TCP_MSG

SERVER

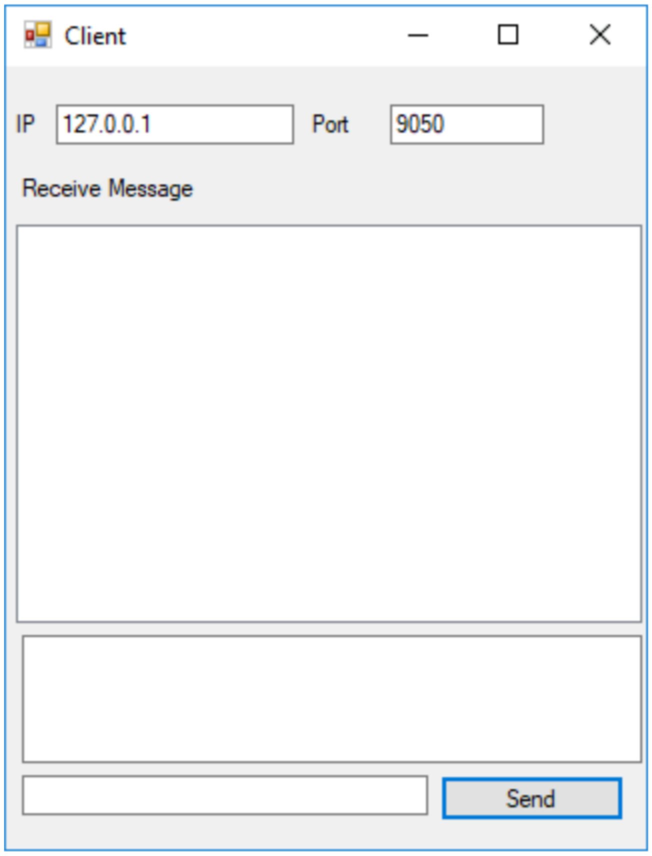
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.Collections;
using System.Threading;
using System.IO;

namespace ServerTCP_
{
        public partial class FormServer : Form
        {

                public FormServer()
                {
                        InitializeComponent();
                }

                delegate void ShowMessage(String Message_);

                public void Show(string Message_)
                {
                        if (lstReceive.InvokeRequired)
                        {
                                ShowMessage message = new ShowMessage(Show); lstReceive.Invoke(message,
                                new object[] {Message_}); return;
                        }
                        lstReceive.Items.Add(Message_);
                }

                public void ShowIP(string Message_)
                {
                        if (txtIP.InvokeRequired)
                        {
                                ShowMessage message = new ShowMessage(ShowIP); txtIP.Invoke(message,
                                new object[] { Message_ }); return;
                        }
                        txtIP.Text = Message_;
                }

                string returnString = "";
                char[] s;


                TcpListener listener;
                TcpClient client;
                Thread listenerThread;

                bool isStart = false;
```

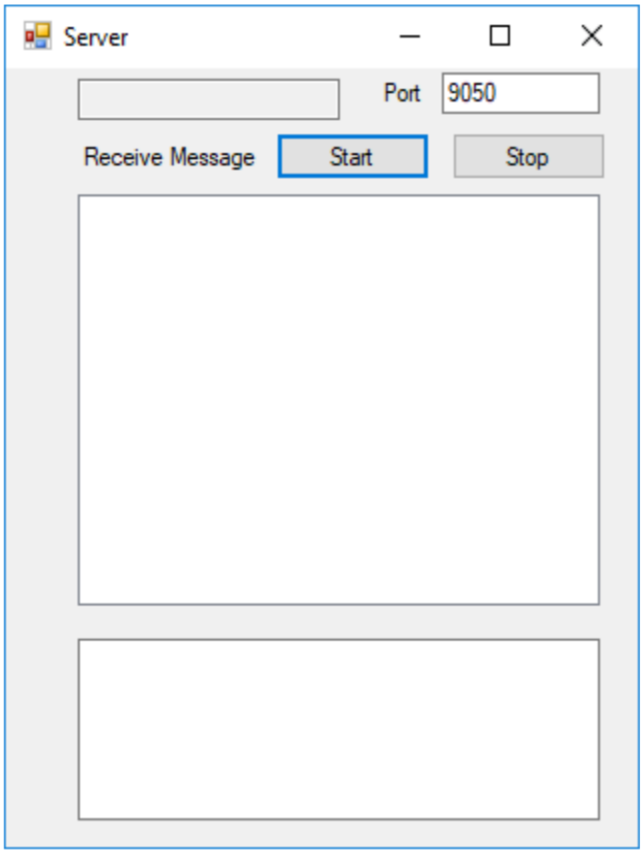
```
public void ListenMessage()
{
        try
        {
            listener = new TcpListener(IPAddress.Any, int.Parse(txtPort.Text));
            listener.Start();
            isStart = true;
            listenerThread = new Thread(new ThreadStart(WaitingConnect)); listenerThread.Start();
        }
        catch (Exception exc)
        {
                MessageBox.Show("Không thể khởi động máy chủ ! \n" + exc.ToString());
        }

}

public void WaitingConnect()
{

        while (isStart)
        {
                IPEndPoint remote = new IPEndPoint(IPAddress.Any, 0); client =
                listener.AcceptTcpClient();
                StreamReader reader = new StreamReader(client.GetStream()); string data = reader.ReadLine();
                if (data.Trim().Length > 0)
                {
                         Show("Nhận thông điệp từ -> " + remote.Address.ToString() + " : " + data);
                        StreamWriter writer = new StreamWriter(client.GetStream());

                        writer.WriteLine(data);
                        writer.Flush();
                }
        }

}

private void btnStart_Click(object sender, EventArgs e)
{
        lstReceive.Items.Clear();
        new Thread(new ThreadStart(ListenMessage)).Start();
        lstReceive.Items.Add("Đang lắng nghe ... ");
}

private void lstReceive_SelectedIndexChanged(object sender, EventArgs e)
{
        if (lstReceive.SelectedItem != null)
                txtResult.Text = lstReceive.SelectedItem.ToString();
}

private void Stop_Click(object sender, EventArgs e)
{
        if (listener != null)
        {
                isStart = false;
                listener.Stop();
```

```
                    }
                }
            }
        }
```

CLIENT

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

namespace ClientTCP
{
    public partial class FormClient : Form
    {
        public FormClient()
        {
            InitializeComponent();
        }
        public delegate void ShowMessage(string Message_);

        public void Show(string Message_)
        {
            if (lstReceive.InvokeRequired)
            {
                ShowMessage message = new ShowMessage(Show); lstReceive.Invoke(message, new
                object[] { Message_ }); return;
            }
            lstReceive.Items.Add(Message_);
        }
        TcpClient tcpClient;
        StreamReader reader;
        StreamWriter writer;

        private void btnSend_Click(object sender, EventArgs e)
        {
            tcpClient = new TcpClient(); tcpClient.Connect(txtIP.Text, int.Parse(txtPort.Text));
            writer = new StreamWriter(tcpClient.GetStream()); writer.WriteLine(txtMessage.Text);
            writer.Flush();
            new Thread(new ThreadStart(ReceiveMessage)).Start();

        }

        private void ReceiveMessage()
        {
            reader = new StreamReader(tcpClient.GetStream()); string strReturn =
            reader.ReadLine(); if (strReturn.Trim().Length > 0)
                Show("Thông điệp từ Server : " + strReturn);

        }

       private void txtMessage_KeyPress(object sender, KeyPressEventArgs e)
       {
            if (e.KeyChar == (char)13)
                btnSend_Click(sender, e);
        }

        private void lstReceive_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (lstReceive.SelectedItem != null)
                txtResult.Text = lstReceive.SelectedItem.ToString();
        }
        }
    }
```


