Protocolo UPD - Datagram Sockets

Programación de servicios y procesos

Contenidos:

1) Clase InetAdress
2) Datagram Sockets

La clase InetAddress

MÉTODOS	MISIÓN
InetAddress getLocalHost()	Devuelve un objeto *InetAddress* que representa la dirección IP de la máquina donde se está ejecutando el programa.
InetAddress getByName(String host)	Devuelve un objeto *InetAddress* que representa la dirección IP de la máquina que se especifica como parámetro (*host*). Este parámetro puede ser el nombre de la máquina, un nombre de dominio o una dirección IP.
InetAddress[] getAllByName(String host)	Devuelve un array de objetos de tipo *InetAddress*. Este método es útil para averiguar todas las direcciones IP que tenga asignada una máquina en particular.
String getHostAddress()	Devuelve la dirección IP de un objeto *InetAddress* en forma de cadena.
String getHostName()	Devuelve el nombre del host de un objeto *InetAddress*.
String getCanonicalHostName()	Obtiene el nombre canónico completo (suele ser la dirección real del host) de un objeto *InetAddress*.

La clase InetAddress

- Los 3 primeros métodos pueden lanzar la excepción UnknownHostException.

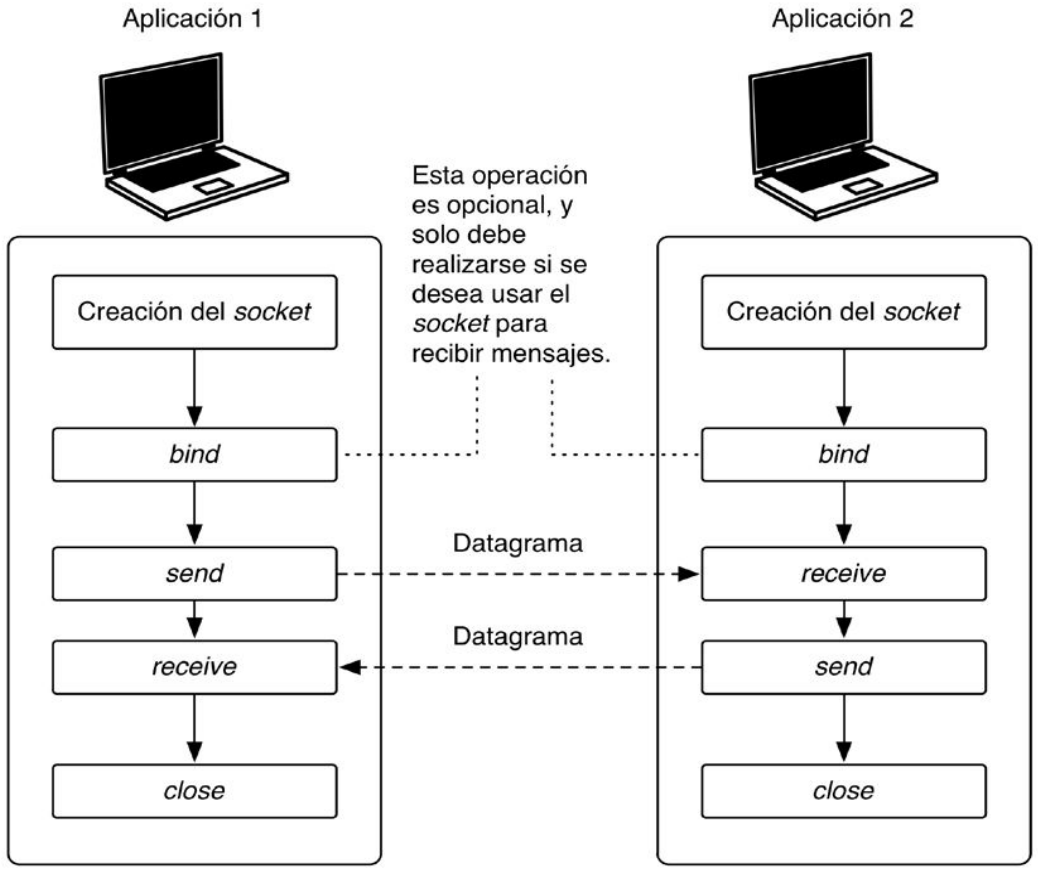
- La forma más típica de crear instancias de InetAddress, es invocando al método estático **getByName(String)** pasándole el nombre DNS del host como parámetro. Este objeto representará la dirección IP de ese host, y se podrá utilizar para construir sockets.

