INSTITUTO POLITÉCNICO NACIONAL.

UNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERIA Y TECNOLOGÍAS AVANZADAS.

Unidad de Aprendizaje: Sistemas Distribuidos.

Integrantes:

Mejía Hernández Karla Montserrat.

Pedroza Velarde Luis Rodrigo.

Grupo: 2TV7.

Profesor: Miguel Félix Mata Rivera.

PROYECTO FINAL.

Fecha de Entrega: 12 junio de 2023.

ÍNDICE

INTRODUCCIÓN

¿Qué Es BitTorrent?

BitTorrent es un protocolo de transferencia de Internet. De forma similar al HTTP (Protocolo de transferencia de hipertexto) y al FTP (Protocolo de transferencia de archivos), la tecnología BitTorrent es una manera de descargar archivos de Internet. Sin embargo, a diferencia de los protocolos http y ftp, BitTorrent es un protocolo de transferencia distribuido.

El protocolo punto a punto (P2P) de BitTorrent busca y descarga fragmentos específicos de archivos entre usuarios de forma simultánea. Por lo tanto, las velocidades de transmisión son superiores a las de los protocolos http y ftp, que solo ofrecen descargas de archivos de forma secuencial desde un único origen

Es un tipo de archivo con extensión .torrent que almacena los datos necesarios para que una aplicación de BitTorrent comparta el contenido. Su tamaño es muy pequeño (entre 10 y 200 KB) porque no incluye el contenido en sí sino **información sobre los archivos y carpetas** de éste, incluyendo la localización de los diferentes «trozos» del archivo de destino en las que será dividido, primero en piezas y luego en bloques.

¿Cómo funciona BitTorrent?

A menudo se define la red BitTorrent como un enjambre de abejas porque su funcionamiento es similar. Así, en lugar de descargar un archivo desde un único servidor el protocolo permite unirse a un swarn (enjambre) donde miles de usuarios descargan y suben simultáneamente trozos de un archivo hasta completar su contenido.

A nivel de usuario su funcionamiento es muy simple. Pongamos como ejemplo a un proveedor como Canonical que pretende distribuir por BitTorrent una de sus distribuciones GNU/Linux. La compañía crea un archivo .torrent y lo publicita en su página web, por correo electrónico u otro medio. El usuario descargar el archivo y lo ejecuta en una aplicación BitTorrent que se encargará de su manejo, descarga y compartición entre múltiples usuarios hasta completar en sus equipos la descarga de una imagen .ISO de Ubuntu.

Bajo este sistema, Canonical ahorra una gran cantidad de infraestructura en servidores dedicados para descarga directa evitando colapsos. En muy poco tiempo, la imagen de Ubuntu (recordemos «troceada» en BitTorrent) se habrá extendido lo suficiente entre miles de usuarios y serán ellos los que contribuirán a su distribución al mismo tiempo que la descargan. Si los usuarios son lo suficientemente «honestos» (como luego veremos) la velocidad de descarga de un archivo Torrent masivo no envidiará a la obtenida por una descarga directa.

JAVA RMI

Se utilizo BitTorrent y para su ejecución de la transferencia de Archivos, se optó por utilizar RMI que es ejecutado en Java.

La invocación remota de métodos (Remote Method Invocation) de Java es un modelo de objetos distribuidos, diseñado específicamente para ese lenguaje, por lo que mantiene la semántica de su modelo de objetos locales, facilitando de esta manera la implantación y el uso de objetos distribuidos.

En el modelo de objetos distribuidos de Java, un objeto remoto es aquel cuyos métodos pueden ser invocados por objetos que se encuentran en una máquina virtual (MV) diferente.

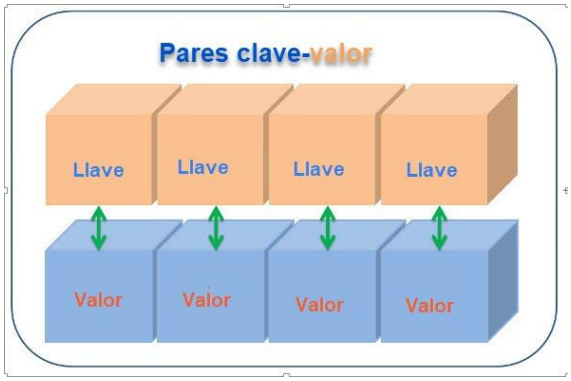
Los objetos de este tipo se describen por una o más interfaces remotas que contienen la definición de los métodos del objeto que es posible invocar remotamente.

Se trata de una implementación independiente de la plataforma, lo que permite que tanto los objetos remotos como las aplicaciones cliente, residan en sistemas heterogéneos. Sin embargo, no es independiente del lenguaje, tanto el objeto servidor Java/RMI como el objeto cliente tienen que ser escritos en Java

OBJETOS HASHMAP

En Java se usa HashMap para almacenar elementos en pares clave/valor y puedes acceder a estos elementos almacenados en un HashMap utilizando la clave del elemento, que es única para cada elemento.

- Los elementos se almacenan en pares clave/valor.Los elementos no mantienen ningún orden cuando se agregan.
- Los datos están desordenados.
- En caso de que haya claves duplicadas, la última anulará a las otras.
- Los tipos de datos se especifican utilizando clases contenedoras en lugar de tipos de datos primitivos.



P2P

Una red P2P es un sistema distribuido que funciona en los ordenadores de los usuarios conectados a ella. Esto significa que son los propios usuarios los que se encargan de almacenar y procesar los datos en lugar de un servidor central. La infraestructura de las redes P2P se basa en esta estructura descentralizada, que las hace rentables y relativamente seguras contra los ciberataques.

Las redes peer to Peer, se construyen con base en protocolos de comunicación creados para funcionar sobre los mismos protocolos de internet (TCP/IP). Estos protocolos permiten a las personas comunicarse de forma directa y sin intermediarios con otros. Es por esta razón, que a estos protocolos se les denomina también, como Layer 7 o protocolos de aplicación.

No necesita ningún servidor central o tercero que coordine la comunicación entre las personas que utilizan el servicio (los usuarios están directamente conectados entre sí). Esto significa que el coste de ejecución de las aplicaciones peer-to-peer tiende a ser menor en comparación con las soluciones de software tradicionales.

TRACKER

Un tracker o rastreador de BitTorrent es un servidor especial que contiene la información necesaria para que los peers se conecten con otros peers asistiendo la comunicación entre ellos usando el protocolo BitTorrent.

Los rastreadores no sólo coordinan la comunicación y distribución de datos entre los peers que tratan descargar el contenido referido por los torrents, sino que también siguen de cerca y sin perder de vista las estadísticas y la información de verificación para cada torrent. Si el rastreador se viene abajo y alguien intenta comenzar una descarga de un torrent, no podrá conectarse con el enjambre de usuarios.

El tracker es el único que sabe dónde se localiza cada peer dentro de un enjambre, por lo que es indispensable su disponibilidad para poder comunicarse con el resto de los usuarios, por lo menos hasta haberse conectado con el enjambre.

