UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERíA DE TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA

DISEÑO Y DESARROLLO DE UN SITEMA DISTRIBUIDO DE BÚSQUEDA DE AUDIO Y STREAMING BASADO EN P2P

Autor: Joaquín Melgarejo Ricca
Tutor: Daniel Díaz Sánchez

21 de junio de 2015

Agradecimientos

A Dani, por su ayuda, sus buenos consejos y su paciencia para explicarme las cosas despacito que hacen de él un mentor espléndido. Gracias también por esas conversaciones, consejos y punto de vista ajenas a este proyecto.

A todos los amigos que llevo de esta carrera. Me considero una persona con suerte por haber llegado a conocer a gente tan extraordinariamente humana y tan buena gente. Gracias por haber compartido tantos años de duro trabajo compaginado con tantas risas. Quisiera poder nombraros a todos pero solo me dejan una página de agradecimientos. No puedo guardar un mejor recuerdo de mi paso por esta universidad gracias a todos vosotros.

A mi padre, por ser un ejemplo inspirador de trabajo duro y esfuerzo, siempre preocupado por el bienestar de los suyos. A mi madre por ser un ejemplo igualmente, así como por su paciencia intentando entender la diferencia entre una asignatura de circuitos y otra. A mis hermanas "las pringuis", por saber hacer ver a su hermano mayor que también puede equivocarse, aunque suele ser raro. A mi novia, por encarnar a la perfección la frase "Detras de todo gran hombre hay una gran mujer", gracias por seguirme a todos lados con esa sonrisa.

Por último, me gustaría mencionar brevemente a Javier Rodriguez, Enrique Fazio y Jaime Borondo, así como a aquellos que mostraron su disponibilidad para ayudarme con las pruebas del proyecto. Aún os debo un café.

A todos, gracias.

Resumen

Abstract

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

Como unas 5 páginas con las siguientes secciones:

1.1. Motivación del proyecto

Como 3 páginas para justificar el proyecto.

1.2. Objetivos

Como 1 página detallando los objetivos que se plantean inicialmente en el proyecto.

1.3. Contenido de la memoria

Como 1 página indicando el contenido de los siguientes capítulos y apéndices.

Capítulo 2

Estado del arte

2.1. Redes Peer-to-Peer

Puede definirse una red peer-to-peer como un sistema descentralizado y distribuido. Un conjunto de dispositivos o máquinas, interconectados mediante un canal de transporte de datos forman una red auto-organizativa con el fin de compatir recursosa través de Internet.

Este tipo de redes ofrecen una serie de características como un encaminamiento o enrutado muy robusto, una elevada eficiencia en la búsqueda de datos o ficheros, anonimato, escalabilidad, tolerancia frente a fallos así como correcciones de los mismos junto con una capacidad de selección de nodos adyacentes.

Dentro de este tipo de redes, no existe la noción de clientes o servidores de manera individual, sino que los nodos reunen esta serie de funciones y tareas, para actuar como clientes y servidores de otros nodos equitativos dentro de la red.

Uno de los retos más interesantes que plantean este tipo de sistemas es un mantenimiento eficaz y a tiempo real de la topología, ya que la libertad conexión y desconexión de la red que se les otorgan a los nodos, dificulta sobremanera esta tarea. Ningún nodo conoce la topología exacta de red totalmente distribuida.

El grado de autonomía que se les confiere a los nodos, permite a éstos decidir que tipos de servicios desean ofrecer a los demás nodos. Estos nodos trabajan sobre direcciones de red temporales, lo que les obliga a implementar

servicios de disponibilidad total, partiendo de la desventaja de que sus direcciones de red hayan podido cambiar en cualquier momento. Es por ello que la escalabilidad y redundancia adquiere una importancia capital frente a otro tipo de sistemas más tradicionales (ya sean centralizados o distribuidos).

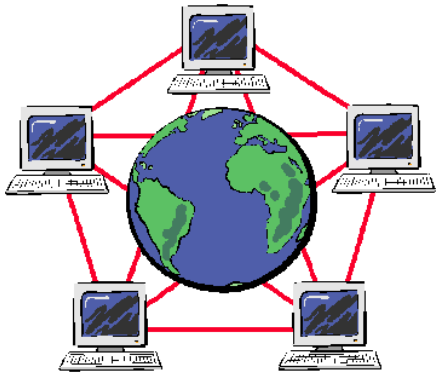

Figura 2.1: Representación de una red P2P sobre Internet

No existe un grado de confianza total desde un nodo hacia otro debido precisamente a su autonomía. Pueden darse ataques dentro de redes P2P con bastante facilidad. La manera más sencilla es conectarse a la red con una identidad múltiple (también llamados *sybils*) y realizar ataques de *Man-in-the-Middle* sobre los mensajes de enrutamiendo, así como sobre los contenidos compartidos dentro de la red. Usando su identidad múltiple, pueden volverse invisibles frente a nodos *"principiantes"*.

Existen tres principios básicos que definen a un sistema totalmente descentralizado diseñado con nodos sobre un sistema auto-mantenido:

• **Principio de distribución recursos:** todos los sistemas P2P incluyen un aspecto de cooperación y distribución de recursos, donde dichos recursos pueden tratarse de recursos físicos como memoria o ancho de banda, así como recursos lógicos, tales como servicios o información. Gracias a esta puesta en conjunto, se pueden lograr objetivos que de otra forma en un sistema de computación *stand-alone* no podría conseguirse. Como es lógico, es creciente en proporción al número de nodos conectados.

• **Principio de descentralización:** consecuencia inmediata de la puesta en conjunto de los recursos. La mayor parte del sistema (sino todo) no operan de manera centralizada. Una faceta interesante de cara a evitar o mi-

nimizar los daños que pueden surgir debido a fallos individuales de los nodos.

• **Principio de auto-mantenimiento:** en el momento que el sistema adquiere el grado de descentralización total, deja de existir un nodo de coordinación de actividades o una base de datos central para guardar la información del sistema. Los nodos deben desarrollar un sistema de organización independiente que se base en la información que puedan adquirir de sus alrededores y vecinos alcanzables (alcanzables en el sentido de la disponibilidad). El comportamiento general es el que se obtiene de estas conductas grupales a nivel local.

2.1.1. Estructura de las redes P2P

Dentro de una perspectiva más general, pueden diferenciarse dos tipos básicos de estructuración de estas redes P2P. Estas redes incluyen a todos los nodos participantes como integrantes de su sistema. Cada uno de estos nodos guarda información acerca de otros nodos conocidos, que se emplea para el enrutamiento de mensajes y paquetes a través de la red, funcionando de manera similar a las *forwarding tables* que pueden encontrarse dentro de los routers.

Existen una serie de enlaces o conexiones entre dos nodos que se "conocen" dentro de la red, como por ejemplo si un nodo conoce la localización de otro dentro de la red, entonces existe una conexión directa entre éstos. La manera en la que se relacionan estos nodos es lo que nos permite clasificar los sistemas en *estructurados* o *desestructurados*.

Sistemas desestructurados. Se dice que un sistema P2P es un sistema desestructurado cuando los enlaces o conexiones entre nodos se realizan de forma arbitraria. Un nodo que desee formar parte de la red, puede copiar de manera temporal los enlaces de otro nodo conocido antes de empezar a formar los suyos propios. Usando mecanismos de inundación de red, *flooding*, se puede reconstruir la topología de un área limitada, gracias a que cuando un nodo recive una petición de esta índole, envía una lista de toda la información disponible concerniente a la petición al nodo emisor.

A pesar de que las técnicas de inundación en la red sean muy eficaces a la hora de recuperar información replicada por toda la red, son muy poco prácticos para hallar contenidos específicos o únicos. Esta solución no permite una gran escalabilidad, ya que la carga de recursos de cada nodo se

incremente de manera lineal con las peticiones y el tamaño del sistema.

Uno de los puntos débiles de los sistemas desesctruturados es la sobrecarga, al no escalar conforme a las necesidades de la red, los nodos pronto se ven desbordados con el manejo de peticiones a gran escala.

Sistemas estructurados. En las redes P2P que se dicen estructuradas, se definen de manera estricta y concisa los enlaces inter-nodos, así como la localización de los recursos compartidos. Dentro de este tipo de sistemas, la mayoría de ellos optan por la implementación de una tabla de hash distribuida (**DHT**) usando diferentes estructuras de datos para satisfacer las necesidades de la red.

Si analizaramos una DHT, podríamos descomponerla en diferentes componentes principales. Basado en un sistema de clave espacial, siendo estas de una longitud considerable, se asigna una única clave (generalmente aleatoria) a cada nodo participante, dentro del espacio de claves que puede llegar a tener la DHT.

Un ejemplo sería emplear claves de 128 bits de longitud que generan un espacio de claves total de $2^{128} - 1$ claves. Cada nodo registra una parte de la topología del sistema, de manera que se hacen uso (de nuevo) de tablas de enrutado conteniendo información acerca de los nodos adyacentes o vecinos. Puestos en común, estos enlaces forman la totalidad de la red. Gracias a esta información contenida dentro de cada nodo, el sistema de encaminamiento pasa por varios nodos hasta llegar al nodo de destino. A esta técnica se la conoce como *key-based routing.*

Un elemento compartido del sistema se mapea con este mismo sistema, asignando una clave única que lo identifica dentro del sistema, usando un valor del elemento como puede ser su nombre o el propio fichero en sí, para obtener un cifrado hash de longitud de clave empleada. De esta manera, las DHTs forman una estructura de búsqueda determinista. A pesar de esta técnica general, la manera en la que los contenidos son asignados a los nodos participanetes es distinto dentro de las distintas variedades de DHT. Este mecanismo de asignación de clave lo único que permite es identificar cada elemento compartido de la red, así como los nodos que lo contienen.

El sistema de encaminamiento facilita la búsqueda pero también la publicación de nuevos contenidos. Es una práctica general, realizar réplicas de todos los contenidos de la red, de tal manera que éstos no se pie-

den debido a desconexiones puntuales de los nodos que albergan el contenido.

Teóricamente, los sistemas usando DHTs garantizan que la búsuqueda y recuperación de un contenido puede realizarse de media en $O(\log N)$veces el número de saltos en la red, siendo N el número de nodos en el sistema. Pueden diferir el encaminamiento de los enlaces inter-nodos a un nivel físico y a nivel de topología, pudiendo afectar de manera considerable la latencia en sistemas basados en DHTs y disminuyendo el rendimiento de las aplicaciones que pudiesen trabajar por encima.

2.1.2. Sistemas semi descentralizados

El modelo de sistema descentralizado, o semi estructurado, dentro de las redes P2P surge ante la necesida de hallar un compromiso entre las ventajas de ambos sistemas opuestos, centralizado y descentralizado, siendo éste el modelo más extendido dentro de las arquitecturas P2P.

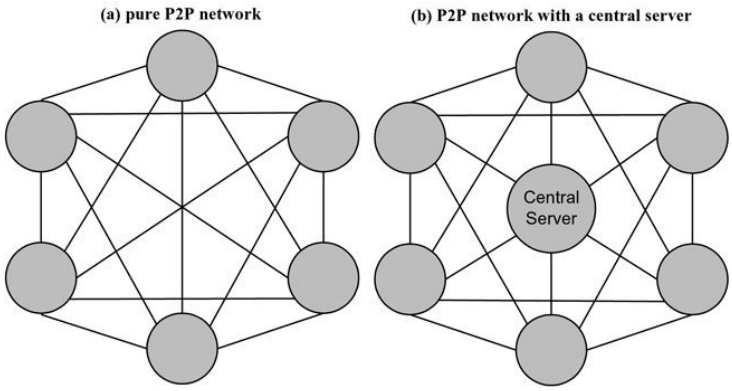

Figura 2.2: Comparativa Sistema distribuido puro vs Sistema semi descentralizado

Este modelo de sistema cuenta con uno o más nodos especializados, llamados Central o Master, cuyas funciones son las mismas que las que desempeñaría un servidor en un sistema centralizado, y cuyas tareas pasan por encaminamiento de tráfico y administración de recursos sin compartir ninguno de éstos. Estos nodos no albergan información de los demás nodos, ni información alguna, funcionan ofreciendo servicios de enrutamiento y control.

La ventaja principal de implementar un sistema de nodos centrales es la distribución de las conexiones, pudiendolas limitar a un número manejable

para éstos. Cada nodo Central se conecta con los demás nodos centrales para obtener un mapa completo de la red.

Esta estructura semi descentralizado permite al sistema tener un alto grado de escalabilidad, reduciendo los saltos en el encaminamiento de paquetes dentro de la red y reduciendo la congestión de tráfico que pueda darse en la red. Como resultado de ser un compromiso entre centralizado y descentralizado, estos nodos centrales no son críticos para la red. En caso de que este sistema central falle, la red puede seguir funcionando ya que los demás nodos tienen enlaces entre ellos permitiendo el intercambio de información.

2.1.3. Aplicaciones y ejemplos de sistemas semi descentralizados

Kademlia

Kademlia es sistema que trabaja implementando una DHT sobre P2P. Funciona como un protocolo de comunicaciones para redes descentralizadas y estructuradas en P2P.

Los nodos contienen un identificador único de 160 bits de longitud, generado a partir de una función hash con la dirección IP del nodo. Este valor le permite ser identificado dentro de la red.

Siendo uno de los sistemas más populares a día de hoy, Kademlia surgió como un sistema que ofrecía una serie de características que otros sistemas contemporáneos no contemplaban en su momento. Para poder evitar saturar la red, Kademlia cuenta con un sistema adaptativo que minimiza el número de mensajes de señalización y configuración que necesita el sistema en base al número de nodos conectados.

La implementación de los nodos también da un salto de calidad, se les dota de más carga lógica y flexibilidad para poder encaminar las peticiones a través de rutas con baja latencia. Siendo éstas peticiones asíncronas y paralelas, se evitan en gran parte los fallos debidos a *timeouts* por culpa de nodos fallidos.

Los nodos en Kademlia almacenan la información de otros nodos en objetos designados como *buckets* de un espacio limitado de ‘kñodos que designa su capacidad. Existe un bucket de capacidad k para cada $0 \leq i \leq 160$, existiendo por tanto 160 buckets distintos de capacidad k. La

capacidad habitual suele ser de k = 20. Por lo tanto, el bucket con índice i contiene el identificador de los nodos cuyos distancia se comprende entre $2^i \leq d(nodo1, nodoN) \leq 2^{i+1}$. Estos buckets lo que permiten es cubrir el espacio de claves de 160 bits posible.

Los nodos con distancias cortas son clasificados en los buckets de menor índice. Su organización se basa en la información que tienen sobre la distancia de red existente entre los otros nodos y él mismo. Gracias a esta información almacenada en los buckets, los nodos son capaces de crear árboles binarios a nivel interno para funciones de enrutamiento.

Un ejemplo de esto sería, para el bucket i, siendo $i \geq 0$, se calcula la distancia entre el nodo con la métrica XOR. Una vez calculada, se clasifica al nodo distante en función del resultado de la operación formando un árbol parecido al siguiente:

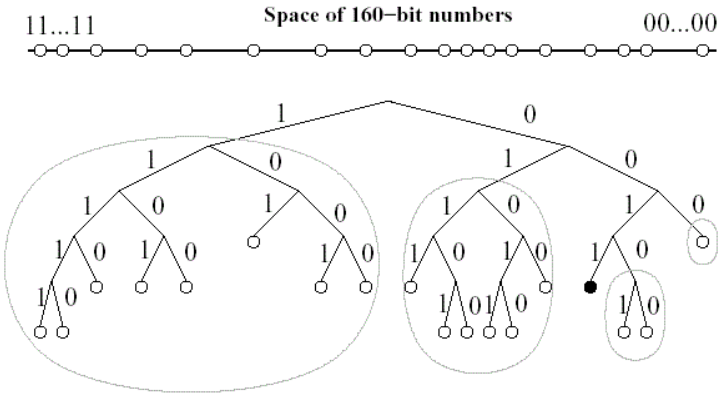

Figura 2.3: El elemento en negro representa la localización del nodo 0011... en el árbol. Las secciones redondeadas muestran los árboles secundarios a los que el nodo 0011... debería tener contacto.