Client
IP
127.0.0.1
Port
9050
Receive Message
Send


Server
Port
9050
Receive Message
Start
Stop

UDP_MSG
SERVER

```
using System;
using System.Collections.Generic; using System.ComponentModel;
using System.Data; using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Net.Sockets;
using System.Threading;
using System.Net;
namespace SrvUDPMSG
{
        public partial class Form1 : Form
        {
                public Form1()
                {
                        InitializeComponent();
                }

                delegate void ShowMessage(String Message);

                public void Show(string Message)
                {
                        if (lstReceive.InvokeRequired)
                        {
                                ShowMessage message = new ShowMessage(Show); lstReceive.Invoke(message, new object[]
                                        {Message});
                                return;
                        }
                        lstReceive.Items.Add(Message);
                }

                public void ShowIP(string Message)
                {
                        if (txtIP.InvokeRequired)
                        {
                                ShowMessage message = new ShowMessage(ShowIP); txtIP.Invoke(message, new object[] { Message });
                                return;
                        }
                        txtIP.Text = Message;
                }


                public string Process(string Message)
                {
                        return Message.Trim();
                }
                public void ListenMessage()
                {
                        try
                        {
                                //      Khởi tạo UdpClient lắng nghe tại cổng chỉ định
                                //      UdpClient udpClient = new
                    UdpClient(int.Parse(txtPort.Text));
                    while (true)
                    {
                              IPEndPoint remote = new
                                        IPEndPoint(IPAddress.Any, 0); ShowIP(remote.Address.ToString());
                              Byte[] receiveBytes = udpClient.Receive(ref remote);
                                        // Chuyển đổi mảng Byte thành chuỗi Unicode để xử lý
                                        string data =
                                                  Encoding.Unicode.GetString(receiveBytes);
                                        string msg = " Thông điệp nhận từ -> " +
                                            remote.Address.ToString() + " : " + data.ToString();
                                        Show(msg);
                                        data = Process(data);
                                        Byte[] sendBytes = Encoding.Unicode.GetBytes(data);
                                        udpClient.Send(sendBytes, sendBytes.Length, remote);
                                        msg = " Gởi tới -> " + remote.Address.ToString() + " : " + data.ToString();
                                        Show(msg);


                                }
                        }
                        catch (Exception exc)
                        {

                                MessageBox.Show(" Không thể khởi động máy chủ ! \n" + exc.ToString());
                        }
                }


                private void btnStart_Click(object sender, EventArgs e)
                {
                        lstReceive.Items.Clear();
                        new Thread(new ThreadStart(ListenMessage)).Start(); lstReceive.Items.Add("Đang lắng nghe ... ");
                }
                private void lstReceive_SelectedIndexChanged(object sender, EventArgs e)
                {
                        if (lstReceive.SelectedItem != null)
                                txtResult.Text = lstReceive.SelectedItem.ToString();
                }


        }
}
```

CLIENT

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


namespace ClientUDPMSG
{
        public partial class Form1 : Form
        {
                public Form1()
                {
                        InitializeComponent();
                }
                                        public delegate void ShowMessage(string Message);


                public void Show(string Message)
                {
                        if (lstReceive.InvokeRequired)
                        {
                                ShowMessage message = new ShowMessage(Show); lstReceive.Invoke(message, new object[] { Message});
                                return;
                        }
                        lstReceive.Items.Add(Message);
                }


                private void btnSend_Click(object sender, EventArgs e)
                {
                        if (txtIP.Text.Trim().Length == 0)
                                MessageBox.Show("Vui lòng nhập vào địa chỉ IP!");
                        else
                        {
                                UdpClient udpClient = new UdpClient();

                                udpClient.Connect(txtIP.Text, int.Parse(txtPort.Text));
                                if (txtMessage.Text.Trim().Length == 0) txtMessage.Text = "Bạn hãy nhập vào thôngđiệp ...";
                                else
                                {
                                        Byte[] sendBytes =
                  Encoding.Unicode.GetBytes(txtMessage.Text); udpClient.Send(sendBytes, sendBytes.Length);
                  IPEndPoint remote = new
                  IPEndPoint(IPAddress.Parse(txtIP.Text), int.Parse(txtPort.Text));
                   Byte[] receivedBytes = udpClient.Receive(ref remote);
                  string message = Encoding.Unicode.GetString(receivedBytes);
                  Show("Nhận thông điệp từ -> " + remote.Address.ToString() + " : " + message);
                   txtMessage.Text = "";
                                }
                        }
                }


                private void txtMessage_KeyPress(object sender,KeyPressEventArgse)
                {
                        if (e.KeyChar == (char)13)
                                btnSend_Click(sender, e);
                }


                private void lstReceive_SelectedIndexChanged(object sender,
                 EventArgs e)
                {
                        if (lstReceive.SelectedItem != null)
                                txtResult.Text = lstReceive.SelectedItem.ToString();
                }


        }
}
```

CHƯƠNG 4

CHUONG4_bai1: TCPClientSample

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class TcpClientSample
{
    public static void Main()
    {
      byte[] data = new byte[1024];
      string input, stringData;
      TcpClient server;
      try
      {
          server = new TcpClient("127.0.0.1", 9050);
      } catch (SocketException)
      {
          Console.WriteLine("Unable to connect to server"); return;
      }
      NetworkStream ns = server.GetStream(); int recv = ns.Read(data, 0,
      data.Length);
      stringData = Encoding.ASCII.GetString(data, 0, recv); Console.WriteLine(stringData);
      while(true)
      {
          input = Console.ReadLine(); if (input == "exit")
            break; ns.Write(Encoding.ASCII.GetBytes(input), 0, input.Length); ns.Flush();

          data = new byte[1024];
          recv = ns.Read(data, 0, data.Length);
          stringData = Encoding.ASCII.GetString(data, 0, recv); Console.WriteLine(stringData);
      }
      Console.WriteLine("Disconnecting from server..."); ns.Close();
      server.Close();
    }
}
```

TCPListener_Sample(Server)

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class TcpListenerSample
{
    public static void Main()
    {
      int recv;
      byte[] data = new byte[1024];
      TcpListener newsock = new TcpListener(9050);
      newsock.Start();
      Console.WriteLine("Waiting for a client...");
      TcpClient client = newsock.AcceptTcpClient();
      NetworkStream ns = client.GetStream();
      string welcome = "Welcome to my test server";
      data = Encoding.ASCII.GetBytes(welcome);
      ns.Write(data, 0, data.Length);
      while(true)
      {
          data = new byte[1024];
          recv = ns.Read(data, 0, data.Length);
          if (recv == 0)
            break;

          Console.WriteLine(
                  Encoding.ASCII.GetString(data, 0, recv));
          ns.Write(data, 0, recv);
      }
      ns.Close();
```

