

Ejercicios Optativos 2

Cristina Díaz García

Diciembre 2018



Índice

Índice general	1
1. I. Desarrollo de servicios no estándar	2
1.1. Ejercicio 2	2
2. Quiz 2	3
3. Quiz 3	3

1. I. Desarrollo de servicios no estándar

1.1. Ejercicio 2

2. Implementar el mismo servicio de “eco” que en el ejercicio anterior pero esta vez sobre TCP. Deberán tenerse en cuenta los siguientes requisitos de implementación:

- Usar la interfaz *Executor* para implementar el servidor concurrente.
- Opcionalmente implementar la otra opción de grupo de hebras y comparar ambas implementaciones
- El cliente le notificará al servidor el final del servicio mediante el cierre del *socket*.

En este ejercicio se han implementado tres clases:

- **TCPCliente:** Es el cliente, al que se le pasan como argumentos al ejecutarlo, primero la dirección IP a la que conectarse, y segundo, el puerto. Si el servidor no estuviera disponible, el cliente se cierra automáticamente. En caso de que se realizara la conexión, envía y recibe los mensajes hasta que se envía un punto, en cuyo caso, el servidor cierra la conexión y el cliente se cierra.

```
java Client
Local connection: /127.0.0.1
Connected to 127.0.0.1:12345
This is an echo server. Type whatever you want me to echo.
Hola
Waiting for a response...
echo: Hola

Waiting for a response...
The connection has been closed.
```

- **TCPServer:** Es el servidor, al que se le pasa como argumento el puerto por el que escuchar. Cada vez que un cliente se intenta conectar, se crea una instancia de EchoProtocol, que es el que “gestiona” la conexión, es decir, es el que hace la función de eco.

```
java Server
Server initialization...ServerSocket[addr=0.0.0.0/0.0.0.0,localport=12345]
Waiting for a new client...
New client, socket Socket[addr=/127.0.0.1,port=48644,localport=12345]
New client, socket Socket[addr=/127.0.0.1,port=48646,localport=12345]
New client, socket Socket[addr=/127.0.0.1,port=48652,localport=12345]
Read timed out
Read timed out
Read timed out
New client, socket Socket[addr=/127.0.0.1,port=49632,localport=12345]
Waiting for a new client...
```

- **EchoProtocol:** Recibe los mensajes del cliente y se los reenvía, siempre que el mensaje no fuera un punto, en cuyo caso cerraría la conexión.


Las ventajas de la implementación con TPC es la fiabilidad de la conexión. En esta implementación, uno de los problemas que tenemos es que estamos asumiendo que los argumentos se van a introducir tanto en el orden correcto como siendo los necesarios, es decir, no vamos a meter una IP incorrecta, ni vamos a introducir una palabra en vez de la IP correspondiente.

1.2. Ejercicio 3


3. A partir del servicio de “eco” dado en clase realizar las siguientes modificaciones para probar las diferentes opciones de sockets sobre TCP: • En el servidor, limitar el tiempo de espera de una nueva conexión dando por finalizado el servidor tras dicho tiempo (ej: varios minutos). Nota: probar a arrancar el servidor justo después de su finalización, antes de implementar la característica que se propone a continuación, ¿qué ocurre? • En el servidor establecer la opción de reutilización del puerto para facilitar el arranque del servidor tras su finalización por el temporizador • En el servidor, limitar el tiempo total del servicio de “eco” (ej: un minuto) • Cambiar en el cliente el tamaño del buffer de escritura a 512 bytes y comprobar si se ha cambiado visualizando el valor anterior y el nuevo. • Deshabilitar el algoritmo de Nagle, que es especialmente recomendado si se va a realizar el eco a nivel de caracteres o mensajes muy pequeños.

2. Quiz 2


La señal más baja es aquella de la que más se pierde la señal, y mientras más mayor el valor absoluto del RSSI negativo, mayor es la pérdida de señal.


 WLAN/signal.png

La gráfica de la evolución temporal es la siguiente:

 WLAN/chart.png

Siempre está en el canal 11 excepto en el paquete 322, que está en el 6, como podemos comprobar:

 WLAN/channel11.png

 WLAN/channel16.png

3. Quiz 3

En este ejercicio el problema es que ciertos paquetes tienen errores en con el checksum o el payload.

WLAN/malformed.png

El tráfico es todo 802.11:

Todo el tráfico coloreado de blanco es de tipo broadcast (802.11) y el negro de tipo error en el checksum.

WLAN/documentacion.png

Referencias

[1] *Coloring rules*, <http://manualwireshark.blogspot.com/>.