Laboratorio UDP

User Datagram Protocol (UDP) es un protocolo de la capa de red. Este es la alternativa frente a TCP cuando lo que más se requiere es velocidad en el envío y no es indispensable que haya una alta confiabilidad.

Este protocolo tcumple con las características antes descritas dado que este no funciona con una comunicación directa cliente-servidor, tal como lo hace TCP, sino que hay envío de datos a través de un stream. El cliente va recibiendo los paquetes enviados mediante TCP pero no hay ninguna corroboración o envío por parte del cliente de que los paquetes llegaron completos o no.

Este es un protocolo que se utiliza bastante en video streaming, ya que no es indispensable que no haya pérdida de ningún frame para que las aplicaciones sigan en funcionamiento.

Adicionalmente, los paquetes que se envían a través de UDP tienen menos información en el header que los que se envían por TCP, ya que la información que es relevante a la hora de un envío UDP son la IP destino, el puerto destino, la IP origen, el puerto origen, el tamaño del mensaje y el mensaje (el cual no puede superar los 64KB).

En el siguiente repositorio se encuentra el desarrollo de UDP – transferencia de archivos y UDP – Streaming: <https://github.com/DanielaRocha6/Lab4>

## Transferencia de archivos

### Estructura del servidor

El programa del servidor se compone de 2 clases. La primera y principal se encarga de recibir los mensajes de los clientes y coordinar los diferentes subprocesos que se encargarán del envío de paquetes al cliente. Estos subprocesos son instancias de la segunda clase que compone al servidor.

La funcionalidad de la clase principal es que al iniciar, pregunta qué archivo desea enviar y cuántas conexiones desea tener.

Una vez definida la cantidad de clientes que se van a conectar, el servidor inicializa un Datagramsocket en el puerto 3001 y luego se queda escuchando hasta recibir la conexión de dicha cantidad de clientes. Esto lo hace leyendo el mensaje y corroborando que sea el saludo del cliente, luego guarda el puerto de conexión de cada cliente y aumenta la cantidad de conexiones.

Cuando se completa la cantidad de clientes conectados, se carga el archivo en una constante de la clase y se procede a inicializar los threads.

Al inicializar cada uno de los threads, se le indica el archivo y el puerto al cual se le va a enviar al cliente. Después de esto, se inicia el envío de paquetes, cada uno de 64000 Bytes a excepción del último, que puede variar.

### Estructura del cliente

El programa del cliente al iniciar, envía un mensaje de inicio a la dirección y puerto asignados para el servidor.

Una vez enviado este, el cliente empieza a esperar la llegada de los paquetes del archivo que envía el servidor y los empieza a almacenar en un arreglo.

Cuando llega el último paquete, identificado con un “end”.

Luego se reconstruye el archivo a partir de los paquetes obtenidos y se guarda el archivo en una carpeta Descargas, donde también estarán los logs.