Los rastreadores se dividen en dos clases, privados y públicos. La principal diferencia entre ellos es que los privados requieren que los peers sean usuarios registrados de un sitio web, mientras que en los rastreadores públicos cualquiera puede comunicarse con ellos. Los rastreadores privados generalmente guardan las estadísticas de tráfico de cada usuario y utilizan un sistema de porcentajes que permite saber si el usuario comparte o no los datos que haya descargado o esté descargando.

TABLA.

La descripción de los códigos utilizados se encuentra a continuación:

ID	Descripción.
Codigo 1.1 **Archivo: ServerImpl.java** **Ubicado en carpeta Server** **Enlace de drive:** https://drive.google.com/file/d/1EcheujPhmXSdhR8yZulTmBBuFqiCVjE6/view?usp=drive_link	El código proporcionado es una implementación de un servidor RMI (Remote Method Invocation) en Java. El servidor implementa la interfaz ServerI y extiende la clase UnicastRemoteObject, lo que permite que los métodos del servidor sean invocados de forma remota por clientes que se conecten a través de RMI.
Codigo 1.2 **Archivo: HiloS.java** **Ubicado en carpeta Server** **Enlace de drive:** https://drive.google.com/file/d/1Z3dXo85cBsJaEPi_v_1FKWj9Du_me36w/view?usp=drive_link	Define una clase llamada HiloS que se utiliza para realizar una verificación periódica de los pares conectados en el servidor y eliminar aquellos que no están activos.
Codigo 1.3 **Archivo: Torrent.java** **Ubicado en carpeta Server**	La clase Torrent implementa las interfaces Serializable y Remote, lo que indica que los objetos de esta clase

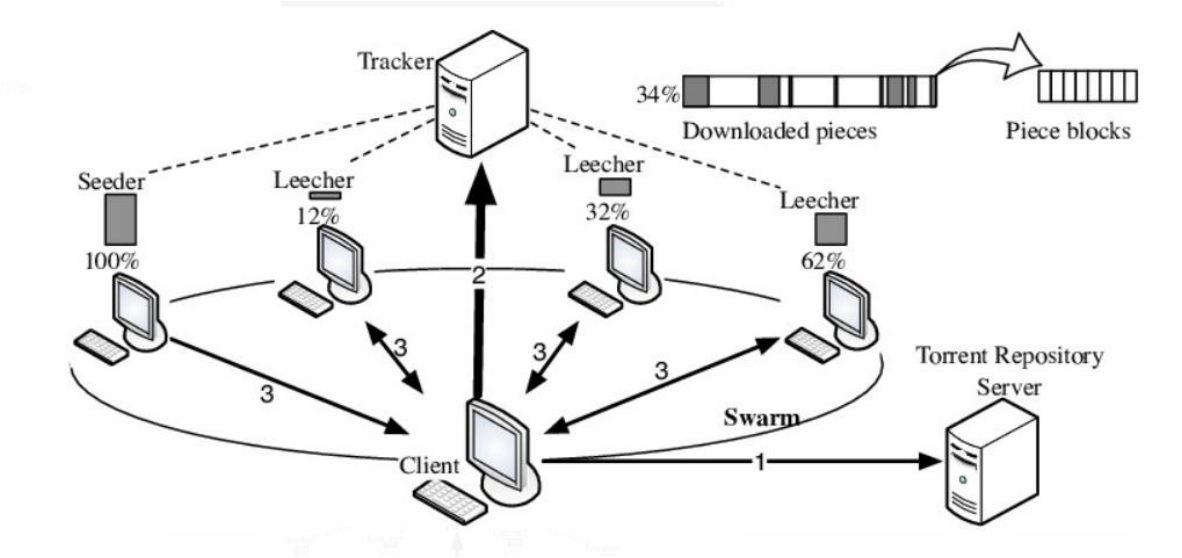
Enlace de drive: https://drive.google.com/file/d/1VENVGY2tDwofoVjgbyPuJE8cqUp1-N4u/view?usp=drive_link	pueden ser serializados para su transferencia y utilizados en invocaciones remotas. Se utiliza para representar la información de un archivo torrent, incluyendo nombre, tamaño, número de piezas y los seeders y leechers asociados.
Código 1.4 **Archivo: PeerData.java** **Ubicado en carpeta Server** **Enlace de drive:** https://drive.google.com/file/d/1nzw2dyf4VOKzVfHFd47eFRzhuSXVfTCz/view?usp=drive_link	La clase PeerData se utiliza para representar la información de un par en un sistema peer-to-peer. Almacena el identificador del par, el número de puerto y la dirección del host. Proporciona métodos para comparar la igualdad de objetos PeerData y obtener una representación de cadena de un objeto PeerData.
Código 1.5 **Archivo: p2pServer.java** **Ubicado en carpeta Server** **Enlace de drive:** **https://drive.google.com/file/d/11UXT_4iJMQhlz_ihtk7BCXUNIY1gp1Yf/view?usp=drive_link**	El código representa un servidor peer-to-peer que utiliza RMI para permitir la comunicación entre pares en la red. El servidor se enlaza en el registro RMI en el host y puerto especificados, y se ejecuta en un bucle hasta que se ingresa un valor específico para salir.
Código 1.6 **Archivo: ServerI.java** **Ubicado en carpeta Server** **Enlace de drive:** https://drive.google.com/file/d/1_yqtm9qZkE5EQhu3MaUEGub_1WRtWolD/view?usp=drive_link	Interfaz ServerI que define los métodos que pueden ser invocados en el servidor del sistema peer-to-peer.
Archivo 1.1 **Archivo: server.policy** **Ubicado en carpeta Server** **Enlace de drive:** https://drive.google.com/file/d/1TjR1olkKfLULngxvVAVe4NVp-TCU_x4f/view?usp=drive_link	Este archivo de política de seguridad permite que cualquier código que se ejecute desde el directorio actual tenga acceso a todos los permisos de seguridad en el entorno de ejecución Java.
Archivo 1.2 **Archivo: client.policy** **Ubicado en carpeta Client** **Enlace de Drive:** https://drive.google.com/file/d/1Dt4pCHONgB8ZYi87FU99NDlrcvHDUaPJ/view?usp=drive_link	Esta declaración de política de seguridad permite que cualquier código ubicado en el directorio actual tenga acceso completo a todos los recursos y acciones del sistema sin restricciones de seguridad.
Código 2.1 **Archivo: ClientI.java** **Ubicado en carpeta Client** **Enlace de Drive:**	La interfaz ClientI define los métodos que un objeto cliente puede invocar de forma remota. Estos métodos permiten obtener piezas de los archivos, verificar la

