***Client-Server Application Documentation***

***By Edizon P. Tabac and Jeanette R. Gabrillo BSIT – 3A***

***Server-Client Application:***

This are application that consists of a client program that connect to a server program’s services. Through naming the roles of the server program, the client requests services from the cloud. The client and the server communicate through a communication layer, also referred as the cloud platform, in a distributed computing environment where the client software and the server program run on different machines and maybe even on various devices.

The following are typical features of a client/server application:

* A client program does not need to be aware of the actual subprograms that provide a service
* A server program can provide multiple services
* Typically, the server program runs on a machine that is remote from the machine running the client program.
* A Client sends messages to server as well as to the connected clients.
* A Client views the messages from the clients and server.
* The client relies on sending a request to another program in order to access a service made available by a server.

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.

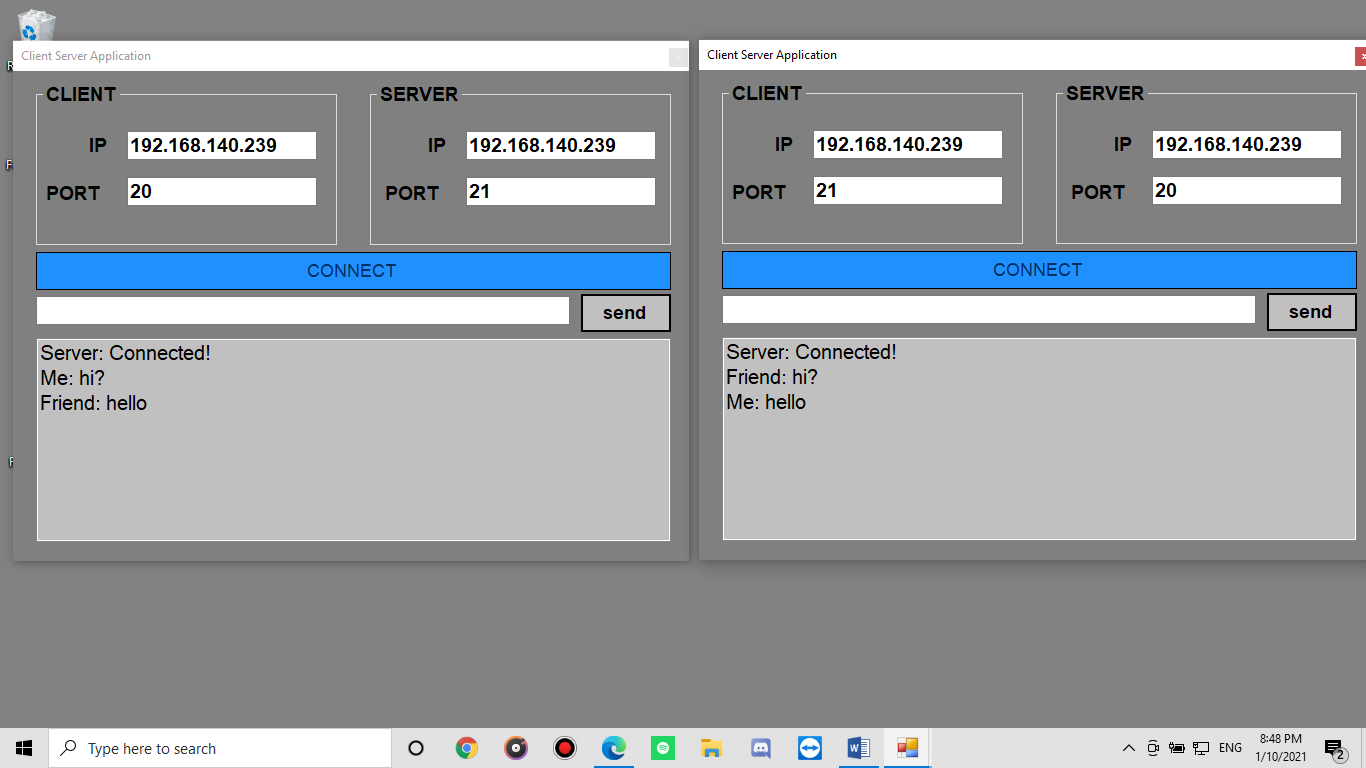
**using System.Net** – provides common methods for sending data to and receiving data from a resource identified by URI.

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.



*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.Udp);

            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(txtlocalPort.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 epRemote, new 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";