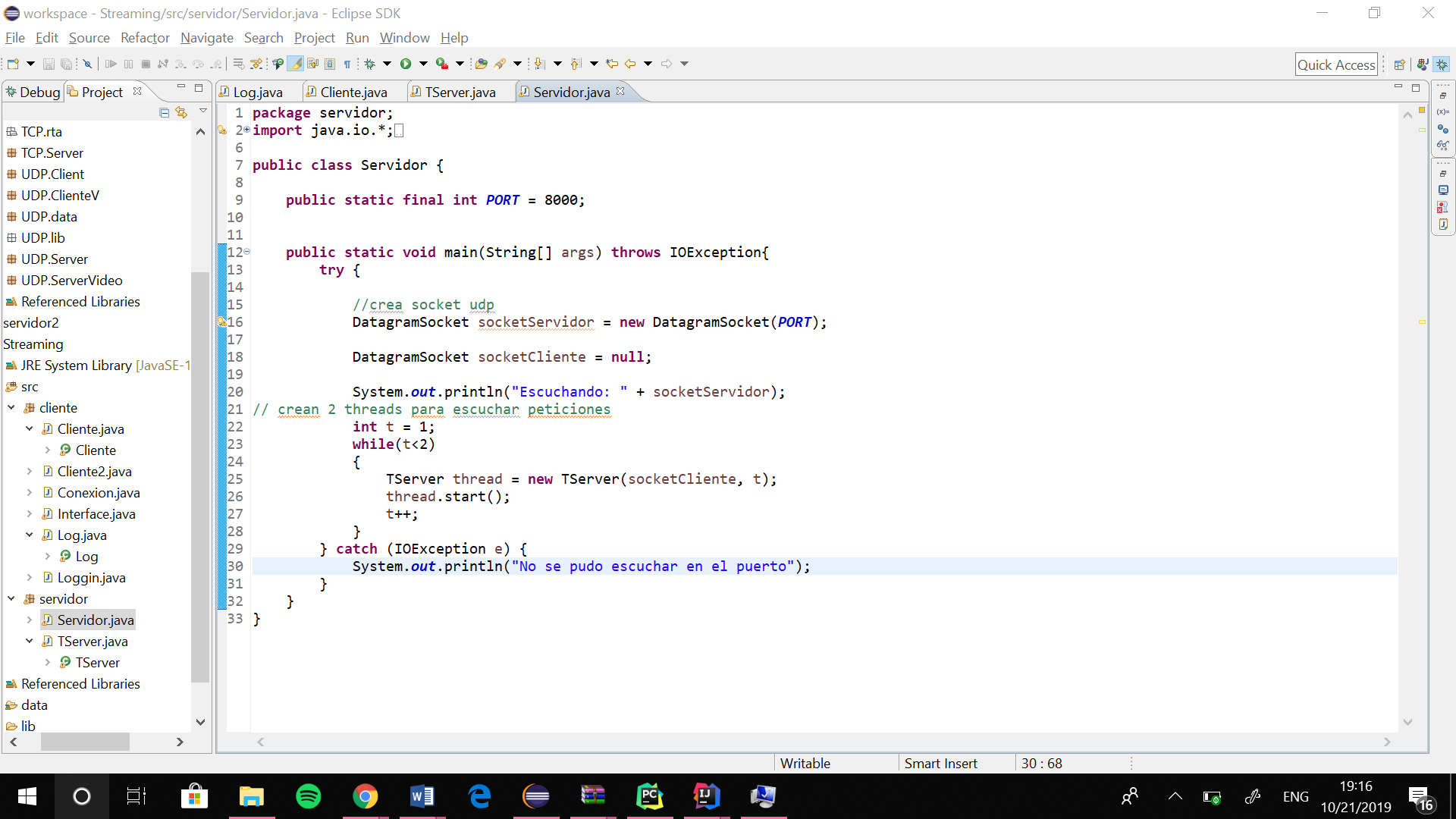
Luego, el genera el hash de dicho archivo y lo compara con el enviado por el servidor, confirmando o no la integridad del archivo.

Por último, se realiza el log que contiene el tiempo de inicio de la transacción, el nombre del archivo guardado, el tamaño del archivo, IP y puerto, resultado de la verificación de integridad y tiempo que tardó la transacción, el cual se almacena también en el directorio Descargas.

## Streaming

### Estructura del servidor

El servidor crea y establece un DatagramSocket en el Puerto 8000 donde escucha las peticiones y espera a que se conecten.