Entre sus propiedades principales, Kademlia usa una aproximácion básica de muchas DHTs. El tamaño de sus claves son de 160 bits de longitud, usando algoritmos de codificación como SHA-1. Las identidades de cada uno de los nodos son únicas, ademas de ser también el resultado de un hash de 160 bits de longitud. El uso de parejas clave-valor permite a sus nodos tener información acerca de nodos adyacentes, al igual que permite un sistema de búsqueda por identificación de clave única para localizar los servidores centrales.

Como se ha comentado antes, el uso de métricas XOR permite a Kademlia hallar las distancias entre puntos dentro del espacio de claves. XOR es simétrico, permitiendo a los usuarios de estas redes recibir y realizar peticiones de enrutamiento de manera precisa gracias a sus tablas de enrutamiento.Esto permite a otros sistemas como Chord[1] obtener información útil acerca del encaminamiento gracias a las peticiones.

Estas métricas consisten en realizar la operación lógica XOR con los identificadores de cada nodo. Esto es, si x e y son los identificadores de dos nodos, entonces $d(x, y) = x \oplus y$. Esta métrica cuenta con las siguientes propiedades:

•$d(x, x) = 0$.

•$d(x, y) > 0$, siempre que $x \neq y$.

•$\forall x, y : d(x, y) = d(y, x)$, siendo esta distancia simétrica.

•$d(x, y) + d(y, z) \geq d(x, z)$, propiedad de triangulación.

•Dadas una distancia Δ y un identificador x, existe otro identificador tal que $d(x, y) = \Delta$. Esta propiedad asegura que las peticiones de búsqueda para un identificador convergen en la misma ruta, independientemente del nodo de origen.

BitTorrent

El protocolo BitTorrent, al igual que muchos otros protocolos de P2P como eD2K,está basado en una implementación de Kademlia para crear una red de distribución de ficheros a gran escala. Una de las ventajas que representa frente a otros protocolos es que trabaja sobre HTTP, de manera que cuando varios nodos descargan un mismo contenido de manera concurrente, al mismo tiempo se distribuyen el contenido entre ellos, permitiendo que el fichero pueda ser descargado simultaneamente entre un gran número de nodos con un incremento mínimo de carga.

El proceso de distribución de un fichero pasa por la implementación de un tracker o rastreador, que permite albergar la información necesaria para

[1]Protocolo y algoritmo para redes P2P sobre DHTs, almacenando pares de clave-valor.

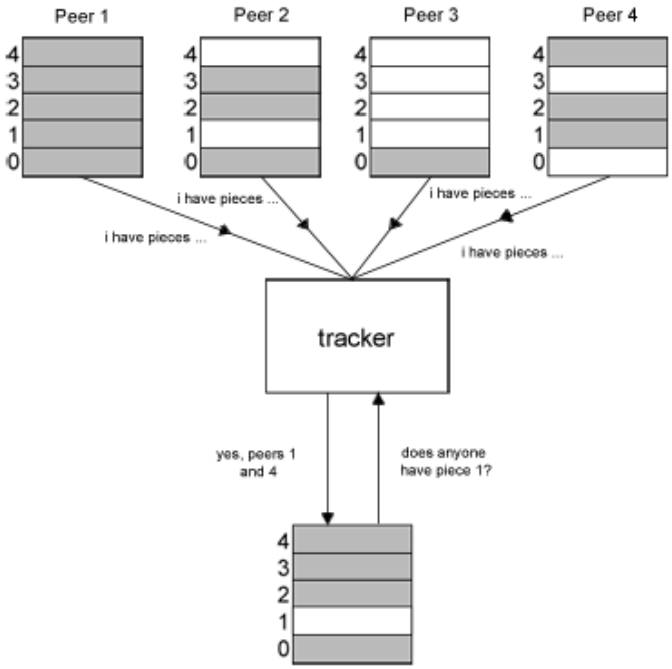

Figura 2.4: Funcionamiento de un tracker en BitTorrent

que otros nodos de la red puedan conectarse a la red e iniciar descargas, sirviendo como único punto de encuentro dentro del sistema. Luego con un servidor web cualquiera, se asocian los archivos de extensión ".torrent" con la aplicación BitTorrent correspondiente para generar los archivos de metadatos a partir de las URLs que proporciona el tracker. Una vez el servidor contenga estos archivos de metadatos, se asocian a una pagina web. Por último, lo único que hace falta es un nodo contenedor del fichero original para poder ponerlo en común.

Los ficheros de metadatos contienen la información necesaria para que cualquier cliente BitTorrent pueda iniciar una descarga del fichero asociado desde cualquier parte de Internet. Esta información contendrá la URL del tracker, así como la información de los trozos o *chunks* del fichero original para poder descargarlos por separado y poder juntarlos en el nodo de destino.

Esta técnica de particionado de ficheros simplifica sobremanera la corrección de fallos y el auto-mantenimiento de la red. Para un nodo cualquiera de la red, es mucho más facil realizar múltiples descargas de ficheros muy pequeños en vez de una única descarga de un fichero más grande, ya es mas

facil realizar una correción de fallos sobre un segmento del fichero que sobre todo el fichero. Si uno de los *chunks* resultase corrupto debido al proceso de descarga, el nodo de destino únicamente tendría que realizar una nueva petición de descarga para ese segmento concreto, en vez de realizar la descarga de todos los segmentos de nuevo. Así mismo, cuando un fichero se encuentra distrbuido por la red, los nodos que albergan el fichero pueden compartir solo uno o varios segmentos del fichero, repartiendo la carga de distribución entre los nodos contenedores del fichero y mejorando la velocidad de descarga.

TomP2P

TomP2P es una implementación avanzada de una DHT que almacena múltiples valores por clave. Cada nodo tiene su propia tabla donde almacena estos valores. Cada uno de los valores de esta clave pueden ser susceptibles a modificación, petición y actualización con claves secundarias.

Se trata de una implementación de un sistema P2P desarrollado en Java, usando un Java NIO como framework de comunicación para manejar conexiones concurrentes.

Los nodos, de manera general y al igual que en muchos otros sistemas basados en P2P, forman parte de una red overly abstracta. esto permite indexar y realizar búsquedas de manera mas sencilla ya que otorga a los nodos un grado de libertad para trabajar independientemente de la red física. Esto resulta muy util en el supuesto de que los nodos usen direcciones IP no estáticas o temporales.

Como la mayoría de los sistemas implementados a partir de Kademlia, su API proporciona una serie de recursos y librerías para formas redes que sean tolerante a fallos, heterogéneas, distribuidas a nivel de carga y escalables.

2.2. Sistemas Distribuidos

Los sistemas Peer-to-peer buscan dar soluciones eficaces a la distribución de servicios y a las aplicaciones que usen datos y recursos informáticos disponibles dentro del ámbito de los ordenadores personales que se encuentran cada vez más presentes en Internet. Esto hace que aumente el interés en vista de que la diferencia de rendimiento que se

puede observar entre ordenadores personales y servidores cada vez es menor mientras que por otra parte las redes con conexión de banda ancha proliferan.

Las conexiones entre ordenadores dentro de una red que se manejan por un gran número de usuarios distintos y pertenecientes a distintas organizaciones son recursos volátiles; los propietatios no garantizan su conexión permanente o bien que se encuentren libre de fallos.

La disponibilidad de los procesos y ordenadores que forman parte de las redes P2P es impredecible. Los servicios no pueden basarse en un acceso garantizado a los recursos individuales del sistema, aunque sin embargo puede diseñarse para tener en cuenta una probabilidad de fallo en la que se tenga acceso a una copia del recurso original. Es importante entender que esta debilidad de los sistemas P2P pueden convertise en uno de sus puntos fuertes mediante la implementación de un sistema de replica para evitar fallos puntuales de la red.

2.2.1. Sistemas distribuidos: Napster

La primera aplicación en la que la demanda de almacenamiento y recuperación de información como servicio a un nivel de escalabilidad global fue la descarga de archivos de música digital. Tanto la necesidad como la viabilidad de una solución P2P fué demostrada por primera vez por el sistema Napster (OpenNap, 2001), nacido para compartir archivos y que proporcionaría un medio para que los usuarios pusieran en común archivos. Napster se hizo muy popular para intercambio de música poco después de su lanzamiento en 1999.

La arquitectura centralizada de Napster incluye índices, pero los usuarios los que suministran los ficheros que se almacenan y se accede a ellas a través de sus ordenadores personales. Por lo tanto, la motivación de Napster y la clave de su éxito fue la puesta a disposición de un gran conjunto de archivos a través de Internet, ampliamente distribuidos entre sus usuarios.

Napster demostró la factibilidad de construir un servicio útil de gran escala que depende casi completamente de datos y equipos de propiedad de los usuarios de Internet. Para evitar inundar los recursos informáticos de los usuarios individuales y sus conexiones de red, Napster tuvo en cuenta la red local, el número de saltos entre el cliente y el servidor y al asignar un servidor a un cliente que pide una canción. Este simple mecanismo de distribución carga permitió a este servicio escalar

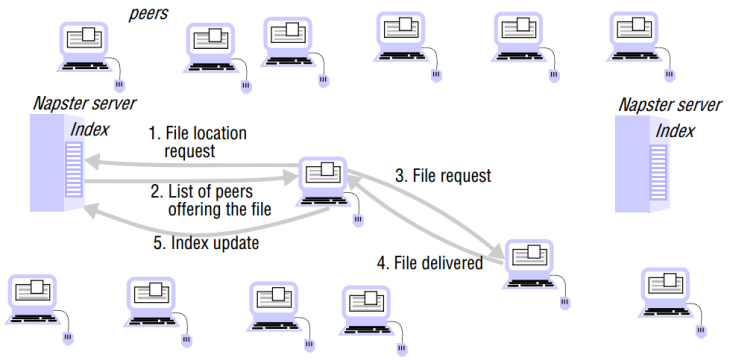

Figura 2.5: Napster: Sistema de distribución centralizado

bajo demanda para satisfacer las necesidades de un gran número de usuarios.

Limitaciones: Napster utilizaba un índice unificado(replicado) de todos los archivos de música. Para la aplicación en cuestión, la necesidad de coherencia entre las réplicas no era elevada ,que aunque en este caso no supondría un obstaculo en su rendimiento, para muchas otras aplicaciones si podría constituir una limitación. A menos que el camino de acceso a los objetos de datos sea distribuido, la retribución de ficheros es probable a que converja en un cuello de botella.

2.2.2. P2P Middleware

Un problema clave en el diseño de aplicaciones P2P es proporcionar un mecanismo que permita a los nodos para acceder a los recursos de la red de una manera rápida y segura dondequiera que se encuentren en a lo largo de toda la red. Los sistemas middleware de P2P están diseñados específicamente para satisfacer la necesidad de colocación automática y la posterior ubicación de los objetos distribuidos.

Requerimientos funcionales: La función del middleware en P2P es la de simplificar la construcción de los servicios que se llevan a cabo a través de varios nodos en una red de amplia distribución . Para lograr esto, debe permitir a los nodos poder localizar y comunicarse con cualquier otro nodo de recursos puestos a disposición de un servicio, aún cuando los recursos están distribuidos.

Otros requisitos importantes incluyen la capacidad de agregar nuevos

recursos y a quitar a voluntad, así como agregar nodos al servicio y eliminarlos. Al igual que otros middleware, el middleware en P2P debe ofrecer una interfaz de programación sencilla a los programadores de aplicaciones independientemente de los tipos de recursos distribuidos que manipula la aplicación.

A pesar de los requerimientos antes mencionados, el middleware de P2P debe buscar una solución a los siguientes problemas:

•**Escalabilidad:** Uno de los objetivos de aplicaciones de P2P es aprovechar los recursos de hardware de un gran número de nodos conectados entre sí. El middleware en P2P debe ser diseñado para ofrecer soporte a las aplicaciones que acceda a un gran número de recursos y nodos.

•**Balanceo de carga:** El rendimiento de cualquier sistema diseñado para explotar una gran cantidad de equipos depende de una distribución equilibrada de la carga de trabajo en cada uno de ellos. En el caso de P2P, esto se logrará mediante una distribución aleatoria de recursos, así como el uso de réplicas de recursos importantes para la red.

•**Optimización para interacciones locales:** La "distancia en red" entre los nodos que interactúan entre sí tiene un impacto sustancial en la latencia de las interacciones individuales, tales como las peticiones de los clientes para el acceso a los recursos. La carga de tráfico en red también se ve afectado por esto. El middleware debe tender a colocar los recursos "cercanos" a los nodos para poder acceder a ellos.

•**Seguridad en un ambiente de confianza heterogénea:** En sistemas de escala mundial con participantes de diversos permisos de propiedad, la confianza debe ser establecida para el uso de mecanismos de autenticación y codificación, asegurando la integridad y privacidad de la información.

•**Anonimato:** El anonimato de los titulares y destinatarios de los datos es una preocupación legítima en muchas situaciones donde pueden encontrarse problemas de censura. Un requisito es que los nodos que contienen los datos deberían ser capaces de negar su responsabilidad por posesión y distribución. El uso de un gran número de nodos en sistemas P2P puede ser útil en el logro de estas propiedades.

La solución de mejor diseño una capa de middleware a escala mundial en sistemas P2P es por lo tanto un problema de difícil solución. Los requisitos

para la escalabilidad y disponibilidad hacen imposible mantener una base de datos en todos los nodos dando las ubicaciones de todos los recursos de la red. El conocimiento de las posiciones de los objetos debe ser particionado y distribuido a lo largo de la red . Cada uno de los nodos es responsable de mantener una parte de ese conocimiento conjunto, detallando la ubicación de los nodos y los objetos. Un alto grado de repetición de este conocimiento conjunto es necesario para garantizar la fiabilidad ante la volatilidad de los nodos y su conectividad de red intermitente.

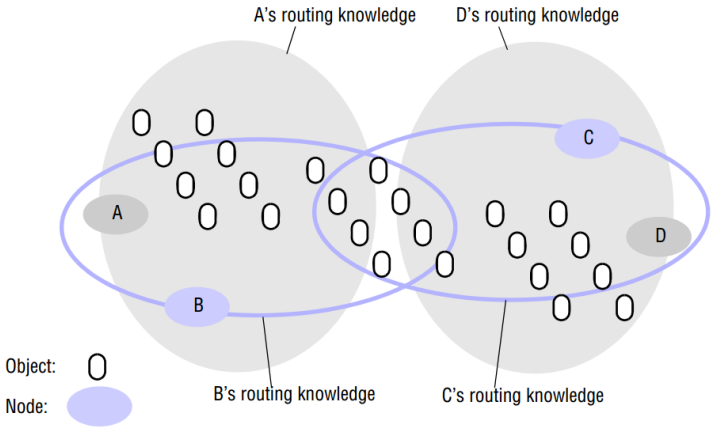

Figura 2.6: Distribución de información en una red Overlay

2.2.3. Redes Overlay

El desarrollo de un sistema de middleware que cumple con los requisitos funcionales y no funcionales, descritos en la sección anterior, es un área activa de investigación, y varios sistemas de middleware han surgido ya. En los sistemas P2P distribuidos existe un algoritmo de superposición de enrutamiento toma responsabilidad de localizar los nodos y objetos. El nombre denota el hecho de que el middleware tiene la forma de una capa que se encarga de enrutar las solicitudes de cualquier nodo a otro nodo que contiene el objeto al que se dirija la petición.

Se le denomina a este tipo de redes como una red Overlay ya que implementa un mecanismo de enrutamiento en la capa de aplicación que es muy distinta de cualquier otra mecanismos de ruta implementados a nivel de red, tales como enrutamiento IP. Estas redes permiten interconectar redes de distintos protocolos, permitiendo una topología más adaptativa en

función de los topologías de cada grupo de nodos conectado a la red.

Los nodos de interconexión lo que permiten es actuar como pasarela o *gateway* entre los distintos protocolos. A éstos nodos de interconexión se les puede clasificar como super-peers.

