Sistemas Distribuidos
Protocolos UDP e IP Multicast en Java

Alberto Lafuente

Mikel Larrea

Dpto. ATC, UPV/EHU

Contenido

- Receptor UDP (udp_r.java)
- Emisor UDP (udp_s.java)
- Ejercicio UDP

- Receptor IP Multicast (ipmulticast_r.java)
- Emisor IP Multicast (ipmulticast_s.java)
- Ejercicio IP Multicast

Receptor UDP (udp_r.java) - 1/3

```
import java.net.*;

// This program waits to receive datagrams sent to a specified port.
// When it receives one, it displays the sending host and port,
// and prints the contents of the datagram as a string.

public class udp_r {
    public static void main(String args[]) throws Exception {
        if (args.length != 1) {
            System.out.println("Usage: java udp_r <port>");
            System.exit(0);
        }
        int port = Integer.parseInt(args[0]);
        byte[] buffer = new byte[1024];
        String s;
```

Receptor UDP (udp_r.java) - 2/3

```
        // Create a socket to listen on the port.
        DatagramSocket socket = new DatagramSocket(port);
        System.out.println("Reception socket created...");
        long expected = 1;

        for(;;) {
            // Create a packet with an empty buffer to receive data
            DatagramPacket packet = new DatagramPacket(buffer,
   buffer.length);
            // Wait to receive a datagram
            socket.receive(packet);
            // Convert the contents to a string
            s = new String(buffer, 0, packet.getLength());
            // Get the seuqence number as a long
            long sequence_number = Long.parseLong(s);
```

Receptor UDP (udp_r.java) - 3/3

```
            if (sequence_number == expected) {
                expected++;
                System.out.println("udp_r: received from " +
                           packet.getAddress().getHostName() + ":" +
                           packet.getPort() + ": " + s);
            }
            else {
                System.out.println("ERROR: unexpected sequence number: "
                                   + sequence_number);
                System.exit(-1);
            }
        }
    }
}
```

Emisor UDP (udp_s.java) - 1/3

```
import java.net.*;

// This program sends periodically a datagram to the specified (host &
   port)

public class udp_s {
    public static void main(String args[]) throws Exception {
        if (args.length != 3) {
            System.out.println("Usage: java udp_s <host> <port> <period in
   ms>");
            System.exit(0);
        }
        // Get the internet address of the specified host and the port
   number
        InetAddress address = InetAddress.getByName(args[0]);
        int port = Integer.parseInt(args[1]);
```

Emisor UDP (udp_s.java) - 2/3

```
// Create a socket, and send the packet through it
DatagramSocket socket = new DatagramSocket();
System.out.println("Sending socket created...");
String s = new String();
long sequence_number = 0;
long period = Long.parseLong(args[2]);

for (;;) {
    sequence_number++;
    Long sequence = new Long(sequence_number);
    s = sequence.toString();
    // Convert the string s to an array of bytes
    byte[] message = new byte[1024];
    message = s.getBytes();
```

Emisor UDP (udp_s.java) - 3/3

```
            // Initialize the packet with data and address
            DatagramPacket packet = new DatagramPacket(message,
    s.length(), address, port);

            // send the packet through the socket
            System.out.println("udp_s: sending message " +
    sequence_number);
            socket.send(packet);

            // Wait for period milliseconds
            Thread.sleep(period);
        }
    }
}
```

Ejercicio UDP

- Probar los programas udp_r y udp_s
  - Probar con varios emisores a un receptor
    - ¿Qué sucede y por qué?
  - Probar con diferentes periodos de envío
    - ¿Se pierde algún mensaje? ¿Cuándo y por qué?

- Adaptar udp_r y udp_s para que el emisor se quede a la espera de la confirmación de la correcta recepción de cada mensaje

Receptor IP Multicast - 1/2

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

public class ipmulticast_r {
    public static void main(String[] args) throws Exception {
        int port = 4000;
        String message = null;
        InetAddress address = null;
        MulticastSocket socket = null;
        DatagramPacket packet = null;
        try {
            address = InetAddress.getByName("224.0.0.1");
        }
        catch (UnknownHostException e) {
            System.out.println("Error: " + e.toString());
        }
```

Receptor IP Multicast - 2/2

```
        try {
            socket = new MulticastSocket(port);
            socket.joinGroup(address);
        }
        catch(IOException e) {
            System.out.println("Error: " + e.toString());
        }
        System.out.println("ipmulticast_r ready...");
        while (true) {
            byte buffer[] = new byte[1024];
            packet = new DatagramPacket(buffer, buffer.length);
            socket.receive(packet);
            message = new String(buffer, 0, packet.getLength());
            System.out.println("Received: " + message);
        }
    }
}
```

Emisor IP Multicast - 1/3

```
import java.net.*;
import java.io.*;
import java.util.Random;
public class ipmulticast_s {
    public static void main(String[] args) throws Exception {
        if (args.length != 1) {
            System.out.println("Usage: java ipmulticast_s <your name>");
            System.exit(0);
        }
        String message;
        int n = 1;
        InetAddress address = null;
        MulticastSocket socket = null;
        DatagramPacket packet = null;
        Random r = new Random();
        long t;
```

Emisor IP Multicast - 2/3

```
try {
    address = InetAddress.getByName("224.0.0.1");
}
catch (UnknownHostException e) {
    System.out.println("Error: " + e.toString());
}
try {
    socket = new MulticastSocket();
    // socket.setTimeToLive(255);
}
catch (IOException e) {
    System.out.println("Error: " + e.toString());
}
```

Emisor IP Multicast - 3/3

```
        while (true) {
            message = args[0] + " sender's message #" +
    Integer.toString(n++);
            byte[] data = new byte[1024];
            data = message.getBytes();
            packet = new DatagramPacket(data, data.length, address, 4000);

            socket.send(packet);
            System.out.println("Sent: " + message);

            t = (r.nextInt(10) + 1) * 100;  // value between 100 and 1000
            Thread.sleep(t);
        }
    }
}
```

Ejercicio IP Multicast

- Probar los programas ipmulticast_r e ipmulticast_s
  - Probar con varios emisores y varios receptores

- “Privatizar” dichos programas
  - Adaptar ipmulticast_r para que muestre sólo los mensajes enviados desde un puerto dado
  - Adaptar ipmulticast_s para que envíe los mensajes desde un puerto indicado por el usuario