En el ejemplo **UT4_Ejem0_TestInetAdress** se define un objeto InetAddress de nombre dir. En primer lugar lo utilizamos para obtener la dirección IP de la máquina local en la que se ejecuta el programa, en el ejemplo su nombre es localhost. A continuación llamamos al método pruebaMetodos() llevando el objeto creado. En dicho método se prueban los métodos de la clase InetAddress.

Después utilizamos el objeto para obtener la dirección IP de la URL www.google.es y volvemos a invocar a pruebaMetodos() (para que funcione en este segundo caso necesitamos estar conectados a Internet). Por último utilizamos el método getAllByName() para ver todas las direcciones IP asignadas a la máquina representada por www.google.es.

Sockets datagram

- Son ***no orientados a conexión***.
- Cuando operan *sobre IP, emplean UDP.*
- Cuando se usan sockets datagram son más difusos los roles del proceso servidor y proceso cliente.
- Pasos para enviar mensajes:
  - ✔ Creación del socket.
  - ✔ Asignación de dirección y puerto (bind). Solo necesaria para poder recibir mensajes.
  - ✔ Envío y/o recepción de mensajes.
  - ✔ Cierre del socket.



Datagram Sockets

En este tipo de comunicaciones se utiliza el protocolo UDP. Esta conexión no es fiable y no garantiza que la información enviada alcance su destino. Además los paquetes enviados pueden llegar en orden diferente al del envío.

Este tipo de sockets se utiliza cuando la entrega rápida es más importante que una entrega garantizada o en casos donde la información cabe en un datagrama. Se utiliza para la transmisión de video y audio en tiempo real donde es posible el reenvío de paquetes. También se usa en NFS (Network File System), DNS (Domain Name Server) o SNMP (Simple Network Management Protocol).

La implementación se hace a través de las clases DatagramSocket y DatagramPacket.

La clase DatagramPacket

Constructors
DatagramPacket(byte[] buf, int length) Constructs a DatagramPacket for **receiving** packets of length length.
DatagramPacket(byte[] buf, int length, **InetAddress** address, int port). Constructs a datagram packet for **sending** packets of length length to the specified port number on the specified host
DatagramPacket(byte[] buf, int offset, int length) Constructs a DatagramPacket for **receiving** packets of length length, specifying an offset into the buffer.
DatagramPacket(byte[] buf, int offset, int length, **InetAddress** address, int port) Constructs a datagram packet for **sending** packets of length length with offset ioffsetto the specified port number on the specified host

La clase DatagramPacket

Methods	
InetAddress	getAddress() Returns the IP address of the machine to which this datagram is being sent or from which the datagram was received
byte[]	getData() Returns the data buffer.
int	getLength() Returns the length of the data to be sent or the length of the data received.
int	getPort() Returns the port number on the remote host to which this datagram is being sent or from which the datagram was received
void	setAddress(InetAddress iaddr) Sets the IP address of the machine to which this datagram is being sent.

La clase DatagramPacket

Methods	
void	**setData**(byte[] buf) Set the data buffer for this packet.
void	**setLength**(int length) Set the length for this packet.
void	**setPort**(int iport) Sets the port number on the remote host to which this datagram is being sent.

