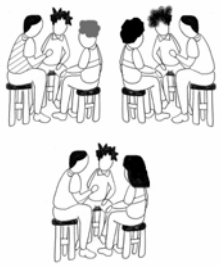
Parece ser una pregunta sencilla, pero en realidad está compuesta de varias preguntas implícitas: ¿Cómo un dispositivo localiza a otro?, ¿Cómo organizar los dispositivos para tratar intereses comunes?, ¿Cómo un dispositivo hace para que sus capacidades sean conocidas?, ¿Qué información es necesaria para identificar exclusivamente un dispositivo?, ¿Cómo los dispositivos intercambian datos?[2].

Con lo siguiente se plantea, que para obtener una solución a un problema utilizando redes P2P hay que dar respuesta a estas preguntas con el desarrollo de herramientas software que sean capaces de solucionarlas.

Para desarrollar la idea de qué es P2P, se tomó como base el siguiente planteamiento: Programar es una forma de pensar, codificar es una forma de hablar y P2P es una forma de interactuar . Donde interactuar se puede entender en cómo una comunidad comparte, colabora y se comunica. En los seres humanos la esencia de la interactividad radica en “la conversación bidireccional receptor-emisor y en el grado en que la comunicación supere ésta” [3]. En P2P la interactividad se basa también en el paso de mensajes bidireccional receptor-emisor al mismo nivel, es decir, de igual a igual ¨Figura 1¨.

Para desarrollar interactividad con una red P2P son necesarios una serie de elementos y características como son los iguales, grupos de iguales, servicios, transporte de red, anuncios, protocolos y la identificación o nombramiento de entidades que conforman la red P2P. [2].

Figura 1. Analogía de una red P2P y un grupo de personas interactuando.



2.1 IGUALES

En una red P2P las entidades denominadas iguales (peers), son los mismos nodos de la red y representan la unidad fundamental de procesamiento de cualquier solución P2P. “Un igual se define como cualquier entidad capaz de ejecutar algún trabajo útil y comunicar los resultados de ese trabajo a otra entidad sobre una red, directa o indirectamente”[2]. Es decir, puedo tener como igual un dispositivo o una aplicación dentro de una red P2P, o una persona dentro de una red

social ¨Figura 2¨.

Figura 2. Analogía de un igual dentro de un grupo interactuando.

Existen tres tipos de iguales dentro de una red P2P que tienen diferentes responsabilidades. Estos iguales son: iguales simples, iguales rendezVous e iguales router.

2.1.1. Iguales Simples

Un igual simple o igual es una aplicación de usuario final, que permite proveer servicios desde el dispositivo y consumir servicios proveídos por otros iguales en la red.

2.1.2. Iguales RendezVous

Un igual rendezVous (RDV) es un igual que implementa y provee recursos para soportar el despliegue y operación de la red P2P, por lo tanto se le considera como un igual de infraestructura. Un igual rendezVous es considerado como un lugar de encuentro o de reunión, en donde un igual puede descubrir o conocer a otros iguales y recursos.

Para lograr esto, los iguales envían consultas de descubrimiento a un igual rendezVous y este devuelve información sobre los iguales y recursos de los que él tiene conocimiento en la red. Para poder realizar este envío de consultas es necesario que el igual rendezVous sea accesible por los iguales, por ende y en general este estará fuera de una red interna y tendrá una IP resoluble desde Internet (IP pública).

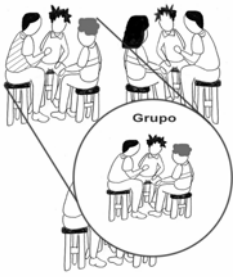
2.1.3. Iguales Router

De la misma forma que un igual rendezVous, un igual router también es de infraestructura y es necesario que se encuentre fuera de una red interna y tenga una IP resoluble desde Internet. Un igual router proporciona un mecanismo para que los iguales se comuniquen con otro igual cuando este está separado de la red por una combinación firewall/NAT. El igual router funciona como un intermediario que los iguales afuera del firewall pueden utilizar para comunicarse con un igual que está adentro y viceversa.

