

Sockets

Transmisión sin conexión mediante datagramas



Transmisión sin conexión mediante datagramas

- ▶ La transmisión sin conexión mediante datagramas es un proceso parecido a la manera en que el correo tradicional se transporta mediante el servicio postal

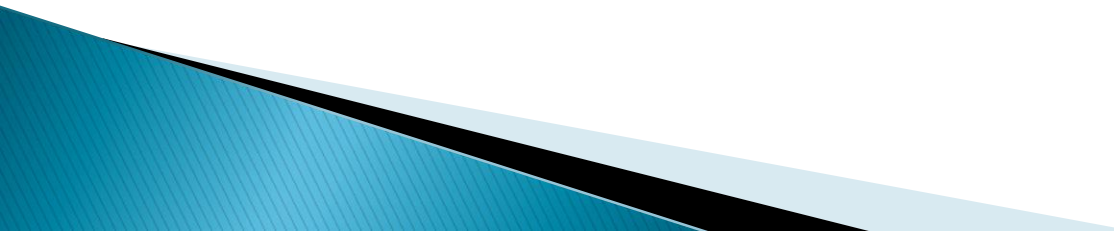
Transmisión sin conexión mediante datagramas

- ▶ Si un mensaje grande no cabe en un sobre, se divide en varias piezas separadas que se colocan en sobres separados, numerados en forma secuencial
- ▶ Cada una de las cartas se envía por correo al mismo tiempo
 - Las cartas podrían llegar en orden, sin orden o tal vez no llegarían (aunque es poco probable)
- ▶ El receptor debe re-ensamblar las piezas del mensaje en orden secuencial, antes de interpretarlo

Ejemplo

- ▶ En el ejemplo siguiente se utilizan datagramas para enviar paquetes de información mediante el protocolo de datagramas de usuario (UDP) entre una aplicación cliente y una aplicación servidor

Cliente

- ▶ En la aplicación cliente, el usuario escribe un mensaje, entonces este mensaje se convierte en un arreglo byte y se coloca en un paquete de datagramas que se envía al servidor
 - ▶ Cuando recibe un paquete del servidor, muestra la información que contiene
- 

Servidor

- ▶ En la aplicación servidor recibe el paquete y muestra la información que contiene, después lo repite de vuelta al cliente

Aplicación servidor

```
public class Servidor extends JFrame {  
    private DatagramSocket socket;  
    ...  
    public Servidor() {  
        // crea DatagramSocket para enviar y recibir paquetes  
        try {  
            socket = new DatagramSocket( 5000 );  
        } catch( SocketException socketException ) {  
            socketException.printStackTrace();  
            System.exit( 1 );  
        }  
        ... }  
}
```