Gracias a esta pasarela, se pueden realizar búsquedas en un dominio local o propio, así como realizar peticiones en otros dominios o segmentos de la red con un funcionamiento protocolario distinto. Los super-peers se encargan de tratar las peticiones procedentes de otros dominios o segmentos y adaptarlos al sistema que tiene por debajo jerárquicamente. Como es comprensible, estos nodos pueden ser un cuello de botella si el volumen de peticiones es muy grande.

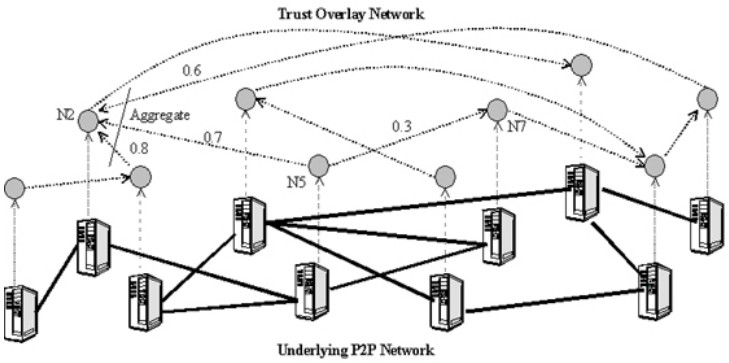

Figura 2.7: La red Overlay trabaja a un nivel de abstracción superior para poder conectar todos los dominios de una red P2P

La red Overlay garantiza que cualquier nodo puede acceder a cualquier objeto por cada solicitud de enrutamiento a través de una secuencia de nodos, haciendo uso de la información en cada uno de ellos para encontrar el objeto de destino. Los sistemas P2P suelen almacenar varias réplicas de objetos con el fin de garantizar la disponibilidad . En este caso, la red Overlay mantiene conocimiento de la ubicación de todas las réplicas y proporciona la solución mas cercana a las peticiones que se realizan a tiempo real.

Las funciones principales de la red Overlay son:

•**Enrutamiento de peticiones sobre objetos:** Un nodo que desee realizar una operación sobre un objeto o fichero realiza una primero una

petición haciendo uso del identificador de dicho objecto a la red Overlay, quien redirecciona la petición a un nodo contenedor de una replica del objeto en cuestión.

•**Inserción de objetos:** Un nodo que desee crear un nuevo recurso disponible para un servicio P2P realiza la operación hash necesaria para dotarlo de un identificador único y anuncia la operación a la red Overlay. La red se encargará de que este nuevo recurso sea alcanzable para cualquier otro nodo.

•**Eliminación de objetos:** Un nodo que desee eliminar un recurso a nivel individual debe anuncialo a la red Overlay.

•**Creación y eliminación de nodos:** Los nodos tienen libertad para unirse o abandonar el servicio. Cuando un nodo se incorpora al servicio, la red Overlay se acarga de asignarle responsabilidades de cara a otros nodos. Cuando un nodo abandona el sistema (como resultado de un fallo o de manera voluntaria), las responsabilidades de éste se reparten entre los nodos activos.

Capítulo 3

Diseño

3.1. Requisitos

Antes de plantear el diseño de la red, primero debemos definir los requisitos necesarios para el desarrollo del sistema. Tomando como ejemplo otros sistemas basados en implementaciones de Kademlia(*Consultar el apartado 2.1.3*), un sistema semi distribuido consta de diferentes entidades.

Las características de los sistemas semi distribuidos(*Apartado 2.1.2*) combina las ventajas de las estructuras puramente Peer-to-Peer con ventajas que presentan las arquitecturas Servidor-cliente. De éste segundo tipo de arquitectura distinguimos dos entidades:

•**Nodo cliente** que permita a los usuarios conectarse a la red y acceder a los servicios de la misma.

•**Nodo de control o Master** que actue asuma las funciones de reconocimiento de la topología, así como la clasificación de contenidos distribuidos por la red.

Existe, por tanto, la necesidad de desarrollar éste segundo tipo de entidades, los nodos Master, dentro de la estructura para garantizar los servicios que desean implementar los nodos clientes.

Requisitos del cliente

En cuanto a las necesidades que presentaba el sistema, los nodos deberían presentar las siguientes funcionalidades y servicios:

•**Capacidad de identificación y clasificación de archivos mediante un sistema de reconocimiento en línea.** El problema que presenta el hecho de que cualquier nodo conectado a la red sea capaz de compartir ficheros nuevos plantea un problema de normalización de los mismos.

Dado que los ficheros que se van a distribuir por el sistema son ficheros de audio, únicamente se necesita identificar el contenido del fichero una vez. Dado que los ficheros de música no estan sujetos a actualizaciones de ningún tipo, no existe la necesidad de realizar comprobaciones periódicas sobre el contenido del mismo. Esto simplifica en parte el proceso de normalización de las canciones. Sin embargo, sí es cierto que existen distintas versiones para una misma canción, ya sea que la duración sea distinta, el artista sea diferente, etc. Por ello es importante poder identificar cada una de estas versiones de una misma canción, asignándoles un identificador distinto. Sin esto, el sistema podría llegar a compartir dos o más ficheros bajo un mismo identificador, que al juntar distintos *chunks* de distintos archivos puede dar como resultado en un archivo corrupto.

El sistema de clasificación en linea se explica en el apartado 4.1.1.

En vista de esta necesidad, se plantean dos maneras de implementar una solución:

- Realizar la identificación y clasificación de los ficheros a un nivel de nodo cliente, haciendo uso del servicio de reconocimiento en red (*Apartado 4.1.1*) y después compartirlo directamente en la red, previa notificación al Master.

- Transmitir el fichero de audio de manera directa al nodo de control para que éste realice la clasificación con la herramienta de reconocimiento en red. Como es comprensible, este segundo planteamiento es poco viable debido al cuello de botella que presenta el envio de archivos de este tamaño al nodo de control y más sabiendo que la potencial escalabilidad de la red puede provocar una congestión en la cola de peticiones del nodo de control.

•**Capacidad de inserción de nuevos ficheros:** Uno de los objetivos a lograr en este proyecto es poder crear una biblioteca musical a partir de la contribución de cada uno de los usuarios conectados a la red a través de su nodo/s. Para ello, debe ser capaz de poner en común su propia música

si lo desea, a través del sistema de clasificación de ficheros comentado anteriormente.

•**Reproducción de contenidos en Streaming:** Como parte de estudio dentro de este proyecto, es interesante probar diferentes servicios que pueden aportar los nodos conectados a la red. Un servicio de reproducción de música en Streaming plantea por tanto, una serie de condicionantes adicionales a los anteriormente comentados.

•**Descarga de ficheros existentes:** Cada usuario debe ser capaz de descargar los archivos existentes dentro de la red. Como ya sabemos de la teoría, cualquier nodo sólo conoce una parte de la topología y contenidos existentes dentro de la red, representando una limitación para cualquier usuario.

Requisitos del Master

Sin la implementación de un nodo de control, o nodo Máster, este sistema por sí solo plantea las siguientes deficiencias:

- Los nodos no son capaces de conocer la topología exacta de la red, el número de usuarios conectados ni la disponibilidad de los contenidos.

- Los nodos no son capaces de conocer la totalidad de los nodos compartidos, ni el valor de los contenidos.

A menos que se implementase un protocolo parecido a BitTorrent, donde se usasen elementos como Trackers y servidores web para anunciar e informar de la disponibilidad de los contenidos al resto de los usuarios, los nodos del sistema no son capaces de conocer el valor de los contenidos compartidos por la red.

La solución planteada en este proyecto consiste en estructurar los contenidos de la red, dentro de un árbol dinámico contenido en un nodo de control, donde los nodos pertenecientes a la red puedan consultar la disponibilidad de los contenidos, así como las características de los mismos.

Otro de los grandes problemas que se presentan en el diseño de este tipo de redes es la capacidad de conexión a la red. Existen mecanismos que permiten a los usuarios conectarse a una red sin conocer la localización de

dicha red. Mediante algoritmos de descubrimiento en las redes locales, un usuario dispondría de la capacidad teorica de conectarse a una red P2P si existiese un nodo perteneciente a dicha red dentro de su alcance.

Una solución sencilla a este problema es crear un nodo que sirva a modo de puente para aquellos nodos que deseen conectarse a la red, gracias a un mecanismo de *bootstrapping*. Este mecanismo lo que permite es que cualquier nodo pueda "engancharse" a una red a través de un nodo conectado a la misma. Para poder realizar esta técnica, se debe poner a disposición de todos los nodos, un nodo "puente". Para ello, cualquier usuario, o más bien cualquier nodo, debe conocer la dirección de éste nodo puente, de lo contrario, se tendría que optar por un sistema de descubrimiento parecido al mencionado anteriormente.

Los servicios de hosting online como *Amazon Web Services*, *Google Cloud Plataform* y otros similares permiten alojar servidores que hagan uso de servicios en red, con una dirección IP estática. Esto ofrece la posibilidad de alojar el nodo puente en un lugar concreto de la red, donde cada nodo pueda acceder a él para poder conectarse cuando lo desee. Lo único que los otros nodos necesitan es la dirección estática del nodo puente, algo que se les puede proporcionar en su diseño.

Surge, por tanto, una oportunidad de unificar ambos problemas planteados en una única solución; crear un nodo central que actue además de nodo puente. El hecho de apostar por esta decisión de diseño altera el tipo de sistema P2P, pasando de un sistema distribuido a un sistema semi descentralizado. Así mismo, surgen otras complicaciones asociadas a este tipo de sistema, problemas de comunicación entre nodo Master y otros nodos, indisponibilidad del nodo Master, saturación de peticiones de conexión a la red via UDP, etc.

Idealmente, esta solución se implementaría haciendo uso de mas de un nodo central, conectados en malla entre sí, de manera que ante la indisponibilidad de servicios de un nodo Master puntual, los otros nodos centrales podrían asumir sus funciones, minimizando el impacto de la pérdida de este nodo central.

Aún en el caso de que el único nodo Master del sistema se encontrase indisponible, la red sería capaz de continuar funcionando, permitiendo a los usuarios de la misma disponer de los servicios de publicación de nuevos contenidos, así como la capacidad de descarga de ficheros existentes previos

al fallo del nodo central, aunque en este caso la disponibilidad de dichos contenidos no estaría asegurada.

En este proyecto, se implementa un único nodo central que funciona a modo de índice de contenidos. Siendo accesible a todos los demás nodos de la red, contiene diferentes estructuras de objetos de manera que permiten al nodo Master mantener un control de todos los contenidos que se distribuyen por la red.

Al tratarse del único nodo central debe reunir las siguientes características:

- Sistema bootstrap para otros nodos: Como ya se ha explicado anteriormente, el nodo central debería funcionar como nodo puente para otros nodos que deseen conectarse a la red existente. Esto se consigue a través del alojamiento de un servidor UDP en la dirección del nodo Master, a la escucha de nuevas peticiones para responder con los parámetros de conexión. No existe ningún mecanismo de autenticación de los usuarios ya que se desea respetar el principio de privacidad de los mismos.

Los mecanismos de conexión a la red se explican en el capítulo posterior de *Protocolo de Comunicación*.

- Capacidad de respuesta a las peticiones: Como parte esencial de este tipo de sistema semi descentralizado, la red debe ser escalable en la medida en la que los nodos centrales tengan capacidad de respuesta ante las peticiones de los otros nodos que soliciten información. Como se explicará más adelante, el *Protocolo de Comunicación* del sistema se diseña sobre P2P, ya que el establecimiento de conexiones TCP para la comunicación entre nodos hace uso de un número mayor de recursos por parte de ambos extremos. Por tanto, la capacidad de manejo y respuesta de distintos paquetes de distintos usuarios es clave para ofrecer un servicio de estas características.

- Agilidad en las búsquedas: El sistema de clasificación que contenga el nodo central debe estar lo suficiente optimizado como para que las búsquedas que se realizen sean rápidas. En otros sistemas, se plantea este mismo problema optando por la implementación de bases NOSQL como MongoDB. En este proyecto se optó por la construcción de un sistema de búsqueda basado en estructuras de clave valor como HashMaps. Esto permite implementar una solución más a medida de los requisitos del sistema.

- Rapidez frente a las actualizaciones de la red : Al tratarse de una red muy dinámica, donde se distribuyen, crean y destruyen distintos ficheros con bastante facilidad, la estructura de clasificación de contenidos de la red debe permanecer atenta ante estos cambios para poder ofrecer un servicio a tiempo real.

La estructura principal de clasificación de contenidos de la red es son distintas estructuras de tipo clave valor, que se organizan de la siguiente manera formando una estructura parecida a un árbol.

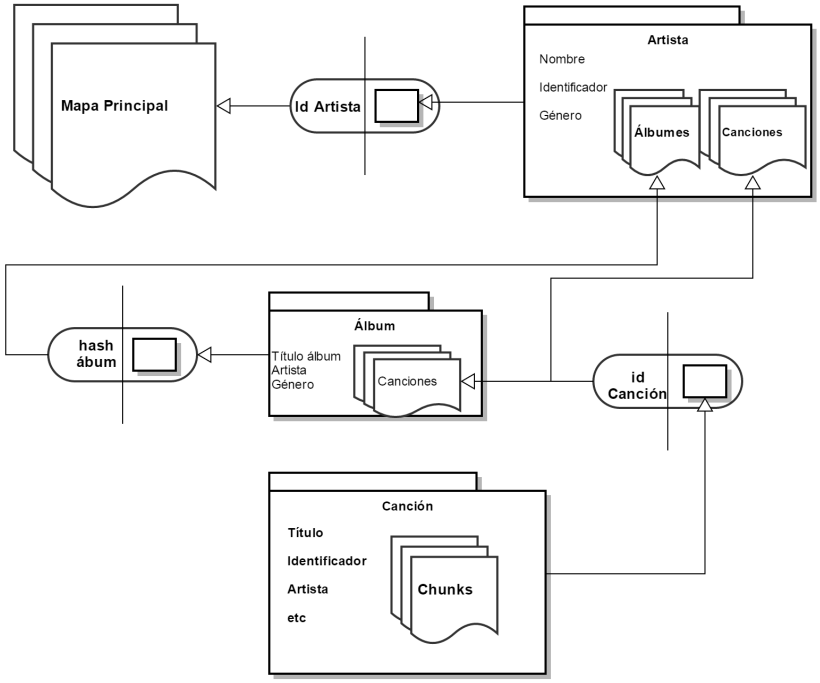

Figura 3.1: Visualización de la estructura de clasificación tipo árbol

Para dotar al sistema de un sistema de prevención de fallos, se desarrolla una segunda estructura complementaria a la principal para intentar proporcionar mecanismos de búsqueda alternativos en caso de fallo en la primera estructura.

Esta segunda estructura debe estar igual de actualizada que la estructura principal, sin embargo debe manipular la información que recoge de la actividad de la red, de una manera distinta para proporcionar un servicio de

búsqueda de apoyo a la red principal.

El nodo central hace uso de esta estructura secundaria cuando los algoritmos de búsqueda(*Apartado 3.2.3*) sobre la estructura principal no encuentran la información referente a una petición. Sirve además, como sistema de correción de errores, en caso de que la información de la estructura principal se encuentre incompleta o sea errónea.

La estructura secundaria es un mapa que alberga las parejas clave-valor de los ficheros de Metadatos Md5Properties(*Apartado3.2.1*), de manera que resulte facil acceder a la información contenida en ella.

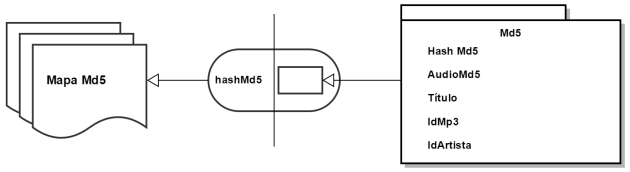

Figura 3.2: Visualización de la estructura secundaria del nodo de control

3.2. Arquitectura

3.2.1. Estructuras y Metadatos

De la misma manera que otros protocolos basados en Peer-to-Peer (*Apartado 2.1.3*), se definen una serie de ficheros adjuntos a los archivos que se distribuyen por la red, llamados ficheros de Metadatos. Estos ficheros contienen información relevante acerca de cada uno de los recursos, creando uno o más tipos de ficheros de Metadatos por archivo compartido.