```
        client.Close();
        newsock.Stop();
    }
}
```

Bai3 BinaryUDPServer

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class BinaryUdpSrvr
{
    public static void Main()
    {
        byte[] data = new byte[1024];
        IPEndPoint ipep = new IPEndPoint(IPAddress.Any, 9050);
        UdpClient newsock = new UdpClient(ipep);
        Console.WriteLine("Waiting for a client...");
        IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);
        data = newsock.Receive(ref sender);
        Console.WriteLine("Message received from {0}:", sender.ToString());
        Console.WriteLine(Encoding.ASCII.GetString(data, 0, data.Length));
        string welcome = "Welcome to my test server"; data = Encoding.ASCII.GetBytes(welcome);
        newsock.Send(data, data.Length, sender); byte[] data1 = newsock.Receive(ref sender); int test1
        = BitConverter.ToInt32(data1, 0); Console.WriteLine("test1 = {0}", test1); byte[] data2 =
        newsock.Receive(ref sender); double test2 = BitConverter.ToDouble(data2, 0);
        Console.WriteLine("test2 = {0}", test2); byte[] data3 = newsock.Receive(ref sender); int test3
        = BitConverter.ToInt32(data3, 0); Console.WriteLine("test3 = {0}", test3); byte[] data4 =
        newsock.Receive(ref sender); bool test4 = BitConverter.ToBoolean(data4, 0);
        Console.WriteLine("test4 = {0}", test4.ToString()); byte[] data5 = newsock.Receive(ref
        sender); string test5 = Encoding.ASCII.GetString(data5); Console.WriteLine("test5 = {0}",
        test5); newsock.Close();
    }
}
```

BinaryUDPClient

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class BinaryUdpClient
{
    public static void Main()
    {
        byte[] data = new byte[1024];
        string stringData;
        UdpClient server = new UdpClient("127.0.0.1", 9050); IPEndPoint sender = new
        IPEndPoint(IPAddress.Any, 0);

        string welcome = "Hello, are you there?"; data =
        Encoding.ASCII.GetBytes(welcome); server.Send(data, data.Length);
         data = new byte[1024];
        data = server.Receive(ref sender);
        Console.WriteLine("Message received from {0}:", sender.ToString()); stringData = Encoding.ASCII.GetString(data, 0,
        data.Length); Console.WriteLine(stringData);

        int test1 = 45;
        double test2 = 3.14159;
        int test3 = -1234567890;
        bool test4 = false;
        string test5 = "This is a test.";
        byte[] data1 = BitConverter.GetBytes(test1); server.Send(data1, data1.Length);
        byte[] data2 = BitConverter.GetBytes(test2); server.Send(data2, data2.Length);
        byte[] data3 = BitConverter.GetBytes(test3); server.Send(data3, data3.Length);
        byte[] data4 = BitConverter.GetBytes(test4); server.Send(data4, data4.Length);
        byte[] data5 = Encoding.ASCII.GetBytes(test5); server.Send(data5, data5.Length);
        Console.WriteLine("Stopping client"); server.Close();
    }
}
```

BinaryUDPDataTest

```
using System;
using System.Net;
using System.Text;
class BinaryDataTest
{
    public static void Main()
    {
        int test1 = 45;
        double test2 = 3.14159;
        int test3 = -1234567890;
        bool test4 = false;
        byte[] data = new byte[1024];
        string output;
        data = BitConverter.GetBytes(test1);
        output = BitConverter.ToString(data);
        Console.WriteLine("test1 = {0}, string = {1}", test1, output);
        data = BitConverter.GetBytes(test2);
        output = BitConverter.ToString(data);
        Console.WriteLine("test2 = {0}, string = {1}", test2, output); data = BitConverter.GetBytes(test3);
        output = BitConverter.ToString(data);
        Console.WriteLine("test3 = {0}, string = {1}", test3, output); data = BitConverter.GetBytes(test4);
        output = BitConverter.ToString(data);

        Console.WriteLine("test4 = {0}, string = {1}", test4, output);
    }
}
```

Bai   BinaryNetworkByteOrder

```
using System;
using System.Net;
using System.Text;
class BinaryNetworkByteOrder
{

    public static void Main()
    {
        short test1 = 45;
        int test2 = 314159;
        long test3 = -123456789033452;
        byte[] data = new byte[1024];
        string output;
        data = BitConverter.GetBytes(test1);
        output = BitConverter.ToString(data);
        Console.WriteLine("test1 = {0}, string = {1}", test1, output);
        data = BitConverter.GetBytes(test2);
        output = BitConverter.ToString(data);
        Console.WriteLine("test2 = {0}, string = {1}", test2, output); data = BitConverter.GetBytes(test3); output =
        BitConverter.ToString(data);
        Console.WriteLine("test3 = {0}, string = {1}", test3, output); short test1b =
        IPAddress.HostToNetworkOrder(test1);
        data = BitConverter.GetBytes(test1b); output = BitConverter.ToString(data); Console.WriteLine("test1 = {0},
        nbo = {1}", test1b, output); int test2b = IPAddress.HostToNetworkOrder(test2);
        data = BitConverter.GetBytes(test2b); output = BitConverter.ToString(data); Console.WriteLine("test2 = {0},
        nbo = {1}", test2b, output); long test3b = IPAddress.HostToNetworkOrder(test3);
        data = BitConverter.GetBytes(test3b); output = BitConverter.ToString(data); Console.WriteLine("test3 = {0},
        nbo = {1}", test3b, output);
    }
}
```

NetworkOrderClient

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class NetworkOrderClient
{
    public static void Main()
    {
        byte[] data = new byte[1024];
        string stringData;
        TcpClient server;
        try
        {
            server = new TcpClient("127.0.0.1", 9050);
        } catch (SocketException)
        {
            Console.WriteLine("Unable to connect to server"); return;
        }
        NetworkStream ns = server.GetStream(); int recv = ns.Read(data, 0,
        data.Length);
        stringData = Encoding.ASCII.GetString(data, 0, recv); Console.WriteLine(stringData);
        short test1 = 45; int test2 =
        314159;
        long test3 = -123456789033452;
        short test1b = IPAddress.HostToNetworkOrder(test1); data = BitConverter.GetBytes(test1b);
        Console.WriteLine("sending test1 = {0}", test1); ns.Write(data, 0, data.Length);
        ns.Flush();
        int test2b = IPAddress.HostToNetworkOrder(test2); data = BitConverter.GetBytes(test2b);
        Console.WriteLine("sending test2 = {0}", test2); ns.Write(data, 0, data.Length);
        ns.Flush();
        long test3b = IPAddress.HostToNetworkOrder(test3); data = BitConverter.GetBytes(test3b);
        Console.WriteLine("sending test3 = {0}", test3); ns.Write(data, 0, data.Length);
        ns.Flush();
        ns.Close();
        server.Close();
    }
}
```

