EXTENSIÓN DE UNA PLATAFORMA QUE SOPORTE UN SISTEMA PARA LA PROVISIÓN DE SERVICIOS EN UN ENTORNO DE REDES SUPERPUESTAS P2P

GABRIEL RAMIRO MUÑOZ SAMBONÍ

EDWIN FERNANDO LARA QUINCHUA

Director: Ing. PABLO AUGUSTO MAGÉ IMBACHÍ

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
LÍNEA DE INVESTIGACIÓN INGENIERÍA DE LA COLABORACIÓN E INGENIERÍA DEL SOFTWARE BASADA EN LA COLABORACIÓN
DEPARTAMENTO DE SISTEMAS
POPAYÁN
JUNIO DE 2009

TABLA DE CONTENIDO

CAPITULO 1

INTRODUCCION

La revolución tecnológica aplicada a la comunicación, en su sentido más amplio, ha permitido construir una red mundial de computadores que se comunican entre si.

El uso de Internet se ha generalizado al punto de llegar a convertirse en un medio de comunicación imprescindible para millones de usuarios. Su continuo crecimiento, los grandes avances tecnológicos obtenidos en los equipos de los usuarios y el alto uso de estos sistemas, han provocado que surjan una serie de necesidades, convirtiendo a Internet en una infraestructura comercial para la recuperación de recursos y servicios. Por otro lado el crecimiento de desarrollo e implantación de nuevos servicios en la red de redes (Internet) y la dificultad que esto involucra, ha motivado la búsqueda de modelos alternativos al modelo tradicional Cliente/Servidor, que alivien esta situación dentro de la comunidad de investigadores en redes.

Las soluciones Cliente/Servidor dependen del ancho de banda, del hardware robusto y de las facilidades de rendimiento para prestar un rendimiento óptimo. Los sistemas P2P tienen la capacidad de servir recursos con una alta disponibilidad y a bajo costo, mientras maximizan el uso de los recursos de cada nodo conectado a la red.

Las redes superpuestas P2P han emergido como un novedoso paradigma y como una alternativa al modelo tradicional Cliente/Servidor. Su auge se debe principalmente a la intrínseca potencia de computación existente en los nodos (peers) que la componen y por que se caracterizan principalmente la potencial flexibilidad para hacer uso compartido de sus recursos informáticos (memoria, capacidad de almacenamiento, ancho de banda, etc.) de manera directa sin requerir la mediación de un servidor central.

Las propuestas iniciales de sistemas P2P se diseñaron para solucionar problemas específicos, con plataformas y protocolos específicos. Actualmente se llevan a cabo proyectos con la participación de la academia y la industria para desarrollar una arquitectura global que brinde soluciones a este tipo de problemas.

1.1 Justificación

Desde sus inicios, las redes superpuestas P2P se han visto fortalecidas con numerosas aplicaciones desarrolladas con el propósito de proveer soluciones específicas. Dichas aplicaciones implementan protocolos diseñados especialmente para soportar algún tipo de servicio en particular. Por ejemplo, los sistemas basados en el protocolo Fasttrack [1], como Morpheus o Kazaa, se especializan en la distribución de material multimedia. Gnutella y otros sistemas compatibles, tienen como objetivo compartir archivos. Napster ha sido usado para la misma tarea, en concreto con archivos de música en formato mp3. Skype es capaz de establecer comunicaciones de voz con una buena calidad mediante el uso de VoIP. De igual forma se pueden hallar soluciones cuyo objetivo es el mantenimiento de bases de datos distribuidas, como Mariposa [2].

Toda esta proliferación de aplicaciones ha llevado a algunos autores [3] a proponer una división por categorías, dependiendo de la topología de red y del tipo de servicio que presten en: computación distribuida (SETI@home, Avaki), aplicaciones para el intercambio de archivos (Napster, FreeNet, Gnutella, Kazaa), y colaboración (Groove, Magi).

Aunque existen aplicaciones integradoras de servicios como MSN, Yahoo o AOL Messenger con un paradigma lógico P2P, éstas hacen uso de servidores centralizados, además su código no es abierto, lo cual impide que se pueda llevar a cabo un estudio detallado de los protocolos que utilizan. Pese a todos estos esfuerzos aún no se han generado estándares que definan los servicios y los protocolos que se deben aplicar en los contextos de las redes superpuestas P2P.

Como se puede observar, la gran mayoría de las aplicaciones P2P se especializan en una tarea en particular; además, cada implementación utiliza un protocolo P2P propietario e incompatible con el resto de aplicaciones (Figura 1), lo que significa que los usuarios de diferentes soluciones P2P no tienen la posibilidad de comunicarse entre ellos (falta de interoperabilidad), provocando que la comunicación global entre dichos usuarios este limitada a “islas” P2P, cuyas fronteras están proporcionadas por los propios sistemas P2P que son empleados.

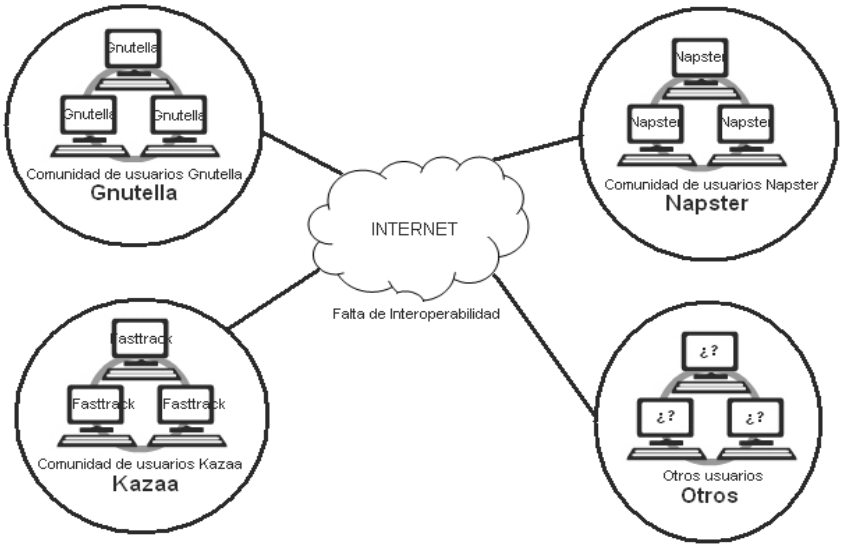


Figura 1. Falta de interoperabilidad en los sistemas P2P

El problema de la interoperabilidad derivado de la falta de estandarización de las aplicaciones P2P, supone una fragmentación de la base de usuarios, lo que repercute en las posibilidades de crecimiento de las mismas, por ejemplo, una aplicación diseñada para el uso compartido de archivos no puede comunicarse con una de mensajería instantánea y viceversa. Sin embargo, hoy en día se vienen proponiendo algunas plataformas de desarrollo, cuyo objetivo fundamental es proporcionar mecanismos para el desarrollo de aplicaciones P2P de propósito general (mensajería instantánea, distribución de material multimedia, descubrimiento de nodos, compartir archivos, etc.), buscando, de esta

manera, reducir al máximo la ausencia de interoperabilidad, lo que repercute en beneficio de toda la comunidad de usuarios P2P.

La aparición de un número cada vez mayor de este tipo de plataformas genera preguntas tales como, ¿Cuál de todas las plataformas para el desarrollo de aplicaciones P2P de propósito general es la más apropiada?, ¿Cuáles y qué características deben poseer los servicios que se puedan crear y/o adaptar a la filosofía P2P?, ¿Qué clase de modelos están establecidos para la provisión de servicios y cuál es su funcionamiento?

Cuando se habla del término plataforma, en el ámbito de las ciencias de la computación, se hace referencia a la configuración de hardware o de software presente en el computador. En el caso de la configuración de hardware, el término es usado para referirse a la arquitectura del procesador. Respecto a la configuración de software, el término plataforma se usa para hacer referencia al sistema operativo que se ejecuta en el computador [4].

Una plataforma para el desarrollo de aplicaciones, es una descripción conceptual y un conjunto de mecanismos de soporte que definen cómo un grupo de aplicaciones similares deberían ser desarrolladas [4]. Usualmente, una plataforma para el desarrollo de aplicaciones incluye: una arquitectura, patrones de diseño, protocolos, APIs[1] y posiblemente otras herramientas [5]. Algunos ejemplos de plataformas para el desarrollo de aplicaciones son: .NET y J2EE.

El creciente número de plataformas para el desarrollo de aplicaciones P2P ha generado la necesidad de llevar a cabo un estudio comparativo entre dichas plataformas con el objetivo de identificar la más apropiada para el desarrollo de aplicaciones P2P de propósito general, que sea lo suficientemente flexible para permitir la adición de un soporte software que facilite la inclusión de un sistema para la provisión de servicios. Este nuevo soporte software equivale a la extensión de la plataforma. Para esto es necesario especificar, ¿Cuáles deben ser las características para una extensión de una plataforma que soporte un sistema de provisión de servicios en un entorno de redes superpuestas P2P?, ¿Es posible validar dicha extensión mediante la implementación de una aplicación que use servicios de mediana complejidad?

Para dar respuesta a estos interrogantes, este proyecto de grado diseña e implementa una extensión para una plataforma que soporte un sistema para la provisión de servicios en un entorno de redes superpuestas P2P, además, se apoya en otros dos trabajos de grado: “Perfiles para la definición, descripción y construcción de servicios telemáticos orientados a una arquitectura Overlay Network Peer to Peer”, cuyo objetivo es definir un conjunto de perfiles que permitan dar soporte al modelado, descripción y construcción de servicios telemáticos para una arquitectura de redes superpuestas Peer to Peer, y otro denominado “Especificación e implementación de un nuevo modelo para la provisión de servicios en redes superpuestas P2P”, cuyo objetivo es especificar un nuevo modelo de provisión de recursos y servicios para un entorno de redes superpuestas P2P. Estos proyectos se vienen realizando en paralelo a este trabajo de grado y junto con este hacen parte de un proyecto general denominado “Contribución a un sistema de negociación de recursos y servicios en el ámbito de una red superpuesta P2P”.

[1] Application Programming Interface

1.2 Objetivos

1.2.1 Objetivo General

- Modelar e implementar una extensión de una plataforma para dar soporte a un sistema de provisión de servicios en un entorno de redes superpuestas P2P.

1.2.2 Objetivos Específicos

- Analizar las plataformas más relevantes propuestas en el área de las redes superpuestas P2P, e identificar, mediante un estudio comparativo, cual es la más apropiada para el desarrollo de aplicaciones P2P de propósito general.

- Especificar los requisitos para una extensión de una plataforma que soporte un sistema para la provisión de servicios P2P.

- Diseñar el prototipo de la extensión de la plataforma de red superpuesta P2P que soporte un sistema para la provisión de servicios P2P.

- Validar la extensión que soporta un sistema de gestión, mediante la implementación de una aplicación que use servicios de mediana complejidad bajo un entorno de redes superpuestas P2P.

1.3 Contribuciones

Los aportes más significativos de este trabajo de grado son:

- El análisis de las principales características de los sistemas P2P.

- La identificación, análisis y comparación de las plataformas más relevantes para el desarrollo de aplicaciones P2P.

- La creación de una extensión para la provisión de recursos y servicios P2P que permita obtener cierto grado de control en entornos descentralizados, mejorando de esta manera la plataforma existente.

- La integración del concepto de perfiles, agentes y sistemas P2P cuyo resultado permite obtener las bases para la creación de una plataforma lo suficientemente flexible y abierta para el desarrollo y soporte de servicios P2P.

CAPITULO 2

MARCO TEORICO Y CONCEPTUAL

2.1 Breve historia de los sistemas P2P

Antes de adentrarnos en la historia, nos parece de suma importancia realizar una pequeña introducción a lo que es el P2P. Aunque no existe una definición estricta, si es ampliamente aceptado que los sistemas P2P se caracterizan por su control descentralizado, la alta autonomía de sus nodos y la heterogeneidad en términos de potencia de procesamiento y de recursos hardware y software [6].

Los sistemas P2P poseen una arquitectura de red distribuida donde los nodos comparten parte de sus propios recursos hardware, por ejemplo: potencia de procesamiento, capacidad de almacenamiento, enlaces de red o impresoras. Estos recursos son necesarios para proporcionar el servicio y los contenidos ofrecidos por la red, esto es, intercambio de archivos o espacios de trabajo compartidos para colaboración. Los recursos son alcanzables sin la necesidad de recurrir a entidades intermediarias. Los participantes de tales redes son a su vez proveedores y solicitantes de recursos (servicios y contenido) [7].

La historia de los sistemas P2P se remonta originalmente a finales de la década del 70, sin embargo, no es si no hasta finales de la década del 90 que el término P2P se hizo popular gracias al famoso caso Napster. El P2P entró en escena como un nuevo paradigma en redes de computación, una nueva tecnología para conectar a los internautas y una manera de utilizar eficazmente los recursos dispersos a lo largo y ancho de Internet [8].

Las redes Usenet y FidoNet desarrolladas principalmente para el intercambio de noticias entre varios campus universitarios de los EEUU son consideradas el origen de esta revolucionaria tecnología.

Usenet fue creada en 1979 por Tom Truscott y Jim Ellis, dos estudiantes de la universidad de Carolina del Norte (EEUU). El objetivo era desarrollar un mecanismo para que dos o más computadores pudieran intercambiar información. Usenet no tenía autoridades de gestión centralizadas (servidores), la distribución de los contenidos era responsabilidad de cada nodo y la información (toda o parte) se replicaba por toda la red [8].

Fidonet (http://www.fidonet.org/) empezó como un experimento del norteamericano Tom Jennings en el año de 1984 en la ciudad de San Francisco (EEUU). Jennings quería utilizar su computadora y uno de los primeros módems que salieron al mercado para intercambiar archivos con un amigo suyo que tenía un equipo similar y que vivía en la ciudad de Baltimore (EEUU), para lo cual diseñó una serie de protocolos y creó un programa llamado Fido que permitía realizar una conexión de acceso telefónico entre ambos computadores sin la necesidad de intervención humana. De este modo el sistema se podía conectar automáticamente con el otro computador mientras su amigo no lo

utilizaba, generalmente durante las horas de la noche, cuando las tarifas del servicio telefónico eran más baratas [8].

A partir de los años 90 corporaciones como Intel y Boeing empezaron a utilizar redes P2P para realizar operaciones con gran volumen de cálculos, utilizando miles de computadores simultáneamente. Este tipo de iniciativas se extendió a proyectos científicos que operaban con una gran cantidad de datos, lo cual les permitió prescindir en gran medida de los costosos servidores [8].

A mediados de 1996 el programador australiano Adam Hinkley desarrolló Hotline Connect para el sistema operativo Mac OS. Hotline Connect era una red P2P descentralizada para el intercambio de archivos entre empresas y universidades, la cual permitía almacenar los archivos a intercambiar en los computadores de aquellos usuarios que querían actuar como servidores, siendo estos últimos los que permitían, restringían o condicionaban el acceso del resto de usuarios a la red. En caso de que un servidor se cayera no existía ningún otro lugar del cual seguir descargando ese mismo archivo, y no quedaba más remedio que cancelar la descarga y empezar de cero en otro servidor. Este sistema en el que cada usuario dependía de un único servidor no tardó en quedar obsoleto, por otra parte, al ser una aplicación desarrollada fundamentalmente para una plataforma minoritaria como Mac OS, no atrajo la atención del público en general [9].

En mayo de 1999 el uso de las redes P2P se hizo masivo. Shawn Fanning y Sean Parker, estudiantes de la Northeastern University (EEUU) crearon Napster, una red P2P centralizada cuyo principal objetivo era intercambiar archivos de música en formato mp3. Aunque ya existían aplicaciones que permitían el intercambio de archivos mp3 como IRC y Usenet, Napster se presentó como la primera aplicación especializada en el intercambio de este tipo de archivos. Napster utilizaba servidores centrales para almacenar las listas de los usuarios y de los archivos que cada uno poseía. El programa se hizo famoso rápidamente pues cualquier usuario podía conectarse y descargar completamente gratis incluso los discos que aun no salían al mercado. Esto lo llevó a ser demandado por la RIAA (*Record Industry Association of America*), asociación que representa a la industria discográfica estadounidense. En el 2001 Napster perdió la batalla legal y se condenó a sus creadores al cierre total de la red en Julio de ese mismo año. Hoy en día Napster se ha convertido en un servicio pago al que la mayoría de internautas le dan la espalda [9].

Durante un tiempo el intercambio de archivos fue a la deriva, al principio se seguía utilizando Napster mediante servidores no oficiales a los que se podía acceder gracias a un programa llamado Napigator. También surgieron programas como Winmx (cerrado en 2005 por amenazas de la RIAA), e iMesh. Luego apareció Audiogalaxy, otra red P2P centralizada para el intercambio gratuito de música que también se cerró por orden judicial en Junio de 2002 [9].

Acabar con las redes P2P centralizadas era relativamente sencillo, solo bastaba cerrar los servidores que almacenaban las listas de usuarios y archivos compartidos y ya estaba. Sin embargo, y para evitar su continuo cierre, las redes P2P progresivamente se fueron perfeccionando para dar paso a redes P2P descentralizadas más sofisticadas. De esta manera ya no se dependía de servidores centrales, por lo que no se tenía constancia de los archivos que se compartían haciendo más difícil su control.

Con la aparición de Gnutella, Napster y Audiogalaxy quedaron obsoletas. Luego, en el año 2002, se produjo un éxodo masivo de los internautas hacia las redes P2P descentralizadas, entre las cuales estaban: Kazaa, Grokster, Piolet, Morpheus y Ares [9].
En el año 2003 se produjo un hecho que ayudó a la expansión de las redes P2P: Grokster y Morpheus que habían sido denunciadas por la RIAA ganaron el juicio en el 2003. De esta manera siguieron apareciendo redes P2P cada vez más complejas, como: Lphant, Shareaza, eMule, aMule y ML Donkey, las cuales hacían uso del protocolo eDonkey 2000. La utilización del protocolo BitTorrent, proporcionaba, según sus desarrolladores, una mayor velocidad de descarga que la que se alcanzaba con el protocolo eDonkey 2000, pero a costa de una menor variedad y longevidad de los archivos en la red [9].

En el año 2004 la compañía española MP2P Technologies, dirigida por Pablo Soto, emprende un ambicioso proyecto de investigación con el fin de dar inicio a la siguiente generación de aplicaciones P2P. El resultado es Omemo (http://www.omemo.com/es/), un sistema P2P cuya principal diferencia con los sistemas P2P tradicionales radica en el hecho de que lo que se comparte ya no son archivos si no espacio de almacenamiento. El sistema unifica el espacio libre que aporta cada usuario para formar un único disco virtual con una capacidad de almacenamiento nunca antes vista que permite almacenar datos a nivel mundial. La ventaja de la asignación de un espacio del disco duro para compartir información es que aunque el usuario se desconecte los datos permanecen disponibles en el disco global, por lo que hay una mayor disponibilidad de archivos, algo que no ocurre en los sistemas P2P convencionales.

Aunque el P2P ofrece grandes posibilidades en diversos campos de la ciencia y la tecnología, desde sus inicios fue usado principalmente para el intercambio de archivos que en su gran mayoría se encontraban protegidos por *copyright*, lo que inevitablemente lo llevó con el tiempo a ser asociado con la piratería.

El objetivo principal de aplicaciones como Kazaa, LimeWire o eDonkey no era mejorar la utilización de los recursos, la auto-organización o cualquier otro beneficio importante del paradigma P2P, sino mas bien permitir el acceso fácil y gratis a archivos de música o video.

En lo referente a acciones legales, el 12 de septiembre de 2006, MetaMachine, la compañía propietaria de eDonkey, se comprometió con la RIAA a pagar una multa de US$ 30 millones en un acuerdo extrajudicial para evitar posibles demandas de la industria discográfica; debido a esto, en el sitio web de eDonkey (http://www.edonkey.com/) se publicó un aviso que informaba sobre la ilegalidad de compartir música y vídeos que tuvieran *copyright*. De la misma manera el programa cliente eDonkey2000 dejó de funcionar, desplegando este mismo mensaje e iniciando su desinstalación automáticamente. No obstante, la red eDonkey 2000 no pudo ser cerrada, y aun se encuentra en funcionamiento [9].

Respecto al protocolo BitTorrent también han existido acciones legales en su contra. A finales de marzo de 2006, TorrentPluribrain, un buscador de Torrents, tuvo que cerrar sus servidores debido a una denuncia interpuesta por la *Société Civile des Producteurs Phonographiques* de París, pues, pese a que su desarrollador es español, los servidores se encontraban localizados en Francia.

Como se ha podido observar, las redes P2P fueron concebidas básicamente para incrementar la participación y mejorar la utilización de los recursos dispersos a lo largo y ancho de Internet. Por consiguiente, el futuro de las aplicaciones y de los protocolos P2P debe estar enfocado principalmente en mejorar el rendimiento, la escalabilidad y la adaptación a entornos heterogéneos.

2.2 Tipos de redes P2P

Las redes P2P se dividen básicamente en: redes centralizadas, redes puras o totalmente descentralizadas y redes híbridas.

2.2.1 Red centralizada

La primera generación de redes P2P empleaba una estructura de red similar a la del modelo Cliente/Servidor. Los servidores mantenían bases de datos con la información de los nodos y de los archivos que cada uno poseía. Cada vez que un nodo se conectaba o desconectaba de la red, las bases de datos se actualizaban. Los mensajes de búsqueda y control se enviaban a los servidores, los cuales comparaban las solicitudes de los nodos con el contenido de sus bases de datos y enviaban los resultados al nodo en cuestión, el cual, una vez informado, se contactaba directamente con el nodo poseedor del recurso.

Este tipo de redes tiene la ventaja de tener una administración dinámica y una disposición permanente de contenidos, pero presenta desventajas como la falta de privacidad de los usuarios y los inconvenientes que trae consigo el hecho de depender de servidores centralizados [10].

2.2.2 Red pura o totalmente descentralizada

La segunda generación de redes P2P utilizaba un modelo distribuido donde no existían servidores centrales y todos los nodos podían desempeñar los mismos roles. Cada nodo actuaba como servidor y como cliente a la vez (*servent*), manteniendo cierto número de conexiones con otros nodos. El tráfico de red estaba conformado principalmente por peticiones, respuestas y mensajes de control que facilitaban el descubrimiento de los nodos [10].

Las redes de este tipo no requieren de gestión central, lo cual les permite prescindir del uso de servidores; por consiguiente, la comunicación se establece directamente entre los nodos, los cuales actúan como puntos de enlace.

2.2.3 Red híbrida

Hoy en día la gran mayoría de aplicaciones P2P consideradas de tercera generación implementan un modelo mixto. Dentro de este modelo ciertos nodos actúan como super

nodos, ayudando a gestionar el tráfico dirigido hacia los demás miembros de la red. Además, los nodos cliente mantienen un pequeño número de conexiones abiertas, cada una de ellas a un super nodo. De igual forma, los super nodos están conectados entre si. Esta topología permite la escalabilidad de la red, pues reduce el número de nodos involucrados en el encaminamiento y manejo de los mensajes, y disminuye el volumen de tráfico entre ellos [10].

2.3 Características de los Sistemas P2P

El modelo P2P es el resultado natural de las tendencias de descentralización en la ingeniería de software. Utilizando la infraestructura actual, compuesta de redes, servidores y clientes, el P2P ofrece un modelo ortogonal al modelo Cliente/Servidor, dado que los dos modelos coexisten, se intersecan y se complementan [8].

El modelo P2P no es la panacea, pero si ofrece grandes ventajas; algunas de las características más relevantes de los sistemas P2P se describen a continuación:

2.3.1 Aprovechamiento de los recursos

Gran parte de los computadores de escritorio permanecen subutilizados la mayor parte del tiempo; desperdician potencia de procesamiento, capacidad de almacenamiento, entre otros. La gran mayoría son utilizados para llevar a cabo tareas que no requieren de grandes prestaciones, lo cual conlleva a un desaprovechamiento de tan valiosos recursos. Los sistemas P2P pueden hacer uso de ellos, incrementando así el valor de la red.

2.3.2 Descentralización

Uno de los objetivos primordiales de la descentralización es el énfasis que se hace en la propiedad y el control de los recursos por parte de los usuarios.

En una red completamente descentralizada cada nodo es igual a los demás, esto es, no existe una entidad central que administre los recursos del sistema.

2.3.3 Escalabilidad

La potencia de procesamiento de una red P2P es directamente proporcional al número de peers que la conforman, es decir, a mayor número de peers mayor es el poder de cómputo y viceversa.

2.3.4 Anonimato

Garantizar el anonimato de los usuarios es uno de los objetivos primordiales de los sistemas P2P.

Cuando se establece una comunicación se puede implementar el anonimato a nivel de remitente, de receptor o el anonimato mutuo, en el cual se ocultan las identidades del remitente y del receptor uno del otro y de los demás peers [11].

Existen determinadas técnicas [11] diseñadas para implementar alguno de los tipos de anonimato mencionados anteriormente, a continuación se describen las más relevantes:

- Multidifusión: se puede utilizar como mecanismo para implementar el anonimato de lado del receptor. Un grupo de multidifusión está conformado por usuarios que desean permanecer en el anonimato. Si un usuario está interesado en obtener un documento, se suscribe a un grupo y genera una nueva petición, luego, el usuario que posee dicho documento lo publica dentro del grupo. De esta manera la identidad del solicitante permanece oculta del remitente y de los demás miembros del grupo. Además, esta técnica puede tomar ventaja de redes subyacentes que soporten la multidifusión (Ethernet o Token ring).

- Falsear la dirección del remitente: para protocolos de comunicación como UDP se puede lograr el anonimato de lado del remitente falseando su dirección IP. Esto sin embargo requiere de cambios al protocolo, lo cual no siempre es factible, pues la mayoría de los ISP actuales filtran los paquetes que provienen de direcciones IP inválidas.

- Falsear la identidad: además de cambiar la dirección de origen, el anonimato se puede lograr cambiando la identidad de las partes que intervienen en la comunicación. En Freenet por ejemplo, un peer que provee un documento, que puede estar fuera de su propia caché o pertenecer a otro peer, puede reclamar su autoría. Por consiguiente un peer puede ser considerado inocente de proveer un documento por que existe la probabilidad de que el autor sea alguien más.

- Encubrir el camino: en vez de comunicarse directamente las partes lo pueden hacer a través de nodos intermedios, es decir, un peer que quiere ocultar su identidad crea un *camino encubierto* de peers, cuyo punto final es el peer con el cual desea establecer la comunicación. Se pueden obtener diferentes grados de anonimato variando la longitud de los caminos o cambiándolos con determinada frecuencia.

- Alojamiento involuntario: un enfoque interesante para implementar el anonimato consiste en alojar los documentos de forma involuntaria, es decir, un nodo que cree un documento puede forzar a otro nodo a que lo aloje en su caché. Debido a que el alojamiento es involuntario no siempre se puede responsabilizar a un nodo de la posesión de un documento.

Otra técnica consiste en dividir los archivos en múltiples componentes y almacenarlos en diferentes servidores. Los archivos solo pueden ser reconstruidos a su estado original mediante claves cifradas que son administradas por usuarios de confianza [12].

2.3.5 Rendimiento

Los sistemas P2P optimizan su rendimiento a medida que aumenta el número de peers conectados a la red, lo que incrementa la capacidad de almacenamiento y los ciclos de cómputo. Debido a la naturaleza descentralizada de estos sistemas, el rendimiento se ha visto influenciado por tres tipos de recursos: procesamiento, almacenamiento y la red. El ancho de banda es crucial cuando se propagan muchos mensajes o cuando los peers llevan a cabo la transferencia de una gran cantidad de archivos, lo cual puede limitar la escalabilidad del sistema [11].

Existen tres enfoques claves para optimizar el rendimiento: replicación, *caching* y enrutamiento inteligente.

1. Replicación: permite ubicar copias de archivos cerca de aquellos peers que los solicitan con mayor frecuencia. Por consiguiente, los cambios que se hagan sobre un archivo deben ser propagados a todas sus replicas. La distribución geográfica de los peers también ayuda a reducir la congestión de la red. En combinación con el enrutamiento inteligente, la replicación ayuda a disminuir los tiempos de respuesta al enviar las solicitudes a los peers más cercanos. La replicación también hace frente al problema de la desaparición de los peers, es decir, debido a que la gran mayoría de los peers son computadores personales en vez de servidores dedicados, no hay garantía de que estos permanezcan conectados a la red permanentemente [11].

2. *Caching*: reduce la longitud del camino que se debe recorrer para hallar un archivo y por consiguiente el número de mensajes que intercambian los peers. Reducir la transmisión de este tipo de mensajes es de suma importancia pues la latencia en la comunicación es un serio problema de rendimiento que enfrentan los sistemas P2P. El uso de réplicas ayuda a balancear la carga y a disminuir la latencia. En Freenet por ejemplo, cuando se encuentra un archivo y se lo propaga hasta el peer solicitante, dicho archivo se replica en la cache local de todos los peers que conforman el camino de propagación [11].

3. Enrutamiento inteligente: crear grupos de peers que compartan intereses comunes puede ayudar a reducir el número de mensajes que se transmiten por la red y a su vez el número de peers que procesan una solicitud antes de hallar un resultado. Sistemas como Pastry buscan optimizar su rendimiento moviendo regularmente los datos a través de la red. La ventaja de este enfoque es que los peers deciden con quién contactarse y cuándo establecer una conexión basándose solo en información local [11].

2.3.6 Seguridad

Gran parte de la información que viaja por una red P2P tiene un alto valor intrínseco para sus usuarios, por lo que su seguridad es un tema de considerable importancia.

Los sistemas P2P comparten la mayoría de sus necesidades de seguridad con los sistemas distribuidos, a saber: cadenas de confiabilidad entre peers y objetos compartidos, esquemas de intercambio de claves, cifrado, resúmenes digitales y firmas. Además, estos sistemas deben soportar diversos niveles de acceso a los recursos, por tal motivo se enfocan en aspectos tales como: confidencialidad, autenticación, autorización, integridad, y reusabilidad [11].

2.3.7 Transparencia y Usabilidad

La transparencia determina la usabilidad de un sistema P2P y su objetivo principal es proporcionar al usuario y a las aplicaciones una visión de los recursos del sistema como un todo, es decir, como si fueran gestionados por una sola máquina virtual. La distribución física de los recursos debe ser transparente [11].

La transparencia se puede dividir en:

- Transparencia de localización: los usuarios y las aplicaciones no conocen en cual peer reside el recurso accedido, o si éste es local o remoto, lo cual implica que el recurso puede migrar hacia distintos peers sin que las aplicaciones se vean afectadas.

- Transparencia de identificación: los espacios de nombres de los recursos son independientes de la topología de la red y de la propia distribución de los mismos.

- Transparencia de replicación: los usuarios y las aplicaciones no saben cuántas unidades hay de cada recurso, o si se añaden o eliminan copias del mismo.

- Transparencia de paralelismo: una aplicación puede ejecutarse en paralelo, sin que tenga que especificarlo, y sin consecuencias sobre la ejecución, salvo por cuestiones de rendimiento.

- Transparencia de concurrencia: el acceso simultáneo sobre un recurso compartido por parte de varias aplicaciones no debe afectar la ejecución de estas.