El fín último de estos ficheros de Metadatos es la agilización de los procesos de clasificación y búsqueda llevadas a cabo por la DHT. Para ello, en este proyecto se hace uso de distintos ficheros, algunos únicamente exclusivos de los nodos de control.

Mp3Properties

Para poder llevar a cabo un sistema de clasificación de contenidos optimizado para este tipo de redes, debemos conocer las propiedades de cada uno de los ficheros. Cada uno de los ficheros compartidos por la red necesita tener un archivo adjunto con la información requerida por el sistema, de tal manera que se puedan desarrolar unos algoritmos de clasificación(*Apartado 3.2.2*) y búsqueda(*Apartado 3.2.3*) adecuados a la estructura. Dada la necesidad de conocer las propiedades de cada canción, se hace uso del sistema de reconomiento en red (*Apartado 4.1.1*) para poder extraer la información necesaria.

Existen otros métodos que podrían resultar más sencillos de implementar, como la lectura de los *tags* de cada fichero ".mp3". Esta otra solución plantea un problema de confianza en la autenticidad de los ficheros que se comparten. Esto es, un usuario puede compartir una canción de origen desconocido, ya sea legítimo o copia ilegal, sin embargo el problema es la precisión de la información de los tags del fichero. Un archivo puede haber sido manipulado por un usuario u otra persona, previamente a ser compartido, dando a lugar un fichero con propiedades erróneas. Para salvar este obstáculo, se hace uso del servicio de reconocimiento en red, que contrasta la canción contra una base de datos inmensa, proporcionando información en base al audio analizado.

Una vez obtenidos los campos de información que nos interesan, el Objeto resultante es el siguiente:

```
class Mp3Properties {

    public String fileName;
    public String title;
    public String artist;
    public String idArtist;
    public String bitRate;
    public String audioMd5;
    public String md5;
    public String sampleRate;
    public String release;
    public String idMp3;

    public Map<Integer, String> order = new HashMap<Integer, String>();

}
```

Figura 3.3: Estructura del Objeto Mp3Properties

•**fileName:** Es importante que el nombre del archivo sea adecuado.En primer lugar, es capital para poder localizar los archivos dentro de los nodos que albergan el fichero. El nombre del archivo debería ser lo mas conciso y fiel a la canción, de tal manera que no se presta a confusión con otros ficheros, ni se dan errores de carga y descarga de ficheros incorrectos debido al nombre de fichero.

•**title:** El título de la canción es importante dentro de la clasificación dentro de su artista. Es raro que un artista o grupo componga dos canciones que compartan título. Sin embargo, los títulos de las canciones no suelen ser únicos ya que hay muchos artistas que emplean el mismo título para sus propias canciones. Este campo de información no resulta muy útil desde una perspectiva de clasificación global.

•**artist:** El nombre del grupo o artista de la canción no será la primera opción que se emplee para la clasificación de artistas, debido a las posibles variantes del mismo o repeticiones de nombres.

•**idArtist:** El identificador de cada grupo musical o artista es único, lo que sí permite llevar a cabo un sistema de clasificación global en base a este valor, agrupando las canciones dentro de sus artistas y formando un mapa con los identificadores de los artistas.

•**bitRate:** Dato útil de cara a la implementación del servicio del servicio *Streaming de música*. Permite al sistema adaptar los *streams* de información,

ajustándolos a la tasa binaria de descarga de cada usuario.

•**audioMd5:** Valor de resumen realizado sobre el propio audio, obteniendo un valor hash único de cada canción. Garantiza que cada fichero pueda tener un identificador en base al contenido del audio, pudiendo distinguir entre las versiones de cada canción. Se trata de un campo de información muy útil ya que permite distinguir entre versiones de una misma canción, así como el agrupamiento de las mismas versiones en una sola entrada dentro del sistema de clasificación.

•**md5:** Valor de resumen de un fichero, realizado sobre las propiedades de cada fichero. Puede usarse como otro identificador para clasificar cada canción, en caso de que el algoritmo de clasificación por identificador *idMp3* no se realizase de manera correcta.

•**sampleRate:** Se trata de un campo de información que muestra el tamaño de la muestra de audio analizada por el sistema de reconocimiento en red.

•**release:** Álbum o disco al cual pertenece la canción. De la misma manera que se pueden realizar agrupaciones por artistas, el posible implementar un sistema de clasificación por álbumes, reduciendo el tamaño de cada grupo, aunque aumentando el número de cada grupo.

•**idMp3:** Identificador principal del archivo. Este campo de información resulta clave para la clasificación de las canciones dentro del sistema. Al ser un identificador único, proporcionado por el servicio de reconocimiento en red, se puede realizar diversas agrupaciones de canciones, siempre identificadas por este valor. Más allá de ser un valor que permite su clasificación en estructuras clave-valor, se emplea para los servicios de distribución y descarga de ficheros(*Apartado 4.1.3*) .

•**Map**$< Integer, String >$ **order:** Este mapa contiene la dirección de cada uno de los segmentos del fichero, así como el órden en el que deben manipularse en el caso que éstos se deseen combinar de nuevo(*Apartado* **??**).

ArtistProperties

Gracias al servicio de reconocimiento en red, podemos obtener información de las canciones y de los artistas de las canciones. Únicamente se

almacena la información de los artistas de las canciones que se comparten a través de la red. En el momento que no se distribuya ninguna canción de este artista, este fichero se borraría del árbol automáticamente despues de que la última copia de la última canción fuese borrada también. Aún pudiendo obtener más información de los autores, la información pragmática que es necesaria para la estructuración del arbol es la siguiente:

```
class ArtistProperties {

    public String artist;
    public String idArtist;
    public String genre;

    public Map<String, ReleaseProperties> releases = new HashMap<String, ReleaseProperties>();
    public Map<String, Mp3Properties> order = new HashMap<String,Mp3Properties>();

}
```

Figura 3.4: Estructura del Objeto ArtistProperties

•**artist:** Nombre del grupo Musical. Esta información es útil para el usuario aunque no se hace uso de ella para la estructuración del árbol. Cada artista o grupo musical tiene su propio archivo. Dado que la información que se almacena en estos ficheros proviene de un servicio externo, es muy posible que existan archivos de propiedades de artistas bajo el nombre de un grupo y otro que exista bajo el nombre del componente principal de un grupo. En cualquier caso esta información no afecta a como se organiza el árbol.

•**idArtist:** Se trata del identificador principal del Objeto. Este identificador sirve como clave del mapa principal de la estructura. Este sistema de organización resulta un poco complejo debido a las numerosas iteraciones que se deben realizar con distintos pares clave valor para poder acceder a la información contenida en la estructura principal. Sin embargo, como ya se comentó anteriormente en el apartado de *Mp3Properties*, cada archivo de propiedades de cada canción contiene el identificador del artista, lo que permite crear esta estructura principal superior para un mayor control y organización que en lugar de usar un único mapa con las parejas clave valor de *Mp3Properties*.

•**genre:** Como un sistema alternativo de organización, se puede llegar a implementar un sistema de búsqueda y recuperación de canciones por género.

•**Map**$< hash(Release), ReleaseProperties >$ **releases:** Siendo ésta otra estructura tipo HashMap dentro de un mapa, este segundo mapa actua como un segundo nivel de organización. Cada canción generalmente está

incluida en un disco o álbum, lo que permite ordenar de manera más compacta las canciones de cada artista creando un Objeto tipo *ReleaseProperties* como se comenta a continuación. La clave de cada pareja dentro de este mapa es el valor hash del título del disco. Esto se debe a que el sistema de reconocimiento externo no otorga un identificador a cada álbum, de manera que debemos hallar un sistema de idenzación para estos objetos, empleando la misma misma función hash que se aplica a los ficheros compartidos por la red P2P.

•**Map**$< idMp3, Mp3Properties >$ **order:**.En el caso de que no se pudiese ordenar las canciones dentro de los Objetos tipo *ReleaseProperties*, el Objeto *ArtistProperties* debe ser capaz de almacenar las canciones que pertenecen a este artista. Para ello se crea este segundo mapa, para poder almacenar los Objetos *Mp3Properties* que se clasificarían en el mapa anterior. Su clave es el identificador de cada canción.

ReleaseProperties

Como se ha comentado anteriormente, este Objeto podría entenderse como una virtualización de la información útil que podría aportar un álbum o disco a un sistema de organización de canciones. Entre las propiedades de este Objeto, encontramos de nuevo otra estructura de mapa que sirve como tercer nivel de clasificación dentro de la estructura principal.

```
class ReleaseProperties {

    public String release;
    public String IDArtist;
    public String genre;

    public Map<String, String> songs = new HashMap<String, String>();

}
```

Figura 3.5: Estructura del Objeto ReleaseProperties

•**Release:** De nuevo, se trata de un campo de información que resulta útil a los usuarios que hagan uso de este Objeto dentro del Master, y que permite otros sistemas de clasificación en un futuro.

•**idArtist:**Es importante conocer el Objeto *ArtistiProperties* al que se le relaciona. A este nivel, el identificador del artista permite su correcta

inserción dentro de la estructura principal.

•**genre:**Este campo de información resulta relevante para poder clasificar correctamente los álbumes o discos de música por estilos musicales en caso de que se desee implementar un sistema de búsqueda de este tipo.

•**Map**< $idMp3, Mp3Properties$ >**songs:** La manera de ordenar los Objetos *Mp3Properties* que le corresponden a este álbum podría aproximarse de distintas maneras, ya fuese con una estructura tipo Lista o tipo Mapa. Finalmente se optó por la implementación con una estructura tipo Mapa debido a que las búsuquedas por clave valor dentro de este tipo de estructuras son mas rápidas, lo cual dotan al nodo Master una agilidad superior.

Es importante garantizar que el sistema de clasificación de contenidos de la red sea lo más eficiente posible. Es por ello que se implementa de manera secundaria y complementaria a la estructura principal, un segundo HashMap. Se hace uso de este segundo mapa para intentar localizar Objetos *Mp3Properties* a través de sus valores de hash **md5** que proporciona el servicio de reconocimiento en red.

Esta segunda estructura se emplea en casos muy puntuales dentro de los algoritmos de búsqueda a modo de apoyo y corrección de errores. Estos casos se contemplan en el apartado de Algoritmos de Clasificación y *Algoritmos de Búsqueda* del siguiente capítulo.

Md5Properties

Este segundo mapa contiene unas parejas clave valor, donde el contenido de cada pareja es el Objeto *Md5Properties*.

```
class Md5Properties {

    public String md5;
    public String audioMd5;
    public String fileName;
    public String title;
    public String IDMp3;
    public String IDArtist;

}
```

Figura 3.6: Estructura del Objeto Md5Properties

•**md5:** Como ya se comentó anteriormente, cada uno de los Objetos tipo

Mp3Properties contiene un campo de información con el valor hash md5 de la canción. Este valor md5 es único para cada canción, idenpendientemente de la versión, el artista que versione la canción, etc. Por tanto, el valor hash md5 de cada fichero es único y el mismo para ficheros identicos. Al tratarse de un valor único, pueden ejecutarse algoritmos de clasificación y búsqueda en función de este valor.

•**audioMd5:** De la misma manera que se tiene un valor a través de la función hash md5 de las propiedades de la canción. Al tratarse de otro identificador único, pueden ejecutarse algoritmos de clasificación y búsqueda en función de este valor.

•**fileName:** Es importante poder blindar el sistema de clasificación frente a propiedades susceptibles de cambio por parte de los usuarios de la red, como el cambio del nombre del fichero. Para ello, se almacena el valor original del fichero original en caso de que éste se modifique a posteriori. En el supuesto de que un usuario haya realizado una correción o mejora sobre este campo de información, los algoritmos de clasificación modificarán este valor.

•**title:** De la misma manera que en otros elementos de las mismas características, es importante manejar campos de información comprensibles para los usuarios, en caso de que se necesiten cambiar elementos del sistema que las correciones automáticas no son capaces de llevar a cabo.

•**idMp3:** Este identificador resulta capital a la hora de realizar búsquedas por el sistema de clasificación de ficheros. El hecho de que este campo de información esté contenido en este Objeto es debido que se ha tenido que hacer uso de la estructura de apoyo para poder clasificar o buscar un fichero en concreto, por tanto habrá operaciones que requieran modificar el valor de este campo, según los mecanismos que hagan uso de este campo.

•**idArtist:** De la misma manera que el identificador del archivo ".mp3" resulta clave para realizar modificaciones en la estructura principal, el identificador del artista es también relevante, según los mecanismos que hagan uso de esta información.

Estructuras contenedoras

En vista de que existen varios valores y campos de información que permiten clasificar la información de maneras complementarias, el nodo central hace uso de estructuras contenedoras, donde se incluyen los ficheros de Metadatos y otros Objetos que permiten agilizar la búsqueda de información, ya sea acerca de la topología, los contenidos o ambos.

En total, el nodo central cuenta con cinco estructuras contenedoras, dos de comentadas anteriormente(*Apartado 3.1*), la estructura principal o *arbol* y el mapa secundario conteniendo los ficheros de Metadatos Md5Properties. Las demás estructuras se plantean con el fin de agilizar las peticiones recibidas por el nodo central, siendo estas las siguientes:

•**Mapa**$< IdMp3, Mp3Properties > contenidos$**:** Este mapa contiene una lista actualizada de todos los contenidos disponibles en la red. Estos contenidos no se encuentran clasificados, como sí se encuentran en la estructura principal. Su uso principal es para la respuesta de peticiones de "catalogo"(*Apartado 4.1.3*) por parte de los nodos clientes.

•**Mapa**$< IdMp3, Lista < Nodos >> nodosPorCancion$**:** Esta estructura relaciona los contenidos con los nodos que los contienen. Haciendo uso de esta estructura, se pueden crear facilmente mecanismos de réplica de contenidos(*Apartado 4.2.3*), entre otros para evitar la pérdidas de contenidos frente a las desconexiones de los nodos.

•**Mapa**$< Nodo, ParametrosDeRed > nodos$**:** Esta estructura permite al nodo central mantener un control sobre los nodos clientes conectados a la red, conociendo sus parámetros de red como su dirección IP, el puerto del cliente u otros campos de información.

3.2.2. Algorítmos de clasificación

Ya que el sistema requiere de una estructura de clasificición actualizada en tiempo real, el nodo central debe manejar dicha estructura con agilidad y rapidez para poder incorporar lo más eficazmente posible la información para futuros usos. Los algoritmos de clasificación desarrollados trabajan sobre estructuras tipo mapa, ordenados por una relación clave valor que permite modificar y recuperar los contenidos de la estructura de una manera organizada y optimizada.

La clasificiación de los contenidos se produce cuando un nodo cliente notifica al nodo Master la carga de un nuevo contenido a la red. Cuando un nodo cliente carga un nuevo contenido a la red, éste no sabe si se trata de un fichero nuevo o de si se trata de un fichero que existe por la red. La función de comparar y clasificar estos contenidos compete únicamente al nodo central.

Dado el caso de que un cliente cargara un nuevo fichero de audio a la red, existiendo una versión distinta de esta misma canción, tendría que clasificarse bajo un identificador distinto. Este proceso de distinción podría llegar a implementarlo un nodo cliente durante el proceso carga. Aún llegando a realizar una distinción entre estas dos versiones (los identificadores de audio serían distintos), el cliente debería notificar al nodo central la carga del nuevo fichero, bien para que el nodo central sepa que el nodo cliente posee este fichero y por tanto sería equivalente a que el nodo cliente hubiese realizado una réplica del fichero ya distribuido(*Apartado 4.2.3*), o bien para crear una nueva entrada dentro de la estructura de clasificación de información.