https://drive.google.com/file/d/1iMqRjs39u48btzkhQFGzm-6_sjp9zuFV/view?usp=drive_link	conexión, obtener información de progreso y obtener identificadores de cliente.
Código 2.2 **Archivo: ClientImpl.java** **Ubicado en carpeta Client** **Enlace de Drive:** **https://drive.google.com/file/d/1xgeErjK6M9bEk1or6f21xOp61-FO9Cl7/view?usp=drive_link**	El código implementa la lógica de un cliente P2P que permite a los usuarios buscar y descargar archivos compartidos por otros peers en la red. Aquí es donde los archivos se dividen en pequeños pedazos para después ser compartidos y hacer la transferencia más simple.
Código 2.3 **Archivo: CryptoException.java** **Ubicado en carpeta Client** **Enlace de Drive:** **https://drive.google.com/file/d/1loRT5jWO9fxZZiiy6gJlRqmVt3e1HjXN/view?usp=drive_link**	Proporciona una forma de lanzar excepciones personalizadas relacionadas con operaciones criptográficas.
Código 2.4 **Archivo: CryptoUtils.java** **Ubicado en carpeta Client** **Enlace de Drive:** **https://drive.google.com/file/d/1DESJB5J_9YUbds05KaBTWd2-C7ez0Y-T/view?usp=drive_link**	Proporciona métodos para cifrar y descifrar datos utilizando el algoritmo de cifrado AES.
Código 2.5 **Archivo: Fragmentacion.java** **Ubicado en carpeta Client** **Enlace de Drive:** **https://drive.google.com/file/d/1ETFZm7Tjl7l6MyoNv4mTdv6-rz1AGOnG/view?usp=drive_link**	Proporciona métodos para fragmentar y unir archivos en varias partes.
Código 2.6 **Archivo: HiloP.java** **Ubicado en carpeta Client** **Enlace de Drive:** **https://drive.google.com/file/d/1rrzOYKsxvK3ttEJUS6k03gsJBBZL-AvT/view?usp=drive_link**	Representa la lógica para descargar archivos de un peer remoto en una red P2P, utilizando un hilo de ejecución para realizar la descarga de forma concurrente y manejar posibles errores o interrupciones durante el proceso.
Código 2.7 **Archivo: HiloV.java** **Ubicado en carpeta Client** **Enlace de Drive:** **https://drive.google.com/file/d/1bzur_nBWovvyCQW-S7PzrAhL60fyAFho/view?usp=drive_link**	Actualiza la barra de descarga, durante la ejecución de un hilo.

Código 2.8 **Archivo: p2pClient.java** **Ubicado en carpeta Client** **Enlace de Drive:** https://drive.google.com/file/d/1_HmdceJ3c6k3g7izlwS6gQXkY729l9rF/view?usp=drive_link	Describe la conexión del cliente (Peer) con el server.
Código 2.9 **Archivo: progreso.java** **Ubicado en carpeta Client** **Enlace de Drive:** https://drive.google.com/file/d/1svkcjdvyml_un1tiYHlNaFkc6qXfjsun/view?usp=drive_link	Verifica el proceso de la descarga en cada uno de los clientes. Proporciona métodos para obtener y establecer información sobre las piezas descargadas, calcular el progreso en términos de porcentaje y verificar si el archivo se ha descargado por completo.
Código 2.10 **Archivo: prueba.java** **Ubicado en carpeta Client** **Enlace de Drive:** https://drive.google.com/file/d/1_lRXVsqmedgQMSQJFYclhzESxceCYy26/view?usp=drive_link	Se utiliza para probar la conexión y el acceso a un objeto remoto registrado.
Código 2.11 **Archivo: VentanaD.java** **Ubicado en carpeta Client** **Enlace de Drive:** **https://drive.google.com/file/d/1T7DCtMyJqzNg1RJhJH07NUkYpKE1dsYa/view?usp=drive_link**	Despliega la ventana de progreso utilizando un JFrame para que se marque también cuando el archivo se descarga por completo.

DESARROLLO

Diagrama o descripción de la RED



Descripción Del Funcionamiento

Primero inicializamos el Servidor con el siguiente código dentro de nuestra terminal:

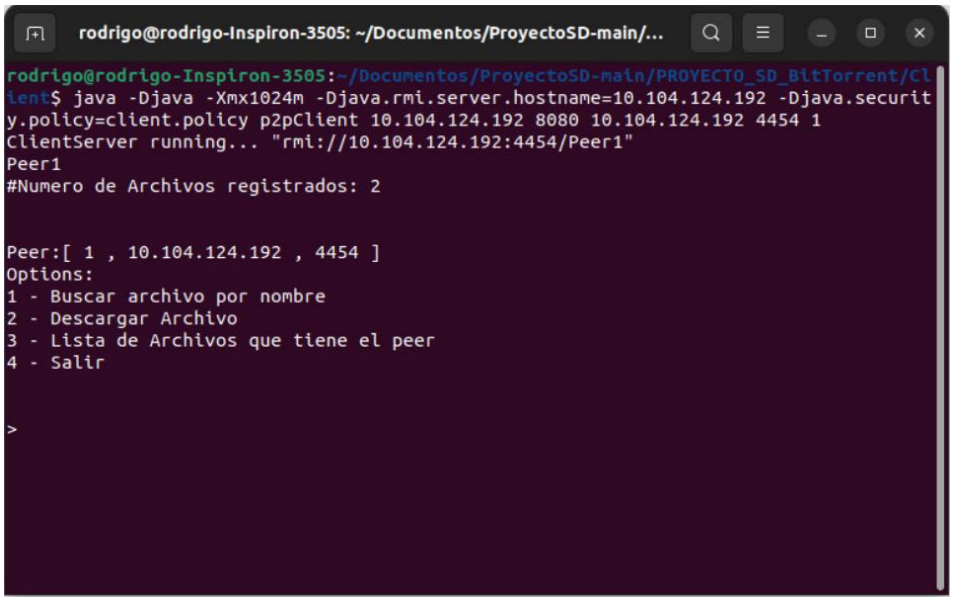
```
java -Djava.rmi.server.hostname=192.168.0.5 -
Djava.security.policy=server.policy p2pServer 192.168.0.5 8080
```

```
rodrigo@rodrigo-Inspiron-3505:~/Documentos/ProyectoSD-main/PROYECTO_SD_BitTo
rrent/Server$ java -Djava.rmi.server.hostname=10.104.124.192 -Djava.security
.policy=server.policy p2pServer 10.104.124.192 8080
Server "rmi://10.104.124.192:8080/server" running...
Enter 0 to exit:
```