2.3.8 Tolerancia a fallas

Las fallas en un sistema P2P se deben principalmente a la incapacidad de dicho sistema para responder a las solicitudes de servicios de sus usuarios. El grado de tolerancia a las fallas determina la robustez de la red. Entre otros factores, las fallas se deben a solicitudes mal realizadas y contenidos inalcanzables o no disponibles.

La tolerancia a las fallas expresa la capacidad del sistema para seguir operando correctamente ante el mal funcionamiento de alguno de sus componentes, enmascarando la falla, lo que implica detectarla y continuar prestando el servicio, todo ello de manera transparente para al usuario [12].

2.3.9 Interoperabilidad

El concepto de interoperabilidad clásico hace referencia al acceso uniforme a múltiples fuentes de datos heterogéneas y autónomas. El objetivo de la interoperabilidad es intercambiar datos y funcionalidad, cooperando con un fin común [11].

Los principales problemas que se enfrentan al desarrollar un sistema integrado son: resolver la heterogeneidad de las fuentes, respetar su autonomía y diseñar el sistema de tal forma que se pueda administrar y mantener frente a cambios en la cantidad (escalabilidad) o en la estructura de las fuentes (evolución).

2.4 Componentes de los sistemas P2P

A continuación se describen los elementos esenciales de una red P2P.

2.4.1 Peers

Un peer es un nodo de una red P2P y es la unidad básica de procesamiento, capaz de ejecutar procesos y comunicar los resultados a otros peers en la red. En una conexión punto a punto un peer hace referencia a cada uno de los extremos.

2.4.2 Identificadores

En una red P2P debe existir un mecanismo que permita la identificación de los recursos de manera única, similar al que implementan aplicaciones P2P para el intercambio de archivos como Napster.

En una red P2P ideal los dispositivos deben ser capaces de participar sin tener en cuenta el sistema operativo o el transporte de red. Es así como un esquema de designación de entidades independiente del sistema es un requisito indispensable para el desarrollo de redes P2P flexibles.

2.4.3 Protocolos

El intercambio de datos depende de un protocolo que dictamina qué datos y en qué orden se envían. En este contexto, un protocolo es un conjunto de normas que permiten el

intercambio de información entre dos o más peers mediante reglas previamente definidas y acordadas por las partes.

2.4.4 Servicios

Los servicios hacen referencia a las funciones y a la capacidad que los peers ofrecen a la red P2P. Los servicios son el motivo por el cual los peers se reúnen en grupos; sin ellos solo se tendría un conjunto de nodos incapaces de cooperar mutuamente y de compartir sus recursos.

Gran parte de la funcionalidad requerida para desarrollar una red P2P la proporcionan los protocolos de red. Tales protocolos, también considerados como servicios, brindan los mecanismos esenciales para la construcción de servicios P2P más complejos.

2.4.5 Capa de transporte

La capa de transporte encapsula el envío y la recepción de mensajes entre los peers y es la responsable de todos los aspectos de la transmisión de datos. La capa de transporte esta compuesta de protocolos de bajo nivel como UDP o TCP, y de protocolos de alto nivel como HTTP o SMTP.

2.5 Tipos de Aplicaciones P2P

Existen muchas aplicaciones que se han desarrollado con base en la tecnología P2P las cuales se pueden enmarcar en: Mensajería instantánea, administración y distribución de información y colaboración.

2.5.1 Mensajería instantánea

Las aplicaciones de mensajería instantánea son programas regularmente gratuitos, de fácil instalación y uso y que requieren de una conexión a Internet para su activación. Una de las aplicaciones más reconocidas y que se puede considerar como la pionera en el mundo de la mensajería instantánea es ICQ[1], creada a finales de los años 90 por la empresa de software israelí Mirabilis. La arquitectura de ICQ es un híbrido entre las arquitecturas Cliente/Servidor y P2P (Figura 1), ya que aunque se utiliza un servidor central para inspeccionar a los usuarios que se encuentran conectados y para notificar a aquellos interesados en la conexión de nuevos usuarios, la comunicación entre ellos se establece directamente, es decir, sin la intermediación del servidor.

1 ICQ significa “I seek you” o “te busco”.

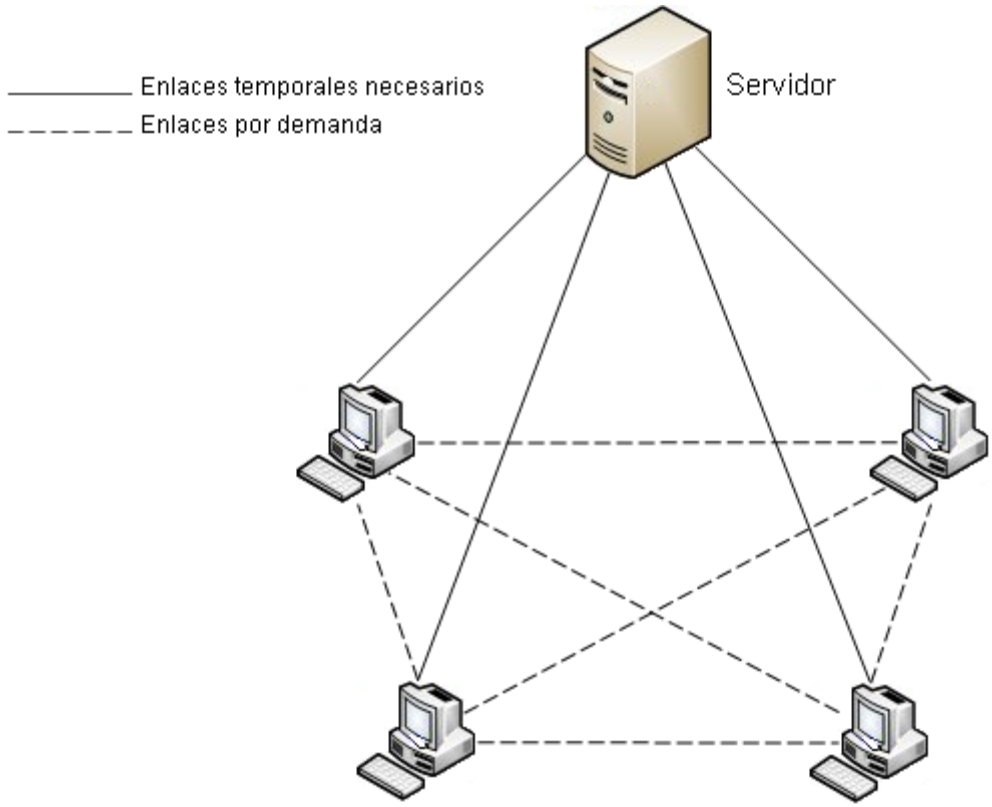


Figura 2. Red ICQ

En los últimos años han surgido diferentes aplicaciones de mensajería instantánea que hoy en día son muy populares como: MSN Messenger, AOL Internet Messenger, Yahoo Messenger, entre otras. Una de las limitaciones de estas aplicaciones es la poca o nula interoperabilidad entre ellas, ya que cada una implementa su propio protocolo que por lo general es incompatible con los demás. Sin embargo, existen soluciones que buscan lidiar con este problema como es el caso de Jabber, el cual proporciona los mecanismos necesarios para que los usuarios de diferentes sistemas de mensajería instantánea puedan interactuar entre sí.

2.5.2 Administración y distribución de información

El intercambio de archivos es el área más popular en el mundo del P2P cuya aplicación insignia es el controvertido Napster, gracias al cual se podían intercambiar archivos de música en formato mp3 completamente gratis. Napster empleaba una arquitectura de red similar a la de ICQ, la cual empleaba servidores centralizados que mantenían bases de datos con la información y la ubicación de los archivos disponibles pero no los archivos en sí; en otras palabras, el servidor se limitaba a responder a las consultas acerca de los archivos existentes y la dirección de la máquina que los contenía, pues el proceso de descarga se llevaba a cabo directamente entre los usuarios.

Otras variantes de aplicaciones P2P para el intercambio de archivos son: Gnutella, Freenet, MojoNation, entre otras.

2.5.3 Colaboración

Este tipo de aplicaciones permiten que los miembros de un grupo de usuarios puedan establecer comunicación de manera colaborativa posibilitando la organización y la coordinación de espacios de trabajo en línea para la realización de proyectos [13].

En teoría la colaboración incrementa la productividad, disminuyendo el tiempo que supone la realización de múltiples revisiones por parte de los participantes de un proyecto, permite que los equipos trabajen en entornos distribuidos geográficamente y disminuye la necesidad de almacenamiento en servidores centralizados utilizando para ello la replicación automática.

El rango de utilidades suministradas por las aplicaciones de colaboración puede ir desde la mensajería instantánea hasta la videoconferencia, pasando por los juegos en línea y la telefonía IP. Uno de los mejores ejemplos de una aplicación P2P colaborativa es Skype, el cual ofrece servicios de voz sobre IP (VoIP)[2] y mensajería instantánea.

2.6 Conclusiones

La descentralización es un concepto fundamental dentro de la computación P2P que influye directamente en el diseño de las aplicaciones y en aspectos tales como la seguridad, escalabilidad, disponibilidad y las estructuras de datos.

La naturaleza ad-hoc de los sistemas P2P también afecta la forma en la que éstos se conciben, es decir, el hecho de que un usuario pueda aparecer o desaparecer en cualquier momento, sin previo aviso, es un factor determinante en el desarrollo de sistemas P2P.

Adicional a las características de seguridad clásicas de los sistemas distribuidos, los sistemas P2P se distinguen por su anonimato, escalabilidad, rendimiento, tolerancia a fallas e interoperabilidad.

Las aplicaciones P2P para la distribución de información centran sus esfuerzos en asegurar altos niveles de disponibilidad de datos. Las aplicaciones P2P colaborativas y de mensajería instantánea se preocupan por la conectividad (naturaleza ad-hoc), seguridad, anonimato e interoperabilidad.

2 Protocolo de Internet también llamado Voz sobre IP, es un grupo de recursos que hacen posible que la señal de voz viaje a través de Internet empleando el protocolo IP.

CAPITULO 3

ESTUDIO COMPARATIVO DE LAS PLATAFORMAS IDENTIFICADAS

La mayoría de las aplicaciones P2P mencionadas en la sección 2.5 del capítulo 2 brindan soluciones particulares a problemas análogos, aplicaciones cuyas diferencias radican fundamentalmente en los protocolos que implementan, los cuales, por su naturaleza propietaria, son incompatibles unos con otros, dando lugar a comunidades aisladas cuyos miembros son incapaces de interactuar con usuarios de otras aplicaciones, limitando así la interoperabilidad.

Aunque existen soluciones que integran múltiples servicios bajo el paradigma lógico P2P, estas hacen uso de servidores centralizados, además su código es cerrado, lo que impide el estudio detallado de los protocolos de red que implementan. Por consiguiente, surge la necesidad de crear una plataforma para el desarrollo de aplicaciones P2P interoperables y de propósito general, de tal forma que los desarrolladores no tengan que ocuparse más en el diseño de los protocolos de red. Es así como surgen las siguientes plataformas:

3.1 Plataformas P2P

3.1.1 Anthill

Anthill [15] nace como una plataforma para soportar el diseño, la implementación y la evaluación de aplicaciones P2P resistentes y adaptables. Para tal fin se basa en el paradigma de los sistemas multi-agente (MAS[1]) y la programación evolutiva, tomada de los sistemas adaptativos complejos. En teoría un sistema Anthill esta compuesto de una red dinámica y adaptable de agentes que viajan por la red, interactuando y cooperando entre sí con el fin de solucionar problemas complejos.

Se desarrolló un prototipo Java del entorno de ejecución de Anthill con base en JXTA, lo cual trajo consigo múltiples beneficios, tales como la utilización de diferentes capas de transporte para la comunicación, incluyendo TCP/IP y HTTP. Dicho prototipo permitía además, ocuparse de aspectos relacionados con Firewalls y NATs [15]. Adicionalmente Anthill incluía un ambiente de simulación para ayudar a los desarrolladores a evaluar y analizar el comportamiento de las aplicaciones P2P.

En la actualidad el desarrollo de Anthill se ha suspendido para dar paso a PeerSim [16], un simulador más escalable y liviano escrito igualmente en Java, cuyo desarrollo hace parte del proyecto BISON[2] [17] de la Comisión Europea. El objetivo del proyecto BISON es explorar el uso de ideas derivadas de los sistemas adaptativos complejos, que permitan la construcción de sistemas de información robustos y con la capacidad de auto organizarse, los cuales serán desplegados en entornos de red altamente dinámicos como los sistemas P2P.

[1] Multi-Agent Systems: colección de agentes autónomos capaces de observar su entorno y realizar cálculos simples basados en dichas observaciones.
[2] Biology-Inspired techniques for Self Organization in dynamic Networks.

3.1.2 GNUnet

GNUnet es una plataforma para el desarrollo de aplicaciones P2P seguras, anónimas y descentralizadas, enfocadas a la distribución de contenidos [18]. Como su nombre lo deja entrever, GNUnet hace parte del proyecto GNU.

Esta plataforma utiliza un modelo simple y económico basado en el exceso para la asignación de los recursos, es decir, cuando los recursos son escasos, se asignan con base en el comportamiento previo de los peers, incluyendo los periodos de abundancia de recursos. La información sobre dichos periodos es utilizada para inducir a la economía con confianza, la cual hace referencia al hecho de prevenir el abuso de la red, detectando a aquellos peers que la utilizan sin aportar nada a cambio para limitar su impacto y recompensando a aquellos que si contribuyen a su buen funcionamiento con la prestación de un mejor servicio [19].

La idea básica es simple, un peer guarda un registro de las transacciones previas llevadas a cabo con otros peers con el fin de identificar a los que presentan un buen comportamiento. Aquellos peers que contribuyen constantemente a la red se ganan la confianza de sus peers asociados. Si un peer se halla en un ambiente donde los recursos son escasos, entonces utiliza la información almacenada en su registro para determinar cuáles solicitudes debe atender y cuáles debe ignorar [19].

Para el intercambio de mensajes, los peers utilizan un servicio de transporte basado en protocolos como UDP, TCP, HTTP y SMTP, los cuales se enmarcan dentro de tres contextos diferentes [20]:

1. La primera y más importante familia de protocolos corresponde a los protocolos *peer-to-peer*, los cuales definen los mecanismos para el intercambio de información entre peers, mensajes para el descubrimiento de peers, cifrado, disponibilidad y mensajes de aplicación específicos. Todos los peers en la red deben soportar un determinado subconjunto básico de mensajes *peer-to-peer*, los cuales se intercambian mediante un demonio[3] denominado *gnunetd*. Los mecanismos para el intercambio de mensajes se encuentran encapsulados en los servicios de transporte.

2. La segunda familia de protocolos corresponde a los protocolos Cliente/Servidor, los cuales se utilizan entre las herramientas GNUnet y el *gnunetd*. GNUnet utiliza una conexión TCP, típicamente vía *loopback*, para permitir que los clientes envíen solicitudes al *gnunetd*. Para estos clientes, también conocidos como herramientas GNUnet, *gnunetd* actúa como servidor y no como peer. GNUnet no cifra el tráfico TCP que fluye entre el cliente y el servidor, pues esta conexión no esta permitida, a menos que sea a través de una LAN de confianza o vía *loopback*. En el archivo de configuración se especifican los clientes a los cuales se les permite conectarse al *gnunetd*.

3. La tercera familia de protocolos GNUnet es utilizada en la capa de transporte, en la cual se encapsula el envío y la recepción de mensajes *peer-to-peer*. Cada peer

[3] En sistemas UNIX/Linux se le llama así a un programa que se ejecuta continuamente en segundo plano para brindar algún tipo de servicio.

> GNUnet debe soportar al menos un protocolo de transporte. Los protocolos de transporte difieren en el rendimiento, los requerimientos del sistema y la facilidad de uso. Cada protocolo de transporte también define su propio formato para las direcciones de los peers. Dichas direcciones pueden ser URLs, correos electrónicos o combinaciones del tipo IP:PUERTO. Cada peer cuenta con un único identificador que se obtiene al aplicar una función *hash* a su clave pública.

Cuando un peer desea establecer comunicación con otro, selecciona un par de claves, las cuales son utilizadas para la identificación, autenticación y el cifrado de esta. Además, la comunicación es confidencial, es decir, ningún peer fuera de la red puede observar los datos que viajan a través de ella. Para ocultar a los peers que intervienen en la comunicación, GNUnet utiliza un enfoque que consiste en utilizar peers intermediarios, es decir, peers que retransmiten mensajes a otros peers [21].

En aras de un alto nivel de anonimato, los desarrolladores de GNUnet sacrifican en parte el rendimiento. Aunque teóricamente es posible que una descarga mediante GNUnet sea incluso más rápida que la de un usuario conectado mediante módem o la que se realiza desde un servidor saturado, en la práctica, esto depende fundamentalmente de la forma en la que se encuentre distribuido el contenido a través de la red [18].

El protocolo GAP (*GNUnet's Anonymity Protocol*) brinda los mecanismos necesarios para llevar a cabo la transferencia de archivos de forma anónima y segura, ocultando la identidad de los peers que intervienen en la comunicación, unos de otros y de otras entidades, incluyendo *routers* y adversarios activos y pasivos [22].

GNUnet utiliza un mecanismo de codificación para el intercambio de archivos que consiste en transferir bloques de tamaño fijo (1Kb), utilizando para ello el protocolo GAP. El propósito de dividir la información en bloques de tamaño fijo es ocultar el tipo de datos que viajan a través de la red; dicha división se lleva a cabo utilizando tres tipos de bloques: *Data Blocks*, *Indirection Blocks* y *Root Blocks* [21].

Cuando se envía un archivo se divide en múltiples *Data Blocks*, para cada uno de los cuales se calcula un valor *hash* de 160 bits. Luego un árbol de *Indirection Blocks* es calculado recursivamente, donde cada *Indirection Block* contiene, entre otros, un máximo de 25 valores *hash* de consulta. Finalmente se genera el *Root Block*, el cual contiene la descripción del archivo y el *hash* de consulta para obtener la raíz del árbol de *Indirection Blocks* [21].

Para extraer el contenido del archivo se llevan a cabo múltiples solicitudes de búsqueda que utilizan listas de tres palabras clave para descubrir los *Root Blocks*; un peer que reciba una consulta y tenga almacenado un *Root Block* con uno de esos valores, lo retorna cifrado junto con un valor *hash* para probar que efectivamente posee el contenido solicitado; luego de que un *Root Block* es descubierto, el peer puede descargar el archivo correspondiente mediante la emisión de múltiples solicitudes de descarga. El peer utiliza el *hash* de consulta incluido en el *Root Block* para obtener y descifrar el *Indirection Block* raíz; luego utiliza los valores *hash* de consulta incluidos en este último para obtener y descifrar todos los *Indirection Blocks* adicionales; finalmente los valores *hash* de consulta incluidos en los *Indirection Blocks* son utilizados para obtener y descifrar los *Data Blocks* del archivo [21].

Como se puede observar, existen dos tipos de solicitudes: solicitudes de búsqueda y solicitudes de descarga. Ambos tipos contienen tres valores adicionales que son utilizados para procesar la solicitud: una dirección de retorno, una prioridad y un TTL[4].

Un problema inherente al diseño de GNUnet se deriva del hecho de tener que dividir los archivos en múltiples partes asociadas. Al descargar un archivo las partes se obtienen separadamente, comportamiento del que puede valerse un atacante para interceptar dichas partes y así poder determinar quien inició la descarga. Otro problema tiene que ver con la resistencia a la censura; GNUnet es vulnerable a los ataques del tipo *rubber-hose*[5] [21].

3.1.3 Gnutella

Gnutella, más que una plataforma, es un protocolo abierto para la búsqueda e intercambio de información y para la creación de grupos descentralizados de usuarios. Aunque el protocolo Gnutella soporta el modelo tradicional de búsqueda Cliente/Servidor, lo que realmente lo distingue es su modelo P2P descentralizado (Figura 3).

En una red Gnutella un nodo es capaz de realizar las tareas de un servidor y un cliente a la vez, razón por la cual se le denomina *servent*[6]. Los *servents* proporcionan interfaces a través de las cuales se pueden generar consultas, analizar los resultados y aceptar las consultas de otros *servents*, verificando las coincidencias en su conjunto local de datos, y de hallarlas, responder con la información solicitada. Estos *servents* también son los responsables de manejar el tráfico que propaga la información utilizada para mantener la integridad de la red [23].

Debido a su naturaleza descentralizada, una red de *servents* que implementan el protocolo Gnutella es altamente tolerante a fallos, es decir, el funcionamiento de la red no se verá interrumpido si un subconjunto de *servents* se desconecta [23].

Con el fin de unirse a la red, un nuevo *servent* inicialmente se conecta a uno o varios *servents* que ya hagan parte de ella y que por lo general casi siempre se encuentran disponibles (por ejemplo en http://gnutellahosts.com) [23]. Cuando el *servent* se ha unido a la red empieza enviando mensajes para interactuar con otros *servents*.

[4] TTL (Time To Live – Tiempo de vida): contador en el interior de los mensajes que determina su propagación y especifica cuántos *hops* (saltos) puede dar, antes de ser descartado o devuelto.
[5] Proceso mediante el cual se extraen las claves cifradas de los usuarios.
[6] Palabra derivada de la conjugación de los términos ingleses *server* y *client*.

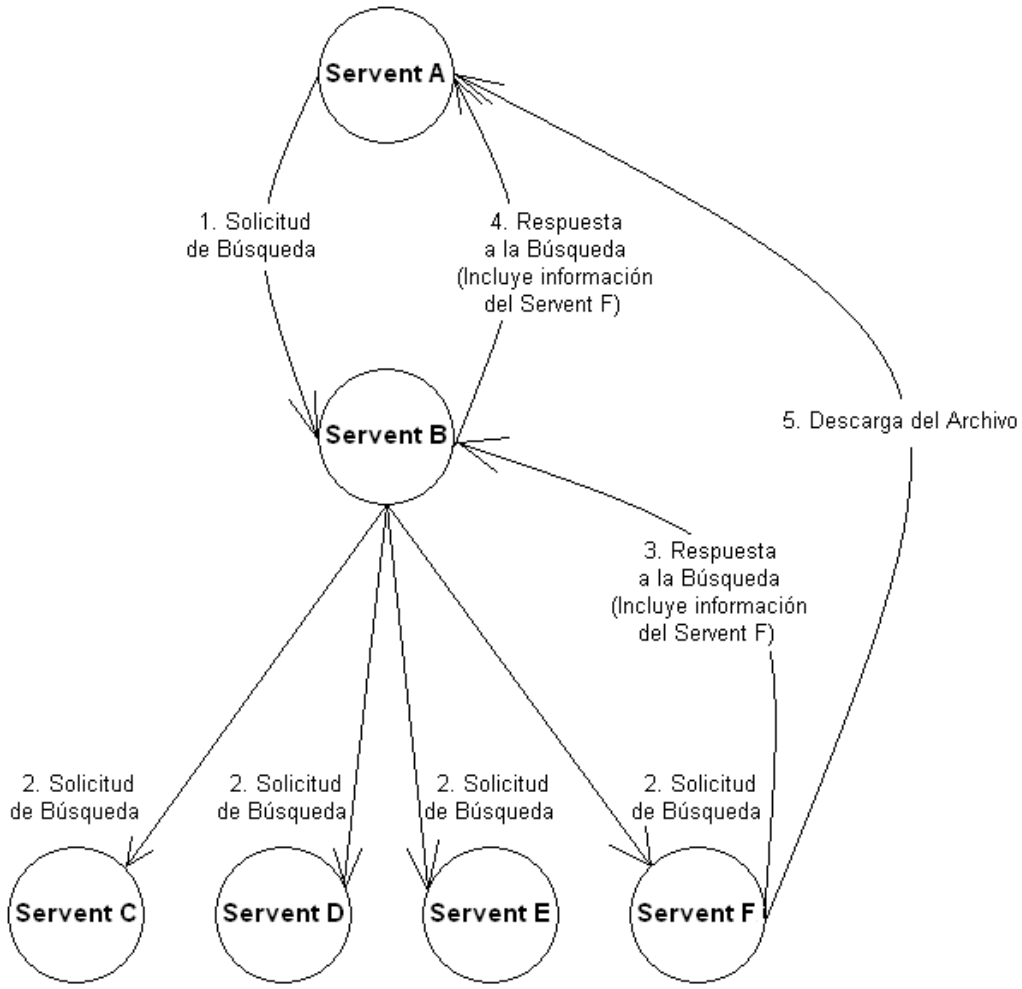


Figura 3. Modelo P2P descentralizado de la red Gnutella

El protocolo Gnutella brinda los mecanismos necesarios mediante los cuales los mensajes pueden ser “difundidos” (enviados a todos los *servents* con los cuales el *servent* remitente tiene conexiones TCP abiertas) o “propagados hacia atrás” (enviados de regreso a un *servent* específico a través de la ruta inicial utilizada para difundir el mensaje) [23].

Primero, cada mensaje tiene un identificador generado al azar. Segundo, cada *servent* almacena en su memoria información sobre los mensajes difundidos recientemente para evitar volver a enviarlos y para llevar a cabo la propagación hacia atrás. Tercero, los mensajes se marcan con un TTL y con información de la ruta a través de la cual han sido propagados.

Los mensajes existentes en una red Gnutella son [23]:

- Mensajes para unirse a la red (*PING*, *PONG*): Un *servent* que se une a la red inicia enviando mensajes tipo *PING* para anunciar su presencia. Cuando un *servent* recibe un mensaje *PING* lo reenvía a todos sus *servents* asociados y crea un mensaje *PONG* utilizado para llevar a cabo la propagación hacia atrás. El mensaje *PONG* contiene información acerca del *servent* que responde, como su dirección

IP y el número y tamaño de los archivos que comparte.

- Mensajes de búsqueda (*QUERY*, *QUERY RESPONSE*): Los mensajes tipo *QUERY* son difundidos y contienen una cadena de búsqueda definida por el usuario, la cual es contrastada con los archivos almacenados localmente por los *servents* que reciben dicho mensaje. Los mensajes tipo *QUERY RESPONSE* son propagados hacia atrás como respuesta a los mensajes tipo *QUERY* e incluyen la información necesaria para descargar un archivo.

- Mensajes para la transferencia de archivos (*GET*, *PUSH*): La descarga de archivos se lleva a cabo directamente entre los *servents* interesados utilizando para ello mensajes tipo *GET* y *PUSH*.

Aunque es muy útil para el descubrimiento, el diseño del mensaje tipo *PONG* pone en riesgo el anonimato de los usuarios de la red Gnutella, ya que a través de éste mecanismo se difunden públicamente las direcciones IP de los *servents* que ya hacen parte de ella, y de las cuales se puede valer un usuario mal intencionado para, entre muchas otras cosas, comprometer o suplantar la identidad de otro usuario [24].

La naturaleza de la red Gnutella requiere que los *servents* encaminen el tráfico de red (representado en los diferentes tipos de mensajes) de acuerdo a las siguientes reglas [25]:

1. Los mensajes tipo *PONG* sólo pueden ser enviados de regreso por la misma ruta que recorre el mensaje tipo *PING*.

2. Los mensajes tipo *QUERY RESPONSE* sólo pueden ser enviados de regreso por la misma ruta que recorre el mensaje tipo *QUERY*.

3. Los mensajes tipo *PUSH* sólo pueden ser enviados de regreso por la misma ruta que recorre el mensaje tipo *QUERY RESPONSE*. Un *servent* que reciba un mensaje tipo *PUSH* con un identificador x, descartará todos aquellos mensajes tipo *QUERY RESPONSE* cuyo identificador difiera de x.

4. Un *servent* reenviará todos los mensajes tipo *PING* y *QUERY* a todos sus *servents* asociados, excepto al que esta generando dichos mensajes.

5. Un *servent* debe decrementar el TTL e incrementar el número de *hops* antes de reenviar el mensaje a cualquiera de sus *servents* asociados. Si después de decrementar el valor del TTL este llega a cero, el mensaje no se sigue reenviando.

6. Un *servent* que reciba un mensaje con el mismo identificador de uno que haya recibido previamente, debe evitar reenviarlo a sus *servents* asociados, pues se supone que ya ha recibido tal mensaje, evitando de esta manera desperdiciar el ancho de banda de la red.

No siempre es posible establecer una conexión directa con un *servent* con el fin de descargar un archivo. El *servent* puede hallarse detrás de un *firewall* que impide las conexiones entrantes hacia su puerto Gnutella. Si no se puede establecer la conexión, el *servent* que desea descargar el archivo puede solicitar que este se comparta, utilizando

para ello un mensaje tipo *PUSH*. Un *servent* puede solicitar un archivo enviando de regreso un mensaje tipo *PUSH* al *servent* que envió el mensaje tipo *QUERY RESPONSE* describiendo el archivo a descargar. El *servent* al que se le envía la solicitud *PUSH* debería, a partir de la recepción del mensaje tipo *PUSH*, tratar de establecer una nueva conexión TCP/IP con el *servent* solicitante. Si esta conexión no se puede establecer, entonces es muy probable que el *servent* que emitió la solicitud *PUSH* también se encuentre detrás de un *firewall*, en cuyo caso la transferencia del archivo no se puede llevar a cabo; de lo contrario, se inicia el proceso de descarga [25].

En ambientes dinámicos, donde opera Gnutella, los *servents* se unen o abandonan la red intempestivamente y las conexiones son poco fiables. Para enfrentar esta situación, después de unirse a la red, un *servent* envía periódicamente mensajes tipo *PING* a sus *servents* asociados con el fin de descubrir nuevos *servents*. Esta información es muy útil, pues permite que el *servent* se reconecte a la red en caso de un fallo inesperado [23].

Los *servents* deciden dónde conectarse basados en información local, de esta manera se crea una red dinámica y auto organizada de entidades independientes. En definitiva, una red Gnutella cuenta con *servents* como sus nodos y con conexiones TCP abiertas como sus *links* [23].

Los protocolos que se acaban de estudiar, además de presentar características arquitecturales muy diferentes, solo brindan los mecanismos necesarios para el desarrollo de aplicaciones P2P de propósito específico, aplicaciones que aunque son conceptualmente análogas, difieren profundamente en los detalles de implementación. Esta situación impide la reutilización de software, disciplina fundamental de la Ingeniería del Software que busca, entre otras cosas, reducir el tiempo de desarrollo, los costos derivados de mantenimiento y mejorar la calidad de los sistemas a desarrollar [26].

3.1.4 JXTA

Consientes de esta problemática, Bill Joy y Mike Clary de Sun Microsystems crean el Proyecto JXTA, el cual es concebido para estandarizar un conjunto común de protocolos para el desarrollo de aplicaciones P2P, a través de los cuales se establece una red virtual sobre la red física subyacente, independientemente de su topología. La red virtual permite estandarizar la forma en que los *peers* se descubren unos a otros, se auto organizan en grupos, publican y descubren recursos, y se comunican y monitorean entre si [27].

Básicamente, JXTA es un conjunto de protocolos abiertos, codificados en XML, que le permiten a cualquier dispositivo conectado a la red, comunicarse y colaborar al estilo P2P.

El objetivo principal del proyecto JXTA es proporcionar los protocolos básicos necesarios para el desarrollo de aplicaciones y servicios P2P de propósito general, que además de poder ejecutarse en cualquier dispositivo con un corazón digital, sean interoperables, independientes del lenguaje de programación, del sistema operativo y de la red P2P [28].

