3.6. CLASES PARA SOCKETS UDP

Los sockets UDP son más simples y eficientes que los TCP pero no está garantizada la entrega de paquetes. No es necesario establecer una "conexión" entre cliente y servidor, como en el caso de TCP, por ello cada vez que se envíen datagramas el emisor debe indicar explícitamente la dirección IP y el puerto del destino para cada paquete y el receptor debe extraer la dirección IP y el puerto del emisor del paquete.

El paquete del datagrama está formado por los siguientes campos:

1. Cadena de bytes conteniendo el mensaje
2. Longitud del mensaje
3. dirección ip destino
4. N° de puerto destino

El paquete java.net proporciona las clases DatagramPacket y DatagramSocket para implementar sockets UDP.

Clase DatagramPacket.

Esta clase proporciona constructores para crear instancias de los datagramas que se van a recibir y de los datagramas que van a ser enviados:

CONSTRUCTOR	MISIÓN
DatagramPacket(byte[] buf, int length)	**Constructor para datagramas recibidos**. Se especifica la cadena de bytes en la que alojar el mensaje (*buf*) y la longitud (*length*) de la misma.
DatagramPacket(byte[] buf, int offset, int length)	**Constructor para datagramas recibidos**. Se especifica la cadena de bytes en la que alojar el mensaje, la longitud de la misma y el offset (*offset*) dentro de la cadena.
DatagramPacket(byte[] buf, int length, InetAddress addrss, int port)	**Constructor para el envío de datagramas**. Se especifica la cadena de bytes a enviar (*buf*), la longitud (*length*), el número de puerto de destino (*port*) y el el host especificado en la dirección *addrss*.
DatagramPacket(byte[] buf, int offset, int length, InetAddress address, int port)	**Constructor para el envío de datagramas**. Igual que el anterior pero se especifica un offset dentro de la cadena de bytes.

El siguiente ejemplo utiliza el tercer constructor para construir un datagrama de envió. El datagrama se enviará a la direcci6n IP de la maquina local, que lo estará esperando por el puerto 12345. El mensaje está formado por la cadena “Enviando Saludos!!” que es necesario codificar en una secuencia de bytes y almacenar el resultado en una matriz de bytes. Después será necesario calcular la longitud del mensaje a enviar. Con InetAddress.getLocalHost() obtengo la dirección IP del host al que enviare el mensaje, en este caso el host local:

1. mensaje: Enviando Saludos I!
2. Longitud: 19
3. destino: 192.168.21 IP del host local
4. port: 12345

```
int port =          12345;   //puerto por el que escucha
InetAddress destino          = InetAddress.getLocalHost();//IP host local

byte[] mensaje     = new byte[1024]; //matriz de bytes
String Saludo      = "Enviando Saludos        !!";
mensaje = Saludo.getBytes();          //codificarlo a bytes para enviarlo

//construyo el datagrama a enviar
DatagramPacket envio       = new DatagramPacket
(mensaje, mensaje.length, destino, port);
```

Para definir el destino de un host con una IP concreta, por ejemplo 80.120.54.1, escribo lo siguiente:

```
InetAddress destino = InetAddress.getEyName("80.120.54.1");
```

Algunos métodos importantes de la clase DatagramPacket son:

MÉTODOS	MISIÓN
InetAddress getAddress ()	Devuelve la dirección IP del host al cual se le envía el datagrama o del que el datagrama se recibió.
byte[] getData()	Devuelve el mensaje contenido en el datagrama tanto recibido como enviado.
int getLength()	Devuelve la longitud de los datos a enviar o a recibir.
int getPort()	Devuelve el número de puerto de la máquina remota a la que se le va a enviar el datagrama o del que se recibió el datagrama.
setAddress (InetAddress addr)	Establece la dirección IP de la máquina a la que se envía el datagrama.
setData (byte [buf])	Establece el búfer de datos para este paquete.
setLength (int length)	Ajusta la longitud de este paquete.
setPort (int Port)	Establece el número de puerto del host remoto al que este datagrama se envía.

