Instituto Politécnico Nacional

Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas


Nombre de los alumnas:

Campos Rodríguez Diana

López Vega Camila

Unidad de aprendizaje:

Sistemas Distribuidos

Profesor: Miguel Félix Mata Rivera

Grupo: 2TV7

# Proyecto final: Simulación de BitTorrent

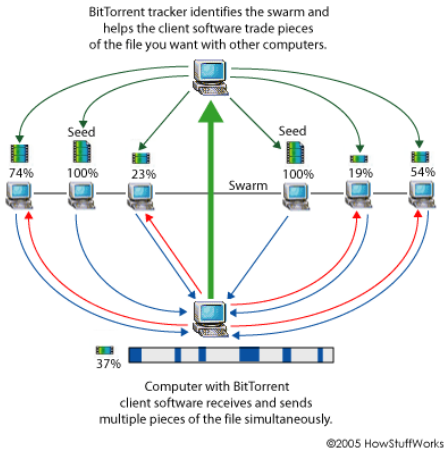
Ciudad de México a 10 de enero del 2024

## ¿Qué es un sistema distribuido?

Es un conjunto de programas informáticos que utilizan recursos computacionales en varios nodos de cálculo distintos para lograr un objetivo compartido común. La finalidad de los sistemas distribuidos es eliminar los cuellos de botella o los puntos de error centrales de un sistema, usan nodos distintos para comunicarse y sincronizarse a través de una red común. Estos nodos suelen representar dispositivos de hardware físicos diferentes, pero también pueden representar procesos de software diferentes u otros sistemas encapsulados recursivos.

## ¿Qué es un Bit Torrent?

Es un protocolo de transferencia de Internet. Al igual que http (protocolo de transferencia de hipertexto) o ftp (protocolo de transferencia de archivos), Bit Torrent es un modo de descargar archivos desde Internet. Pero, a diferencia de ellos, Bit Torrent es un protocolo de transferencia distribuida. El protocolo punto a punto localiza usuarios con archivos que otros usuarios desean y descarga partes de los archivos de estos usuarios de forma simultánea



## Partes de un Bit Torrent

- **Torrent:** Contiene metadatos de los archivos y carpetas que se van a compartir, así como información que ayuda a la transferencia de archivos entre nodos.
- **Tracker:** Es un servidor que coordina las conexiones entre los pares que participan en la descarga.
- **Peer:** Cliente Bit Torrent que está descargando o compartiendo archivos en la red, cada par puede enviar y recibir datos de otros pares.
- **Seeder:** Es un par que ha descargado el archivo completo y ahora comparte todas las partes del archivo, son esenciales para mantener el archivo disponible para descarga
- **Leecher:** Es un par que está descargando el archivo, pero aún no completa la descarga, al completar la descarga un leecher puede convertirse en un seeder y comparte archivos.
- **Pieza:** Es un archivo grande que se divide, cada pieza tiene un tamaño especifico y se le asigna un hash único (para evitar corrupción de datos durante la transferencia). Los pares intercambian estas piezas durante la descarga.
- **Puerto:** Son para la transferencia de datos entre los pares.

## ¿Qué es P2P?

P2P son las siglas de peer-to-peer y es un modelo de comunicación y distribución de recursos en una red de computadoras donde cada nodo tiene funciones tanto de cliente como de servidor, los dispositivos interactúan entre sí.

P2P es común en diversas aplicaciones como transmisión de medios, mensajería instantánea, envío y descarga de archivos, etc.

En caso de un Bit Torrent, los usuarios descargan y comparten entre si partes de un archivo en lugar de depender de un servidor central para la descarga.