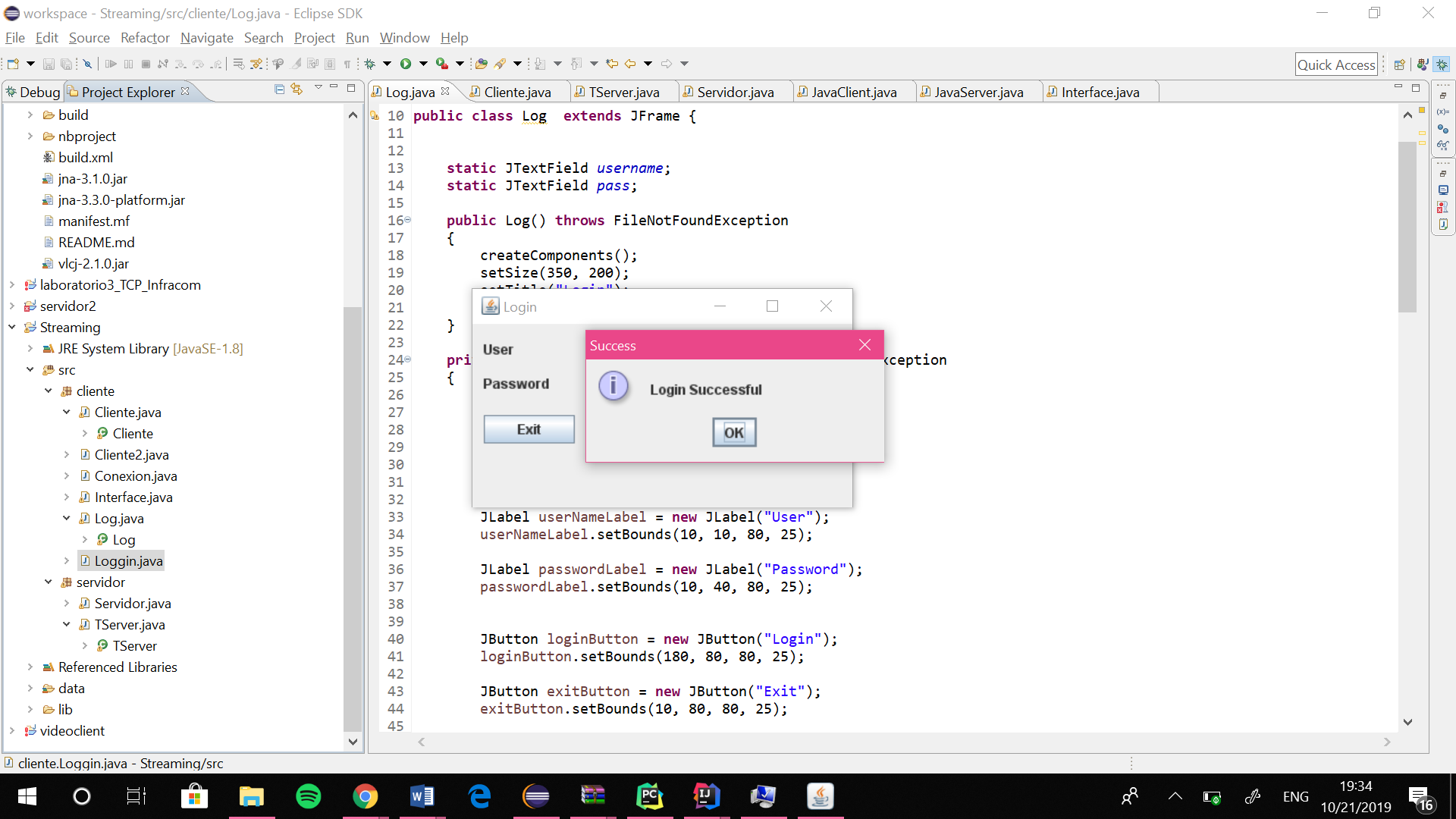


Se quiere crear un grupo de multicast, con MulticastSocket y se genera una IP en los valores disponibles de multicast. Con InetAddres se une al grupo.