El siguiente ejemplo utiliza el primer constructor de DatagramPacket para construir un datagrama de recepción. El mensaje se aloja en la variable bufer, se obtiene la longitud y el mensaje contenido en el datagrama recibido, el mensaje se convierte a String. A continuación, visualiza el número de puerto de la máquina que envía el mensaje y su dirección IP:

```
byte[] bufer       = new byte[1024];
DatagramPacket recibo = new DatagramPacket(bufer, bufer.length);

int bytesRec       = recibo.getLength();//obtengo longitud del mensaje
String paquete=  new String(recibo.getData());//obtengo mensaje
System.out.println("Puerto origen del mensaje:  "          + recibo.getPort());
System.out.println("IP de origen       : "        +
recibo.getAddress() .getHostAddress () ) ;
```

Clase DatagramSocket

Da soporte a sockets para el envio y reception de datagramas UDP. Algunos de los constructores de esta clase, que pueden lanzar is excepcion SocketException, son:

CONSTRUCTOR	MISIÓN
DatagramSocket ()	Construye un socket para datagramas, el sistema elige un puerto de los que están libres.
DatagramSocket (int port)	Construye un socket para datagramas y lo conecta al puerto local especificado.
DatagramSocket (int port, InetAddress ip)	Permite especificar, además del puerto, la dirección local a la que se va a asociar el socket.

El siguiente ejemplo construye un socket para datagrama y no lo conecta a ningim puerto, el sistema elige el puerto:

DatagramSocket socket = new DatagramSocket();

Para enlazar el socket a un puerto especifico, por ejemplo al puerto 12345, escribimos:

DatagramSocket socket = new DatagramSocket (12345) ;

Algunos métodos importantes son:

MÉTODOS	MISIÓN
receive (DatagramPacket paquete)	Recibe un **DatagramPacket** del socket, y llena *paquete* con los datos que recibe (mensaje, longitud y origen). Puede lanzar la excepción ***IOException***.
send (DatagramPacket paquete)	Envía un **DatagramPacket** a través del socket. El argumento *paquete* contiene el mensaje y su destino. Puede lanzar la excepción ***IOException***.
close ()	Se encarga de cerrar el socket.
int getLocalPort ()	Devuelve el número de puerto en el host local al que está enlazado el socket, -1 si el socket está cerrado y 0 si no está enlazado a ningún puerto.
int getPort()	Devuelve el número de puerto al que está conectado el socket, -1 si no está conectado.
connect(InetAddress address, int port)	Conecta el socket a un puerto remoto y una dirección IP concretos, el socket solo podrá enviar y recibir mensajes desde esa dirección.
setSoTimeout(int timeout)	Permite establecer un tiempo de espera límite. Entonces el método ***receive()*** se bloquea durante el tiempo fijado. Si no se reciben datos en el tiempo fijado se lanza la excepción ***InterruptedIOException.***

Siguiendo con el ejemplo inicial, una vez construido el datagrama lo enviamos usando un DatagramSocket, en el ejemplo se enlaza al puerto 34567. Mediante el método send() se envia el datagrama:

```
//construyo datagrama a enviar indicando el host destino y puerto
DatagramPacket envio = new DatagramPacket
(mensaje, mensaje.length, destino, port);
DatagramSocket socket = new DatagramSocket(34567);
socket.send(envio);//envio datagrama a destino y port
```

En el otro extremo, para recibir el datagrama usamos también un DatagramSocket. En primer lugar, habrá que enlazar el socket al puerto por el que se va a recibir el mensaje, en este caso el 12345. Después se construye el datagrama para recepción y mediante el método receive() obtenemos los datos. Luego obtenemos la longitud, la cadena y visualizamos los puertos origen y destino del mensaje:

```
DatagramSocket socket = new DatagramSocket(12345);
//construyo datagrama a recibir
DatagramPacket recibo = new DatagramPacket(bufer, bufer.length);
socket.receive(recibo);//recibo datagrama

int bytesRec = recibo.getLength();//obtengo numero de bytes
String paquete= new String(recibo.getData());//obtengo String

System.out.println("Numero de Bytes recibidos: "+ bytesRec);
System.out.println("Contenido del Paquete: " + paquete.trim());
System.out.println("Puerto origen del mensaje: " + recibo.getPort());
System.out.printin("IP de origen:" + recibo.getAddress().getHostAddress());
System.out.println("Puerto destino del mensaje:" + socket.getLocalPort());
socket.close(); //cierro el socket
```

