PRÁCTICA 5

5.1. Objetivo

El objetivo de la práctica consiste en programar un servidor de eco UDP utilizando los conceptos y las clases de Java explicadas en la práctica anterior. El servidor escuchará en un puerto específico y devolverá una cadena de texto predefinida.

5.2. Conceptos básicos

UDP

En UDP no se ofrece nigún tipo de control o fiabilidad en la transmisión de datos: los paquetes enviados pueden llegar, no llegar o hacerlo de forma desordenada. Ofrece la ventaja de transmisiones rápidas al no tener la necesidad de establecer una conexión previa antes de la transmisión de datos y no tener la necesidad de implementar mecanismos de control de errores:

- Es un protocolo no orientado a conexión
- Envía paquetes de datos (datagramas) independientes, sin garantía de llegada.
- Permite *broadcast* y *multicast*.
- Protocolos de nivel de aplicación que usan UDP: DNS, TFTP, etc.

5.3. Sockets

Una aplicación que funciona como servidor es una aplicación que está ofreciendo un determinado servicio a otros tipos de procesos (clientes), que lo solicitan. Será necesario, por tanto, que el servidor esté registrado en un puerto determinado y que éste sea conocido por los clientes a los que les prestará el servicio correspondiente. Los datos enviados por los clientes al puerto del servidor serán recibidos y tratados por la aplicación servidor.

Los puertos:

- son independientes para TCP y UDP
- se identifican por un número de 16 bits (de 0 a 65535)
- algunos de ellos están reservados (de 0 a 1023), puesto que se emplean para servicios conocidos como HTTP, FTP, etc. y no deberían ser utilizados por aplicaciones de usuario.

Para la implementación de la aplicación servidor será necesario crear un socket especificando el puerto asignado.

5.3.1 Sockets en Java

Java incluye la librería java.net para la utilización de sockets, tanto TCP como UDP. Será necesario importarla para su utilización.

Los sockets UDP son no orientados a conexión. Los clientes no se conectaran con el servidor sinó que cada comunicación será independiente, sin poderse garantizar la recepción de los paquetes ni el orden de los mismos.

Ejercicio

Implementar un servidor de eco UDP. El formato de ejecución será:

```
java ServidorUDP <puerto_servidor>
```

En donde:

- ***ServidorUDP*** es el nombre de vuestro programa.
- ***puerto_servidor*** será el puerto en el que queremos que escuche nuestro servidor de eco UDP.

El programa deberá:

- mostrar por pantalla la IP del equipo en el que se ejecuta el servidor (IP local)
- del datagrama recibido mostrar por pantalla:
    - IP de la aplicación cliente
    - Puerto desde donde envía la aplicación cliente
- Retornar al cliente un mensaje de texto predefinido en el programa.
- Esperar un tiempo máximo (definido por programa) a una comunicación de un cliente. Pasado ese tiempo el programa mostrará por pantalla un mensaje de *timeout* y finalizará la ejecución.

Para ello se deberán utilizar las clases ya conocidas: *DatagramSocket, DatagramPacket, InetAddress*.

El programa deberá controlar las excepciones que puedan generar la obtención de la IP local (*UnknownHostException*), la apertura del socket (*SocketException*), temporización en la espera de un paquete (*SocketTimeoutException*) y cualquier otra excepción más genérica (*Exception*) en cuyo caso se imprimirá el error mediante el método *getMessage()*.