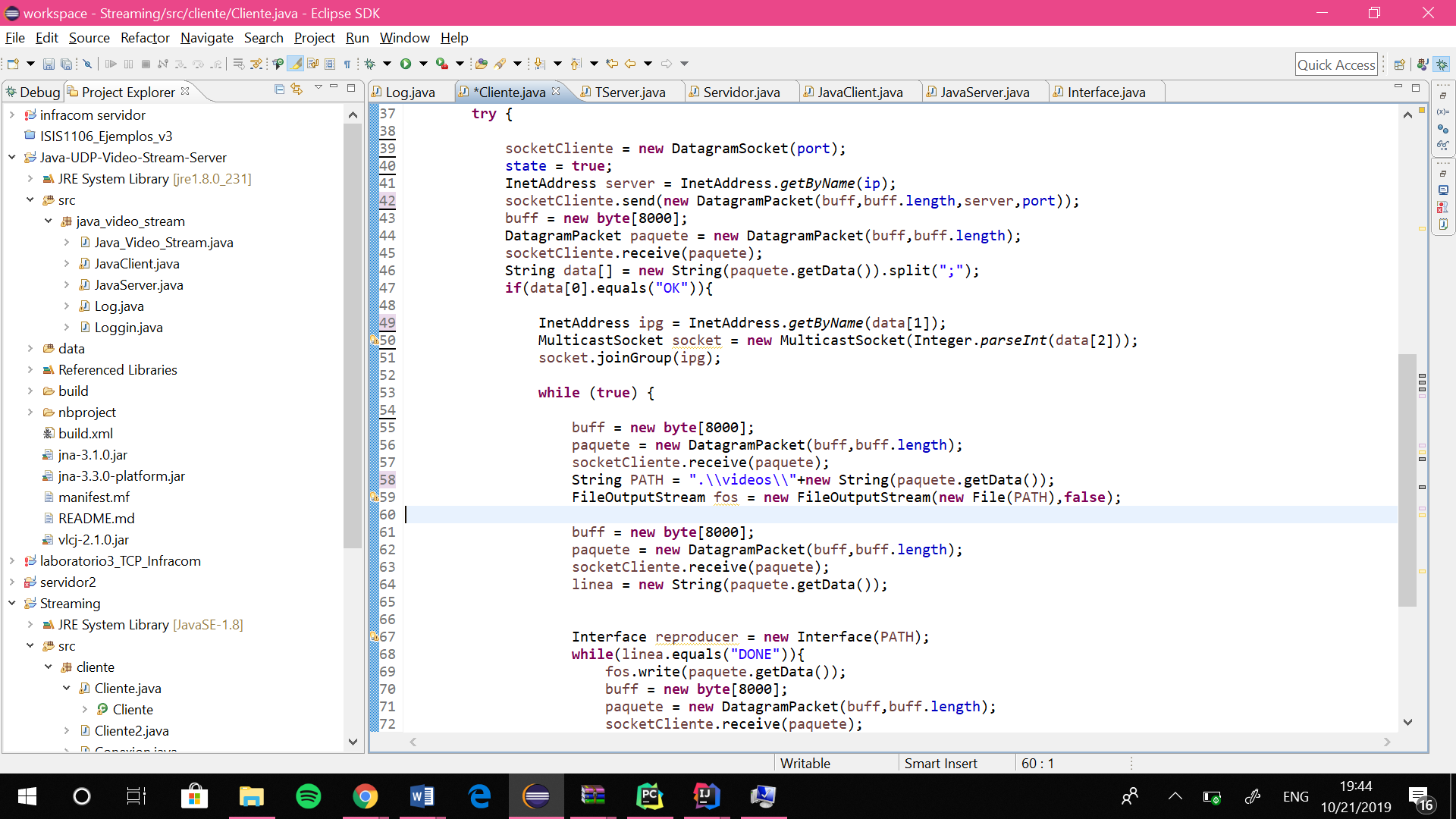
Se usa el API de VCL player

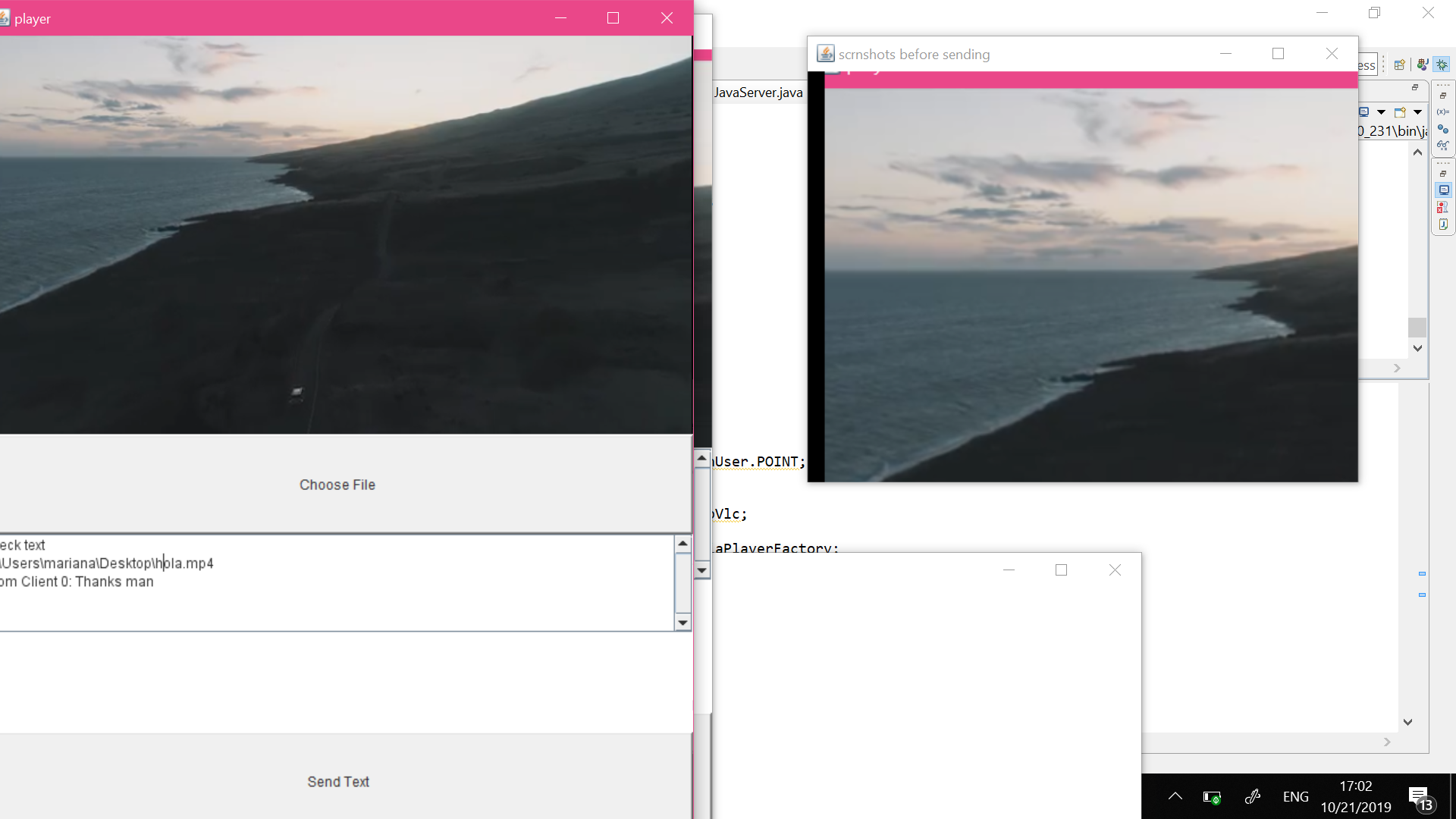
### Estructura del cliente

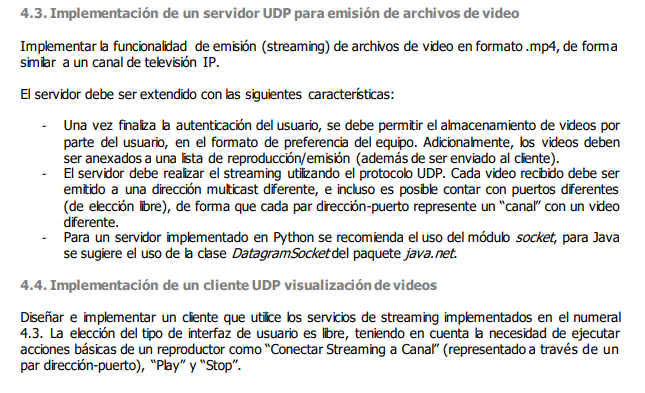
El cliente se autentica y se conecta con el servidor



El socket UDP se crea y establece y se envía el estado que ya está preparado al servidor, luego el cliente recibe el mensaje de confirmación y se inicializa el socket multicast.







## Análisis de resultados

#### Realice pruebas de desempeño para la transferencia de archivos con UDP. Establezca comparativos y análisis de parámetros en las pruebas de las aplicaciones de TCP y UDP.

Cuando se realizó la transferencia de archivos, se halló que el 100% de las veces el paquete llegaba completo. Sin embargo, al realizar cliente y servidor UDP se notó que cuando se hacía envíos a varios clientes, en promedio el 30% de los clientes no recibían completos los paquetes y por ende, la integridad de los archivos se veía comprometida.

#### ¿Son congruentes los resultados obtenidos en las dos actividades de este laboratorio con la teoría presentada en la clase teórica?

Sí. Los resultados son bastante coherentes con lo que se ha aprendido en la clase teórico. Dado que, como se planteaba, por el protocolo TCP no hay pérdida de paquetes gracias a su forma de corroborar que al cliente le lleguen todos los paquetes y que si no es así, se reenvíen. Mientras que en UDP la pérdida de paquetes es frecuente, ya que no hay garantía de la integridad de los archivos.

