Client-Server Application Documentation

By Maria Carmen Taylaran and John Loyd Villarino

BSIT-3B

Server-Client Application

Just a basic chat program inside your desktop was the first version. The idea is to use the datagram sockets to connect the client to the server. We need to provide the following method to access the functions relevant to datagram sockets:

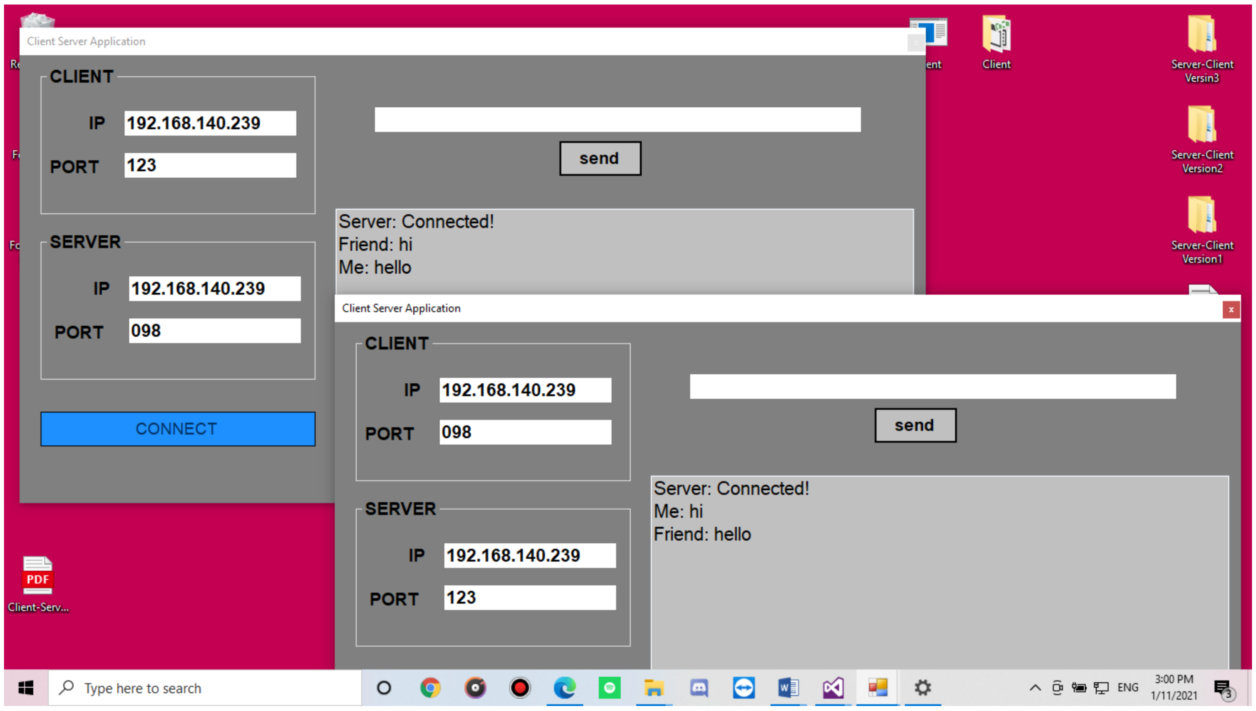
using System.Net.Sockets – refers to establishes connection to a remote host.

using System.Net – provides common methods for sending data to and receiving data from a resource identified by URI.

```
using System.Net;

using System.Net.Sockets;
```

Server-Client Application Version1:

This application required an IP Address t together with a port. Two Windows Application are needed to test run the program .After that you must make sure the configuration of the Client to the server is correct, so you can able to connect if you were successfully connected from client to server you can see “Server: Connected” on the list box, and then you can began chatting.