2.2. GRUPOS

En un red P2P, el concepto de grupo de iguales permite subdividir el espacio de la red. Por definición “un grupo es un conjunto de iguales formado para servir a un interés común o un objetivo específico de los iguales involucrados”[2]. Visto desde otro punto de vista, al crear un grupo de iguales se crea un espacio en la red sobre el cual se proveen servicios a los iguales miembro que no son accesibles por otros iguales en la red P2P ¨Figura 3¨.

Figura 3. Analogía de un grupo de iguales



.

Los objetivos de un grupo de iguales se pueden generalizar en tres objetivos principales:

Ofrecer una aplicación en la que los iguales puedan colaborar como un grupo: Un grupo de iguales es formado para intercambiar servicios que los miembros no quieren tener disponibles para la red P2P entera, debido por ejemplo a la naturaleza privada de los datos.

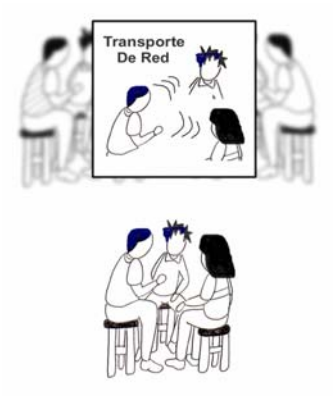
Implementar requerimientos de seguridad a los iguales involucrados: Un grupo de iguales puede implementar mecanismos de autenticación para restringir el acceso a solo los iguales que puedan unirse al grupo y utilizar los servicios ofrecidos por este.

Brindar información de estado de los miembros del grupo: Los miembros de un grupo de iguales podrían monitorizar otros miembros del mismo grupo, con el fin de mantener un nivel mínimo de servicios para la(s) aplicación(es) del grupo de iguales.

2.3. TRANSPORTE DE RED

Para intercambiar datos, los iguales deben emplear algún tipo de mecanismo para manejar la transmisión de datos sobre la red. El transporte de red ¨Figura 4¨ es el responsable de todos los aspectos de la transmisión de los datos, es decir, partir los datos en paquetes manejables agregando cabeceras apropiadas a un paquete para controlar su destino y asegurar que un paquete llegue a su destino.

Figura 4. Analogía de un transporte de red.



Entre los transportes de red se tienen los de bajo nivel, como UDP o TCP, o de alto nivel como HTTP o SMTP. En P2P, el concepto de transporte de red puede ser dividido en tres partes:

2.3.1. Puntos finales (Endpoints):

Son el origen o destino de cualquier pieza de datos que son transmitidos sobre la red. Un Endpoint corresponde a las interfaces de red utilizadas para enviar y recibir datos. El endpoint proporciona el acceso a la interfaz de red.

2.3.2. Tuberías (Pipes):

Son canales de comunicación unidireccionales, asíncronos y virtuales conectando dos o más Endpoints. Es una abstracción usada para representar el hecho de que dos Endpoints están conectados.

2.3.3. Mensajes:

Son contenedores de datos siendo transmitidos sobre un pipe desde un endpoint a otro.

Para enviar datos de un igual a otro, el igual emisor empaqueta los datos en un mensaje y luego este se envía usando un output pipe; mientras que el igual receptor recibe el mensaje usando un input pipe para luego extraer los datos de este.

Los pipes son canales unidireccionales, con el fin de otorgar soporte y no excluir ningún transporte de red, partiendo del hecho de que cualquier transporte de red bidireccional puede ser modelado usando dos pipes unidireccionales.

2.4. SERVICIOS

Los servicios proveen a los iguales la posibilidad de ejecutar "trabajo útil" en un igual remoto. Un trabajo útil se entiende como una transferencia de archivos, monitoreo de dispositivos, cálculos, etc. Los servicios son la motivación para reunir dispositivos en una red P2P, ya que sin estos no se tendría una red P2P.

