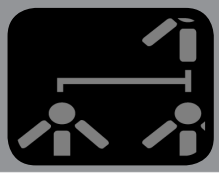
ARQUITECTURA EN CAPAS PARA EL DESARROLLO DE UNA APLICACIÓN BASADA EN REDES PEER-TO-PEER



AUTOR
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

AUTOR
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
COLOMBIA

AUTOR
MPE. HENRY ARGUELLO FUENTES
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

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

Vol 7|No 19|Bucaramanga|Colombia|, Ene, Feb, Mar, Abr |2008|ISSN 1657-8236

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].

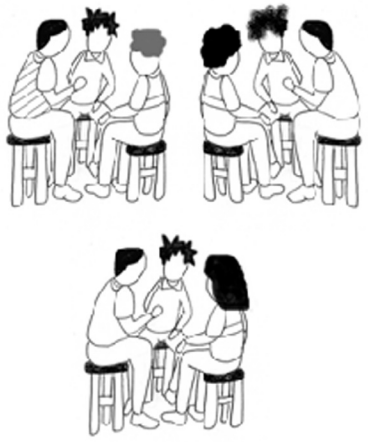


Figura 1. Analogía de una red P2P y un grupo de personas interactuando.

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¨.

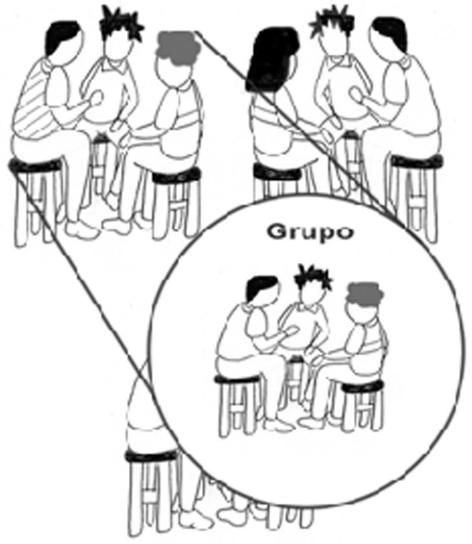


Figura 3. Analogía de un grupo de iguales

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.

Vol 7|No 19|Bucaramanga|Colombia|, Ene, Feb, Mar, Abr |2008|ISSN 1657-8236

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.

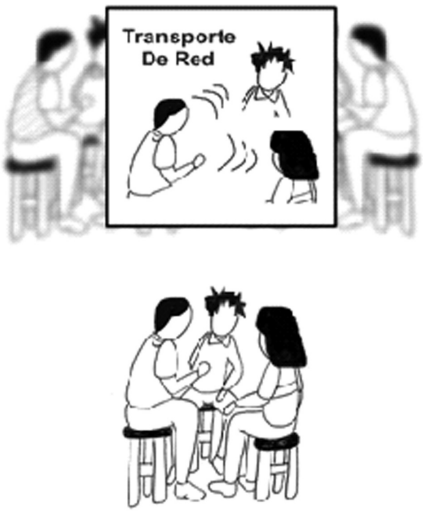


Figura 4. Analogía de un transporte de red.

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.

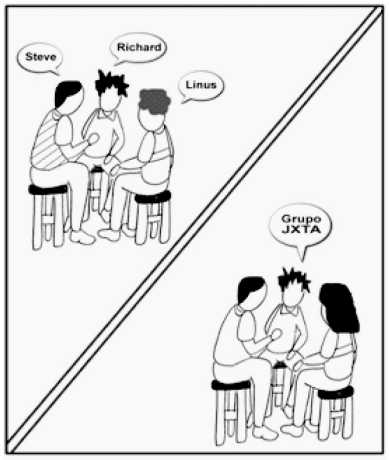


Figura 5. Analogía al nombramiento de entidades en una red P2P.

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.

