4. Sockets UDP

- Con UDP **no** se establece “conexión” entre cliente y servidor
  - El emisor indica explicitamente la dirección IP y el puerto del origen y del destino **en cada datagrama**
  - El receptor ha de extraer del datagrama recibido la dirección IP y el puerto del emisor
- Los datos transmitidos pueden llegar fuera de orden o incluso perderse

Direcciones IP en Java

Clase `InetAddress`

- **Algunos métodos importantes**
  - `static InetAddress getByName(String nombre)`
    - Obtiene la dirección IP asociada a un nombre
    - `InetAddres dirIP = InetAddress.getbyName("www.upv.es")`
  - `static InetAddress[] getAllByName(String nombre)`
    - Obtiene las direcciones IP asociadas a un nombre
    - La información obtenida puede imprimirse cómodamente mediante el método `Arrays.toString(Object[] a)` (incluir java.util.Arrays)

Datagrama UDP

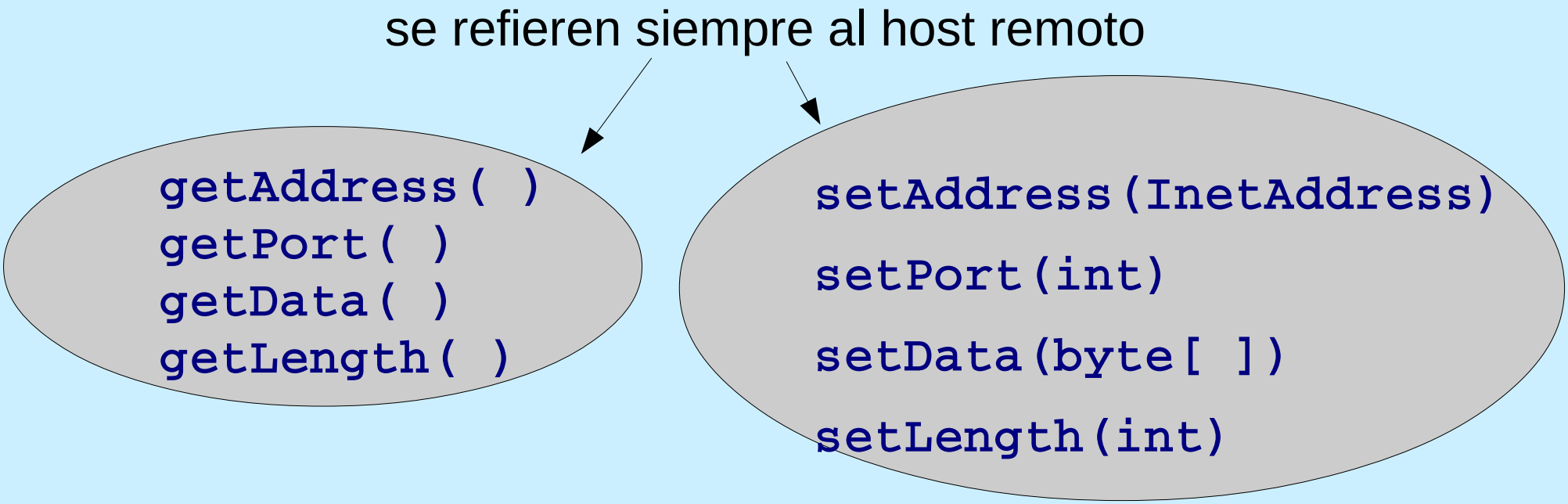
Clase `DatagramPacket`

- **Constructores**
  - `DatagramPacket(byte buf[ ], int longitud)`
    - Crea un datagrama UDP de esa longitud para recibir
  - `DatagramPacket(byte buf[ ], int longitud, InetAddress dirIP, int puerto)`
    - Crea un datagrama UDP con ese buffer y de esa longitud para enviarlo a la dirección IP y puerto que se indican

Datagrama UDP (II)

Clase **`DatagramPacket`**

- **Algunos métodos importantes**



Socket UDP

Clase **`DatagramSocket`**

- **Constructores**
  - **`DatagramSocket() throws SocketException`**
    - Crea un socket UDP que escucha en un puerto libre
  - **`DatagramSocket(int puerto) throws SocketException`**
    - Crea un socket UDP que escucha en ese puerto
- Método **`getLocalPort()`**
  - `int p = ds.getLocalPort();`

Socket UDP (II)

Clase `DatagramSocket`

- **Algunos métodos importantes**
  - `send(DatagramPacket p) throws IOException`
    - Envía un datagrama
    - El `DatagramPacket` incluye los datos a enviar, su longitud y la dirección IP y el puerto del destino
  - `receive(DatagramPacket p) throws IOException`
    - Recibe datagramas. El método es bloqueante
    - Cuando el método retorna, el buffer del `DatagramPacket` contiene los datos recibidos , la dirección IP y el puerto de quien envía el datagrama
  - `close()`

Conversiones

- El contenido de un `DatagramPacket` son bytes. Por tanto, para enviar strings deben convertirse a bytes::

```
String s = “ejemplo”;
byte[ ] buffer = new byte[2048];
buffer = s.getBytes();
```

- Al recibir un `DatagramPacket p` para extraer el string de los bytes recibidos puede hacerse mediante:

```
String s = new String(p.getData(), 0, p.getLength());
```

Cliente UDP

```
import java.net.*;
import java.io.*;

public class ClienteUDP{
  public static void main(String[] args) throws IOException {
    DatagramSocket s = new DatagramSocket();
    InetAddress dir = InetAddress.getByName("zoltar.redes.upv.es");
    String msg = "Hola, esto es un mensaje \r\n";
    byte[] buf = new byte[256];
    buf = msg.getBytes();
    DatagramPacket p = new DatagramPacket(buf, buf.length, dir, 7777);
    s.send(p);
    s.receive(p); // se bloquea hasta que recibe un datagrama
     String ans = new String(p.getData(), 0, p.getLength()());
    System.out.println(ans);
    s.close();
  }
}
```

- El cliente envía un datagrama a un servidor y muestra la respuesta por pantalla

Servidor UDP

```
import java.net.*;
import java.io.*;

public class ServidorUDP{
  public static void main(String[] args) throws IOException {
    DatagramSocket s = new DatagramSocket(7777);
    DatagramPacket p = new DatagramPacket(new byte[512], 512);
    while(true){
       s.receive(p); // se bloquea hasta que recibe un datagrama
       s.send(p);
    }
  }
}
```

- Envía de vuelta el datagrama recibido, sin modificarlo, a la dirección IP y puerto de origen