**Laboratorio 3: Actividad 1 – Implementación de servidor y cliente TCP**

F. Bedoya, D. Granada, N. Orjuela, F. Rojas

**Grupo #4**

**Sección #1**

**Infraestructura de Comunicaciones**

**Departamento de Ingeniería de Sistemas y Computación**

**Universidad de Los Andes**

**Bogotá, Colombia**

1. **Hipervínculos**

* [Repositorio con el código de cliente y servidor](https://github.com/coldblade2000/ServidorTCPInfracom/)

En este repositorio también se encuentra el readme con las instrucciones para poner en funcionamiento al cliente y servidor.

* [Capturas de tráfico de las pruebas realizadas](https://uniandes-my.sharepoint.com/personal/f_bedoya_uniandes_edu_co/_layouts/15/onedrive.aspx?id=%2Fpersonal%2Ff%5Fbedoya%5Funiandes%5Fedu%5Fco%2FDocuments%2Fcapturas%20pruebas)
* [Video corto en el que se explica el desarrollo realizado y su funcionamiento](https://youtu.be/Lgb-VMsWUyE)

1. **Requerimientos del Servidor de Transferencia de Archivos**

Para poder implementar el protocolo TCP en el servidor se crearon dos clases java: MainServer.java para correr el servidor en sí y ThreadServer.java para manejar las conexiones.

ThreadServer tiene como atributos un CyclicBarrier, el hash del archivo escogido por el usuario, el socket del cliente, un DataOutputStream, un DataInputStream, una instancia de la clase MainServer, el path del archivo a escoger y el id del cliente.

Para iniciar la transferencia primero se les pregunta a los clientes conectados si están listos para recibir archivos, si lo están, se verifica que haya suficientes clientes conectados para enviar el archivo, si no hay, los pone a esperar, si hay, llama al método que envía los archivos y verifica que los clientes respondan que recibieron el archivo y logea la respuesta en un log.

Texto

Descripción generada automáticamente

**Figura 1. Proceso de envío de archivos por parte de ThreadServer.**

El otro método de la clase es el de enviar archivo, en donde encuentra el archivo a enviar, manda el hash del archivo, su tamaño y luego manda por pedazos el archivo. Mientras ocurre esto se va contabilizando cuando dura esta transferencia.



**Figura 2. Método de envío de archivos de ThreadServer.**

MainServer tiene como atributos un BufferedWriter, un FileWriter, una lista de ThreadServer, el socket del servidor, el socket del cliente, el path del archivo y el hash del archivo.

En el constructor de esta clase se busca el archivo escogido por el usuario y se calcula su hash. En el método main se crea el log de la sesión y se le pregunta al usuario cuantos clientes quiere que se conecten y cual archivo quiere mandar, para luego iniciar el servidor.

Texto

Descripción generada automáticamente

**Figura 3. Inicio del proceso de envío de archivos en MainServer.**

Cuando se inicia el servidor se inicia un nuevo ThreadServer por cliente, y finalmente hacer un thread.join() y se espera a que todos los ThreadServer terminen y así cerrar todos los socket.

Texto

Descripción generada automáticamente

**Figura 4. Conexión de los ThreadServer con los clientes.**

1. **Requerimientos del cliente TCP**

Para poder implementar el protocolo TCP en el cliente se crearon dos clases java: main.java para correr el cliente en sí y clientThread.java para simular los múltiples clientes que van a hacerle peticiones al servidor.

ClientThread tiene como atributos su id y la cantidad de clientes para poder crear el nombre de los archivos de prueba. El puerto en el que se realiza la transferencia es el 25505.

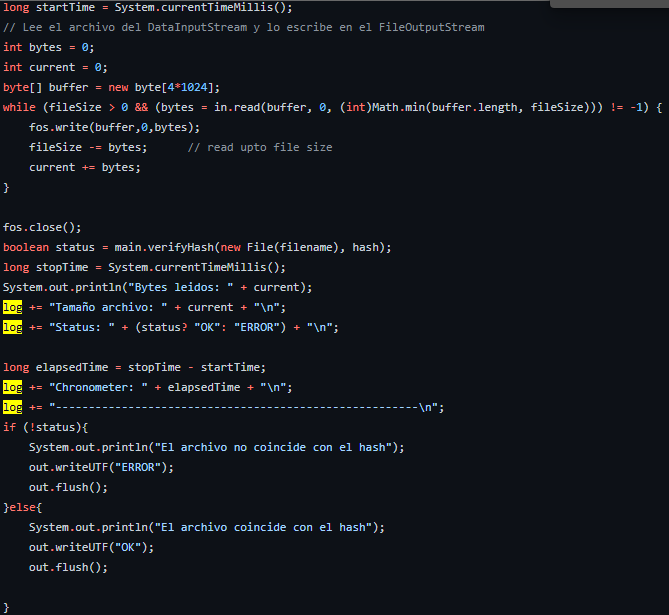
Para iniciar la transferencia, luego de que el cliente se conecte con el servidor se hace un handshake preguntándole al cliente si está listo para empezar a recibir archivos, si lo está, se recibe primero el hash del archivo y el tamaño del archivo.

