Ejercicios Optativos 2

Cristina Díaz García

Diciembre 2018

Índice

1. I. Desarrollo de servicios no estándar

1.1. Ejercicio 1

.

1. Implementar un servicio de "eco" sobre UDP que lea las líneas de teclado hasta leer una línea que contenga solamente un punto. Deberán tenerse en cuenta los siguientes requisitos de implementación:

- El servidor UDP debe ser concurrente. Nota: tener en cuenta que no puedo tener dos procesos UDP asociados al mismo puerto en una misma máquina.
- Modelar el final del servicio con el envío de un "." solo en una línea (<CRLF>.<CRLF>).
- El cliente UDP debe gestionar un temporizador de retransmisiones y establecer un número máximo de retransmisiones.
- Tanto el cliente como el servidor deben controlar que los mensajes del servicio de eco pertenezcan todos a la misma transacción chequeando el puerto.

En este ejercicio se han implementado tres clases:

- **ClientUDP:** 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 puerto no se proporcionara, por defecto se usa el puerto 6789. Se introduce lo que se quiera, y al enviar, si el mensaje es un ".", se cierra la conexión, y en caso contrario, se envía al servidor. Para cada envío, hay un tiempo máximo para recibir la respuesta (3 segundos), y un número máximo de intentos (7 intentos). Si no se recibiera respuesta, se cerraría la conexión.

```
socket.setSoTimeout(TIMEOUT); // Temporizador para cada envio
try {
socket.receive(receivePacket);
receivedResponse = true;
} catch (SocketTimeoutException e) { // Expiro el temporizador
tries += 1;
System.out.println("Timeout, "+(MAXTRIES - tries)+" mas");
}
} while ((!receivedResponse) && (tries < MAXTRIES));
```

- **ServerUDP:** Es el servidor, al que se le pasa como argumento el puerto por el que escuchar. La IP es localhost. Cada vez que un cliente se intenta conectar, se crea una instancia de *ServerUDPImpl*, que es el que "gestiona" la conexión, es decir, es el que hace la función de eco.

```
Servidor iniciado.
Vamos a usar el puerto 54321
Nueva peticion de servicio
Nueva peticion de servicio
Nueva peticion de servicio
Nueva peticion de servicio
```

- **ServerUDPImpl:** Recibe los mensajes del cliente y se los reenvía.

Las ventajas de la implementación con UDP es la rapidez de la conexión frente a la posible implementación en TCP. 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.

2. II. Desarrollo de servicios estándar

2.1. Ejercicio 2

2. Desarrollar un cliente de smtp simplificado que sea capaz únicamente de enviar mensajes de correo electrónico. El protocolo SMTP está definido en el RFC 821 que se adjunta.

- Esta aplicación debe probarse en la máquina virtual de Linux, con el servidor de correo estándar instalado en dicha máquina.
- De los comandos del protocolo propuestos en el RFC sólo tendrán que implementar: HELO, MAIL, RCPT, DATA, QUIT (implementación mínima, pág. 41).
- La interfaz de usuario será muy sencilla:
  - Compose
    - Introducir una dirección de correo destino (opcionalmente podrían ser varias)
    - Introducir el asunto del mensaje (el *Subject*)
    - Introducir el cuerpo del mensaje (podrá ser una o varias líneas, hasta finalizar con un punto)
  - Quit
    - Cerrar la conexión
- Para comprobar que los mensajes enviados con tu aplicación han llegado correctamente utiliza el cliente de correo estándar de Linux Mail. Un ejemplo sería:

```
[alumno@localhost ~]$ Mail
Mail version 8.1 6/6/93.  Type ? for help.
"/var/spool/mail/alumno": 1 message 1 new
>N  1 alumno@localhost.loc  Thu Oct 17 13:57  22/759   "Prueba"
& 1
Message 1:
From alumno@localhost.localdomain  Thu Oct 17 13:57:30 2013
Date: Thu, 17 Oct 2013 13:57:29 +0200
From: alumno@localhost.localdomain
To: alumno@localhost.localdomain
Subject: Prueba

Estimados alumnos:
espero que la práctica no les resulte difícil.

Saludos,
Lidia
&
```

- Nota: No usar la API javax.mail
- Nota: La dirección de correo de este usuario es nombre_usuarionombre_dominio. Para obtener el nombre del usuario actual, usar los siguientes métodos de Java:
  - nombre_usuario: System.getProperty("user.name")
  - nombre_dominio: InetAddress.getLocalHost().getHostName()