**--------------APLICACIONES EN COMUNICACIONES EN RED------------**

**PRÁCTICA 1:**

Servicio de transferencia de archivo

**Alumno:**

Meza Vargas Brandon David

**Grupo:**

3CM16

**Profesor:**

Moreno Cervantes Axel Ernesto

**índice**

[**Introducción** 3](#_Toc97233754)

[**Desarrollo** 4](#_Toc97233755)

[**Cliente** 4](#_Toc97233756)

[**Función chooser** 4](#_Toc97233757)

[**Función sendFiles** 5](#_Toc97233758)

[**Función downloadFiles** 7](#_Toc97233759)

[**Servidor** 7](#_Toc97233760)

[**Función connect** 7](#_Toc97233761)

[**Función recieveFiles** 9](#_Toc97233762)

[**Función sendFiles** 10](#_Toc97233763)

[**Función chooser** 11](#_Toc97233764)

[**Pruebas de funcionamiento** 11](#_Toc97233765)

[**Conclusiones** 17](#_Toc97233766)

[**Bibliografía** 17](#_Toc97233767)

**índice de Ilustraciones**

[Ilustración 1. Función chooser del cliente. 5](#_Toc97233785)

[Ilustración 2. Función sendFiles del cliente. 6](#_Toc97233786)

[Ilustración 3. Función downloadFiles(); 7](#_Toc97233787)

[Ilustración 4. Función connect del servidor() 8](#_Toc97233788)

[Ilustración 5. Función recieveFiles 9](#_Toc97233789)

[Ilustración 6. Función sendFiles del servidor 10](#_Toc97233790)

[Ilustración 7. Función chooser del servidor. 11](#_Toc97233791)

[Ilustración 8. Interfaz usuario. 12](#_Toc97233792)

[Ilustración 9. Listado de archivos. 12](#_Toc97233793)

[Ilustración 10. Mandando archivos al servidor 13](#_Toc97233794)

[Ilustración 11. Listado de archivos enviados al servidor. 13](#_Toc97233795)

[Ilustración 12. Descargando archivos. 14](#_Toc97233796)

[Ilustración 13. Listado de archivos descargados por el cliente. 14](#_Toc97233797)

[Ilustración 14. Selección de archivo para eliminar. 15](#_Toc97233798)

[Ilustración 15. Listando archivos. 15](#_Toc97233799)

[Ilustración 16. Archivo para eliminar del cliente. 16](#_Toc97233800)

[Ilustración 17. Listando archivos de nuevo 16](#_Toc97233801)

[Ilustración 18. Activando algoritmo de Nagle 16](#_Toc97233802)

# **Introducción**

Los sockets permiten conectar dos programas en red para que se puedan intercambiar datos. Los sockets están basados en una arquitectura cliente/servidor en donde el servidor esta continuamente a la escucha y a la espera de que alguien se quiera comunicar a él.

Los sockets de flujo son un servicio orientado a la conexión, donde los datos se transfieren sin encuadrarlas en registros o bloques, estos están basados en el protocolo TCP brindando comunicaciones fiables.

En Java las comunicaciones TCP se realizan utilizando la clásica abstracción de socket.

El envío de archivos a través de la red es una característica importante para la gran mayoría de las aplicaciones que hoy día se utilizan (blogs, redes sociales, mensajería instantánea, Declaración de impuestos, educación en línea, etc.), sin embargo, no todas las aplicaciones disponibles permiten el envío de archivos de gran tamaño (p.e. El correo electrónico no permite enviar archivos de más de 10 o 20 MB). Esto hace necesario el desarrollo de aplicaciones que permitan transferir archivos sin importar el tamaño de éstos.

En la presente práctica se realizará una aplicación que permita el envío y descarga de archivos de un servidor a un cliente y viceversa empleando los sockets de flujo en el lenguaje de programación Java.

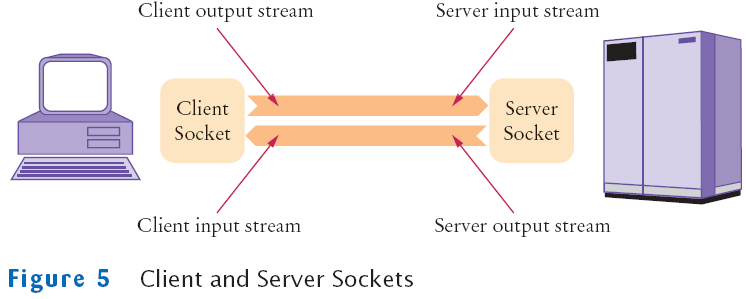


Ilustración . Sockets de flujo

# **Desarrollo**

Para el desarrollo de esta práctica fue necesario implementar dos partes, una de parte del cliente que se conectará al servidor para transferir archivos así como recibirlos y la del servidor que se mantiene a la escucha del cliente para saber cuando mandar o recibir archivos.

## **Cliente**

### **Función chooser**

Esta función se encarga de crear la carpeta donde se almacenarán los archivos del cliente, así como de verificar si lo que quiere hacer el cliente es mandar archivos al servidor o descargar archivos del mismo servidor, esto se determina por medio del parámetro que recibe mandado desde la interfaz del usuario.

Texto

Descripción generada automáticamente con confianza media

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración . Función chooser del cliente.

### **Función sendFiles**

Esta función se encarga de mandar los archivos al servidor, se hace un ciclo que recorre todos los archivos seleccionados por el usuario para mandar al servidor, por cada archivo verificamos si es una carpeta o un simple archivo, en caso de ser una carpeta se manda a llamar la misma función con recursión para crear la carpeta en el servidor y mandar los archivos dentro de ella.