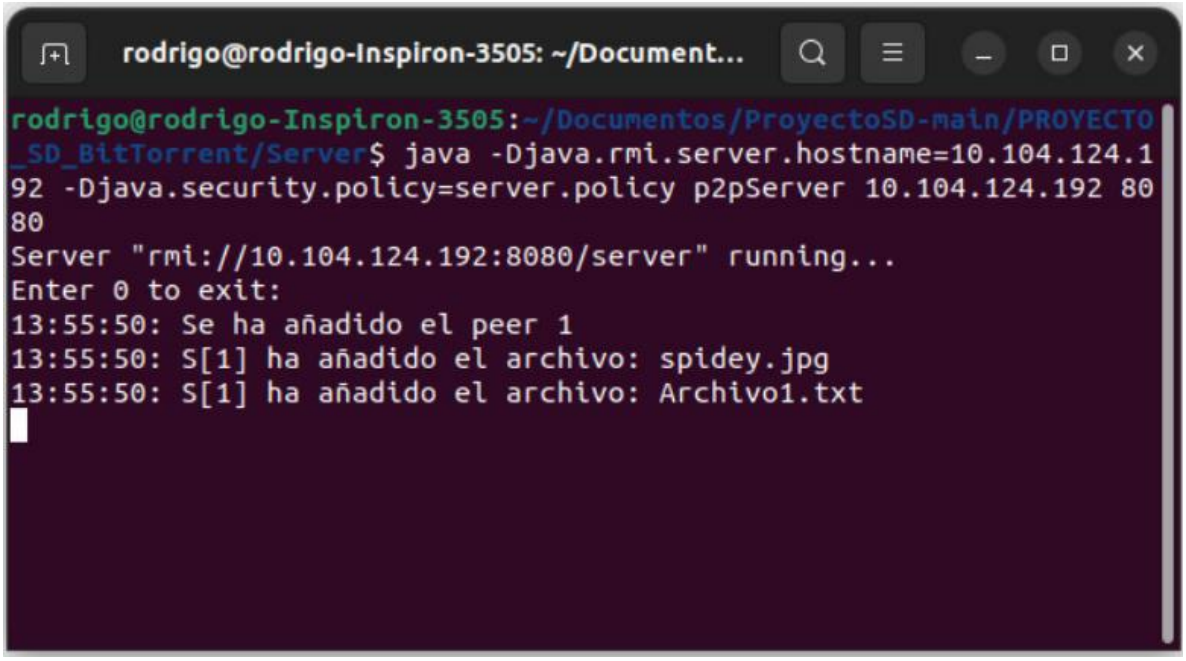
Después inicializamos el cliente:

```
java -Djava -Xmx1024m -Djava.rmi.server.codebase=file:. -
Djava.rmi.server.hostname=192.168.0.5 -
Djava.security.policy=client.policy p2pClient 192.168.0.5 8080
192.168.0.5 4454 1
```

Este ultimo número, nos va a indicar el numero de Cliente que se conectó, o el número de peer, en este caso es el numero uno, pero es posible que se conecten hasta 4 peers

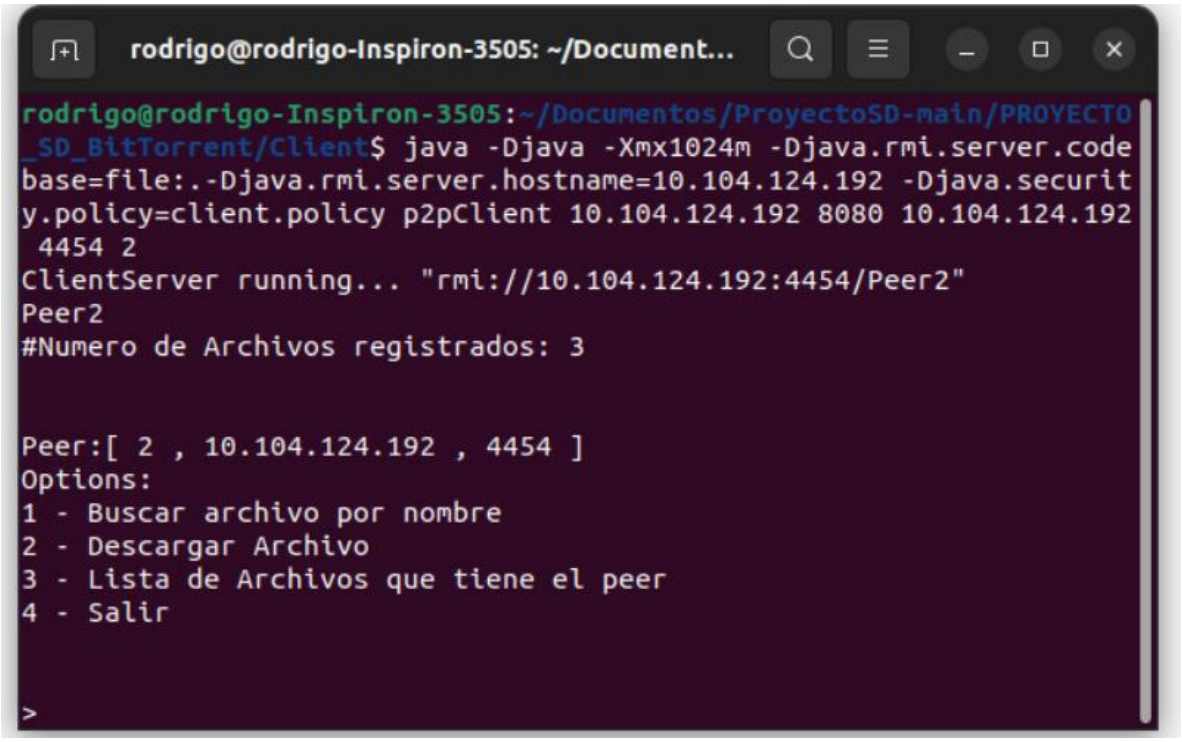


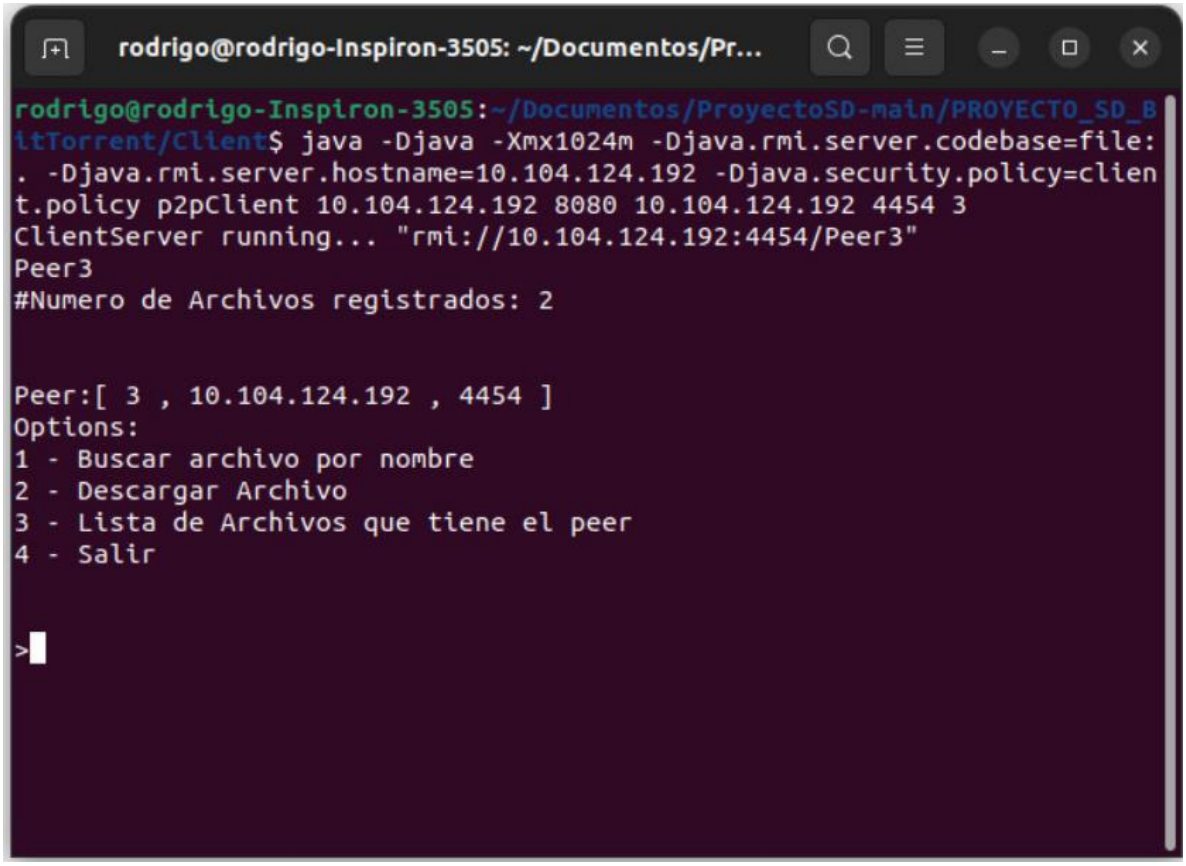
Como podemos ver, se nos muestra un menú en el que como cliente vamos a poder decidir que es lo que queremos ejecutar con nuestros archivos