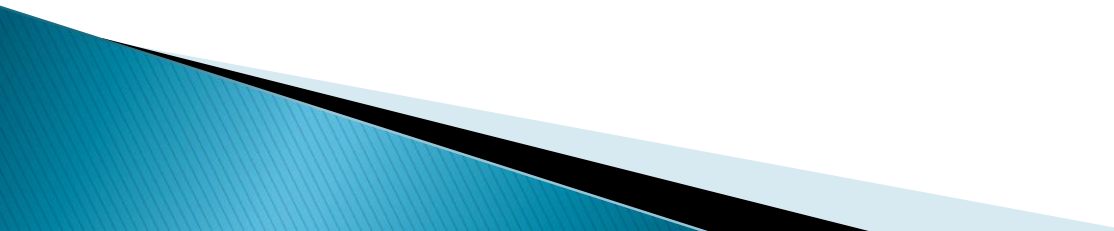

Aplicación servidor

```
private void esperaPorPaquetes(){
    while ( true ) { // loop eterno
try {// recibe paquete, muestra contenido y regresa copia al cliente
    // inicializa paquete
    byte data[] = new byte[ 100 ];
    DatagramPacket recibePaquete =
        new DatagramPacket( data, data.length );
    socket.receive( recibePaquete ); // espera por el paquete
    // muestra informacion del paquete recibido
    muestraMensaje( "\nPaquete recibido:" +
        "\nDel host: " + recibePaquete.getAddress() +
        "\nPuerto del Host: " + recibePaquete.getPort() +
        "\nLongitud: " + recibePaquete.getLength() +
        "\nConteniendo:\n\t" + new String( recibePaquete.getData(),
            0, recibePaquete.getLength() ) );

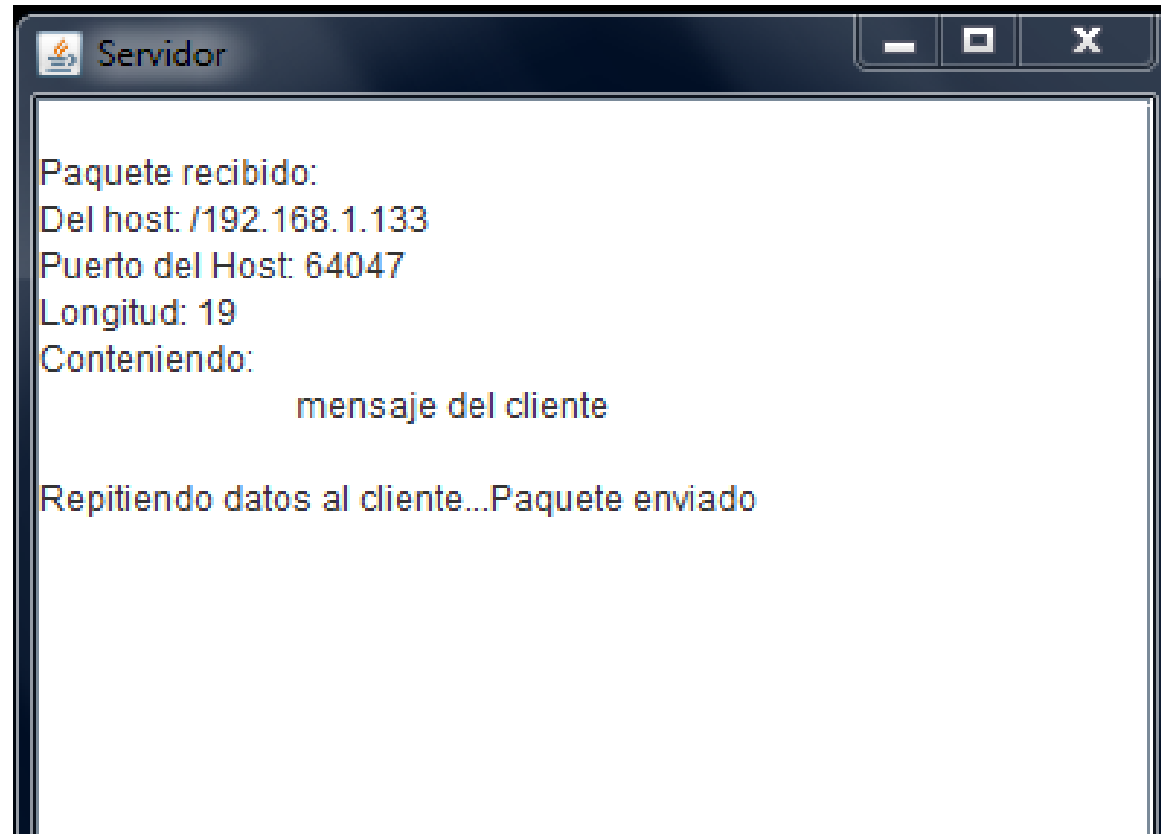
        enviaPaqueteAlCliente( recibePaquete );}
... } // fin while
    } // fin del metodo esperaPorPaquetes
```

Aplicación servidor

```
private void enviaPaqueteAlCliente( DatagramPacket recibePaquete )  
    throws IOException  
{  
    muestraMensaje( "\n\nRepitiendo datos al cliente..." );  
  
    // crea paquete a enviar  
    DatagramPacket sendPacket = new DatagramPacket(  
        recibePaquete.getData(), recibePaquete.getLength(),  
        recibePaquete.getAddress(), recibePaquete.getPort() );  
  
    socket.send( sendPacket ); // envia paquete  
    muestraMensaje( "Paquete enviado\n" );  
}
```



Aplicación servidor (ejecución)



Aplicación cliente

```
public class Cliente extends JFrame {  
    ...  
    private DatagramSocket socket;  
    ...  
}
```

Aplicación cliente

```
entradaCampo.addActionListener(  
    new ActionListener() {  
        public void actionPerformed((ActionEvent event) )  
        {  
            // crea y envia el paquete  
            try {  
                pantallaArea.append( "\nEnviando paquete que contiene: " +  
                    event.getActionCommand() + "\n" );  
                // obtiene el mensaje del campo de texto y  
                // lo convierte a un arreglo de bytes  
                String mensaje = event.getActionCommand();  
                byte datos[] = mensaje.getBytes();  
                // crea enviarPaquete  
                DatagramPacket enviaPaquete = new DatagramPacket( datos,  
                    datos.length, InetAddress.getLocalHost(), 5000 );  
                socket.send( enviaPaquete ); // envia paquete  
                pantallaArea.append( "Paquete enviado\n" );  
                pantallaArea.setCaretPosition(  
                    pantallaArea.getText().length() );  
            } ...  
        }  
    }  
);
```