Es importante notar que los requerimientos de seguridad en JXTA están influenciados por características muy particulares. Primero que todo, JXTA es neutral a los esquemas de criptografía y a los algoritmos de seguridad, esto con el fin de maximizar la flexibilidad y evitar la redundancia. Además, JXTA es una plataforma que se centra en mecanismos y

no en políticas [27], por lo tanto, a diferencia de la mayoría, JXTA no implementa un modelo específico de seguridad de alto nivel como Bell-LaPadula, flujo de información o muralla china; sin embargo, proporciona la funcionalidad básica para crear aplicaciones seguras. Los grupos de *peers*, por ejemplo, posibilitan la creación de entornos seguros, ya que permiten entre otras cosas, establecer mecanismos de seguridad (login/password, PKI, etc.) y publicar contenidos protegidos de acceso restringido; además, los desarrolladores de JXTA han dejado abierta la posibilidad de extender el API para implementar mecanismos que permitan desarrollar sistemas más complejos enfocados a soluciones de seguridad específicas [29].

En una red JXTA los *peers* se comunican entre si a través de canales de comunicación virtuales denominados *pipes*, por consiguiente, tales *pipes* deben asegurar la integridad y la confidencialidad de los datos que viajan a través de ellos, para lo cual se dispone de una amplia gama de alternativas. Una es utilizar VPNs para mover todo el tráfico de red. Otra es crear una versión segura del *pipe*, similar a un túnel protegido, de tal manera que todos los mensajes transmitidos a través de él, estén automáticamente asegurados. Una tercera alternativa es utilizar los mecanismos regulares de comunicación permitiendo que el desarrollador proteja los datos mediante técnicas de cifrado y firmas digitales. Adicionalmente, el anonimato puede garantizarse mediante los servicios de nombrado, de autenticación o de proxy, o mediante alguna de sus posibles combinaciones. JXTA es independiente de la solución elegida por una aplicación particular y puede acomodarse a cualquiera de estas alternativas [29].

Como se mencionó anteriormente, los *peer groups*, además de posibilitar la creación de entornos de computación seguros, permiten el enrutamiento inteligente, esto es, al compartir intereses comunes, los *peers* de un mismo grupo ayudan a reducir la carga de procesamiento de los *peers rendezvous*; de esta forma el tráfico total de la red también se reduce pues se disminuyen los mensajes que viajan a través de ella y por consiguiente los *peers* que procesan una solicitud antes de hallar un resultado. De esta manera el ancho de banda no se desperdicia y los tiempos de respuesta disminuyen considerablemente, lo cual lleva a niveles óptimos el rendimiento de la red.

Dentro de un *peer group* sólo puede existir una instancia de un servicio asociada a un *peer* específico, sin embargo, varias instancias del mismo servicio se pueden instalar de forma redundante en múltiples peers. Este tipo de servicios se denominan *peer group services* y son la clave de la alta disponibilidad y la tolerancia a las fallas de una red JXTA [30].

Como se ha podido observar, gracias a que se centra en mecanismos y no en políticas, la plataforma JXTA permite el desarrollo de una gran variedad de aplicaciones y servicios P2P, ya sea de propósito general o específico; además, por ser *open source*, el proyecto JXTA se encuentra en constante evolución para hacer frente a los nuevos retos que plantea la computación P2P.

3.1.4.1 P2PS: Una plataforma derivada de JXTA

P2PS (*Peer-to-peer Simplified*) [31] es una plataforma liviana para el descubrimiento de servicios P2P y la comunicación basada en *pipes*, cuya arquitectura y funcionalidad están inspiradas en JXTA. Como su nombre lo sugiere, P2PS busca proporcionar una

plataforma simple en la cual se puedan desarrollar aplicaciones P2P de propósito general, ocultando la complejidad de plataformas similares como JXTA. Gracias a que su infraestructura se basa en XML, P2PS permite que la implementación de los servicios de descubrimiento y comunicación sea independiente del lenguaje de programación. Además, la comunicación no se encuentra ligada a un único protocolo, como TCP/IP, lo que permite la transmisión de mensajes a través de múltiples protocolos.

Las diferencias entre P2PS y JXTA radican más en la forma en que se construye la red que en los conceptos fundamentales [32], a saber:

- En JXTA existe un protocolo para la publicación de anuncios y otro para el envío de consultas. En P2PS una consulta es al mismo tiempo un anuncio y por consiguiente se publican del mismo modo.

- P2PS proporciona un lenguaje de descubrimiento más versátil que el de JXTA, permitiendo que los peers soliciten cualquier tipo de anuncio en lugar de restringirlos a tipos predefinidos.

- JXTA y P2PS permiten la transmisión de mensajes a través de múltiples protocolos. En JXTA el enrutamiento se expresa fundamentalmente con mensajes XML, mientras que en P2PS es una cuestión de implementación del protocolo *endpoint resolver*[7].

- En P2PS todos los servicios son instancias remotas que se ponen en contacto a través de *pipes* P2PS. En JXTA también existen los servicios remotos y los peers pueden importar el código del servicio mediante la implementación del módulo.

Aunque P2PS y JXTA son muy similares, la diferencia fundamental y determinante radica en que P2PS no brinda los mecanismos necesarios para la creación y manipulación de grupos de *peers* y sus mecanismos de seguridad asociados [32].

3.2 Comparación entre las plataformas identificadas

A continuación se realiza una evaluación de las características más relevantes de los sistemas P2P (ver capítulo 2, sección 2.3) para cada una de las plataformas identificadas en la sección anterior (sección 3.1).

Gnutella se caracteriza por su modelo P2P altamente descentralizado, lo cual le confiere un alto nivel de tolerancia a las fallas y una gran escalabilidad. En cuanto al anonimato, el diseño del mensaje tipo PONG, utilizado para realizar la propagación hacia atrás, pone en riesgo la identidad de los usuarios de la red, pues es mediante este mecanismo que se difunden públicamente las direcciones IP de los *servents*, las cuales pueden ser utilizadas para fines dañinos, lo cual también pone en riesgo la seguridad. Por otro lado, el rendimiento de una red Gnutella puede verse seriamente comprometido debido al constante flujo de mensajes tipo PING y PONG, los cuales consumen gran parte del ancho de banda de la red.

[7] Protocolo utilizado para establecer canales de comunicación virtuales (*pipes*) entre uno o más peers.

GNUnet ha sido diseñada con el objetivo de permitir el desarrollo de aplicaciones P2P enfocadas a la seguridad, el anonimato y la descentralización. En cuanto a la seguridad, cuando un peer desea establecer comunicación con otro, selecciona un par de claves que utiliza para su identificación, autenticación y cifrado. El anonimato se logra mediante la utilización de peers intermediarios utilizados para la retransmisión de mensajes a otros peers, esto es, se utiliza una técnica para *encubrir el camino* entre dos o mas peers que quieran establecer comunicación. Aunque esta es una buena alternativa para alcanzar un alto nivel de anonimato, el rendimiento de la red disminuye debido a la complejidad de las conexiones. Por otro lado, la descentralización se obtiene gracias a la *economía con confianza*, la cual se utiliza para brindar un mejor servicio a aquellos peers que contribuyen considerablemente al buen funcionamiento de la red y mediante la cual se incrementan los niveles de escalabilidad y de tolerancia a las fallas.

Anthill apareció en escena en septiembre de 2001 como una plataforma que soportaba el diseño, la implementación y la evaluación de aplicaciones P2P. En agosto de 2002 se publico una versión Java de Anthill que incluía algunos componentes de JXTA; esto limitó su interoperabilidad pues se ligó a un lenguaje de programación en particular, sin embargo trajo consigo los siguientes beneficios: la utilización de diferentes capas de transporte para la comunicación (TCP/IP, HTTP) y los mecanismos necesarios para lidiar con Firewalls y NATs. Además, Anthill contenía un ambiente de simulación que permitía evaluar y analizar el comportamiento de las aplicaciones P2P. Poco tiempo después el desarrollo de Anthill se detuvo, razón por la cual existe muy poca información sobre las particularidades de esta plataforma.

JXTA es una plataforma concebida para estandarizar una serie de protocolos comunes para el desarrollo de aplicaciones P2P de propósito general, que sean interoperables e independientes del lenguaje de programación, del sistema operativo y de la topología de red subyacente. En lo que respecta a la seguridad, JXTA es neutral a los esquemas de criptografía y a los algoritmos de seguridad, esto con el fin de maximizar la flexibilidad proporcionando la funcionalidad básica necesaria para la creación de aplicaciones P2P seguras. Los grupos de peers, por ejemplo, facilitan la creación de entornos seguros mediante la utilización de mecanismos como login/password o PKI. Además se pueden utilizar VPNs o crear versiones seguras de los *pipes* que aseguren la integridad y la confidencialidad de los datos que viajan por la red. Los grupos de peers también son el mecanismo fundamental para alcanzar la escalabilidad y la descentralización características de JXTA. El anonimato se puede lograr mediante la utilización de los servicios de nombrado, de autenticación o de proxy. Por otro lado, el enrutamiento inteligente que se lleva a cabo dentro de los grupos de *peers* y que permite reducir el tráfico de red y ahorrar ancho de banda, incrementa notablemente el rendimiento de la red. En cuanto a la tolerancia a las fallas se alcanza niveles óptimos gracias a los servicios de grupo, mediante los cuales es posible ejecutar varias instancias del mismo servicio en múltiples peers de un mismo grupo.

P2PS ha sido diseñada como una plataforma para la creación de aplicaciones P2P de propósito general que busca simplificar el proceso de desarrollo ocultando la complejidad inherente de plataformas similares como JXTA. Sin embargo, y a pesar de compartir muchas de sus características con JXTA, la ausencia de los mecanismos necesarios para la creación y manipulación de grupos de *peers* da como resultado una disminución considerable de características deseables como la seguridad, escalabilidad, descentralización, rendimiento y la tolerancia a las fallas.

Los resultados de la evaluación previa se plasman en la Tabla 1.

Características	Plataformas P2P				
	Gnutella	**GNUnet**	**Anthill**	**JXTA**	**P2PS**
Descentralización	ALTO	ALTO	ND	ALTO	BAJO
Escalabilidad	ALTO	ALTO	ND	ALTO	BAJO
Anonimato	MEDIO	ALTO	ND	ALTO	ALTO
Rendimiento	MEDIO	MEDIO	ND	ALTO	BAJO
Seguridad	MEDIO	ALTO	ND	ALTO	BAJO
Tolerancia a Fallas	ALTO	ALTO	ND	ALTO	BAJO
Interoperabilidad	ND	ND	BAJO	ALTO	ALTO
Documentación	ALTO	ALTO	BAJO	ALTO	MEDIO

ND = No Definido[8]

Tabla 1. Comparación entre las plataformas identificadas

3.3 Selección de la plataforma

El análisis de la Tabla 1 permite concluir que la plataforma que presenta los mejores resultados en cuanto a las características más relevantes de los sistemas P2P es JXTA, lo cual deja entrever su gran flexibilidad y adaptación, razón por la cual ha sido elegida como la plataforma sobre la cual implementar la extensión propuesta.

3.3.1 JXTA

El proyecto JXTA busca mejorar una serie de deficiencias que se encuentran presentes en

[8] La documentación disponible no brinda la información necesaria para poder dar un resultado fiable.

la gran mayoría de los sistemas P2P actuales [29], a saber:

- Interoperabilidad: JXTA esta diseñado para permitir que los peers se puedan localizar fácilmente unos a otros, se comuniquen, participen en actividades grupales y brinden sus servicios de manera transparente, tanto dentro como fuera de sus grupos o a usuarios de otros sistemas P2P.

- Independencia de la plataforma: JXTA esta diseñado para ser independiente del lenguaje de programación, del sistema operativo y de la plataforma de red.

- Ubicuidad: JXTA esta diseñado para ser implementado en cualquier dispositivo con un corazón digital.

La licencia de JXTA se otorga utilizando un modelo *open source* similar al que emplea *Apache Software Foundation*, lo cual permite a los desarrolladores descargar tanto los binarios como el código fuente sin costo alguno, modificarlos y enriquecerlos, y volver a distribuir el código a otros miembros de la comunidad, siempre y cuando todos los cambios den crédito a quienes hayan colaborado y el nombre JXTA se use sólo con previa autorización.

3.3.1.1 Arquitectura de JXTA

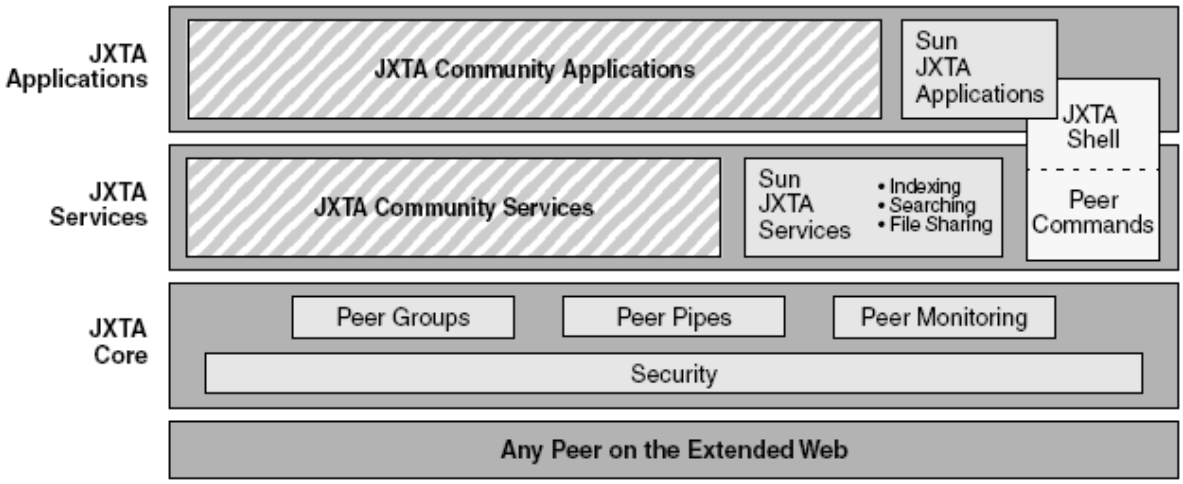


Figura 4. Arquitectura de JXTA [29]

La arquitectura de JXTA está dividida en 3 capas (ver anexo 1):

1. Capa del núcleo: Es la responsable de gestionar los protocolos JXTA (ver anexo 2) mediante los cuales los *peers* pueden comunicarse. Encapsula los componentes básicos de P2P (ver anexo 3).

2. Capa de servicios: Servicios de red que no son absolutamente necesarios pero si comúnmente utilizados o deseables en un entorno P2P.

3. Capa de aplicaciones: En esta capa se encuentran las aplicaciones P2P.

La provisión de servicios P2P es el proceso que engloba todas las tareas que conducen a proporcionar, modificar o eliminar servicios P2P con base en las solicitudes realizadas por los *peers*. En este trabajo de grado se va un paso más allá y se propone la creación de una extensión para la gestión de servicios P2P que además de las características mencionadas anteriormente permita llevar a cabo tareas de administración y control sobre los recursos y servicios de una red P2P.

3.4 Gestión de redes

La creciente complejidad de las redes y su dispersión geográfica plantean serios obstáculos a la gestión eficaz de redes y sistemas. Los administradores de redes de la actualidad deben hacer frente a problemas tales como el acceso a Internet, la seguridad de la red, la computación Cliente/Servidor, la proliferación de aplicaciones distribuidas, las exigencias cambiantes de los usuarios, los crecientes costos de los servicios y las reducciones de personal [35].

Con el objetivo de hacer frente a estos desafíos surgieron los sistemas de gestión de red, a través de los cuales el administrador puede gestionar tanto pequeños grupos de trabajo como grandes empresas desde una única plataforma de gestión [35].

Algunos de los objetivos de los sistemas de gestión de red son asistir a los administradores en la consecución de los siguientes aspectos [36]:

- Uso eficiente de los recursos de red, maximizando el valor de las inversiones realizadas.
- Mantener un alto nivel de disponibilidad de los servicios de red.
- Ofrecer mecanismos para detectar, diagnosticar y recuperarse frente a problemas en la red.
- Asignar recursos de red a los servicios que se implantan sobre ella.
- Asistir en el diseño y planificación de la implantación de nuevos recursos de red.

En cuanto a los recursos a gestionar, no tienen por qué ser sólo recursos de red, sino que también pueden ser servicios y aplicaciones o sistemas finales [36]. Así, según el objeto de la gestión, se puede hablar de gestión de negocios, gestión de servicios y gestión de red (Figura 7).

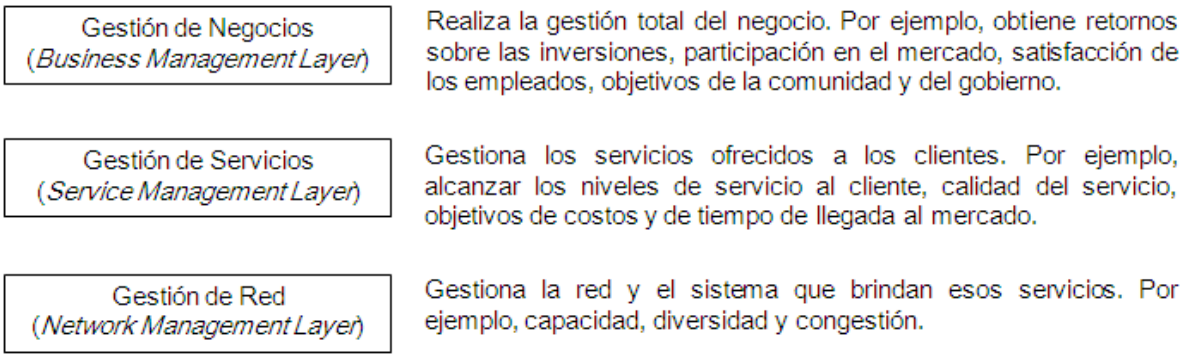


Figura 7. Capas de gestión de servicios

En [37] se introduce el concepto de áreas funcionales de la gestión de redes, es decir, qué es lo que se gestiona de las redes. Estas áreas constituyen el modelo FCAPS, acrónimo de: *Failure*, *Configuration*, *Accounting*, *Performance* y *Security*, es decir, gestión de fallos, de configuración, de contabilidad, de rendimiento y de seguridad [36].

- Gestión de Fallos: Consiste en la monitorización o seguimiento del sistema gestionado con el fin de detectar posibles problemas, y las políticas de actuación en caso de fallo para recuperarse de estos problemas.

- Gestión de Configuración: Incluye todas las tareas que tienen que ver con la recolección de información de la red, modificación del comportamiento de los dispositivos y almacenamiento de la información que determina este comportamiento. Entre estas tareas están el inventario de la red, control de versiones, salvaguarda y restauración de la configuración de los elementos, modificación de ésta ante la detección de errores o por la necesidad de cambios administrativos, etc.

- Gestión de Contabilidad: Tiene como objetivo medir qué se hace de la red y distribuir los costes entre los distintos usuarios de acuerdo con distintas políticas.

- Gestión de Rendimiento: Consiste en realizar un seguimiento de la adecuación de la red a su propósito, manteniendo la calidad de servicio deseada mediante la obtención de estadísticas que incluyen el para qué y el cómo se está utilizando la red.

- Gestión de Seguridad: Tratar de controlar el acceso a los recursos por parte de los distintos usuarios posibles y proteger la información en tránsito.

A lo largo del tiempo han aparecido diferentes modelos de gestión definidos por diferentes organismos de estandarización. A continuación se mencionan los más relevantes:

3.4.1 Modelo de gestión OSI y arquitectura TMN

La ISO (*International Standarization Organization*) definió el modelo OSI-SM (*Open Systems Interconnection – Systems Management*) con el objetivo de integrar la gestión en el modelo de comunicaciones OSI de 7 niveles [36].

El marco de gestión OSI-SM tenía una serie de características muy útiles, de tal forma que la ITU (*International Telecommunication Union*) decidió reutilizarlo para su propuesta de arquitectura de gestión TMN (*Telecommunication Management Network*) [37]. Este último modelo define una arquitectura física (estructura y componentes), un modelo organizativo de gestión, un modelo funcional (servicios y funciones de gestión) y un modelo de información, y ha alcanzado una difusión considerable entre los operadores de telecomunicaciones [36].

OSI-SM y TMN definen los siguientes conceptos de gestión:

- Un mecanismo de comunicaciones basado en CMIS (*Common Management Information Service*) y CMIP (*Common Management Information Protocol*).

- El lenguaje GDMO (*Guidelines for the Definition of Managed Objects*), mediante el cual se especifica la información de gestión.

3.4.2 Modelo de gestión de red SNMP

Este es el modelo de gestión integrada definido para Internet por el IETF (*Internet Engineering Task Force*) y se basa en los siguientes elementos [36]:

- El protocolo SNMP (*Simple Network Management Protocol*), que define las comunicaciones entre gestor y agente.

- El lenguaje de definición de información de gestión SMI (*Structure of Management Information*), que normaliza la sintaxis.

- Un modelo de información basado en MIBs (*Management Information Bases*). El IETF ha definido numerosas MIBs estándar con el objeto de proporcionar una serie de conceptos comunes, y sigue definiendo más, avanzando hacia la normalización semántica.

SNMP está basado en el modelo gestor/agente, compuesto por un gestor, un agente, una base de datos de información de gestión, objetos gestionados y los protocolos de red. El gestor proporciona la interfaz entre el gestor humano y el sistema de gestión. El agente proporciona la interfaz entre el gestor y los dispositivos físicos gestionados [36].

3.4.3 Modelo de gestión WBEM

El modelo de gestión basado en Web o WBEM (*Web Based Enterprise Management*) surge en el seno del DMTF (*Distributed Management Task Force*) con el objetivo de lograr

la interoperabilidad entre los modelos de gestión anteriores. Sin embargo, también se puede usar directamente como un modelo de gestión más [36].

Como su nombre lo indica, la idea clave de este modelo es reutilizar las tecnologías de la Web sobre las que existe gran conocimiento y que ofrecen buenos resultados: HTTP y XML.

Los elementos de este modelo de gestión son:

- Modelo de información: Esquemas CIM (*Common Information Model*).
- Protocolo de comunicaciones: Emplea HTTP y XML de manera conjunta para el intercambio de información.
- Lenguaje de especificación de información de gestión: basado en CIM.

3.4.4 JMX

Las extensiones de gestión de Java (JMX) consisten en un conjunto de especificaciones y herramientas para la gestión de aplicaciones Java. JMX especifica la arquitectura de gestión y los APIs así como un conjunto de servicios de gestión que facilitan la instrumentación de aplicaciones y de recursos de sistema para su supervisión y control [38].

JMX ha sido diseñado para ser compatible con las tecnologías de gestión existentes, incluyendo SNMP y WBEM, lo que le confiere una gran flexibilidad y facilita enormemente su adopción como estándar de gestión.

JMX permite la abstracción de todo de tipo de recursos en elementos gestionables, lo que permite manejar de una manera homogénea la gran diversidad de elementos a gestionar. Así, se utilizan aplicaciones de gestión para monitorizar y controlar elementos gestionables, típicamente de manera remota.

Aunque JMX está pensado en principio para aplicaciones Java, se puede utilizar para otros ámbitos, sin embargo, para ello ha de incluirse necesariamente una capa Java sobre la cual utilizar JMX. En nuestro caso eso no es un problema pues utilizamos la implementación de JXTA para Java por ser ésta la más avanzada.

3.4.4.1 Arquitectura de JMX

JMX presenta una arquitectura en capas (Figura 8), en la que los elementos gestionados y las aplicaciones de gestión pueden conectarse de manera dinámica, en tiempo de ejecución.

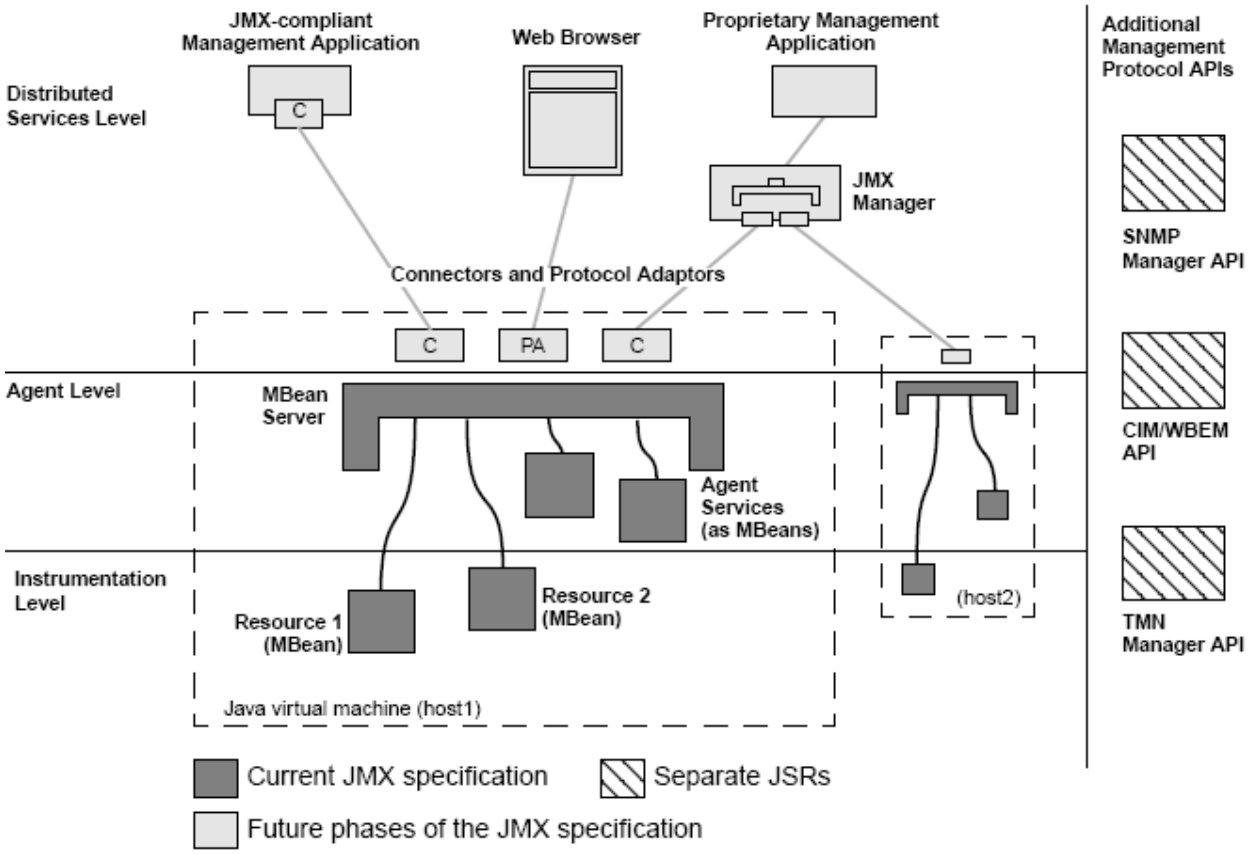


Figura 8. Arquitectura de JMX [38]

3.4.4.1.1 Nivel de instrumentación

Antes de profundizar más en los detalles arquitectónicos de JMX, vamos a introducir el concepto de MBean. Los MBeans son las entidades que representan a los elementos gestionables [38]. Una MBean está compuesta por una interfaz de gestión (que proporciona los atributos y operaciones aplicables a un determinado recurso), junto con las notificaciones que puede emitir dicho recurso y sus meta-datos asociados. Lógicamente, dicha interfaz de gestión ha de ir acompañada de una implementación que la realice, y envíe las notificaciones.

Por tanto, este nivel proporciona las herramientas para describir la información de gestión deseada. Cada objeto gestionado queda representado por su MBean, pudiendo tener diferentes niveles de detalle en función de los objetivos deseados [38].

Para instrumentar una aplicación hay que definir primero los recursos a gestionar y posteriormente la estrategia a seguir para gestionarlos.

Al definir los recursos a gestionar, debemos definir cuáles son los atributos, las operaciones y las notificaciones relativos a cada recurso:

- Los atributos contienen el estado de un recurso.

- Las operaciones ejecutables sobre dicho recurso generalmente provocan un cambio de estado en el recurso, aunque no necesariamente, puesto que una operación puede proporcionar información sobre su estado. Las operaciones pueden tener parámetros, y corresponden a llamadas a métodos Java.

- Las notificaciones son emitidas por el recurso, típicamente de manera asíncrona, para informar de un determinado suceso.

La primera forma de recolección de información de gestión es mediante sondeo del agente. Bien un gestor remoto o bien algún componente del sistema pueden sondear eventualmente o periódicamente por el valor de un cierto atributo. Sin embargo este esquema puede resultar muy costoso en términos de ancho de banda si la frecuencia de sondeo es muy elevada o el volumen de datos muy grande. Así, el operador debe decidir un compromiso entre ancho de banda consumido y tiempo de respuesta.

Sin embargo, en muchas circunstancias, dicho compromiso no es aceptable. Por ejemplo, ante errores graves (pero raros) del sistema. En ese caso es fundamental que el operador conozca la existencia del error en tiempo real, sin embargo, el coste en términos de ancho de banda puede resultar prohibitivo.

La alternativa más razonable a esto son las notificaciones. En vez de sondear al agente para comprobar si el valor de un parámetro se encuentra dentro de un rango, o si ha ocurrido algún evento significativo, se delega en el agente para notificar al gestor (u otros componentes del sistema).

Para implementar la arquitectura de notificaciones, JMX define dos entidades: El emisor de notificaciones y el receptor de notificaciones. La otra entidad que interviene en el modelo es el servidor de MBeans, que pone en contacto ambas entidades y permite que las notificaciones se envíen de forma remota.

3.4.4.1.2 Nivel de agente

El nivel de agente está formado por el servidor de MBeans y por los servicios de agente. El servidor de MBeans actúa de registro de MBeans, para ello, cada MBean registrado en él tiene asociado un nombre de objeto que actúa de clave del registro [38]. Una vez que se posee una referencia a un servidor de MBeans se puede:

- Controlar el ciclo de vida de los MBeans, incluyendo instanciación, registro y des-registro.

- Suscribir un receptor de notificaciones a un determinado emisor, incluyendo el propio servidor de MBeans. Esto último es debido a que el propio servidor está a su vez registrado como MBean y como tal emite notificaciones.

- Realizar una consulta al servidor por algún conjunto de MBeans. Dicha consulta se realiza en base al nombre de objeto así como en base a atributos que posea.

- Sondear los MBeans, consultando el valor de sus atributos, si son de lectura y modificándolos, si son de lectura y escritura.

- Invocar las operaciones de gestión que ofrezca la MBean.

3.4.4.1.3 Nivel de servicios distribuidos

El nivel de servicios distribuidos también es conocido como el nivel de gestor remoto [38]. En este nivel se proporciona acceso remoto al agente JMX de dos maneras diferentes: mediante el uso de conectores y de adaptadores.

