Manisa Celal Bayar University

Computer Network Programming Project

Project's Name : Cins Apartment Management System

Student Number:

190315028

190315058

Student Name and Surname :

Mehmet Can Tekin

Havva Beste Tekçeli

Contents

Abstract

We purpose in this project, providing some informations to the clients by the server. It is about weather forecast, exchange rate, and sending to information to the clients about who opened to outer gate. These informations sending an outgoing message to everyone by the server. We decided, it should communicate according to the TCP protocol. In other words, we thought that every information that needs to be received should be conveyed to the clients definitively.Each clients, server ,and card reader had to have a different interface. Thus, three different Visual Studio Projects were created. We tried to improve it as much as possible by dealing with many situations such as the disconnection of the server, the clients are not connected, or the card reader needs the server to connect and it must try to connect until the server comes. As a result, platforms were created where multi-clients can connect to the server, send and receive messages , and see the weather conditions and exchange rates from the server.In other hand, a card reader that can send information to the clients via the server.

1. Introduction

In this project, we suppose, 8 families live in CinsApartment(No:101). All families have a computer/laptop that are connected to the same ‘’IP Subnet’’. There is a single server and there is a single outer-gate of CinsApartment controlled by Card Reader which is connected to Server. All families can join/write/read the chat on the screen. Also all families can get informations who unlock the outer gate of Cins Apartment with date/time record. In the other hand, all families can get weather forecast informations and also can get currency Exchange information from the server. The server is taken these informations from the website which are ‘’weather.com” and “doviz.com” . We decided to do this project with using asynchronous sockets,because asynchronous world, the program post operations and check the result later. The client send a message but does not wait for an immediate return, the reply may arrive now or hours later. In our project, the weather forecast information and exchange rate information comes from the server immediately after the client is connected. Information is automatically refreshed in the background every 5 minutes. While these are happening, the client can send messages and other clients can see the message and the informations are updated in the same way in other clients.Additionally, when these are occurring, if someone opens the gate with a card reader, information will send to the clients who open the outer gate.

2. Cins Apartment Server

A server is a computer program or device that provides services to another computer program and its user, also known as the client.Therefore, it was necessary to write the server code first in order to provide service to the clients. When we decided to coding for the server, we started by investigating exactly what an asynchronous socket methods is. We learnt, these methods allow you to start a network function and specify a delegate to call when the network function completes. Meanwhile, the rest of the program can go happily on its way doing other things. In simple, we simply need to understood how to approach for the algorithm in 4 steps.Figure 1 shows how this will happen in more detail.

- Bind()
- Listen()
- BeginAccept()
- BeginSend()/BeginReceive()

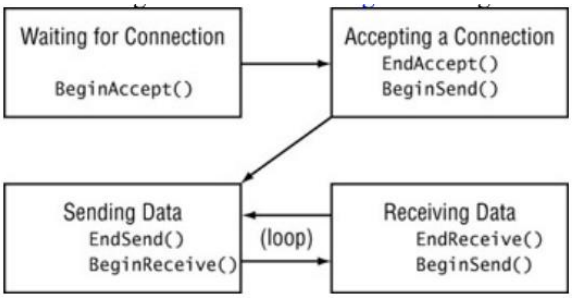


Figure 1

First of all, we needed to create a **sv_Socket** which data type is Socket . After that we specify some properties of this socket as protocol type, socket type and address family.

- *AddressFamily* to define network type
- *SocketType* to define type of data connection
- *ProtocolType* to define a specific network protocol

Then, we created an object – **server_iep** – which is belongs to IPEndPoint . It is used when binding sockets to local addresses, or when connecting sockets to remote addresses.So, we bind **sv_socket** , then listen and ready to begin accept.

```
sv_Socket = new Socket(AddressFamily.InterNetwork,
                       SocketType.Stream,
                       ProtocolType.Tcp);
IPEndPoint server_iep = new IPEndPoint(IPAddress.Any, 9050);
sv_Socket.Bind(server_iep);
sv_Socket.Listen(9);
sv_Socket.BeginAccept(new AsyncCallback(sv_AcceptCallback), sv_Socket);
```

2.1 Server Methods

2.1.1 sv_AcceptCallback()

