Práctica 4

Desarrollo de un cliente y un servidor básico sobre TCP en Java. El servidor ofrecerá la funcionalidad de cifrar texto. Para ello, el cliente le enviará una clave (número natural) y una línea de texto y el servidor cifrará el mensaje usando cifrado César.

Tarea 1: Especificación del Cliente (clase Cliente.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 comando. Por ejemplo:
  `java Cliente 192.168.1.2 12345`
- Tras conectarse al servidor, recibirá del mismo un mensaje indicando que inició la conexión al servicio.
- Una vez conectado el cliente, deberá pedir de forma continua pares de líneas de texto al usuario (la primera es un número y la segunda un texto). La clave envíela al servidor con `print(clave+"\r\n")` tras leerlo y use `println` para enviar el texto.
- Tras cada envío, el cliente deberá esperar la respuesta del servidor (que contendrá una cadena con el texto codificado. Por ejemplo, si el cliente envía `2` y `hola` el servidor responderá `jqnc`).
- Se supone que los datos introducidos por el usuario son correctos y el programa no necesita comprobar ninguna clave (siempre serán números naturales) ni texto (no contendrán tildes ni ñ).
- Cuando el usuario quiera terminar escribirá por teclado como clave el valor `0`. Note que cuando el usuario meta como clave un `0`, no debe leer el texto.
- Cuando el cliente detecte que el usuario desea terminar enviará al servidor la clave `0` (usando `println`), 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 2: Especificación del Servidor (clase Servidor.java)

Las especificaciones del servidor:

- El puerto por el que recibirá peticiones será pasado como argumento en la línea de comando. Por ejemplo: `java Servidor 12345`
- Una vez establecida la conexión con un cliente, enviará al cliente la cadena: "`Bienvenido al servicio de cifrado`" (sin comillas).
- Luego esperará a recibir pares de datos, clave y texto a cifrar, (dos líneas de texto, lea cada una con `readLine`) desde el socket conectado.
- La primera línea la convertirá a número y lo usará como clave para cifrar la segunda línea de texto (en el anexo se muestra información de cómo hacer el cifrado).
- El servidor cuando reciba como clave `0`, enviará al cliente la cadena `OK` y cerrará la conexión.
- Una vez cerrada la conexión, el servidor volverá a esperar una nueva petición de conexión y servicio.
- Al igual que el cliente, el servidor deberá informar por la salida estándar (pantalla) de su estado en cada momento.
- El servidor sólo puede tener un cliente en espera (**la cola de clientes pendientes debe ser 1**).
- Si el servidor no recibe datos de un cliente conectado en 40 segundos (use el método `setSoTimeout()` antes de la recepción), cierra el socket conectado.

Tarea 3: Captura de trazas

Póngase de acuerdo con un compañero y simule los siguientes comportamientos tomando con Wireshark la traza del tráfico generado.

Comportamiento 1 (traza 1 – **p4e1-7.pcapng**):

- Inicie el servidor en un equipo y posteriormente el cliente en otro diferente.
- En el cliente envíe un único mensaje y posteriormente escriba `0` para finalizarlo.

Comportamiento 2 (traza 2 – **p4e8.pcapng**):

- Sin tener activo ningún servidor intente iniciar el cliente.

Comportamiento 3 (traza 3 – **p4e9-10.pcapng**):

- Inicie el servidor en un equipo y posteriormente el cliente en otro.
- Espere más de 40 segundos desde la inicialización del cliente y luego intente escriba el mensaje de `0` para finalizar los envíos.

Comportamiento 4 (traza 4 – **p4e11-12.pcapng**):

- Inicie el servidor en un equipo.
- Posteriormente arranque 3 clientes que intenten conectarse hacia ese servidor (todos los clientes pueden estar en la misma máquina, lanzados desde diferentes terminales).
- Escriba `0` en los clientes que han logrado conectarse para finalizar los envíos.

Tarea 4: Análisis de nuestro protocolo de ECO en TCP

Responda a las siguientes preguntas usando las trazas capturadas anteriormente.

Usando la traza 1 (**p4e1-7.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. Muestra una captura de pantalla con dicha información.

Ejercicio 2. Los mensajes enviados por el cliente (clave y texto), ¿van en el mismo segmento TCP o en segmentos separados? ¿Por qué?