Imagen que contiene interior, televisión, monitor, pantalla

Descripción generada automáticamente

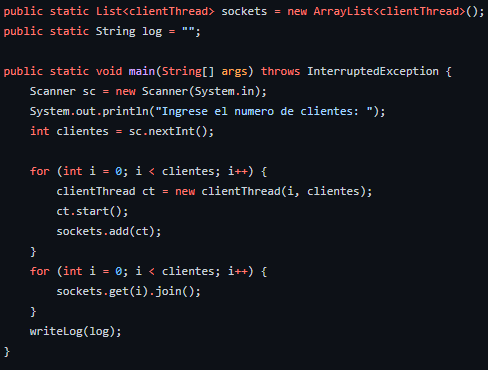
**Figura 5. Proceso de handshake por parte del cliente.**

Luego se empieza a contabilizar el tiempo y se empieza a leer el archivo transferido por bloques. Ya con el archivo completamente leído se hashea el archivo, se para la contabilización de tiempo y se verifica el hash recibido con el calculado y se imprime error si el archivo no coincide con el hash. Finalmente, se logean todos los procedimientos en los archivos log correspondientes.



**Figura 6. Lectura por parte del cliente del archivo enviado.**

Main tiene como atributos la lista de clientThreads y el log. En el método main se crean los clientThreads solicitados por el usuario y se conectan con el servidor. Los otros métodos son para escribir los logs que mandan los thread y la verificación del hash recibido por el servidor.



**Figura 7. Creación de los clientes.**

1. **Análisis de desempeño del servicio**
2. Pruebas Individuales
   1. Archivo 100MB - 1 Cliente

|  |  |  |  |
| --- | --- | --- | --- |
| Archivo | Éxito | Tiempo medido por cliente (ms) | Tasa cliente (B/ms) |
| 100MB | OK | 808 | 129 774.2574 |
| Puerto cliente | Bytes cliente-servidor | Tiempo medido por servidor (ms) | Puerto servidor |
| N/A | 104 857 600 | 404 | 25505 |

Como se puede observar, la prueba fue exitosa. El tiempo medido por el cliente fue mayor al tiempo medido por servidor debido a que en el cliente se contabiliza la lectura y la verificación del hash enviado por el servidor.

* 1. Archivo 100MB - 5 Clientes

|  |  |  |  |
| --- | --- | --- | --- |
| Archivo | Éxito | Tiempo medido por cliente (ms) | Tasa cliente (B/ms) |
| 100MB | OK | 2004.2 | 52 318.93025 |
| Puerto cliente | Bytes cliente-servidor | Tiempo medido por servidor (ms) | Puerto servidor |
| N/A | 104 857 600 | 1454.6 | 25505 |

Como se puede observar, la prueba fue exitosa, acá la tasa cliente es mucho menor que la obtenida al inicio, esa tasa es casi 2.5 veces mayor a esta.

* 1. Archivo 100MB - 10 Clientes

|  |  |  |  |
| --- | --- | --- | --- |
| Archivo | Éxito | Tiempo medido por cliente (ms) | Tasa cliente (B/ms) |
| 100MB | OK | 3466.2 | 30 251.45693 |
| Puerto cliente | Bytes cliente-servidor | Tiempo medido por servidor (ms) | Puerto servidor |
| N/A | 104 857 600 | 2870 | 25505 |

Como se puede observar, la prueba fue exitosa. Acá la tasa cliente disminuyó y es casi el 25% del valor de la primera tasa obtenida.

* 1. Archivo 250MB - 1 Cliente

|  |  |  |  |
| --- | --- | --- | --- |
| Archivo | Éxito | Tiempo medido por cliente (ms) | Tasa cliente (B/ms) |
| 250MB | OK | 2033 | 128 944.4171 |
| Puerto cliente | Bytes cliente-servidor | Tiempo medido por servidor (ms) | Puerto servidor |
| N/A | 262 144 000 | 983 | 25505 |

Como se puede observar, la prueba fue exitosa. El valor de la tasa cliente obtenida es bastante similar a la obtenida con el archivo de 100MB, aunque los tiempos medidos por cliente y servidor son evidentemente mayores.