El algoritmo general de clasificación se define por los siguientes pasos:

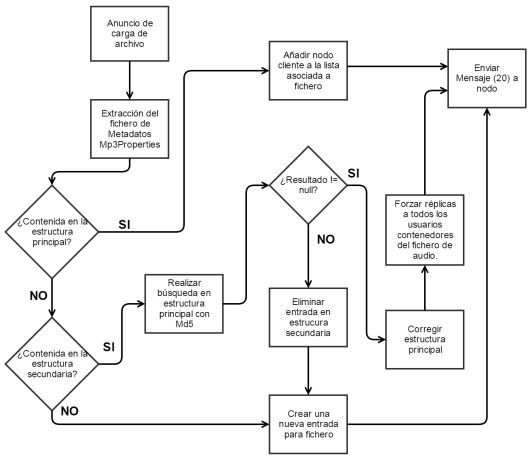

Figura 3.7: Visualización del algoritmo general de clasificación

- En el momento que el nodo central recibe la notificación de la carga de un nuevo fichero, extrae el identificador del fichero de audio y descargaría el fichero de propiedades directamente desde la red Peer-to-peer. Este es el único caso en el que el nodo central hace uso del servicio de descarga de ficheros.

- Una vez descargado, procede a realizar, realiza una consulta en la primera estructura para determinar si se encuentra almacenado bajo ese identtificador. Primero procedería a buscar el fichero de Metadatos correspondiente al artista. Si éste existiera, se realizaría una búsqueda por identificador dentro del mapa de canciones contenido del artista para comprobar si existe el fichero de propiedades. En caso de que existiese, se accedería a la lista *nodosPorCanción 3.2.1* y se añadiría al nodo cliente. Por último se enviaría un paquete de confirmación(*Apartado 4.2.2*) al nodo cliente.

- En el caso de que la búsqueda sobre la estructura principal no diese un resultado, se realizaría otra búsqueda sobre la estructura secundaria para verificar que el fichero cargado fuese nuevo, usando el valor *md5* como clave. Si la búsqueda sobre este segunda estructura no diese un resultado, se crearían los archivos de Metadatos ArtistProperties, ReleaseProperties y Md5Properties a partir del fichero Mp3Properties. Se crearía una nueva entrada en la estructura principal, añadiendo los ficheros de Metadatos ArtistProperties y ReleaseProperties, y añadiendo el fichero de Metadatos Md5Properties a la estructura secundaria. Por último se enviaría un paquete de confirmación(*Apartado 4.2.2*) al nodo cliente.

- En el caso de que la búsqueda sobre la estructura principal no diese un resultado, se realizaría otra búsqueda sobre la estructura secundaria para verificar que el fichero cargado fuese nuevo, usando el valor *md5* como clave. Si la búsqueda sobre este segunda estructura diese un resultado, se realizaría una búsqueda en la estructura principal usando los valores del fichero de Metadatos Md5Properties.

En el caso de que se encontrara un resultado positivo, el nodo central realizaría una corrección de errores usando los parámetros del fichero Mp3Properties descargado y forzando a los otros nodos clientes contenedores de este fichero que actualizasen su fichero de propiedades a través del mecanismo de réplica(*Apartado 4.2.3*).

Si por el contrario, la búsqueda realizada en la estructura principal con los valores del fichero de Metadatos Md5Properties no diese resultado, se procedería a eliminar la entrada de la estructura secundaria. Se crearía una nueva entrada en la estructura principal, añadiendo los ficheros de Metadatos ArtistProperties y ReleaseProperties, y añadiendo el fichero de Metadatos Md5Properties a la estructura secundaria. Por último se enviaría

un paquete de confirmación(*Apartado 4.2.2*) al nodo cliente.

3.2.3. Algorítmos de búsqueda

Una vez esté definida la estructura principal y las secundarias, el nodo central debe ser capaz de acceder a esa información contenida de una forma fácil y rápida de tal manera que pueda tener a muchas peticiones de manera simultánea. En base a las peticiones recibidas por los nodos clientes el nodo central de ser capaz de decidir que busquedas debe realizar sobre las de estructuras. Para ello se definen unos algoritmos de búsqueda especialmente diseñados para cada tipo de petición. El nodo central será capaz de distinguir entre las peticiones gracias al protocolo de comunicación.

Cada una de las estructuras funciona de una manera distinta a las otras con lo cual el algoritmo de búsqueda sobre cada una de ellas debe ser diferente. La estructura principal, al ser la más compleja tendrá por tanto el algoritmo de búsqueda más complejo.

Pueden definirse dos tipos de búsquedas, dependiendo de la información que se quiera recabar. Un nodo cliente puede solicitar el nodo central una petición muy concreta sobre un valor muy concreto o una búsqueda general.

- En el caso de realizar una búsqueda muy concreta se busca hallar el valor asociado a la clave. Al utilizar estructuras de tipo clave valor, este tipo de búsquedas están muy optimizadas, ya que cumplen con los requisitos de agilidad y rapidez que se necesitan. Indiferentemente de la estructura sobre la que se realice la búsqueda, el nodo central únicamente busca un resultado usando una clave concreta. Siempre que el valor que se esté buscando se halle dentro de la estructura sobre la que se esté realizando la búsqueda, la petición será respondida de manera inmediata.

- Existen otro tipo de búsquedas más generales donde el nodo central debe iterar sobre la estructura correspondiente hasta hallar un resultado no nulo. Estas búsquedas son menos rápidas pero pueden llegar a proporcionar resultados no nulos donde el otro tipo de búsquedas no.

Es importante hallar un compromiso entre ambos tipos de búsquedas. Los algoritmos de búsqueda que emplea el nodo central buscan siempre responder a las peticiones con un resultado no nulo. Existen dos tipos de búsquedas que se pueden realizar:

• **Búsqueda sobre un fichero de audio concreto.** Dado el caso de que el nodo central recibiese una actualización sobre un contenido particular como por ejemplo la carga de un fichero nuevo a la red el nodo central podrá hacer uso del identificador para acceder a la información contenida bajo ese identificador dentro de las estructuras dotando de una mayor agilidad a los algoritmos de clasificación. Los algoritmos de búsqueda son complementarios a los algoritmos de clasificación.

• **Acceso a una lista determinada.** En el supuesto que un nodo cliente realice una petición sobre toda una lista completa, contenida en algunas de las estructuras, como por ejemplo la descarga del catálogo nota central deberá responder a esta petición con varios mensajes. En cada uno de estos mensajes, se incluirá la información contenida en una entrada de la lista. Gracias al protocolo de comunicación(*Apartado 4.2*) el nodo central es capaz de enviar ráfagas de mensajes hasta poder enviar toda la lista.

3.3. Software a usar

La implementación de todo este sistema se realizará a un nivel de abstracción superior al de la red de transporte. Para la red de transporte se usa el sistema TomP2P, una implementación en Java de una DHT(*Apartado 2.1.3*).

Gracias a las funciones y a la biblioteca de este sistema se podrán crear las entidades antes mencionadas y poder conectarlas. Únicamente este sistema para transportar los paquetes de datos de los ficheros de audio. La implementación de las demás estructuras y objetos se realiza haciendo uso de estas funciones pero trabajando fuera del sistema TomP2P.

3.3.1. Funciones especiales

De manera ejemplificante se comentará brevemente algunas de las funciones principales de las que hacen uso los nodos clientes relacionando el nivel de transporte y de abstracción superior.

- SplitFile(): Es preferible que los nodos clientes compartan un mayor número de ficheros de menor tamaño que al contrario. Esto permite a la vez manejar mejor los paquetes que circulan a través de ella. Por tanto un

fichero de audio, como un archivo MP3, tendrá que ser partido en otros archivos de menor tamaño. Esta función lo que permite generar una lista de ficheros, conteniendo los segmentos el fichero de audio original. Ver Figura 4.3.

- MergeFile(): De la misma manera que es importante generar segmentos a partir de un fichero de audio original, es igual de importante poder combinarlos de nuevo. De esta manera lo que hace un nodo cliente, es solicitar los segmentos de audio uno a uno, pudiendo éstos venir de distintos nodos clientes, y poder fusionarlos en un nuevo fichero que sea idéntico al original.

Capítulo 4

Desarrollo

4.1. Servicios desarrollados

Uno de los objetivos principales de este proyecto es poder ofrecer una serie de servicios a los usuarios a través de los nodos clientes. En algunos casos, estos servicios trabajan en plano secundario, de manera que son invisibles al usuario, pero que resultan indispensables para el correcto funcionamiento de la red.

4.1.1. Servicio de reconocimiento en de audio en red

Este servicio es muy particular y se distingue del resto de los servicios implentados en el sistema. Una vez planteado el problema de la normalización de canciones (*Apartado 3.1*), existen distintas soluciones:

- Creación de una base de datos común para la identificación de los ficheros a través de un script de análisis de audio. La implementación de esta solución plantea la creación de una nueva entidad dentro del sistema de nodos, o bien la incorporación de una nueva tarea para el nodo de control. Ya que se desea implementar un sistema lo más distribuido posible, esta solución no es la más deseable. Si el nodo de control albergase además esta base de datos, haría que la dependencia de éste fuese aún más crítica que antes, ya que para poder ofrecer este servicio, deben evitarse los fallos en el nodo Master.

- Consulta de una base de datos externa. La dependencia de un servicio ajeno a la red para ofrecer un servicio puede no ser los más interesante de cara a la implementación de este proyecto, sin embargo, presenta una serie

de ventajas frente a la solución anterior. En primer lugar, el uso de una base de datos voluminosa permite una mejor clasificación de los contenidos que se distribuyen por la red. En segundo lugar, cada nodo cliente es capaz de ejecutar de manera independiente y libre, de tal manera que no se tienen que realizar peticiones de análisis al nodo de control, sino que cada nodo cliente puede contribuir al crecimiento de la estructura de clasificación.

Un servicio de reconocimiento en red, lo suficientemente ágil y completo como para cumplir con las especificaciones de este proyecto, es el servicio que ofrece el proyecto **echonest**. Este servicio es la base de muchas de las aplicaciones de reproducción en Streaming y descarga de canciones como Spotify y otras similares, todas ellas hacen uso de la base de datos de echonest, que cuenta con mas de 36 millones de entradas. Echonest ofrece, además, un servicio de análisis de audio y clasificación gratuito, proporcionando todo tipo de información acerca de archivos de audio, artistas, etc.

Para poder hacer uso de este servicio externo, se desarrolla un script que funciona de la siguiente manera:

- Seleccionado el fichero de audio para su reconocimiento, se realiza una petición HTTP mediante la función POST, que permite la carga del fichero a su servidor online.

- El servidor en red realiza un análisis sobre el fichero, devolviendo un archivo JSON como respuesta al POST realizado.

- El script analiza el contenido del JSON respuesta, creando un archivo de Metadatos Mp3Properties (*Apartado 3.2.1*) del fichero de audio creado. Ya que el servicio en red hace uso de sus propios parámetros de indexación, el archivo de Metadatos asigna estos mismos campos de información a los suyos propios correspondientes.

De esta manera, cada uno de los nodos puede realizar sus propias peticiones de reconocimiento de ficheros. La distribución de este servicio permite una mayor liberación de tareas por parte del nodo de control.

Una vez que se realizan las peticiones y se obtienen los ficheros de Metadatos correspondientes, los nodos clientes comparten los ficheros y notifican al nodo de control la puesta en conjunto de los nuevos contenidos, de manera que la próxima vez que un nodo cliente solicite al nodo central la

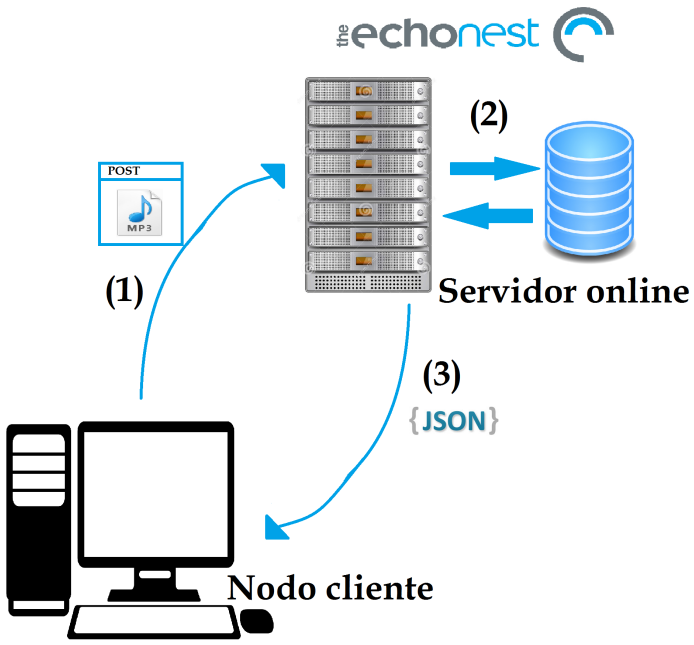

Figura 4.1: Vista previa del proceso de reconocimiento de audio

lista de contenidos de la red, ésta esté actualizada.

4.1.2. Servicio de carga de ficheros

Este servicio permite a los usuarios de los nodos clientes compartir ficheros a través de la red, de manera que puedan distribuirse entre los demás usuarios del sistema.

Para poder asegurar que los ficheros que se comparten correctamente y de manera estructurada, se necesita tratar el archivo a través un proceso que hace uso de diferentes herramientas y servicios diseñados específicamente para esta aplicación. Entre otros, este servicio hace uso del servicio de reconocimiento en red(*Apartado 4.1.1*) para clasificar correctamente el fichero a compartir.

El procedimiento que sigue este servicio se explicará a través del siguiente ejemplo:

- Primero el usuario debe selecionar el fichero que desee compartir a partir de la herramienta de carga de archivos dentro de la interfaz de usuario. El sistema está diseñado de tal manera que sólamente puedan seleccionarse los ficheros tip MPEG-1 Audio Layer III o **MP3**.

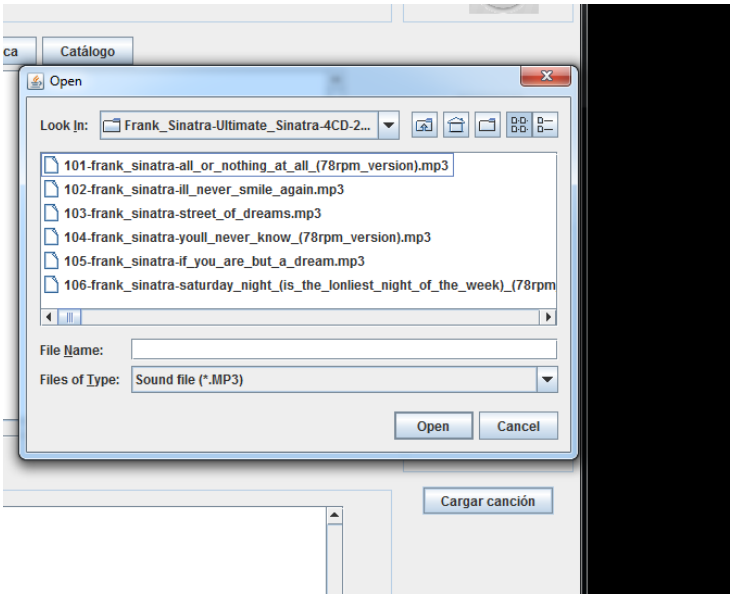

Figura 4.2: Visualización del explorador de archivos para cargar ficheros .mp3

- Una vez seleccionado, comenzará el proceso de identificación en línea del fichero. Este proceso generará un objeto que contenga la información de las propiedades de la canción(*Apartado 3.2.1*). Este archivo de propiedades contiene información importante para la clasificación de este fichero, pero por encima de todo, contendrá un identificante único que lo identificará como esa versión de la canción. El sistema de reconocimiento en red asigna un identificador único a cada versión de cada canción, queriendo decir que si se trata de una canción que otro usuario haya compartido anteriormente, tendrán el mismo identificador.

Particionado de los ficheros

- Ahora que se puede manipular la información sobre este fichero, el siguiente paso es preparar el fichero de audio para poder compartirlo en la DHT. El primer paso es generar segmentos o *chunks* del fichero seleccionado. Los chunks se almacenan en una carpeta diferente a aquella donde se

almacena la música compartida.