La salida muestra la siguiente información:

```
Numero de Bytes recibidos: 19
Contenido del Paquete: Enviando Saludos !!
Puerto origen del mensaje: 34567
IP de origen: 169.254.30.179
Puerto destino del mensaje: 12345
```

3.6.1. GESTION DE SOCKETS UDP

En los sockets UDP no se establece conexión. Los roles cliente-servidor están un poco más difusos que en el caso de TCP. Podemos considerar al servidor como el que espera un mensaje y responde; y al cliente como el que inicia la comunicación. Tanto uno como otro si desean ponerse en contacto necesitan saber en que ordenador y en que puerto esta escuchando el otro.

La figura 3.7 muestra el flujo de la comunicaciOn entre cliente y servidor usando UDP, ambos necesitan crear un socket DatagramSocket:

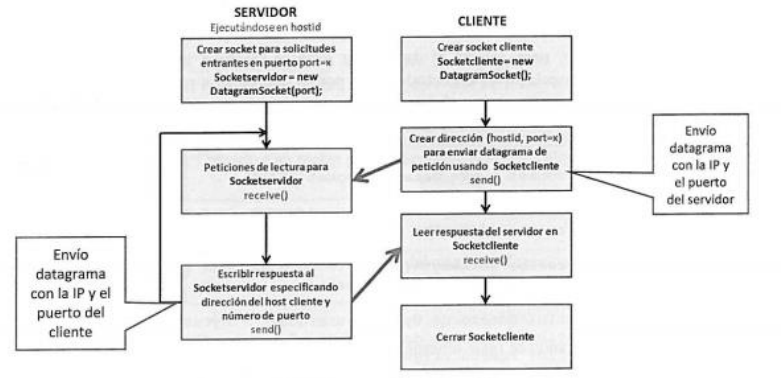

Figura 3.7. Envío y recepción de un datagrama.

■ El servidor crea un socket asociado a un puerto local para escuchar peticiones de clientes. Permanece a la espera de recibir peticiones.

■ El cliente creara un socket para comunicarse con el servidor. Para enviar datagramas necesita conocer su IP y el puerto por el que escucha. Utilizard el metodo send() del socket para enviar la petici6n en forma de datagrama.

■ El servidor recibe las peticiones mediante el metodo receive° del socket. En el

datagrama va incluido ademas del mensaje, el puerto y la IP del cliente emisor de la petition; lo que le permite al servidor conocer la direccion del emisor del datagrama. Utilizando el metodo send() del socket puede enviar la respuesta al cliente emisor.

- El cliente recibe la respuesta del servidor mediante el metodo receive° del socket.
- El servidor permanece a la espera de recibir mas peticiones.

Apertura y cierre de sockets.

Para construir un socket datagrama es necesario instanciar la clase DatagramSocket tanto en el programa cliente como en el servidor, vimos anteriormente algunos ejemplos de corm se usa. Para escuchar peticiones en un puerto UDP concreto pasamos al constructor el ntimero de puerto. El siguiente ejemplo crea un socket datagrama, le pasamos al constructor el numero de puerto 12345 por el que escucha las peticiones y la direcci6n InetAddress en la que se esta ejecutando el programa, que normalmente es InetAddress.getLocalHost():

```
DatagramSocket socket = new DatagramSocket
(12345, InetAddress.getByName("localhost"));
```

Para cerrar el socket usamos el metodo closeo: socketclose0.

Envío y recepción de datagramas.

Para crear los datagramas de envío y de recepción usamos la clase DatagramPacket.