2.5. ANUNCIOS

Un anuncio en P2P se define como "una representación estructurada de una entidad, servicio o recurso hecho disponible por un igual o un grupo de iguales como una parte de una red P2P" [2], es decir, los iguales, grupos de iguales, pipes, endpoints, servicios y hasta un contenido, pueden ser representados como anuncios.

2.6. PROTOCOLOS

Por definición un protocolo es una "manera de estructurar el intercambio de información entre dos o más partes mediante normas que han sido previamente acordadas por todas las partes" [2]. En P2P los protocolos son necesarios para definir los diferentes tipos de interacción que un igual puede hacer como parte de la red P2P.

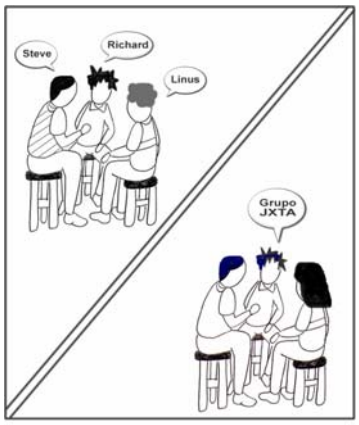
Debido a que los anuncios definen la estructura y representación de los datos, la definición de los protocolos en P2P se simplifica y se centra en organizar el intercambio de los anuncios que contienen la información necesaria para ejecutar alguna funcionalidad de igual.

2.7. IDENTIFICACIÓN O NOMBRAMIENTO DE ENTIDADES

La mayoría de las entidades en una red P2P (Iguales, grupos de iguales, pipes y contenidos) necesitan que se les identifique de manera única en la red ¨Figura 5¨, para esto se definen IDs que en

esencia lo que permiten es la diferenciación de cada entidad permitiéndoles ser ubicadas en la red.

Figura 5. Analogía al nombramiento de entidades en una red P2P.



3. FRAMEWORK JXTA

Escoger un buen Framework es una decisión fundamental en el inicio de cualquier proyecto de software, ya que el futuro del proyecto está ligado a la tecnología utilizada. El ¿Por qué? se escogió JXTA como Framework se fundamentó en sus metas de diseño, las cuales responden a las necesidades del conjunto más grande posible de aplicaciones P2P: independiente de la plataforma, independiente del lenguaje de programación e independiente de los protocolos de red, las cuales definen la esencia de JXTA.

JXTA™ es un conjunto abierto de protocolos peer-to-peer (P2P) generalizados que permiten a cualquier dispositivo en red -sensores, teléfonos celulares, PDAs, laptops, workstations, servers y supercomputadores- comunicarse y colaborar mutuamente como iguales [4].

La característica más relevante de JXTA es que permite llevar Internet dentro de las aplicaciones en lugar de utilizarse sólo en un navegador Web, además de su esencia que lo hace independiente de la plataforma, del lenguaje de programación y de los protocolos de red. Esto significa que las aplicaciones basadas en el Framework JXTA poseen una interfaz y capacidad real, que les permite ejecutarse 24 horas al día si fuere necesario[5].

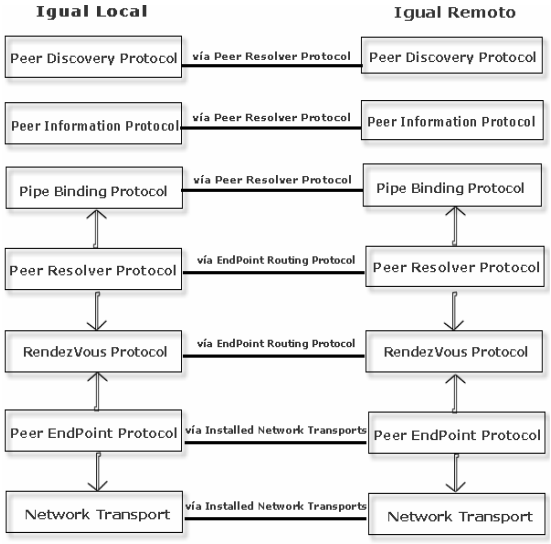
Además, sólo con el Framework JXTA el intercambio de información, como con la transferencia de archivos, puede realizarse de una forma que se denomina *User To User* [6], o de usuario a usuario, lo que significa que los iguales conectados a la red tendrán más seguridad al compartir sus archivos, ya que sus iguales no son anónimos como con las redes P2P para transferencia de archivos tradicionales.

