INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE COMPUTO

Práctica 2

Envió de Canción

Profesor: Axel Ernesto Moreno Cervantes

Alumnos:

Jiménez Rodríguez Alejandro Martin

Núñez Ramírez Valery Aylin

Grupo: 6CM1

Fecha: 08/12/2025

Transmisión de una Canción mediante Sockets de Datagrama en Java

Introducción

En esta práctica se implementó un sistema de transmisión de archivos multimedia (en este caso, una canción en formato MP3) utilizando **sockets de datagrama (UDP)** en Java. A diferencia de TCP, el protocolo UDP no garantiza la entrega confiable de los datos, por lo que fue necesario diseñar un mecanismo adicional para lograr una recepción correcta. Para ello, se implementó una **ventana deslizante con acuses de recibo (ACKs)** que asegura que cada fragmento de la canción llegue íntegro al cliente.

La aplicación completa consta de tres componentes principales:

1. **smusica.java** – servidor UDP encargado de leer la canción, fragmentarla y enviarla.
2. **cmusica.java** – cliente UDP que recibe los fragmentos, los valida mediante números de secuencia y envía los acuses.
3. **App.java** – interfaz JavaFX que ejecuta al cliente y reproduce la canción una vez recibida.

Esta práctica permitió comprender el funcionamiento de los datagramas, así como la gestión manual de confiabilidad sobre un canal no orientado a conexión.

Explicación del Código

1. Servidor: smusica.java

El servidor inicia leyendo un archivo MP3 desde el sistema y preparándolo para enviarlo en bloques de 1024 bytes. Además, establece una **ventana deslizante**, cuyo tamaño es proporcionado por el usuario:

```
byte[] trozo = new byte[1024];

int ventana = s.nextInt();

byte[][] ventanaData = new byte[ventana][];
```

Cada fragmento se envía junto con un **número de secuencia**, lo cual permite al cliente reconstruir la canción en orden. Para empaquetar los datos se utiliza un DataOutputStream:

```
dos.writeInt(numSec);       // Número de secuencia

dos.write(trozo, 0, bytesLeidos);  // Datos del fragmento
```

El servidor envía los paquetes al cliente mediante un DatagramPacket y almacena copias de cada trozo dentro de la ventana, para reenviarlos en caso de pérdida.

Posteriormente, el servidor espera un **ACK** del cliente. Si lo recibe dentro del tiempo límite, avanza la ventana:

```
int ackRecibido = disAcuse.readInt();

if (ackRecibido >= base) base = ackRecibido + 1;
```

En caso de no recibirlo (timeout), se reenvían todos los fragmentos pendientes:

```
catch (SocketTimeoutException e) {

    for (int i = base; i < numSec; i++) {

        socket.send(dpReenviar);

    }

}
```

Finalmente, cuando se envía todo el archivo, se transmite un paquete con número de secuencia **–1**, indicando el fin de la canción.

2. Cliente: cmusica.java

El cliente primero envía un mensaje de inicio al servidor para indicar que está listo:

```
DatagramPacket dpe = new DatagramPacket(

    mensaje.getBytes(), mensaje.length(), direccionServidor, 1234

);

socket.send(dpe);
```

Luego entra en un ciclo en el que recibe datagramas y extrae el número de secuencia:

```
DataInputStream dis = new DataInputStream(

    new ByteArrayInputStream(dpr.getData())

);
```

```
int numSec = dis.readInt();
```

Si el número de secuencia coincide con el esperado, el cliente escribe el fragmento en el archivo MP3:

```
if (numSecExp == numSec) {
    fos.write(data);
    numSecExp++;
}
```

Después envía un **acuse de recibo** indicando el número del último fragmento correcto:

```
dos.writeInt(numSecExp - 1);
socket.send(dp);
```

Cuando recibe un número de secuencia igual a **–1**, el cliente detiene la recepción y concluye la descarga.

3. Interfaz y Reproductor: App.java

La interfaz gráfica implementada con JavaFX permite al usuario iniciar la recepción de la canción mediante un botón. Al activarse, se ejecuta el cliente (cmusica.main) en un hilo separado, lo que evita congelar la interfaz:

```
new Thread(() -> {
    cmusica.main(null);
}).start();
```

Después, la aplicación verifica periódicamente si el archivo ya fue recibido y, cuando lo encuentra, lo reproduce:

```
Media media = new Media(file.toURI().toString());
mediaPlayer = new MediaPlayer(media);
mediaPlayer.play();
```

Además, incluye controles para pausar, reanudar y reiniciar la canción, manipulando el objeto MediaPlayer.

Conclusiones

Núñez Ramírez Valery Aylin

Esta práctica me permitió comprender de forma más profunda el funcionamiento de los datagramas y la diferencia entre UDP y TCP. Implementar manualmente una ventana deslizante me ayudó a visualizar la importancia del control de flujo y de los acuses de recibo. Aunque UDP no garantiza la entrega, es posible lograr un mecanismo confiable con lógica adicional. También reforcé el uso de JavaFX para integrar redes con una interfaz gráfica funcional.

Jimenez Rodriguez Alejandro Martin

Considero que esta práctica fue fundamental para entender que el envío de información multimedia no solo implica mover datos, sino asegurar su integridad. Enfrentar los problemas de pérdida de paquetes y manejar reenvíos me ayudó a comprender procesos que normalmente ocurren “por debajo” en protocolos como TCP. Integrar el cliente con un reproductor también me permitió ver el ciclo completo: desde la transmisión a bajo nivel hasta la experiencia final del usuario.

Bibliografía

- Tanenbaum, A. S. *Computer Networks*. Prentice Hall.
- Oracle. *Java Platform SE Networking Documentation*.
- Oracle. *JavaFX Media API Guide*.
- Kurose, J. & Ross, K. *Computer Networking: A Top-Down Approach*.