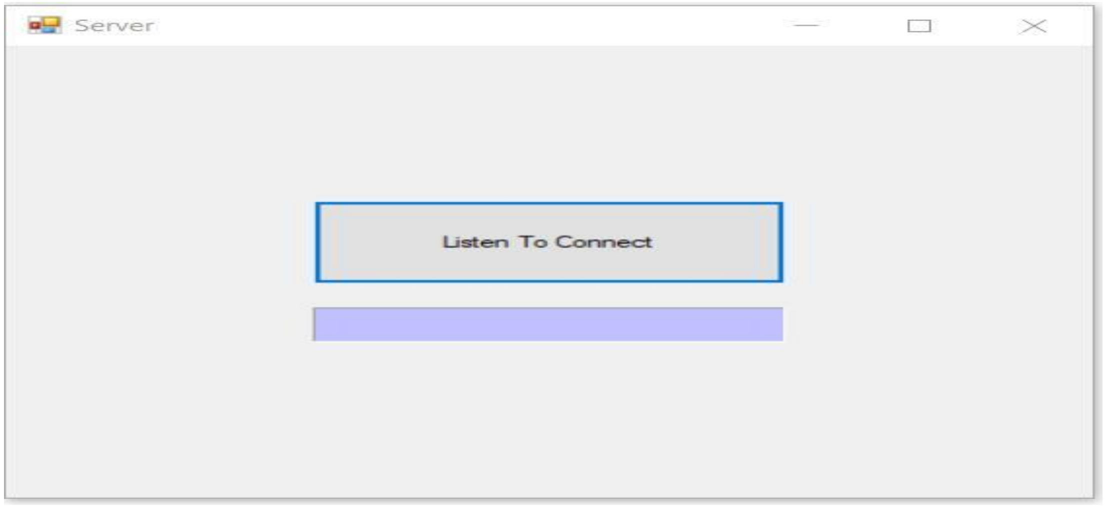
NetworkOrderSrvr

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class NetworkOrderSrvr
{
    public static void Main()
    {
        int recv;
        byte[] data = new byte[1024];
        TcpListener server = new TcpListener(9050);
        server.Start();
        Console.WriteLine("waiting for a client...");
        TcpClient client = server.AcceptTcpClient();
        NetworkStream ns = client.GetStream();
        string welcome = "Welcome to my test server";
        data = Encoding.ASCII.GetBytes(welcome);
        ns.Write(data, 0, data.Length);
        ns.Flush();
        data = new byte[2];
        recv = ns.Read(data, 0, data.Length);

        short test1t = BitConverter.ToInt16(data, 0);
        short test1 = IPAddress.NetworkToHostOrder(test1t); Console.WriteLine("received test1 =
        {0}", test1); data = new byte[4];
        recv = ns.Read(data, 0, data.Length);
        int test2t = BitConverter.ToInt32(data, 0);
        int test2 = IPAddress.NetworkToHostOrder(test2t); Console.WriteLine("received test2 =
        {0}", test2); data = new byte[8];
        recv = ns.Read(data, 0, data.Length);
        long test3t = BitConverter.ToInt64(data, 0);
        long test3 = IPAddress.NetworkToHostOrder(test3t);
        Console.WriteLine("received test3 = {0}", test3);
        ns.Close();
        client.Close();
        server.Stop();
    }
}
```

Control_Server

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using System.IO;
namespace ControlSrv
{
        public partial class Form1 : Form
        {
                const int PORT = 9050;
                const int BUFF = 10000;
                TcpClient client;
                TcpListener listener;
                Thread listenThread;
                byte[] readbuff = new byte[BUFF];
                public Form1()
                {
                        InitializeComponent();
                }

                void SendData(string data)
                {
                        lock (client.GetStream())
                        {
                                StreamWriter sw = new StreamWriter(client.GetStream()); sw.Write(data + (char)13 +
                                (char)10); sw.Flush();
                        }
                }
                void ProcessList(string Flag)
                {
                        string list = "";
                        System.Diagnostics.Process[] pr;
                        pr = System.Diagnostics.Process.GetProcesses(); foreach
                        (System.Diagnostics.Process p in pr) {
                                if (p.MainWindowTitle.Length > 0)
                                        list += "                -" + p.MainWindowTitle + (char)13;
                        }
                        SendData("THONGBAO+" + Flag + "+" + list);
                }
                private void ProcessCommand(string data)
                {
                        string[] DataArr;
                        DataArr = data.Split('+');
                        switch (DataArr[0])
                        {
                                case "SHUTDOWN":
                                        {
                                                if (DataArr[1] == "YES")
                                                {
```

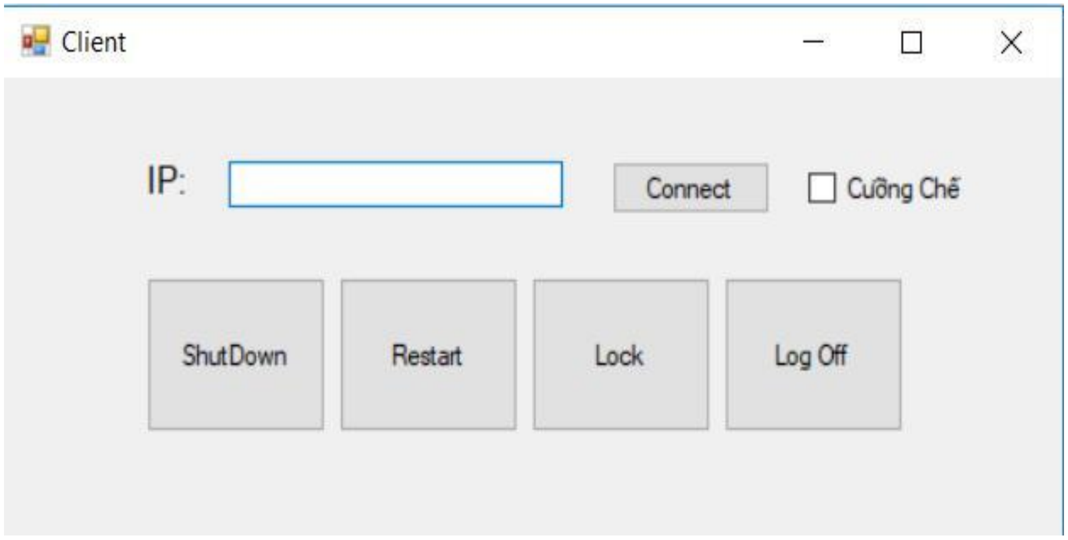
```
                                        if (DataArr[2].Trim('\0') == "OK")
                                        {
                                            System.Diagnostics.Process.Start("shutdown", "-s -f -t0");
                                            break;
                                        }
                                        ProcessList("SHUTDOWN-F");
                                        break;
                                    }
                                    else
                                    {
                                        if (DataArr[2].Trim('\0') == "OK")
                                        {
                                            System.Diagnostics.Process.Start("shutdown", "-s -t 0"); break;
                                        }
                                        ProcessList("SHUTDOWN");
                                        break;
                                    }
                                }

                            case "LOCK":
                                {
                                    if (DataArr[2].Trim('\0') == "OK")
                                    {

System.Diagnostics.Process.Start(@"C:\Windows\System32\rundll32.exe", "user32.dll,LockWorkStation");
                                        break;
                                    }
                                    ProcessList("LOCK");
                                    break;
                                }
                        }
                    }
                    void DoRead(IAsyncResult ar)
                    {
                        int byteRead;
                        string message;
                        try
                        {
                            lock (client.GetStream())
                            {
                                byteRead = client.GetStream().EndRead(ar);
                            }
                            message = Encoding.ASCII.GetString(readbuff, 0, byteRead - 1); ProcessCommand(message);
                            lock (client.GetStream())
                            {
                                client.GetStream().BeginRead(readbuff, 0, BUFF,
                                 new AsyncCallback(DoRead),null);
                            }
                        }
                        catch (Exception e)
                        {
                        }
                }
                void DoListen()
                {
                        listener = new TcpListener(IPAddress.Any, PORT); listener.Start();
                        client = listener.AcceptTcpClient();
                        this.Invoke((MethodInvoker)delegate()
                        {
                                lbStatus.Text = "Client Connected!";
                        });
                        client.GetStream().BeginRead(readbuff, 0, BUFF, new AsyncCallback(DoRead),
null);
                }

                private void btListen_Click(object sender, EventArgs e)
                {
                        listenThread = new Thread(DoListen);
                        listenThread.Start();
                        lbStatus.Text = "Waiting for client to connect!"; btListen.Enabled = false;
                }
        }
}
```