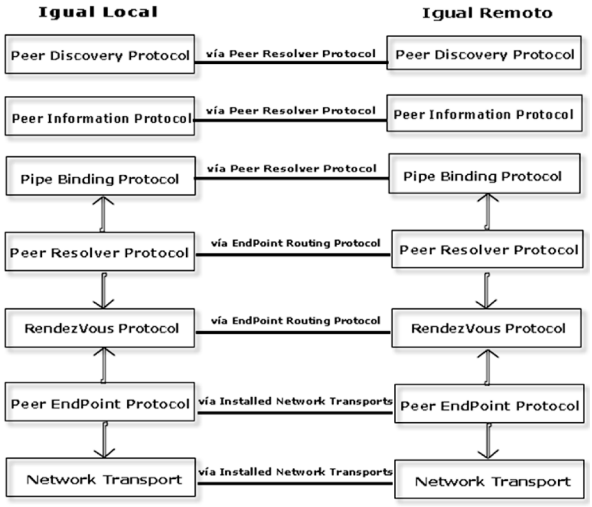


Figura 6. Protocolos JXTA

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.

Vol 7|No 19|Bucaramanga|Colombia|, Ene, Feb, Mar, Abr |2008|ISSN 1657-8236

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¨.

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:

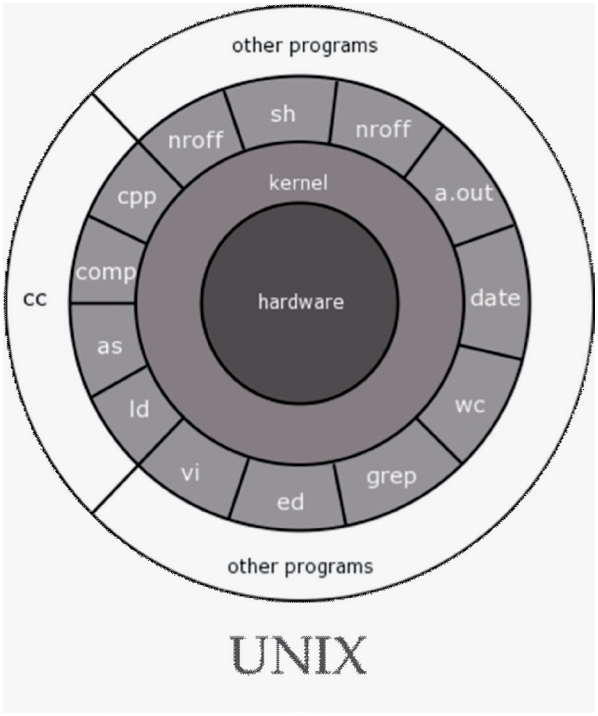


Figura 7. Arquitectura en capas de UNIX

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.

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.

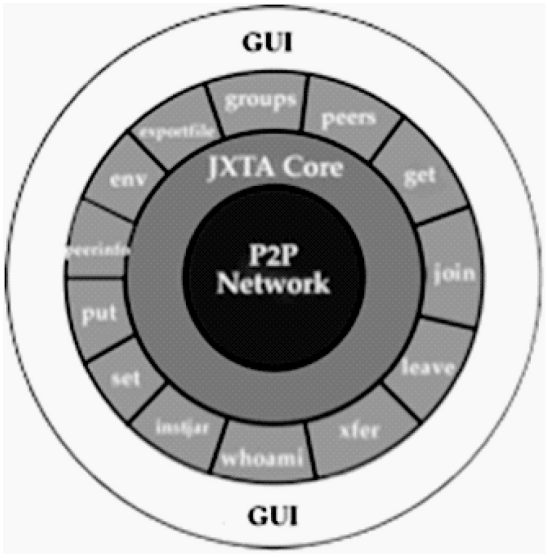


Figura 8. Arquitectura Proyecto sobre redes P2P

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:

- 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].

```
@Action
publicvoidquitApp()
{
clearAllObjects();
this.setVisible(false);
}
```

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

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.