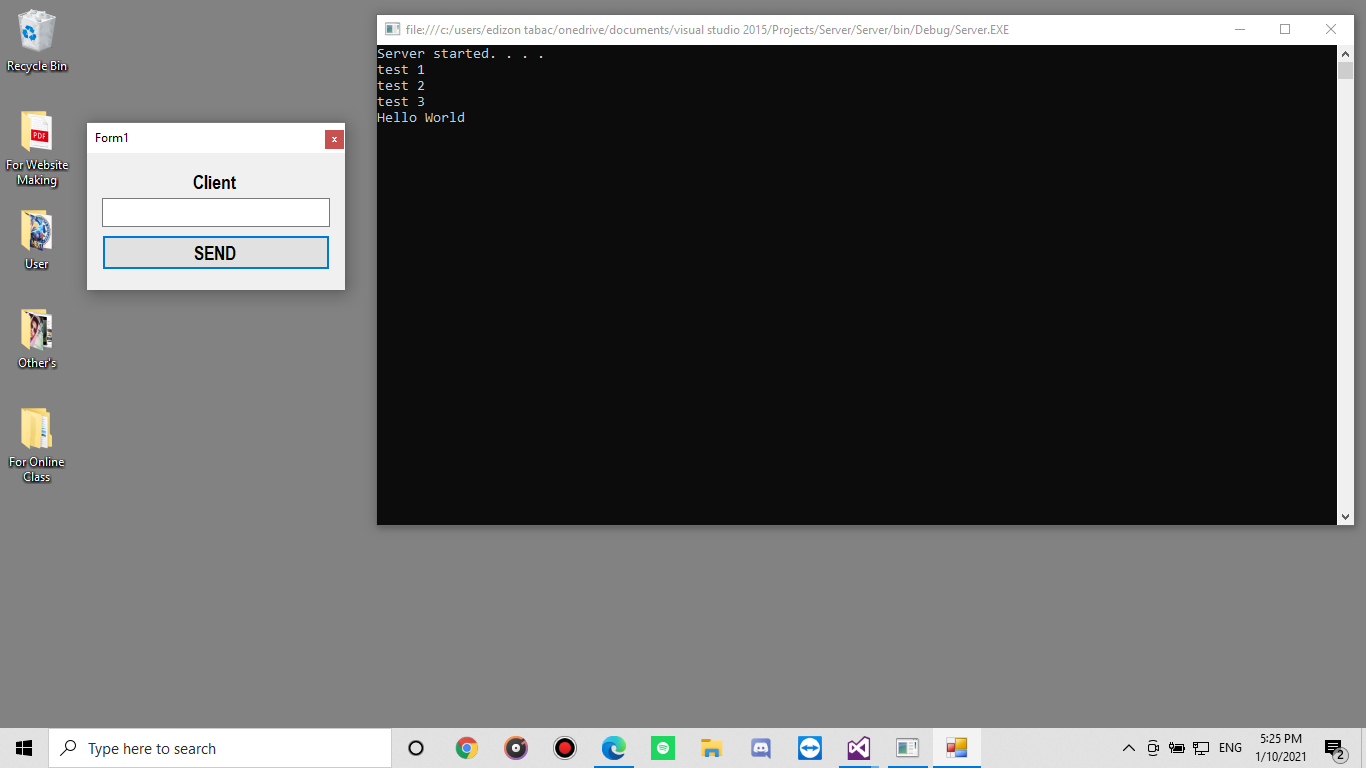
        }

    }

}

***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.



*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());

            }

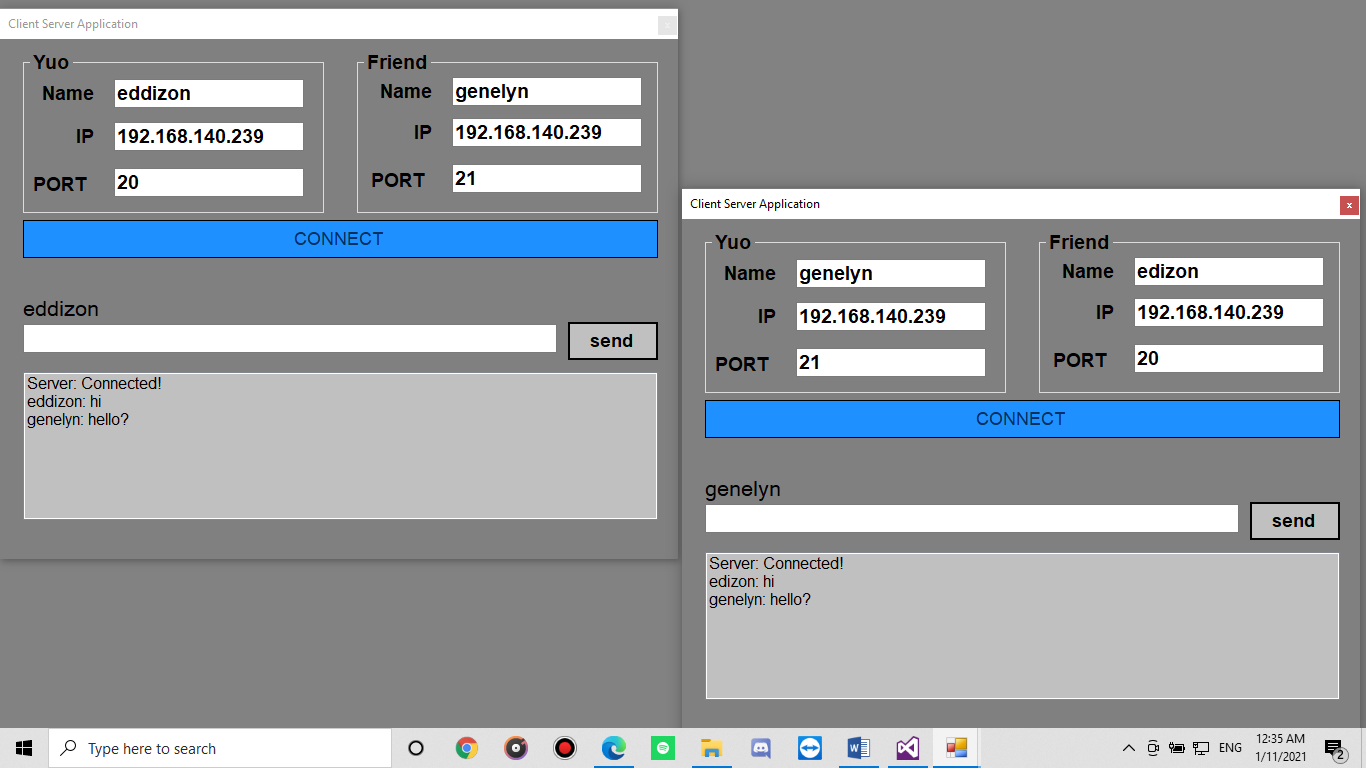
        }

    }

}

***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.



*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.Udp);

            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(txtlocalPort.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 epRemote, 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";

        }

    }

}