Control_Client

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

using System.Net;
using System.Net.Sockets;
using System.IO;
namespace ControlClient
{
        public partial class Form1 : Form
        {
                const int PORT = 9050;
                const int BUFF = 10000;
                TcpClient client;
                byte[] readbuff = new byte[BUFF];

                public Form1()
                {
                        InitializeComponent();
                }

                void SendData(string data)
                {
                        lock (client.GetStream())
                        {
                                StreamWriter sw = new StreamWriter(client.GetStream());
                                sw.Write(data + (char)13);
                                sw.Flush();
                        }
                }
        private void ProcessCommand(string message)
        {
            string[] DataArr;
            DataArr = message.Split('+');
            switch (DataArr[0])
            {
                case "THONGBAO":
                    {
                        if (MessageBox.Show("Programs : \r" + DataArr[2] + "is running on server. Continue ?", "Warning",
    MessageBoxButtons.YesNo, MessageBoxIcon.Question) != DialogResult.No)
                        {
                            if (DataArr[1] == "SHUTDOWN-F")
                                SendData("SHUTDOWN+YES+OK");
                            if (DataArr[1] == "SHUTDOWN")
                                SendData("SHUTDOWN+NO+OK");
                            if (DataArr[1] == "RESTART-F")
                                SendData("RESTART+YES+OK");
                            if (DataArr[1] == "RESTART")
                                SendData("RESTART+NO+OK");
                            if (DataArr[1] == "LOCK")
                                SendData("LOCK+NO+OK");
                            if (DataArr[1] == "LOGOFF")
                                SendData("LOGOFF+NO+OK");
                        }
                        break;
                    }
            }
        }
        void DoRead(IAsyncResult ar)

        {
            int byteRead;
            string message;
            try
            {
                byteRead = client.GetStream().EndRead(ar);
                if (byteRead < 1)
                {
                    return;
                }
                message = Encoding.ASCII.GetString(readbuff, 0, byteRead - 2);
                ProcessCommand(message);
                client.GetStream().BeginRead(readbuff, 0, BUFF,
                 new AsyncCallback(DoRead), null);
            }
            catch (Exception e)
            {

            }
        }
        private void btConnect_Click(object sender, EventArgs e)
        {
            if (txtIP.Text == "")
            {
                MessageBox.Show("Input IP Address Please");
                return;
            }
            try
            {
                client = new TcpClient(txtIP.Text, PORT);
                client.GetStream().BeginRead(readbuff, 0, BUFF,
                new AsyncCallback(DoRead), null);
                MessageBox.Show("Sucessful!");
                btConnect.Enabled = false;
            }
            catch
            {
                MessageBox.Show("Can not connect to server!"); this.Dispose();
            }
        }

        private void btLock_Click(object sender, EventArgs e)
        {
            if (client == null)
            {
                MessageBox.Show("First, connect to server!");
            }
            else
                SendData("LOCK+YES+");
        }
```

```
private void btLogoff_Click(object sender, EventArgs e)
{
        if (client == null)
        {
                MessageBox.Show("First, connect to server !");
        }
        else
                SendData("LOGOFF+YES+");
}

private void btRestart_Click(object sender, EventArgs e)
{
        if (client == null)
        {
                MessageBox.Show("First, connect to server");
            }
            else
            {
                if (checkBox1.Checked == true)
                {
                    SendData("RESTART+YES+");
                }
                else
                {
                    SendData("RESTART+NO+");
                }
            }
        }

        private void btShutdown_Click(object sender, EventArgs e)
        {
            if (client == null)
            {
                MessageBox.Show("First, connect to server!");
            }
            else
            {
                if (checkBox1.Checked == true)
                {
                    SendData("SHUTDOWN+YES+");
                }
                else
                {
                    SendData("SHUTDOWN+NO+");
                }
            }
        }

    }
```

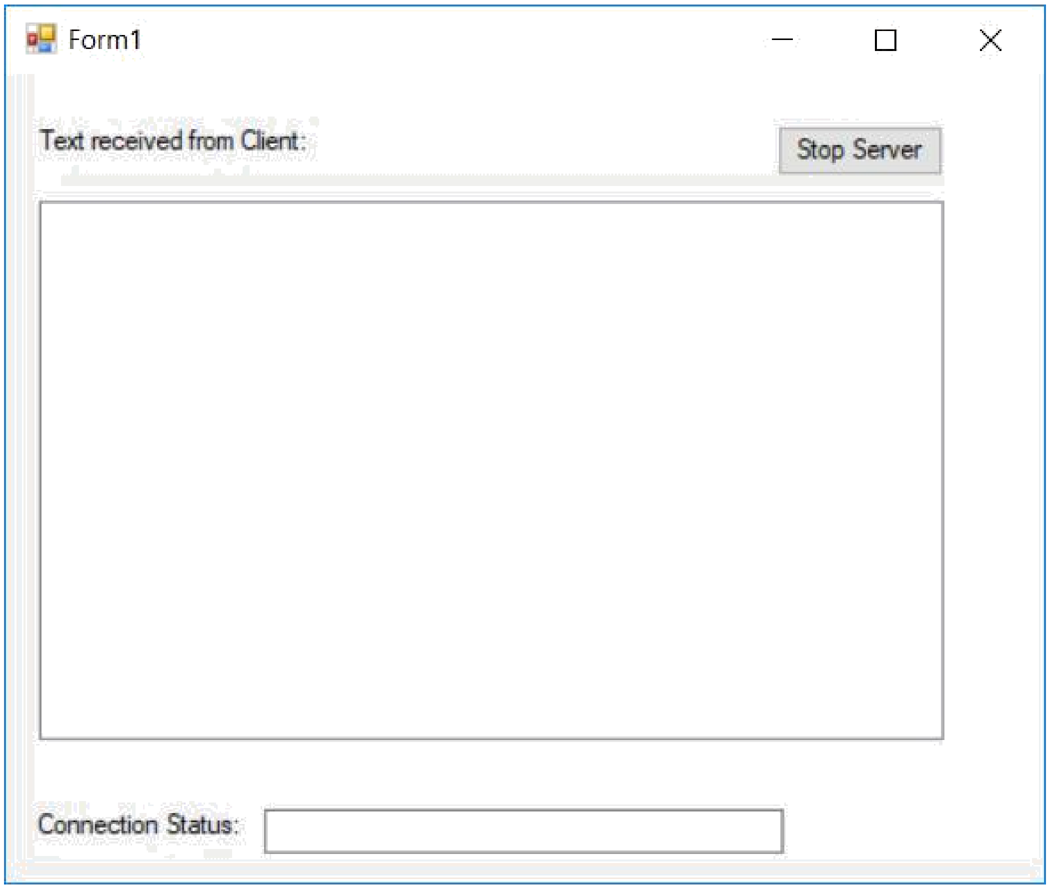


CHƯƠNG 5: TCP_Async

Server

```
using System;

using System.Net;

using System.Net.Sockets;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;


namespace AcSyncTcpSrv

{

    public partial class Form1 : Form

    {


        private byte[] data = new byte[1024];

        private int size = 1024;

        private Socket server;


        public Form1()

        {

            InitializeComponent();

            server = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
          ProtocolType.Tcp);
                                                IPEndPoint iep = new IPEndPoint(IPAddress.Any, 9050);
    server.Bind(iep);

    server.Listen(5);

    server.BeginAccept(new AsyncCallback(AcceptConn), server);


}


private void btnStop_Click(object sender, EventArgs e)

{

    Close();

}


void AcceptConn(IAsyncResult iar)

{

    Socket oldserver = (Socket)iar.AsyncState;