## Listado de archivos del proyecto y descripción

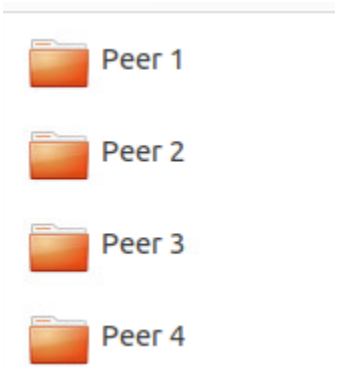
### Carpeta SERVER

- **ServerImpl:** Su propósito es actuar como un servidor para un sistema de intercambio de archivos P2P y extiende UnicastRemoteOnject para la comunicación e invocación remota (JAVA RMI).
- **Torrent:** Contiene información relacionada con el nombre, tamaño, piezas de los archivos y seeders y leechers asociados.
- **P2pServer:** Actua como el servidor principal para el intercambio de archivos así com también gestionar la entrada desde la consola.
- **ServerI:** Define la funcionalidad del servidor en el sistema de intercambio P2P.
- **PeerData:** Encapsula los datos asociados a un par.
- **HiloS:** Realiza verificaciones de los pares conectados al servidor para así mantener la lista de pares conectados y eliminar aquellos que pierden conexión o se desconectan.
- **Server.policy:** Es el archivo de política de seguridad para permitir que cualquier código de la carpeta tenga acceso a los permisos de seguridad.

### Carpeta CLIENT

- **ClientImpl:** Busca y descarga archivos de otros peers en la red, los archivos se dividen en pedazos para después ser compartidos.
- **ClientI:** Permiten obtener las piezas, verificar conexión, verificar información de progreso de descarga y envio y también obtener los identificadores.
- **P2pClient:** Describe la conexión del cliente y el servidor.
- **Fragmentación:** Fragmenta y une los archivos.
- **CryptoUtils:** Cifra y decifra datos usando algoritmo AES.
- **CryptoException:** Envia excepciones relacionadas con la criptografía.
- **HiloP**: Es para descargar los archivos de un peer, maneja errores o interrupciones durante el proceso.

- **Progreso:** Obtiene información sobre las piezas descargadas y calcula el progreso y también verifica cuando este termine.
- **VentanaD:** Despliega una ventana mostrando la barra de progreso de descarga, asimismo muestra un mensaje cuando se termina la descarga.
- **HiloV:** Va actualizando la barra de descarga.
- **Prueba:** Prueba la conexión y el acceso remoto.
- **Client.policy:** Política de seguridad, permite el acceso a los recursos.
- **Dentro de esta carpeta se encuentran las carpetas de los peer y cada uno tiene un archivo diferente (imagen, gif, video, audio).**




## Funcionamiento del proyecto

Primero se compilaron todos los archivos .java del proyecto para generar archivos .class para asegurar que no había ningún error con los códigos y así dar comienzo a ejecutarlo.

Posteriormente, se abrieron varias ventanas de consola para el servidor y 4 clientes (4 peer), para ejecutar el servidor se usó el comando:

**java -Djava.rmi.server.hostname=172.17.0.1 -**
**Djava.security.policy=server.policy p2pServer 172.17.0.1 8080**

En este comando se especifica la dirección ip de la máquina virtual y a que puerto se va a conectar, así también que se hará uso de la política de seguridad.

```
alumno@alumno-virtual-machine:~/ProyectoFinalSistemasDistribuidos
/ProyectoSD_Torrent/Server$ java -Djava.rmi.server.hostname=172.1
7.0.1 -Djava.security.policy=server.policy p2pServer 172.17.0.1 8
080
```

Y así se ve en consola cuando se esta ejecutando el servidor correctamente y en espera de conexión.

```
Server "rmi://172.17.0.1:8080/server" corriendo...
Presiona 0 para salir:
```

Para ejecutar el cliente se usó el siguiente comando:

**java -Djava -Xmx1024m -Djava.rmi.server.codebase=file:. -Djava.rmi.server.hostname=172.17.0.1 - Djava.security.policy=client.policy p2pClient 172.17.0.1 8080 172.17.0.1 4454 1**

Este comando hace referencia a que servidor se va a conectar el cliente, con qué dirección ip y que puerto y también se ejecuta su política de seguridad.

```
alumno@alumno-virtual-machine:~/ProyectoFinalSistemasDistribuidos/ProyectoS
D_Torrent/Client$ java -Djava -Xmx1024m -Djava.rmi.server.codebase=file:. -
Djava.rmi.server.hostname:172.17.0.1 -Djava.security.policy=client.policy p
2pClient 172.17.0.1 8080 172.17.0.1 4454 1
```

Cuando los clientes se conectan correctamente con el servidor, la consola se ve así.