- Conectores: Proporcionan una visión completa del agente JMX. Para ello, los conectores están estructurados en un par cliente-servidor. El servidor se ejecuta en la misma máquina virtual que el servidor de MBeans, y es una MBean en sí mismo. El cliente, en cambio, se ejecuta en una máquina virtual distinta. Ambos se comunican abstrayendo al gestor remoto de los detalles. El resultado de esto es que el gestor remoto accede al servidor como si se ejecutase en la misma máquina virtual, abstrayéndole de todos los detalles.

- Adaptadores: Proporcionan una visión parcial del agente JMX. Ello es así porque deben servir de pasarelas entre protocolos distintos y no totalmente compatibles. Así por ejemplo, un adaptador SNMP traducirá notificaciones JMX a traps SNMP, pero un adaptador HTML, por su propia naturaleza, no podrá enviar notificaciones.

3.5 Elección de la tecnología de gestión

El modelo OSI supuso la primera formalización de los conceptos y técnicas de gestión existentes en su día, en un esfuerzo por lograr su estandarización.

TMN se enfrentó a los problemas de gestionar las enormes redes de telecomunicaciones y sus problemas de interconexión, para lo cual generó un modelo complejo basándose en el modelo OSI.

SNMP se desarrolló para gestionar equipos y redes sencillas, con poca potencia por lo que la simpleza era un requisito imprescindible.

WBEM proporciona una solución más actual a la gestión distribuida de equipos, redes y aplicaciones, compatible con las otras tecnologías a través de CIM.

JMX proporciona las herramientas para gestionar aplicaciones y recursos de una manera sencilla y rápida. Además, JMX es compatible con las anteriores tecnologías de gestión al no imponer ninguna suposición respecto al protocolo a utilizar o a las características del cliente.

El análisis de las características planteadas permite concluir que la plataforma de gestión que mejor se adapta a las necesidades de este proyecto es JMX. A continuación se resumen las razones principales de su elección como tecnología de gestión [38]:

- Debido a su arquitectura modular JMX es muy escalable, además está pensado para adaptarse a las nuevas tendencias en gestión, integrándose con servicios de localización y descubrimiento como Jini, UPnP y SLP, por lo que se adapta bien a los entornos orientados a servicios.

- JMX define las interfaces necesarias para la gestión y no pretende ser un sistema distribuido orientado a objetos de propósito general.

- JMX permite la generación de información de gestión bajo demanda, además existen consolas de gestión que pueden ser utilizadas.

- Se utiliza la implementación de JXTA para Java, y JMX proporciona una manera sencilla y rápida de gestionar recursos y aplicaciones Java.

3.6 Conclusiones

Después de analizar los resultados obtenidos del estudio comparativo entre las plataformas identificadas (sección 3.2) se pudo concluir que la plataforma más apropiada para los propósitos de este trabajo de grado es JXTA, razón por la cual fue seleccionada como la plataforma sobre la cual llevar a cabo la extensión propuesta.

De igual manera, luego de examinar las características de cada uno de los modelos de gestión identificados (sección 3.4) se concluyó que el más apropiado de acuerdo a los requerimientos es JMX, motivo por el cual fue seleccionado como la tecnología de gestión sobre la cual fundamentar el diseño de la extensión.

CAPITULO 4

DESCRIPCION CONCEPTUAL Y TECNICA DEL PROTOTIPO DE SOFTWARE

4.1 Análisis de requisitos

A continuación se formaliza la captura y análisis de requisitos para la construcción de la extensión para la gestión de servicios. Para simplificar su comprensión e ilustrarlos se parte de una visión general de un sistema de gestión que implemente dicha extensión. Esta visión no es exhaustiva y sólo pretende servir de marco de referencia para la realización del análisis.

4.1.1 Visión del sistema

El escenario básico (Figura 1) está compuesto de múltiples peers interconectados a través de una red JXTA, para cada uno de los cuales existe un agente asociado, el cual es el encargado de gestionar los recursos y/o servicios que provee el peer. A su vez, cada agente está asociado a un sistema de gestión remoto, el cual se utiliza para monitorizar la red mediante el sondeo de los atributos de gestión definidos para cada recurso y/o servicio.

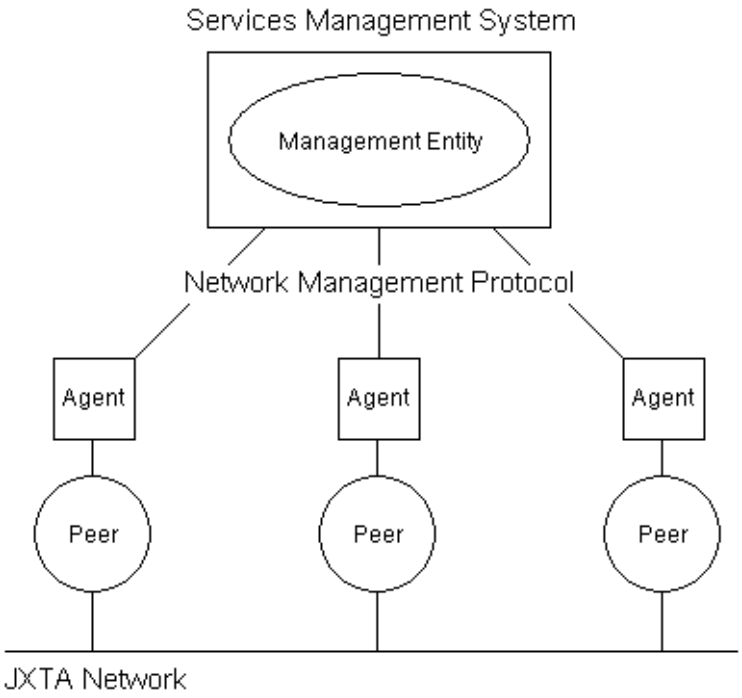


Figura 9. Escenario general de gestión

4.1.2 Requisitos del sistema

Con el objetivo de clasificar los requisitos para la construcción de la extensión se sigue la distinción entre requisitos funcionales y no funcionales propuesta en [39]. Además, los requisitos se distinguen unos de otros por su prioridad, esto es:

- Obligatorios: Son aquellos requisitos que representan las características imprescindibles de la extensión y cuyo no cumplimiento desvirtuaría su naturaleza. Así por ejemplo, es obligatorio que la extensión brinde los mecanismos necesarios para la implementación de la gestión proactiva.

- Opcionales: Como su nombre lo indica, estos requisitos no son obligatorios. De esta manera, los mecanismos que permitan la implementación de políticas de gestión son deseables pero su ausencia no desvirtúa la naturaleza de la extensión.

4.1.2.1 Requisitos funcionales

4.1.2.1.1 Gestión proactiva

Con el objetivo de reducir la carga de procesamiento que supone sondear periódicamente el valor de los atributos de gestión a lo largo y ancho de la red se requiere de la gestión proactiva [40]. La extensión debe proporcionar los mecanismos básicos necesarios para el desarrollo de servicios proactivos, esto es, servicios que puedan informar, a un sistema de gestión, acerca de la variación de alguno de sus atributos o métodos de gestión mediante la generación de notificaciones.

Prioridad: Obligatorio. La emisión de notificaciones reduce la carga de procesamiento del sistema de gestión e incrementa su rendimiento.

4.1.2.1.2 Ejecución de procesos remotos

El administrador de red estará en la capacidad de ejecutar operaciones remotas sobre los servicios, esto facilita la configuración y el mantenimiento de la red. La extensión debe suministrar los mecanismos básicos necesarios para el desarrollo de sistemas de gestión que permitan la ejecución de procesos remotos.

Prioridad: Opcional. La gestión incluye intervención. Es una característica avanzada del sistema, no básica.

4.1.2.1.3 Despliegue de políticas de gestión

Las políticas de gestión permitirán que el administrador de red especifique el comportamiento de los servicios cuando éstos cumplan con ciertas condiciones. La extensión debe suministrar los mecanismos necesarios para el desarrollo de sistemas de

gestión que permitan la implantación de políticas de gestión, por ejemplo, se puede restringir el número de conexiones de chat que un *peer* puede establecer, funcionalidad que enriquece la gestión proactiva.

Prioridad: Opcional. La gestión incluye control. Es una característica avanzada del sistema, no básica.

4.1.2.2 Requisitos no funcionales

4.1.2.2.1 Flexibilidad

El trabajo aquí realizado puede ser refinado y complementado con otros aportes, por ejemplo, la adición de componentes de gestión mucho más complejos, razón por la cual se tiene en cuenta la flexibilidad de la extensión para incorporar nuevos elementos que permitan optimizar la gestión de los recursos y servicios de red. La flexibilidad se garantiza gracias a la arquitectura modular propuesta (ver sección 4.2.2).

Prioridad: Obligatorio. Este requisito forma parte de las buenas prácticas de ingeniería del software.

4.1.2.2.2 Distribución libre

Los resultados de éste trabajo de grado se distribuirán bajo los términos de la Licencia Pública General (*GPL* por sus siglas en inglés) por las ventajas que ésta proporciona, entre las cuales se pueden mencionar:

- El código fuente licenciado bajo GPL debe estar disponible y accesible para copias ilimitadas y a cualquier usuario que lo solicite.

- De cara al usuario final, el software licenciado bajo GPL es totalmente gratuito, pudiendo pagar únicamente por gastos de copiado y distribución.

- GPL contribuye al mejoramiento y evolución del software, ya que la disponibilidad y acceso global de los programas permite la expansión del conocimiento depositado en cada pieza de software.

Prioridad: Obligatorio. Por tratarse de un proyecto de investigación enfocado a la comunidad del software libre.

4.2 Diseño arquitectónico

4.2.1 Componentes de la extensión

La arquitectura general de JXTA ha sido enriquecida con la adición de nuevos componentes en las dos capas superiores (Figura 2).

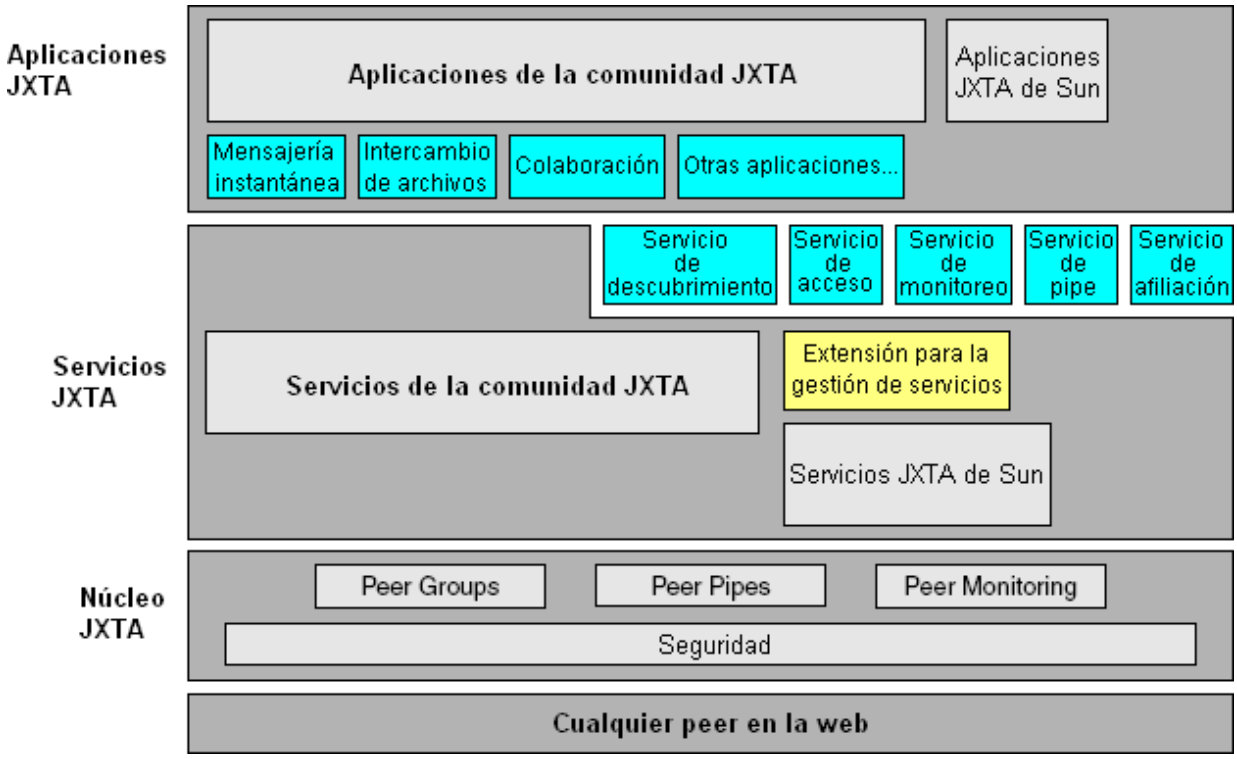


Figura 10. Arquitectura enriquecida de JXTA

En la capa de aplicación se han agregado los bloques correspondientes a las aplicaciones P2P más comunes (ver capítulo 2, sección 2.5), las cuales se pueden enmarcar en: Mensajería instantánea, intercambio de archivos y colaboración.

En la capa de servicios se han añadido los bloques correspondientes a los servicios estándar de: Descubrimiento, acceso, monitoreo, pipe y afiliación, los cuales han sido identificados en el trabajo de grado denominado: “Definición de perfiles UML para servicios telemáticos orientados a una arquitectura P2P” [41], el cual se desarrolló en el marco de un proyecto general denominado: “Contribución a un sistema de negociación de recursos y servicios en el ámbito de una red superpuesta P2P” [7] del cual también hace parte este trabajo de grado. Finalmente se ha incorporado el bloque correspondiente a la Extensión para la gestión de servicios, cuyos componentes han sido diseñados con base en JMX (ver capítulo 3, sección 3.4.4), por ser ésta la plataforma de gestión seleccionada (ver capítulo 3, sección 3.5).

4.2.2 Arquitectura de la extensión

A continuación se presenta la arquitectura general de la extensión para la gestión de servicios P2P (Figura 3).

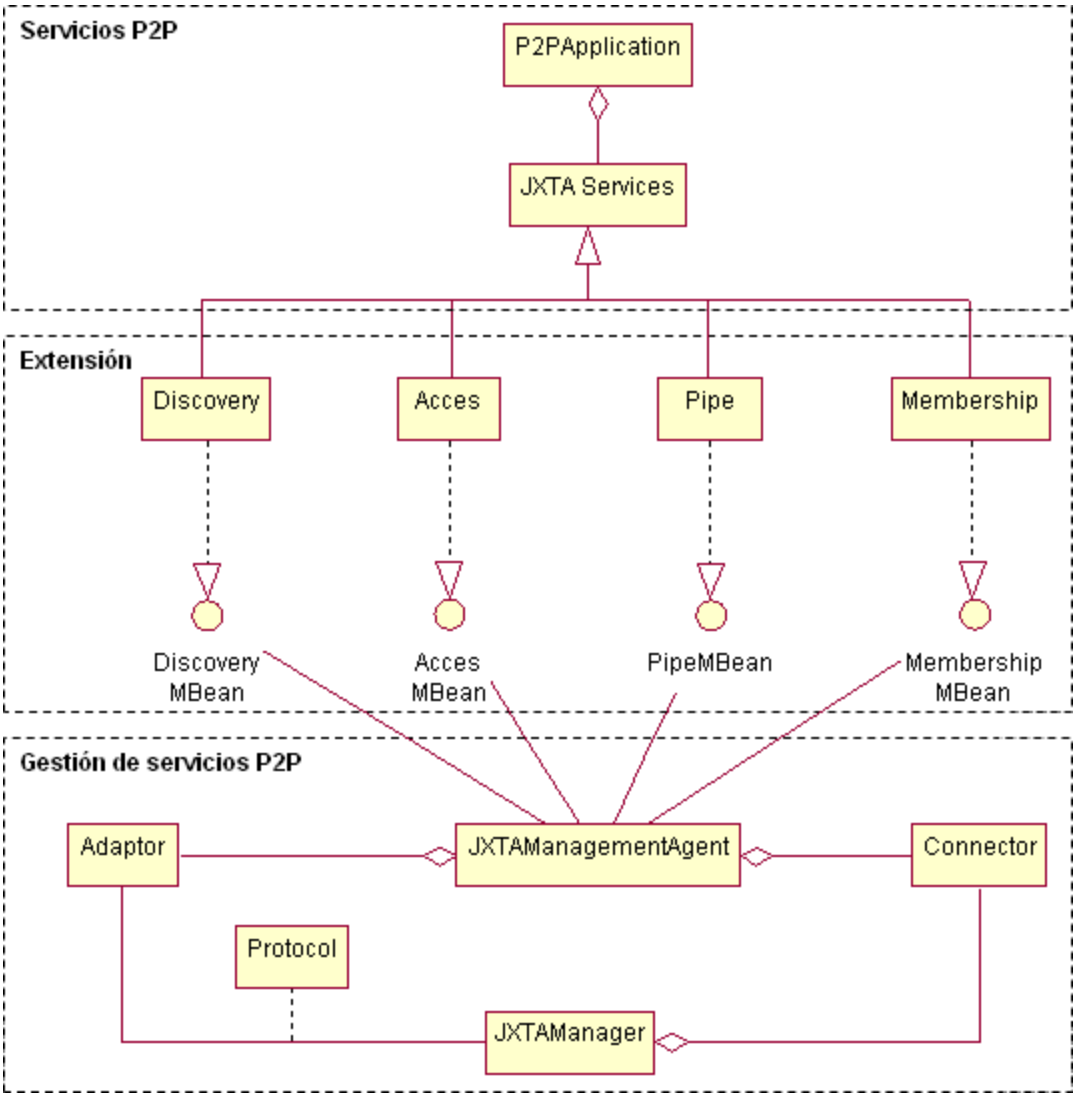


Figura 11. Arquitectura general de la extensión para la gestión de servicios P2P

El bloque superior, denominado **Servicios P2P**, hace referencia a los servicios estándar identificados en [41], los cuales son propios de JXTA.

En el bloque del medio, denominado **Extensión**, se plasman las cuatro interfaces básicas de gestión: *DiscoveryMBean*, *AccesMBean*, *PipeMBean* y *MembershipMBean* junto con sus respectivas implementaciones. Estas interfaces han sido diseñadas con base en la información de gestión más relevante de los servicios estándar propios de JXTA y son el pilar fundamental sobre el cual construir MBeans más avanzados. Así por ejemplo, un

desarrollador puede crear un solo MBean que describa los atributos, operaciones y notificaciones de la aplicación o puede optar por seguir un nivel de granularidad más fina, proporcionando MBeans para todos los objetos relevantes de la aplicación, llegando a tener cientos o incluso miles de ellos distribuidos a lo largo de las cinco áreas funcionales del modelo FCAPS (ver capítulo 3, sección 3.4) en función de los objetivos de gestión deseados.

El bloque inferior, denominado **Gestión de servicios P2P**, contiene los componentes básicos de JMX por ser ésta la tecnología de gestión seleccionada (ver capítulo 3, sección 3.4.4).

4.2.4 Descripción estructural de la extensión

A continuación se presentan los diagramas de clases que describen con mayor detalle la arquitectura general de la extensión para la gestión de servicios (Figura 3).

4.2.4.1 Diagrama general de clases

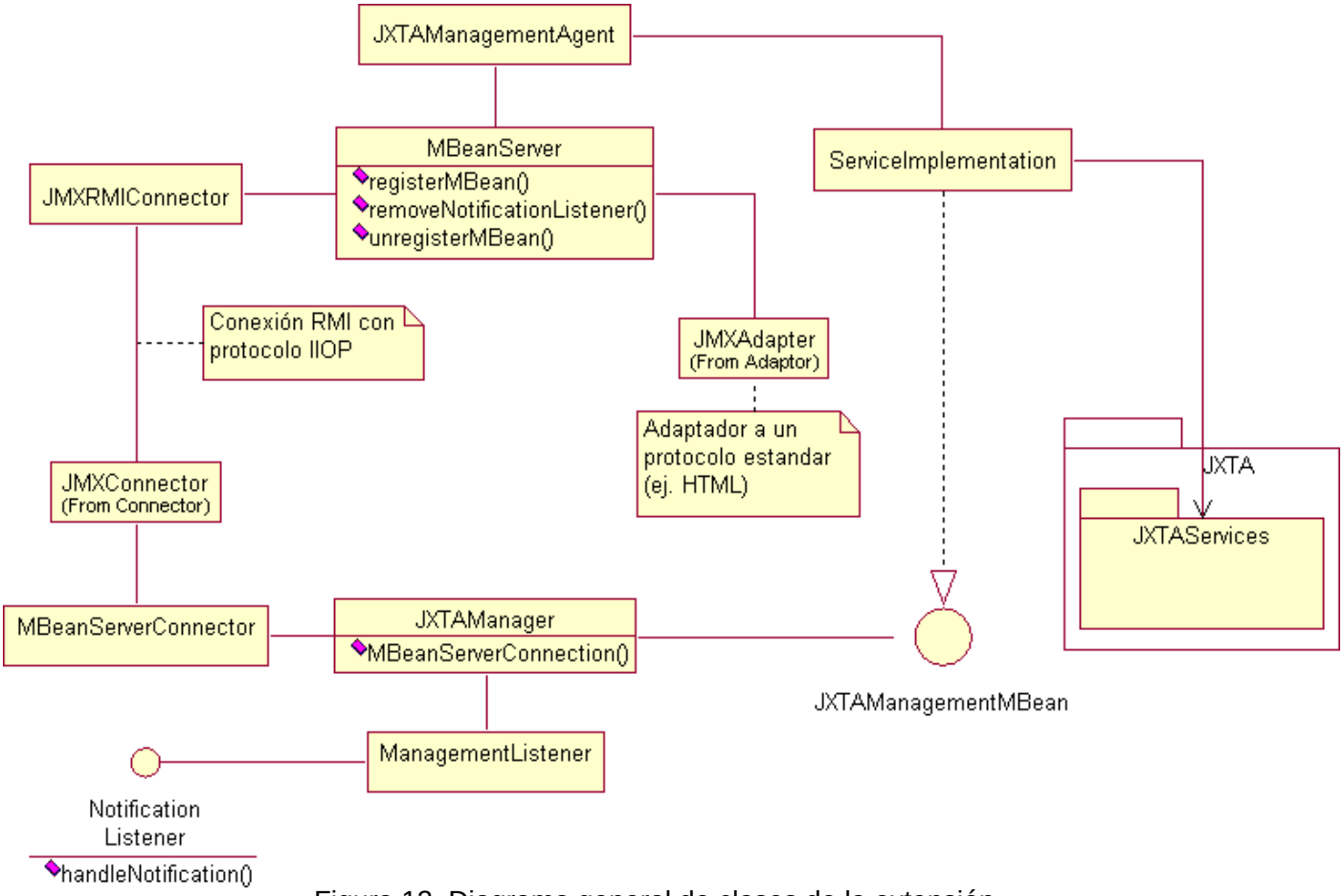


Figura 12. Diagrama general de clases de la extensión

El agente de gestión (*JXTAManagementAgent*) es el encargado de gestionar los servicios P2P (*ServiceImplementation*) los cuales son una abstracción de los servicios P2P que proporciona JXTA. Los servicios P2P implementan las interfaces de gestión de alto nivel (*JXTAManagementMBean*) mediante las cuales emiten notificaciones que son capturadas por un escuchador de notificaciones (*ManagementListener*) el cual las transfiere al sistema de gestión (*JXTAManager*).

El servidor de MBeans (*JXTAManagementAgent*) desempeña dos funciones. En primer lugar actúa de registro de MBeans y en segundo lugar es el intermediario de comunicaciones entre los MBeans y el sistema de gestión (*JXTAManager*) con el cual se comunica a través de un conector (*JMXRMIConnector*) o un adaptador (*JMXAdaptor*) (ver capítulo 3, sección 3.4.4.1.3).

4.2.4.1 Diagrama de clases del servicio de descubrimiento

El servicio de descubrimiento de JXTA denominado *DiscoveryService* proporciona un mecanismo asíncrono para el descubrimiento de *Advertisements* (peers, grupos de peers, *pipes*, módulos, etc.).

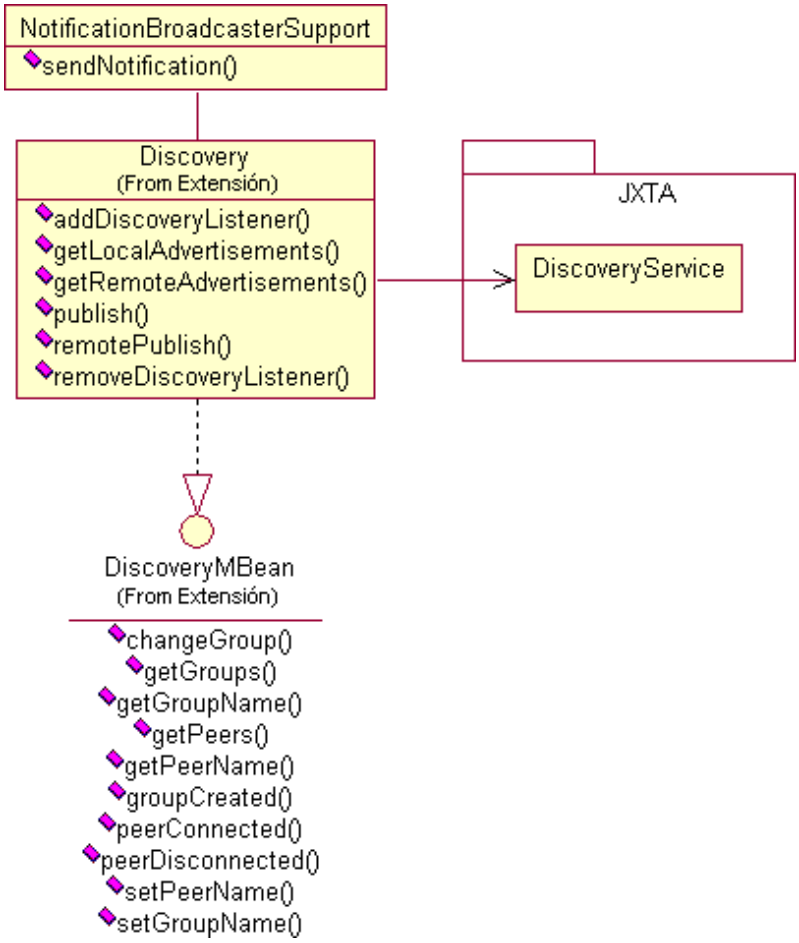


Figura 13. Diagrama de clases del servicio de descubrimiento

La interface de gestión de la Extensión denominada *DiscoveryMBean* expone los métodos necesarios para que el sistema de gestión (*JXTAManager*) administre el descubrimiento de peers y grupos de peers así como sus características asociadas.

4.2.4.2 Diagrama de clases del servicio de acceso

El servicio de acceso de JXTA denominado *AccessService* es utilizado para determinar si un conjunto específico de operaciones están permitidas para un peer en particular.

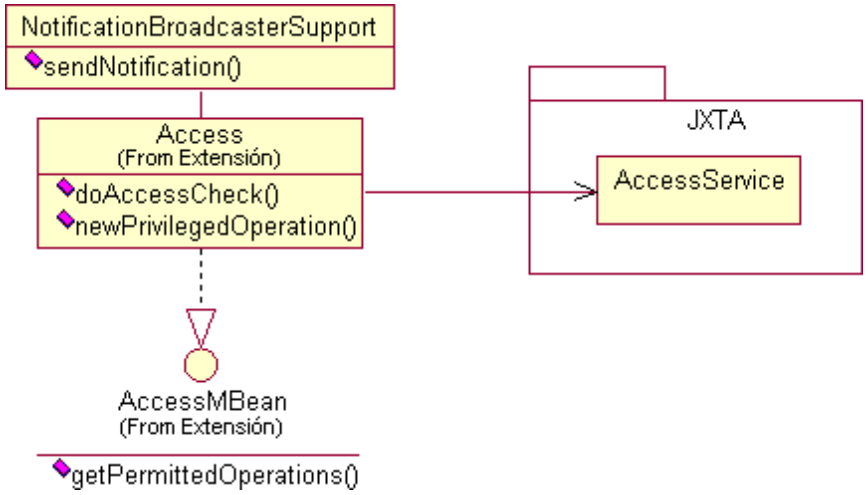


Figura 14. Diagrama de clases del servicio de acceso

La interface de gestión de la Extensión denominada *AccessMBean* le permite al sistema de gestión (*JXTAManager*) saber cuáles son las operaciones a las que un determinado peer tiene acceso.

4.2.4.3 Diagrama de clases del servicio de pipe

El servicio de pipe de JXTA denominado *PipeService* proporciona los mecanismos básicos necesarios para el intercambio de mensajes entre los servicios.

La interface de gestión de la Extensión denominada *PipeMBean* expone los métodos necesarios para que el sistema de gestión (*JXTAManager*) administre el envío y la recepción de mensaje. Mediante esta interface el sistema de gestión puede conocer al remitente y al receptor de un mensaje así como el contenido de este.

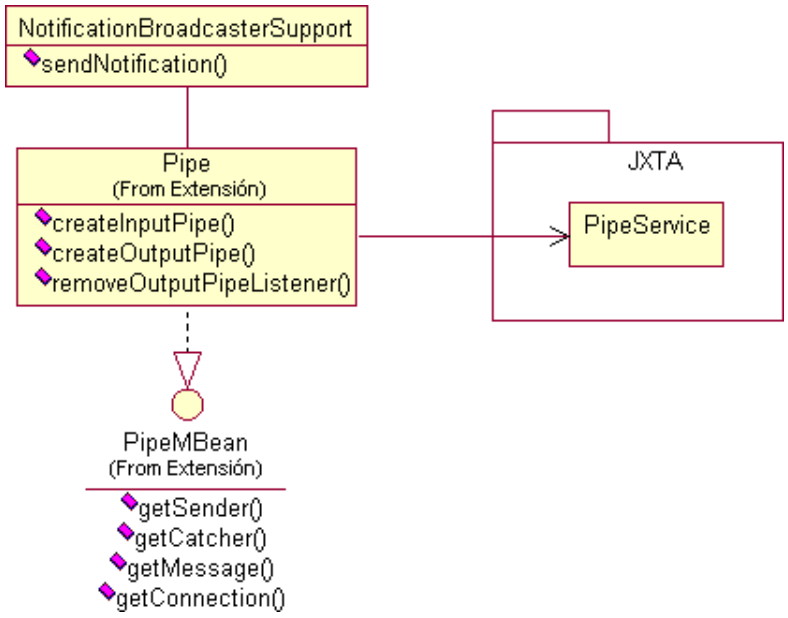


Figura 15. Diagrama de clases del servicio de *pipe*

4.2.4.4 Diagrama de clases del servicio de afiliación

El servicio de afiliación (*MembershipService*) de JXTA le permite a un peer establecer una identificación dentro de un grupo de peers. Las identificaciones son utilizadas por los servicios con el fin de determinar las capacidades de los peers.