La clase DatagramSocket

Constructor Summary	
	DatagramSocket() Constructs a datagram socket and binds it to any available port on the local host machine.
	DatagramSocket(int port) Constructs a datagram socket and binds it to the specified port on the local host machine.
	DatagramSocket(int port, InetAddress laddr) Creates a datagram socket, bound to the specified local address.

La clase DatagramSocket

Method Summary	
void	**close**() Closes this datagram socket.
void	**connect**(InetAddress address, int port) Connects the socket to a remote address for this socket.
int	**getLocalPort**() Returns the port number on the local host to which this socket is bound. (-1 si está cerrado ó 0 si no está enlazado a ningún puerto)
int	**getPort**() Returns the port for this socket.
void	**receive**(DatagramPacket p) Receives a datagram packet from this socket.
void	**send**(DatagramPacket p) Sends a datagram packet from this socket.
void	**setSoTimeout**(int timeout) Enable/disable SO_TIMEOUT with the specified timeout, in milliseconds.

Datagram Socket

Los sockets UDP son más simples 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:

Cadena de bytes conteniendo el mensaje	Longitud del mensaje	Dirección IP destino	Número puerto destino

Datagram Socket

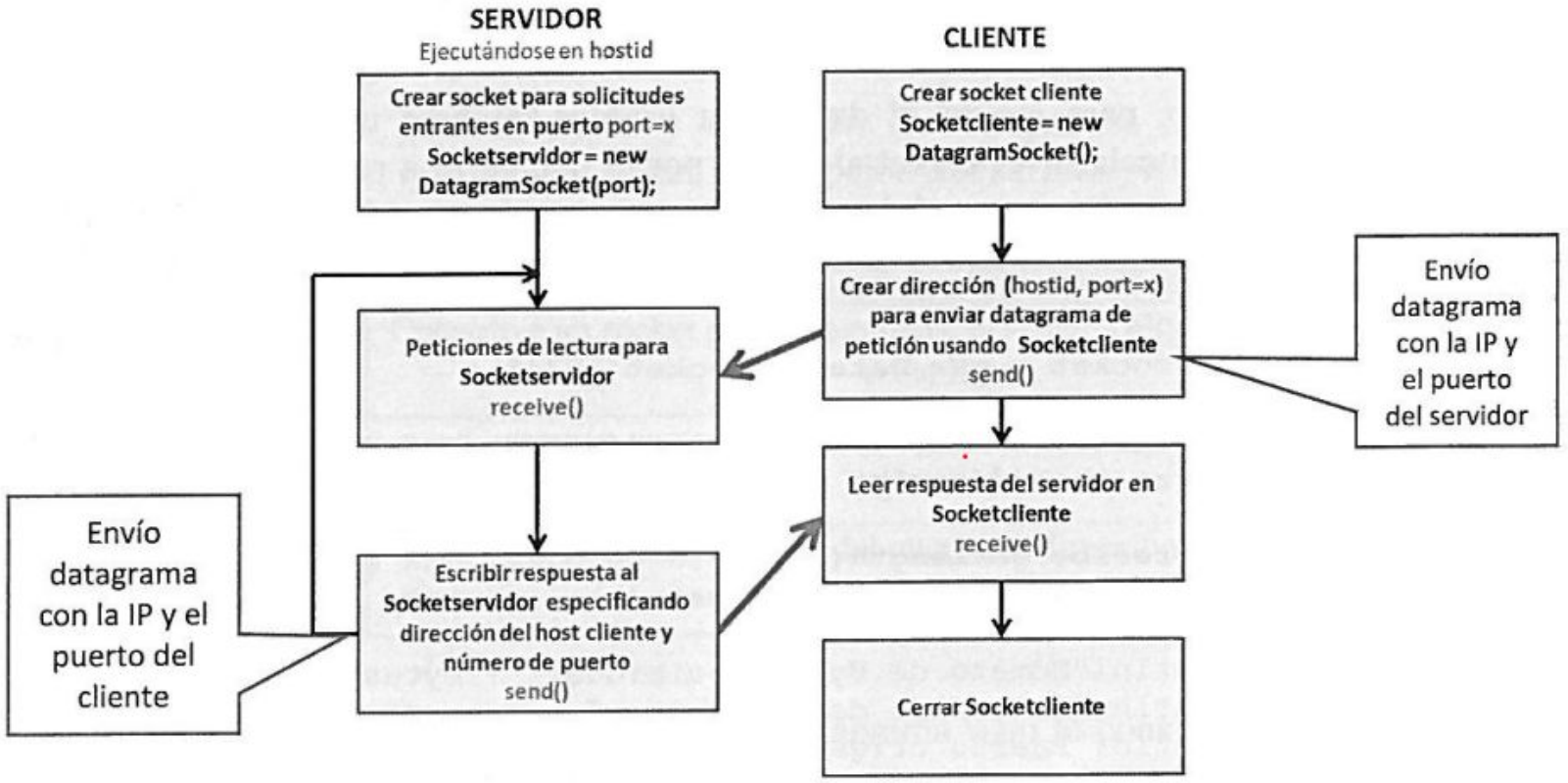


Figura 3.7. Envío y recepción de un datagrama.

Ejemplo comunicación basado en sockets UDP - Servidor

```
 8 public class UDPservidor {
 9   public static void main(String[] argv) throws Exception {
10     byte[] bufer = new byte[1024];//bufer para recibir el datagrama
11     //ASOCIO EL SOCKET AL PUERTO 12345
12     DatagramSocket socket = new DatagramSocket(12345);
13
14     //ESPERANDO DATAGRAMA
15     System.out.println("Esperando Datagrama .......... ");
16     DatagramPacket recibo = new DatagramPacket(bufer, bufer.length);
17     socket.receive(recibo);//recibo datagrama
18     int bytesRec = recibo.getLength();//obtengo numero de bytes