The method is to retrieve the original server socket. This is done using the AsyncState property of the IAsyncResult class. This property passes the original object placed in the BeginAccept() object parameter. Because it is defined as a generic object, it must be typecast into a Socket object. After the original Socket object is retrieved, the EndAccept() method can obtain a new Socket object for the client connection.

```
private void sv_AcceptCallback(IAsyncResult iar)
{
    Socket org_sv_socket = (Socket)iar.AsyncState;
    Socket new_Socket = org_sv_socket.EndAccept(iar);
    try
    {
        client_socket_list.Add(new_Socket);
        sv_SendUrlInfo();
        sendFirstMessages(new_Socket);
        sendFirstEntry(new_Socket);
        new_Socket.BeginReceive(data, 0, size, SocketFlags.None,
            new AsyncCallback(sv_ReceiveCallBack), new_Socket);
        sv_Socket.BeginAccept(new AsyncCallback(sv_AcceptCallback), sv_Socket);
    }
    catch (Exception ex)
    {
        MessageBox.Show("Socket creation problem.");
        sv_Socket.BeginAccept(new AsyncCallback(sv_AcceptCallback), sv_Socket);
    }
}
```

client_socket_list is a list which data type is Socket and holds to new clients sockets into this list.

Simply, **sv_SendUrlInfo()** methods gets information about weather forecast and curremcy exchange from websites. It will be explained later.

After that, **new_socket** is ready to begin receive data with helper method which is called "sv_ReceiveCallBack()" and **sv_socket** is ready to begin accept data.

2.1.2 sv_ReceiveCallBack()

In order to receive data , the **new_socket** must first satisfy the condition of being connected. If new_Socket is connected, it takes to whole data with EndReceive() and linking to **rcv** . **rcv** holds size of the data . If rcv is equal to zero, there is no socket to send, so **new_Socket** will be shutting down and then will be closed, and socket will be deleted by **socketClear()** method. If **rcv** is not equal to zero, **text** will be encoding from byte to string. After the encoding text, **msg_Control_Method()** that decides the path where the data will go to which listbox and send them to clients to their listboxes.It will be ready to receive again with BeginReceive().

```
private void sv_ReceiveCallBack(IAsyncResult iar)
{
    Socket new_Socket = (Socket)iar.AsyncState;
    int rcv;
    if (new_Socket.Connected)
    {
        try
        {
            rcv = new_Socket.EndReceive(iar);
        }
        catch (Exception)
        {
            socketClear(new_Socket);
            return;
        }
        if (rcv != 0)
        {
            string text = Encoding.UTF32.GetString(data,0,rcv);
            msg_Control_Method(text, new_Socket);
            new_Socket.BeginReceive(data, 0, size, SocketFlags.None,
               new AsyncCallback(sv_ReceiveCallBack), new_Socket);

        }
        else if (rcv == 0)
        {
            new_Socket.Shutdown(SocketShutdown.Both);
            new_Socket.Close();
            socketClear(new_Socket);
            return;
        }
    }
}
```

2.1.3 sendsToAllClients()

This method is using for sending datas to every recorded clients in this server.

sv_SendCallBack() method is making to server can receive datas again.

```
2 başvuru
private void sendsToAllClients(string text ,  string header)
{
    text = header + text;
    byte[] messagebyte = Encoding.UTF32.GetBytes(text);
       for (int i=0; i<client_socket_list.Count;i++)
     {
       client_socket_list.ElementAt(i).BeginSend(messagebyte, 0,
       messagebyte.Length, SocketFlags.None,
       new AsyncCallback(sv_SendCallBack), client_socket_list.ElementAt(i));
    }
}
1 başvuru
private void sv_SendCallBack(IAsyncResult iar)
{

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

}
```

2.1.4 sendFirstEntry & sendFirstEntry

These methods sends datas to the clients where data holding in server. When the clients connected, they wil receive old messages from server, if server holds any data.

2.1.4 sv_SendUrlInfo()

In this method, it is aimed to send informations to each client by pulling the informations of the weather forecast and exchange rate from the internet page. We used the HtmlAgilityPack package while writing these methods. Because we pulled the information one by one via html and placed them on the nodes and assigned the values to the necessary labels.

```
private void sv_SendUrlInfo()
{
    string text= "2|"+sv_Weather()+sv_Currency();
    byte[] messagebyte = Encoding.UTF32.GetBytes(text);
    for (int i = 0; i < client_socket_list.Count; i++)
    {
        client_socket_list.ElementAt(i).BeginSend(messagebyte,
                    0, messagebyte.Length, SocketFlags.None,
                    new AsyncCallback(sv_SendCallBack),
                    client_socket_list.ElementAt(i));
    }
}
```

2.1.5 sv_SendCallBack()