En caso de ser un archivo se abre un socket que se conecta al servidor para indicar al servidor que se esta enviando un archivo y se abre uno mas para indicar los datos del archivo que se esta enviado como su tamaño, nombre y si es una carpeta. Finalmente con un ciclo while vamos mandando los bytes del archivo hasta mandarlo por completo.

Texto

Descripción generada automáticamente con confianza media

Imagen que contiene Texto

Descripción generada automáticamente

Ilustración . Función sendFiles del cliente.

### **Función downloadFiles**

Por último tenemos la función downloadFiles(), en esta vamos descargando los archivos del servidor que el usuario selecciono para descargar, este función se ejecuta en un ciclo infinito hasta que el servidor indique que ya se enviaron todos los datos.

El proceso es similar a la función de enviar datos del cliente, la diferencia es que aquí vamos leyendo en lugar de escribir los bytes de los archivos mandados por el servidor.

Tabla

Descripción generada automáticamente

Ilustración . Función downloadFiles();

## **Servidor**

Por parte del servidor las funciones de enviar y recibir datos son muy similares a las del cliente con la diferencia que en estas se aceptan conexiones de socket por parte del cliente, por esto se limitara a colocar el código de cada función.

### **Función connect**

En esta función se hace un ciclo infinito donde un socket siempre estará abierto para esperar datos por parte del cliente, se usaron dos, uno para recibir datos por el puerto 8000 y otr por el puerto 9000 para evitar confusiones y saturaciones al enviar datos por los sockets.

En esta función recibimos una operación que se envía por parte del cliente cuando este indica lo que quiere hacer, en este caso si se manda un 1 el servidor se prepara para recibir archivos, de otro modo manda datos al cliente.

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración . Función connect del servidor()

### **Función recieveFiles**

Imagen que contiene Escala de tiempo

Descripción generada automáticamente

Ilustración . Función recieveFiles

### **Función sendFiles**

Texto

Descripción generada automáticamente con confianza media

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración . Función sendFiles del servidor

### **Función chooser**

Esta función se manda a llamar cuando vamos a mandar archivos solicitados al cliente, esta se encarga de abrir el seleccionador de archivos dentro de la carpeta del servidor para que el usuario pueda seleccionar que archivos quiere descargar.

Texto

Descripción generada automáticamente

Ilustración . Función chooser del servidor.

## **Pruebas de funcionamiento**

En la siguiente ilustración podemos ver la interfaz del usuario.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración . Interfaz usuario.

Si listamos los archivos podemos ver que no hay nada ni en el cliente ni en el servidor (ver ilustración 10).

Escala de tiempo

Descripción generada automáticamente

Ilustración . Listado de archivos.

Primeramente vamos a mandar archivos al servidor, estos serán carpetas con datos a dentro.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración . Mandando archivos al servidor

Una vez mandado podemos listar los archivos para verificar que se enviaron correctamente como se muestra en la siguiente figura.

Escala de tiempo

Descripción generada automáticamente

Ilustración . Listado de archivos enviados al servidor.

Ahora procedamos a descargar esos archivos descargados al cliente.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración . Descargando archivos.

Al ser descargados podemos listarlos de igual forma para verificar que se descargaron correctamente.

Escala de tiempo

Descripción generada automáticamente

Ilustración . Listado de archivos descargados por el cliente.

Ahora eliminemos estos archivos, primeramente al seleccionar la parte de eliminar archivos del servidor se nos abre la ruta del servidor y podemos seleccionar lo que queremos eliminar como se ve en la ilustración.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración . Selección de archivo para eliminar.

Al eliminarlo podemos listar de nuevo y veremos que ya no está.



Ilustración . Listando archivos.

Haremos lo mismo en el cliente pero eliminando un archivo distinto.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración . Archivo para eliminar del cliente.

Si listamos los archivos de nuevo vemos que ya no aparece

Tabla, Escala de tiempo

Descripción generada automáticamente

Ilustración . Listando archivos de nuevo

Se hicieron pruebas activando el algoritmo con la siguiente línea de código:



Ilustración . Activando algoritmo de Nagle

Sin embargo, no se notaron en demasía los cambios que este algoritmo producía por lo tanto los efectos son los mismo al activarlo o no dentro de nuestro código.

# **Conclusiones**

A través de la práctica realizada de envío de archivos usando sockets de flujo pude entender de mejor manera el funcionamiento de estos y la importancia de estos, pues lo presentado en esta práctica representa una de las muchas aplicaciones que se les puede dar a estos sockets, pues aquí solo vimos el funcionamiento elemental de como se pueden enviar archivos a través de un socket para llegar de un cliente a un servidor que se encuentran conectados y viceversa.

Durante el desarrollo de la práctica se presentaron algunas problemáticas, principalmente en el envío y recibo de carpetas ya que estos llegaban incompletos o se cerraba la conexión entre los sockets ya que nunca se terminaban de enviar o recibir y esto causaba una saturación en la conexión, esto se resolvió usando recursión, pues de esta forma se aseguró el envío correcto y completo de los datos.

Cabe mencionar que fue de mucha ayuda usar un entorno gráfico, pues este le da una mejor vista a la aplicación y en caso de que usuarios reales la usen se les facilitará el entendimiento y funcionamiento de esta.

Finalmente considero que fue una buena práctica para poner en marcha lo aprendido en las sesiones referentes a sockets de flujo, pues esta práctica abarco todo lo repasado en las clases.

# **Bibliografía**

1. Buap. (2020). “Sockets”. Obtenido de : <http://www.cs.buap.mx/~mtovar/doc/PCPA/Sockets.pdf>