    Socket client = oldserver.EndAccept(iar);

    conStatus.Text = "Connected to: " + client.RemoteEndPoint.ToString(); string stringData = "Welcome to my
    server";

    byte[] message1 = Encoding.ASCII.GetBytes(stringData); client.BeginSend(message1, 0, message1.Length,
    SocketFlags.None,
          new AsyncCallback(SendData), client);

}

void SendData(IAsyncResult iar)

{

    Socket client = (Socket)iar.AsyncState; int sent = client.EndSend(iar);
    client.BeginReceive(data, 0, size, SocketFlags.None,
          new AsyncCallback(ReceiveData), client);
```

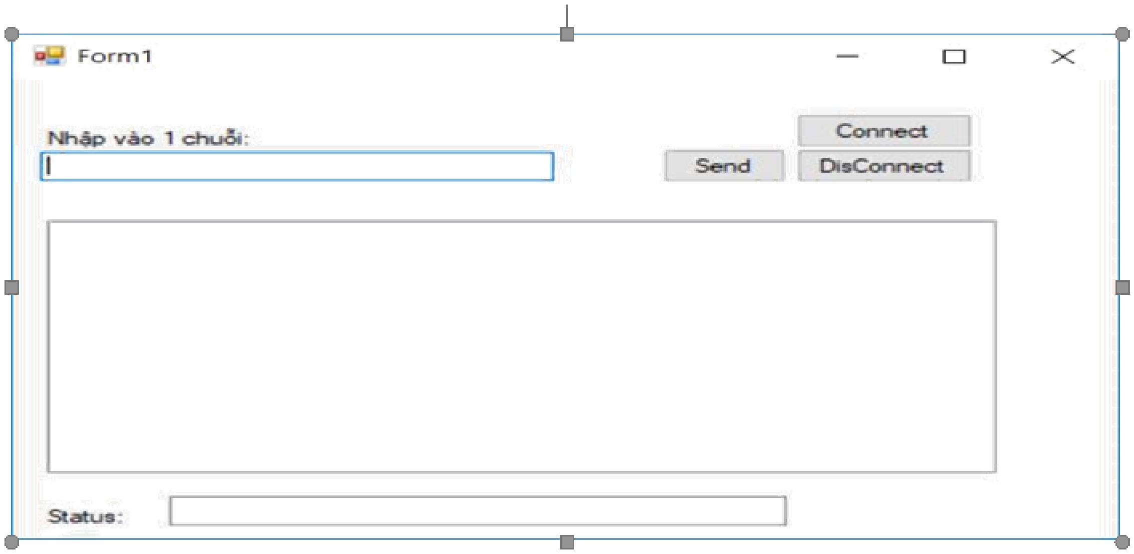
```
        }
        void ReceiveData(IAsyncResult iar)
        {
           Socket client = (Socket)iar.AsyncState;
           int recv = client.EndReceive(iar);
           if (recv == 0)
           {
              client.Close();
              conStatus.Text = "Waiting for client..."; server.BeginAccept(new AsyncCallback(AcceptConn),
              server); return;
           }
           string receivedData = Encoding.ASCII.GetString(data, 0, recv); results.Items.Add(receivedData);
           byte[] message2 = Encoding.ASCII.GetBytes(receivedData); client.BeginSend(message2, 0, message2.Length,
           SocketFlags.None,
                new AsyncCallback(SendData), client);
        }
    }
}
```



CLIENT

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Net.Sockets;
using System.Net;
namespace AcsyncTcpClients
{
    public partial class Form1 : Form
    {
        private Socket client;
        private byte[] data = new byte[1024];
        private int size = 1024;
        public Form1()
        {
            InitializeComponent();
        }
        private void btnConnect_Click(object sender, EventArgs e)
        {
            conStatus.Text = "Connecting...";
            Socket newsock = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
                         ProtocolType.Tcp);
            IPEndPoint iep = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 9050); newsock.BeginConnect(iep, new
            AsyncCallback(Connected), newsock);
        }
        private void btnSend_Click(object sender, EventArgs e)
```

```
{
    byte[] message = Encoding.ASCII.GetBytes(newText.Text); newText.Clear();
    client.BeginSend(message, 0, message.Length, SocketFlags.None, new AsyncCallback(SendData), client);
}
private void btnDisconnect_Click(object sender, EventArgs e)
{
    client.Close();
    conStatus.Text = "Disconnected";


}
void Connected(IAsyncResult iar)
{
    client = (Socket)iar.AsyncState;
    try
    {
        client.EndConnect(iar);
        conStatus.Text = "Connected to: " + client.RemoteEndPoint.ToString(); client.BeginReceive(data, 0, size,
        SocketFlags.None,
                new AsyncCallback(ReceiveData), client);
    }
    catch (SocketException)
    {
        MessageBox.Show("Không thể kết nối đến Server"); conStatus.Text = "Error
        connecting";
    }
}
void ReceiveData(IAsyncResult iar)
{
    Socket remote = (Socket)iar.AsyncState;
    int recv = remote.EndReceive(iar);
    string stringData = Encoding.ASCII.GetString(data, 0, recv); results.Items.Add(stringData);
}
void SendData(IAsyncResult iar)
{
    Socket remote = (Socket)iar.AsyncState; int sent = remote.EndSend(iar);
    remote.BeginReceive(data, 0, size, SocketFlags.None,
            new AsyncCallback(ReceiveData), remote);
        }
    }
}
```



BAI : ThreadTCP

```
//  Bài tập 1 – Chương 5
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;
class ThreadedTcpSrvr
{
   private TcpListener client;
   public ThreadedTcpSrvr()
   {
    client = new TcpListener(9050);
    client.Start();
    Console.WriteLine("Waiting for clients...");
    while(true)
    {
      while (!client.Pending())
      {
       Thread.Sleep(1000);
      }
    ConnectionThread newconnection = new ConnectionThread(); newconnection.threadListener =
    this.client;
    Thread newthread = new Thread(new
          ThreadStart(newconnection.HandleConnection));
    newthread.Start();
  }
}
```

```
   public static void Main()
   {
    ThreadedTcpSrvr server = new ThreadedTcpSrvr();
   }
}
class ConnectionThread
{
   public TcpListener threadListener;
   private static int connections = 0;
   public void HandleConnection()
   {
    int recv;
    byte[] data = new byte[1024];
    TcpClient client = threadListener.AcceptTcpClient(); NetworkStream ns =
    client.GetStream(); connections++;
    Console.WriteLine("New client accepted: {0} active connections", connections);
    string welcome = "Welcome to my test server"; data =
    Encoding.ASCII.GetBytes(welcome); ns.Write(data, 0, data.Length);
    while(true)
    {
      data = new byte[1024];
      recv = ns.Read(data, 0, data.Length);
      if (recv == 0)
       break;
      ns.Write(data, 0, recv);
    }
    ns.Close();
    client.Close();
    connections--;
    Console.WriteLine("Client disconnected: {0} active connections",
                connections);
  }
}
```

