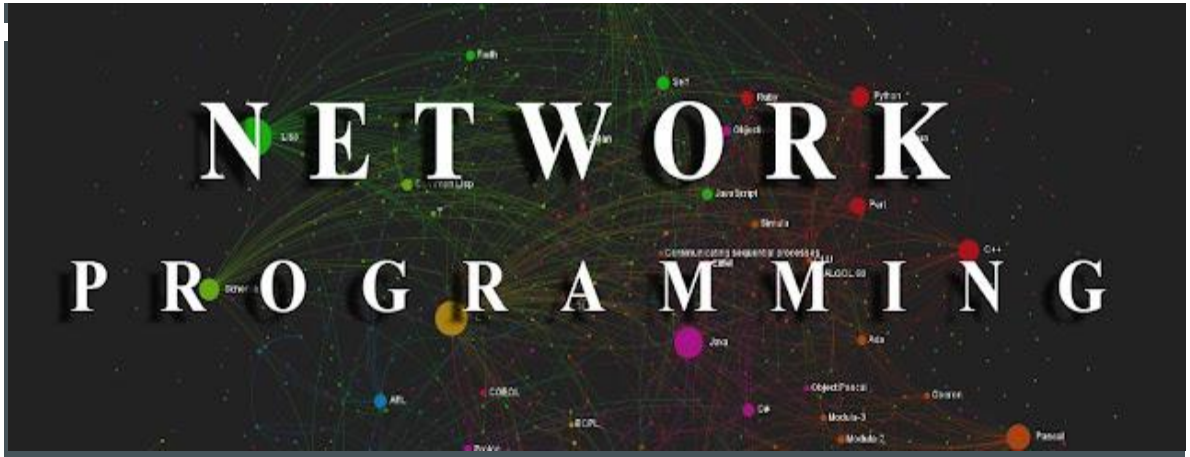


IT432 - NETWORK PROGRAMMING

DR. ALI HUSSEIN AHMED

ALI.HUSSEIN@AUN.EDU.EG

IT DEPT - OFFICE NO. 312

2022

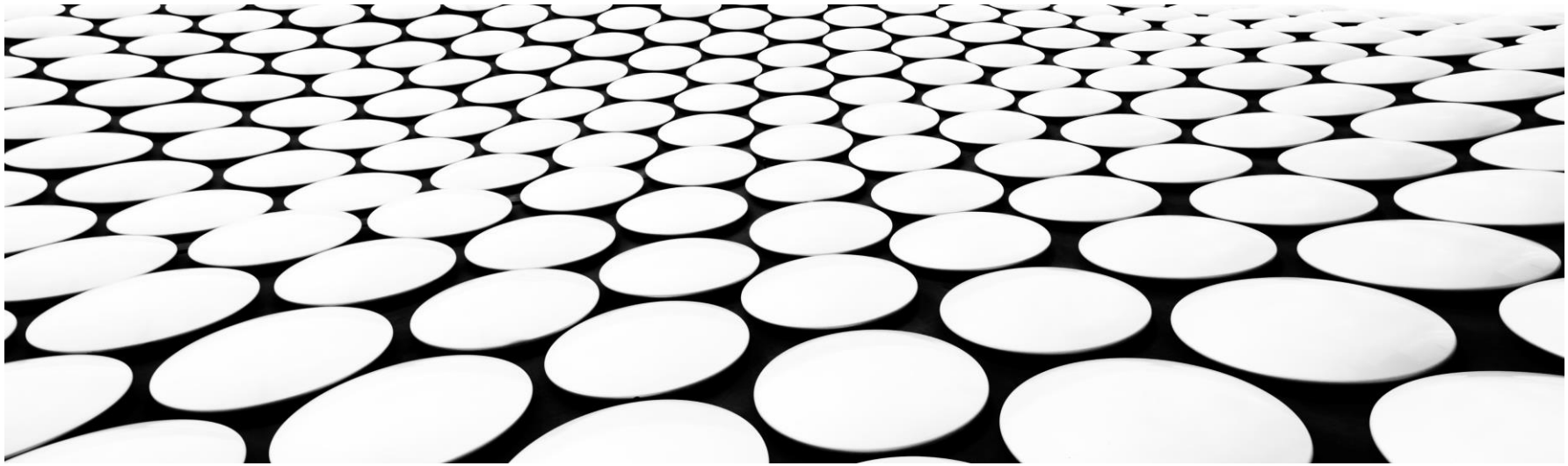
جامعة أسيوط

كلية الحاسبات والمعلومات

كلية معتمدة من الهيئة القومية لضمان الجودة والاعتماد

LECTURE 6

THE HELPER CLASSES , THE MULTITHREADED SERVER



C# SOCKET HELPER CLASSES

- The .NET Framework supports the normal socket interface for advanced network programmers, but it also provides a simplified interface for easier network programming.
- The *simplified socket helper classes* help network programmers create socket programs with simpler statements and less coding—two important benefits for any programmer.
- Here are the three helper classes used for socket programming:
- • TcpClient
- • TcpListener
- • UdpClient

TCPCLIENT

- The methods of the TcpClient class are used to create client network programs that follow the connection-oriented network model.
- The TcpClient methods mirror those in normal socket programming, but many of the steps are compacted to simplify the programming task.
- TcpClient newclient = new TcpClient();
- newclient.Connect("www.isp.net", 8000);
- The TcpClient class will automatically attempt to resolve the hostname to the proper IP address. That's a lot of work already done for you!

GETTING THE STREAM

- The GetStream() method is used to create a NetworkStream object that allows you to send and receive bytes on the socket.

```
NetworkStream ns = newclient.GetStream();
byte[] outbytes = Encoding.ASCII.GetBytes("Testing");
ns.Write(outbytes, 0, outbytes.Length);
byte[] inbytes = new byte[1024];
ns.Read(inbytes, 0, inbytes.Length);
string instring = Encoding.ASCII.GetString(inbytes);
Console.WriteLine(instring);
ns.Close();
newclient.Close();
```