19     String paquete= new String(recibo.getData());//obtengo String
20
```

Variable para recibir y asocio puerto

Recibo el mensaje que me envíe el cliente

Lo paso a string

Ejemplo comunicación basado en sockets UDP - Servidor

```
    //VISUALIZO INFORMACI□N
    System.out.println("N□mero 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
  }
}
```

Recupero información de quien me ha enviado la info

Cierro el socket

Ejemplo comunicación basado en sockets UDP - Cliente

```
7  public class UDPcliente {
8    public static void main(String[] argv) throws Exception {
9      InetAddress destino = InetAddress.getLocalHost();
10     int port = 12345; //puerto al que envio el datagrama
11     byte[] mensaje = new byte[1024];
12
13     String Saludo="Enviando Saludos !!";
14     mensaje = Saludo.getBytes();  //codifico String a bytes
15
16     //CONSTRUYO EL DATAGRAMA A ENVIAR
17     DatagramPacket envio = new DatagramPacket
18                    (mensaje, mensaje.length, destino, port);
19     DatagramSocket socket = new DatagramSocket(34567);//Puerto local
20
```

Recupero información de a qué dirección envío la información. En mi caso es mi propia máquina.

Genero el datagrama a enviar

Ejemplo comunicación basado en sockets UDP - Cliente

```
21      System.out.println("Enviando Datagrama de longitud: "+
22                                              mensaje.length);
23      System.out.println("Host destino : "+  destino.getHostName());
24      System.out.println("IP Destino   : " + destino.getHostAddress());
25      System.out.println("Puerto local del socket: " +
26                                              socket.getLocalPort());
27      System.out.println("Puerto al que envio: " + envio.getPort());
28
29      //ENVIO DATAGRAMA
30      socket.send(envio);
31      socket.close(); //cierro el socket
32    }
33  }
```

Pinto datos de adonde envío y lo que envío

Envío y cierro

Estos proyectos están en los recursos del tema, proyectos: UT4_Ejem2UDPServer y UT4_Ejem2UDPClient.