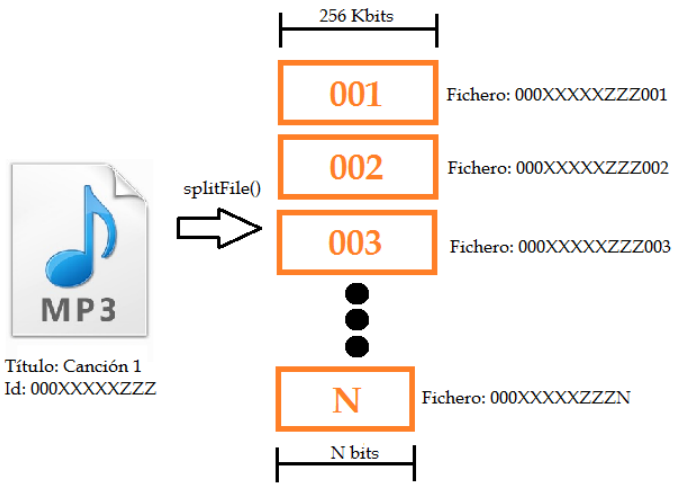

Figura 4.3: Uso de la función splitFile() para separar el fichero

Estos chunks se almacenan como ficheros sin extensión, de tal manera que un fichero de música no podrá ser montado correctamente sin todos sus chunks. El fichero de propiedades contendrá un mapa que almacena el orden y la dirección de cada uno de los chunks.

Para poder partir el fichero de una manera manejable, el servicio hace uso de la función *SplitFile(), Apartado 3.3.1.*

Siguiendo el ejemplo del protocolo BitTorrent, los archivos tienen un tamaño inferior o igual a 256 Kbits. Como se puede ver en la 4.3, no se hace uso de padding para que los chunks tengan todos el mismo tamaño, lo importante es que se encuentren bien enumerados y ordenados para poder realizar la función inversa y juntarlos en los otros nodos de destino.

- A partir de este momento, el fichero de propiedades está completo y listo para poder compartirse por la red. El nodo crea una pareja clave-valor, donde la clave es el identificador de la canción y el valor de dicha clave es el archivo de propiedades. Posteriormente, el nodo comparte los chunks del fichero, cada uno con una clave hash del nombre del fichero.

Hasta aquí, el proceso no serviría de nada si no se le notificase a la red que el usuario ha compartido un nuevo fichero. Es por ello que gracias al

protocolo de comunicación entre Peer-Master(*Apartado 4.2*). El nodo envía un mensaje al nodo Master notificando que el usuario ha compartido el fichero seleccionado. El procedimiento completo se explicará en el Capítulo de *Protocolo de comunicación.*

4.1.3. Servicio de descarga

De la misma forma que un usuario puede compartir y distribuir ficheros de audio por la red, éste es capaz de descargar los existentes en la red mediante el servicio de descarga. En muchos aspectos, el procedimiento es similar a el servicio de reproducción de audio en streaming(*Apartado 4.1.4*).

Para poder descargar un fichero de audio, el usuario debe conocer el identificador del fichero. El procedimiento se explica a través del siguiente ejemplo:

- Los identificadores de los ficheros de audio son secuencias alfanuméricas que no guardan relación con el título de la canción o el nombre del artista. Éstos son asignados por el servicio de reconocimiento de audio en red(*Apartado 4.1.1*), de manera que supondremos el caso en el que el usuario no conozca el identificador del fichero de audio que desee descargar.

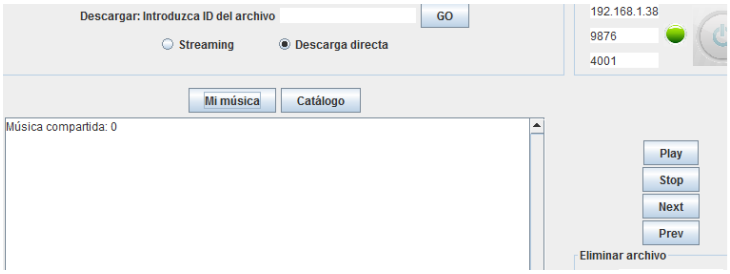

Figura 4.4: Servicio de descarga I

- El usuario puede consultar el "catálogo" de música que alberga el nodo central para averiguar el identificador del fichero que desee descargar. Gracias al protocolo de comunicación(*Apartado 4.2*) existente entre el nodo cliente y el nodo master, el usuario recibe una lista de los contenidos distribuidos por la red.

- El usuario puede entonces, descargar el fichero a través del panel de descarga(*Apartado 4.3.1*), seleccionando la opción de descarga. En el

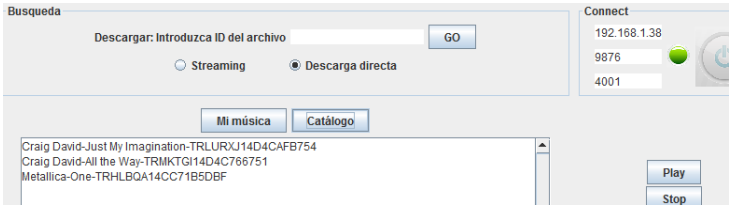

Figura 4.5: Servicio de descarga II

momento que se accione el botón de descarga, el nodo solicita a la red el fichero de propiedades que se encuentra compartido bajo esta clave.

Figura 4.6: Servicio de descarga III

- Una vez descargado el fichero de propiedades de audio de la canción que se desee descargar, el nodo ahora conoce cada identificador de los *chunks*. De esta manera, el nodo solicita y descarga cada uno de los segmentos o *chunks* del fichero de audio.

Combinación de segmentos

Gracias a la función MergeFiles()(*Apartado 3.3.1*), el nodo combina de manera ordenada cada uno de los segmentos. Una vez termina eel proceso de combinación, el nodo añade a su lista de reproducción particular el fichero descargado.

Una vez haya terminado el proceso de de descarga y se haya incluido correctamente el fichero en la lista de reproducción local, el nodo comparte los ficheros descargados, anunciando mediante el sistema de mensajería del protocolo de comunicación al nodo master que ahora el usuario ha descargado este fichero y que lo comparte también.

El procedimiento de este servicio automáticamente comparte lo que descarga, creando una réplica en cada nodo que descarga los ficheros.

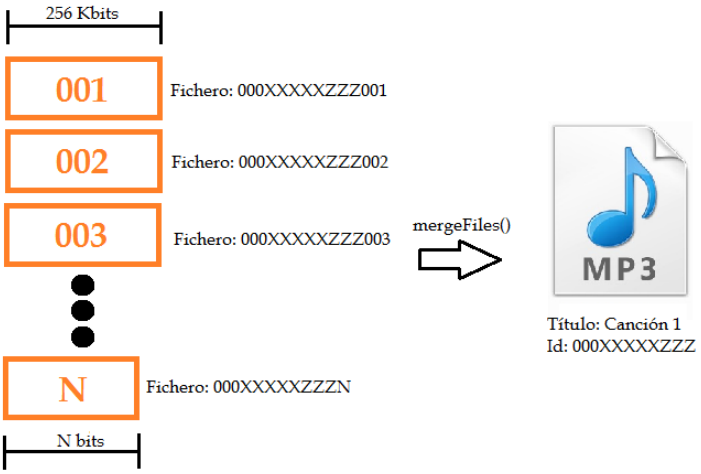

Figura 4.7: Visualización del proceso de unificación de un fichero a partir de los chunks a través de la función mergeFiles()

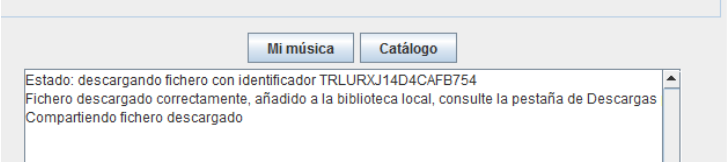

Figura 4.8: Servicio de descarga IV

4.1.4. Servicio de reproducción en Streaming

El servicio de reproducción streaming funciona de una manera muy parecida al servicio de descarga. El nodo cliente solicita a la red los los segmentos de un fichero de audio que se encuentren bajo identificador que el busca.

Para ello organiza todos los segmentos de manera que generen un stream de audio, y poder reproducirlos de una manera directa únicamente teniedo

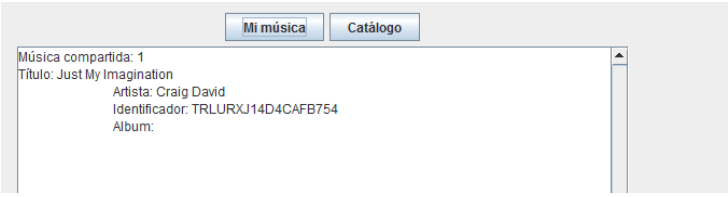

Figura 4.9: Servicio de descarga V

que almacenarlos de una manera temporal. Cuando la reproducción halla finalizado, los segmentos se eliminan sin notificar al nodo central la descarga de éstos.

4.2. Protocolo de comunicación

Al estar formado el sistema por nodos que pueden actuar de forma independiente, realizando diversas tareas de manera asíncrona, se necesita un sistema de señalización entre nodos para poder formar una estructura de control de topología y ficheros distribuidos. Este sistema de señalización se implementará a través de un protocolo de comunicación entre nodos para anunciar las acciones emprenda un usuario o peticiones que un nodo requiera.

En un primer lugar, debe plantearse el sistema que se emplee para comunicarse entre nodos. Dada la implementación realizada hasta ahora, existen dos vías posibles:

Sistema de mensajería sobre P2P: Al funcionar sobre Peer-to-Peer, los nodos pueden aprovechar las conexiones existentes para implementar un protocolo de comunicación. Hasta ahora, el sistema hace uso de Peer-to-Peer únicamente para tráfico de paquetes y ficheros de propiedades de audio, lo cual requeriría modificar la identificación de los paquetes que se transmiten por la red, para distinguir entre ficheros de datos y paquetes pertenecientes al protocolo de comunicación.

El API de TOMP2P permite implementar un servicio de mensajería entre nodos, haciendo uso de unas clases y *Enums* predefinidos para mandar mensajes estandarizados entre los nodos. Permite además crear otra serie de mensajes más personalizables pero no muy flexible, ya que deben anunciar comandos o acciones muy determinadas, de lo contrario, se tendría que desarrollar un protocolo para la identificación y generación de los mensajes compartidos.

Otra de las opciones sería crear canales de escucha cada vez que un nodo desee comunicarse con otro, bien estableciendo una sesión TCP entre ellos o a través de intercambios de datagramas por UDP. En este caso, cada vez que un nodo quisiese comunicarse con un nodo tendría que generar una nueva sesión de intercambio a nivel IP para después intercambiar información a través de los puertos. La implementación de TOMP2P no permite tener un

puerto abierto sin usar mas de un tiempo determinado.

La implementación en Peer-to-peer presenta, por tanto, dos inconvenientes importantes:

- Primero, la necesidad de cargar más la red de transporte en Peer-to-peer para soportar el protocolo de comunicación. Tendrían que modificarse la estructura de los paquetes compartidos por la red para distinguir entre datos y mensajería. A su vez, debería implementarse un sistema de aviso entre nodos para la notificación de mensajes.

En segundo lugar, cuando un nodo inactivo, desee conectarse a la red, debe hacerlo a través del mecanismo de bootstrapping(*Apartado 3.1*), a través de un intercambio de paquetes sobre UDP. En vista de esta necesidad, para compatibilizarlo con un servicio de mensajería sobre P2P deberían definirse unas excepciones en el protocolo de comunicación o implementarlos de manera separada e independiente.

Todo ello supone una alteración de la implementación desarrollada demasiado costosa frente a la solución planteada usando un protocolo sobre UDP.

Sistema de mensajería sobre UDP: Los nodos cuentan con un puerto dedicado para el intercambio de paquetes sobre UDP. Estos puertos pueden permanecer abiertos y activos durante toda la sesión del usuario. Al iniciarse un cliente, el primer paso es intercambiar la información de los parámetros de conexión necesarios para que el nodo cliente pueda conectarse a la red P2P. Este intercambio se realiza sobre UDP.

Haciendo uso de este puerto, se pueden crear y recibir paquetes para comunicarse con los otros nodos, una vez estén conectados a la red. Como los paquetes intercambiados por UDP no son más que un *array de bytes* de un tamaño M, pueden crearse cabeceras, separadores y otros campos, ajustando el tamaño y el contenido en base a los requerimientos del sistema.

Al trabajar sobre UDP, no se añade mas carga a la red de transporte Peer-to-peer, pudiendo además, crear un protocolo de comunicación adaptado a las necesidades de la aplicación y los servicios que trabajan por encima de la red de transporte.

4.2.1. Sistema de mensajería sobre UDP

Una vez establecido el sistema por el cual se realizará el intercambio de información entre nodos, la información debe estructurarse para que cada nodo pueda comprender el significado de cada mensaje ya que aún no se han establecido qué bits representan qué campos de información.

Al no recomendarse el uso de datagramas superiores a 2048 bytes, y teniendo en cuenta que la información que se intercambia entre nodos es corta, ya que sirve como sistema de notificaciones y peticiones entre nodos, se plantea una estructura de mensajes de 1024 bytes.

En dichos mensajes deben incluirse tanto información sobre el emisor, como sobre el receptor para asegurar que los mensajes recibidos son dirigidos a este nodo, y en caso de error, notificar al emisor acerca del fallo. Dados que los identificadores creados para hacer uso de la red Peer-to-peer son únicos e identifican cada nodo, resultan idóneos como identificadores de emisor y receptor.

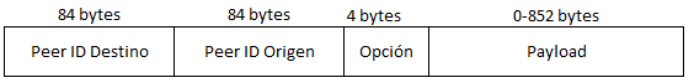

Figura 4.10: Visualización de la estructura de los paquetes de mensajería

Así mismo, los mensajes deben tener un número o indicador para que cada nodo interprete la carga del mensaje en función de este campo de información. Es por ello que seguido de los identificadores de emisor y receptor en el paquete, se añade la "opción" o tipo de mensaje. Al no tener una gran variedad de mensajes, usando números de dos dígitos será más que suficiente para definir todos los tipos posibles de mensajes.

Finalmente, se añade el payload o carga que debe interpretar el receptor identificando el tipo de mensaje. Éste puede hacer uso de los bytes restantes hasta que la suma total de bytes sea 1024. En caso de que la carga hiciese uuso de más bytes, éste se enviaría en varios datagramas de capacidad máxima 1024 bytes. Si la carga de un mensaje supera la capacidad de un datagrama, se enviarían una cola de mensajes, conteniendo la carga inicial ajustada hasta el máximo de la capacidad del paquete. Se estructuraría también el payload, creando un campo "artifical" que permitiese al nodo del

destino identificar correctamente el tipo y contenido final del paquete hasta que el campo de información "artificial" indicase el fín de l cola de mensajes.

4.2.2. Mecanismos de comunicación entre Peer y Master

Para una correcta implementación del sistema, debe desarrollarse un protocolo de comunicación que permita el intercambio de información entre nodos clientes y el nodo central. Es por ello que cada una de estas entidades es capaz de generar sus propios mensajes adaptados a las acciones y peticiones de los usuarios y del sistema, siguiendo la estructura de mensajes comentada en la sección anterior 4.2.1.

Para ello, cada una de las entidades debe poder notificar al receptor las acciones de distribuición de ficheros, las solicitudes de información al nodo central, entre otros servicios.

Los nodos clientes notifican automáticamente cualquier acción relacionada con la red Peer-to-peer, de manera que si el nodo central solicita que el usuario realice una acción en particular, como por ejemplo, crear una réplica de un fichero en concreto(*Apartado 4.2.3*), no tendrá la necesidad de esperar la confirmación de dicha accion.

Cuando los clientes inician un intercambio de información, independientemente de si se trata de una petición o una notificación, siempre esperarán una confirmación por parte del nodo central. Es importante recibir esta confirmación, ya que significa que la red funciona correctamente y pueden ejecutarse los servicios con normalidad.

