Práctica 5

Desarrollo de un cliente y un servidor básico sobre TCP en Java. El servidor ofrecerá un servicio básico de capitalización de textos.

Tarea 1: Especificación del Servidor (clase ServidorTCP.java)

Las especificaciones del servidor:

- El puerto por el que recibirá peticiones será pasado como argumento en la línea de comandos. Por ejemplo: `java ServidorTCP 12345`
- Una vez establecida la conexión con un cliente, esperará a recibir textos a modificar (léalo con `readLine`) desde el socket conectado.
- La respuesta del servidor dependerá de lo que reciba:
  - Si recibe `END` devolverá `OK` y cerrará la conexión.
  - En otro caso devolverá el texto capitalizado.
- Una vez cerrada la conexión, el servidor volverá a esperar una nueva petición de conexión y servicio.
- El servidor deberá informar por la salida estándar (pantalla) de su estado en cada momento.
- El servidor debe poder recuperarse (liberar correctamente los recursos y volver a recibir a otro cliente) si es cliente finaliza la conexión de forma incorrecta.
- El servidor sólo puede tener un cliente en espera (**la cola de clientes pendientes debe ser 1**).

Tarea 2: Especificación del Cliente (clase ClienteTCP.java)

Las especificaciones del cliente:

- La dirección IP y puerto del servidor al que debe conectarse el cliente se le pasará como argumento en la línea de comandos. Por ejemplo:
  `java ClienteTCP 192.168.1.2 12345`
- Una vez conectado el cliente, deberá pedir de forma continua al usuario el texto a capitalizar (una línea de texto) y la enviará el servidor (usando `println`).
- Tras cada envío del texto a tratar, el cliente deberá esperar la respuesta del servidor que contendrá el texto modificado.
- Cuando el usuario quiera terminar escribirá por teclado el valor `TERMINAR`.
- Cuando el cliente detecte que el usuario desea terminar enviará el texto `END`, esperará la respuesta del servidor (`OK`) y cerrará la conexión.
- Durante toda la ejecución el cliente debe informar al usuario (escribiendo por pantalla) su estado (por ejemplo: `Conectado a 192.168.1.2:12345, Esperando la respuesta...`).
- Si el cliente envía datos y se encuentra la conexión cerrada, **cierra de forma ordenada el cliente**.

Tarea 3: Captura de trazas

Simule los siguientes comportamientos tomando con Wireshark la traza del tráfico generado. Si el cliente y el servidor están en la misma máquina, use como IP del servidor la de `loopback` (`127.0.0.1`). Como interfaz para capturar debe usar el interfaz denominado `Adapter for loopback traffic capture`.

Comportamiento TCP1 (traza 1 – **p5e1-4.pcapng**):

- Inicie el servidor y posteriormente el cliente.
- En el cliente envíe un único mensaje y posteriormente escriba `TERMINAR` para finalizarlo.

Comportamiento TCP2 (traza 2 – **p5e5.pcapng**):

- Sin tener activo ningún servidor intente iniciar el cliente.

Comportamiento TCP3 (traza 3 – **p5e6-7.pcapng**):

- Inicie el servidor.
- Posteriormente arranque 3 clientes que intenten conectarse hacia ese servidor.
- Escriba `TERMINAR` en los clientes que han logrado conectarse para finalizar los envíos.

Tarea 4: Análisis de nuestro protocolo en TCP

Responda a las siguientes preguntas usando las trazas capturadas anteriormente.

Usando la traza TCP1 (**p5e1-4.pcapng**):

Ejercicio 1. Identifique una trama de la comunicación y use la opción "`Follow TCP stream`" para ver el intercambio de información entre cliente y servidor. ¿Cuál es el puerto que usa el cliente? ¿Y el servidor?

Ejercicio 2. ¿Cuál es el número de secuencia real (absoluto) que se usa el cliente TCP hacia el servidor? ¿Y las respuestas del servidor al cliente?

Ejercicio 3. Indique los segmentos relacionados con las siguientes actividades y qué métodos de `Socket` y `ServerSocket` son responsables del intercambio de estos segmentos:

a) Inicialización de la conexión.
b) Envío de datos.
c) Finalización de la conexión.

Ejercicio 4. ¿Cuántos números de secuencia se consumen en cada lado (cliente y servidor) durante el inicio y cierre de la conexión?

Usando la traza TCP2 (**p5e5.pcapng**):

Ejercicio 5. ¿Recibe algún tipo de respuesta el intento de conexión del cliente? En caso afirmativo ¿tiene alguna característica especial?

Usando la traza TCP3 (**p5e6-7.pcapng**):

Ejercicio 6. ¿Se logran conectar los 3 clientes? En caso de alguno no se haya podido conectar, ¿se le indica de alguna forma que la cola está llena?

Ejercicio 7. ¿Los clientes en espera (es decir los que están en la cola) tiene inicializada la conexión o esa inicialización se hace cuando se sacan de la cola (con el método `accept`)?