



MUHAMMAD UMAR KHAN 10619

NP ASSIGNMENT#01

Network Programming (NW411)

Assignment # 1

Due Date: October 24, 2021

Please carefully read the following instructions before attempting the assignment.

Instruction: It should be clear that your assignment would **Not** get any credit if, the assignment is submitted **after due date** or **Copied**

Mention complete code (ss) and output (ss).

Question # 1: Implementation

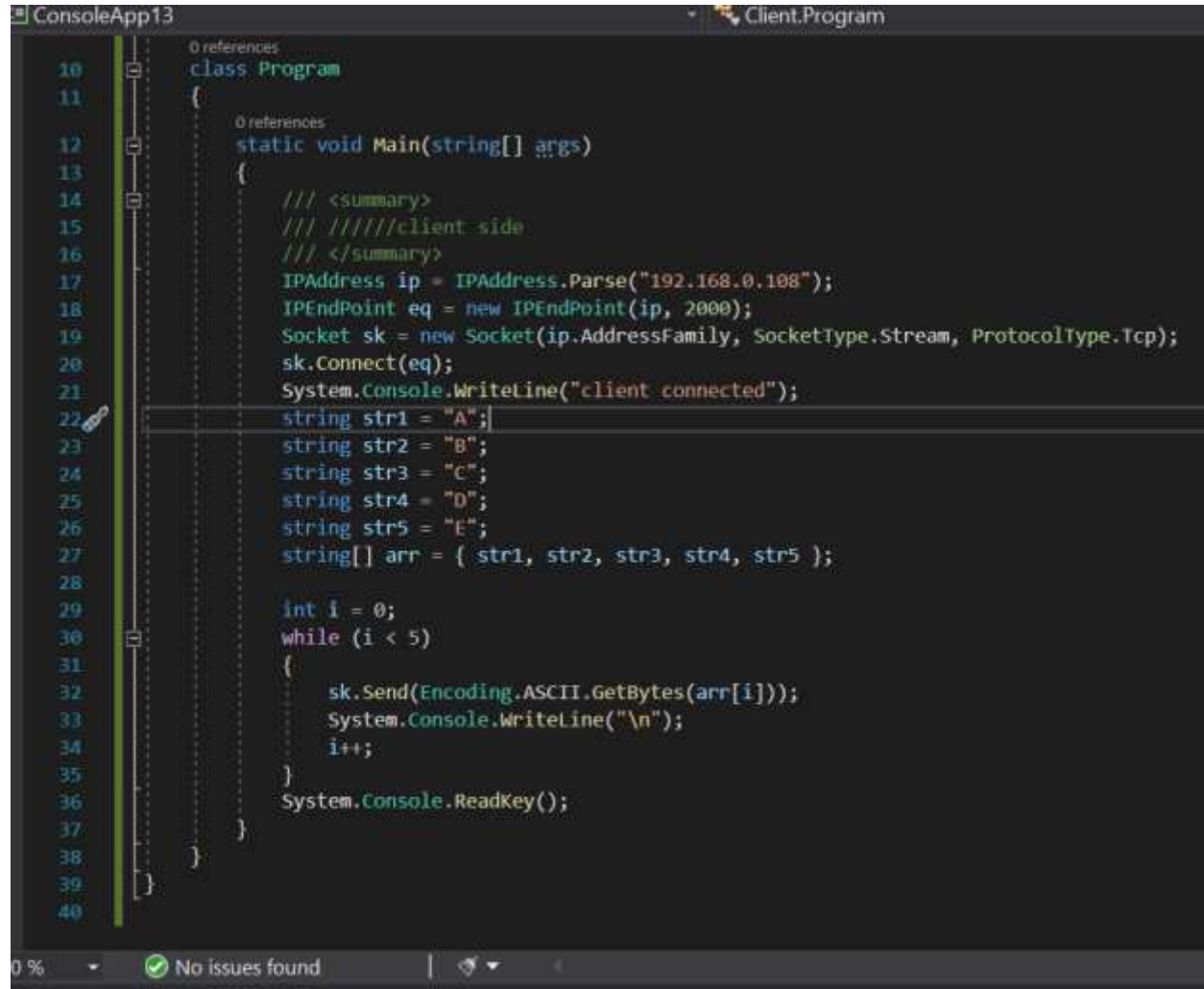
- a. Create a program for client server communication using socket class methods “ i.e. Bind, Listen connect, Accept” (At least 5 messages from each side) on console.
- b. Create a program for multiple clients using threads on console
- c. Repeat (b) with WinForms.