This method is complete BeginSend() and ready to receive datas.

```
private void sv_SendCallBack(IAsyncResult iar)
{
    Socket client = (Socket)iar.AsyncState;
    try
    {
            int sent = client.EndSend(iar);
            client.BeginReceive(data, 0, size, SocketFlags.None,
            new AsyncCallback(sv_ReceiveCallBack), client);
    }
    catch (Exception ex)
    {
    //    MessageBox.Show(ex.Message, Application.ProductName,
    //    MessageBoxButtons.OK, MessageBoxIcon.Error);
        client.Shutdown(SocketShutdown.Both);
        client.Close();
        socketClear(client);
    }
}
```

2.1.6 StartTimer() and asyncTask()

```
private async void StartTimer()
{
    System.Threading.Timer timer = null;
    timer = new System.Threading.Timer((obj) =>
    {
        // This is the method that will be called every 5 minutes
        asyncTask().Wait();

        // Reset the timer
        timer.Change(TimeSpan.FromMinutes(5), TimeSpan.FromMilliseconds(-1));
    }, null, TimeSpan.FromMinutes(5), TimeSpan.FromMilliseconds(-1));
}
1 başvuru
private async Task asyncTask()
{
    sv_Weather();
    sv_Currency();
    sv_SendUrlInfo();
}
```

3. Cins Apartment Client

We have multiple clients for this server. The clients can join the server. After that, clients can send and read messages, see weather forecast and currency exchange. When these things are happening, it should be asynchronous.Thus, we followed the same algorithm for the client as we followed the algorithm for the server. These algorithms almost same with the server. An explanation will be given about the methods we have written, different from these algorithm.

3.1 Client Methods

3.1.1 btnConnect_Click()

It is a method used for client to connect to the server. It is also notifies the client if there is a problem in the connection. It controls socket is null or the client is already connected. After that, clientNameCheck method checks if the namespace is empty or not. If the name field is empty, an information such as "The name must be entered first" comes from **clientNameCheck()** method.

```
private void btnConnect_Click(object sender, EventArgs e)
{
    if (!isConnected || new_Socket == null)
    {
        if (clientNameCheck(textsToText(txt_cliName.Texts)))
        {
            try
            {
                new_Socket = new Socket(AddressFamily.InterNetwork,
                                        SocketType.Stream, ProtocolType.Tcp);
                IPEndPoint server_iep = new IPEndPoint
                                        (IPAddress.Parse("127.0.0.1"), 9050);
                new_Socket.BeginConnect(server_iep,
                            new AsyncCallback(cli_ConnectCallback), new_Socket);
                isConnected = true;
                online_btn.Checked= true;
            }
            catch (Exception ex)
            {
            MessageBox.Show(ex.Message,
              Application.ProductName, MessageBoxButtons.OK, MessageBoxIcon.Error);
            } } }
    else
    {
        if (new_Socket.Connected && new_Socket!=null)
        {
            MessageBox.Show("Already Connected.");
        }
        else
        {
            MessageBox.Show("Server lost.");
            isConnected = false;
            txt_cliName.Enabled = true;
        } } }
```

3.1.2 OrWhich()

It is a method that separates the usage areas of the data according to the bytes at the beginning of the incoming data. Accordingly, it also decides where the data will appear.It is aimed to make easier and more convenient interaction between server and clients with this method.

```
private void OrWhich(string text)
{
    string type = "";
    string mainText = "";
    for (int i = 0; i < 2; i++)
    {
        type = text.Substring(0, 1);
        mainText = text.Substring(2);
    }
    switch (Int32.Parse(type))
    {
        case 1:
            lstbx_Chat.Items.Add(mainText); // 1| message
            break;

        case 2:
            placeHolder(mainText);  // 2| weather/currency
            break;
        case 3:
            lstbx_OuterGate.Items.Add(mainText);  // 3| outer-gate
            break;
    }

}
```

3.1.3 btnDisconnect_Click()

It provides the client to disconnect from the server by pressing the disconnect button with this method. If the client is not connected, it send information about it to the client.

```
private void btnDisconnect_Click(object sender, EventArgs e)
{
    try
    {
        if (new_Socket!=null)
        {
            new_Socket.Shutdown(SocketShutdown.Both);
            new_Socket.Close();
            new_Socket = null;
            isConnected = false;
            online_btn.Checked = false;

        }
        else
        {
            MessageBox.Show("Already disconnected.");
        }
    }
    catch (Exception ex)
    {
    //   MessageBox.Show(ex.Message, Application.ProductName,
    //   MessageBoxButtons.OK, MessageBoxIcon.Error);
    }

}
```

3.1.4 Other Methods

Other methods are quietly similar with the Client Apartment Server methods. Although some methods show minor changes, they show same functions. The general logic of the algorithm is the same. These mention methods are **RcveCallBack , SendCallBack ,ConnectionCallBack(** in Cins Apartment Client.