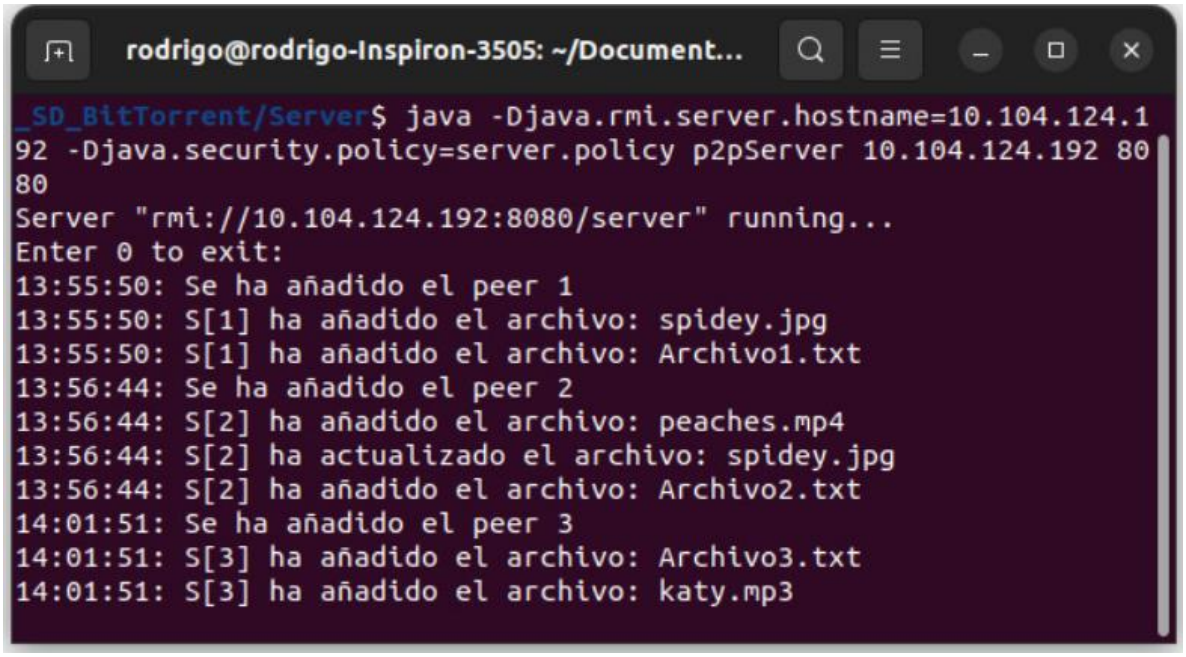
Una vez que el cliente se ha conectado, en la terminal del servidor podemos ver que se ha efectuado la conexión y los archivos con los que cuenta este par. Cada vez que el cliente haga algún cambio, lo podremos visualizar dentro de esta terminal.

Después inicializaremos peer 2 y peer 3:

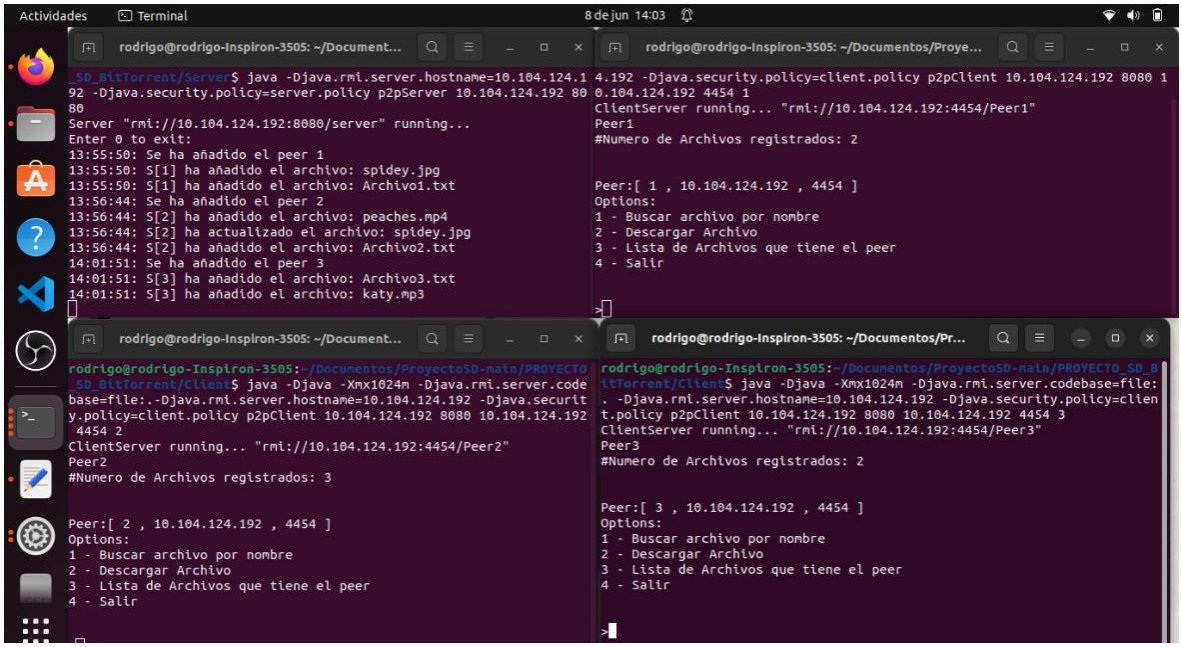




Como ya mencionamos, el servidor reacciona con cada movimiento que efectúen los clientes:

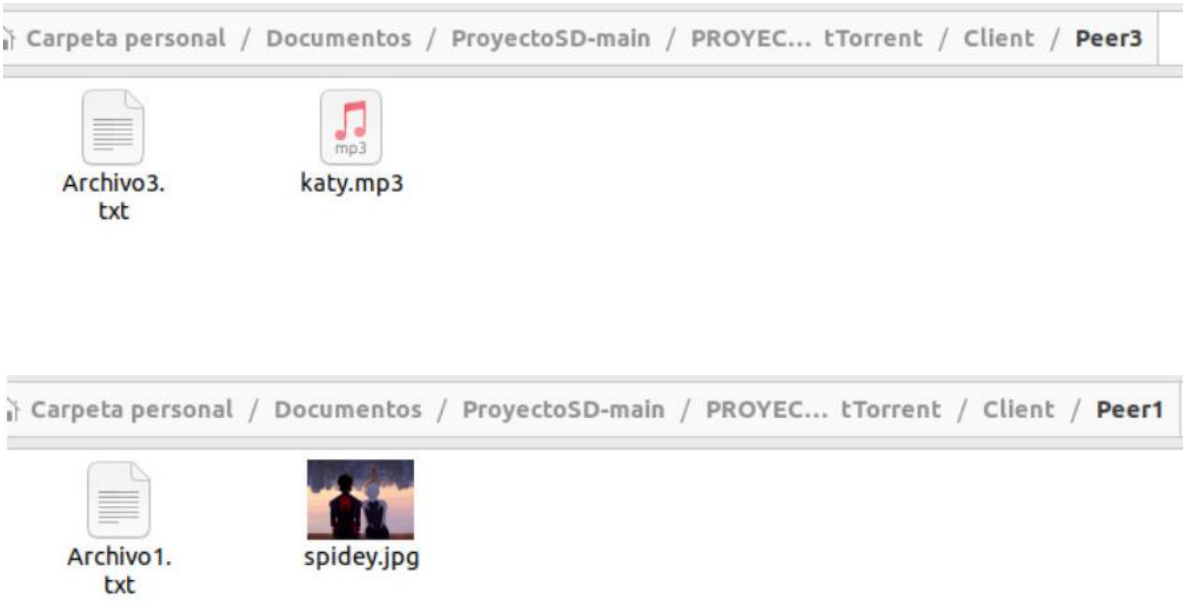


En la siguiente imagen podemos ver toda la conexión con el servidor y clientes:

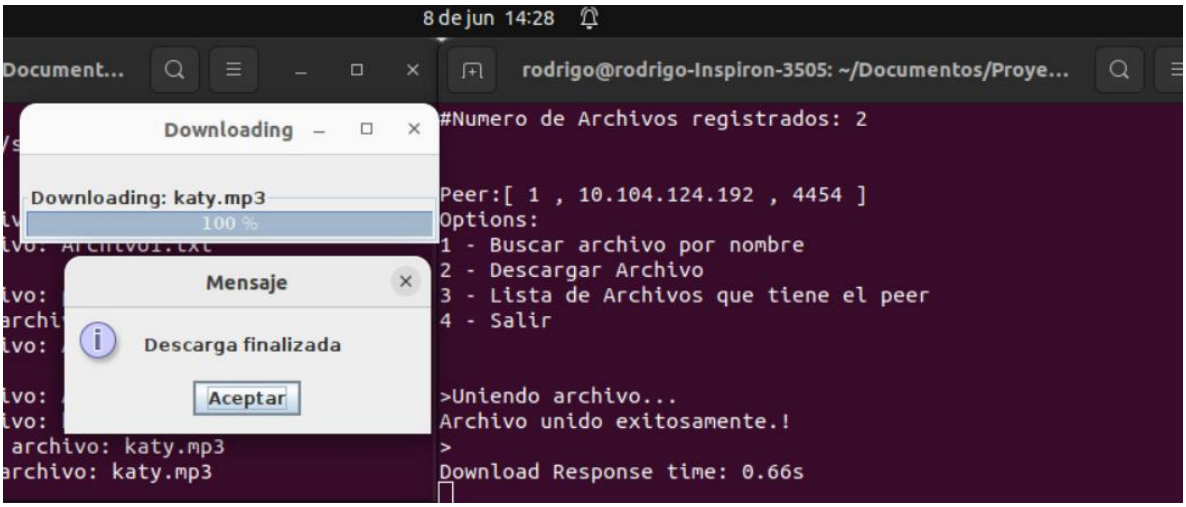


Pruebas Y Resultados

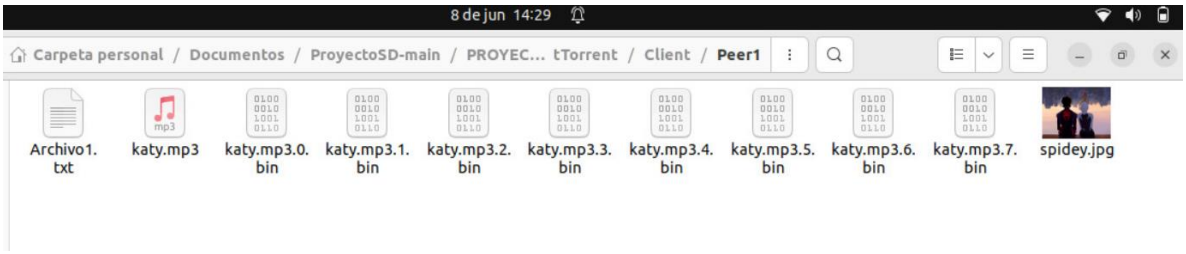
Como observamos, tenemos diversos archivos de texto, imágenes, audios y videos en los diferentes peers, como prueba vamos a descargar el archivo Katy.mp3 el cual vive en el Peer 3 desde el Peer 1