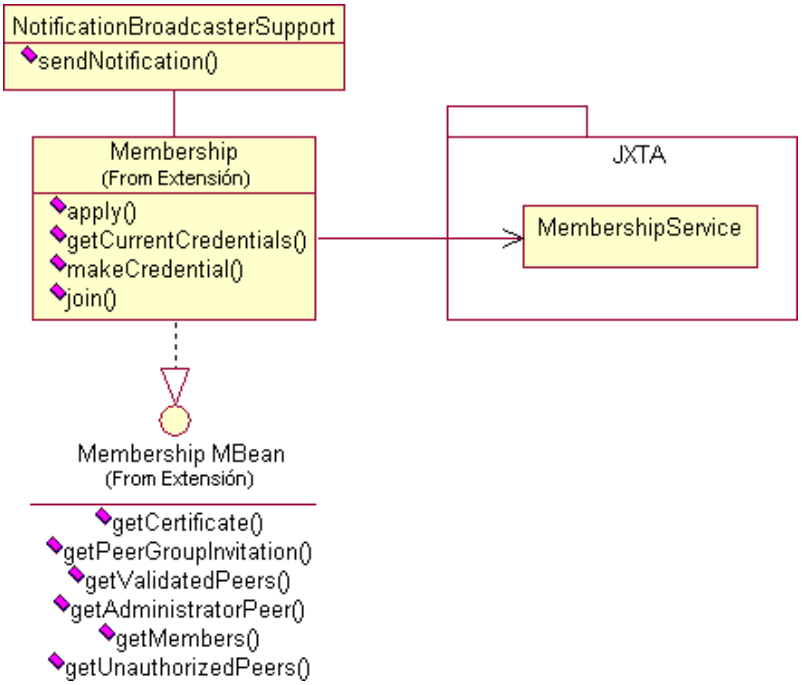


Figura 16. Diagrama de clases del servicio de afiliación

La interface de gestión de la Extensión denominada *MembershipMBean* le permite al sistema de gestión (*JXTAManager*) administrar los mecanismos de seguridad y acceso asociados a los grupos de peers así como sus características asociadas.

4.3 Validación de la extensión

La validación de la extensión se lleva a cabo mediante la implementación de una aplicación P2P de mediana complejidad, desarrollada en el marco del proyecto general denominado "Contribución a un sistema de negociación de recursos y servicios en el ámbito de una red superpuesta P2P" [7]. La aplicación P2P agrupa 3 servicios específicos de JXTA (*Pipe*, *Discovery* y *Publish*) los cuales conforman la funcionalidad de una agenda colaborativa P2P (ver anexo 4).

Con el fin de asociar funcionalidad de gestión a la aplicación P2P, se incorporan interfaces de gestión las cuales emiten notificaciones y eventos a un sistema de gestión P2P desarrollado en este proyecto, con el fin de tener control y dar visibilidad y veracidad a los resultados obtenidos sobre la gestión de servicios (sección 4.4).

A partir del escenario descrito en la sección 4.1.1 y de los requisitos definidos en la sección 4.1.2 se identifica a los actores involucrados y se definen los casos de uso del sistema de gestión P2P.

4.3.1 Actores

Los actores fundamentales relacionados con el sistema de gestión son:

- Gestor: Se encarga de gestionar los servicios P2P mediante la monitorización de sus atributos y operaciones de gestión.

- Servicio: Hace referencia al servicio P2P (agenda colaborativa P2P).

4.3.2 Diagrama de casos de uso

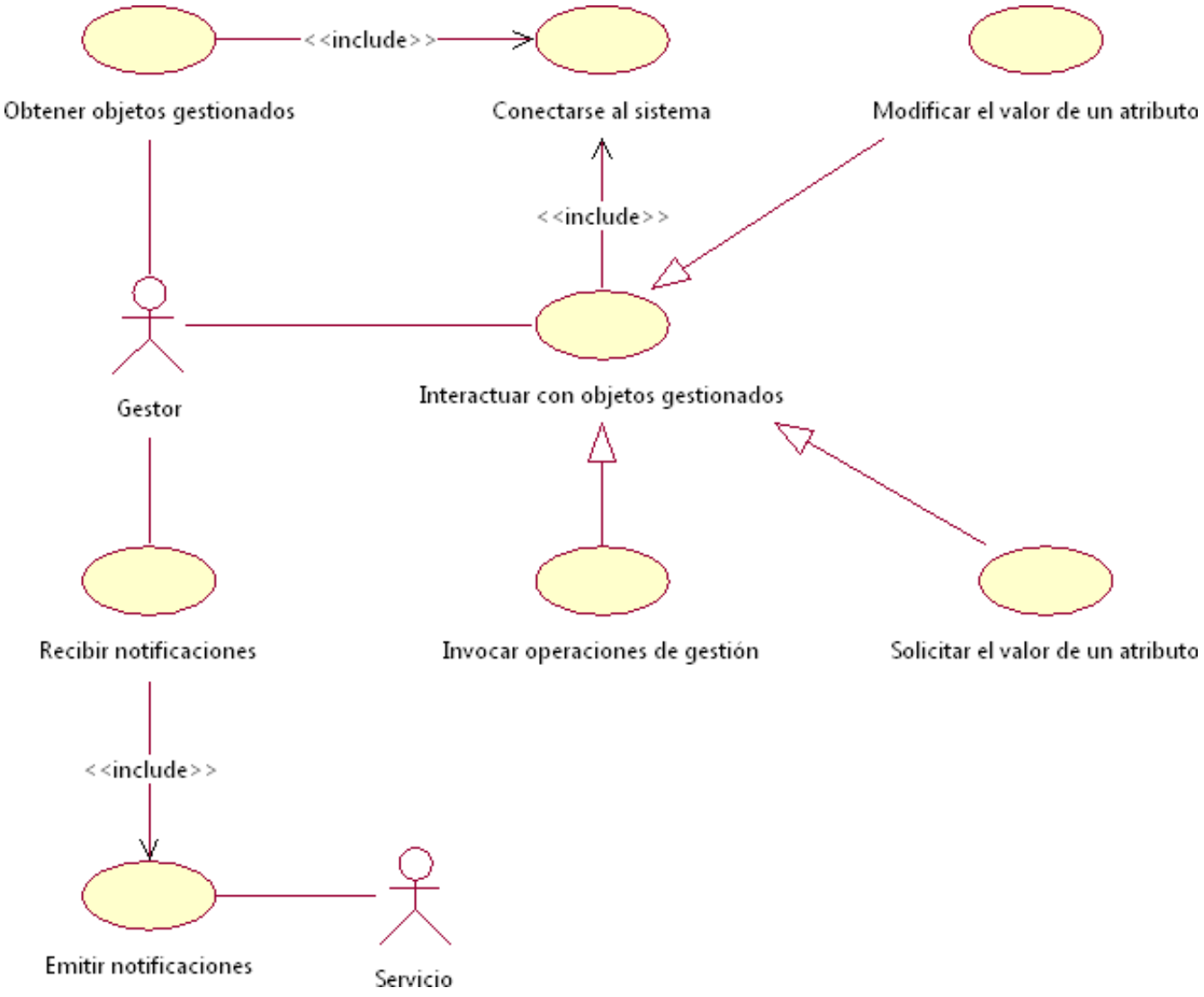


Figura 17. Diagrama de casos de uso del sistema de gestión

4.3.3 Especificación formal de los casos de uso

A continuación se lleva a cabo una descripción detallada de los casos de uso del sistema de gestión P2P.

4.3.3.1 Caso de uso: Conectarse al sistema

Código	**Nombre**		
CU-1	**Conectarse al sistema.**		
Resumen	El gestor se conecta al sistema con un nombre de usuario y una contraseña.		
Prioridad: Alta.		**Complejidad:** Baja.	
Actor(es) Involucrado(s)			
1	Gestor.		
Caso(s) de Uso Relacionado(s)			
1	Obtener objetos gestionados.		
Precondiciones			
1	El sistema debe estar listo para recibir conexiones.		
Postcondiciones			
1			
Flujo Básico			
Acción del Actor	**Acción del Sistema**	**Datos de Entrada**	**Datos de Salida**
1 Ingresa al caso de uso.	Solicita nombre de usuario y contraseña.		
2 Ingresa nombre de usuario y contraseña.	Valida los datos. EX-1.1 = Datos incorrectos.	• Nombre de usuario. • Contraseña.	
3	Muestra el menú principal.		
4 Fin.			
Excepciones			
Código:	**Nombre:**		
EX-1.1	**Datos incorrectos**		
Acción del Actor	**Acción del Sistema**	**Datos de Entrada**	**Datos de Salida**
1	Rechaza la conexión.		Mensaje: “El usuario no está registrado”.
2	Solicita datos nuevamente.		
3 Fin.			

Prototipo de pantalla


Tabla 2. Caso de uso: Conectarse al sistema

4.3.3.2 Caso de uso: Interactuar con objetos gestionados

Código	**Nombre**		
CU-2	**Interactuar con objetos gestionados.**		
Resumen	Con el objetivo de poder realizar acciones de supervisión y control de los recursos de red el gestor debe estar en la capacidad de interactuar con los objetos gestionados.		
Prioridad: Alta.		**Complejidad:** Alta.	
Actor(es) Involucrado(s)			
1	Gestor.		
Caso(s) de Uso Relacionado(s)			
1	Invocar operaciones de gestión.		
2	Solicitar el valor de un atributo.		
3	Modificar el valor de un atributo.		
4	Emitir notificaciones.		
Precondiciones			
1	El gestor debe estar conectado al sistema.		
Postcondiciones			
1			
Flujo Básico			
Acción del Actor	**Acción del Sistema**	**Datos de Entrada**	**Datos de Salida**
1 Selecciona acción sobre el objeto.	Realiza la acción sobre el objeto.		
2 Fin.			
Excepciones			
Código:	**Nombre:**		
Acción del Actor	**Acción del Sistema**	**Datos de Entrada**	**Datos de Salida**
1			

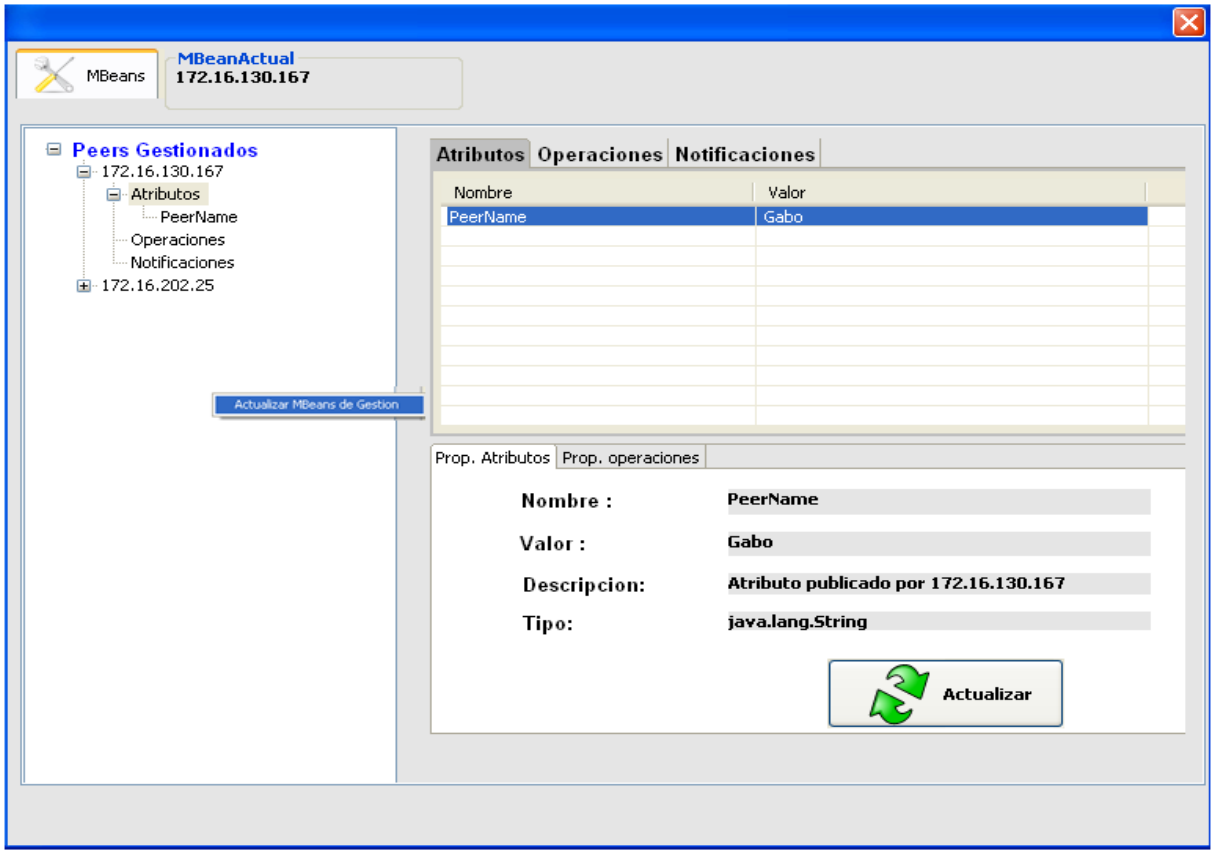
Prototipo de pantalla


Tabla 3. Caso de uso: Interactuar con objetos gestionados

4.3.3.3 Caso de uso: Invocar operaciones de gestión

Código	**Nombre**		
CU-3	**Invocar operaciones de gestión.**		
Resumen	El gestor invoca operaciones de gestión (por ejemplo, cambiar el estado de un peer – conectado/desconectado) que modifican el estado de los objetos.		
Prioridad: Alta.		**Complejidad:** Alta.	
Actor(es) Involucrado(s)			
1	Gestor.		
Caso(s) de Uso Relacionado(s)			
1	Interactuar con objetos gestionados.		
Precondiciones			
1	El gestor debe estar conectado al sistema y tanto el objeto como la operación deben existir.		
Postcondiciones			
1	El estado del sistema se ha modificado.		
Flujo Básico			
Acción del Actor	**Acción del Sistema**	**Datos de Entrada**	**Datos de Salida**
1 Define los parámetros de la operación.			
2 Invoca la operación.	Recibe la orden.	• Parámetros.	
3	Ejecuta la operación en el objeto. EX-3.1 = Parámetros incorrectos.		
4	Envía la respuesta.		
5 Fin.			
Excepciones			
Código:	**Nombre:**		
EX-3.1	**Parámetros incorrectos.**		
Acción del Actor	**Acción del Sistema**	**Datos de Entrada**	**Datos de Salida**
1	No ejecuta la operación.		Mensaje: “Los parámetros son incorrectos.”

Tabla 4. Caso de uso: Invocar operaciones de gestión

4.3.3.4 Caso de uso: Solicitar el valor de un atributo

Código	**Nombre**		
CU-4	**Solicitar el valor de un atributo.**		
Resumen	El gestor consulta el valor de los atributos de gestión definidos para cada objeto.		
Prioridad: Alta.		**Complejidad:** Baja.	
Actor(es) Involucrado(s)			
1	Gestor.		
Caso(s) de Uso Relacionado(s)			
1	Interactuar con objetos gestionados.		
Precondiciones			
1	El gestor debe estar conectado al sistema y tanto el objeto como el atributo deben existir.		
Postcondiciones			
1	El estado del sistema es el mismo que antes de solicitar el valor del atributo.		
Flujo Básico			
Acción del Actor	**Acción del Sistema**	**Datos de Entrada**	**Datos de Salida**
1 Selecciona un objeto.			
2 Selecciona un atributo.			
3 Envía la petición.	Obtiene el valor del atributo.	• Nombre del atributo.	
4	Retorna el valor del atributo.		
5 Fin.			
Excepciones			
Código:	**Nombre:**		
Acción del Actor	**Acción del Sistema**	**Datos de Entrada**	**Datos de Salida**

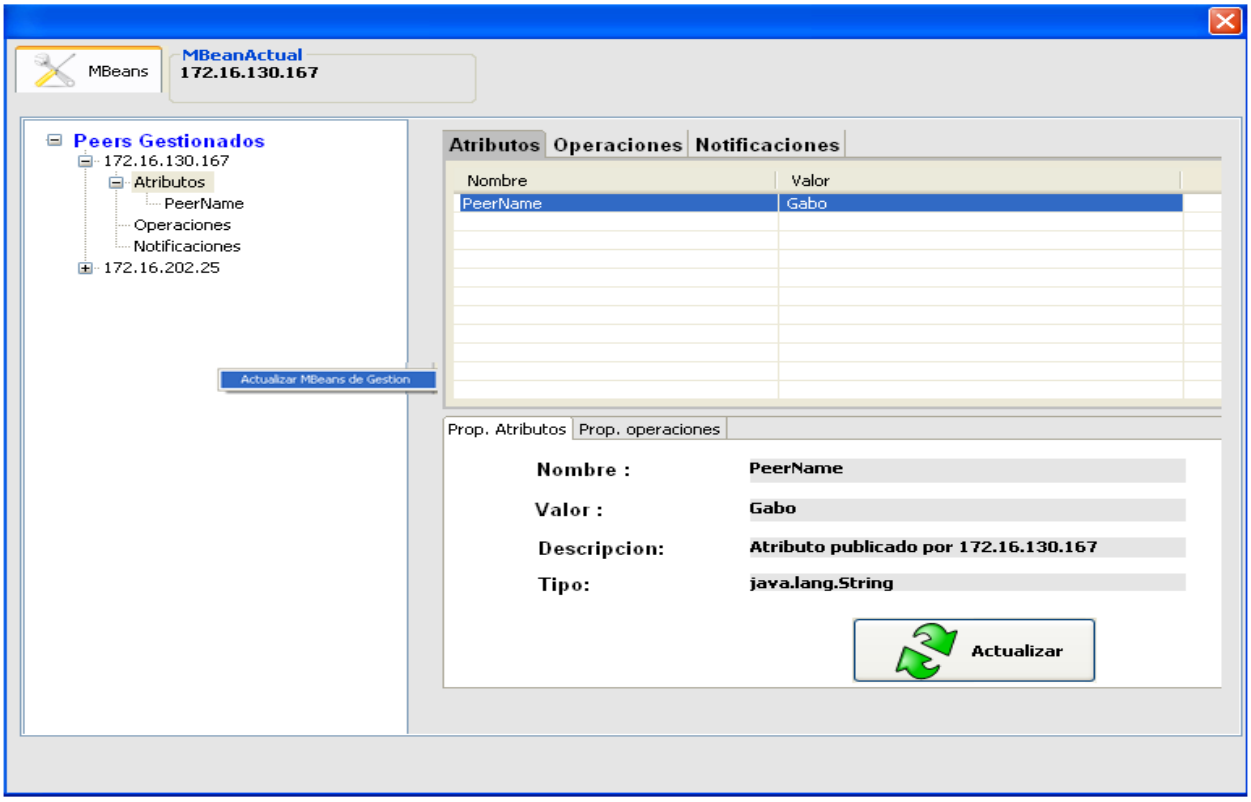
Prototipo de pantalla


Tabla 5. Caso de uso: Solicitar el valor de un atributo

4.3.3.5 Caso de uso: Modificar el valor de un atributo

Código	**Nombre**
CU-5	**Modificar el valor de un atributo.**
Resumen	Los atributos de gestión pueden ser de lectura y/o escritura en cuyo caso el gestor puede modificar su valor.
Prioridad: Alta.	**Complejidad:** Baja.

Actor(es) Involucrado(s)	
1	Gestor.
Caso(s) de Uso Relacionado(s)	
1	Interactuar con objetos gestionados.
Precondiciones	
1	El gestor debe estar conectado al sistema y tanto el objeto como el atributo deben existir.
Postcondiciones	
1	

Flujo Básico				
Acción del Actor		**Acción del Sistema**	**Datos de Entrada**	**Datos de Salida**
1	Selecciona un objeto.			
2	Selecciona un atributo.			
3	Define el nuevo valor para el atributo.			
4	Envía la petición.	Modifica el valor del atributo. EX5.1 = Valor incorrecto.	• Valor del atributo.	
5	Fin.			

Excepciones				
Código:		**Nombre:**		
EX5.1		**Valor incorrecto**		
Acción del Actor		**Acción del Sistema**	**Datos de Entrada**	**Datos de Salida**
1		No cambia el valor del atributo.		Mensaje = “Valor de atributo incorrecto.”

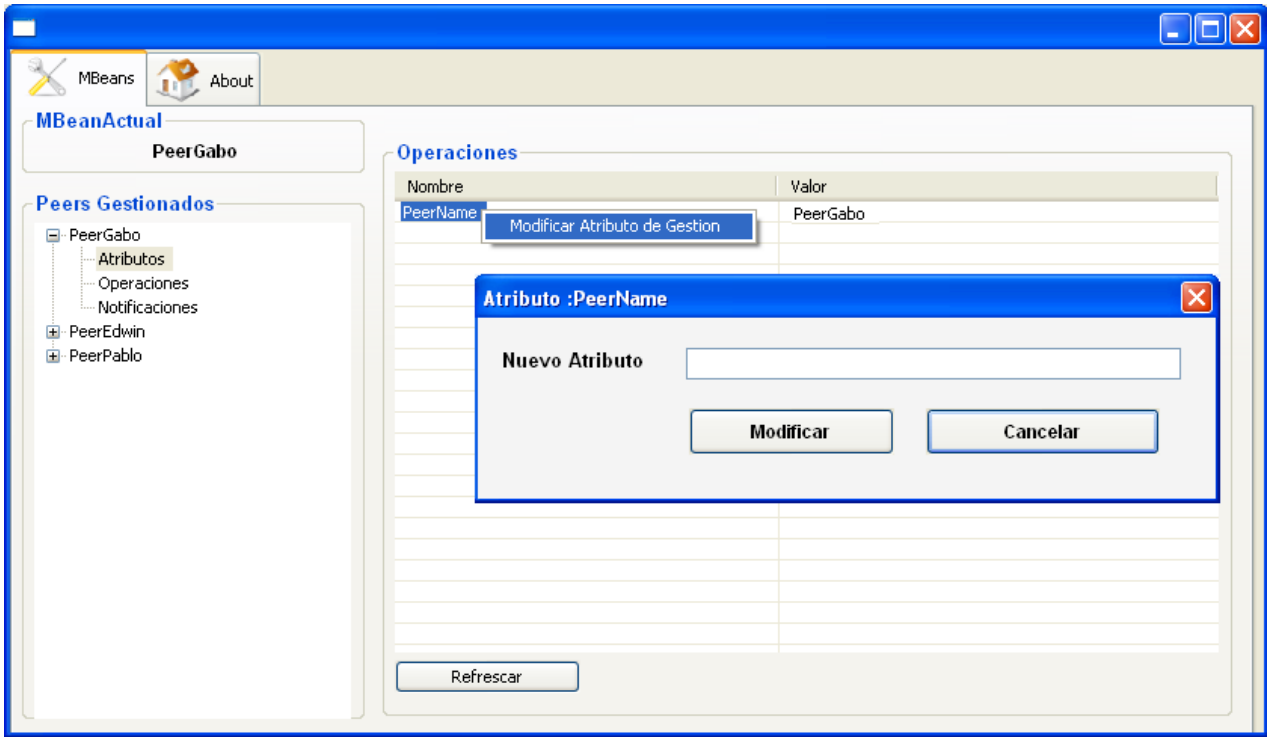
Prototipo de pantalla


Tabla 6. Caso de uso: Modificar el valor de un atributo

4.3.3.6 Caso de uso: Emitir notificaciones

Código	**Nombre**		
CU-6	**Emitir notificaciones.**		
Resumen	Para que los servicios puedan definir su modelo de gestión de manera adecuada describen y emiten notificaciones de gestión.		
Prioridad: Alta.		**Complejidad:** Media.	
Actor(es) Involucrado(s)			
1	Servicio.		
Caso(s) de Uso Relacionado(s)			
1			
Precondiciones			
1	El servicio debe estar en ejecución.		
Postcondiciones			
1			
Flujo Básico			
Acción del Actor	**Acción del Sistema**	**Datos de Entrada**	**Datos de Salida**
1 Define las notificaciones a emitir.			
2 Registra las notificaciones.	Procesa las notificaciones.	• Notificaciones.	
3	Añade las notificaciones a su base de información de gestión.		
4 Emite una notificación.	Envía la notificación.		
5 Fin.			
Excepciones			
Código:	**Nombre:**		
Acción del Actor	**Acción del Sistema**	**Datos de Entrada**	**Datos de Salida**
1			

Tabla 7. Caso de uso: Emitir notificaciones

4.3.3.7 Caso de uso: Recibir notificaciones

Código	**Nombre**		
CU-7	**Recibir notificaciones.**		
Resumen	El servicio emite una notificación cada vez que el suceso desencadenante de la misma ocurre. El gestor recibe las notificaciones.		
Prioridad: Alta.		**Complejidad:** Media.	
Actor(es) Involucrado(s)			
1	Gestor.		
2	Servicio.		
Caso(s) de Uso Relacionado(s)			
1	Interactuar con objetos gestionados.		
Precondiciones			
1	El servicio debe estar en ejecución y el suceso que desencadena la notificación debe haber ocurrido.		
Postcondiciones			
1			
Flujo Básico			
Acción del Actor	**Acción del Sistema**	**Datos de Entrada**	**Datos de Salida**
1 El servicio emite una notificación.	Envía la notificación al gestor de manera asíncrona.		
2 El gestor recibe la notificación.			
3 Fin.			
Excepciones			
Código:	**Nombre:**		
Acción del Actor	**Acción del Sistema**	**Datos de Entrada**	**Datos de Salida**
1 El servicio emite una notificación.			

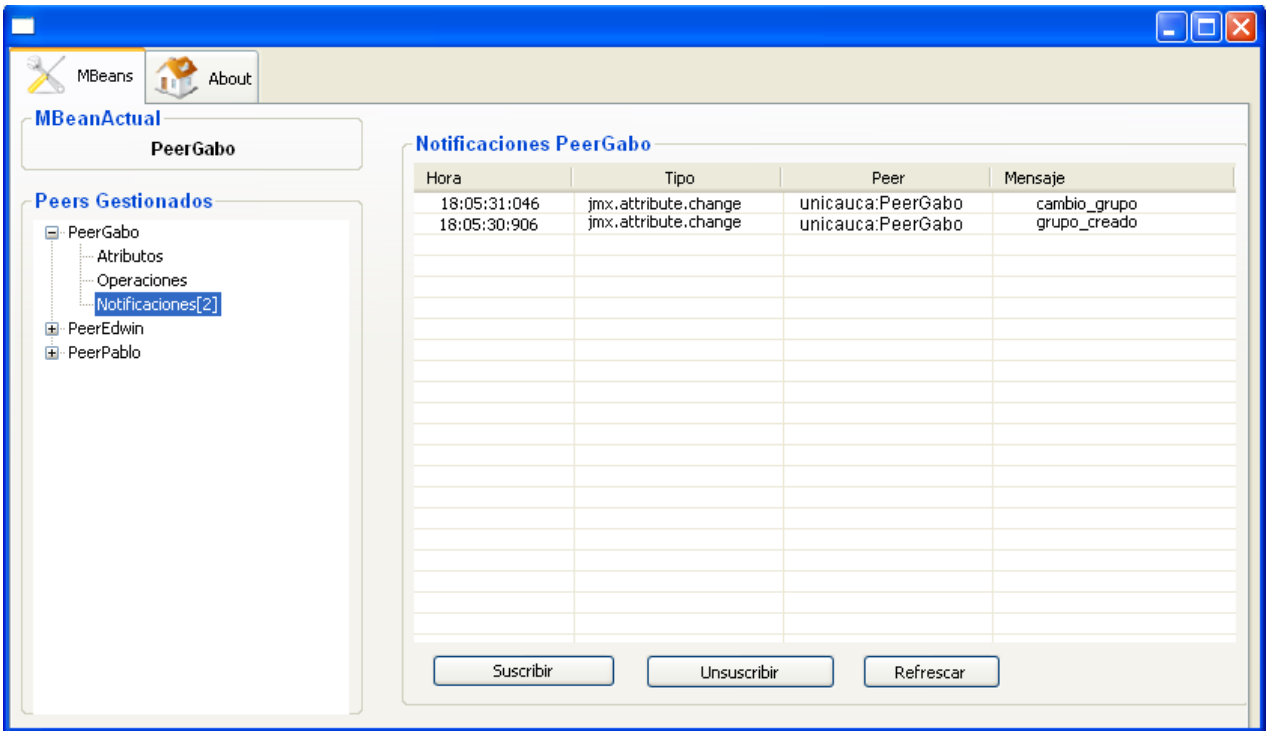
Prototipo de pantalla


Tabla 8. Caso de uso: Recibir notificaciones

4.3.3.8 Caso de uso: Obtener objetos gestionados

Código	**Nombre**		
CU-8	**Obtener objetos gestionados.**		
Resumen	Después de haber establecido conexión con el sistema el gestor envía una petición para obtener los objetos gestionados y sus características.		
Prioridad: Alta.		**Complejidad:** Alta.	
Actor(es) Involucrado(s)			
1	Gestor.		
Caso(s) de Uso Relacionado(s)			
1	Conectarse al sistema.		
Precondiciones			
1	Debe existir una conexión válida.		
Postcondiciones			
1			
Flujo Básico			
Acción del Actor	**Acción del Sistema**	**Datos de Entrada**	**Datos de Salida**
1 Envía la petición.	Procesa la petición y genera la lista de objetos gestionados.		
2 Recibe la lista de objetos gestionados.			
3 Fin.			
Excepciones			
Código:	**Nombre:**		
Acción del Actor	**Acción del Sistema**	**Datos de Entrada**	**Datos de Salida**
1			