4. Card Reader

Since we do not physically have the card reader used to open the outer gate, we simulated it with code. It is an application that connects to the server. After connects to the server, it sends information messages to the clients about who opened to outer gate for which apartment through the server.

If the server is not exist or not connected, the card reader tried to connect to the server every 5 seconds and sends an acknowledge as show on Figure 4.1 .

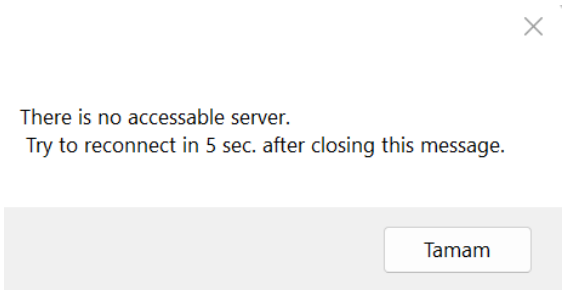
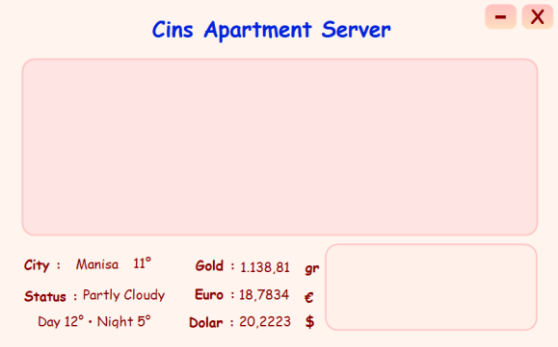


Figure 4.1

5. Testing

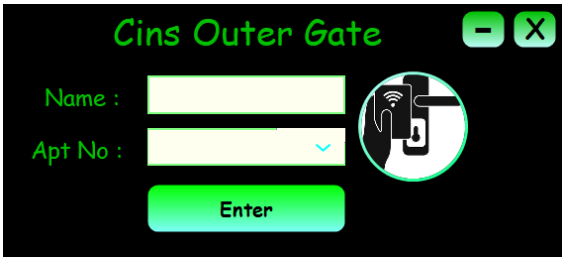
5.1 Appearence of Applications

Server



Client

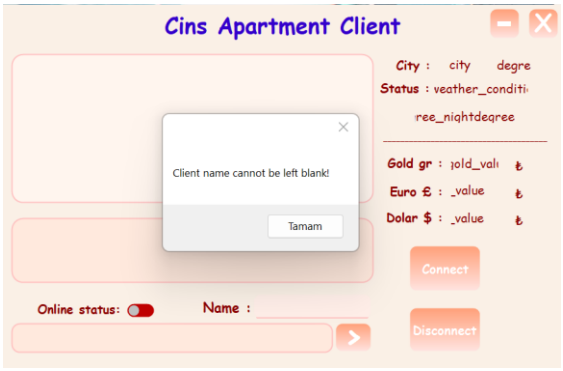
Card Reader



5.2 Client Connect to Server

If client try to connect without name, it will come acknowledge and client will not connect to server.

Output:



After writing name and click to connection button.

Output:

If the connection is successful, client's online status will turn green , and weather forecast and currency exchange informations will receive from server. It will not allow to change name after the connection.

When the server is shutting down and try to connect.

Output:

Online status will be turn red. Weather forecast and currency exchange informations will stay but it won't be updated anymore. Client can able to change the name.

After that, if the client is click to connect button **first time**, it will be acknowledge like on Figure 5.2.1 and click to connect button in **second time** it will be on acknowledge like on figure 5.2.2 .

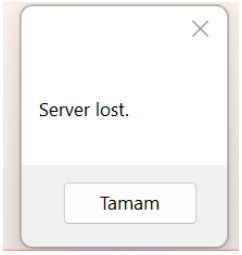


Figure 5.2.1

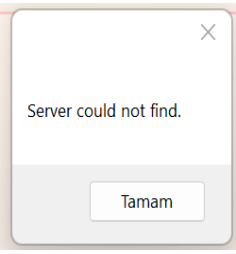
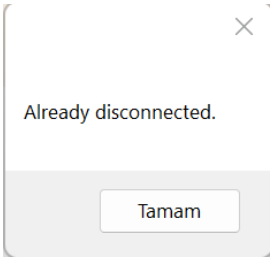


Figure 5.2.2

If client is already disconnected and click on button which is ''Disconnected'' . It will be send information.