Vol 7|No 19|Bucaramanga|Colombia|, Ene, Feb, Mar, Abr |2008|ISSN 1657-8236

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.

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

Figura 11. Instancia de un objeto Shell

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.

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).

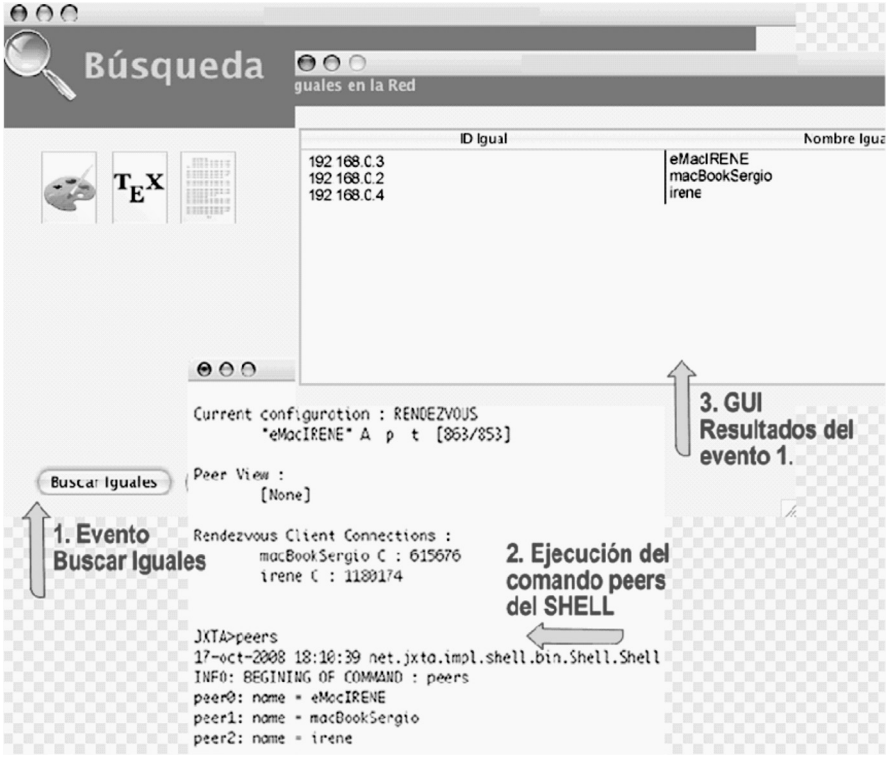


Figura 12. Interacción entre la GUI del proyecto U2U y el SHELL U2U (Capas 3 y 4 de la arquitectura en capas propuesta)

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".

```
Public void executeCmd (String comando)
{
        /*
        Implementación del códigonecesario
        paraejecutarcadauno de los
        comandosproporcionadospor el SHELL    */

}
```

Figura 13. Método executeCmd de la clase 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.

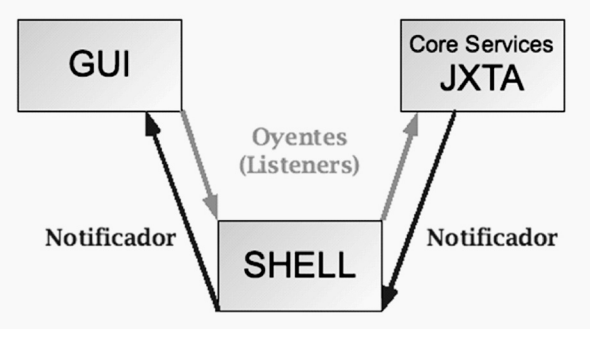


Figura 14. Modelo de oyentes manejado en la arquitectura en capas.

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

Vol 7|No 19|Bucaramanga|Colombia|, Ene, Feb, Mar, Abr |2008|ISSN 1657-8236

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.