**Servidor:**

```
Server "rmi://172.17.0.1:8080/server" corriendo...
Presiona 0 para salir:
13:28:34: Se ha añadido el peer 1
13:28:34: S[1] ha añadido el archivo: cancion.mp3
13:28:54: Se ha añadido el peer 2
13:28:54: S[2] ha añadido el archivo: barbie.mp4
13:29:06: Se ha añadido el peer 3
13:29:06: S[3] ha añadido el archivo: jack.jpg
13:29:20: Se ha añadido el peer 4
13:29:20: S[4] ha añadido el archivo: patricio.gif
13:29:20: S[4] ha añadido el archivo: pelicula.mp4
```

**Cliente 1:**


**Cliente 2:**

**Cliente 3:**


**Cliente 4:**

Y esto muestra cada una de sus opciones:

**Opción 1**

```
Peer 1:
IP:172.17.0.1
Puerto:4454

Menu:
1. Buscar archivo
2. Descargar archivo
3. Lista de archivos
4. Salir


Opcion:1
Introduce el nombre del archivo: jack.jpg

Los siguientes peers tienen el archivo (jack.jpg) :
3


Tiempo de descarga: 0.01s
```

**Opción 2 y como se refleja en el servidor**

```
Peer 1
Numero de archivos registrados: 1


Peer 1:
IP:172.17.0.1
Puerto:4454

Menu:
1. Buscar archivo
2. Descargar archivo
3. Lista de archivos
4. Salir


Opcion:2
Introduce el nombre del archivo: patricio.gif
Descarga completa
```

```
13:34:09: S[1] ha actualizado el archivo: patricio.gif
```

**Opción 3**

```
Peer 1
Numero de archivos registrados: 2


Peer 1:
IP:172.17.0.1
Puerto:4454

Menu:
1. Buscar archivo
2. Descargar archivo
3. Lista de archivos
4. Salir


Opcion:3


Archivos compartidos:
patricio.gif
cancion.mp3
```

**Opción 4 y como se refleja en el servidor**

```
Peer 2
Numero de archivos registrados: 1


Peer 2:
IP:172.17.0.1
Puerto:4454

Menu:
1. Buscar archivo
2. Descargar archivo
3. Lista de archivos
4. Salir


Opcion:4
Adios . . .
```

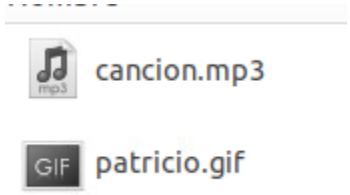
```
13:43:45: Se ha eliminado el peer 2
```

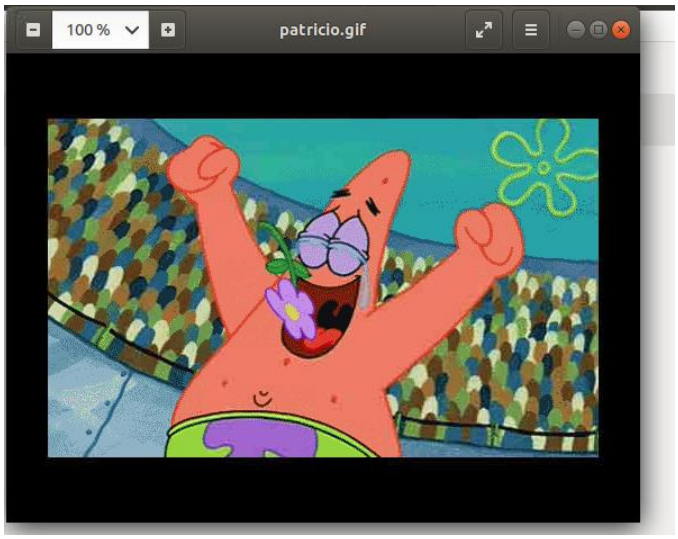
## Casos de funcionamiento

### Caso 1: Descarga de una imagen

```
Peer 1
Numero de archivos registrados: 1


Peer 1:
IP:172.17.0.1
Puerto:4454

Menu:
1. Buscar archivo
2. Descargar archivo
3. Lista de archivos
4. Salir


Opcion:2
Introduce el nombre del archivo: patricio.gif
Descarga completa
```