El protocolo de comunicación, siguiendo la estrucura de los mensajes comentada en la sección anterior 4.2.1, se compone de los siguientes mensajes:

•Mensaje tipo -1: *Error: mensaje erróneo, por favor, reenvie de nuevo.*

•Mensaje tipo 0: *Anuncio de desconexión.*

•Mensaje tipo 1: *Solicitud de los parámetros de conexión a la red P2P.*

•Mensaje tipo 2: *Respuesta afirmativa de conexión a la red, envio de*

Origen	Destino	Tipo	Payload
Master/Cliente	Master/Cliente	-1	null
Cliente	Master	0	Identificador de nodo origen
Cliente	Master	1	Identificador de nodo origen
Master	Cliente	2	Parámetros de conexión
Cliente	Master	5	Identificador del archivo IdMp3
Cliente	Master	6	Identificador del archivo IdMp3
Master	Cliente	9	Metadatos Mp3Properties
Cliente	Master	13	null
Master	Cliente	14	tipo:carga
Master	Cliente	20	ok

Cuadro 4.1: Protocolo de comunicación.

parámetros de conexión.

•Mensaje tipo 5: *Anuncio de distribución de contenido, identificador del archivo IdMp3 en el Payload .*

•Mensaje tipo 6: *Anuncio de eliminación de fichero de audio, identificador del archivo IdMp3 en el Payload.*

•Mensaje tipo 9: *Solicitud de réplica, archivo de Metadatos Mp3Properties en el Payload, apartado 4.2.3.*

•Mensaje tipo 13: *Solicitud de todos los ficheros de audio distribuidos, por favor, enviéme sus identificadores.*

•Mensaje tipo 14: *Respuesta a la solicitud de todos los ficheros de audio. Tipo 1, carga contiene coordenadas de un fichero. Tipo 0, fin de la lista de ficheros.*

•Mensaje tipo 20: *ACK.*

4.2.3. Replica de la información

Existe un mecanismo de réplica automatizado y controlado por el nodo Master. Este mecanismo se implementa con con la intención de evitar la indisponibilidad de los contenidos distribuidos por la red.

Dado el caso de que un nodo cliente, contenedor de un número N de archivos, pasase a ser inalcanzable por los otros nodos clientes, ya siendo por una desconexión temporal o por fallo del cliente, debe asegurarse de que dichos archivos siguieran estando disponibles a pesar de esta desconexión del nodo.

Para ello, el nodo central, contiene en su estructura una lista de ficheros compartidos, el número de nodos que comparten ese fichero y sus identificadores. Implementando un sistema de réplica 3:1, en el caso de que el número de nodos clientes fuese superior a tres, existirían tres copias por cada archivo compartido.

El mecanismo de réplica actua en dos escenarios concretos, todo ellos en el suspuesto de que la red estuviese formada por más de tres nodos clientes:

- Dado el caso de que un usuario comparta una canción nueva que no se encontraba anteriormente en el sistema de clasificación de archivos, el nodo Master seleccionaría otros nodos aleatorios para solicitar que creasen una réplica de este nuevo fichero, hasta que hubiese por lo menos 3 nodos clientes que albergaran el nuevo archivo.

- Dado el caso de que un nodo cliente dejase de compartir un fichero de audio, el nodo central recibiría una notificación mediante el protocolo de comunicación(*Apartado 4.2.2*). El nodo central eliminiaría de su estructura de clasificación, la entrada de éste nodo para la lista de nodos compartiendo el fichero anunciado. Automáticamente se realiza una comprobación de si dicha lista contiene menos de tres nodos compartiendo el fichero. Si este número fuese inferior a 3, el nodo seleccionaría de manera aleatoria otros nodos para que creasen una réplica del fichero, hasta que la lista nuevamente tuviese como mínimo, otros tres nodos.

Desde la perspectiva del cliente, el procedimiento es similar al servicio de descarga de ficheros. Consiste en que a partir de la información recibida, el nodo cliente procesaría la carga del paquete anunciando la acción de réplica, para obtener el identificador del fichero. Una vez conocido este identificador, el nodo cliente realizaría los mismos pasos que realizaría en el servicio de descarga(*Apartado 4.1.3*), hasta concluir con la notificación al nodo central de que el fichero se haya descargado correctamente.

4.3. Interfaz gráfica

4.3.1. Peer

Para la implementación de los nodos o Peers, tuvo que tenerse en cuenta las necesidades que presentaba el sistema a desarrollar *Sección* 3.1. Gracias al uso de las librerías de TomP2P, la configuración de los Objetos Peer como tales necesitaban pocos ajustes y las funciones básicas de conexión, desconexión, inserción y recuperación de contenidos otorgaban la libertad de implementar un sistema a un nivel de abstracción superior, usando la red P2P como sistema de transporte.

La interfaz gráfica diseñada para cada usuario está desarrollada a través de las librerías *java.swing.** y *java.awt.** . Esta interfaz permite al usuario interactuar creando un nodo que se conecte a la red mediante un mecanismo de *bootstrapping* y poder realizar acceder a los siguientes servicios:

•Conexión a la red P2P.

•Consulta de archivos compartidos por la red.

•Descarga de ficheros .

•Carga de archivos de archivos locales a la red.

•Reproducción en Streaming de contenidos en la red.

•Reproducción de archivos locales.

En este apartado se comentarán desde un punto de vista general las funciones de la interfaz gráfica(*Apartado* **??**).

Panel de conexión

Como elemento fundamental de la interfaz, se tiene el panel de los parámetros de conexión. Este panel permite al usuario introducir la dirección IP del nodo central para realizar una petición de conexión a la red.

•MasterIP: Dirección IP del nodo central al que se desee conectar. Debe introducirse la dirección en un formato IPv4 (256.256.256.256).

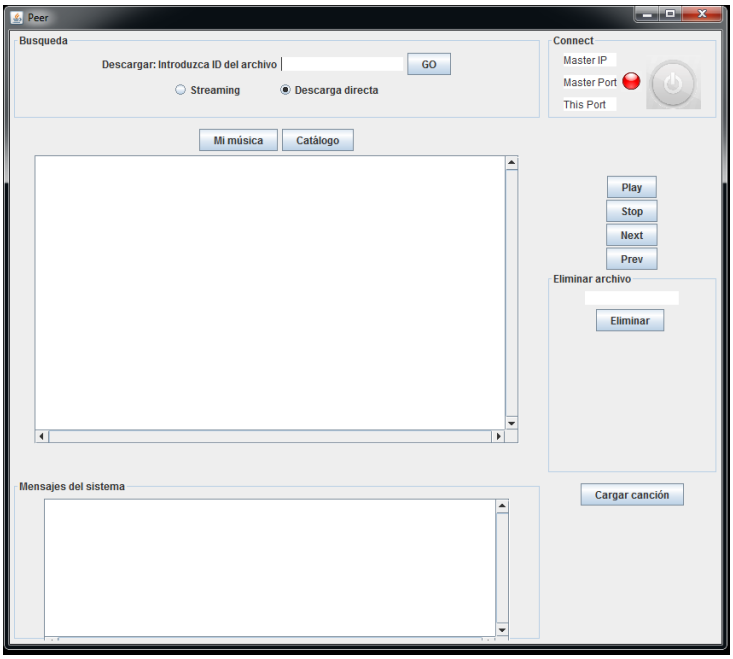

Figura 4.11: Vista previa de el cliente desarrollado en Java

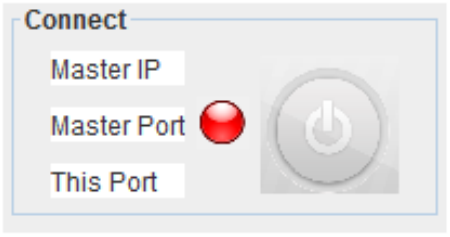

Figura 4.12: Vista previa del panel de parámetros de conexión a la red

•MasterPort: Puerto del nodo central donde se aloja el servidor UDP para el iintercambio de información según el protocolo establecido. Debe introducirse el puerto en un formato numérico (9999).

•This Port: Puerto local al que desee conectarse el usuario. Los nodos hacen uso de 2 puertos, uno para establecer la conexión de su objeto Peer, y otro puerto en el que se establece el cliente UDP para el intercambio de

mensajes con el servidor UDP del nodo central. Éste segundo puerto debe ser siempre el mismo, de manera que el nodo central conozca el puerto de cada uno de los nodos para poder comunicarse con ellos. Por simplicidad, se asigna el mismo puerto a todos los nodos, impidiendo establecer dos nodos dentro de un mismo terminal.

•Indicador lumínico: Este indicador permite al usuario saber si su nodo se encuentra conectado a la red. En el momento que se presione el botón de conexión, se inicia un proceso de intercambio de información entre el nodo central y éste nodo, que culmina con la conexión del nodo a la red P2P, siendo éste el momento en el que el indicador lumínico cambia de color a verde. En el momento en el que se presiona de nuevo el botón de conexión, se realiza el proceso de desconexión de la red, que termina cuando el indicador lumínico cambia de color a rojo.

•Botón de conexión: El usuario podrá conectarse y desconectarse de manera libre a la red mediante la pulsación de este botón. En el momento que dicho botón se presiona, se activa un proceso u otro dependiendo de si el nodo se encuentra conectado a la red. En caso de que el usuario desee conectarse a la red mediante el mecanismo de *bootstrapping (Consulte el apartado 3.1)*, el botón dispara el proceso de petición de parámetros de conexión al nodo central (Consulte el apartado 3.2) y cambia de aspecto, indicando que el proceso se ha realizado con éxito.En caso contrario, si el usuario desea desconectarse, el botón cambiará de aspecto de nuevo después de haber realizado una desconexión segura de la red.

Panel de control de archivos

El usuario debe poder acceder a los contenidos que se encuentren dentro de su sistema local, así como poder consultar todos los archivos que se comparten por la red.

Figura 4.13: Vista previa del contro de archivos

Para ello, el usuario dispone de dos botones que le permiten acceder bien a la información que contiene el cliente sobre los archivos que contie-

ne y comparte, o bien la totalidad de los archivos que se comparten por la red:

•**Mi música:** Despliega en la sección inferior, toda la información útil de todos los ficheros existentes en el sistema local. Permite conocer las canciones que forman parte de la lista de reproducción.

•**Catálogo:** El mecanismo de este botón, lo que permite al usuario es conocer la información de todos los ficheros que se comparten por la red. Para ello, el nodo realiza una petición al nodo central mendiante el protocolo de comunicación (*Consulte el apartado 3.2*) y despliega en la sección inferior toda la información como resultado de la petición anterior.

Panel de carga

De la misma manera que un nodo tiene la capacidad de descargar ficheros, éste cuenta con la capacidad de poner en conjunto y distribuir ficheros que se encuentren dentro del sistema del nodo.

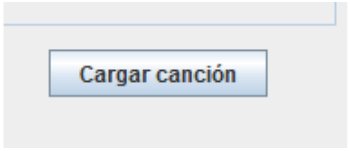

Figura 4.14: Visualizazión del panel de Carga de ficheros

Panel de descarga

El acceso a los servicios de descarga de ficheros y reproducción en streaming pasan por el uso de este segundo panel. En él podemos observar que existe una entrada de texto para introducir el identificante del fichero al que se desee acceder.

Figura 4.15: Visualización del panel de opciones de descarga del Peer

•Entrada de texto: Mediante la introducción del identificador del archivo al que desee accederse, el nodo tranforma este valor de mediante la función hash que emplea el sistema para obtener una clave de 160 bits. Esta clave permite al nodo recuperar el archivo de propiedades de la canción que se desee y realizar una descarga de todos los *chunks* del fichero de audio.

•Botón de "Descarga directa": Al tener selecionado este modo, el nodo realiza una descarga directa desde la red P2P del fichero, contruyendo el fichero completo, integrándolo a la lista de canciones almacenadas por el nodo local, antes de compartir el fichero y sus segmentos.

•Botón de "Streaming": En el momento que el usuario selecione esta opción, el proceso de recuperación del archivo varía del anterior, iniciando un proceso de reproducción en streaming del contenido seleccionado.

•Botón "GO": Inicia el proceso de descarga o reproducción en Streaming a partir del identificador introducido en la entrada de texto.

Panel de reproducción de archivos

Como se puede observar, el cliente del nodo cuenta con una sencilla implementación de un reproductor mp3 que se controla mediante los botones que se hallan en el panel.

Figura 4.16: Visualización del panel de reproducción de archivos

•**Play:** Cuando se acciona este botón, el reproductor integrado crea una lista aleatoria con los archivos de audio que se encuentran descargados dentro del nodo. Un vez creada, el cliente comienza a reproducir cada uno de los elementos de la lista de reproducción generada.

•**Stop:** Detiene la lista de reproducción totalmente, el reproductor reiniciará el proceso de creación de una lista aleatoria en el momento que se

accione el boton *PLAY*.

•**Prev:** Selección de la pista reproducida anteriormente dentro de la lista de reproducción aleatoria generada.

•**Next:** Selección de la pista a continuación dentro de la lista de reproducción aleatoria generada.

Panel de actividad

La actividad que vaya generando el nodo se irá registrando en el siguiente panel, donde el usuario podrá conocer más en detalle los parámetros de los procesos que ocurren cuando se activan los diferentes servicios del nodo.

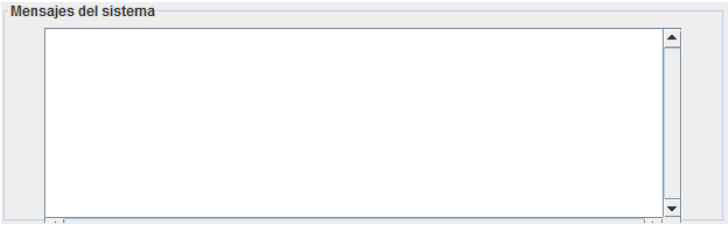

Figura 4.17: Visualización del panel de actividad del nodo

A medida que un usuario va interactuando con la red, accediendo a los diferentes servicios que proporcionan los otros nodos, se van añadiendo anotaciones a este panel. Proporciona un grado de detalle mayor que el que se observa en el resto de paneles.

Este ejemplo muestra la actividad de un usuario que se ha conectado y ha compartido un fichero con la red:

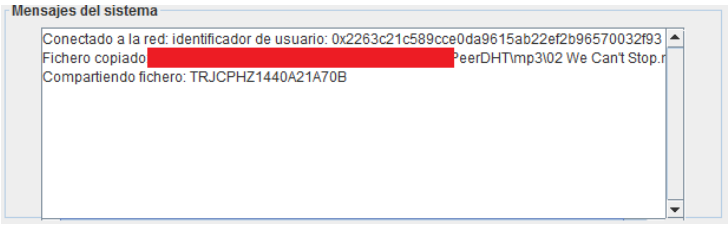

Figura 4.18: Ejemplo 2

Panel de eliminación

En el momento que un usuario desee dejar de compartir y eliminar un fichero de su sistema local, se elimina el fichero deseado y los segmentos del mismo a través de este panel.

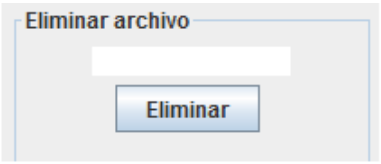

Figura 4.19: Visualización del panel de eliminación de contenidos a nivel local

Para ello, el usuario debe introducir en la entrada de texto del panel, el identificador del archivo que desee eliminar. En el momento que se accione el botón, el nodo eliminará el fichero de sus dependecias, junto con los *chunks* del mismo, de manera que deje de compartir el fichero con el resto de la red.

Posteriormente, el protocolo de comunicación notifica al nodo central los cambios dentro de este nodo(*Apartado 4.2*) para evitar que otros nodos tengan problemas intentando acceder a archivos que ya no se compartan por la red.

4.3.2. Master

Como se explicó anteriormente (*Apartado 3.1*), el nodo de control o Master, no se diseña para que un usuario haga uso de él como si fuese un nodo cliente más con funciones addicionales, sino que procura diseñarse de manera que la interacción con el usuario sea méramente de consulta.

Se limita el acceso mediante la interfaz ya que no se desea alterar o modificar la información que manipula el nodo de control, sino que simplemente se puedan visualizar sus contenidos de la misma manera que realizan los nodos a través del protocolo de comunicación(*Apartado 4.2*).

