Technical University of Cartagena

Telecommunications Engineering School

DIGITAL CONTENTS LABORATORY

Laboratory Content 3. UDP Multicast Chat

Professor:

Antonio Javier García Sánchez.

1. Objectives

In this laboratory content, the student will implement an **UDP Multicast Chat**. The functionality is comprised in the following items:

- Several clients written the messages to send
- The messages are dispatched via UDP to a server
- The server receives messages and display them per screen

Figure 1 depicts the services to implement.

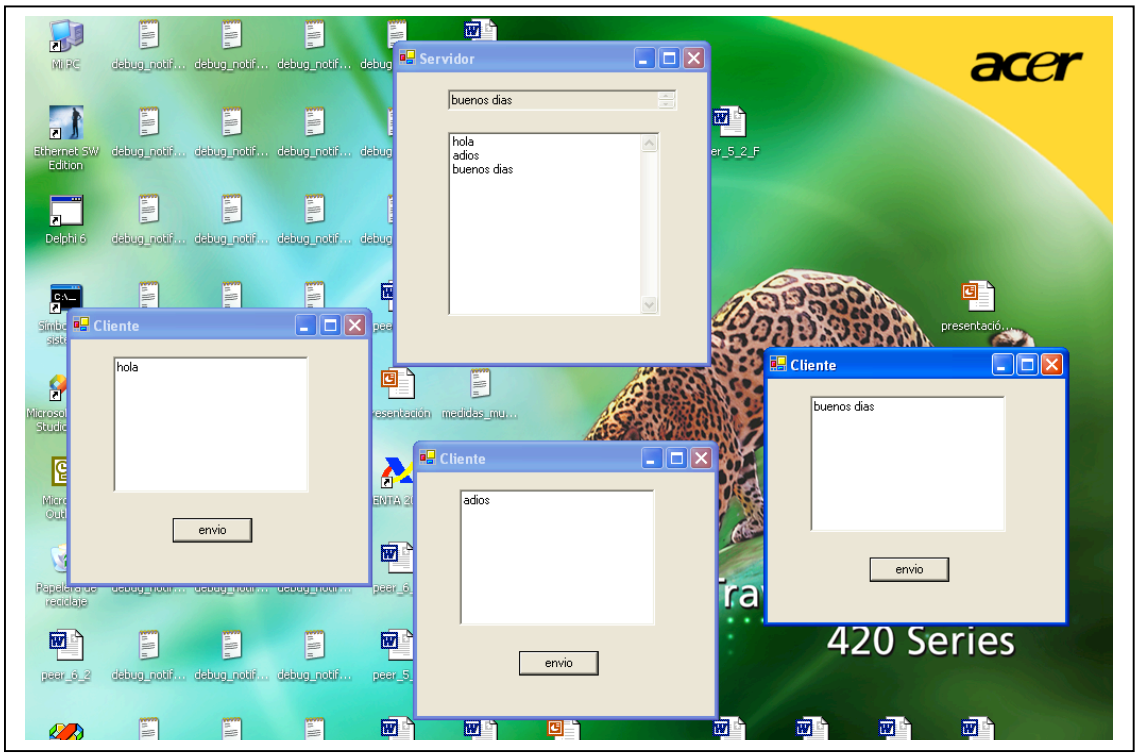

Figure 1. Initial aspect of services to develop

2. Laboratory Content Description

2.1.Server Application

Server requirements are the following ones:

- To create a multicast group
- To receive data from the entire clients joining to the multicast group
- To display messages sent by clients in an application

2.1.1. Multicast group creation

The using paths to include are two:

```
using System.Net;
using System.Net.Sockets;
```

The *UdpClient* class is employed to generate a multicast group. *UdpClient* provides network services through the User Datagram Protocol (UDP). One of methods selects *JoinMulticastGroup,* which internally supports the creation and the connection to the multicast group.

```
UdpClient udpclient = new UdpClient(8080);
IPAddress multicastaddress=IPAddress.Parse("224.0.0.1");
udpclient.JoinMulticastGroup(multicastaddress);
```

IPEndPoint points out a network end-point like an IP address and port number. In the case of the server, it has not to join with any another server; therefore, the reference is null.

```
IPEndPoint remote=null;
```

2.1.2. Data Reception

The server remains waiting for receiving data of clients. To this end, the *Receive* method belong to the *UdpClient* class receives an array of bytes.

2.1.3. Server Graphical User Interface (GUI)

Data from clients have to be displayed in the server. This goal is accomplished by means of *ListBox* and *Textbox* components:

- *ListBox,* containing a history of all the messages received
- *Textbox*, containing the last message received

Server collects an array of bytes in the *Receive* method. *ListBox* and *Textbox* components support *strings*. To convert array of bytes to *string*, .NET framework solves it using the *Encoding* class. *Encoding* represents a character codification. In this class, the *Unicode* type is employed to code each character as two consecutive bytes.

NOTE. As commented, the server application is locked until receiving data from clients, resulting in the not execution of the GUI. To solve it, the student will implement a *thread* including the reception code of data dispatched by clients.

The implementation of *threads* is carried out by the *Thread* class, adding the following *using*:

```
using System.Threading;
```

An example would be the following one:

```
Thread t = new Thread(new ThreadStart(MyThreadMethod));
t.Start();

private void MyThreadMethod()
            {
                  while(true)
                  {
                     .
                     //Código de recepción de datos
                     .
                  }
```

2.2. Client Application

Client requirements to develop are the following ones:

- To add the multicast group
- To send messages to the client

2.2.1. Multicast group inclusión

As in the server case, the *UdpClient* class is employed, following the same steps. However, the sentence *IPEndPoint* is written so:

```
IPEndPoint remoteep = new IPEndPoint( multicastaddress, 8080 );
```

2.2.2. Client Graphical User Interface (GUI)

The student will employ the *RichTexBox* component to write texts in the PC screen. These texts are *strings* to send. Additionally, the student will implement a *button* to run the delivery of messages to the server.

2.2.3. Messages Delivery

The client sends messages to the server, employing the *Send* method belong to *UdpClient* class. A datagram is transmitted to the remote end-point at the following syntax:

public int Send(byte[], int, IPEndPoint);

Note that the *strings* must be converted to array of bytes to be compatible with the *Send* method. As previos case, the student will use the *Encoding* class.