TCPLISTENER

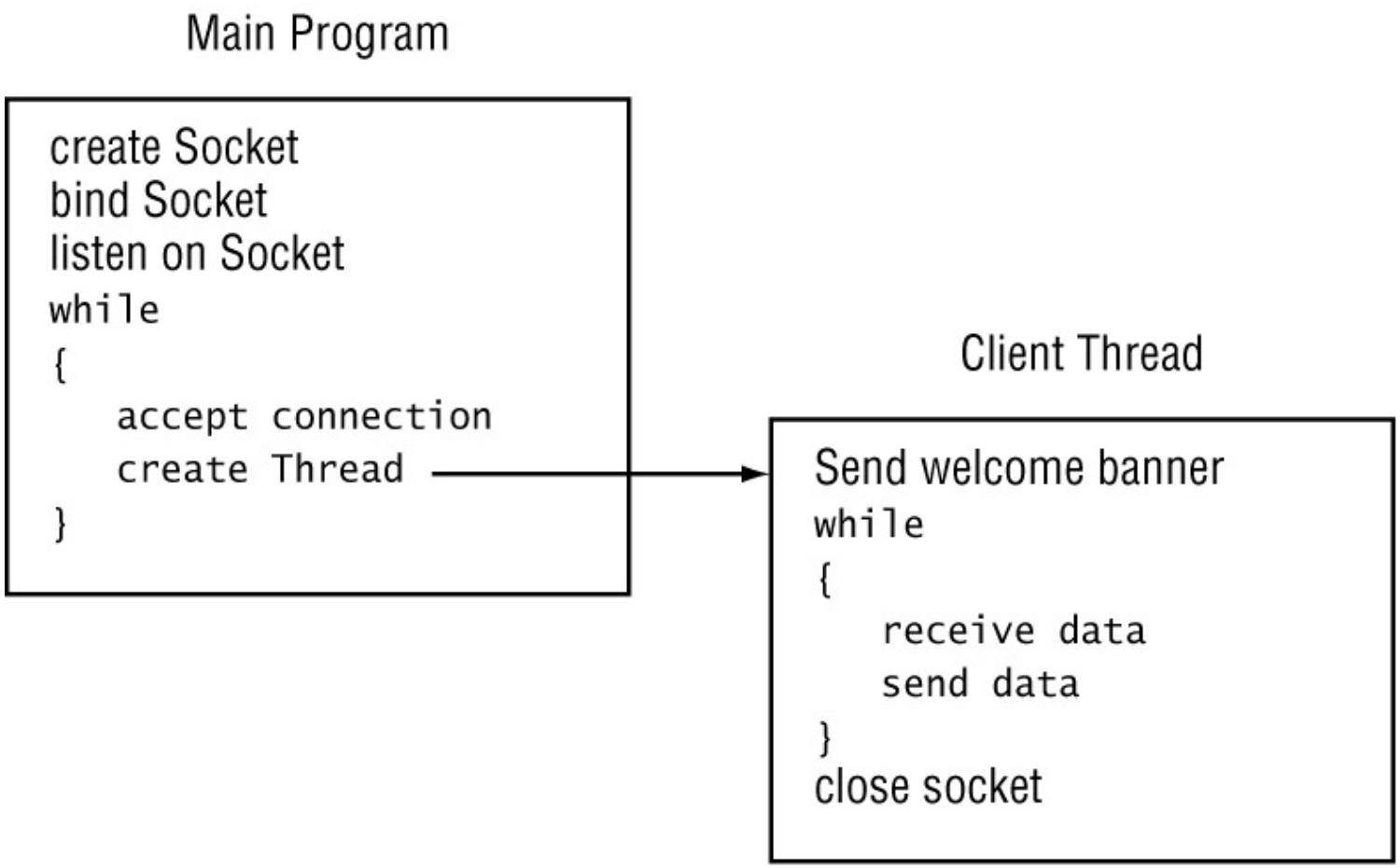
- the TcpListener class simplifies server programs; their class constructors are similar as well. Here are the three constructor formats:
- •TcpListener(int port)binds to a specific local port number
- •TcpListener(IPEndPoint ie)binds to a specific local EndPoint
- •TcpListener(IPAddress addr, int port) binds to a specific local IPAddress and port number

```
TcpListener newserver = new TcpListener(9050);
newserver.Start();
TcpClient newclient = newserver.AcceptTcpClient();
NetworkStream ns = newclient.GetStream();
byte[] outbytes = Encoding.ASCII.GetBytes("Testing");
ns.Write(outbytes, 0, outbytes.Length);
byte[] inbytes = new byte[1024];
ns.Read(inbytes, 0, inbytes.Length);
string instring = Encoding.ASCII.GetString(inbytes);
Console.WriteLine(instring);
ns.Close();
newclient.Close();
newserver.Stop();
```

UDPCLIENT

- For applications that require a connectionless socket, the UdpClient class provides a simple interface to UDP sockets.
- You may be wondering why there is not a listener version of the UDP helper class. The answer is simple: you don't need one.
- Remember, UDP is a connectionless protocol, so there is no such thing as a client or server; there are only UDP sockets either waiting for or sending data. You do not need to bind the UDP socket to a specific address and wait for incoming data.

MULTITHREADING



MULTITHREAD SERVER, WHY?

- A multithreaded server is any server that has more than one thread.
- Because a transport requires its own thread, multithreaded servers also have multiple transports.
- The number of thread-transport pairs that a server contains defines the number of requests that the server can handle in parallel.
- The normal server situation is that a single server handles many clients.
- Each client have its own handling thread , socket, and streams.