3.1. PROTOCOLOS DEL FRAMEWORK JXTA

Los protocolos JXTA están basados sobre mensajes XML. Cada protocolo se encarga exactamente de un aspecto fundamental de la red P2P y es semi-independiente de los otros protocolos. Sin embargo, los protocolos no son enteramente independientes entre sí, ya que cada capa en la pila de protocolos de JXTA depende de la capa de más abajo para proporcionar conectividad hacia otros iguales ¨Figura 6¨.

Todos los protocolos definidos por la especificación de protocolos JXTA son implementados como servicios llamados *Core Services.*

Figura 6. Protocolos JXTA



Los protocolos JXTA son descritos a continuación:

3.1.1 Peer Discovery Protocol

Habilita a los iguales para descubrir servicios de otros iguales en la red.

3.1.2 Peer Resolver Protocol

Permite a los iguales enviar y procesar peticiones genéricas.

3.1.3 Rendezvous Protocol
Maneja los detalles de la propagación de mensajes entre iguales.

3.1.4 Peer Information Protocol
Provee a los iguales con un método para obtener información de estado de otros iguales en la red.

3.1.5 Pipe BInding Protocol
Posee un mecanismo para atar un canal de comunicación virtual a un endpoint.

3.1.6 Endpoint Routing Protocol
Provee un conjunto de mensajes utilizados para permitir el enrutamiento de mensajes de un igual hacia otro.

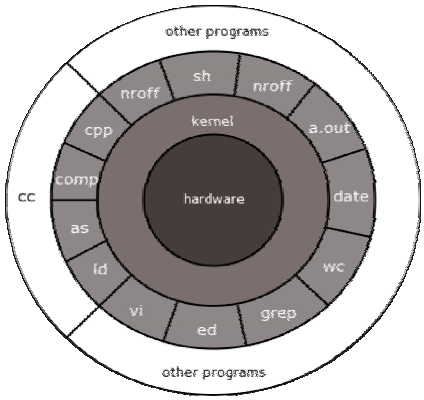
4. PLANTEAMIENTO DE LA ARQUITECTURA DE LA APLICACIÓN

A continuación se mostrará el diseño de la arquitectura planteada para el software basado en una red P2P.

En el diseño de una arquitectura de aplicación influyen varios factores como son la definición de los módulos principales de la aplicación y sus responsabilidades, la interacción que existe entre estos módulos, el control y flujo de los datos, los protocolos de interacción y comunicación, y la ubicación del hardware. Para el desarrollo de la arquitectura de aplicación aquí expuesta se manejaron las siguientes herramientas: el paradigma orientado a objetos (POO), la metodología de desarrollo rápido extreme programming (XP) y el ingenio de los desarrolladores. El POO se utiliza básicamente como modelo mental, con el cual se analiza la solución en términos de los actores (objetos) que intervienen en el sistema y como estos interactúan para dar solución al problema. De la metodología de desarrollo, extreme programming, se utilizan los planteamientos más globales, centrándose en la simplicidad y la característica OAOO “Once and only once¨ [1].

Para determinar la arquitectura fundamental del proyecto en desarrollo, que de ahora en adelante se denominará U2U, se tuvo en cuenta la segunda meta de diseño del sistema operativo UNIX, la cual tiene como propósito el desarrollo de utilerías simples que realizan tareas especificas de la mejor manera posible y que al combinarlas en una línea de comandos se puede obtener casi cualquier resultado posible [7] ¨Figura 7¨.

Figura 7. Arquitectura en capas de UNIX



La arquitectura UNIX propone en su tercera capa el utilitario del sistema llamado SHELL.