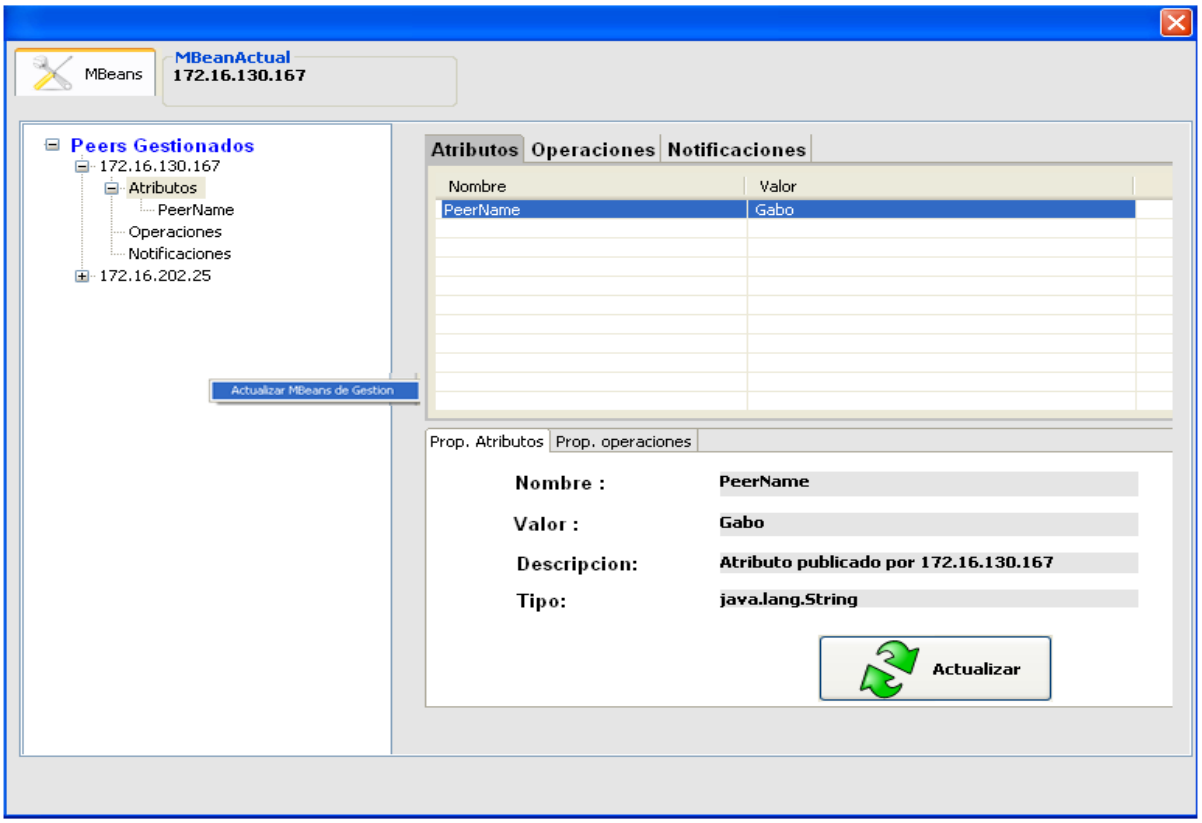
Prototipo de pantalla


Tabla 9. Caso de uso: Obtener objetos gestionados

4.3.4 Diagramas de secuencia

4.3.4.1 Diagrama de secuencia: Conectarse al sistema

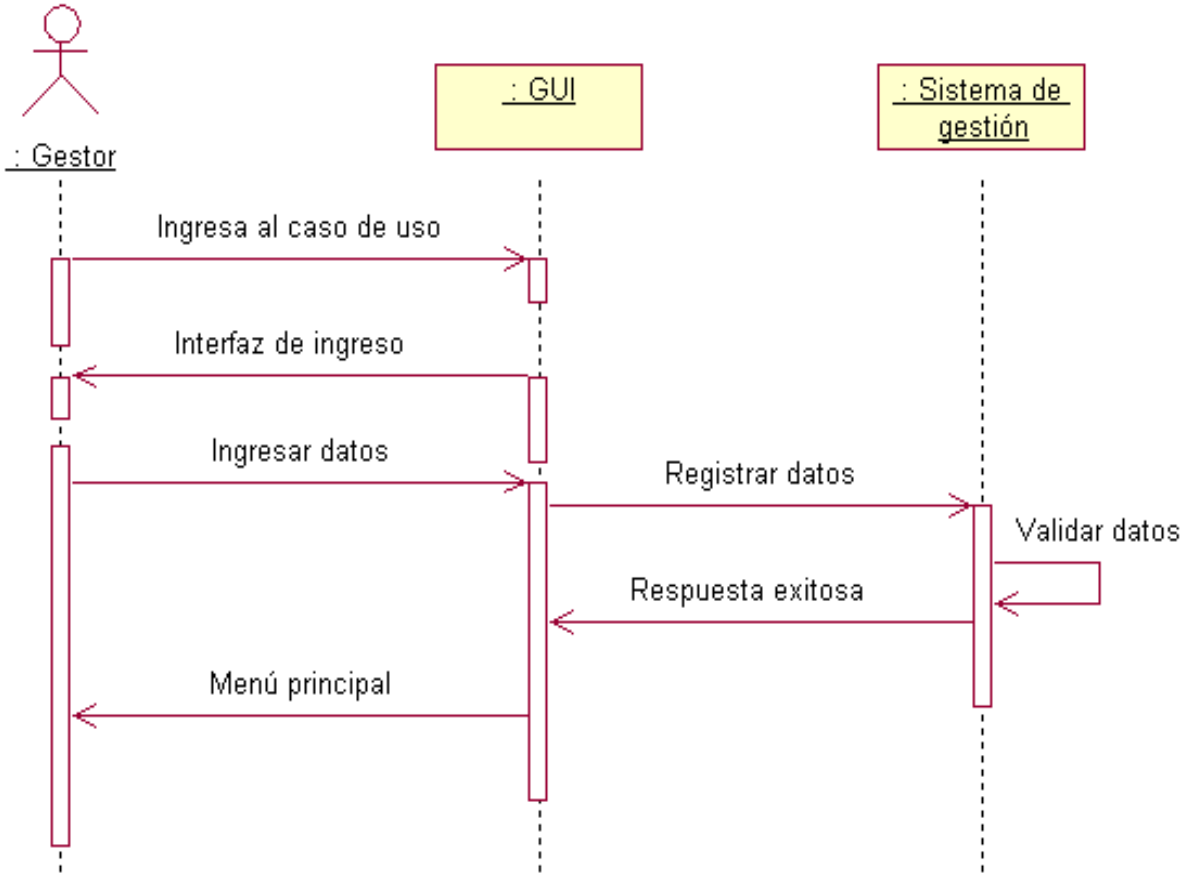


Figura 18. Diagrama de secuencia: Conectarse al sistema

“Validar datos” hace referencia a la acción que realiza el sistema de gestión y que consiste en consultar en una pequeña base de datos si el usuario que desea ingresar al sistema ya existe.

4.3.4.2 Diagrama de secuencia: Interactuar con objetos gestionados

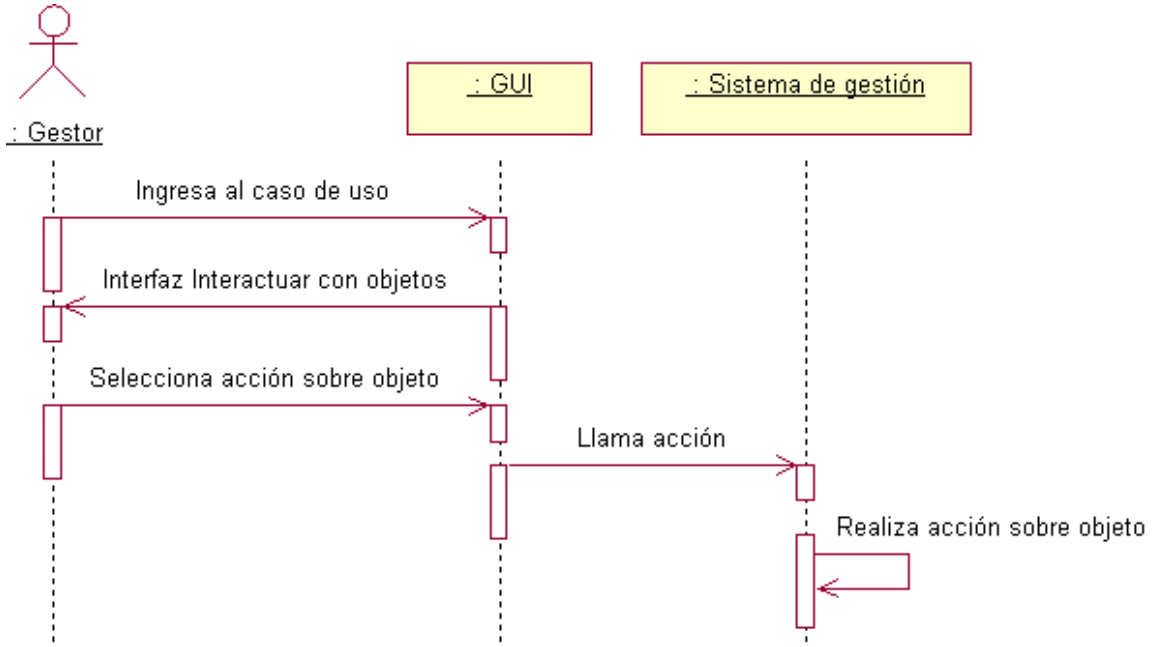


Figura 19. Diagrama de secuencia: Interactuar con objetos gestionados

"Realiza acción sobre el objeto" hace referencia a las solicitudes que le hace el sistema de gestión a los *peers* para obtener su información, por ejemplo, obtener el nombre.

4.3.4.3 Diagrama de secuencia: Invocar operaciones de gestión

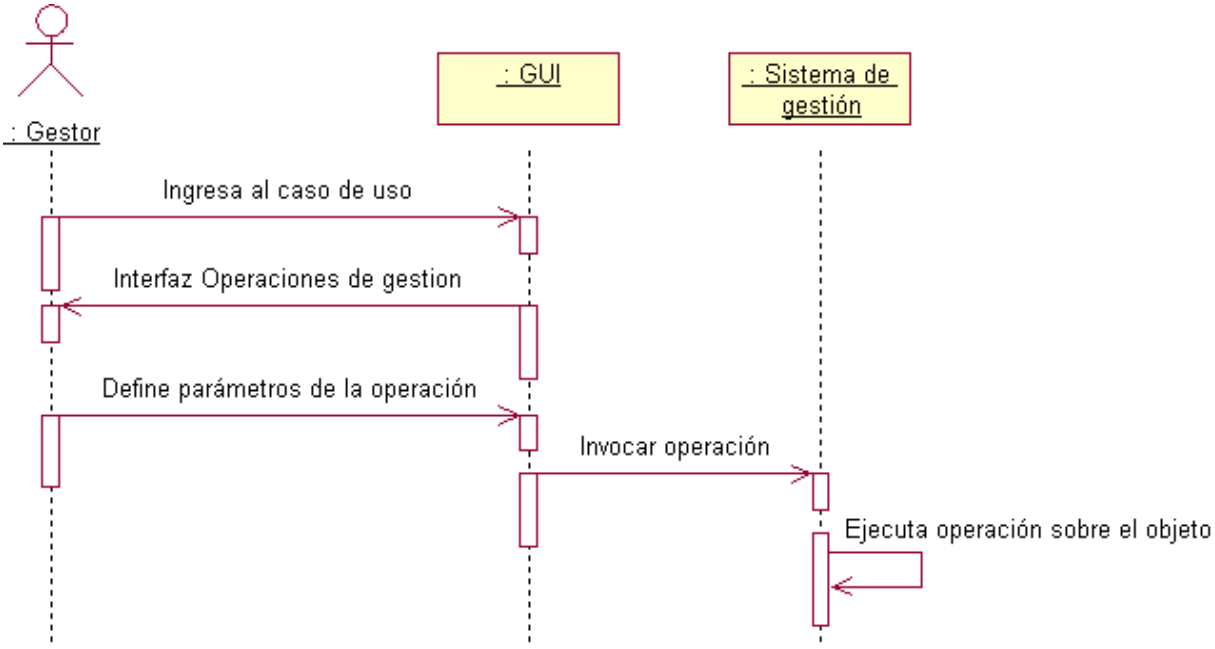


Figura 20. Diagrama de secuencia: Invocar operaciones de gestión

“Ejecuta operación sobre el objeto” hace referencia a las acciones que el sistema de gestión lleva a cabo sobre los *peers*, por ejemplo, cambiar su estado o restringir el número de conexiones de chat que puede establecer con otros *peers*.

4.3.4.4 Diagrama de secuencia: Solicitar el valor de un atributo

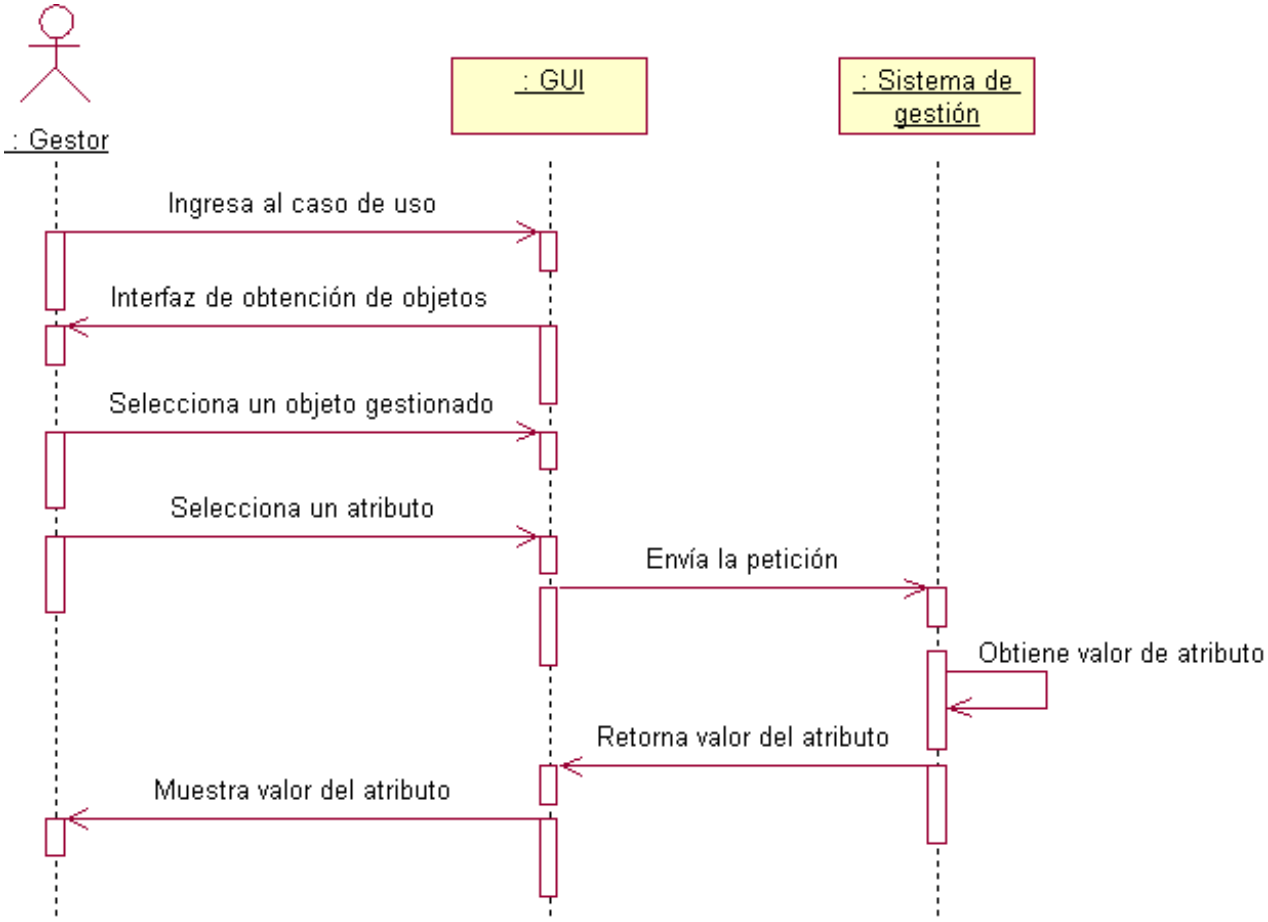


Figura 21. Diagrama de secuencia: Solicitar el valor de un atributo

4.3.4.5 Diagrama de secuencia: Modificar el valor de un atributo

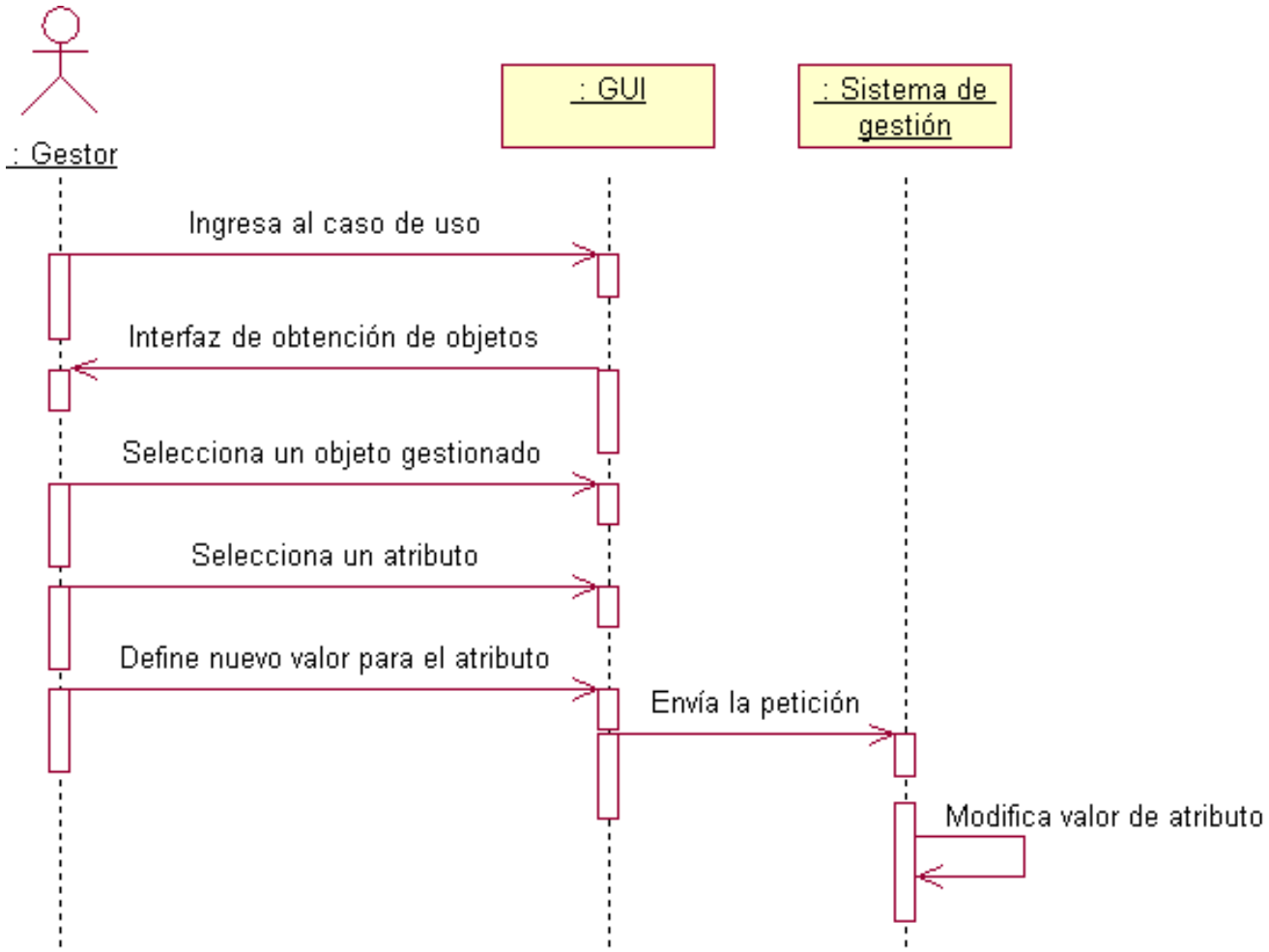


Figura 22. Diagrama de secuencia: Modificar el valor de un atributo

"Modifica valor de atributo" hace referencia al resultado de una acción que ejecuta el sistema de gestión sobre un *peer*, por ejemplo, cuando el sistema de gestión modifica el nombre de un *peer*.

4.3.4.6 Diagrama de secuencia: Emitir notificaciones

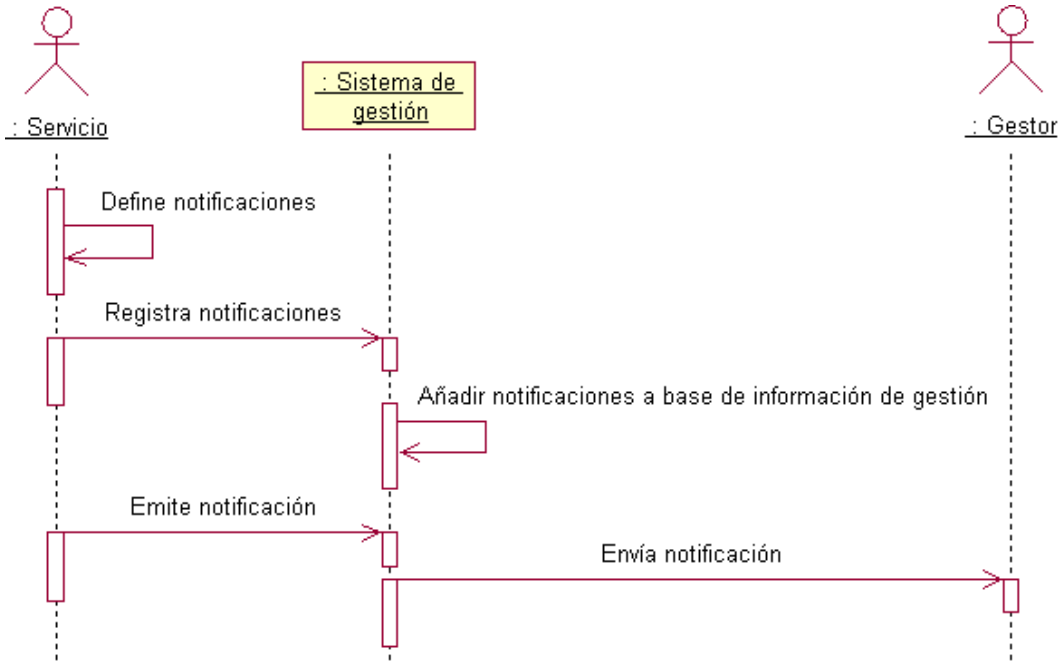


Figura 23. Diagrama de secuencia: Emitir notificaciones

4.3.4.7 Diagrama de secuencia: Recibir notificaciones

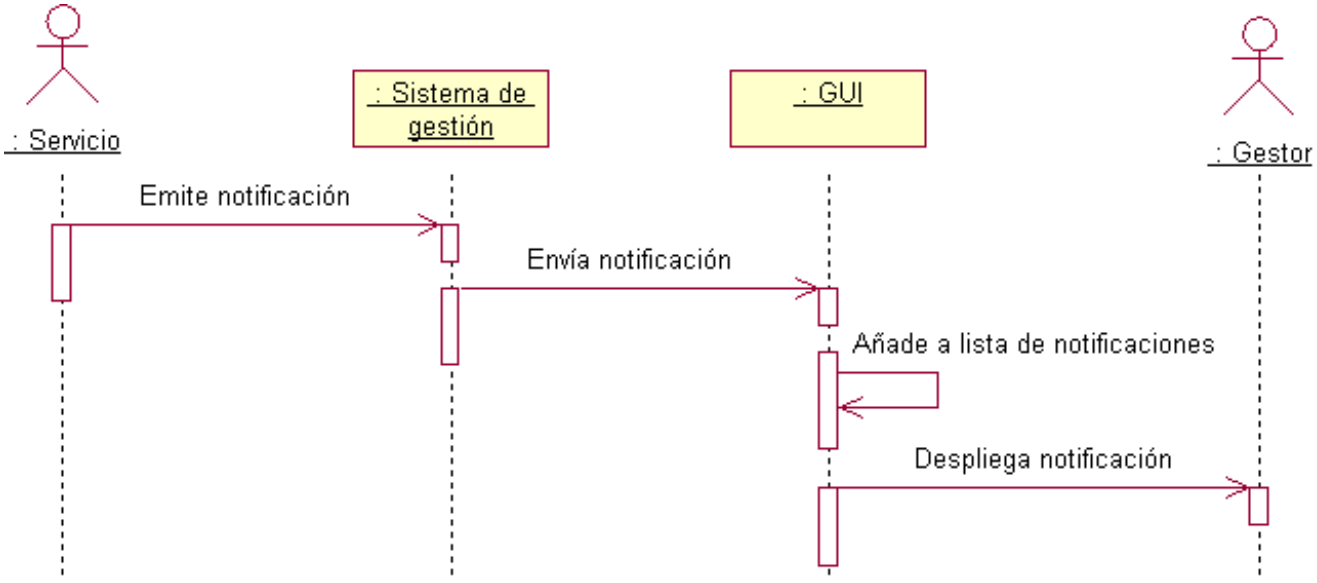


Figura 24. Diagrama de secuencia: Recibir notificaciones

4.3.4.8 Diagrama de secuencia: Obtener objetos gestionados

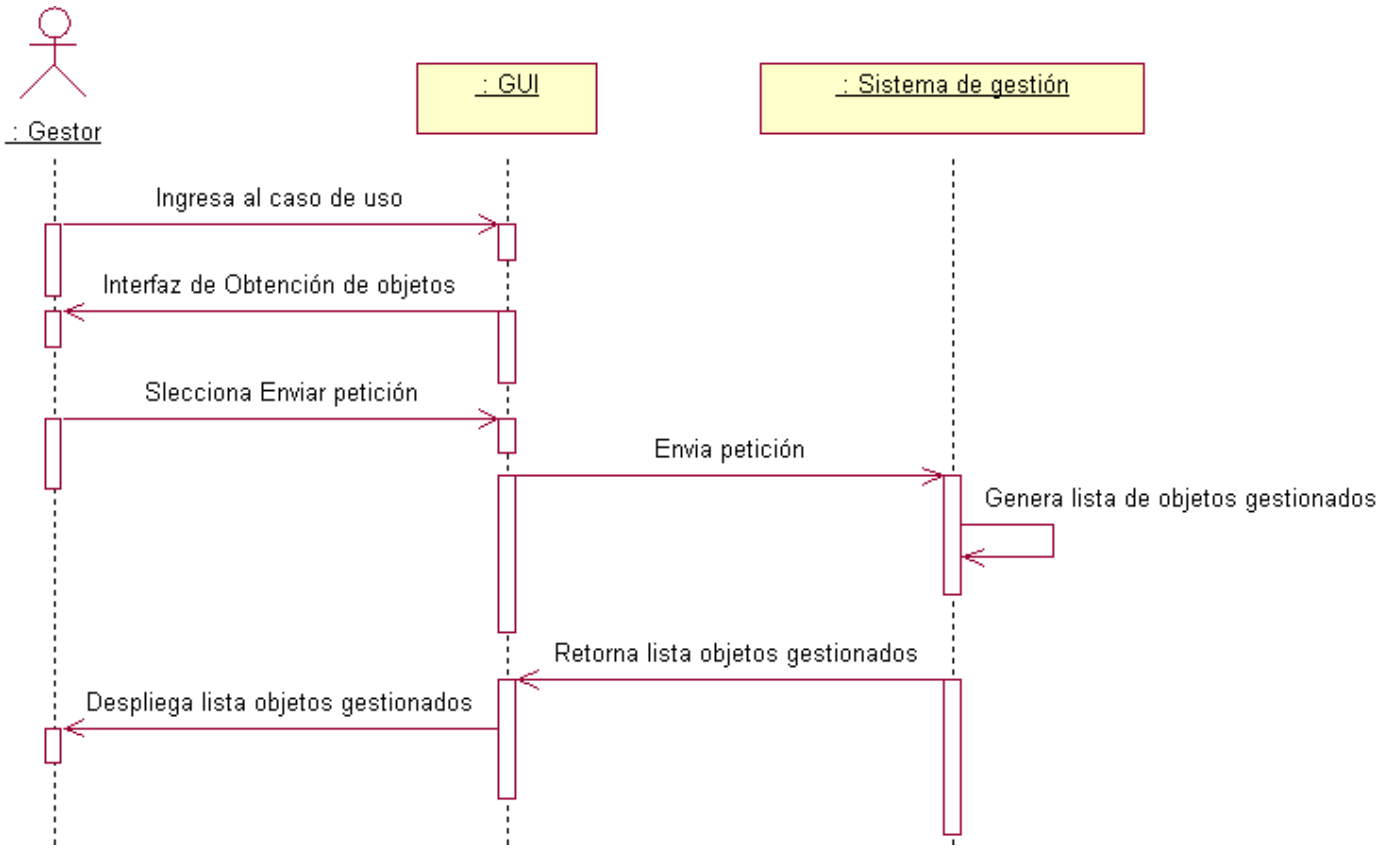


Figura 25. Diagrama de secuencia: Obtener objetos gestionados

4.4 Descripción funcional del módulo de gestión de servicios P2P

La gestión de una aplicación P2P es muy compleja debido a su comportamiento *ad-hoc*; para proveer un modelo que sirva como API de gestión de dichas aplicaciones, en el marco de este proyecto se crea una extensión de una plataforma para el desarrollo de aplicaciones P2P de propósito general, la cual ofrece interfaces de gestión que son incorporadas en una aplicación P2P de mediana complejidad (ver anexo 4), las cuales emiten notificaciones. Para el control de las notificaciones y eventos se desarrolla un modulo de gestión, el cual es el encargado de dar soporte a un grupo de servicios en una red P2P.

El entorno en el que se ha concebido el módulo de gestión es una arquitectura distribuida compuesta por un conjunto heterogéneo de servicios que presentan una serie de tareas comunes. Con el objetivo de facilitar la gestión de dichos servicios, se desea centralizar todas las notificaciones en un punto de acceso único y uniforme.

El objetivo fundamental del diseño del módulo de gestión consiste en centralizar la gestión de distintos servicios distribuidos en una red P2P, ofreciendo una interfaz uniforme de acceso a todos ellos. Las actividades de gestión de dichos servicios serán comunes en su mayoría, como por ejemplo cambiar un atributo, ejecutar un método, detener el servicio, solicitar datos básicos, listar métodos ofrecidos, entre otros.

La razón para gestionar estos servicios P2P es unificar toda la información obtenida de las consultas realizadas a los mismos en un único informe que además atienda a las interfaces de gestión incorporadas en el desarrollo.

El módulo de gestión es flexible y escalable con lo cual pretende que el sistema sea capaz de atender a las necesidades de servicios de muy distinta naturaleza, así como simplificar al máximo el proceso de incluir nuevos servicios al entorno de la aplicación P2P en la que se enmarque el sistema de gestión.

El desarrollo del sistema de gestión se realizó con base en una arquitectura de tres capas las cuales se describen:

- Capa de Presentación.
- Capa de Negocio.
- Capa de Datos.

En la capa de presentación se implementa toda la funcionalidad asociada a la fachada que se le presenta al gestor encargado de realizar el control sobre los servicios P2P en una red P2P.

En la capa de negocio se implementa toda la funcionalidad asociada a la lógica del negocio de la gestión de los servicios P2P en la red P2P. En esta capa se incorpora la funcionalidad de JXTA y JMX, los cuales ofrecen los mecanismos necesarios para que el módulo de gestión interactúe con servicios P2P. En este caso el módulo de gestión interactúa con una agenda P2P colaborativa la cual asocia tres servicios P2P específicos (pipe, descubrimiento y publicación), los cuales emiten notificaciones y eventos a dicho sistema.

En la capa de datos se tiene acceso a una base de datos la cual almacena información pertinente a la gestión de las aplicaciones P2P e información del sistema de gestión como lo es información de los usuarios administradores, información estadística de la interacción de notificaciones y eventos que los servicios P2P generan. Esto con el fin de dar soporte a la necesidad de almacenar información que nos sirva como base para tener un control más seguro de los servicios P2P asociados al sistema de gestión.

4.5 Conclusiones

La implementación de los componentes de gestión de la extensión sobre una aplicación P2P (sección 4.3) permite obtener mayor control sobre el servicio prestado, por ejemplo, gracias a dichos componentes es posible saber cuándo un usuario esta chateando, publica un evento, crea o se cambia de grupo.

La implantación de políticas de gestión permite reducir la existencia de usuarios parásitos, esto es, usuarios que consumen los recursos de red sin aportar nada a cambio. Esto es posible gracias a la gestión de los canales de subida y descarga asociados a un usuario en particular.

Debido a la descentralización característica de las redes P2P es importante resaltar que la viabilidad de los sistemas de gestión para este tipo de entornos está directamente relacionada con la capacidad que deben tener los servicios P2P para generar notificaciones de manera asíncrona, esto es, sería muy poco práctico que un sistema de gestión estuviera sondeando permanentemente la red P2P, pues los servicios que se ejecutan sobre este tipo de redes tienen comportamientos *ad-hoc*.