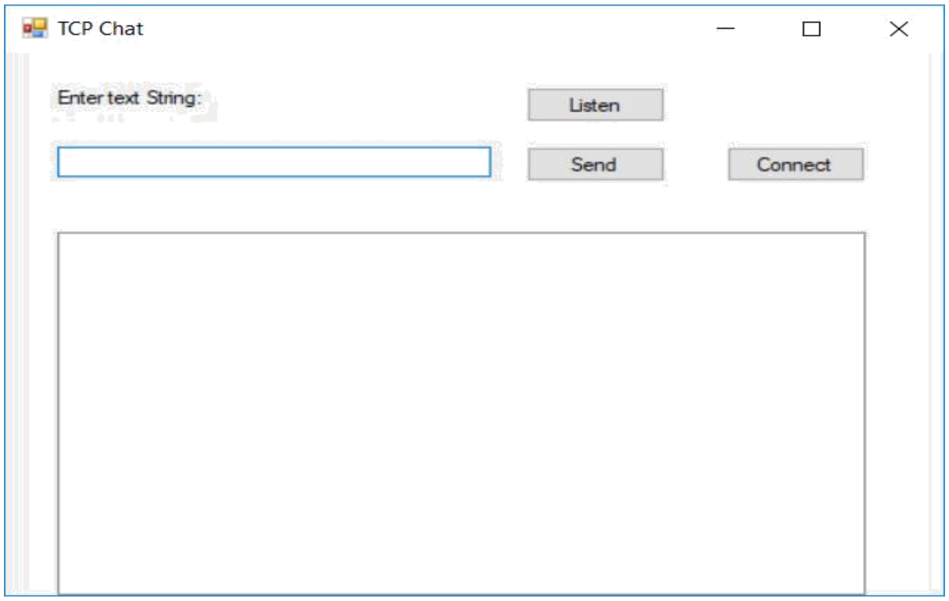
Client:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.Threading;
namespace _9._7TCPChat
{
     public partial class TcpChat : Form
     {
```

```
private static Socket client;

private static byte[] data = new byte[1024];

public TcpChat()

{
      InitializeComponent();

}

private void ButtonConnect_Click(object sender, EventArgs e)

{

      results.Items.Add("Connecting...");

      client = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

      IPEndPoint iep = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 9050); client.BeginConnect(iep, new

      AsyncCallback(Connected), client);

}

private void ButtonListen_Click(object sender, EventArgs e)

{

      results.Items.Add("Listening for a client...");

      Socket newsock = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

      IPEndPoint iep = new IPEndPoint(IPAddress.Any, 9050);

      newsock.Bind(iep);

      newsock.Listen(5);
```

```
    newsock.BeginAccept(new AsyncCallback(AcceptConn), newsock);
}

private void ButtonSend_Click(object sender, EventArgs e)

{

    byte[] message = Encoding.ASCII.GetBytes(newText.Text); newText.Clear();

    client.BeginSend(message, 0, message.Length, 0, new
     AsyncCallback(SendData), client);


}

void SendData(IAsyncResult iar)

{

    Socket remote = (Socket)iar.AsyncState;

    int sent = remote.EndSend(iar);

}

void AcceptConn(IAsyncResult iar)

{

    Socket oldserver = (Socket)iar.AsyncState;

    client = oldserver.EndAccept(iar);

    results.Items.Add("Connection from: " + client.RemoteEndPoint.ToString()); Thread receiver = new Thread(new
    ThreadStart(ReceiveData)); receiver.Start();
}

void Connected(IAsyncResult iar)

{

    try

    {

        client.EndConnect(iar);

        results.Items.Add("Connected to: " + client.RemoteEndPoint.ToString()); Thread receiver = new Thread(new
        ThreadStart(ReceiveData)); receiver.Start();
    }

    catch (SocketException)

    {

        results.Items.Add("Error connecting");

    }

}

void ReceiveData()

{

    int recv;

    string stringData;
```

```
            while (true)

            {

                recv = client.Receive(data);

                stringData = Encoding.ASCII.GetString(data, 0, recv); if (stringData == "bye")
                    break;

                results.Items.Add(stringData);

            }

            stringData = "bye";

            byte[] message = Encoding.ASCII.GetBytes(stringData);

            client.Send(message);

            client.Close();

            results.Items.Add("Connection stopped");

            return;

        }

        private void TCPChat_KeyPress(object sender, KeyPressEventArgs e)

        {

            if (e.KeyChar == 13)

                ButtonSend_Click(sender, e);

        }

    }

}
```



BAI: NETWORK_STREAM

```
using System;
using System.Collections.Generic; using System.Linq; using
System.Text;
using System.IO;
using System.Net;
using System.Net.Sockets;


class StreamTcpSrvr
{
        public static void Main()
        {
                string data;
                IPEndPoint ipep = new IPEndPoint(IPAddress.Any, 9050); Socket newsock = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
                newsock.Bind(ipep);
                newsock.Listen(10);
                Console.WriteLine("Waiting for a client...");
                Socket client = newsock.Accept();
                IPEndPoint newclient = (IPEndPoint)client.RemoteEndPoint; Console.WriteLine("Connected with {0} at port {1}",
newclient.Address, newclient.Port);
                NetworkStream ns = new NetworkStream(client);
                StreamReader sr = new StreamReader(ns);
                StreamWriter sw = new StreamWriter(ns);
                string welcome = "Welcome to my test server";
                sw.WriteLine(welcome);
                sw.Flush();
                while (true)
                {
                        try
                        {
                                data = sr.ReadLine();
                        }
                        catch (IOException)
                        {
                                break;
                        }


                        Console.WriteLine(data);
                        sw.WriteLine(data);
                        sw.Flush();
                }
                Console.WriteLine("Disconnected from {0}", newclient.Address);
                sw.Close();
                sr.Close();
                ns.Close();
        }
}
```

Client

```
using System;
using System.Collections.Generic; using System.Linq; using
System.Text;
using System.IO;
using System.Net;
using System.Net.Sockets;


class StreamTcpClient
{
            public static void Main()
            {
                    string data;
                    string input;
                    IPEndPoint ipep = new
IPEndPoint(IPAddress.Parse("127.0.0.1"), 9050);
                    Socket server = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
                    try
                    {
                            server.Connect(ipep);
                    }
                    catch (SocketException e)
                    {
                            Console.WriteLine("Unable to connect to server."); Console.WriteLine(e.ToString());
                            return;
                    }
                    NetworkStream ns = new NetworkStream(server); StreamReader sr = new StreamReader(ns);
                    StreamWriter sw = new StreamWriter(ns); data = sr.ReadLine();