El SHELL de UNIX es un intérprete de comandos. Su tarea es tomar los comandos enviados por el usuario, interpretarlos y llamar a las rutinas correspondientes.

Cuando el SHELL lee una línea de comandos, extrae la primera palabra, asume que éste es el nombre de un programa ejecutable, lo busca y lo ejecuta. El SHELL suspende su ejecución hasta que el programa termina, tras lo cual intenta leer la siguiente línea de órdenes.

Basándose en la meta de diseño de UNIX, se planteó el propósito principal de la arquitectura base para el proyecto U2U: realizar una arquitectura basada en un aplicativo SHELL que de soporte a utilerías simples pero fundamentales en el manejo de una red P2P ¨Figura 8¨.

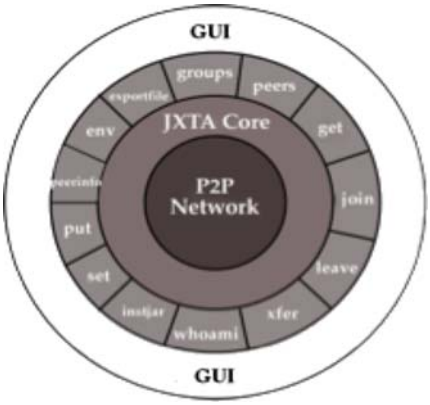
La arquitectura del proyecto U2U propone el diseño de cuatro capas:

Primera Capa: Red Peer To Peer
Segunda Capa: Servicios fundamentales de JXTA (Protocolos JXTA)
Tercera Capa: Aplicativo SHELL U2U
Cuarta Capa: Interfaz Gráfica del Usuario

4.1. Descripción de las capas de la arquitectura en capas

En la primera capa de la arquitectura se encuentra la red P2P construida, es decir, la definición de los iguales Edge, iguales Rendezvous e iguales Relay, y todo lo referente a la configuración de la red.

Figura 8. Arquitectura Proyecto sobre redes P2P



En la segunda capa se encuentran los servicios fundamentales de JXTA y otros servicios implementados sobre la red. Estos servicios proporcionan la base de comunicación para la red P2P y la prestación de otros servicios como Servicio para la transmisión de archivos entre iguales, Servicio de procesamiento distribuido, entre otros.

En la tercera capa se encuentra el aplicativo SHELL U2U, que está basado sobre el SHELL de JXTA [8]. El SHELL de JXTA está conformado por una serie de comandos básicos como groups, peers, get, join, leave, whoami, entre otros. Estos comandos permiten acceder a funciones o aplicaciones pequeñas que ejecutan tareas específicas sobre la red P2P. Algunos de éstos comandos se encargan de crear grupos, ingresar a un grupo, abandonar un grupo, buscar iguales en la red, transmitir mensajes, publicar anuncios, entre otras funciones, dentro de la red P2P.

En la cuarta capa se tiene la Interfaz Gráfica del Usuario. Esta es la última capa en la arquitectura propuesta y es la capa que se comunica con el SHELL U2U. De esta manera, al ejecutarse un evento en la interfaz gráfica que involucre la ejecución de alguna funcionalidad sobre la red P2P, el evento se comunicará directamente con el SHELL U2U y éste último realizará las operaciones necesarias para llevar a cabo la tarea propuesta.

4.2. Metas de diseño de la arquitectura en capas

Para conseguir el diseño de la anterior arquitectura se propusieron las siguientes metas de diseño:

3 Primero: Separar las funcionalidades de la interfaz gráfica de usuario de las funcionalidades de la red P2P. Es decir, se centralizarán las funcionalidades de la red P2P en objetos que expondrán métodos los cuales serán llamados en los eventos generados en la interfaz gráfica.

- Segundo: Utilizar como base para la ejecución de funcionalidades P2P el Shell U2U, el cual se extiende del SHELL JXTA para implementar nuevos comandos y soportar mejoras a la red, como por ejemplo la implementación de un servicio de transferencia de archivos entre iguales.