CAPITULO 5

PRUEBAS

Las pruebas son un elemento crítico para la calidad del software. La importancia de los costos asociados a los errores promueve la definición y aplicación de un proceso de pruebas minucioso y bien planificado. Las pruebas permiten validar y verificar el software, entendiendo como validación el proceso que determina si el software satisface los requisitos y verificación como el proceso interno que determina si los productos de una fase satisfacen las condiciones de dicha fase [26].

Las pruebas de caja negra se llevan a cabo sobre la interfaz del software, intentando demostrar que las funciones son operativas, que las entradas se manejan de forma adecuada y que se produce el resultado esperado [26].

Las pruebas de caja blanca se centran en la estructura lógica interna del software y se basan en un examen detallado de los procedimientos y caminos lógicos del sistema [26].

Las pruebas aquí realizadas se basan en el enfoque de caja negra debido a las peculiaridades del sistema de gestión como lo es la arquitectura Agente/Gestor, la cual conlleva consigo la complejidad de las conexiones de red, además de la dificultad exponencial de realizar pruebas de caja blanca basadas en el seguimiento de todos los caminos posibles.

Las pruebas de caja negra buscan hallar errores en cinco categorías [26]:

1. Funciones incorrectas o ausentes.
2. Errores de interfaz.
3. Errores en estructuras de datos o en el acceso a bases de datos externas.
4. Errores de rendimiento.
5. Errores de inicialización y terminación.

A continuación se describe el plan de pruebas mediante el cual se busca cubrir el mayor número posible de las categorías anteriormente mencionadas seguido de la descripción detallada de la aplicación P2P en conjunto con la aplicación de gestión las cuales agrupan la funcionalidad de la prueba de la extensión de la plataforma en donde se implementa un conjunto de interfaces de gestión basadas en un sistema de notificación de eventos proporcionado por RMI y JMX, que permiten extender una aplicación P2P agregando *listeners* y manejadores de eventos necesarios para poder leer y escribir atributos y eventos de los nodos de una red P2P, de los que se obtiene las referencias necesarias para desarrollar sobre las mismas capas de gestión y monitorización de mas alto nivel. La idea es mostrar las bases para desarrollar capas de integración de más alto nivel, bases que proporciona la extensión para la gestión de servicios.

5.1 Plan de pruebas

Durante todo el proceso de implementación del sistema de gestión se llevaron a cabo pruebas de unidad e integración. Las pruebas realizadas al término del desarrollo del sistema se dividen en dos fases: Pruebas de requerimientos y pruebas de funcionalidad.

5.1.1 Pruebas de requerimientos

Las pruebas de requerimientos se basan en la validación de los requisitos iniciales del sistema [26]. En este caso particular los requisitos se dividen en funcionales y no funcionales (ver capítulo 4, sección 4.1.2) y con este tipo de pruebas se busca comprobar que cada uno de ellos, de acuerdo a su prioridad, se obtuvo satisfactoriamente con el resultado final.

En las tablas 1 y 2 se plasma el estado final de los requisitos del sistema de gestión.

Requisitos Funcionales	Prioridad	¿Cumple?		Comentarios
		Si	No	
Gestión proactiva	Obligatorio	✓		Los servicios informan al sistema de gestión la variación de sus atributos de gestión mediante la generación de notificaciones.
Ejecución de procesos remotos	Opcional	✓		El gestor tiene la posibilidad de ejecutar operaciones remotas sobre los servicios.
Despliegue de políticas de gestión	Opcional	✓		El gestor puede especificar el comportamiento de un servicio cuando este cumple con ciertas condiciones.

Tabla 10. Estado final de los requisitos funcionales

Requisitos No Funcionales	Prioridad	¿Cumple?		Comentarios
		Si	No	
Flexibilidad de la extensión	Obligatorio	✓		El sistema de gestión es flexible y permite añadir nuevos elementos para optimizar la gestión de los recursos y servicios de red.
Distribución libre	Obligatorio	✓		El sistema de gestión se distribuye bajo los términos de la Licencia Pública General tal como lo publica la *Free Software Foundation*.

Tabla 11. Estado final de los requisitos no funcionales

5.1.2 Pruebas de funcionalidad

Las pruebas de funcionalidad buscan verificar si el comportamiento del sistema es el adecuado y si los métodos generan los resultados esperados.

Las pruebas están basadas en los casos de uso del sistema de gestión (ver capítulo 4, sección 4.3.2) y sus resultados se describen en la siguiente tabla.

Funcionalidad Probada	¿Correcto?		Comentarios
	Si	No	
Conectarse al sistema	✓		La validación del nombre de usuario y la contraseña para el ingreso al sistema se realiza satisfactoriamente.
Interactuar con objetos gestionados	✓		El gestor tiene la posibilidad de interactuar con los objetos gestionados mediante una serie de acciones definidas.
Invocar operaciones de gestión	✓		El gestor invoca con éxito operaciones de gestión que modifican el estado de los objetos.
Solicitar el valor de un atributo	✓		El gestor obtiene el valor de los atributos deseados sin ningún problema.
Modificar el valor de un atributo	✓		El gestor modifica el valor de los atributos deseados sin ningún problema.
Recibir notificaciones	✓		El gestor recibe oportunamente las notificaciones generadas por los servicios cada vez que ocurre el suceso desencadenante.
Obtener objetos gestionados	✓		El gestor obtiene sin problemas los objetos gestionados y sus características.

Tabla 12. Resultado de las pruebas funcionales para el sistema de gestión

5.1.3 Funcionalidad de la agenda colaborativa

Para la validación de la extensión se desarrolla una aplicación P2P, la cual agrupa tres servicios específicos dentro de su funcionalidad (pipe, descubrimiento y publicación).

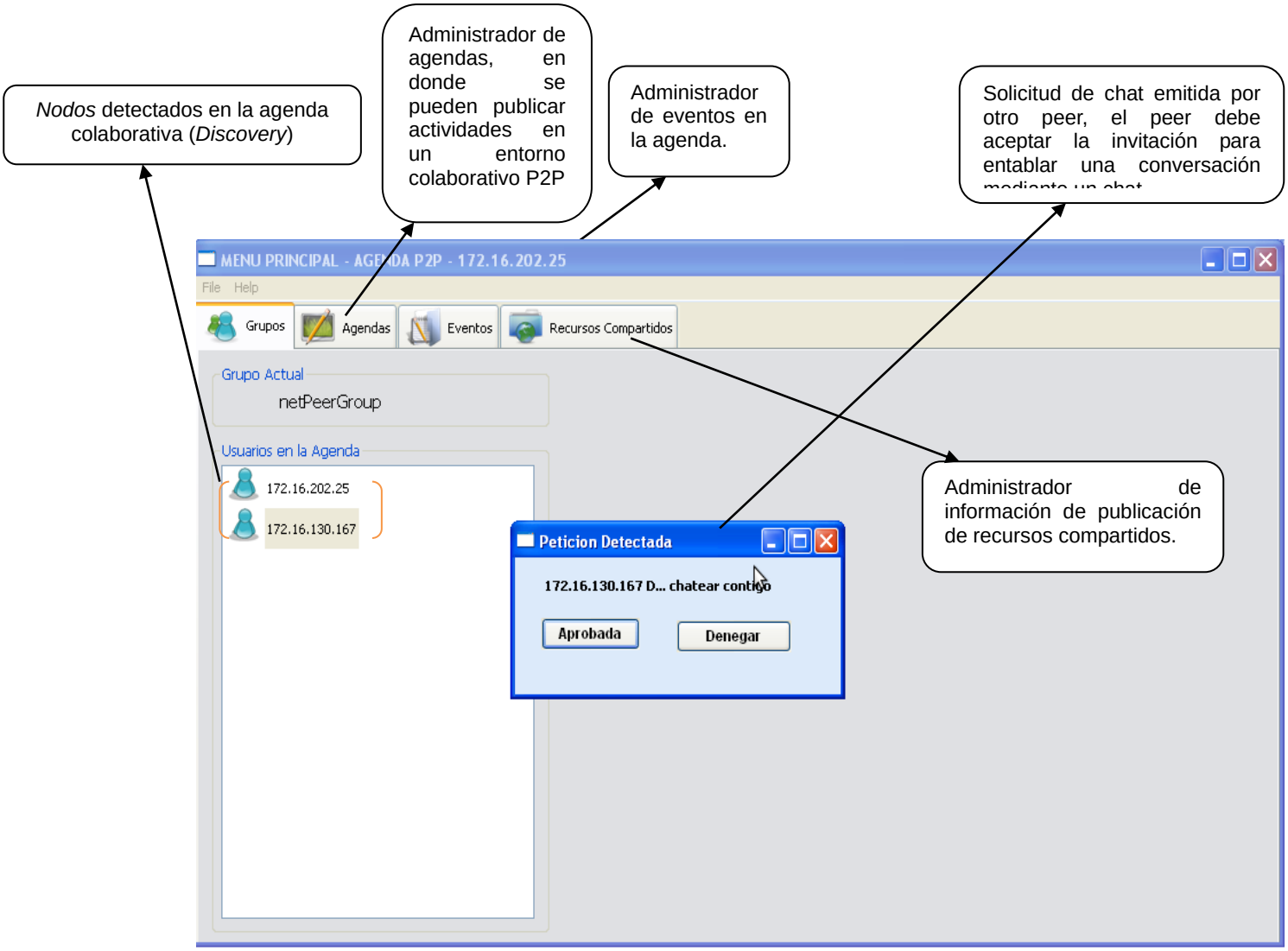


Figura 26. Opciones de la agenda colaborativa

La figura 26 ilustra las diferentes opciones que ofrece la agenda colaborativa P2P. Dentro de su funcionalidad el aplicativo de la agenda colaborativa detecta usuarios que están conectados, en este caso cada usuario al registrarse por defecto se registra en el grupo *NetPeerGroup*, como podemos observar en la figura 26 cada Agenda tiene su lista de usuarios conectados en el momento, esta lista se actualiza automáticamente detectando así cuando un usuario se conecta o desconecta del aplicación.

La aplicación cuenta con la funcionalidad de búsqueda de grupos, usuarios, y demás recursos, que los otros nodos desean compartir.

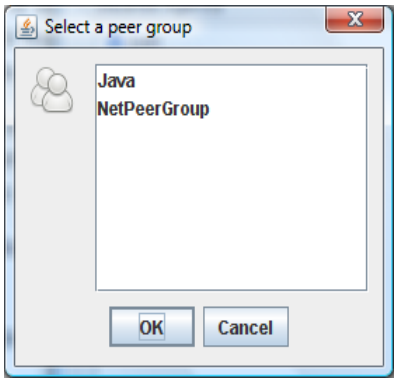


Figura 27 Selección de un grupo a unirse

En la figura 27 observamos los diferentes grupos disponibles en la red, podemos ver el grupo Java creado, y el *NetPeerGroup* disponibles para que los usuarios de la agenda puedan asociarse a estos.

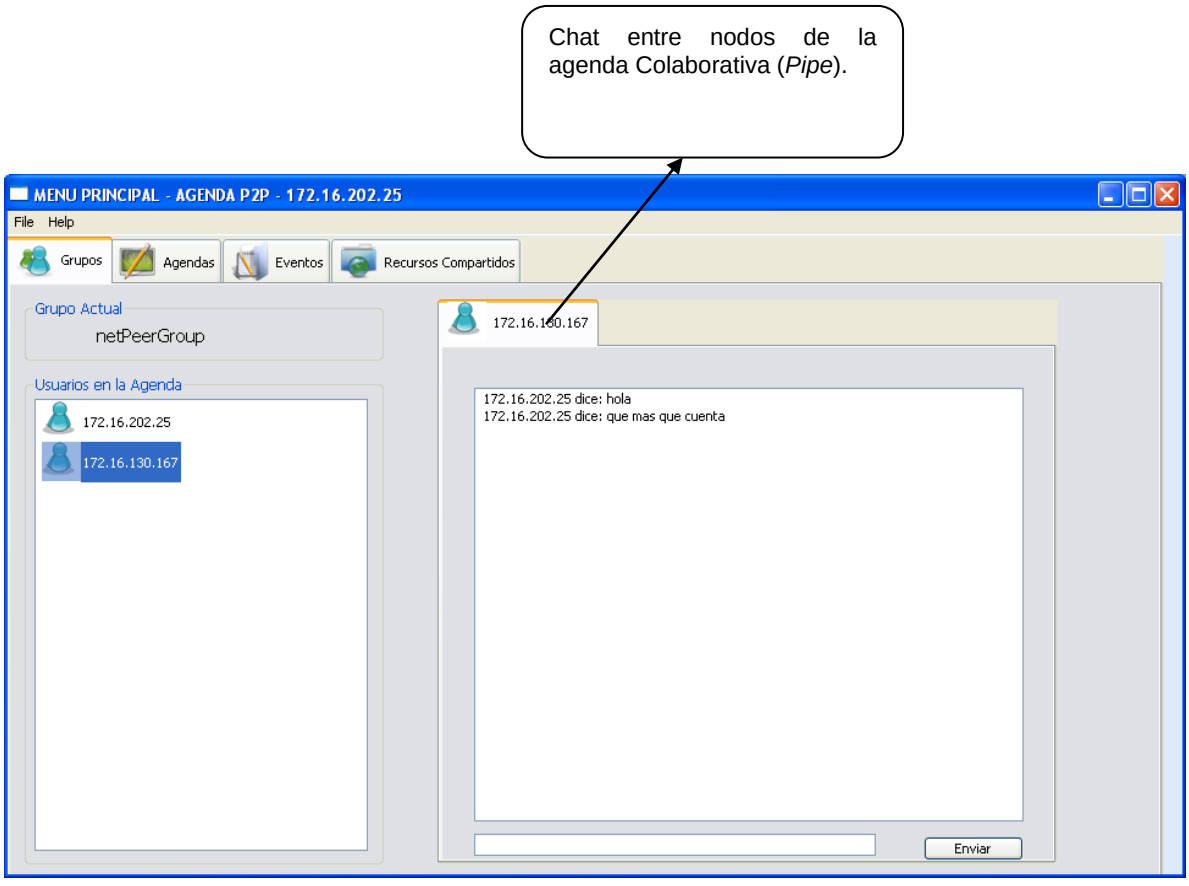


Figura 28. Funcionalidad del servicio *pipe*

La figura 28 ilustra la funcionalidad del servicio de *Pipe* asociado en la agenda colaborativa P2P, el cual permite interactuar con otros nodos a través de un chat.

Cada usuario puede chatear con otro usuario que este conectado enviando una petición de chat; cuando esta es aceptada el usuario puede iniciar una conversación.

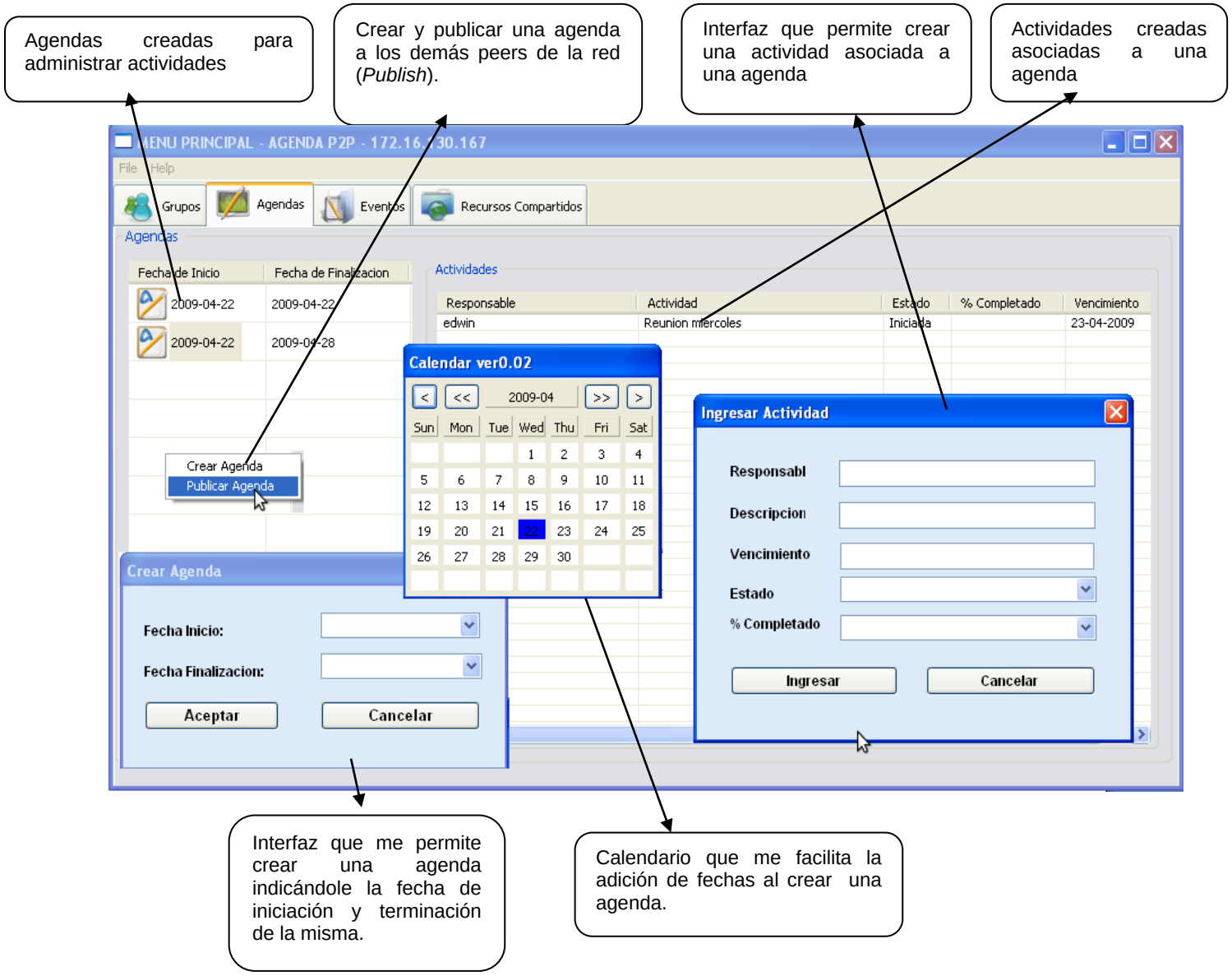


Figura 29. Funcionalidad para la creación de agendas

La figura 29 muestra las diferentes funcionalidades que componen la creación de agendas y actividades dentro del entorno de la agenda colaborativa P2P.

En la agenda compartida se pueden registrar y eliminar las actividades que un equipo de trabajo deberá cumplir o desarrollar en una fecha limite. Cada actividad es registrada por el usuario con el rol de grupo y cada participante podrá modificarlo pero con algunas restricciones.

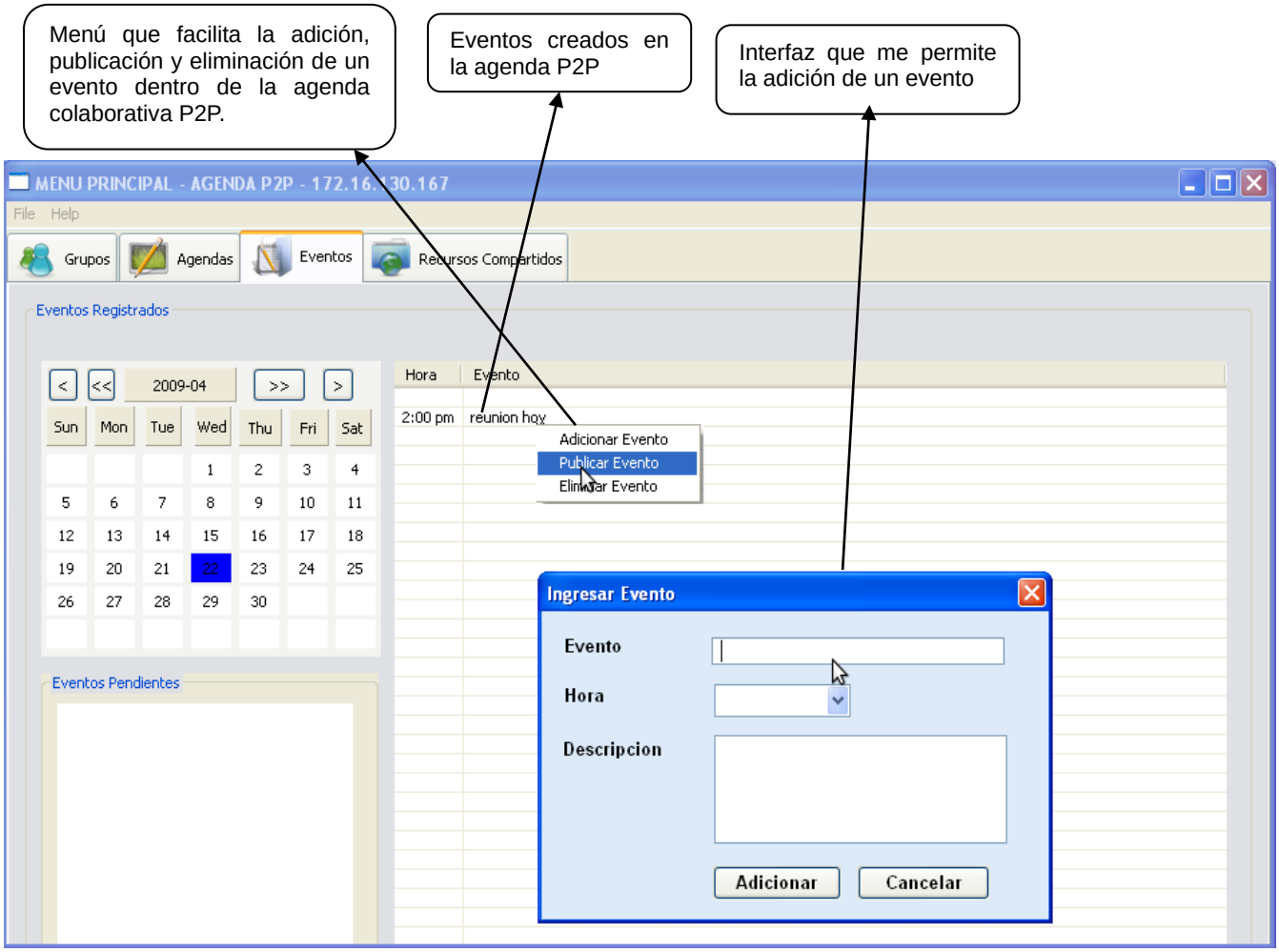


Figura 30. Funcionalidad para la creación de eventos

La figura 30 muestra las diferentes funcionalidades que componen la creación de eventos dentro del entorno de la agenda colaborativa P2P. El administrador de la agenda cuenta con funcionalidades de ingreso, eliminación y consulta de eventos o sucesos para un grupo. La aplicación alerta al usuario de los eventos próximos a suceder.

5.1.4 Funcionalidad del sistema de gestión de servicios P2P

La aplicación de gestión tiene como propósito general dar soporte a interfaces de gestión las cuales están integradas en aplicaciones P2P de propósito general que agrupan servicios en una red P2P.

Por medio de estas interfaces nuestra extensión provee un mecanismo base para proveer administración, que en las redes P2P es más complejo de implementar, a diferencia de los sistemas centralizados, y con esto se busca proveer acceso a información relacionada con el funcionamiento de cada peer en la red.

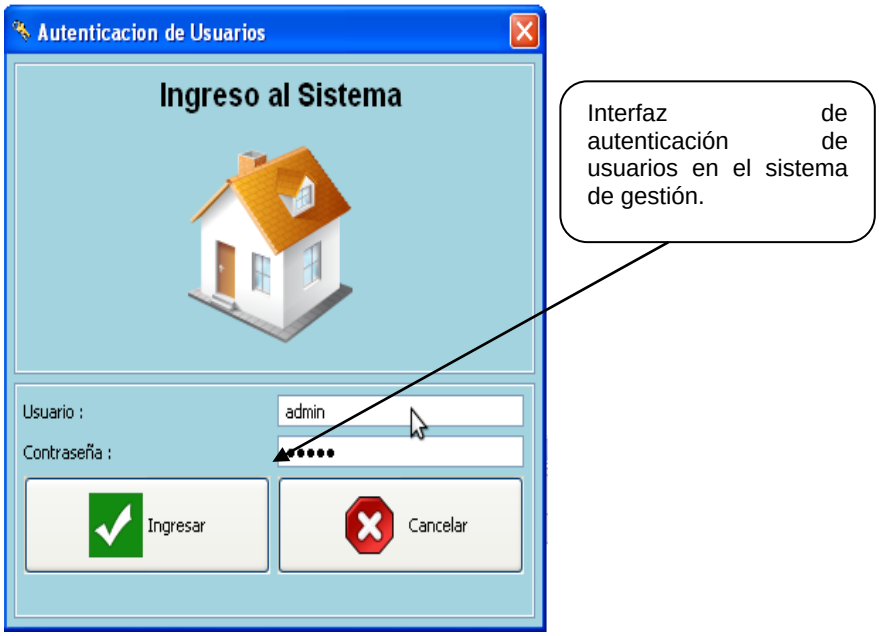


Figura 31. Ingreso al sistema de gestión

Es importante tener un control en el acceso a la aplicación, ya que la información que se procesa será solamente de interés de quien desea administrar diferentes aspectos del sistema que se ejecuta sobre la red.

La Figura 31 ilustra la autenticación de la identidad de usuario, el cual es el encargado de monitorear la información que el sistema de gestión obtiene en una red P2P.

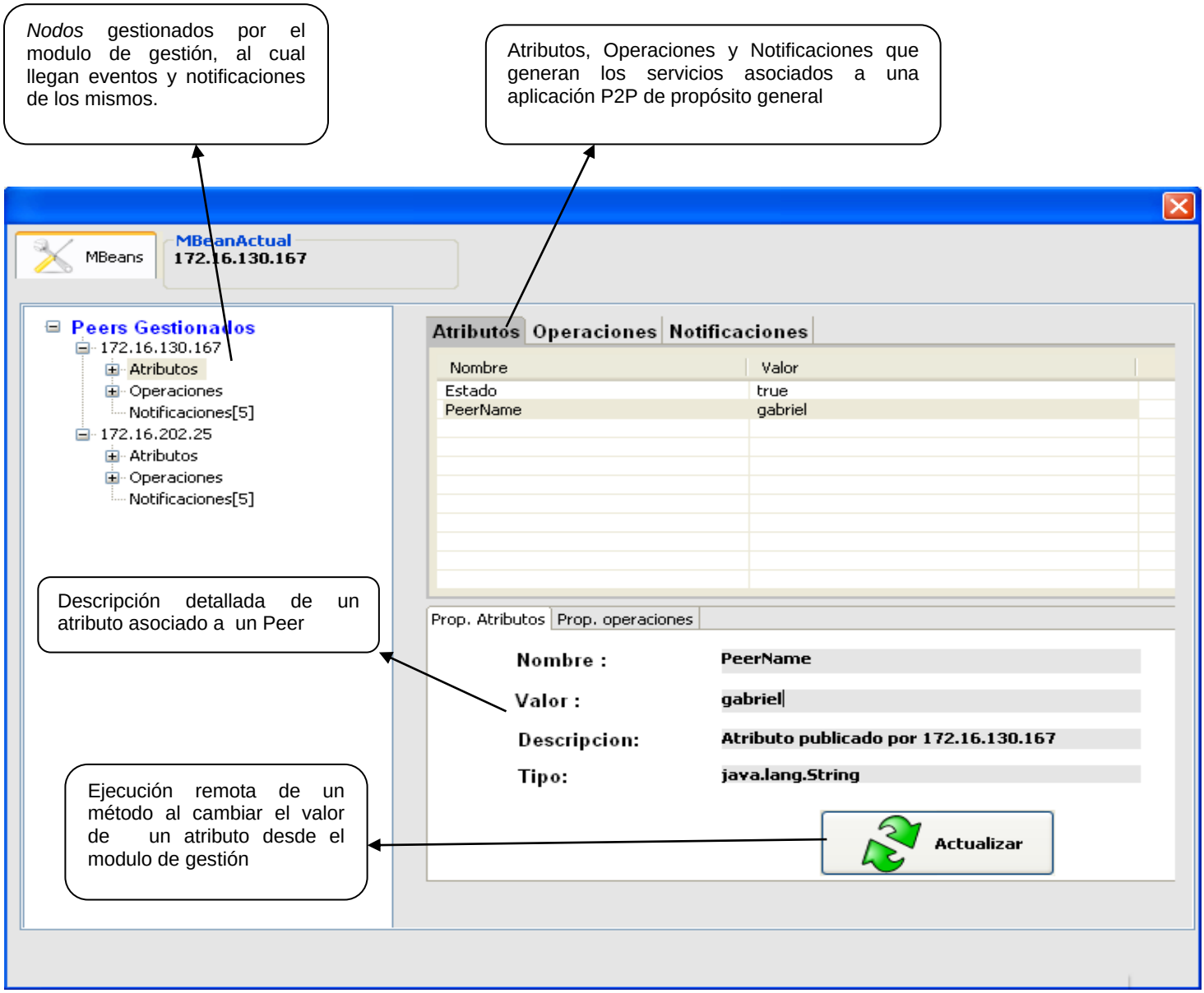


Figura 32. Interfaz del sistema de gestión

La aplicación de gestión implementa un servicio de descubrimiento el cual identifica todos los nodos asociados a una red P2P.

En la etapa de descubrimiento se obtiene los siguientes elementos.

- Identificación del nodo (datos generales del nodo)
- Atributos generales de los peers.

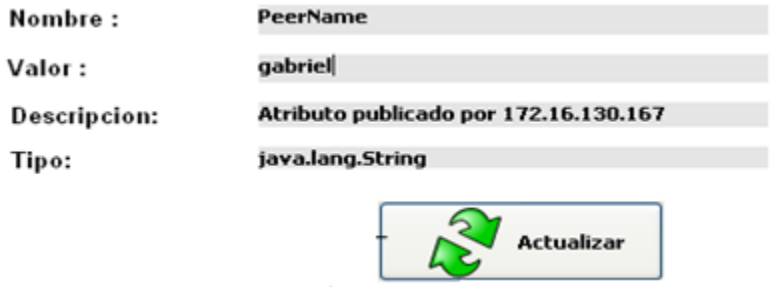
Los datos generados por el servicio de descubrimiento del sistema de gestión pueden ser consumidos por un sistema de control de usuarios, donde se ejerzan políticas de control de acceso, o privilegios de uso.

En la parte derecha de la interfaz de la aplicación de gestión se especifica los atributos generales de gestión de cada nodo

Los atributos de gestión describen las características generales de un nodo específico, como lo son el nombre, el estado entre otros.

Con estos atributos se puede identificar las características de un peer y poder personalizar diferentes funcionalidades.

Cada atributo se puede describir con más detalle, y así obtener información mas relevante que describa al nodo.



Los atributos pueden ser administrados por el sistema de gestión de forma que permita, por ejemplo, la modificación del nombre del nodo, del identificador, entre otros atributos.

