Redes de Computadoras

Informe Obligatorio 2 - 2023

Facultad de Ingeniería

Instituto de Computación

Grupo: redes12

Docente: Jorge Visca

Integrantes

Gianfranco Stefanoli - **CI : 4.975.126-4**

Guzman Pieroni - **CI : 4.975.738-7**

Facundo Larrosa - **CI : 5.108.386-9**

Índice

1. Introducción

Se presenta aquí toda la información correspondida al informe del laboratorio 2, incluyendo instrucciones de ejecución, estructura del cliente y servidor, pruebas realizadas sobre las mismas, entre otras cosas.

2. Instrucciones de Ejecución

2.1 Servidor

Para realizar la ejecución del Servidor, en este caso denominado como “Server.py” se debe situar en la carpeta donde se encuentra el archivo, abrir el cmd y escribir el siguiente comando: **python Server.py <ServerIP> <ServerPort> .**
Se aclara que el ***ServerIP*** y ***ServerPort*** identifican al puerto y el ip que el cliente va a utilizar para bindear su conexión TCP.

2.2 Cliente

Para ejecutar el Cliente, se debe ejecutar el archivo “Client.py” dentro de la carpeta del entregable tal y como se especifica debajo. Este recibe tres argumentos que son la IP del servidor de donde se quiere recibir el video, el puerto de este y el puerto VLC dentro de la computadora del cliente que va a recibir los paquetes UDP con el video en tiempo real.

python cliente.py <ServerIP> <ServerPort> <PuertoVLC>.

2.3 Lanzamiento de VLC

Se utiliza el comando:

cvlc -vvv videoplayback.mp4 –sout "#transcode{vcodec=mp4v,acodec=mpga}:rtp{proto=udp, mux=ts, dst=ServerIP, port=<RandomPort>}" --loop --ttl 1

Donde el **ServerIP** y **RandomPort** son el IP y puerto del servidor al cual se le enviaran los datagramas por UDP. Se aclara que ServerIP = 127.0.0.1 (localhost) ya que se estará ejecutando desde un terminal del host donde se ejecuta el servidor.

2.4 Conexión RTP

Desde el cliente se debe acceder a la ruta donde se tiene instalado VLC y ejecutar el comando que hace que se comiencen a recibir los datagramas desde el servidor y se los entrega al cliente en el puerto **<PuertoVLC>** .

El mismo es el siguiente : ***vlc http://IP_Cliente:<Puerto VLC>***

3. Estructura del Sistema

3.1 Interfaz del Sistema

Se define el protocolo **CONTROLSTREAM** presentado en la letra del obligatorio.
El mismo sirve para generar interacción entre el cliente y el servidor TCP en ejecución.
El mismo consta de cuatro comandos particulares que se describen por sí mismos:

- **CONECTAR <PUERTO_DEL_CLIIENTE>\n**
- **INTERRUMPIR\n**
- **CONTINUAR\n**
- **DESCONECTAR\n**

3.2 Servidor

El módulo del servidor comienza abriendo un socket de tipo TCP, bindeado en una ip y puerto recibidos por parámetro.
A partir de allí, el mismo abre dos hilos, uno llamado "*AddConections*" y el otro "*SendData*". El primero se encarga de aceptar conexiones continuamente por parte de los clientes, y dirigirlos al hilo del método "*ClientConection*" donde se reciben comandos de control. Luego en "SendData" se enviarán los datagramas del video a los clientes que corresponda.
Cuando se recibe el comando "CONECTAR", el cliente es añadido a una lista para luego ser atendido. También puede pausar o reanudar los envíos de datagramas con los comandos "INTERRUMPIR" o "CONTINUAR", e incluso cerrar las conexiones si recibe un "DESCONECTAR".
El método "SendData" se encarga de transmitir continuamente los datagramas de video a cada uno de los clientes que se encuentren en la lista y cuya transmisión no se encuentre interrumpida, obteniéndose los mismos desde el flujo de video reproducido por VLC.

3.3 Cliente

El módulo cliente comienza creando un socket TCP.
Luego de esto, el mismo se intenta conectar al socket TCP del servidor con el **ServerIP y ServerPort**. Una vez que la conexión se ha realizado con éxito se abre un hilo con el procedimiento '*consoleData*' el cual recibe como parámetros el socket TCP y el **PuertoVLC**. Este procedimiento genera un bucle infinito que permite introducir comandos en la terminal, los cuales serán enviados al servidor a través de la conexión TCP. Siendo reconocidos sólo los comandos del protocolo **CONTROLSTREAM**

4. Obstáculos

4.1 Port Forwarding

Uno de los obstáculos encontrados se dio al querer realizar una prueba ejecutando al cliente y servidor desde dos direcciones IP distintas.
La conexión TCP no se estaba pudiendo realizar de forma correcta, ya que el servidor no reconoció a ningún cliente.
Se procedió a configurar el reenvío de puertos por parte del router tanto del lado del servidor como del cliente. En particular, el problema se efectuaba en que el router no reenviaba los paquetes recibidos hacia el servidor (o el cliente) ya que eso no estaba configurado.
Luego de realizado lo anteriormente dicho se logró alcanzar el cometido y generar una conexión TCP correcta entre servidor y cliente, permitiendo enviar y recibir los datagramas del video.

4.2 IP en vez de Localhost

Dentro del comando de conexión RTP (***vlc http://<IP_Cliente>:<Puerto VLC>***) se estaba ingresando “localhost” (o 127.0.0.1) en vez de ingresar el Ip del cliente que va a recibir el video. En cierto momento se creía que era lo mismo pero no fue así, ya que el video por parte del cliente no estaba siendo recibido. Esto se debe a que “localhost” no identifica al equipo a nivel de Red, sino de forma local.

4.3 Threads

Se utilizaron hilos (threads) para poder lograr que varios clientes sean atendidos de forma concurrente por el Servidor.
Se usa la librería “***threading***”.
Cada thread es generado cuando se necesita manejar múltiples conexiones simultáneas de manera eficiente. Esto ayuda a atender múltiples clientes y paralelizar tareas como es el caso en protocolos orientados a conexión (TCP)

4.4 Mutex

Otro de los problemas que se logró solventar es el de la mutuoexclusion de la lista de clientes conectada al servidor.
En ese caso se utilizó la librería “***threading***" con el método “***Lock()***” que se encarga de dormir a los threads mientras otro thread está haciendo uso de algún recurso compartido. De esta forma se garantiza la consistencia de los datos y se aumenta la seguridad en los mismos.
Es importante usar la funcionalidad de manera adecuada para evitar bloqueos y asegurarse de que la aplicación siga siendo eficiente y correcta.

4.5 Lista de Clientes

La lista de clientes utilizada mantiene los clientes conectados, y como un atributo de ella aparece un Booleano capaz de identificar si el cliente está interrumpido o no. Cada vez que se desea transmitir datos por parte de un cliente, se recorre toda la lista, lo cual resultaría ineficiente para una lista de gran escala, como punto en contra para la implementación presentada, ya que se recorren todos los clientes, en vez de recorrer solo los activos.