                    Console.WriteLine(data);
                    while (true)
                    {
                            input = Console.ReadLine();
                            if (input == "exit")
                                    break;
                            sw.WriteLine(input);
                            sw.Flush();
                            data = sr.ReadLine();
                            Console.WriteLine(data);
                    }
                    Console.WriteLine("Disconnecting from server...");
                    sr.Close();
                    sw.Close();
                    ns.Close();
                    server.Shutdown(SocketShutdown.Both); server.Close();
            }
}
```

BAI: ServerFile (Không có form)

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


namespace Server
{
        public partial class WindowServer : Form
        {
                private byte[] data = new byte[1024*1024];
                private int size = 1024*1024;
                 private Socket server;
                private int count=0;


                public WindowServer()
                {
                        InitializeComponent();
                }
                private void WindowServer_Load(object sender,  EventArgs e)
                {


                }
                private void cbChoose_CheckedChanged(object sender, EventArgs e)
                {
                        if (cbChoose.CheckState == CheckState.Checked)
                        {
                                txtIP.Enabled = false;
                                txtPort.Enabled = false;
                        }
                        else

                        {
                                txtIP.Enabled = true;
                                txtPort.Enabled = true;
                        }
                }


                //Sự kiện nhấn nút Start Listening
                private void btConn_Click(object sender, EventArgs e)
                {
                    server = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
                    if (cbChoose.Checked)
                                {
                                        IPEndPoint iep = new IPEndPoint(IPAddress.Any, 9050); server.Bind(iep);
                                }
                                else
                                {
                                    if (txtIP.Text == "" || txtPort.Text == "")
                                        MessageBox.Show("Input your address first!");
                                        else
                                        {
                                                IPAddress ip = IPAddress.Parse(txtIP.Text);
                                                IPEndPoint iep = new IPEndPoint(ip, Convert.ToInt32(txtPort.Text));
                                                erver.Bind(iep);
                                        }
                                }
                                txtStatus.Text = "Waiting for client..."; server.Listen(5);
                                server.BeginAccept(new AsyncCallback(AcceptConn), server); //Bắt đầu việc chấp nhận kết nối từ client
                }
                void AcceptConn(IAsyncResult iar)
                {
                        lbCount.Text = "";
                        Socket oldserver = (Socket)iar.AsyncState;

                        Socket client = oldserver.EndAccept(iar); //Kết thúc việc kết nối
                        count++;
                        server.BeginAccept(new AsyncCallback(AcceptConn), oldserver);
                        //Chấp nhận kết nối nếu có thêm client yêu cầu
                        lbCount.Text = Convert.ToString(count); txtStatus.Text = "Server is connecting to client...";
                        string stringData = "Welcome to my server";
                        byte[] message1 = Encoding.ASCII.GetBytes(stringData);
                        client.BeginSend(message1, 0, message1.Length, SocketFlags.None,
                        new AsyncCallback(SendData), client); //Gửi thông điệp chào mừng tới client
                }
                //Hàm gửi dữ liệu
                void SendData(IAsyncResult iar)
                {
                        Socket client = (Socket)iar.AsyncState;
                        int sent = client.EndSend(iar); //Kết thúc việc gửi dữ liệu
                        client.BeginReceive(data, 0, size, SocketFlags.None, new AsyncCallback(ReceiveData), client);
                        //Bắt đầu nhận dữ liệu từ socket
                }
                //Hàm nhận dữ liệu từ client gửi đến
                void ReceiveData(IAsyncResult iar)
                 {
                      Socket client = (Socket)iar.AsyncState;
                    int recv = client.EndReceive(iar); //Kết thúc nhận dữ liệu
                              if (recv == 0)
                              {
                                      client.Close();
                                        count--;
                                        if (count <= 0)
                                        {
                                                    txtStatus.Text = "Waiting for client...";
                                        }

                                      lbCount.Text= Convert.ToString(count);
                                       server.BeginAccept(new AsyncCallback(AcceptConn), server); //Chấp nhận kết nối
                                      mới đến server
                                      return;
                              }
                              string receivedData = (Encoding.ASCII.GetString(data, 0, recv));
                              string recvData = receivedData.Replace(" ", "");
                              if (File.Exists(recvData) && recvData != "")
                              {
                                      StreamReader SRD = new StreamReader(recvData);
                                       string mess = SRD.ReadToEnd(); //Đọc tất cả nội dung trong file mà client yêu cầu
                                       byte[] message2 =
                                      Encoding.ASCII.GetBytes(mess);
                                      client.BeginSend(message2,
                                      0,message2.Length,
                                      SocketFlags.None,new
                                      AsyncCallback(SendData), client);
                                      //Bắt đầu việc gửi nội dung văn bản
                                      sang client
                              }
                              else
                              {
                                      string mess = "Your path was wrong!\r\n Please re-send your path.";
                                      byte[] message2 =
                                        Encoding.ASCII.GetBytes(mess);
                                      client.BeginSend(message2, 0, message2.Length, SocketFlags.None,
                                                    new AsyncCallback(SendData), client);
                                        //Gửi thông điệp yêu cầu client nhập lại đường dẫn
                              }
                }
                private void btClose_Click(object sender, EventArgs e)
                {
                Close();
                }

            }
        }
```

CLIENT

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


namespace Client
{
       public partial class WindowClient : Form
       {
              private Socket client;
              private const int size =1024*1024; private byte[] data = new
              byte[size]; public WindowClient() {

                     InitializeComponent();
              }


              private void btConnect_Click(object sender, EventArgs e)
              {
                     if (client == null)
                     {
                            Socket newsock = new
                           Socket(AddressFamily.InterNetwor,
                           SocketType.Stream, ProtocolType.Tcp);
                  if (txtIP.Text == "" || txtPort.Text == "")
                                 MessageBox.Show("Input your address first!");
                 else
                 {
                           IPEndPoint iep = new
                           IPEndPoint(IPAddress.Parse(txtIP.Text)
                           , Convert.ToInt32(txtPort.Text));
                           newsock.BeginConnect(iep, new
                            AsyncCallback(Connected), newsock);
                            //Bắt đầu việc kết nối từ server

                            }
                     }
                     else
                     {
                            MessageBox.Show("You are on connection");
                     }
              }
              //Hàm kết nối client với server
              void Connected(IAsyncResult iar)
              {
                     try
                     {
                            client = (Socket)iar.AsyncState; client.EndConnect(iar);
                             //Kết thúc việc kết nối
                             txtStatus.Text = "Connected to: " +
        client.RemoteEndPoint.ToString(); client.BeginReceive(data, 0, size,
                            SocketFlags.None,
                 new AsyncCallback(ReceiveData), client); //Bắt đầu nhận dữ liệu từ socket
                     }
                     catch (SocketException se)
                     {
                            string str;
                            str = "\nConnection failed, is the server running?\n" +
                            se.Message;
                            MessageBox.Show(str);
                     }
```

```
            }
            //Hàm nhận dữ liệu được gửi qua từ server
            void ReceiveData(IAsyncResult iar) {
                  Socket remote = (Socket)iar.AsyncState;
                  int recv = remote.EndReceive(iar);
                   //Kết thúc việc nhận dữ liệu
        string stringData = Encoding.ASCII.GetString(data, 0,recv);
                  txtShow.Text = stringData;
                   //Hiển thị dữ liệu nhận  được
            }
            private void btDisconn_Click(object sender, EventArgs e)
            {


                  if (client != null)
                  {
                        client.Close();
                        client = null;
                                  txtStatus.Text = "no connection";
                  }
                  else
                  {
                        string noti = "Connect first!";
                        MessageBox.Show(noti); //Thông báo khi client
được kết nối
                  }
            }

            private void btSendStr_Click(object sender, EventArgs e)
            {
                  if (client != null)
                  {
                        if (txtDir.Text != "")
                        {
                              byte[] message =
                              Encoding.ASCII.GetBytes(txtDir.Text);
                              txtDir.Clear();
                              client.BeginSend(message, 0, message.Length,SocketFlags.None,
                              new AsyncCallback(SendData), client);
                               //Bắt đầu gửi dữ liệu từ socket
                        }
                        else
                        {
                              string noti = "Input your path first."; MessageBox.Show(noti);
                        }
                  }
                  else
                  {
                        string noti = "Connect first!";
                        MessageBox.Show(noti);
```

```
                }
            }


            //Hàm gửi dữ liệu đi
            void SendData(IAsyncResult iar)
            {
                try
                {
                    txtShow.Clear();
                    Socket remote = (Socket)iar.AsyncState;
                    int sent = remote.EndSend(iar); //EndSend()
                    remote.BeginReceive(data, 0, size,
SocketFlags.None,
                                 new AsyncCallback(ReceiveData), remote);
                                  //Bắt đầu nhận dữ liệu từ socket
                }
                catch (SocketException se)
                {
                    MessageBox.Show(se.ToString());
                }
            }


            private void btClose_Click(object sender, EventArgs e)
            {
                if (client == null)
                {
                    Close();
                }
                else
                {
                    string noti = "Disconnect first.";
                    MessageBox.Show(noti);
                }
            }


}
```