Como puede observarse, en el único panel de la interfaz, existen una serie de botones que proporcionan información en base a las posibles consultas que puede realizar un nodo cliente:

•**Ver Arbol:** Esta petición muestra la información que puede apreciarse según la distribución de los artistas, reflejando datos como el identificador

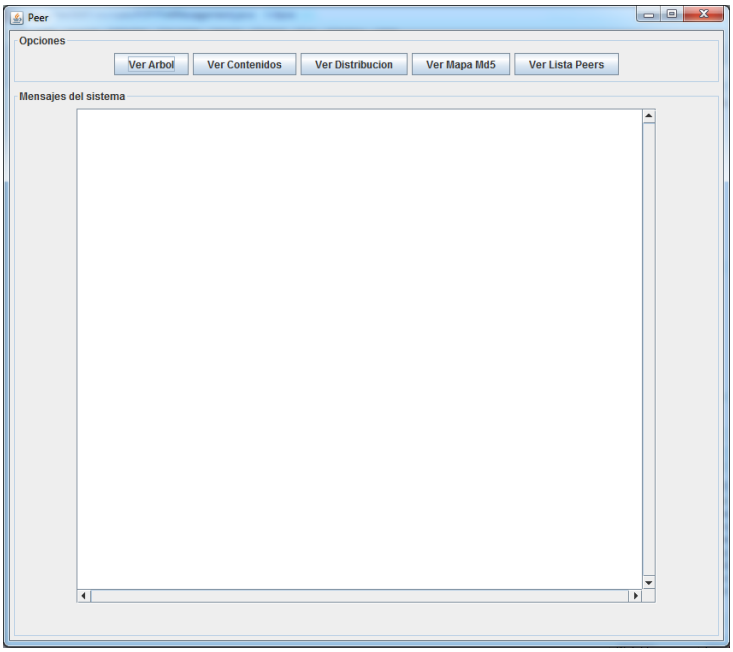

Figura 4.20: Vista previa de la interfaz del nodo Master

de cada artista, el número de álbumes pertenecientes a este artista y el número de canciones.

Figura 4.21: Master:Ver Arbol

•**Ver contenidos:** En el caso que un nodo cliente realizase este petición al nodo central, éste unicamente devolvería los identificadores de cada fichero de audio distribuido por la red para que el cliente descargase todos ellos.

•**Ver distribución:** En el caso de que el usuario desee conocer la distribución de los ficheros por todo la red, esta petición mostraría cada identificante de cada nodo cliente que albergase el fichero, mostrando una lista de nodos cliente por cda fichero compartido.

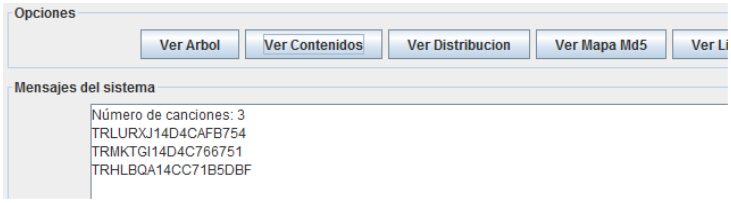

Figura 4.22: Master: Ver Contenidos

Figura 4.23: Master: Ver Distribucion

•**Ver Mapa Md5:** Como ya se comentó en anteriormente(*Apartado 3.1*), existe una segunda estructura secundaria y complementaria a la estructura principal que sirve como apoyo en caso de fallos de clasificación. Mediante esta petición, el usuario puede consultar los ficheros de Metadatos Md5Properties(*Apartado 3.2.1*) que se emplean en este mapa.

•**Ver Lista Peers:** Ya que el nodo central tiene constancia de la topología global del sistema, el usuario puede consultar el identificador de cada uno de los nodos clientes activos dentro de la red.

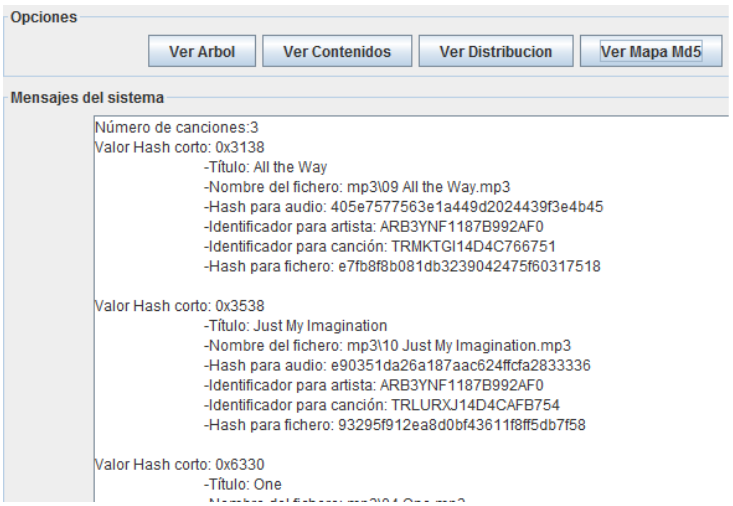

Figura 4.24: Master: Ver Md5

Figura 4.25: Master: Ver Lista Peers

Capítulo 5

Pruebas

Una vez finalizada la implementación inicial del sistema, los clientes desarrollados y el nodo central fueron puestos a prueba. La evaluación de este sistema se realizó de una manera progresiva, partiendo desde la ejecución de ambas entidades en una sola máquina, hasta poder crear una red con distintas conectas tanto en LAN como sobre Internet.

5.1. Plataforma

El sistema se desarrolló para ejecutarse en ordenadores personales, independientemente del sistema operativo, siempre que éste último dispusiera de un conpilador de Java con la versión adecuada. Este sistema ha sido probado en máquinas usando Windows 7, Linux Debian y MacOS.

Para poder evaluar los resultados de las pruebas, cada uno de las entidades, tanto nodos clientes como nodo Master, crearon un registro de todas las actividades, intercambios de mensajes y errores detectados durante las pruebas. Los resultados expuestos en el apartado 5.3 son el resultado de cálculos realizados a partir de estos datos.

5.2. Simulación del proyecto

La simulación de este proyecto se realizó en distintas fases, probando distintos aspectos desde que se desarrolló un prototipo funcional de las entidades, evolucionando en consecuencia a medida que se fueron corrigiendo los errores encontrados. Este procedimiento permitío corregir distintas

facetas del planteamiento inicial de los servicios y estructuras, optimizando las entidades y la red entera.

Una vez que se desarrolló un primer cliente no experimental y capaz de ser sometido a distintas pruebas de control, se realizó una evaluación progresiva a medida que se observaban con detalle las respuestas de las diferentes acciones y servicios que empleaba el cliente.

Las pruebas que se realizaron con este primer modelo de sistema presentaron una serie de errores que se comentarán en el apartado 5.3, analizando su origen y las consecuencias de los mismos y comentando la corrección de éstos.

5.2.1. Plan de pruebas

El plan de pruebas se realizó de una manera progresiva a medida que se evaluaban las respuestas de los clientes en entornos diferentes, añadiendo siempre un grado de dificultad cada vez superior. El objetivo de aumentar el nivel de dificultad exponencialmente era evaluar la respuesta del sistema en caso de que escalara de una manera muy rápida si se implementase una red real. Se realizaron un total de cuatro pruebas independientes:

•**Prueba I:** Puesta a prueba del nodo Master y del nodo cliente en una sola máquina. Siendo esta la primera prueba, el objetivo era evaluar los servicios básicos del cliente, uno a uno y comprobar que el protocolo de comunicación establecido entre nodo cliente y nodo Master funcionase correctamente.

Se iniciaron dos nodos clientes y un nodo Master en una sola máquina, cada uno con sus carpetas personales correspondientes para probar el intercambio de ficheros entre ambos nodos clientes, y asegurar que cada uno de ellos escribía en un registro diferente (las funciones de escritura al registro apuntaban a un mismo documento, por tanto era necesario separar ambos clientes). La prueba evaluó los aspectos de la tabla 5.1.

•**Prueba II:** Puesta a prueba del nodo Master y del nodo cliente en distintas máquinas a través de una red LAN privada. Para poder simular de la manera más sencilla posible, era necesario que los distintos nodos pudieran formar una red Peer-to-peer desde distintas máquinas. Por tanto, haciendo uso de máquinas distintas, se establecieron dos nodos clientes y un nodo de control. Los nodos clientes deberían conectarse al nodo Master

Servicio	Resultado esperado	Evaluación
Conexión	Poder conectarse a la red mediante el método de bootstrapping al nodo central.	Satisfactorio
Desconexión	Desconexión segura de la red.	Satisfactorio
Carga	Capacidad de carga de un fichero de audio	Satisfactorio
Reconocimiento	Capacidad de identificación en red de una canción.	Satisfactorio
Descarga	Capacidad de descarga de ficheros contenidos en la red.	Satisfactorio
Eliminación de contenidos	Capacidad de eliminicación de contenidos	Satisfactorio

Cuadro 5.1: Plan de pruebas I

usando la dirección IP privada dentro de la LAN.

Al no tratarse de un entorno totalmente cerrado y dentro de una misma máquina, los resultados mostraron un retraso casi despreciable (frente al tiempo requerido por las operaciones de cómputo) en las operaciones de intercambio de ficheros, ya que la tasa binaria que ofrecia la red LAN, reducía mínimamente el retraso sufrido por el intercambio de paquetes. Esta prueba evaluaba los mismos parámetros que en la prueba anterior, pero en un escenario más exigente. Ver tabla 5.2

- Resultado de prueba de servicio de Reconocimiento*: Se produjo un error de reconocimiento debido al servicio externo en el primer intento, véase el apartado de resultados 5.3 para más información.

•**Prueba III:** Puesta a prueba del nodo Master y del nodo cliente en catorce máquinas distintas dentro de una red LAN privada. Haciendo uso de las áulas de telemática de la Universidad Carlos III de Madrid, se quiso someter al nodo central a un test de éstres, observando como trabaja con un número muy superior de nodos clientes con respecto a las pruebas anteriores.

Para esta prueba se pudieron disponer de catorce máquinas, conectadas

Servicio	Resultado esperado	Evaluación
Conexión	Poder conectarse a la red mediante el método de bootstrapping al nodo central.	Satisfactorio
Desconexión	Desconexión segura de la red.	Satisfactorio
Carga	Capacidad de carga de un fichero de audio	Satisfactorio
Reconocimiento	Capacidad de identificación en red de una canción.	Satisfactorio*
Descarga	Capacidad de descarga de ficheros contenidos en la red.	Satisfactorio
Eliminación de contenidos	Capacidad de eliminicación de contenidos	Satisfactorio

Cuadro 5.2: Plan de pruebas II

entre sí en una LAN, propia de la universidad. Aún pudiendo ejecutar distintos clientes en una sola máquina, pudiendo añadir más nodos a la red, el objetivo de la prueba era determinar si el nodo central sería capaz de manejar las peticiones de un número, en principio diez veces superior, al utilizado en la prueba anterior.

Queriendo hacer un uso razonable de todos los nodos clientes de manera simultánea, se solicitó ayuda a personas de la titulación de Grado en Ingeniería de las Tecnologías de la Telecomunicación para poder evaluar con criterio los servicios de los clientes. Los aspectos que se evaluaron en cada uno de los clientes fueron se muestran en la tabla 5.3

•**Prueba IV:** Puesta a prueba del nodo Master y del nodo cliente en quince máquinas distintas usando la red pública. Haciendo uso de las mismas máquinas de la Universidad Carlos III de Madrid y un ordenador conectado a Internet a través de una red distinta, se ejecutaron quince nodos clientes en las máquinas disponibles. El nodo central se estableció en una máquina ajena a esta red, usando una dirección IP pública distinta y una velocidad de conexión más limitada que en la prueba anterior. Los objetivos principales de la prueba eran dos.

Servicio	Resultado esperado	Evaluación
Conexión	Poder conectarse a la red mediante el método de bootstrapping al nodo central.	Satisfactorio
Desconexión	Desconexión segura de la red.	Satisfactorio
Carga	Capacidad de carga de un fichero de audio	Satisfactorio
Reconocimiento	Capacidad de identificación en red de una canción.	Insatisfactorio, el servicio de reconocimiento en red no permite la identificación de varios archivos de manera simultánea con una sola clave de desarrollador.
Descarga	Capacidad de descarga de ficheros contenidos en la red.	Satisfactorio
Eliminación de contenidos	Capacidad de eliminicación de contenidos	Satisfactorio
Replica automática	Sistema automatizado de réplica en una red con más de tres nodos	Satisfactorio, aunque la elección de nodos parece determinista en vez de una elección aleatoria
Reproducción en Streaming	Reproducción de contenidos compartidos por la red	Satisfactorio

Cuadro 5.3: Plan de pruebas III

En primer lugar, establecer el nodo central en una red local distinta a la de las máquinas contenedoras de los nodos clientes para probar el intercambio de mensajes a través de la red pública. Para ello, los parámetros de conexión de la red deberían incluir la dirección pública y privada del

nodo central. Para ello, tuvo que modificarse la interfaz gráfica del nodo cliente para poder admitir una dirección IP privada y otra pública.

En segundo lugar, el uso de una red distinta con una conexión de banda ancha limitada pondría a prueba la capacidad de conexión de un nodo. Pudiendo probar cualquiera de las dos entidades en estas circunstancias, se optó por usar la entidad máster ya que éste recibe numerosas peticiones a través de paquetes pequeños, con lo cual no debería suponer un inconveniente muy grande para el nodo central no disponer de conexión de banda ancha.

Los aspectos a evaluar en esta última prueba fueron los que se incluyen en la tabla 5.4.

5.2.2. Criterios de aceptación

5.3. Resultados

Como 1 página indicando el contenido de los siguientes capítulos y apéndices.

Servicio	Resultado esperado	Evaluación
Conexión	Poder conectarse a la red mediante el método de bootstrapping al nodo central.	Satisfactorio
Desconexión	Desconexión segura de la red.	Satisfactorio
Carga	Capacidad de carga de un fichero de audio	Satisfactorio
Reconocimiento	Capacidad de identificación en red de una canción.	Insatisfactorio, el servicio de reconocimiento en red no permite la identificación de varios archivos de manera simultánea con una sola clave de desarrollador.
Descarga	Capacidad de descarga de ficheros contenidos en la red.	Satisfactorio
Eliminación de contenidos	Capacidad de eliminicación de contenidos	Satisfactorio
Replica automática	Sistema automatizado de réplica en una red con más de tres nodos	Satisfactorio, aunque la elección de nodos parece determinista en vez de una elección aleatoria
Reproducción en Streaming	Reproducción de contenidos compartidos por la red	Satisfactorio
Respuesta del nodo central	Capacidad de control de la red inalterada, sin diferencias con respecto a otras pruebas	Satisfactorio, aunque las operaciones de envio de listas funcionaba con mayor lentitud

Cuadro 5.4: Plan de pruebas IV

Capítulo 6

Historia del proyecto

Esta parte suele ser confidencial en los proyectos reales; pero es muy importante en proyectos académicos pues permite reflexionar sobre el trabajo realizado y aprender para el futuro. Recoge el ritmo de realización del proyecto: sus alegrias y sus fracasos. Conviene ser muy realista en la descripción de los problemas, las soluciones adoptadas, o los replanteamientos a que hubo que someterse. Debe ir acompañado de estimaciones de coste realistas.

Importante indicar el tiempo dedicado a cada tarea.

Capítulo 7

Conclusiones y trabajos futuros

Básicamente hay que repasar los objetivos propuestos y comentar su grado de satisfacción. Los comentarios iniciales deben ser los positivos.

Hay que destacar los aspectos originales o más creativos.

Dado el caracter académico del proyecto (punto que no se aplicaría en un proyecto real, probablemente), cabe reseñar los aspectos negativos: objetivos incumplidos o dificultades inesperadas o subvaloradas en el anterproyecto.

Por último cabe una parte dedicada a futuras prolongaciones del proyecto: "¿qué más haría si hubiera más tiempo?"

Bibliografía

Recomendamos utilizar bibtex.