La primera meta se soporta íntegramente en el paradigma orientado a objetos (POO) y en lo propuesto en extreme programming (XP), con relación a reutilización, responsabilidad, simplicidad y OAOO.

4.2.1 Uso del Swing Application Framework dentro de la Capa "GUI"

Para evitar al máximo escribir bloques de código redundantes en cada evento de la interfaz grafica de usuario (GUI), se utiliza el Swing Application Framework JSR 296. Este framework otorga entre otras cosas, un ciclo de vida de aplicación y acciones (@Actions) que permiten escribir métodos los cuales pueden ser enlazados con varios objetos de la interfaz gráfica. En la ¨Figura 9¨ se puede observar un ejemplo de cómo codificar un método como una acción.

El enlace entre la GUI y el código se puede definir como una relación de uno a muchos, en donde varios objetos de la GUI tales como botones, menús, cajas de verificación, etc., llaman a un único bloque de código definido por un método con la anotación @Action cuando se genera su evento por defecto, sin generar redundancia y con el beneficio de mantener solo un método. Esto permite al desarrollador evitar escribir varios métodos que ejecutan la misma tarea en diferentes clases que conforman el desarrollo, logrando así dos objetivos; No escribir código redundante y cumplir con la característica *Once And Only Once* del desarrollo con eXtreme Programming[1].

Figura 9. Método con anotación @Action, el cual puede ser llamado desde cualquier componente de la GUI realizando un enlace.

```
@Action
public void quitApp()
{
        clearAllObjects();
        this.setVisible(false);
}
```

La segunda meta de diseño se centra en el SHELL U2U, el cual es una nueva implementación del SHELL de JXTA que ofrece otros comandos adicionales como el comando u2ufs, el cual realizará todas las tareas necesarias para la ejecución de un servicio de compartición de ficheros sobre la red P2P.

4.2.2 Uso del Shell U2U

Todo el manejo de la comunicación P2P se centralizara en el SHELL U2U a través de comandos que encapsulan el manejo de los protocolos ofrecidos por el framework JXTA.

El primer desafío de la implementación de ésta arquitectura fue solucionar la manera en que se pudieran comunicar la capa de la GUI y la capa del SHELL U2U, de tal manera que ambas capas fuesen independientes. Para lograr esto se aplicó ingeniería inversa al código fuente del SHELL JXTA para entender su funcionamiento, manipulación y lograr agregar nuevas funcionalidades. Uno de los resultados de la aplicación de la ingeniería inversa fue la definición del SHELL U2U. En la¨Figura 11¨ se puede apreciar la manera cómo se crea un nuevo objeto de la clase Shell que se encuentra dentro del paquete net.jxta.impl.shell.bin del SHELL U2U.

Figura 11. Instancia de un objeto Shell

```
//instancia de la clase Shell para el proyecto U2U
 shell = new  Shell(netPeerGroup);
```

La ¨Figura 11¨ muestra un nuevo objeto del SHELL U2U que se inicializa con el grupo inicial al que pertenece el igual que ingresa a la red P2P.

La diferencia entre el SHELL JXTA y el SHELL U2U, es que el segundo no implementa todos los comandos del primero, al no considerarse todos los comandos del SHELL JXTA necesarios para el propósito del proyecto U2U. Adicionalmente el SHELL U2U implementa otros nuevos comandos que permiten el manejo de nuevos servicios sobre la red P2P y finalmente, el SHELL U2U no se ejecuta dentro de la consola de comandos, ya que se considera una interfaz poco manejada por usuarios finales, sino que se utiliza oculto como una implementación que ayuda a ejecutar tareas solicitadas por el usuario dentro de una GUI enriquecida.

Figura 12. Interacción entre la GUI del proyecto U2U y el SHELL U2U (Capas 3 y 4 de la arquitectura en capas propuesta)

En la “Figura 12” se puede apreciar la interacción entre la GUI del proyecto U2U y el SHELL U2U. El usuario genera un evento presionando el botón “Buscar Iguales” (Paso 1), este evento desencadena la ejecución de un comando en el SHELL, que se encuentra oculto para el usuario, y realiza las funciones necesarias para buscar los iguales que se encuentren conectados a la red P2P (Paso 2), luego el SHELL envía los resultados obtenidos a la GUI y ésta se encarga de mostrarlos finalmente al usuario (Paso 3).