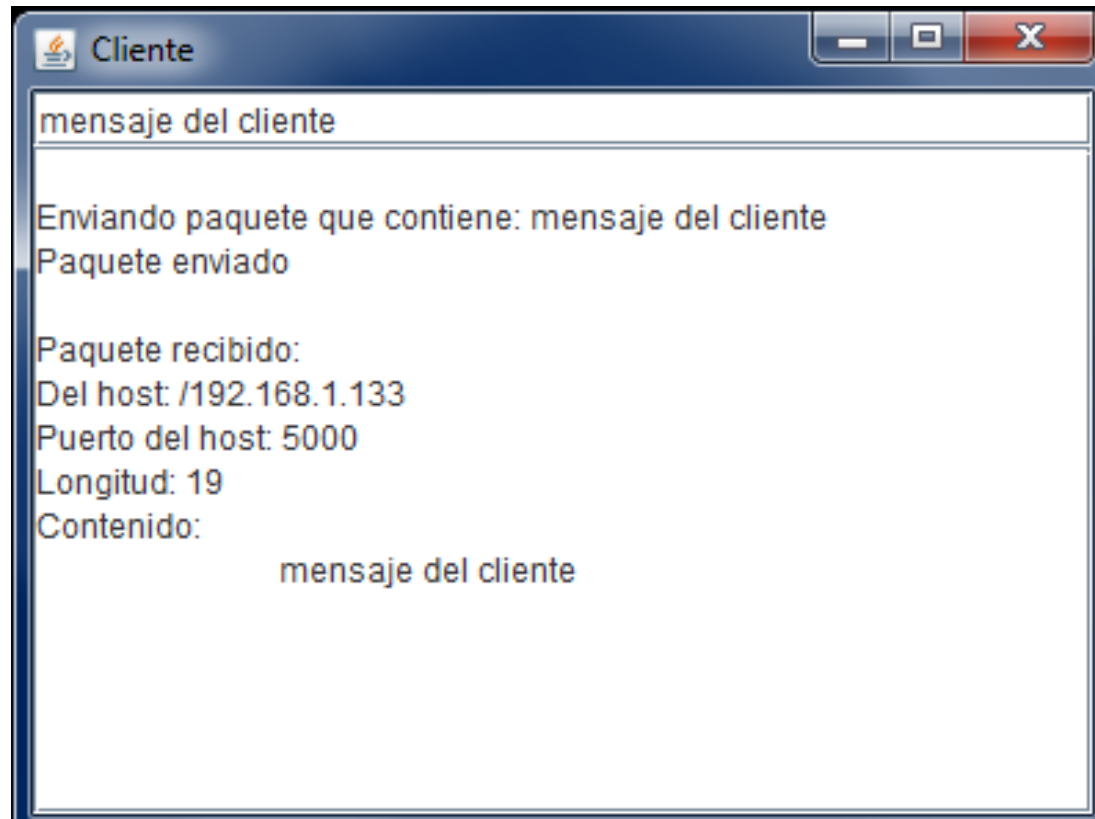
Aplicación cliente

```
// creacion sw DatagramSocket para enviar y recibirpaquetes
try {
    socket = new DatagramSocket();
}
```

Aplicación cliente

```
private void esperaPorPaquetes(){  
    while ( true ) { // loop eterno  
        // recibe el paquete y muestra el contenido  
        try {  
            // inicializa el paquete  
            byte datos[] = new byte[ 100 ];  
            DatagramPacket recibePaquete = new DatagramPacket(  
                datos, datos.length );  
            socket.receive( recibePaquete ); // espera por el paquete  
            // muestra el contenido del paquete  
            muestraMensaje( "\nPaquete recibido:" +  
                "\nDel host: " + recibePaquete.getAddress() +  
                "\nPuerto del host: " + recibePaquete.getPort() +  
                "\nLongitud: " + recibePaquete.getLength() +  
                "\nContenido:\n\t" + new String( recibePaquete.getData(),  
                    0, recibePaquete.getLength() ) );  
        } ...  
    }  
}
```

Aplicación Cliente



Fin