Server – high level view

Create a socket

Bind the socket

Listen for connections

Accept new client connections

Read/write to client connections

Shutdown connection

CLIENT – HIGH LEVEL VIEW

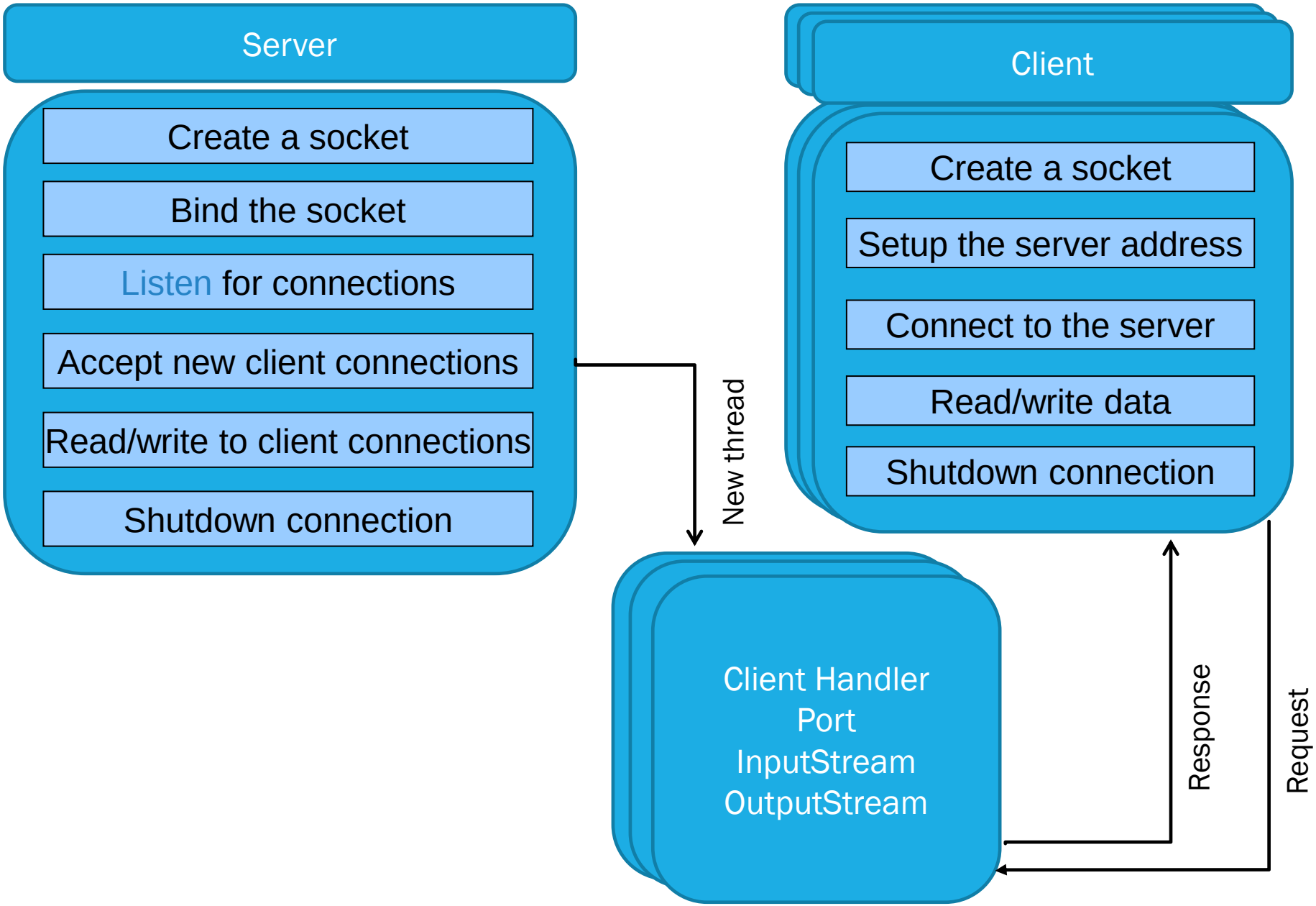
Create a socket

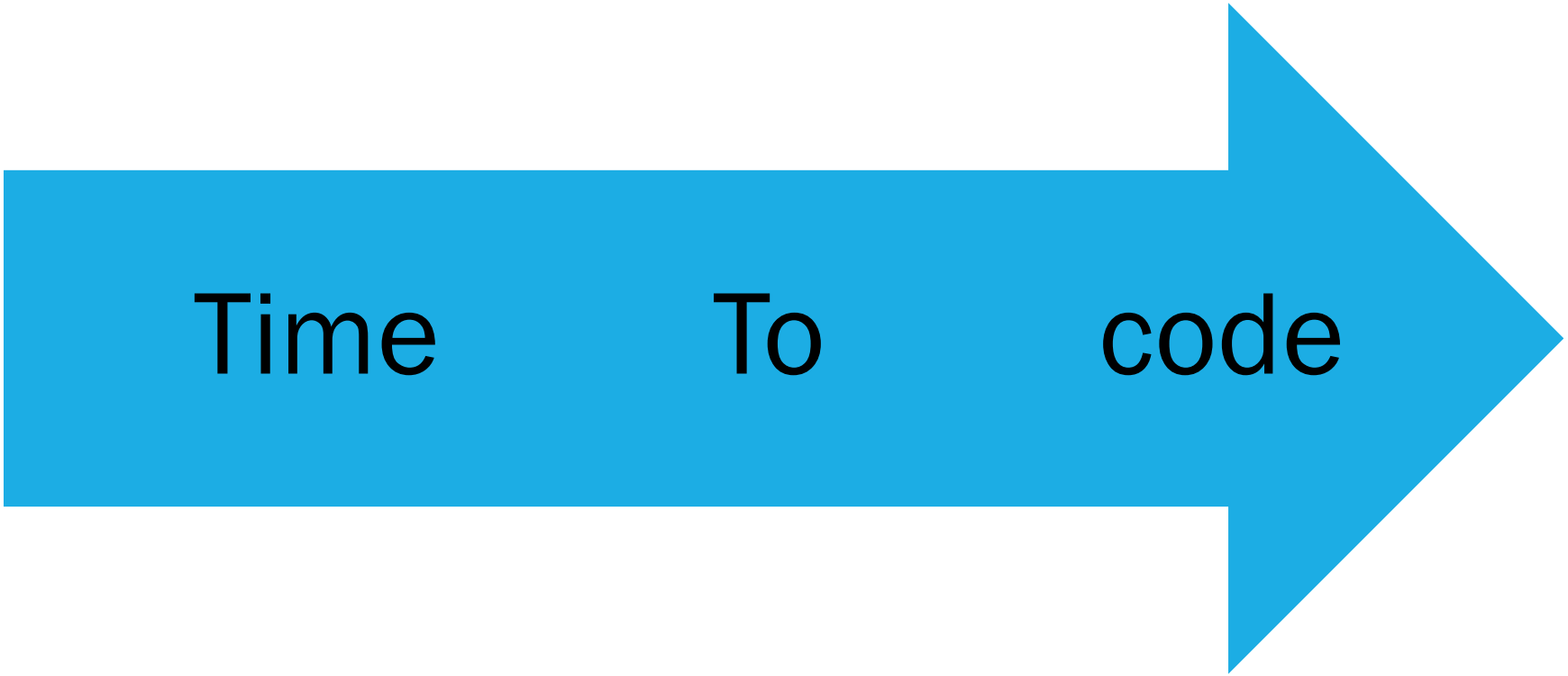
Setup the server address

Connect to the server

Read/write data

Shutdown connection


Server
Create a socket
Bind the socket
Listen for connections
Accept new client connections
Read/write to client connections
Shutdown connection
New thread
Client
Create a socket
Setup the server address
Connect to the server
Read/write data
Shutdown connection
Client Handler
Port
InputStream
OutputStream
Response
Request


Time To code

THE MULTITHREADED SERVER

```
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(8000);
        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";
```

```
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();
Console.WriteLine("Client disconnected: {0} active
connections",
                  connections);
    }
}
```

- The next step is to try any client (say the client from the previous lecture)

C:\Users\Dr. Ali\source\repos\binarystreams\binarystreams\bin\Debug\net6.0\binarystreams.exe

```
Waiting for clients...
New client accepted: 1 active connections
New client accepted: 2 active connections
New client accepted: 3 active connections
```

Asynchronous TCP Client

Enter text string:

Connect

Send

Welcome to my test server
abc

Connection Status: Connected to: 127.0.0.1:8000

Asynchronous TCP Client

Enter text string:

Send

Welcome to my test server
cvd

Connection Status: Connected to: 127.0.0.1:8000

Asynchronous TCP Client

Enter text string:

Connect

Send

Disconnect

Welcome to my test server
qwe

Connection Status: Connected to: 127.0.0.1:8000

27
28
29

100 %

Output

- Thank U