Curso

2011 | 2012

Práctica 4

Redes de Computadores

David Morales Sáez
Alberto Manuel Mireles Suárez

Servidor TCP

Se ha creado un servidor TCP que acepte y muestre los mensajes GET enviados desde un navegador externo por pantalla. Para su uso, se ha de ejecutar desde el servidor el programa desde consola:

```
java TCPServer puerto
```

Por otro lado, el cliente ha de configurar su navegador web para que redirija las peticiones a la dirección ip del servidor y al puerto escogido.

El código del servidor es el siguiente:

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

public class TCPServer
{
    public static void main(String args[]) throws Exception
    {
     String clientSentence;
     String capitalizedSentence;
     int port = Integer.parseInt(args[0]);
     ServerSocket welcomeSocket = new ServerSocket(port);

     while(true)
     {
          Socket connectionSocket = welcomeSocket.accept();
          BufferedReader inFromClient = new BufferedReader(new
InputStreamReader(connectionSocket.getInputStream()));
          DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());
          clientSentence = inFromClient.readLine();
          while(clientSentence!=null)
          {
               capitalizedSentence =
clientSentence.toUpperCase() + '\n';
               outToClient.writeBytes(capitalizedSentence);
               System.out.println(clientSentence);
               clientSentence = inFromClient.readLine();
          }
       }
    }
}
```

Servidor y cliente UDP

Hemos creado un servidor UDP que acepte solicitudes de Ping a través del puerto deseado. Esta solicitud será respondida al cliente solicitante. El servidor está siempre esperando solicitudes y, cuando recibe una, la responde al cliente. Una vez hecho esto, vuelve a esperar otra solicitud. El código es el siguiente:

```java
import java.io.*;
import java.net.*;
import java.util.*;
/* * Server to process ping requests over UDP. */
public class UDPServer
{
    private static final double LOSS_RATE = 0.3;
    private static final int AVERAGE_DELAY = 100; //
milliseconds
    public static void main(String[] args) throws Exception
    {
     // Get command line argument.
     if (args.length != 1)
     {
          System.out.println("Required arguments: port");
          return;
     }
     int port = Integer.parseInt(args[0]);
     // Create random number generator for use in simulating
     // packet loss and network delay.
     Random random = new Random();
     // Create a datagram socket for receiving and sending UDP
packets
     // through the port specified on the command line.
     DatagramSocket socket = new DatagramSocket(port);
     // Processing loop.
     while (true)
     { // Create a datagram packet to hold incomming UDP packet.
          DatagramPacket request = new DatagramPacket(new
byte[1024], 1024);
          // Block until the host receives a UDP packet.
          socket.receive(request);
          // Print the recieved data.
          printData(request);
          // Decide whether to reply, or simulate packet loss.
          if (random.nextDouble() < LOSS_RATE)
          {
               System.out.println(" Reply not sent.");
               continue;
          }
          // Simulate network delay.
          Thread.sleep((int) (random.nextDouble() * 2 *
AVERAGE_DELAY));
          // Send reply.
          InetAddress clientHost = request.getAddress();
          int clientPort = request.getPort();
          byte[] buf = request.getData();
          DatagramPacket reply = new DatagramPacket(buf,
buf.length, clientHost, clientPort);
          socket.send(reply);
          System.out.println(" Reply sent.");
     }
    }


    /* * Print ping data to the standard output stream. */
    private static void printData(DatagramPacket request) throws
Exception
    {
     // Obtain references to the packet's array of bytes.
     byte[] buf = request.getData();
     // Wrap the bytes in a byte array input stream,
     // so that you can read the data as a stream of bytes.
     ByteArrayInputStream bais = new ByteArrayInputStream(buf);
     // Wrap the byte array output stream in an input stream
reader,
     // so you can read the data as a stream of characters.
     InputStreamReader isr = new InputStreamReader(bais);
     // Wrap the input stream reader in a bufferred reader,
     // so you can read the character data a line at a time.
     // (A line is a sequence of chars terminated by any
combination of \r and \n.)
     BufferedReader br = new BufferedReader(isr);
     // The message data is contained in a single line, so read
this line.
     String line = br.readLine();
     // Print host address and data received from it.
     System.out.println( "Received from " +
request.getAddress().getHostAddress() +": " + new
String(line) );
    }
}
```

Además, hemos creado el cliente UDP que se encarga de enviar solicitudes al servidor. Enviará 10 solicitudes, esperando un segundo entre recepción de solicitud y envío de la solicitud. Además, muestra por pantalla la respuesta de las solicitudes. El código del cliente es el siguiente:

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

public class UDPClient {

    public static void main (String args[]) throws Exception {

     //Use DataGramSocket for UDP connection
     DatagramSocket s = new DatagramSocket();

     // convert string "hello" to array of bytes, suitable for
     // creation of DatagramPacket

     // Now create a packet (with destination addr and port)
     InetAddress addr = InetAddress.getByName(args[0]);
```

```
      int port = Integer.parseInt(args[1]);

      for (int i=0; i<10; i++)
      {
            String str = String.format("PING %d %d\r\n", i,
System.nanoTime());
            byte outBuf[] = str.getBytes();

            DatagramPacket outPkt = new DatagramPacket(outBuf,
outBuf.length,

addr, port);
            s.send(outPkt);

            // create a packet buffer to store data from packets
received.
            byte inBuf[] = new byte[1000];
            DatagramPacket inPkt = new DatagramPacket(inBuf,
inBuf.length);
            // receive the reply from server.
            s.receive(inPkt);

            // convert reply to string and print to System.out
            String reply = new String(inPkt.getData(), 0,
inPkt.getLength());
            System.out.println(reply);
            Thread.currentThread().sleep(1000);
      }
     }
}
```

Para su uso el servidor ha de llamar a la aplicación de la siguiente forma:

```
      java UDPServer puerto
```

y el cliente ha de hacer la llamada de la siguiente forma

```
      java UDPClient DIR_Servidor_IP puerto
```