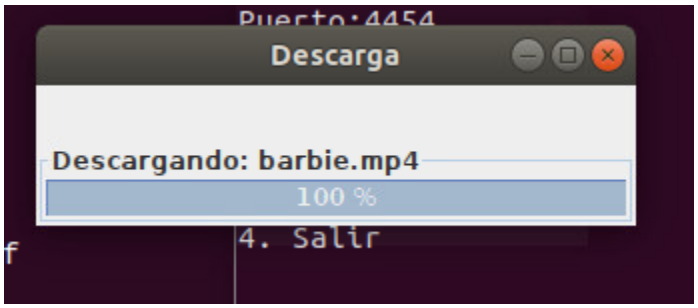
Nos dirigimos a la carpeta del Peer 1 para comprobar que si este la imagen y que se abra correctamente.

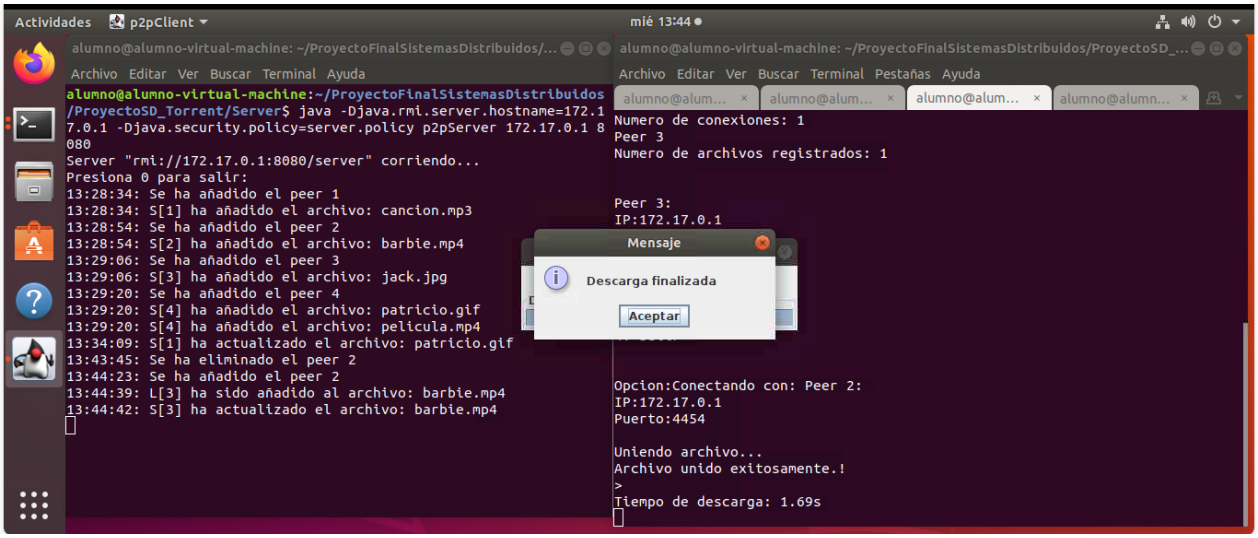


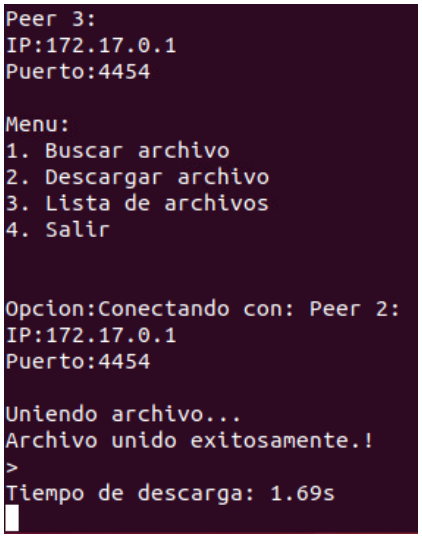



**Caso 2: Descarga de un video de duración 4 minutos**

La consola abre una ventana que muestra el progreso de la descarga y cuando finaliza muestra un mensaje de que la descarga finalizó.