Para enviar usamos el metodo send() de DatagramSocket pasando como parámetro el DatagramPacket que acabamos de crear:

```
DatagramPacket datagrama = new DatagramPacket(
mensajeEnBytes,// el array de bytes
mensajeEnBytes.length,// su longitud
InetAddress.getByName("localhost"), // maquina destino
PuertoDelServidor); // puerto del destinatario
socket.send(datagrama);
```

Para recibir usamos el método receive() de DatagramSocket pasando como parámetro el DatagramPacket que acabamos de crear. Este método se bloquea hasta que se recibe un datagrama, a menos que se establezca un tiempo límite (timeout) sobre el socket.

```
DatagramPacket datagrama new DatagramPacket(new byte[1024], 1024);
socket.receive(datagrama);
```

El siguiente ejemplo crea un programa servidor que recibe un datagrama enviado por un programa cliente. El programa servidor permanece a la espera hasta que le llega un paquete del cliente; en este momento visualiza: el número de bytes recibidos, el contenido del paquete, el puerto y la IP del programa cliente y el puerto local por el que recibe las peticiones:

```
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;

public class UDPservidor {
public static void main(String[] argv) throws Exception {
byte[] bufer = new byte[1024];//bufer para recibir el datagrama
//ASOCIO EL SOCKET AL PUERTO 12345
DatagramSocket socket = new DatagramSocket(12345);
```

```
//ESPERANDO DATAGRAMA
System.out.println("Esperando Datagrama…") ;
DatagramPacket recibo = new DatagramPacket(bufer, bufer.length);
socket.receive(recibo);//recibo datagrama
int bytesRec = recibo.getLength();//obtengo número de bytes
String paquete= new String(recibo.getData());//obtengo String

//VISUALIZO INFORMACIoN
System.out.println("Nnmero de Bytes recibidos: "+ bytesRec);
System.out.println("Contenido del Paquete : "+ paquete.trim()); System.out.println("Puerto
origen del mensaje: "+recibo.getPort());
System.out.println("IP de origen : "+
recibo.getAddress().getHostAddress());
System.out.println("Puerto destino del mensaje:" +
socket.getLocalPort());
socket.close(); //cierro el socket
}
}
```

El programa clíente envía un mensaje al servidor (maquina destino, en este caso es la maquina local, localhost) al puerto 12345 por el que espera peticiones. Visualiza el nombre del host de destino y la direccion IP. Tambien visualiza el puerto local del socket y el puerto al que envía el mensaje:

```
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;

public class UDPcliente {
public static void main(String[] argv) throws Exception {
InetAddress destino = InetAddress.getLocalHost();
int port = 12345; //puerto al que envio el datagrama
byte[] mensaje = new byte[1024];
String Saludo="Enviando Saludos !!";
Mensaje = Saludo.getBytes(); //codifico String a bytes

//CONSTRUYO EL DATAGRAMA A ENVIAR
DatagramPacket envio = new DatagramPacket
(mensaje, mensaje.length, destino, port);
DatagramSocket socket = new DatagramSocket(34567 ):://Puerto local

System.out.println("Enviando Datagrama de longitud: "+
mensaje.length);
System.out.println("Host destino : "+ destino.getHostName());
System.out.println("IP Destino : " + destino.getHostAddress());
System.out.println("Puerto local del socket: " +
socket.getLocalPort());
System.out.println("Puerto al + envio.getPort());que envio: "

//ENVIO DATAGRAMA
socket.send(envio);
socket.close(); //cierro el socket
}
}
```

ACTIVIDAD 3.6

Crea un programa cliente usando sockets UDP que envíe el texto escrito desde la entrada estándar al servidor. El servidor le devolverá la cadena en mayúsculas. El proceso de entrada de datos finalizara cuando el cliente introduzca un asterisco. Crea un programa servidor que reciba cadenas de caracteres, las muestre en pantalla y se las envíe al emisor en mayúscula. El proceso servidor finalizara cuando reciba un asterisco.

Establece un tiempo de espera de 5000 milisegundos con el método setSoTimeout(4000) para hacer que el método receive() del programa cliente se bloquee. Pasado ese tiempo controlar si no se reciben datos lanzando la excepción InterruptedIOException, en cuyo caso visualiza un mensaje indicando que el paquete se ha perdido. Para probarlo ejecuta el programa cliente sin ejecutar el servidor. Puedes ejecutar varios programas cliente a la vez.