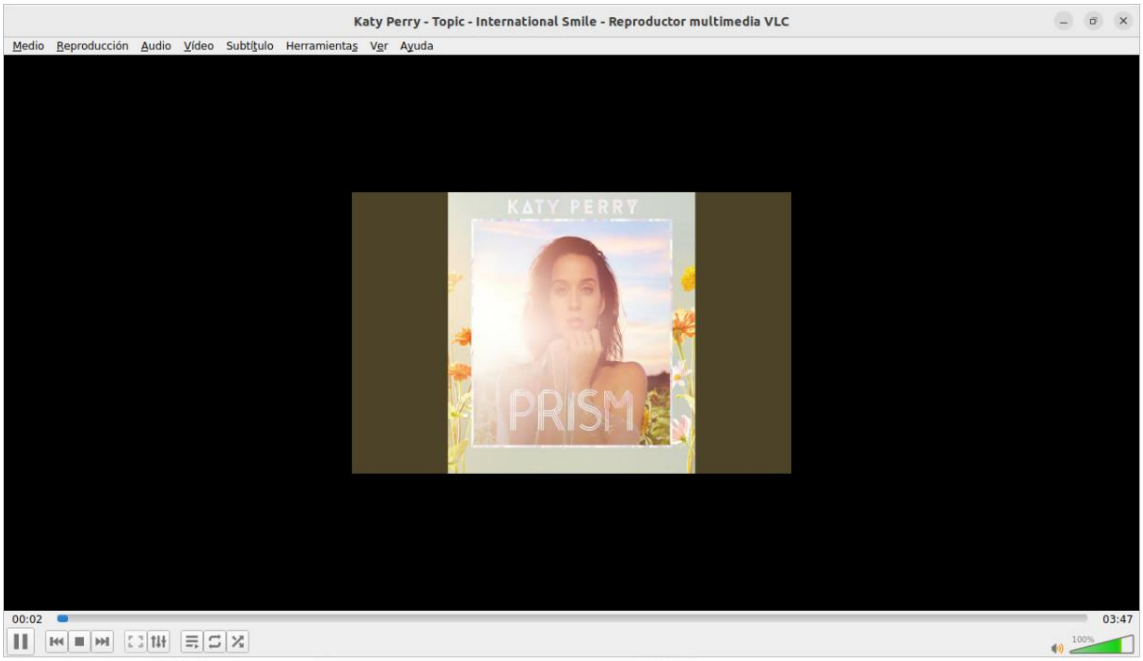


Como podemos ver, nos aparece que se ha descargado el archivo:

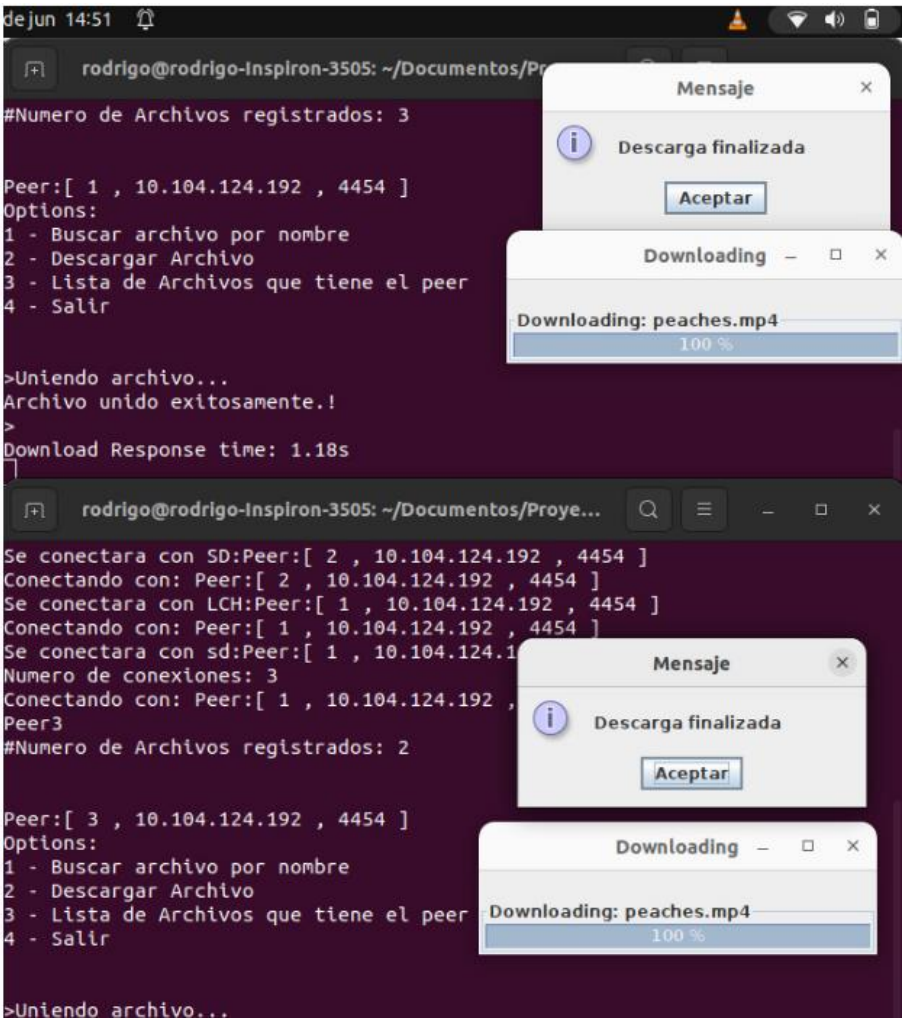


Ahora la carpeta del Peer 1 contiene los siguientes archivos descargados, podemos ver que para su descarga, el archivo se dividió en pequeñas particiones que son las que se nos muestra en la carpeta, una vez ensamblado el archivo ya se puede reproducir, en el servidor también se actualiza y muestra en la lista que se añadió el archivo al directorio del Peer 1

