Repaso - Conexiones UDP

Más información en la documentación oficial:
https://docs.oracle.com/javase/tutorial/networking/datagrams/index.html

Las clases DatagramPacket y DatagramSocket se utilizan para crear clientes y servidores basándonos en el protocolo UDP.

- **DatagramSocket** es el mecanismo para enviar y recibir información.
- **DatagramPacket** es el contenedor de la información y enviará los paquetes a través del DatagramSocket creado.

El lado del cliente

En este caso es un cliente que se conecta a un servidor. Envía un mensaje y espera recibir una respuesta del servidor.

Pasos para enviar:

1. Preparo el DatagramSocket por donde enviaré y recibire los DatagramPackets. Si os fijais , de momento, no se especifica ni puerto, ni IP.
2. Preparo el DatagramPacket que necesita 4 parametros:
    - Mensaje a enviar en bytes
    - Longitud del mensaje a enviar
    - IP del servidor (en clase InetAddress)
    - Puerto del servidor
3. Envio el DatagramPacket a través del DatagramSocket.

Pasos para recibir:

1. Ya tengo creado el DatagramSocket por donde espero recibir la respuesta.
2. Preparo un DatagramPacket vacio donde recibire la información.
3. Relleno el DatagramPacket vacio con la información que recibo del DatagramSocket.
4. Muestro por pantalla la respuesta.

```
public class ClienteWeb {
    public static void main(String args[]){
        try {
            // Preparo la conexión del DatagramSocket
            DatagramSocket socketUDP = new DatagramSocket();

            // Preparo el DatagramPacket a enviar
            byte[] mensaje = "texto a enviar".getBytes();
            InetAddress hostServidor = InetAddress.getByName("localhost");
            int puertoServidor = 6789;

            // Construimos un datagrama para enviar el mensaje al servidor
```

```
            DatagramPacket peticion = new DatagramPacket(mensaje, mensaje.length,
hostServidor, puertoServidor);

            // Enviamos el datagrama
            socketUDP.send(peticion);

            // Preparo un DatagramPacket vacio que contendrá la respuesta
            byte[] bufer = new byte[1024];
            DatagramPacket respuesta = new DatagramPacket(bufer, bufer.length);
            socketUDP.receive(respuesta);

            // Muestro por pantalla la respuesta.
            System.out.println("Respuesta: " + new String(respuesta.getData()));

            // Cerramos el socket
            socketUDP.close();

        } catch (IOException e) {
            System.out.println("Socket: " + e.getMessage());
        }
    }
}
```

El lado del servidor

El desarrollo del servidor en este tipo de conexiones es muy similar al cliente. Solo se diferencia el tipo de constructor utilizado para el DatagramSocket. En este caso se especifica el puerto y/o la direccion IP a utilizar:

- DatagramSocket(int port)
- DatagramSocket(int port, InetAddress laddr)

Pasos a realizar:

1. Abrir un DatagramSocket especificando el puerto donde esperaremos las conexiones.
2. Construimos (al igual que en el cliente) un DatagramPacket vacio que rellenaremos con la peticion del cliente.
3. Recibimos la peticion y realizamos las acciones pertinentes para "calcular" la respuesta.
4. Construimos el DatagramPacket con la respuesta.
5. Enviamos al respuesta por el DatagramSocket.

```
public class Server {
   public static void main (String args[]){
       try {
           // Abrimos un DatagramSocket en el puerto 6789
           DatagramSocket socketUDP = new DatagramSocket(6789);
           byte[] bufer = new byte[1000];
           while (true) {
               // Construimos el DatagramPacket para recibir peticiones
               DatagramPacket peticion = new DatagramPacket(bufer, bufer.length);
```

```
                // Leemos una petición del DatagramSocket
                socketUDP.receive(peticion);

                /* Tenemos el mensaje del cliente, y calculamos la repuesta*/
                String mensajeRepuesta = "......";

                // Construimos el DatagramPacket para enviar la respuesta
                DatagramPacket respuesta = new
DatagramPacket(mensajeRepuesta.getBytes(),
mensajeRepuesta.getLength(),peticion.getAddress(), peticion.getPort());

                // Enviamos la respuesta, que es un eco
                socketUDP.send(respuesta);
            }
        } catch (IOException e) {
            System.out.println("IO: " + e.getMessage());
        }
    }
}
```

Otro tipo de conexiones - Multicast

La clase MulticastSocket se utiliza para enviar paquetes a múltiples clientes simultáneamente. Para poder utilizarla es necesario establecer un grupo multicast, que es un grupo de direcciones IP que comparten el mismo puerto.

Cuando se envía un mensaje a un grupo de multicast, todo los equipos que pertenezcan a ese grupo recibirán el mensaje. La pertenencia al grupo es transparente al emisor, es decir, el emisor no conoce el número de miembro del grupo ni sus direcciones IP.

Un grupo multicast se especifica mediante una dirección IP de clase D y un numero de puerto UDP estándar.

- Las direcciones desde la 224.0.0.0 a la 239.255.255.255 están destinadas para ser direcciones multicast.
- Algunas direcciones Multicast están reservadas y no se pueden utilizar en desarrollos reales.

Ejemplo de aplicación: No existen arquitecturas definidas para este modelo de comunicación. Sin embargo, lo más habitual es utilizarlo para contenido multimedia donde un servidor envía información y una serie de clientes la reciben. Este es el ejemplo que muestra a continuación:

Los clientes solo recibiran mensajes del servidor. Pasos:

1. Creo el MulticastSocket.
2. Me uno al grupo de la dirección IP multicast seleccionada.
3. Recibo la información en un DatagramPacket vacio y la muestro por pantalla.
    - Fijaros que todo esta dentro de un bucle while para que este siempre recibiendo información y mostrandola.

```
public class cliente {
    public static void main(String args[]) {
```

```
        try {
            MulticastSocket socketUDP = new MulticastSocket(6789);
            socketUDP.joinGroup(InetAddress.getByName("230.0.0.1"));

            try {
                while (true) {
                    // Construimos un datagrama para recibir el mensaje
                    byte[] mensaje = new byte[1024];
                    DatagramPacket peticion = new DatagramPacket(mensaje,
mensaje.length);
                    // Recibo el datagrama
                    socketUDP.receive(peticion);
                    // Mostramos la respuesta por pantalla
                    String respuesta = new String(peticion.getData());
                    System.out.println(respuesta);
                }
            } catch (Exception e) {
                System.out.println("Excepcion en while");
            }

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

El servidor enviará mensajes a los clientes. Pasos:

1. Creamos el MulticastSocket.
2. Creamos un DatagramaPacket.
3. Enviamos el DatagramPacket a través del MulticasSocket.
    - Fijaros que el servidor solo envía un mensaje y finaliza. Lo he hecho asi para simplificar la lógica del programa y centrarnos solo en como hacemos las conexiones.

```
public class Server {
   public static void main (String args[]){
       try {
           // Creamos la conexion multicast
           MulticastSocket socketUDP = new MulticastSocket(6789);

           // Construimos el DatagramPacket para enviar el mensaje
           String mensaje = "Hola a todos, esto es un mensaje multicast";
           DatagramPacket respuesta = new DatagramPacket(mensaje.getBytes(),
mensaje.length(), InetAddress.getByName("230.0.0.1"), 6789);

           // Enviamos el mensaje
           socketUDP.send(respuesta);
       } catch (IOException e) {
           System.out.println(e.getMessage());
       }
```

```
    }
}
```