CLIENT
IP 192.168.140.239
PORT 123
SERVER
IP 192.168.140.239
PORT 098

```
Server: Connected!
Friend: hi
Me: hello
```

CLIENT
IP 192.168.140.239
PORT 098
SERVER
IP 192.168.140.239
PORT 123

```
Server: Connected!
Me: hi
Friend: hello
```

Figure 1. Server Client Version 1- UI

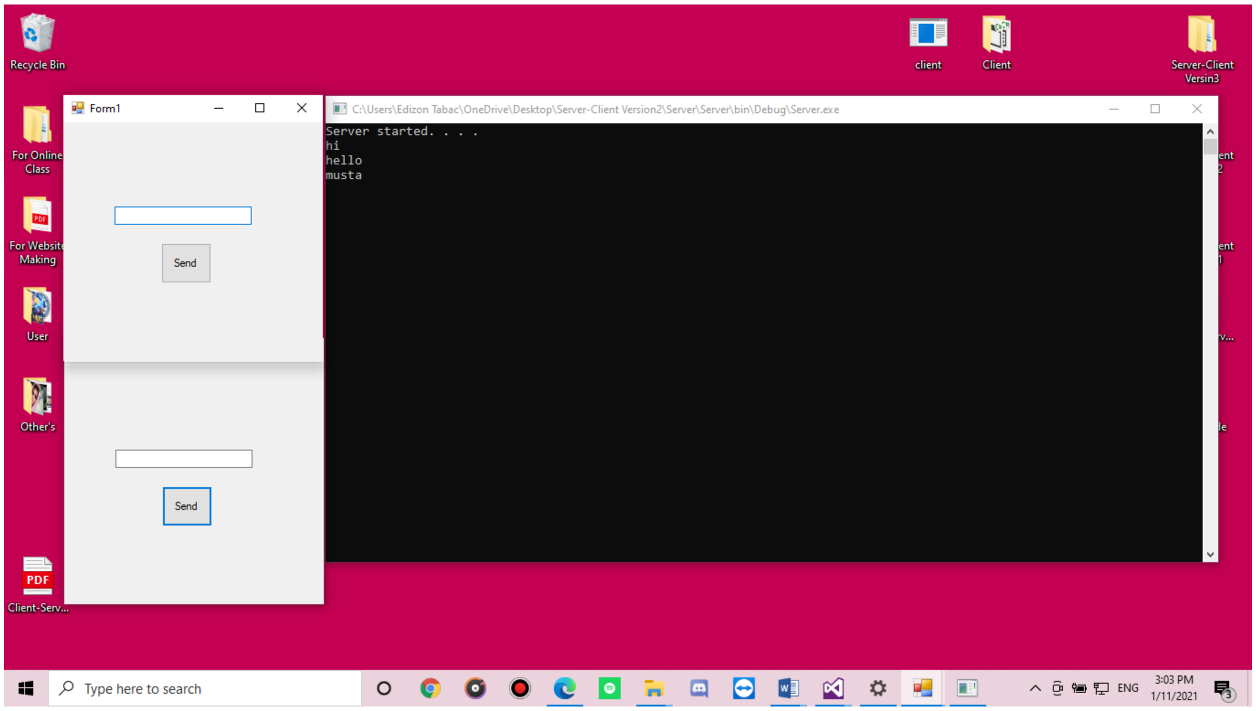
Source code

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;

namespace Server_Client_app
{
    public partial class Form1 : Form
    {
        Socket socket;
        EndPoint epLocal, epRemote;
        byte[] buffer;

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            //setup Socket
            socket = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Ud
p);

            socket.SetSocketOption(SocketOptionLevel.Socket, SocketOptionName.ReuseAddress,
 true);



            //get local IP
            txtlocalIP.Text = GetlocalIP();
            txtremoteIP.Text = GetlocalIP();
        }

        private void btnConnect_Click(object sender, EventArgs e)
        {
            //bind Socket
            epLocal = new IPEndPoint(IPAddress.Parse(txtlocalIP.Text), Convert.ToInt32(txtlocalPo
rt.Text));
            socket.Bind(epLocal);
            //connect to Remote IP
            epRemote = new IPEndPoint(IPAddress.Parse(txtremoteIP.Text), Convert.ToInt32(txtre
motePort.Text));
            socket.Connect(epRemote);