La manera como se comunica la interfaz gráfica de la aplicación (Capa 4) con el SHELL (Capa 3) es a través de un objeto de la clase shell, el cual proporciona los métodos necesarios para la ejecución de los comandos generados por los eventos solicitados en la GUI “Figura 13”.

Figura 13. Método executeCmd de la clase Shell

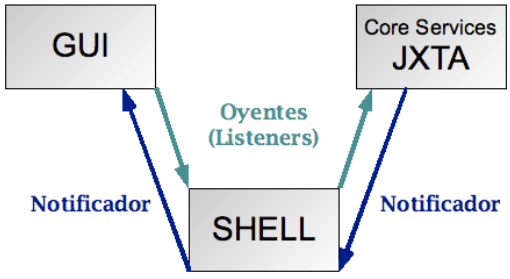
```
Public void executeCmd(String comando)
{
        /*
         Implementación del código necesario
        para ejecutar cada uno de los
        comandos proporcionados por el SHELL  */

}
```

En la mayoría de los casos los resultados de la ejecución de un comando son respuestas asíncronas debido a la naturaleza distribuida de la red P2P. Para manejar estos resultados asíncronos se utilizan interfaces oyentes o Listener, las cuales permiten implementar un oyente para eventos específicos, permitiendo capturar las respuestas asíncronas de dicho evento.

En la "Figura 14" se puede apreciar el modelo general de oyentes manejados en la arquitectura en capas propuesta. La GUI de la aplicación se define como un oyente del SHELL y a su vez, el SHELL se define como un oyente de los protocolos JXTA. De esta manera, un evento generado en la GUI informa al SHELL de la ejecución de un comando en especial y éste comando realiza las tareas necesarias sobre la pila de protocolos de JXTA. Cuando las tareas sobre la pila de protocolos de JXTA han culminado se le notifica al SHELL que el proceso ha terminado a través del oyente y del notificador. El SHELL a través de otro notificador informa a la GUI, por medio de su oyente, de los resultados obtenidos de la ejecución del evento.

Figura 14. Modelo de oyentes manejado en la arquitectura en capas.



4.3. Desarrollo de una aplicación P2P utilizando la arquitectura basada en capas.

Para desarrollar una aplicación con la arquitectura basada en capas descrita, el desarrollador debe incluir dentro de sus librerías el Framework JXTA para J2SE [9] y la Liberia del proyecto Shell U2U. Estas dos librerías le proporcionarán al desarrollador el acceso a los protocolos de JXTA, a algunos comandos básicos del SHELL de JXTA y, por parte del proyecto U2U, el acceso a la implementación de los oyentes entre las capas de la arquitectura propuesta y la posibilidad de explorar cómo crear un servicio sobre la red P2P basados en la arquitectura en capas a través de la implementación del comando u2ufs.

Una vez que el desarrollador haya incluido las librerías, debe diseñar los servicios que quiere ofrecer sobre la red P2P, de acuerdo al propósito de su aplicación. Al tener definidos el servicio o los servicios que desea prestar sobre la red P2P, se debe crear un comando sobre el Shell por cada servicio creado. Al final, se deben enlazar la GUI de la aplicación con los servicios anteriormente desarrollados con la ayuda del Shell U2U y los comandos creados.

5. CONCLUSIONES

Con la apropiación del concepto de una red P2P, se puede lograr dimensionar la tecnología a niveles independientes de los protocolos utilizados (Gnutella, GNUnet, JXTA, etc.). Además, un conocimiento fuerte sobre la composición y funcionamiento de las redes P2P permite plantear soluciones basadas en este tipo de redes a ciertos problemas en donde otras arquitecturas de red no aplican.