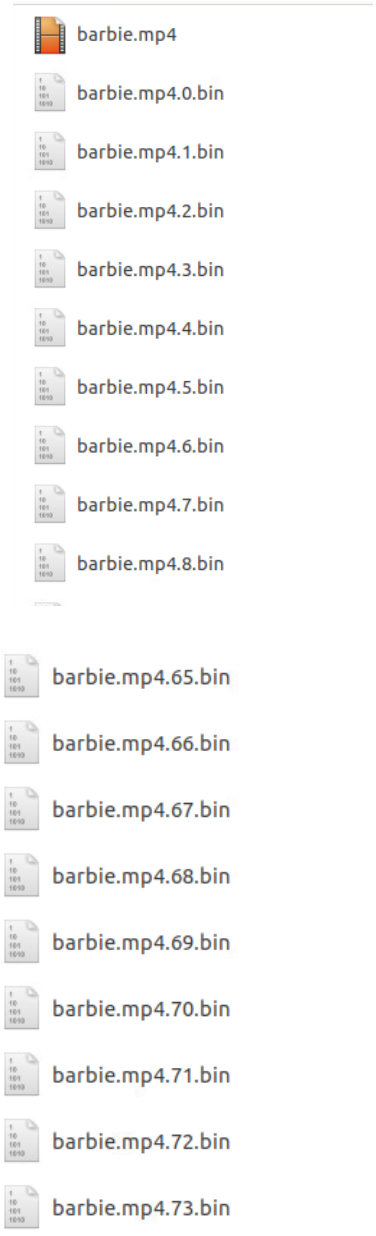


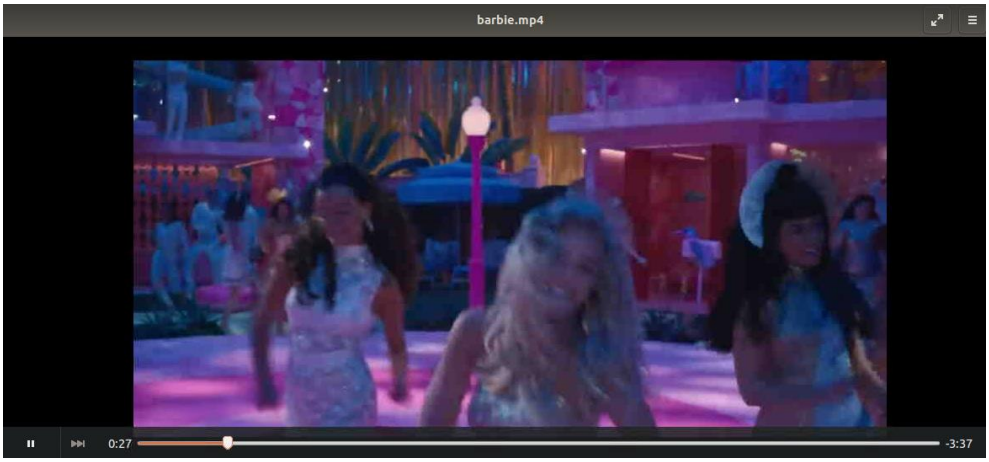



```
13:44:39: L[3] ha sido añadido al archivo: barbie.mp4
13:44:42: S[3] ha actualizado el archivo: barbie.mp4
```

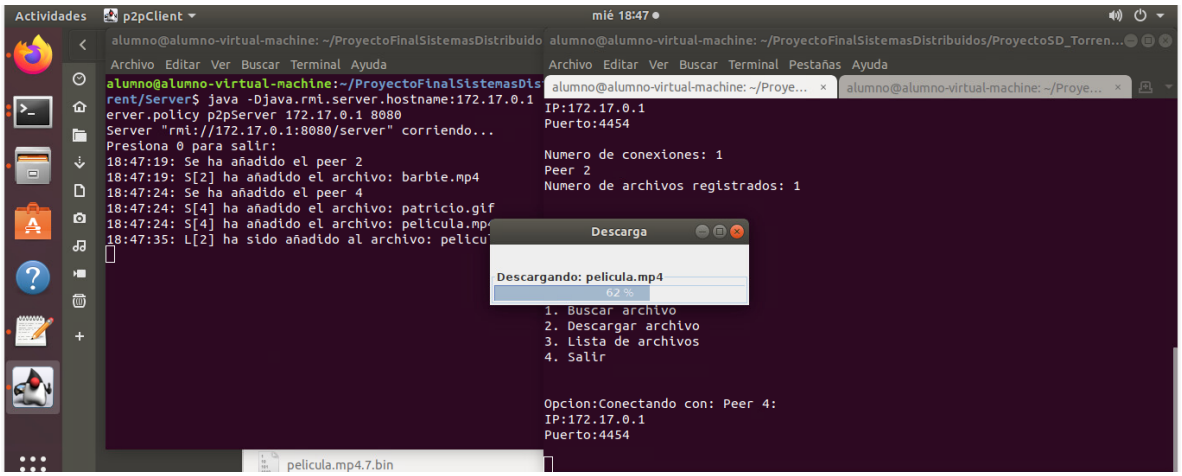
Al ir a la carpeta a verificar que el archivo se haya descargado bien, se puede ver las piezas en las que se dividió y posteriormente se unieron.