            //Listen Specific POrt
            buffer = new byte[1500];
            socket.BeginReceiveFrom(buffer, 0, buffer.Length, SocketFlags.None, ref epRemote, ne
w AsyncCallback(MessageCallBack), buffer);

            btnConnect.Enabled = false;
            listMessage.Items.Add("Server: Connected!");
        }

        private void MessageCallBack(IAsyncResult aResult)
        {
            try
            {
                byte[] recieveData = new byte[1500];
                recieveData = (byte[])aResult.AsyncState;

                //convert byte[] to string
                ASCIIEncoding aEncoding = new ASCIIEncoding();
                string recieveMEssage = aEncoding.GetString(recieveData);

                //Adding this message into textbox
                listMessage.Items.Add("Friend: " + recieveMEssage);

                buffer = new byte[1500];
                socket.BeginReceiveFrom(buffer, 0, buffer.Length, SocketFlags.None, ref epRemote,
new AsyncCallback(MessageCallBack), buffer);
            }
            catch(Exception ex)
            {
                MessageBox.Show(ex.ToString());
            }
        }

        private void btnSend_Click(object sender, EventArgs e)
        {
            //convert message string to byte[]
            ASCIIEncoding aEncoding = new ASCIIEncoding();
            byte[] sendingMessage = new byte[1500];
            sendingMessage = aEncoding.GetBytes(txtMessage.Text);

            //sending the encoded message
            socket.Send(sendingMessage);

            //add to the list box
            listMessage.Items.Add("Me: "+ txtMessage.Text);

            txtMessage.Clear();

        }

        private void btnExit_Click(object sender, EventArgs e)
        {
            Close();
        }

        private string GetlocalIP()
        {
            IPHostEntry host;
            host = Dns.GetHostEntry(Dns.GetHostName());

            foreach(IPAddress ip in host.AddressList)
            {
                if (ip.AddressFamily == AddressFamily.InterNetwork)
                {
                    return ip.ToString();
                }
            }
```

```
            return "127.0.0.1";
        }



    }
}
```

Server-Client Application Version 2

This application has a the same method to the version 1. But on this case it has two application to be run; Client and the Server. The Client is running through Windows Form Application, while the Server was on a Console Application. The Client send messages to the Server, while the server records all the messages. The server here cannot send any messages to the Client.