Para darle un adecuado manejo al framework JXTA, que se utiliza como base del proyecto, es necesario conocer de forma detallada su estructura, lo que implica el estudio de los protocolos básicos de funcionamiento del framework que se definen a través de clases e interfaces. Este conocimiento detallado sobre lo que significa cada protocolo dentro JXTA y cómo se utilizan, permite utilizar las funcionalidades de cada protocolo dentro del software en desarrollo, ya que éstos proporcionan los fundamentos para crear y poner en funcionamiento una red P2P.

El SHELL JXTA proporciona algunos comandos que muestran características básicas de una red P2P, pero estos comandos no logran explotar todas las características brindadas por una red P2P como permitir el intercambio eficiente de archivos, ejecutar computación distribuida o establecer una comunicación interactiva entre dos o más iguales.

Sin embargo, el Shell de JXTA proporciona una base que permite a los desarrolladores incluir más comandos dentro de su estructura de paquetes para poder implementar aquellas características que le hagan falta.

En este proyecto se logró no sólo entender el funcionamiento y estructura del framework JXTA sino también entender la composición del SHELL JXTA y basándose en esto se diseño un nuevo SHELL que implementa algunos comandos ya desarrollados en el SHELL JXTA y otros comandos nuevos que permiten ejecutar servicios creados para ejecutar nuevas tareas sobre la red P2P, como el comando u2ufs que permite el intercambio de ficheros dentro de una red P2P de manera óptima.

La arquitectura basada en capas propuesta en este artículo brinda a los desarrolladores una manera más adecuada para llevar a cabo la implementación de una aplicación basada en redes P2P, ya que no sólo se soporta en un framework robusto como lo es JXTA, sino que además brinda una capa de enlace (tercera capa de la arquitectura: el SHELL) entre la GUI y los protocolos JXTA , de esta manera logrando que todas las funcionalidades que brinde la aplicación final queden encapsuladas dentro de los comandos del SHELL independientes de la GUI que se maneje. De esta forma, se podrían generar sobre el SHELL de JXTA tantos comandos como funciones fueran posibles sobre una red P2P y utilizar sólo los comandos necesarios para cada aplicación P2P que se quiera desarrollar.

6. AGRADECIMIENTOS

La arquitectura en capas mostrada en el presente artículo es la base del proyecto titulado “Software para el mejoramiento en la trasmisión de datos utilizando redes P2P” el cual fue apoyado por la vicerrectoría de Investigación y extensión de la Universidad Industrial de Santander gracias al premio obtenido en la convocatoria de proyectos promisorios 2008.

7. REFERENCIAS

[1] Cortizo, Jóse. Et al. “eXtreme Programming”, http://www.esp.uem.es/jccortizo/xp.pdf, Fecha de consulta: Marzo 2008.

[2] Brendom Wilson, “JXTA”, New Riders, Junio 2002

[3] Lamarca, María Jesús, “Interactividad”, http://www.hipertexto.info/documentos/interactiv.htm , Fecha de consulta: Agosto 2008.

[4] JXTA Java™ Standard Edition v2.5: Programmers Guide, Septiembre 2007.

[5] Daniel Brookshier, “'Make Every PC a Server' – Is That JXTA's Killer App?”, http://jxta-sys-con.com, Fecha de consulta: Abril 2008.

[6] J. Taylor, From P2P to Web Services and Grids: Peers in a Client/Server World, Springer, Octubre 2004.

[7] William Stallings, Sistemas Operativos, Editorial Prentice Hall, 2003.

[8] Proyecto Shell de JXTA, https://jxse-shell.dev.java.net/, Fecha de consulta: Marzo 2008.

[9] Proyecto JXSE, https://jxta.dev.java.net/ , Fecha de consulta: Marzo 2008.

[10] Juan Carlos Soto, Presentación: “Project JXTA: An open P2P applications platform, Introduction and update”, Sun Microsystems, http://www.jxta.org., Fecha de consulta: Abril 2008.

[11] Dr Simon See, Presentación: “Project JXTA Technology overview ”, Sun Microsystems, http://www.jxta.org., Fecha de consulta: Abril 2008.