Question # 2:

- a. State all the techniques for client server communication. Mention the type of socket programming.
- b. what is blocking and how to avoid it. what are the problems of TCP and UDP programming and how to circumvent it?

Q1(a):

CLIENT TO SERVER:

A screenshot of a C# code editor window titled "ConsoleApp13" with a sub-window "Client.Program". The code is for a client program that connects to a server at 192.168.0.108 on port 2000. It sends five strings: "A", "B", "C", "D", and "E" in a loop. The status bar at the bottom shows "0 %", a green checkmark, and "No issues found".

```
10 0 references
11 class Program
12 {
13     0 references
14     static void Main(string[] args)
15     {
16         /// <summary>
17         /// /////client side
18         /// </summary>
19         IPAddress ip = IPAddress.Parse("192.168.0.108");
20         IPEndPoint eq = new IPEndPoint(ip, 2000);
21         Socket sk = new Socket(ip.AddressFamily, SocketType.Stream, ProtocolType.Tcp);
22         sk.Connect(eq);
23         System.Console.WriteLine("client connected");
24         string str1 = "A";
25         string str2 = "B";
26         string str3 = "C";
27         string str4 = "D";
28         string str5 = "E";
29         string[] arr = { str1, str2, str3, str4, str5 };
30
31         int i = 0;
32         while (i < 5)
33         {
34             sk.Send(Encoding.ASCII.GetBytes(arr[i]));
35             System.Console.WriteLine("\n");
36             i++;
37         }
38         System.Console.ReadKey();
39     }
40 }
```



PAF-Karachi Institute of Economics and Technology

College of Computing and Information Sciences

```
8:
9: namespace Server //server-side
10: {
11:     class Program
12:     {
13:         static void Main(string[] args)
14:         {
15:             IPAddress ip = IPAddress.Parse("192.168.0.108");
16:             IPEndPoint eq = new IPEndPoint(ip, 2000);
17:             Socket sk = new Socket(ip.AddressFamily, SocketType.Stream, ProtocolType.Tcp);
18:             sk.Bind(eq);
19:             sk.Listen(1);
20:             Socket cl = sk.Accept();
21:             Console.WriteLine("Server Connected");
22:             byte[] arr = new byte[100];
23:             int i = 0;
24:             while (1 < 5)
25:             {
26:                 cl.Receive(arr);
27:                 Console.WriteLine(Encoding.ASCII.GetString(arr));
28:                 i++;
29:             }
30:             //cl.Receive(arr);
31:             //Console.WriteLine(Encoding.ASCII.GetString(arr));
32:             Console.ReadKey();
33:         }
34:     }
35: }
```

```
client connected
Server: Connected
A
B
C
D
E
```



PAF-Karachi Institute of Economics and Technology

College of Computing and Information Sciences

```
12 class Program
13 {
14     0 references
15     static void Main(string[] args)
16     {
17         try
18         {
19             byte[] data = new byte[1024];
20
21             TcpClient tcpClient = new TcpClient("192.168.0.108", 2347);
22             NetworkStream NST= tcpClient.GetStream();
23
24             var serializer = new XmlSerializer(typeof(string[]));
25             var stringArr = (string[])serializer.Deserialize(tcpClient.GetStream());
26
27             foreach (string s in stringArr)
28             {
29                 Console.WriteLine(s);
30             }
31
32             string input = Console.ReadLine();
33             NST.Write(Encoding.ASCII.GetBytes(input), 0, input.Length);
34             NST.Flush();
35
36         }
37         catch (Exception e)
38         {
39             Console.Write(e.Message);
40         }
41     }
42
43     Console.ReadKey();
44 }
45
46 }
```