#### ¿Cómo implementaría un servicio de streaming usando TCP?

Dado que TCP tiene limitaciones integradas debido a su objetivo inicial que pueden afectar servicios de Streaming, como el transporte confiable de datos y la inclusión de mecanismos de congestión, es posible usarse en las circunstancias adecuadas. Por ejemplo, TCP se puede combinar con la transmisión adaptativa HTTP, en donde los usuarios pueden tolerar una latencia de 20 o 30 segundos entre la información enviada y mostrada. La descarga de videos en segmentos se realiza a través de TCP y puede existir un reproductor del lado del cliente que intente mantener fragmentos por delante de donde está el usuario. Si el reproductor no puede descargar lo suficientemente rápido como para adelantar al usuario, cambia a un fragmento de menor resolución y más pequeño en la siguiente solicitud.

Usando HTTP cada fragmento tiene una URL diferente por tanto se pueden almacenar en cache. TCP opera usando un circuito virtual, es decir, un flujo continuo de datos que contiene una secuencia de bytes, en donde cada byte no llega hasta que los anteriores lo hayan hecho. Si se envía una secuencia de 10 paquetes y se pierde el primero, los otros 9 se guardan en el buffer al final mientras se solicita el faltante.

#### ¿Es posible desarrollar aplicaciones UDP que garanticen la entrega confiable de archivos? Qué consideraciones deben tenerse en cuenta para garantizar un servicio de entrega confiable utilizando dicho protocolo. Justifique su respuesta.

Sí, es posible, manejando correctamente Threads y realizando esperas suficientes de tal manera que se espere a la llegada de cada paquete. Evitando que lleguen en desorden o no lleguen.