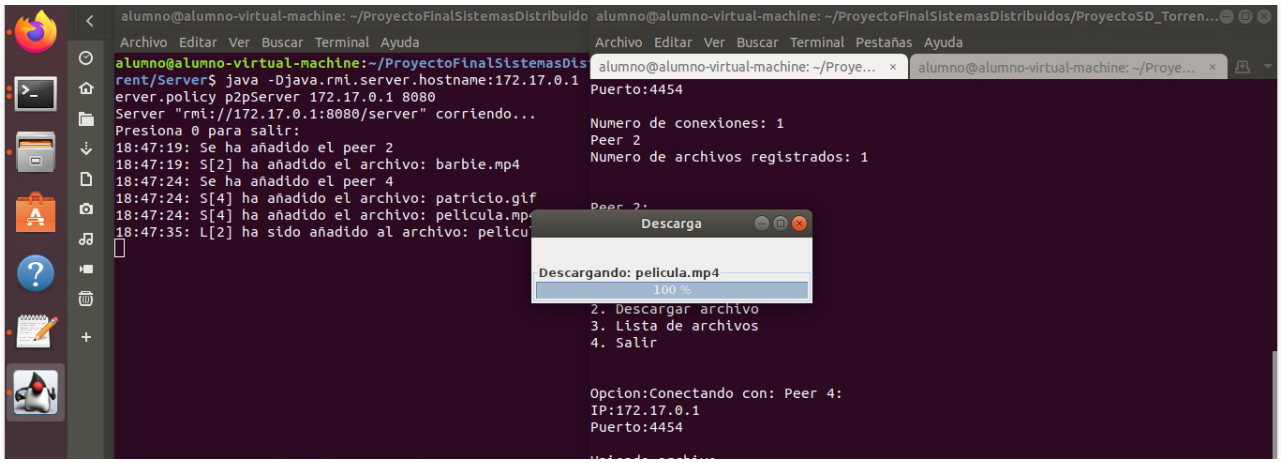
Y esta es la reproducción de dicho video

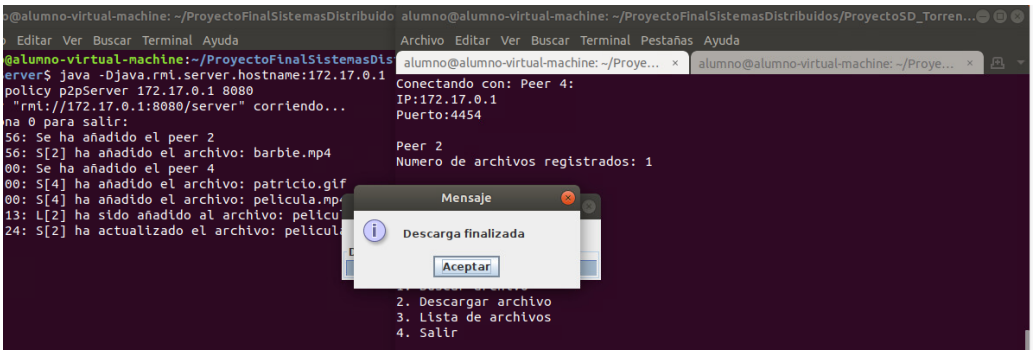



**Caso 3: Descarga de una película completa con interrupción y retoma de la descarga.**

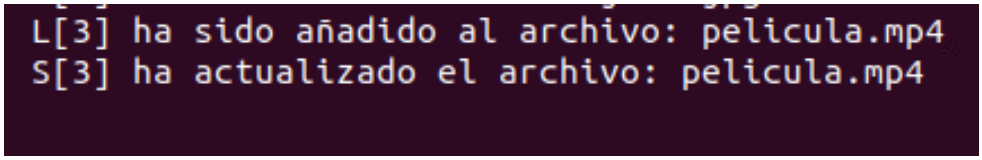



En este punto de la descarga se interrumpió la conexión a internet de la maquina virtual, posteriormente se volvió a conectar y se retomo la descarga donde se había quedado.