5.3 Client Sends Message

If there are multiple clients are connected to server and send messages. All of the clients will receive the message from the server.

Output for clients:

Output for server:

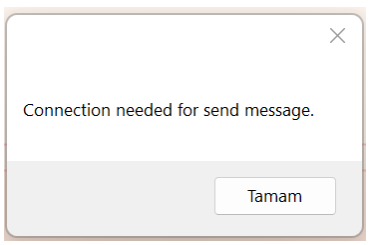
If one client is disconnected from the server. Message wont send to who is disconnected.

Output for clients:

Output for server:

Who is disconnected from server and try to send message .

Output:



However, If server didn't close and has some datas, server will send all to the clients who is connected again or when there is a new client is connected.

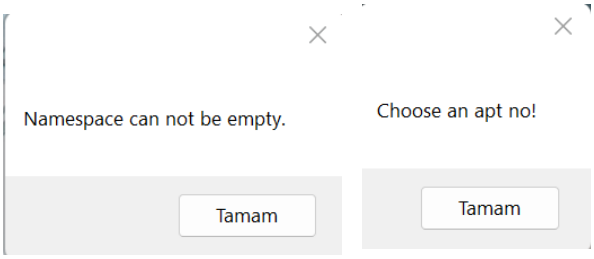
Output: For new client:

Output :After connection for new client:

5.4 Card Reader

If card reader is connected to server and when the click to enter button without writing name and selecting to apartment number.

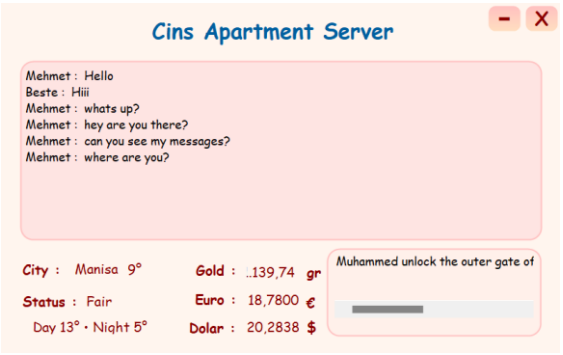
Output:



After fill in the blanks and click to enter button.

Output for clients :

Output for server:



The information of who opened to outer-gate will be in a different listbox. Information will not come to where clients can see the message in chat listbox. At the same time, this information will also be kept in a different listbox by the server.

When the server is closed.

Output:

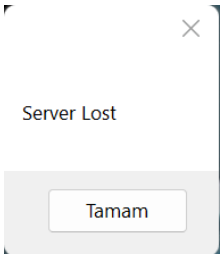


Figure 5.3.1

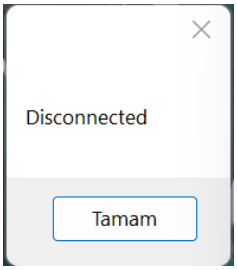


Figure 5.3.2

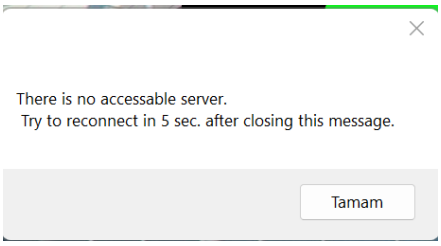
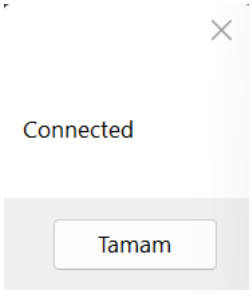


Figure 5.3.3

Firstly, It will be seen ‘’Disconnected’’ message on Figure 5.3.2 , after try to click on enter button, it will be seen " Server Lost’’ message. On Figure 5.3.3 , this message will be in loop every 5 second until server comes.

When server come again.

Output:



It will be seen ‘’ Connected’’ message and it will be connected to server again and can able to send information again.

Output for server:

6.Conclusion and Decision

While doing this project, we learned how to communicate between multi clients and servers by working over TCP protocol. We have experienced how a certain algorithm that we need to follow is developed and how we can do it asynchronously while implementing it. In this project, we learned the general working logic of sockets, how they work asynchronously, how to pull data from the internet and transfer it to other users through the server, and many other things.

When we started this project first time, we followed a lot of wrong paths and we did not know how to solve it. We named the project 'HowToFixMy' to understand how to fix it after creating a new project multiple times, recoding and doubling down on logical errors and bugs. We thought that this should be the file name of the projects in order not to forget our mistakes and what we did.Also we liked the idea of this project name.