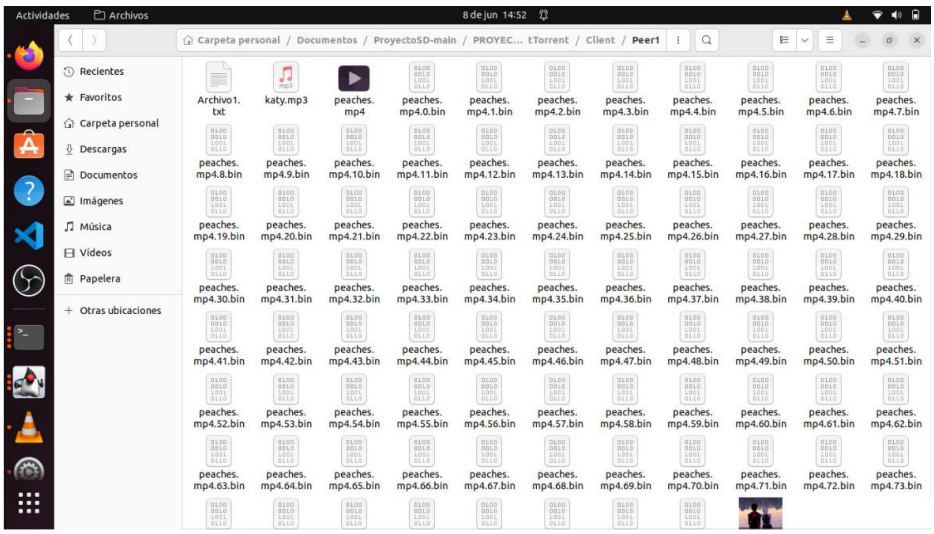


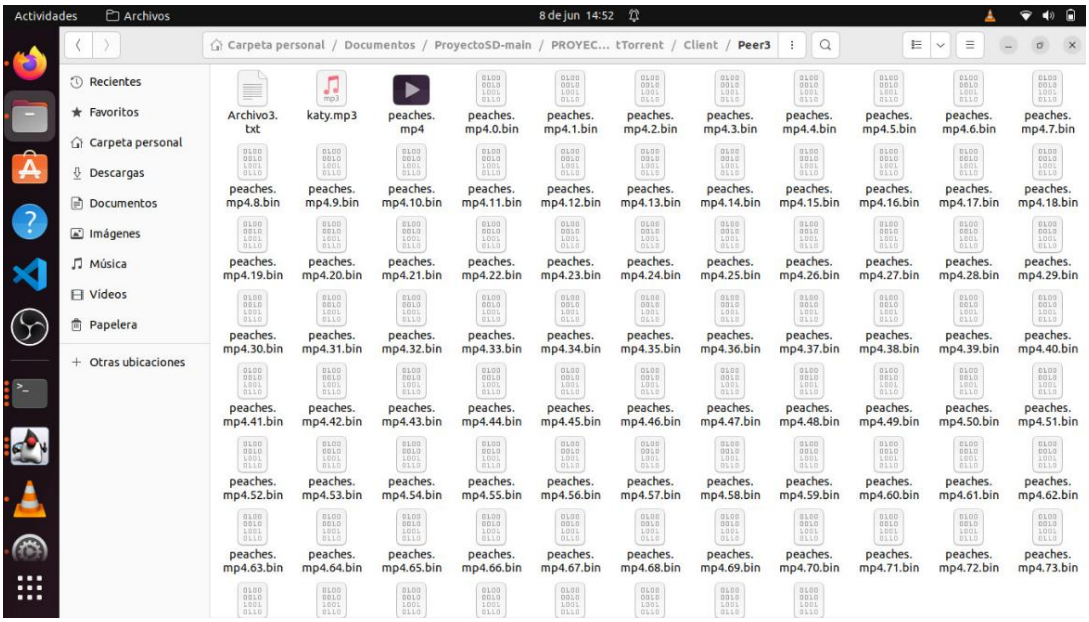


Como siguiente prueba, vamos a descargar el archivo de video “Peaches.mp4”, alojado en la carpeta del Peer 2, desde el Peer 3 y Peer 1 simultáneamente, teniendo el siguiente resultado:

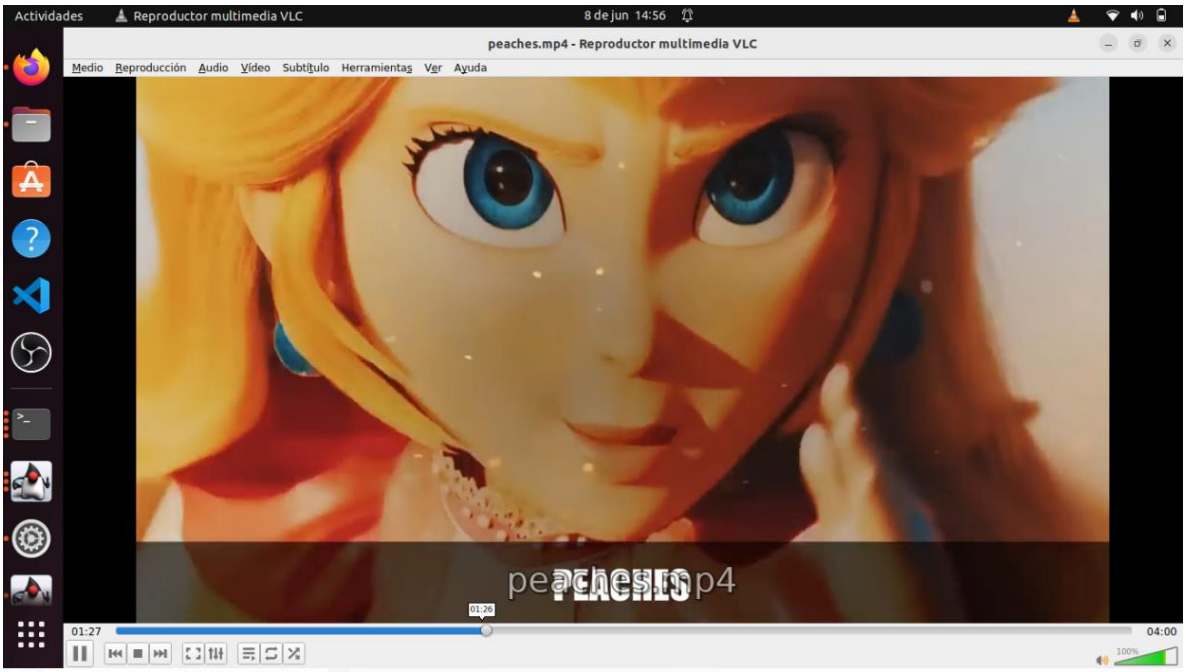


Y podemos ver que nuestro archivo se encuentra en ambos Peers de donde de hizo la descarga:





Y finalmente observamos que nuestro archivo se puede reproducir:



Código.

Github.

https://github.com/rodri0319/ProyectoFinalSistemasDistribuidos.git

Google Drive.

https://drive.google.com/drive/folders/1xQOWbgPEZPs6-RTeJY2l6n5iZGY_ghS0?usp=drive_link

CONCLUSIONES

Como conclusión, tenemos que el protocolo BitTorrent implementado para este proyecto funciona de manera correcta, teniendo la posibilidad de tener en tiempo real el despliegue de las acciones que van realizando los peers en sus respectivas carpetas compartidas, esto se puede visualizar fácilmente en la consola del servidor, el cual actualiza constantemente a los Peers para ver que archivos estan compartiendo, asi como aquellos archivos que van descargando de otros peers.

La posibilidad que tiene este programa es conectar hasta 5 Clientes, que fueron los que probamos, asi mismo se realizo una prueba para ver si era posible que los peers descargaran un archivo de forma simultanea proveniente de otro peer, teniendo un resultado exitoso.

Hablando de los archivos que nos dimos a la tarea de probar si era posible transferir, nos dimos cuenta de que tenemos la posibilidad de compartir archivos .txt, imágenes .jpg, jpeg, además de imágenes con movimiento como lo fue un .gif. Hablando de lo que fueron archivos grandes, nos dimos a la tarea de probar con elementos como una canción en .mp3 y un video en .mp4, dichas descargas fueron logradas de manera exitosa, además de que se pudo observar de mejor manera el proceso de fragmentación que lleva a cabo el código, asi como las barras de descarga y mensajes en pantalla que llamamos con JFrame.

Finalmente, el uso de RMI nos beneficia ya que asi fue fácil entender que el uso de llamadas a métodos remotos fue una manera sencilla de implementar el protocolo BitTorrent para este proyecto, de igual manera pudimos entender de mejor manera el uso de conceptos como Trackers, Leechers y Seeders.

BIBLIOGRAFIA Y CYBERGRAFIA.

[1] https://www.redeszone.net/tutoriales/internet/que-son-peer-seed-leech-ratio-p2p/

[2] https://www.youtube.com/watch?v=IQYYqjfnajU&t=102s

[3] https://www.youtube.com/watch?v=vyXOYtdUiCw