Tarea #1
Redes de Computadores

Profesor: Pablo Serrano
Ayudantes: Ignacio Cofré ignacio.cofre.14@sansano.usm.cl
Javier Ramos javier.ramos.14@sansano.usm.cl
Benjamín Riquelme benjamin.riquelme@sansano.usm.cl
Catherin Vargas catherin.vargas.13@sansano.usm.cl

27 de marzo de 2019

1. Objetivos

- Conocer en detalle la arquitectura cliente servidor.
- Aprender a implementar un cliente y servidor utilizando Sockets en Java.
- Simular el comportamiento del protocolo FTP.

2. Introducción

Los sockets son un mecanismo que nos permiten establecer un enlace entre dos programas que se ejecutan independientes el uno del otro (generalmente un programa cliente y un programa servidor) Java por medio de la librería java.net nos provee dos clases: Socket para implementar la conexión desde el lado del cliente y ServerSocket que nos permitirá manipular la conexión desde el lado del servidor.

La arquitectura cliente-servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, quien le da respuesta.

3. Tarea

La primera parte de la tarea consiste en programar un servidor que sea capaz de:

- Enlazarse a un socket y escuchar por conexiones entrantes en los puertos a especificar.
- Cuando este servidor inicie se debe crear un Thread Pool.
- Por cada conexión entrante asignarle un thread para su ejecución.

El servidor deberá procesar las solicitudes de conexión realizando un handshake en tres pasos, descrito de manera resumida en la siguiente figura:

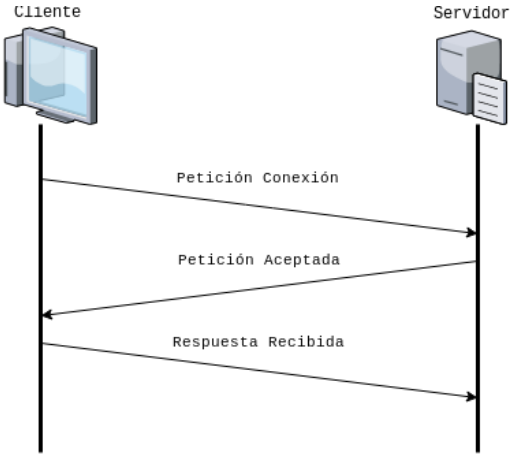

Figura 1: Protocolo de handshake en tres pasos.

Una vez establecida la conexión, el servidor debe ser capaz de realizar las siguientes acciones:

- `ls`: listar el contenido del directorio actual.
- `get nombre_archivo`: descargar un archivo desde el servidor al directorio local.
- `put nombre_archivo`: subir un archivo desde el directorio local al servidor.
- `delete nombre_archivo`: eliminar un archivo en el servidor.

El servidor debe generar un archivo `log.txt`, en el cual mantiene registro de las direcciones IP de los clientes que se conectan a él, además de las acciones realizadas durante dicha conexión. Se presenta a continuación un ejemplo de `log.txt`:

```
DATETIME              EVENT       DESCRIPTION
23-03-2019 20:20:00   connection  192.168.1.45 conexion entrante
23-03-2019 20:23:00   command     192.168.1.45 ls
23-03-2019 20:23:10   connection  192.168.1.32 conexion entrante
23-03-2019 20:23:20   error       conexion rechazada por 192.168.1.32
23-03-2019 20:23:30   response    servidor envia respuesta a 192.168.1.45
23-03-2019 20:25:30   command     192.168.1.45 get main.py
23-03-2019 20:25:40   response    servidor envia respuesta a 192.168.1.45
23-03-2019 20:26:20   command     192.168.1.45 delete main.py
23-03-2019 20:26:35   response    servidor envia respuesta a 192.168.1.45
23-03-2019 20:28:13   command     192.168.1.45 put main_finalfinal.py
23-03-2019 20:28:21   response    servidor envia respuesta a 192.168.1.45
...
```

3.1. Malla Antigua (ILI)

Deberán agregar método de autenticación, esto significa que al intentar conectarse al servidor se debe pedir usuario y contraseña (los cuales deben estar en un archivo usuarios.txt en el servidor, puede haber más de un usuario) si las credenciales coinciden se puede acceder sin problemas, en caso contrario se debe reintentar o cerrar la conexión.

3.2. Bonus - 10 puntos

Se le debe agregar una funcionalidad extra al servidor, la cual consiste en mantener el estado de la descarga de archivos. Esto consiste en que el cliente pueda pausar una descarga en curso, y posteriormente reanudarla sin perder el progreso previo. Considere que para poder cumplir con esto, se debe identificar a los usuarios de alguna forma.

4. Restricciones

- No pueden usar la librería de Apache, pero sí utilizar sus interfaces y métodos como guía, también pueden crear sus propias interfaces e implementaciones.
- No se puede usar ninguna implementación de java para la generación del ThreadPool. Ej: Executor, ExecutorService. Ahora sin embargo, pueden ver esas implementaciones e intentar entender cómo funcionan por debajo.
- La única librería que está permitido usar (aparte de socket de Java) es una librería para manejar JSON.

5. Consideraciones

- Consultas sobre la tarea se deben realizar en moodle o enviar un correo a **catherin.vargas.13@sansano.usm.cl** o **benjamin.riquelme@sansano.usm.cl**

6. Reglas de entrega

- La tarea se realiza en **grupos de 2 personas**.
- La fecha de entrega es el día **12 de Abril a las 23:55**.
- La entrega debe realizarse a través de Moodle, en un archivo comprimido **ZIP** y debe llevar el nombre **T1-Rol1-Rol2**.
- Debe entregar todos los archivos fuente necesarios para la correcta ejecución. Recuerde que el código debe estar identado, comentado, sin warnings y sin errores.
- Se aplicará un descuento de 5 puntos al total de la nota por cada Warning, Error o Problema de ejecución.
- Debe entregar un **MAKEFILE** o similar para cada uno de los componentes que utilizará.
- Debe entregar un **README** con nombre y rol de cada integrante, además de la información necesaria para ejecutar los archivos. Se piden instrucciones resumidas, no un informe o testamento.
- No se aceptan entregas que no puedan ser ejecutadas desde una consola de comandos. Incumplimiento de esta regla significa **nota 0**.
- Cada hora o fracción de atraso, se penalizará con un descuento de **10 puntos**.
- Copias serán evaluadas con **nota 0**.