```
C:\Users\Edizon Tabac\OneDrive\Desktop\Server-Client Version2\Server\Server\bin\Debug\Server.exe
Server started. . . .
hi
hello
musta
```

Source Code for Client Windows Form

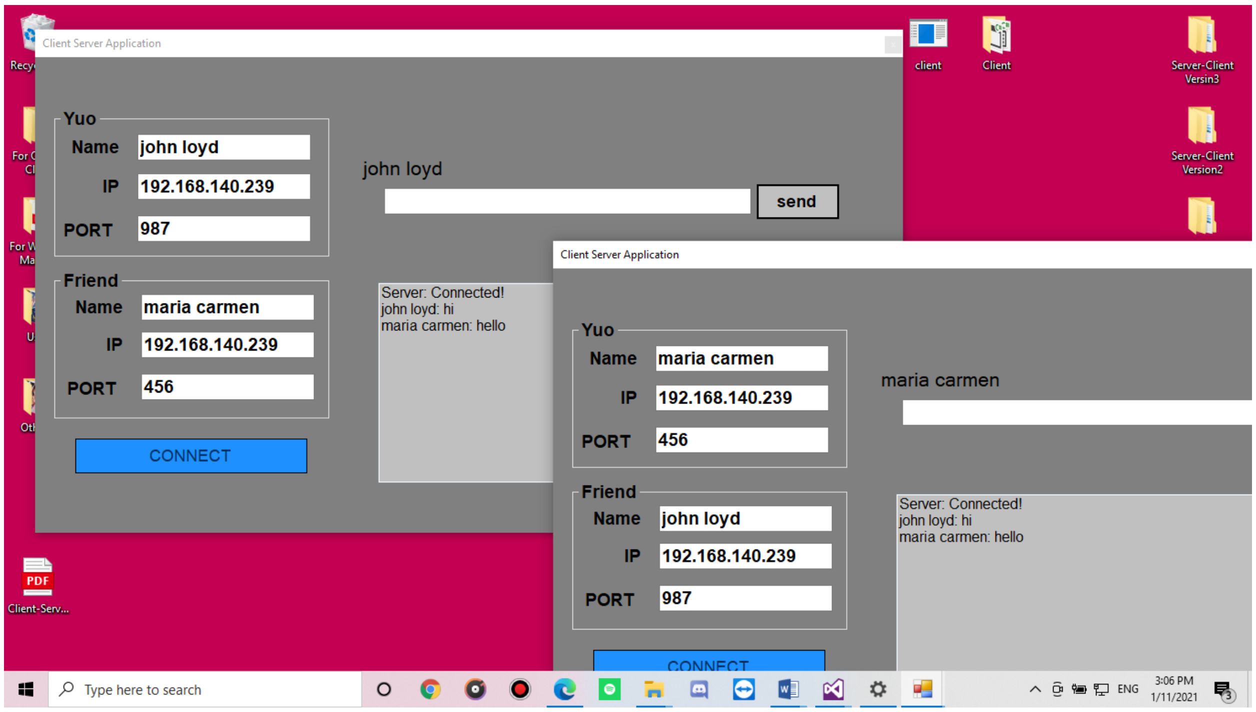
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
namespace Client
{
    public partial class Form1 : Form
    {
        string serverIP = "localhost";
        int port = 8080;
        public Form1()
        {
            InitializeComponent();
        }

        private void Send_Click(object sender, EventArgs e)
        {
            TcpClient client = new TcpClient(serverIP, port);
            int byteCount = Encoding.ASCII.GetByteCount(message.Text+1);
            byte[] sendData = new byte[byteCount];
            sendData = Encoding.ASCII.GetBytes(message.Text);
            NetworkStream stream = client.GetStream();
            stream.Write(sendData, 0, sendData.Length);
            stream.Close();
            client.Close();
            message.Text = "";


        }

        private void Send_KeyDown(object sender, KeyEventArgs e)
        {
            if(e.KeyCode==Keys.Enter||e.KeyCode==Keys.Return)
            {
                Send.PerformClick();
            }
        }
    }
}
```

Source Code for Server Console Application

```
using System;

using System.Collections.Generic;

using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Net;
using System.Net.Sockets;

namespace Server
{
    class Program
    {
        static void Main(string[] args)
        {
            IPAddress ip = Dns.GetHostEntry("localhost").AddressList[0];
            TcpListener server = new TcpListener(ip, 8080);
            TcpClient client = default(TcpClient);
            try
            {
                server.Start();
                Console.Write("Server started. . . . \n");



            }
            catch (Exception x)
            {
                Console.Write(x.ToString());

            }
            while(true)
            {
                client = server.AcceptTcpClient();
                byte[] recieveBuffer = new byte[100];
                NetworkStream stream = client.GetStream();
                stream.Read(recieveBuffer, 0, recieveBuffer.Length);
                StringBuilder msg = new StringBuilder();
                foreach(byte b in recieveBuffer)
                {
                    if(b.Equals(00))
                    {
                        break;
                    }else
                    {
                        msg.Append(Convert.ToChar(b).ToString());
                    }
                }
                Console.WriteLine(msg.ToString());
            }
        }
    }
}
```