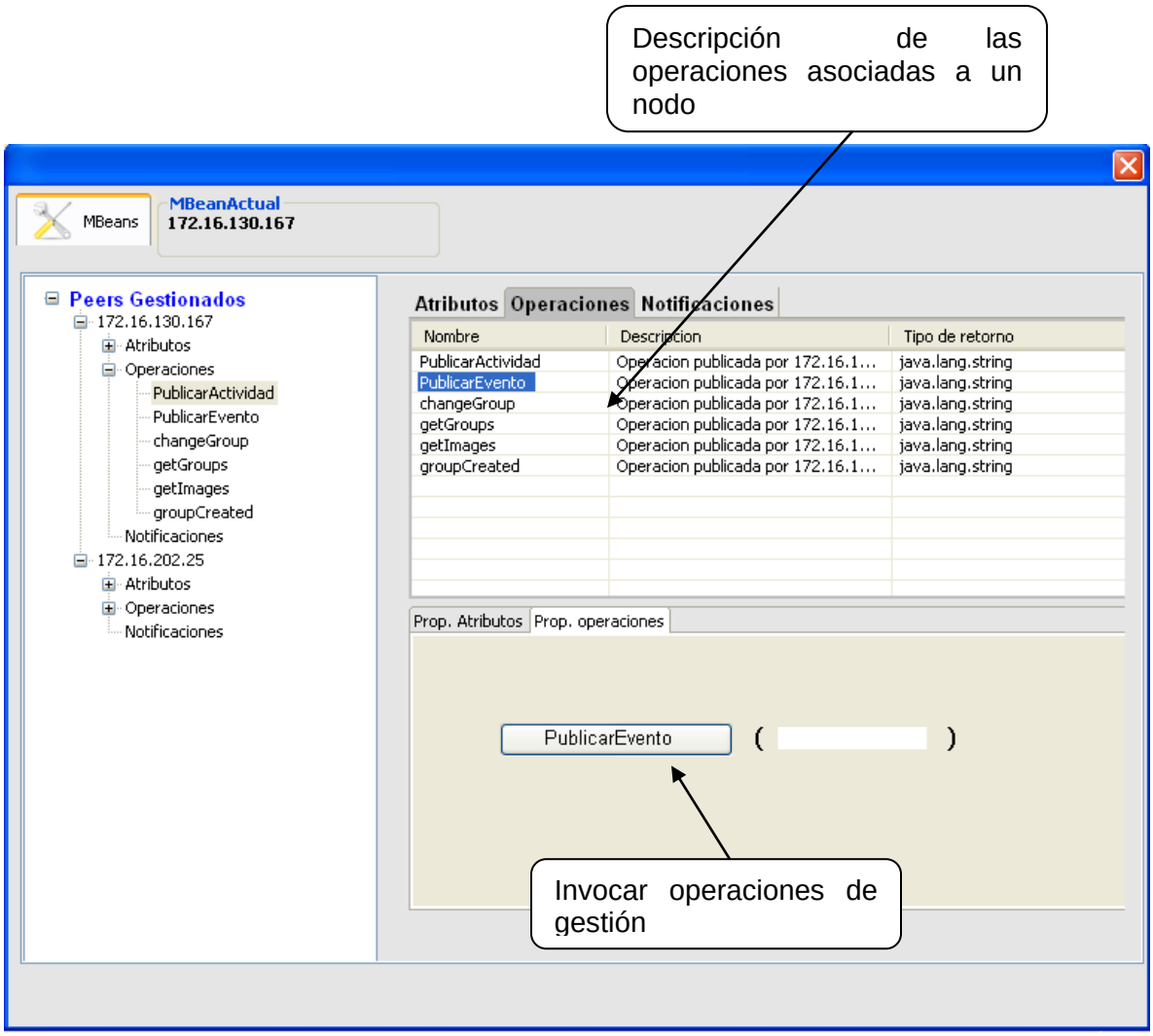


Figura 33. Invocación de operaciones de gestión

La Figura 33 ilustra la funcionalidad general al invocar una operación que esta asociada a un nodo, un ejemplo mas descriptivo es cuando el sistema desea obtener los grupos asociados a un peer; mediante la invocación del método *getGroups()* el sistema de gestión podrá obtener los grupos.

Esta funcionalidad proporciona capacidades para implementar controles más robustos para gestionar credenciales u otros sistemas de identificación y unión de peers a las redes.

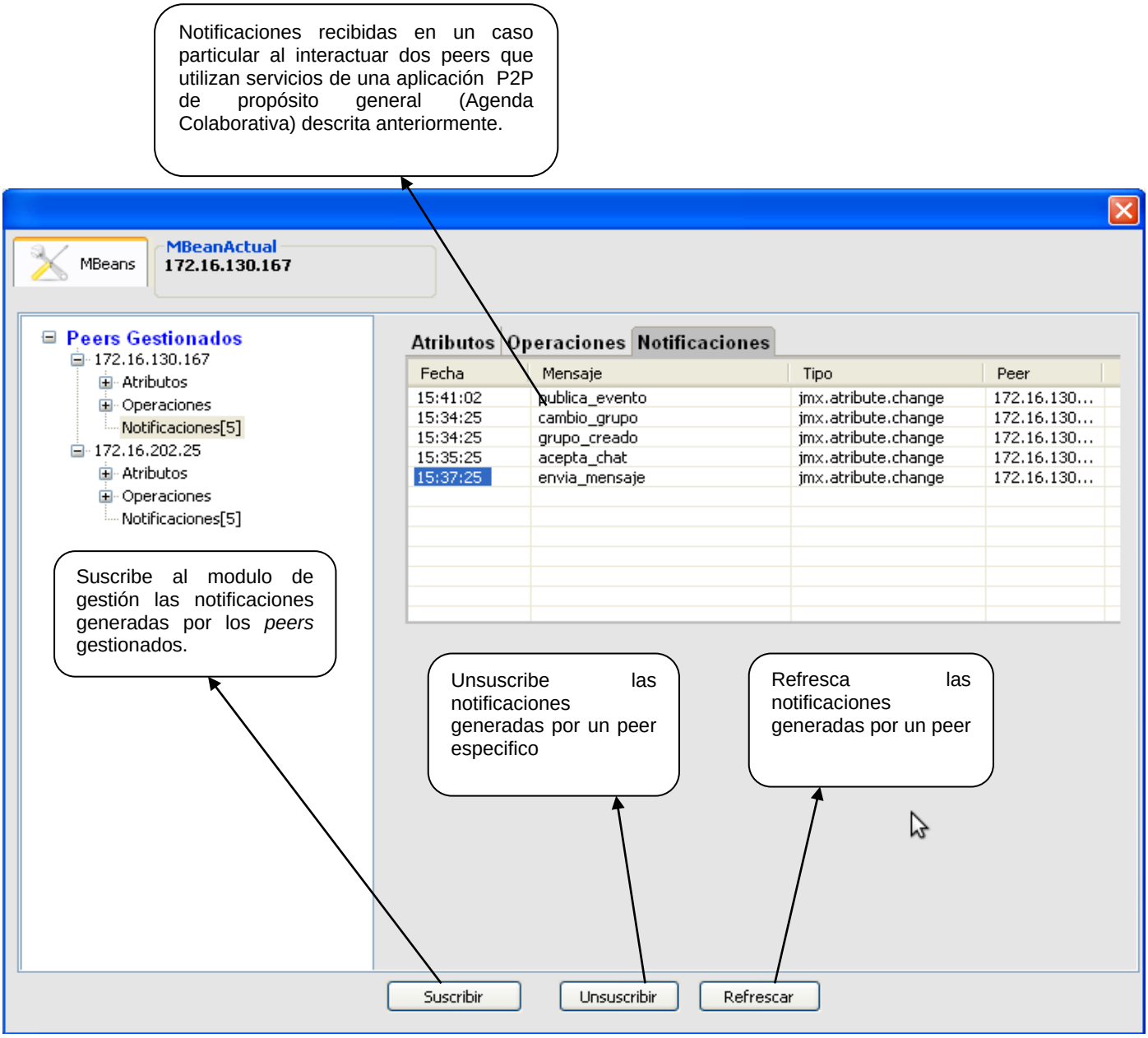


Figura 34. Recepción de notificaciones

La Figura 34 muestra diferentes notificaciones recibidas en el sistema de gestión que generan los servicios P2P, la cual agrupa diferentes funcionalidades como lo son:

Suscribir: esta opción permite al sistema de gestión activar el envió de notificaciones desde el nodo hacia el sistema, esto con el fin de administrar los nodos que solo le interesen al nodo gestor.

Unsuscribir: impide el envió de notificaciones de un peer al sistema de gestión.

Refrescar: Actualiza las notificaciones asociadas a un nodo.

5.2 Conclusiones

Las notificaciones sirven como un punto para gestionar estados de un peer en un momento determinado. Gestionar estados es poder administrar o poder tener mas control sobre el rol de un peer en la red.

La eficiencia en una red se logra con el control de trafico, cantidad de nodos que acceden a una red, con la cantidad de trabajo que esta generando un nodo, todo esto con el fin de volver mas robusta la red p2p.

La información que el sistema de gestión administra es fundamental como base para la gestión de redes P2P, puede ser utilizada para formar criterios para el control de tráfico, paquetes, control de seguridad, como también para que las implementaciones de redes P2P sean más seguras, eficientes y se pueda proveer a terceros los cuales puedan tener más capacidades de control y administración de la red.

Los usuarios de una red se pueden restringir de acuerdo a varios criterios, que dependen del propósito de la red P2P, por ejemplo en una red de distribución de contenidos, se puede restringir de acuerdo a la proporción de la subida y bajada de archivos en la red; en una red de distribución de procesamiento, o también llamada de procesador compartido, la política puede estar orientada por la continuidad y duración de la conectividad de un nodo en la red. Es decir, la funcionalidad de la red define las reglas y requisitos que los nodos deben cumplir.

La extensión ofrece un medio para poder obtener meta-información de los nodos en una red, mediante interfaces de gestión incorporadas en los servicios las cuales interactúan como agentes encargados de obtener estos datos que pueden servir como puntos para determinar la restricción o no a ciertos nodos en una red P2P.

Esta información puede ser utilizada por terceros que deseen controlar, por ejemplo, el ancho de banda el cual es afectado por el tráfico que generan los nodos en la red, el cual determina el tamaño de los paquetes que viajan. Con nuestra aplicación de gestión podemos supervisar el trafico normal que genera un nodo, por medio de notificaciones y eventos como lo son, mensajes enviados, creación de grupos entre otros; a través de la incorporación de una interface de gestión a la aplicación que agrupa uno o varios servicios, con lo cual dependiendo de la información que el nodo genera, se pueda formar un criterio para saber si un nodo esta alterando el flujo normal en la red, y así se pueda ejercer este tipo de controles.

Según los servicios que presta una red p2p se podría controlar el número de nodos conectados a esta. En las aplicaciones P2P que incorporan interfaces de gestión se obtiene la información necesaria para conocer cuantos nodos están conectados a la red lo cual puede ser tomado como referencia para controlar la conexiones de la red.

Todos los factores anteriormente mencionados influyen en la calidad de servicio general de la red, nuestra extensión es una contribución a la gestión de los factores funcionales a nivel de servicios que caracterizan las funcionalidades de una red P2P.

CAPITULO 6

RECOMENDACIONES, CONCLUSIONES, PERSPECTIVAS Y TRABAJO FUTURO

En este trabajo de grado se han integrado los esfuerzos realizados en otros proyectos relacionados llevados a cabo en esta facultad. Específicamente, en este trabajo se modeló e implementó una extensión que agrega nueva funcionalidad a una plataforma para el desarrollo de aplicaciones P2P de propósito general, extensión que posibilita y facilita el soporte a un sistema de provisión de recursos y servicios en un entorno de redes superpuestas P2P. Para cumplir con este objetivo general se llevó a cabo la división del mismo en una serie de objetivos interrelacionadas de propósito más específico. A continuación se describen las actividades desarrolladas para dar cumplimiento a dichos objetivos.

6.1 Cumplimiento de objetivos

La primera actividad, enmarcada en el capítulo 2, consistió en hacer un recuento de la historia de la computación P2P, buscando situar al lector en el contexto que dio origen a este tipo de sistemas y sus implicaciones tecnológicas, económicas, legales y sociales. Luego se realizó una clasificación de los tipos de redes P2P, se detallaron sus características más relevantes y finalmente se identificaron los tipos de aplicaciones dentro de los cuales se enmarcan los sistemas P2P actuales.

Un estudio exhaustivo permitió identificar las plataformas más relevantes para el desarrollo de aplicaciones P2P, a saber: Anthill, GNUnet, Gnutella, JXTA y P2PS (ver capítulo 3, sección 3.1). Posteriormente se realizó una comparación entre dichas plataformas (ver capítulo 3, sección 3.2) a la luz de las características más relevantes de los sistemas P2P identificadas en [3] (ver capítulo 2, sección 2.3), comparación de cuyo resultado se pudo concluir que la plataforma más flexible y adaptable y por lo tanto más apropiada para el desarrollo de aplicaciones P2P de propósito general es JXTA. El desarrollo de estas actividades permitió dar cumplimiento al primer objetivo específico (ver capítulo 1, sección 1.2). Finalmente y luego de estudiar las características de diferentes tecnologías de gestión de redes, a saber: OSI-SM, SNMP, WBEM y JMX (ver capítulo 3, sección 3.4) se llegó a la conclusión de que JMX era la tecnología de gestión que mejor se adaptaba a las necesidades de éste proyecto (ver capítulo 3, sección 3.5) razón por la cual fue elegida como la tecnología de gestión sobre la cual fundamentar el diseño de la extensión propuesta.

La especificación de los requisitos para la creación de la extensión propuesta que soportara el sistema para la provisión de servicios P2P, permitió la consecución del segundo objetivo específico. Posteriormente se llevó a cabo el diseño del prototipo de la extensión para la gestión de servicios P2P, para lo cual se mezclaron componentes tanto de JXTA como de JMX, lográndose así el tercer objetivo específico. Para cumplir con el cuarto y último objetivo específico planteado, se desarrolló una aplicación P2P de mediana complejidad (ver capítulo 4, sección 4.3), mediante la cual se validó la extensión

para la gestión de servicios P2P (ver capítulo 5, sección 5.1.3). Todas estas actividades se enmarcaron dentro de los capítulos 4 y 5.

6.2 Trabajo futuro

Los resultados obtenidos en este trabajo de grado abren el camino a una nueva serie de actividades de investigación relacionadas con las redes superpuestas P2P. Por ejemplo, en la actualidad no existe una arquitectura estándar para el desarrollo de aplicaciones P2P ni tampoco se poseen los documentos que especifiquen los servicios que este tipo de redes puedan soportar.

El potencial del P2P como sistema de distribución de contenidos barato, rápido, fiable y robusto, lo convierten en una tecnología muy útil para Internet. Una propuesta de investigación interesante podría estudiar la manera de integrar los sistemas de gestión P2P con los DRM[1], gracias a los cuales los usuarios podrían obtener contenidos fiables y de calidad, los proveedores podrían distribuir contenidos de forma segura y los ISP (*Internet Service Providers*) podrían adoptar diferentes papeles en la gestión de derechos digitales, para lo cual serían necesarios sistemas de gestión complejos, fundamentados en el modelo FCAPS, que además de administrar los recursos de red permitieran la gestión del negocio y de los servicios (ver capítulo 3, sección 3.4). La gestión de derechos digitales solventaría en gran parte los problemas que experimentan los usuarios de las redes P2P actuales, esto es, la ausencia de garantías sobre la autenticidad de los contenidos, descripción incompleta o catalogación errónea.

Por otro lado, la computación P2P permite y facilita la creación de redes sociales, razón por la cual sería muy importante estudiar propuestas que estimulen y definan la utilización de las redes superpuestas P2P en dicho contexto, como también en el de los sistemas colaborativos. Se podría estudiar la forma de utilizar los sistemas P2P con el objetivo de optimizar la búsqueda de información y mejorar la calidad de los servicios dentro de las redes sociales. Una vez que los *peers* generen su “lista de amigos”, podrían utilizarlas para enrutar las consultas inteligentemente, lo cual reduce el tiempo de búsqueda y el tráfico de red al minimizar el número de mensajes que circulan por ella.

Un aspecto esencial de la innovación en los servicios de la nueva generación de Internet es la interoperabilidad. Para estimular y facilitar el intercambio de información harán falta nuevos dispositivos, mejor adaptados a cada entorno de uso, pero también será esencial estandarizar un modelo genérico para la prestación de servicios soportados por la computación P2P, en el cual, se basen las soluciones de los fabricantes. La situación actual refleja una oferta de servicios cada vez mayor, sin embargo aún no se ha fijado un estándar global que los regule.

Otra propuesta de investigación interesante se relaciona con el estudio de los mecanismos de seguridad necesarios para poder realizar la fusión entre los modelos P2P y B2B[2] (acceso a las aplicaciones web, autenticación, cifrado de los mensajes XML, etc.)

[1] Digital Rights Management: Gestión de Derechos Digitales.
[2] Business-to-Business.

cuyo resultado constituye un diseño novedoso, aplicable al comercio electrónico actual, dirigido hacia las empresas y conducido por ellas mismas, eliminando a los interlocutores que dictaminan cómo y con quién deben mantener sus relaciones comerciales.

6.3 Conclusiones y contribuciones

La presencia no controlada de usuarios con las capacidades de brindar servicios genera problemas para los administradores de red. El incremento del tráfico, debido a la gran cantidad de información que intercambian los usuarios, congestiona las redes, forzando a los administradores a restringir el número de puertos que utilizan las aplicaciones P2P. Otro problema generado por este tipo de redes tiene que ver con los costos de interconexión que pagan los operadores, pues los recursos se ubican en diferentes dominios, además, los canales de descarga consumen un alto ancho de banda, a diferencia de los canales de subida, los cuales poseen un ancho de banda muy limitado. Debido a esto se hace necesario investigar sobre sistemas de gestión para redes P2P que permitan llevar a cabo tareas de administración y control sobre los recursos y servicios de este tipo de redes.

El creciente número de plataformas para el desarrollo de aplicaciones P2P genera la necesidad de llevar a cabo un estudio comparativo entre dichas plataformas con el objetivo de identificar la más apropiada para el desarrollo de aplicaciones P2P de propósito general, que sea lo suficientemente flexible y adaptable para permitir la adición de un soporte software que facilite la inclusión de un sistema para la gestión de servicios P2P. Es así como el análisis de los resultados obtenidos del estudio comparativo entre las plataformas identificadas (ver capítulo 3, sección 3.2), en el cual se describen las principales características de cada una de ellas, permite concluir que la plataforma más apropiada de acuerdo a los requerimientos, a saber, flexibilidad y adaptación, es JXTA, razón por la cual se selecciona como la plataforma sobre la cual llevar a cabo la extensión propuesta (ver capítulo 3, sección 3.3). De la misma manera, luego de examinar las características de cada uno de los modelos de gestión identificados (ver capítulo 3, sección 3.4), se concluye que el que mejor se acopla con la plataforma seleccionada es JMX, motivo por el cual se elige como la tecnología de gestión sobre la cual fundamentar el diseño de la extensión (ver capítulo 3, sección 3.5).

Es importante resaltar que la descentralización es un factor fundamental dentro de la computación P2P, que influye directamente en el diseño de este tipo de sistemas y en aspectos tales como la seguridad, escalabilidad, disponibilidad y las estructuras de datos. La viabilidad de la implementación de sistemas de gestión en entornos de redes superpuestas P2P, cuya naturaleza descentralizada dificulta la gestión de los recursos, esta dictaminada por la capacidad que deben tener los servicios P2P para generar notificaciones de manera asíncrona cada vez que ocurran los eventos que las desencadenan, es decir, debido a la naturaleza *ad-hoc* de los usuarios de una red P2P, sería muy poco práctico que el sistema de gestión revisara, uno a uno, el estado actual de cada usuario para ver si sus atributos u operaciones han cambiado.

Gracias a la contribución realizada con este trabajo de grado se pueden establecer los mecanismos que faciliten el control y la propiedad de los recursos de una red P2P. Con

los mecanismos que brinda la extensión, los servicios P2P están en la capacidad de emitir notificaciones que son capturadas por un sistema de gestión remoto. La extensión también permite modificar el valor de los atributos de los recursos e invocar operaciones sobre dichos recursos (ver capítulo 5, sección 5.1.3). Toda esta funcionalidad incrementa el rendimiento de la red P2P y la calidad de los servicios que brinda.

La validación de la extensión se llevó a cabo mediante la adición de los componentes de gestión suministrados por la extensión en una aplicación que brinda servicios P2P de mediana complejidad denominada: “Agenda colaborativa P2P” (ver anexo 4). Además se desarrolló un “Sistema de gestión de servicios P2P”, el cual es el encargado de administrar todos los recursos y servicios de red. Gracias a la adición de los componentes de gestión se logró obtener mayor control sobre los servicios P2P prestados, por ejemplo, el sistema de gestión esta en la capacidad de saber cuándo un peer crea o se cambia de grupo, envía o recibe peticiones de chat, mensajes o archivos, publica una actividad o un evento, todo lo cual es posible gracias a la generación de notificaciones de manera asíncrona (ver capítulo 5, sección 5.1.3).

CAPITULO 7

BIBLIOGRAFIA Y GLOSARIO

[1] A. Wierzbicki, N. Leibowitz, M. Ripeanu, R. Wozniak, “Cache Replacement Policies Revisited: The Case of P2P Traffic”, Polish-Japanese Institute of Information Technology, Tangium Networks, University of Chicago.
[Disponible en]: http://www.globus.org/alliance/publications/papers/cachedreplacement.pdf

[2] M. Stonebraker, P. M. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, C. Staelin and A. Yu, “Mariposa: A wide-area distributed database system”, VLDB Journal, vol. 5(1), pp. 48-62, 1996.

[3] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, Z. Xu, “Peer-to-Peer Computing”, HP Labs, Rutgers University, University of California at Santa Barbara.
[Disponible en]: http://www.hpl.hp.com/personal/Dejan_Milojicic/p2p_o.pdf

[4] Girish NS, “Introduction to BREW - a mobile handset application development platform”, An article giving a general introduction of BREW - a mobile handset application development platform like Symbian, Windows Smartphone etc., 14 Aug 2006.
[Disponible en]: http://www.codeproject.com/useritems/introduction_brew.asp
Fecha de ingreso: Agosto 2008.

[5] G. Xexéo, B. Braga, J. Nogueira D’Almeida Jr, A. Vivacqua, J. Moreira de Souza, B. Faria de Miranda, B. Kinder Almentero, R. Castilho, J. Oliveira, M. Ramirez, C. Osthoff, “Collaborative Editing of Ontologies in a Peer-to-Peer Framework”, COPPE – UFRJ, Rio de Janeiro, Brazil.
[Disponible en]: http://www.jxta.org/research/P02-GeraldoCoppeer.pdf

[6] E. Halepovic, “Performance Evaluation and Benchmarking of the JXTA Peer-To-Peer Platform”, Department of Computer Science, University of Saskatchewan, August 5, 2004.
[Disponible en]:
http://library2.usask.ca/theses/submitted/etd-08132004-120924/unrestricted/Halepovic-JXTA.pdf

[7] P. Magé, “Contribución a un sistema de negociación de recursos y servicios en el ámbito de una red superpuesta P2P”, Departamento de Sistemas, Facultad de Ingeniería Electrónica y Telecomunicaciones, Universidad del Cauca, Febrero de 2005.

[8] T. Sundsted, "The practice of peer-to-peer computing: Introduction and history", IBM Developer Works, March 1, 2001.
[Disponible en]: http://www.ibm.com/developerworks/java/library/j-p2p/
Fecha de ingreso: Abril de 2008.

[9] Wikipedia the Free Encyclopedia, “Historia de las aplicaciones P2P”, 15 August 2008.
[Disponible en]: http://es.wikipedia.org/wiki/Historia_de_las_aplicaciones_P2P
Fecha de ingreso: Mayo de 2008.

[10] P. Backx, T. Wauters, B. Dhoedt, P. Demeester, “A comparison of peer-to-peer architectures”, Broadband Communication Networks Group (IBCN), Department of Information Technology (INTEC), Ghent University, Belgium.
[Disponible en]:
http://users.ugent.be/~pbackx/files/A%20comparison%20of%20peer-to-peer%20architectures.pdf

[11] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, Z. Xu, “Peer-to-Peer Computing”, HP Labs, Rutgers University, University of California at Santa Barbara.
[Disponible en]: http://www.hpl.hp.com/personal/Dejan_Milojicic/p2p_o.pdf

[12] R. Flenner, M. Abbott, T. Boubez, F. Cohen, N. Krishnan, A. Moffet, R. Ramamurti, B. Siddiqui, F. Sommers, “Java P2P Unleashed”, Sams Publishing, September 12, 2002. ISBN: 0-672-32399-0.

[13] Skype-Guide for Network Administrators
[Disponible en]: http://www.skype.com/security/guide-for-network-admins.pdf

[14] J. Lloret, M. Granados, J. Diaz, J. Jimenez, F. Boronat, “Public Domain P2P File-sharing Networks Measurements and Modeling”.
[Disponible en]: http://personales.gan.upv.es/jlloret/pdf/icisp2006.pdf

[15] Anthill, Official Site, 2008.
[Disponible en]: http://www.cs.unibo.it/projects/anthill/
Fecha de ingreso: Octubre de 2008.

[16] PeerSim: A Peer-to-Peer Simulator, Official Site, 2008.
[Disponible en]: http://peersim.sourceforge.net/
Fecha de ingreso: Octubre de 2008.

[17] BISON - Biology-Inspired techniques for Self-Organization in dynamic Networks, Official Site, 2008.
[Disponible en]: http://www.cs.unibo.it/bison/
Fecha de ingreso: Octubre de 2008.

[18] GNUnet, Official Site, 2008.
[Disponible en]: http://gnunet.org/
Fecha de ingreso: Octubre de 2008.

[19] C. Grothoff, “An Excess-Based Economic Model for Resource Allocation in Peer-to-Peer Networks”, Department of Computer Sciences, Purdue University, West Lafayette, Indiana, USA.
[Disponible en]: http://grothoff.org/christian/ebe.pdf

[20] GNUnet – The Protocols, Official Site, 2008
[Disponible en]: http://gnunet.org/protocol.php3?xlang=English
Fecha de ingreso: Octubre de 2008.

[21] D. Kügler, “An Analysis of GNUnet and the Implications for Anonymous, Censorship-Resistant Networks”, Federal Office for Information Security, Bonn, Germany.
[Disponible en]: http://www.freehaven.net/anonbib/cache/kugler:pet2003.pdf

[22] K. Bennett, C. Grothoff, “GAP – practical anonymous networking”, Department of Computer Sciences, Purdue University, West Lafayette, Indiana, USA.
[Disponible en]: http://gnunet.org/download/aff.pdf

[23] M. Ripeanu, I. Foster, A. Iamnitchi, “Mapping the Gnutella Network: Properties of Large-Scale Peer-to-Peer Systems and Implications for System Design”, Computer Science Department, University of Chicago, Chicago, IL, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL.
[Disponible en]: http://people.cs.uchicago.edu/~matei/PAPERS/ic.pdf

[24] I. Ivkovic, “Improving Gnutella Protocol: Protocol Analysis and Research Proposals”, Software Architecture Group (SWAG), Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada.
[Disponible en]: http://www.cs.cornell.edu/people/egs/615/gnutella.pdf

[25] “The Gnutella Protocol Specification v0.4 - Document Revision 1.2”.
[Disponible en]: http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf

[26] R. S. Pressman, “Ingeniería del Software: Un enfoque práctico”, McGraw Hill. 5ª Edición.

[27] B. Traversat, M. Abdelaziz, M. Duigou, J.C. Hugly, E. Pouyoul, B. Yeager, “Project JXTA Virtual Network”, Sun Microsystems Inc, Palo Alto, CA USA, October 28, 2002.

[28] Sun Microsystems Inc., Project JXTA, Official Site, 2008.
[Disponible en]: https://jxta.dev.java.net/
Fecha de ingreso: Octubre de 2008.

[29] L. Gong, “Project JXTA: A Technology Overview”, Sun Microsystems Inc, Palo Alto, CA USA, April 25, 2001.
[Disponible en]:
http://www.cs.princeton.edu/courses/archive/fall02/cs597C/P2P/Projects/JXTA/jxta_TechOverview.pdf

[30] S. Li, “Making P2P interoperable: The JXTA story. A hands-on, working introduction to the latest P2P technology”, 01 Aug 2001.
[Disponible en]: http://www-128.ibm.com/developerworks/java/library/j-p2pint1.html
Fecha de ingreso: Octubre de 2008.

[31] P2PS, Official Site, 2008.
[Disponible en]: http://www.trianacode.org/p2ps/
Fecha de ingreso: Octubre de 2008.

[32] I. Wang, "P2PS (Peer-to-Peer Simplified)", Proceedings of 13th Annual Mardi Gras Conference - Frontiers of Grid Applications and Technologies, Schools of Physics and Computer Science, Cardiff University, Cardiff, Wales, United Kingdom, February 2005.
[Disponible en]: http://www.trianacode.org/papers/pdf/p2ps.pdf

[33] JXTA Community, "JXTA v2.0 Protocols Specification", Sun Microsystems Inc, Palo Alto, CA USA, 2007.
[Disponible en]: https://jxta-spec.dev.java.net/JXTAProtocols.pdf

[34] R. Paredes, D. Rueda, "Definición de perfiles UML para servicios telemáticos orientados a una arquitectura P2P", Universidad del Cauca, Facultad de Ingeniería Electrónica y Telecomunicaciones, Departamento de Telemática, Junio de 2008.

[35] RAD Group, "Gestión de Redes", RAD Data Communications, Tel Aviv, Israel.
[Disponible en]: http://www2.rad.com/catalog/spanish/images/radch7s.pdf

[36] A. Guerrero, "Especificación del comportamiento de gestión de red mediante ontologías", Universidad Politécnica de Madrid, Departamento de Ingeniería de Sistemas Telemáticos, Madrid, 2007.
[Disponible en]: http://oa.upm.es/909/01/ANTONIO_GUERRERO_CASTELEIRO.pdf

[37] International Telecommunication Union – Telecommunication Standardization Sector (ITU-T), "Overview of TMN Recommendations", Recomendación M.3000, Febrero de 2000.

[38] Sun Microsystems Inc., "Java Management Extensions (JMX) Specification, version 1.4 Final Release", Santa Clara, California, U.S.A, November 9, 2006.
[Disponible en]:
http://java.sun.com/javase/6/docs/technotes/guides/jmx/JMX_1_4_specification.pdf

[39] J. O'Sullivan, "Towards a Precise Understanding of Service Properties", Faculty of Information Technology, Queensland University of Technology, 2006.
[Disponible en]: http://eprints.qut.edu.au/16503/1/01front.pdf

[40] M. Santillán, "Desarrollo de una herramienta de gestión remota de pasarelas de servicios domésticas", Universidad Politécnica de Madrid, Departamento de Ingeniería de Sistemas Telemáticos, Noviembre de 2004.

[41] R. Paredes, D. Rueda, Trabajo de Grado: "Definición de perfiles UML para servicios telemáticos orientados a una arquitectura P2P", Universidad del Cauca, Facultad de Ingeniería Electrónica y Telecomunicaciones, Departamento de Telemática, Junio de 2008.