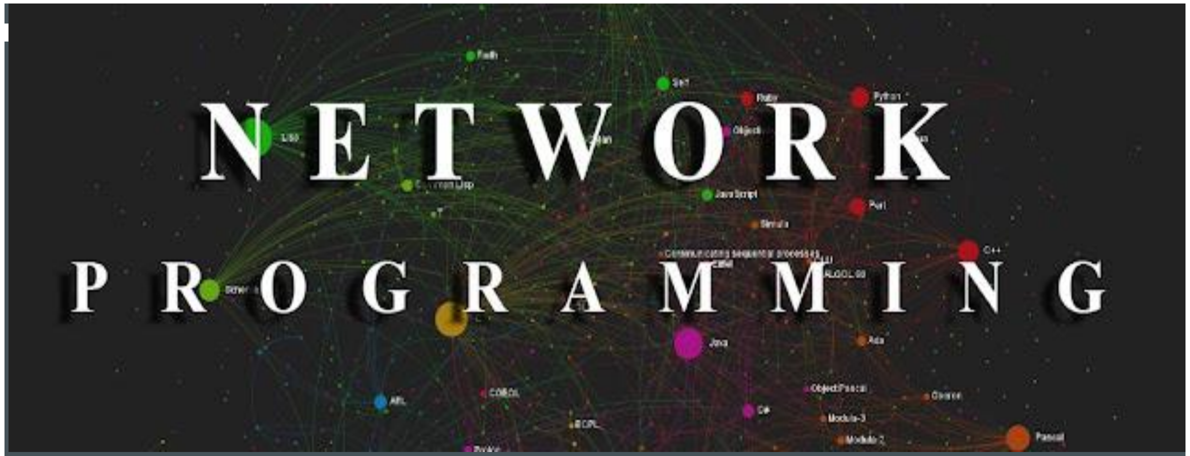


IT432 - NETWORK PROGRAMMING

DR. ALI HUSSEIN AHMED

ALI.HUSSEIN@AUN.EDU.EG

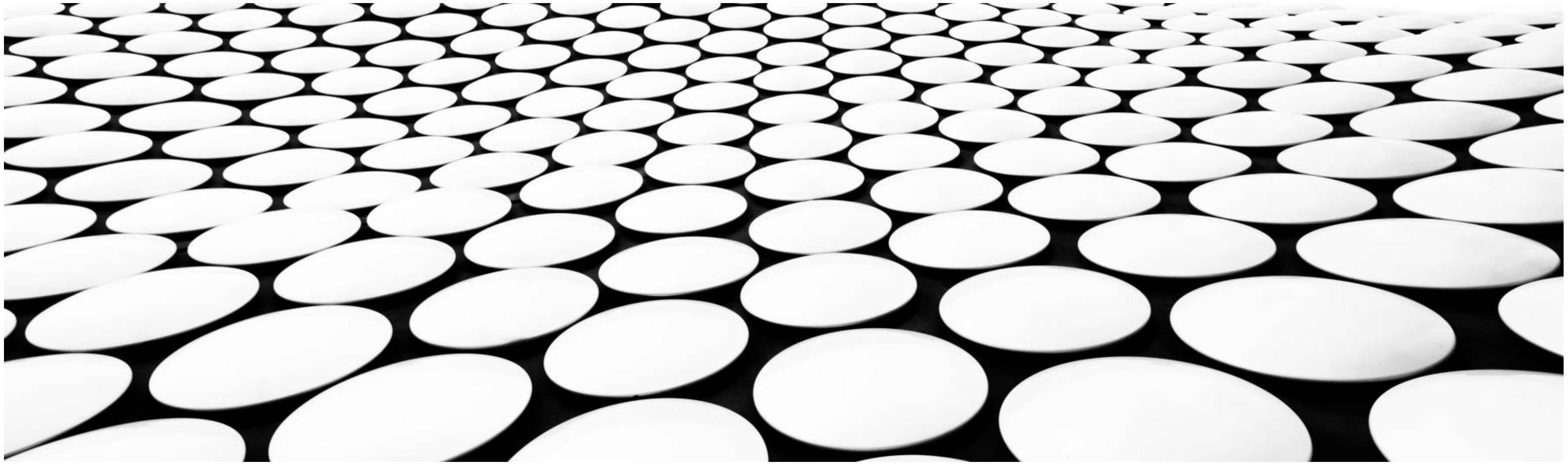
IT DEPT - OFFICE NO. 312

2022

كلية معتمدة من الهيئة القومية
لضمان الجودة والاعتماد

LECTURE 4

CREATING THE CLIENT . . .