* 1. Archivo 250MB - 5 Clientes

|  |  |  |  |
| --- | --- | --- | --- |
| Archivo | Éxito | Tiempo medido por cliente (ms) | Tasa cliente (B/ms) |
| 250MB | OK | 4481 | 58 501.2274 |
| Puerto cliente | Bytes cliente-servidor | Tiempo medido por servidor (ms) | Puerto servidor |
| N/A | 262 144 000 | 3299.8 | 25505 |

Como se puede observar, la prueba fue exitosa. Acá ocurre lo mismo que ocurrió con el archivo de 100MB y 5 clientes, es más, se obtuvo también una tasa cliente casi idéntica a la obtenida en el literal b.

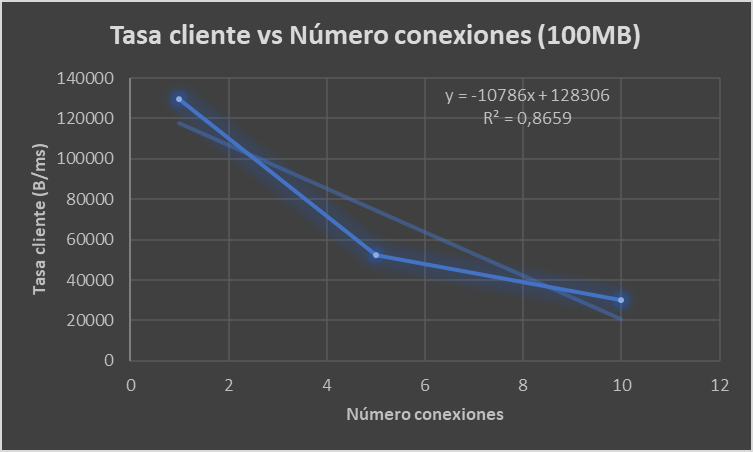
* 1. Archivo 250MB - 10 Clientes

|  |  |  |  |
| --- | --- | --- | --- |
| Archivo | Éxito | Tiempo medido por cliente (ms) | Tasa cliente (B/ms) |
| 250MB | OK | 8815.6 | 29 736.3764 |
| Puerto cliente | Bytes cliente-servidor | Tiempo medido por servidor (ms) | Puerto servidor |
| N/A | 262 144 000 | 7365.5 | 25505 |

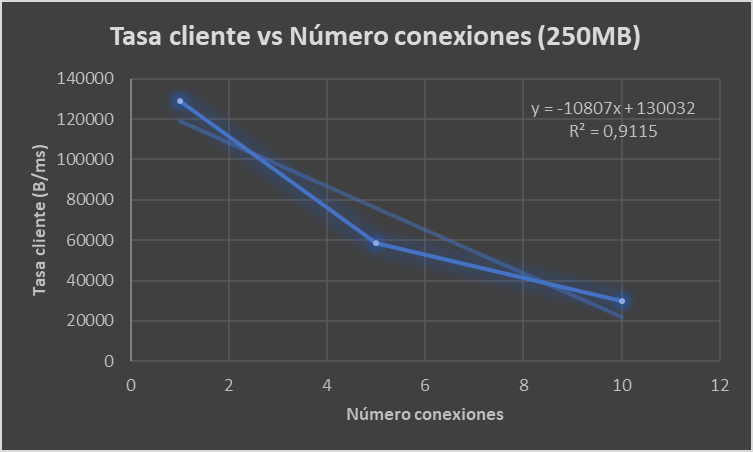
Como se puede observar, la prueba fue exitosa. Esta última prueba sigue el patrón y se obtuvo una tasa cliente bastante similar a la obtenida en el literal c.

1. Análisis global

Lo observado anteriormente se puede evidenciar en las figuras 8 y 9, en donde se tienen tasas con valores bastantes similares en ambos casos, y también en donde la tasa de transferencia de los clientes tiene una relación inversamente proporcional con el número de conexiones: por cada conexión se disminuye en aproximadamente 10 800 Bytes por milisegundo la tasa de transferencia. Esto es debido a que el ancho de banda tiene que ser compartido por más conexiones, por lo que a cada conexión le tocaría menos ancho de banda.



**Figura 8. Tasa de transferencia del cliente en función del número de conexiones (archivo 100MB).**



**Figura 9. Tasa de transferencia del cliente en función del número de conexiones (archivo 250MB).**