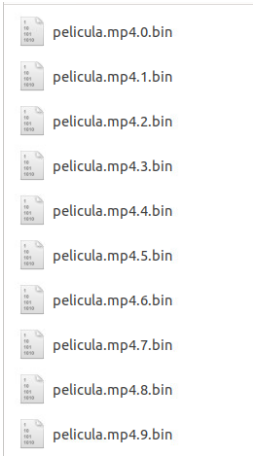

Así se ve reflejado en el servidor

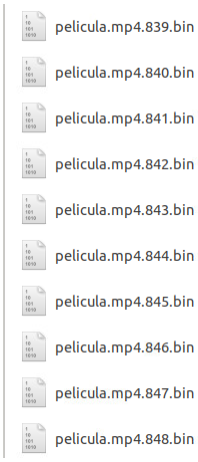


Y así en el cliente

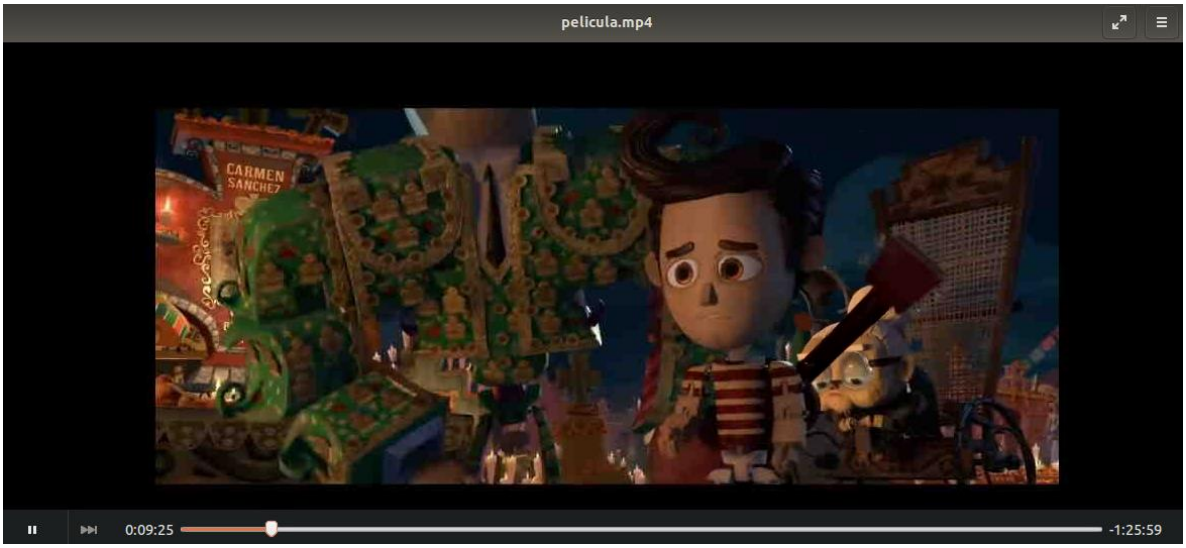
```
Peer 2:
IP:172.17.0.1
Puerto:4454

Menu:
1. Buscar archivo
2. Descargar archivo
3. Lista de archivos
4. Salir


Opcion:Conectando con: Peer 4:
IP:172.17.0.1
Puerto:4454

Uniendo archivo...
Archivo unido exitosamente.!
>
Tiempo de descarga: 36.45s
```

En la carpeta las piezas que se dividió la película fueron 848.



Y esta es la reproducción.




NOTA: Al finalizar el servidor muestra cuanto tiempo se estuvo ejecutando

```
0
Tiempo total: 1168.68s
```