% | No issues found



PAF-Karachi Institute of Economics and Technology

College of Computing and Information Sciences

```
0 references
class Program
{
    0 references
    static void Main(string[] args) //server-side
    {
        TcpListener tcpListener = new TcpListener(IPAddress.Any, 2347);
        tcpListener.Start();

        int a = 1;
        while (a==1)
        {
            TcpClient tcpClient = tcpListener.AcceptTcpClient();
            byte[] data = new byte[1024];
            NetworkStream ns = tcpClient.GetStream();
            string[] CONTAINER = new string[] { "LETTER ONE", "LETTER TWO", "LETTER THREE" };
            var sizer = new XmlSerializer(typeof(string[]));
            sizer.Serialize(tcpClient.GetStream(), CONTAINER);
            tcpClient.GetStream().Close();
            tcpClient.Close();

            int COLLECT = ns.Read(data, 0, data.Length);

            string IDENTITY = Encoding.ASCII.GetString(data, 0, COLLECT);

            Console.WriteLine(IDENTITY);
        }
    }
}
```

✓ No issues found

OUTPUT:

```
LETTER ONE
LETTER TWO
LETTER THREE
```



- a. **State all the techniques for client server communication. Mention the type of socket programming.**

It has three types:

- **Presentation layer (client tier)**

In this category of client-server setting, the user interface, marketing logic and data logic are present in the same system. This kind of service is reasonable but it is hard to manage due to data variance that allots replication of work. One-tier architecture consists of layers.

For example, Presentation, Business, Data Access layers within a single software package. The data is usually stored in the local system or a shared drive. Applications which handle all the three tiers such as MP3 player, MS Office come under one-tier application

- **Application layer(Business tier)**

In this type of client-server environment, the user interface is stored at client machine and the database is stored on the server. Database logic and business logic are filed at either client or server but it needs to be maintained. If Business Logic and Data Logic are collected at a client side, it is named as fat client thin server architecture. If Business Logic and Data Logic are handled on the server, it is called thin client fat server architecture. This is considered as affordable.

In two-tier architecture, client and server have to come in direct incorporation. If a client is giving an input to the server there shouldn't be any intermediate. This is done for rapid results and to avoid confusion between different clients. For instance, online ticket reservations software use this two-tier architecture.

- **Database layer (Data tier)**

The Three-tier architecture is split into 3 parts, namely, The presentation layer (Client Tier), Application layer (Business Tier) and Database layer (Data Tier). The Client system manages Presentation layer; the Application server takes care of the Application layer, and the Server system supervises Database layer.

There are two types of socket programming:

- **Stream socket**

This is a type of network which has connection less point for sending and receiving packets. It is similar to mailbox. The letters (data) posted into the box are collected and delivered (transmitted) to a letterbox (receiving socket).



- Datagram socket

stream socket is type of interprocess communications socket or network socket which provides a connection-oriented, sequenced, and unique flow of data without record boundaries with well-defined mechanisms for creating and destroying connections and for detecting errors. It is similar to phone. A connection is established between the phones (two ends) and a conversation (transfer of data) takes place.

- a. what is blocking and how to avoid it. what are the problems of TCP and UDP programming and how to circumvent it?**

The default mode of socket calls is blocking. A blocking call does not return to your program until the event you requested has been completed. For example, if you issue a blocking `recvfrom()` call, the call does not return to your program until data is available from the other socket application. A blocking `accept()` call does not return to your program until a client connects to your socket program.

How to overcome it?

Change a socket to nonblocking mode using the `ioctl()` call that specifies command `FIONBIO` and a fullword (four byte) argument with a nonzero binary value. Any succeeding socket calls against the involved socket descriptor are nonblocking calls.

Alternatively, use the `fcntl()` call using the `F_SETFL` command and `FNDELAY` as an argument.

Nonblocking calls return to your program immediately to reveal whether the requested service was completed. An error number may mean that your call would have blocked had it been a blocking call.

UDP PROBLEMS:

- The delivery of data to the destination cannot be guaranteed in UDP.
- There is no sequencing of data in UDP. If the order is required, it has to be managed by the application layer.
- There is no retransmission of lost packets in the User Datagram Protocol (UDP).
- UDP has an 8 bytes fixed-length header.

TCP PROBLEMS:

- TCP is heavy-weight.
- TCP doesn't support Broadcasting.
- TCP is comparatively slower than UDP

We cannot overcome all problems in both of them but some of them can be solved using set of skills for socket programming accordingly.