Ejercicio 3. ¿Cuál es el puerto que usa el cliente? ¿Y el servidor? ¿En qué campos de la cabecera del segmento TCP están cada uno?

Ejercicio 4. ¿Cuál es el número de secuencia que se usa el cliente TCP hacia el servidor? ¿Y las respuestas del servidor al cliente?

Ejercicio 5. 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 6. ¿Cuántos números de secuencia se consumen en cada lado (cliente y servidor) durante el inicio y cierre de la conexión?

Ejercicio 7. Observe el tamaño de la ventana deslizante del cliente y del servidor en cada segmento de envío de datos. ¿Cambia este valor? ¿Qué valores toma en el cliente y en el servidor?

Usando la traza 2 (**p4e8.pcapng**):

Ejercicio 8. ¿Recibe algún tipo de respuesta el intento de conexión del cliente? En caso afirmativo ¿tiene alguna característica especial?

Usando la traza 3 (**p4e9-10.pcapng**):

Ejercicio 9. ¿Qué ocurre cuando pasan los 40 segundos en el servidor? ¿Qué segmentos envía el servidor y qué recibe por respuesta por parte del cliente?

Ejercicio 10. ¿Intenta el cliente enviar el `0` que escribimos por teclado? En caso afirmativo, ¿qué respuesta recibe y qué significa esa respuesta?

Usando la traza 4 (**p4e11-12.pcapng**):

Ejercicio 11. ¿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 12. ¿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`)?

Anexo: El cifrado César

El cifrado César, o cifrado por desplazamiento, es una técnica muy simple de cifradoConsiste en reemplazar cada letra del texto original por otra letra que se encuentra un número fijo (clave) de posiciones más adelante en el alfabeto. Por ejemplo, con un clave de 3, la A sería sustituida por la D (situada 3 lugares a la derecha de la A), la B sería reemplazada por la E, etc. Cuando pase de la letra z o la Z, vuelve a la a o A, respectivamente. Por ejemplo, con clave 3, la Z se convierte en C.

El cifrado considerado por el servidor, solo considerará claves positivas y menores que 26, y en el texto solo hay cifrar las letras normales (mayúsculas y minúsculas). El pseudocódigo sería:

```
FUNCION encriptar(texto, clave): STRING
  numLetras = 'Z' – 'A' + 1
  clave = clave % numLetras
  res = ""
  PARA cada letra c en texto HACER
    SI c es una letra normal ENTONCES
      SI (c es mayúscula Y c + clave > 'Z') O
         (c es minúscula Y c + clave > 'z') ENTONCES
            c = c – numLetras
      FINSI
      c = c + clave
    FINSI
    Concatenar a res la letra c
   FINPARA
   DEVOLVER res
FINFUNCION
```

Nota sobre la memoria

- Si elabora la memoria en Word, se aconseja utilizar la plantilla proporcionada para la práctica. En cualquier caso, la memoria debe contener toda la información que se pide en la plantilla y seguir su estructura.
- La memoria de esta práctica se entregará en conjunto a la memoria de la 5.
- Cuando la práctica consista en el desarrollo de un código, **en la memoria se explicará el esquema del mismo**, únicamente detallando (y explicando) las partes más significativas del mismo (**incluyendo todas las sentencias relacionadas con sockets**).
- Dicha memoria debe constar de una portada donde se indique el conjunto de prácticas que incluye la memoria, así como todos los datos del alumno.
- La memoria de cada práctica debe empezar en una nueva página.
- No es necesario copiar el enunciado completo de la práctica pero sí debe copiarse el enunciado de cada ejercicio antes de indicar su respuesta. Debe utilizarse algún sistema de estilos que permita distinguir lo que es el enunciado de lo que es la respuesta al ejercicio.

- Para cada ejercicio que obtenga la información de algún proceso realizado en el ordenador (traza de wireshark, comando...) realice una captura (con <alt>+<impr pant> sólo capturaremos la ventana activa actual). Además de incluirla captura se deben utilizar las herramientas de dibujo del procesador de texto usado para marcar la parte donde se observa lo que pide el ejercicio. Finalmente en el texto añada una pequeña descripción de la captura.
- El formato de entrega de las prácticas será PDF. Además del fichero de la memoria, deberá entregarse el código desarrollado (los `.java`) y las trazas de wireshark (los `.pcapng`).