Server-Client Application version 3

This version was also based on the version 1. There is no much difference on it, the only unique on this version is the added name field.



john loyd 192.168.140.239 987
maria carmen 192.168.140.239 456

```
Server: Connected!
john loyd: hi
maria carmen: hello
```

maria carmen 192.168.140.239 456
john loyd 192.168.140.239 987

Figure 3. Server Client Version 3

Source code

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;

namespace Server_Client_app
{
    public partial class Form1 : Form
    {
        Socket socket;

        EndPoint epLocal, epRemote;

        byte[] buffer;


        public Form1()

        {

            InitializeComponent();

        }


        private void Form1_Load(object sender, EventArgs e)

        {

            //setup Socket

            socket = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Ud
p);


            socket.SetSocketOption(SocketOptionLevel.Socket, SocketOptionName.ReuseAddress,
 true);




            //get local IP

            txtlocalIP.Text = GetlocalIP();

            txtremoteIP.Text = GetlocalIP();

        }


        private void btnConnect_Click(object sender, EventArgs e)

        {

            try

            {

                if (checkconfig() == true)

                {
                    //bind Socket

                    epLocal = new IPEndPoint(IPAddress.Parse(txtlocalIP.Text), Convert.ToInt32(txtloc
alPort.Text));

                    socket.Bind(epLocal);

                    //connect to Remote IP

                    epRemote = new IPEndPoint(IPAddress.Parse(txtremoteIP.Text), Convert.ToInt32(
txtremotePort.Text));

                    socket.Connect(epRemote);


                    //Listen Specific POrt

                    buffer = new byte[1500];

                    socket.BeginReceiveFrom(buffer, 0, buffer.Length, SocketFlags.None, ref epRemot
e, new AsyncCallback(MessageCallBack), buffer);


                    btnConnect.Enabled = false;

                    listMessage.Items.Add("Server: Connected!");

                    label7.Text = Youname.Text;

                }


                else

                {

                    MessageBox.Show("configuration error!", "server client", MessageBoxButtons.OK,
MessageBoxIcon.Error);

                    return;

                }




            }catch(Exception ex)

            {

                MessageBox.Show("invalid configuration","server-
client", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }
```

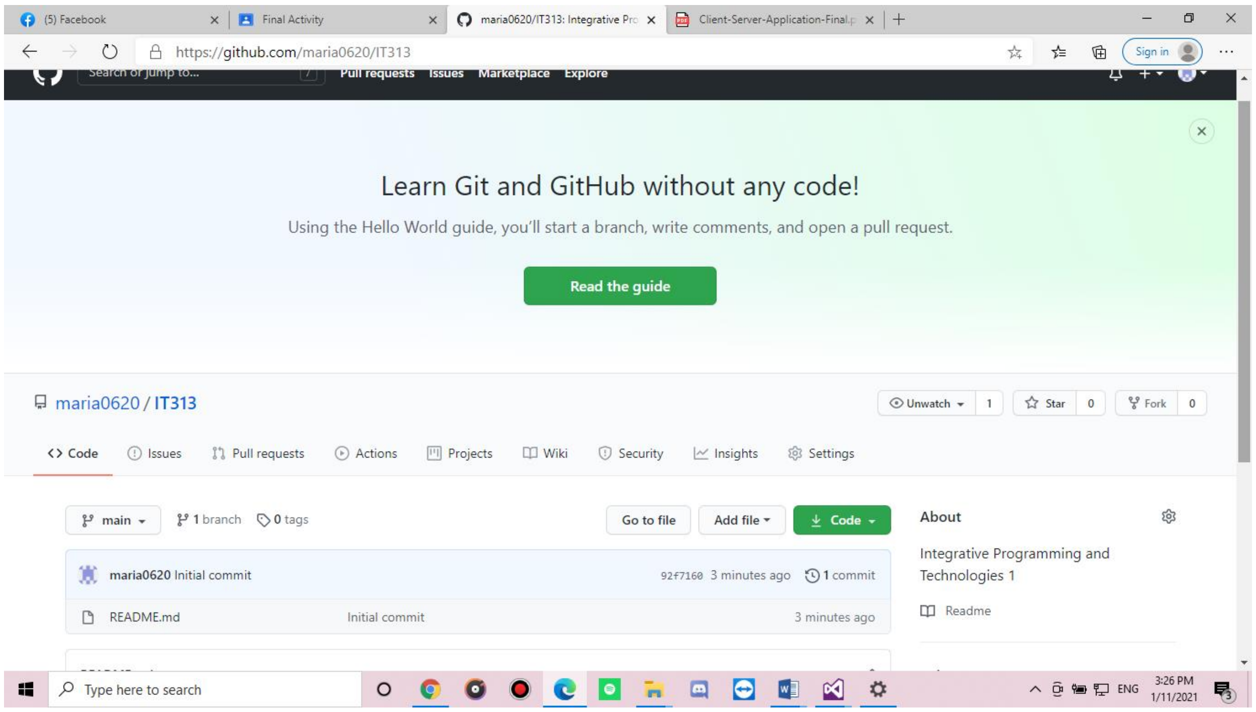
```
        }
        private bool checkconfig()
        {
            if (Youname.Text != "" && Fname.Text != "" && txtlocalIP.Text != "" && txtremoteIP.Text !
= "" && txtremotePort.Text != ""&&txtlocalPort.Text!="")
            {
                return true;
            }
            else
                return false;
        }



        private void MessageCallBack(IAsyncResult aResult)
        {
            try
            {
                byte[] recieveData = new byte[1500];
                recieveData = (byte[])aResult.AsyncState;

                //convert byte[] to string
                ASCIIEncoding aEncoding = new ASCIIEncoding();
                string recieveMEssage = aEncoding.GetString(recieveData);

                //Adding this message into textbox
                listMessage.Items.Add(Fname.Text+": " + recieveMEssage);

                buffer = new byte[1500];
                socket.BeginReceiveFrom(buffer, 0, buffer.Length, SocketFlags.None, ref epRemote,
new AsyncCallback(MessageCallBack), buffer);
```

```
        }
        catch(Exception ex)
        {
            MessageBox.Show(ex.ToString());
        }
    }

    private void btnSend_Click(object sender, EventArgs e)
    {
        //convert message string to byte[]
        ASCIIEncoding aEncoding = new ASCIIEncoding();
        byte[] sendingMessage = new byte[1500];
        sendingMessage = aEncoding.GetBytes(txtMessage.Text);

        //sending the encoded message
        socket.Send(sendingMessage);

        //add to the list box
        listMessage.Items.Add(label7.Text+": "+ txtMessage.Text);

        txtMessage.Clear();

    }

    private void btnExit_Click(object sender, EventArgs e)
    {
        Close();
    }

    private void panel1_Paint(object sender, PaintEventArgs e)
```

```
        {

        }

        private string GetlocalIP()
        {
            IPHostEntry host;
            host = Dns.GetHostEntry(Dns.GetHostName());

            foreach(IPAddress ip in host.AddressList)
            {
                if (ip.AddressFamily == AddressFamily.InterNetwork)
                {
                    return ip.ToString();
                }
            }

            return "127.0.0.1";
        }

    }
}
```

All of the document are uploaded at gethub:

[maria0620/IT313: Integrative Programming and Technologies 1 (github.com)]


https://github.com/maria0620/IT313

Learn Git and GitHub without any code!

Using the Hello World guide, you'll start a branch, write comments, and open a pull request.

maria0620 / IT313

maria0620 Initial commit 92f7160 3 minutes ago 1 commit

README.md Initial commit 3 minutes ago

About

Integrative Programming and Technologies 1