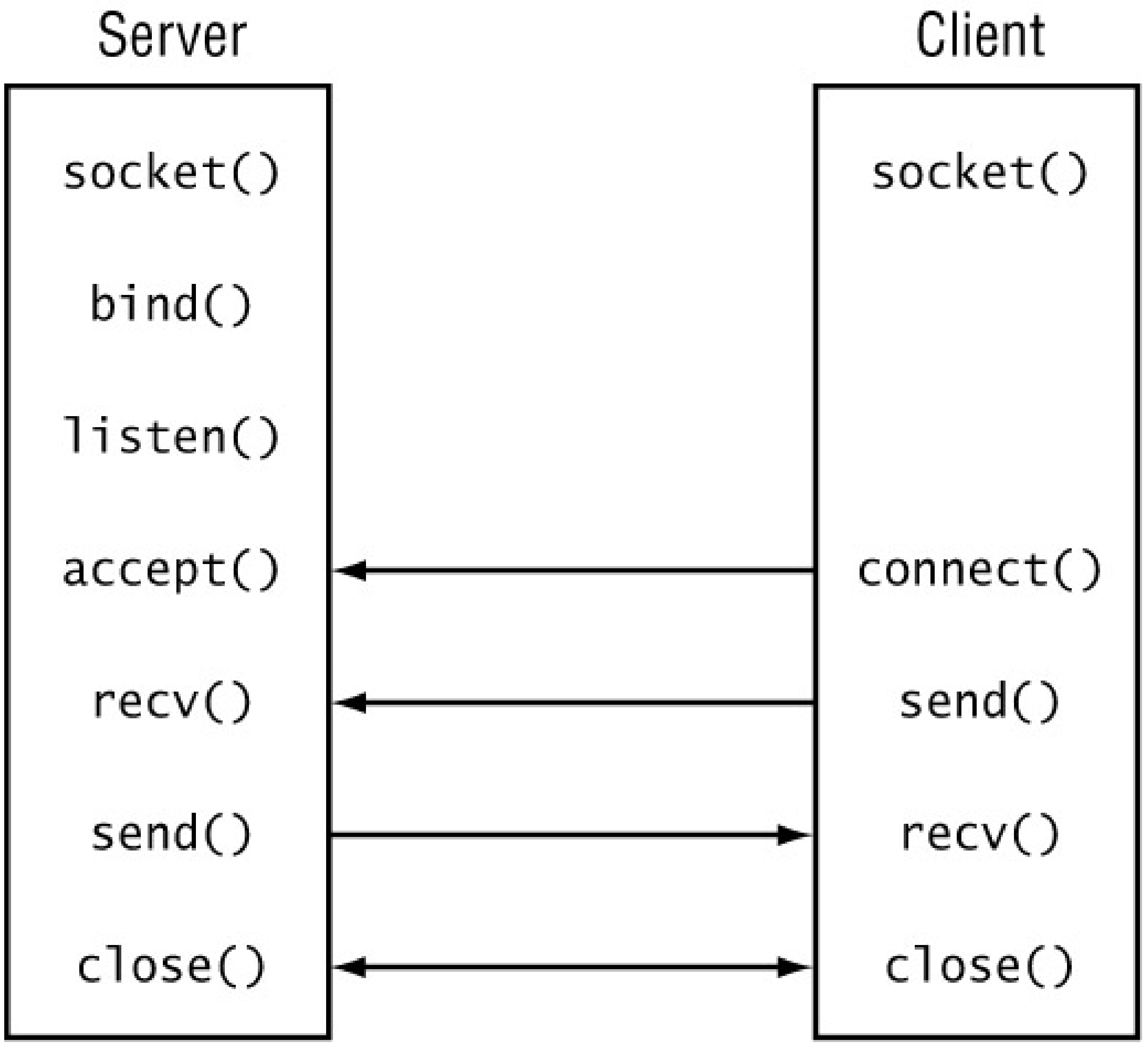


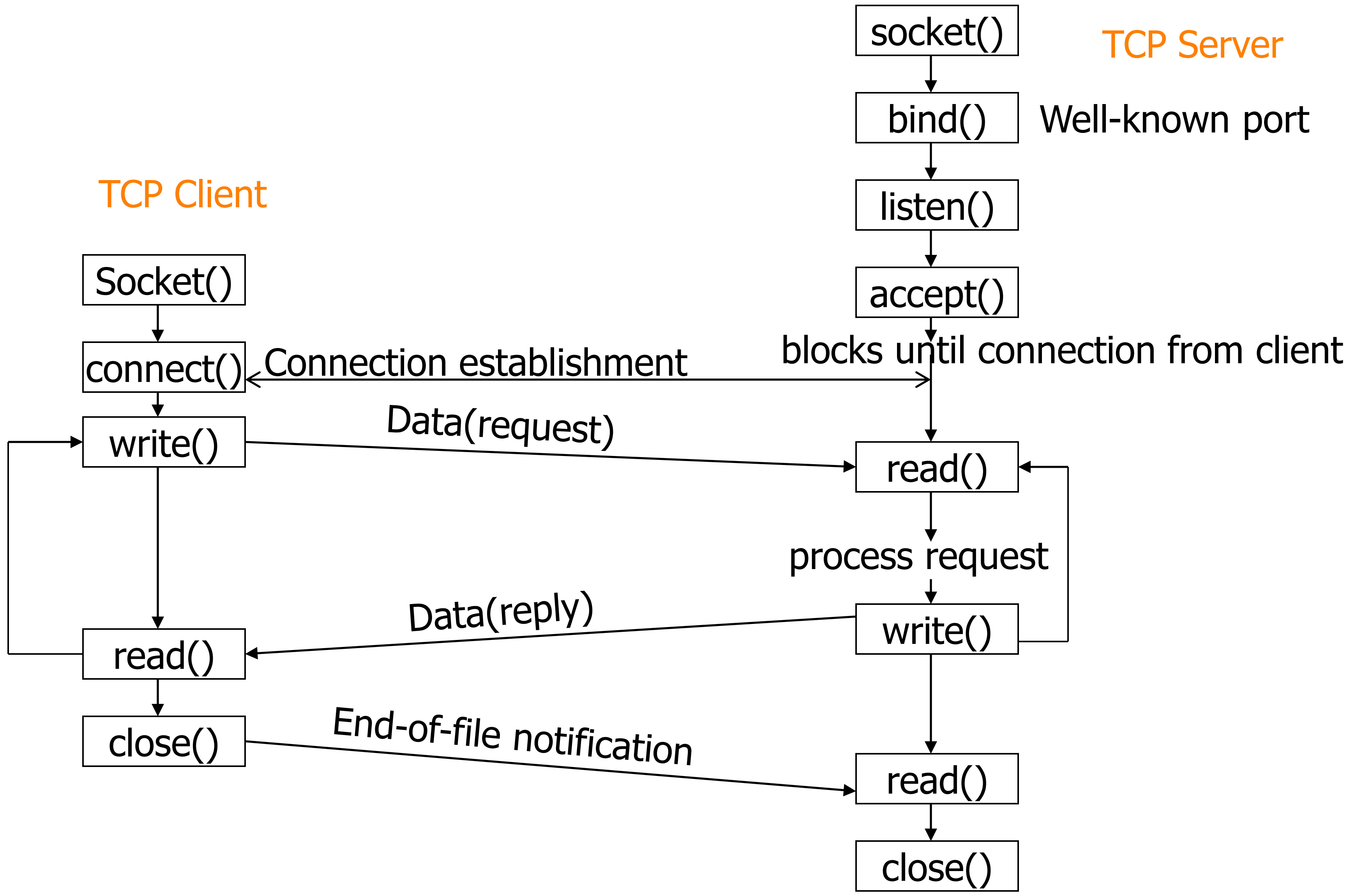
SOCKETS

SOCKET TYPES

- Connection-Oriented Sockets
- Connectionless Sockets

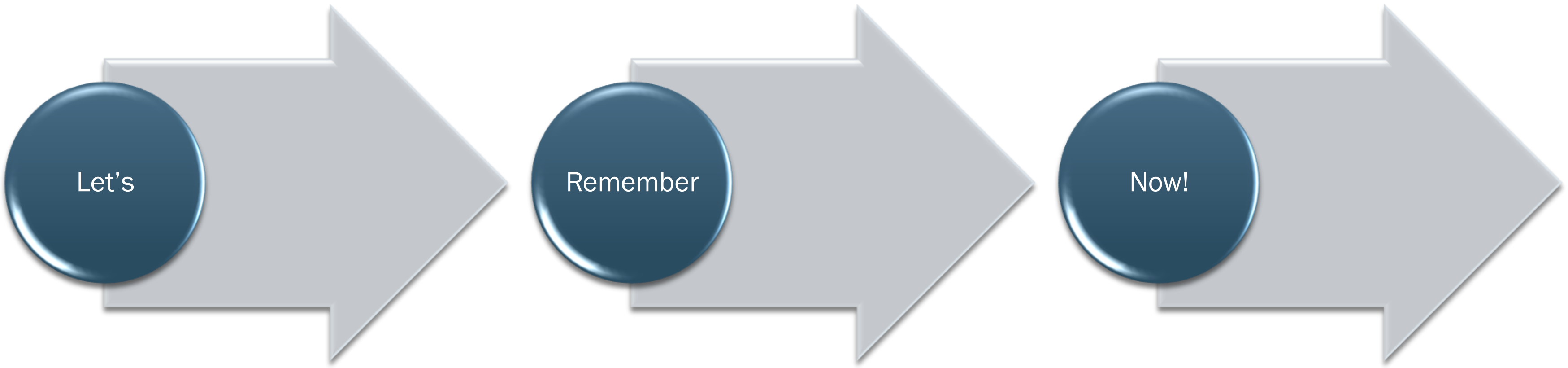
STEPS


Server
Client
socket()
bind()
listen()
accept()
recv()
send()
close()
socket()
connect()
send()
recv()
close()


socket()
TCP Server
bind()
Well-known port
listen()
TCP Client
Socket()
accept()
blocks until connection from client
connect()
Connection establishment
write()
Data(request)
read()
process request
Data(reply)
read()
write()
close()
End-of-file notification
read()
close()

THE CLIENT FUNCTIONS

- In a connection-oriented socket, the client must bind to the specific host address and port for the application.
- For client programs, the connect() function is used instead of the listen() function:
- Once the connect() function succeeds, the client is connected to the server and can use the standard send() and receive() functions to transmit data back and forth with the server.


Let's
Remember
Now!

ECHO SERVER

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class SimpleTcpSrvr
{
    public static void Main()
    {
        int recv;
        byte[] data = new byte[1024];
        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 clientep = (IPEndPoint)client.RemoteEndPoint;
```

```
 string welcome = "Welcome to my test server";
        data = Encoding.ASCII.GetBytes(welcome);
        client.Send(data, data.Length, SocketFlags.None);
        while (true)
        {
            data = new byte[1024];
            recv = client.Receive(data);
            if (recv == 0)
                break;
            Console.WriteLine(Encoding.ASCII.GetString(data, 0,
               recv));
            client.Send(data, recv, SocketFlags.None);
        }
        Console.WriteLine("Disconnected from {0}",clientep.Address);
        client.Close();
        newsock.Close();
    }
}
```

NEXT TOPICS

- TCP CLIENT
- When TCP Goes Bad
- Using Fixed-Sized Messages

SIMPLE CLIENT

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
namespace client {
    class Program     {
        static void Main(string[] args)         {
            byte[] bytes = new byte[1024];
            IPAddress host = IPAddress.Parse("127.0.0.1");
            IPEndPoint hostep = new IPEndPoint(host, 9050);
            Socket sock = new Socket(AddressFamily.InterNetwork,
                SocketType.Stream, ProtocolType.Tcp);
            sock.Connect(hostep);
            Console.WriteLine("Socket connected to {0}",
                    sock.RemoteEndPoint.ToString());
            // Encode the data string into a byte array.
```

SIMPLE CLIENT CONT.

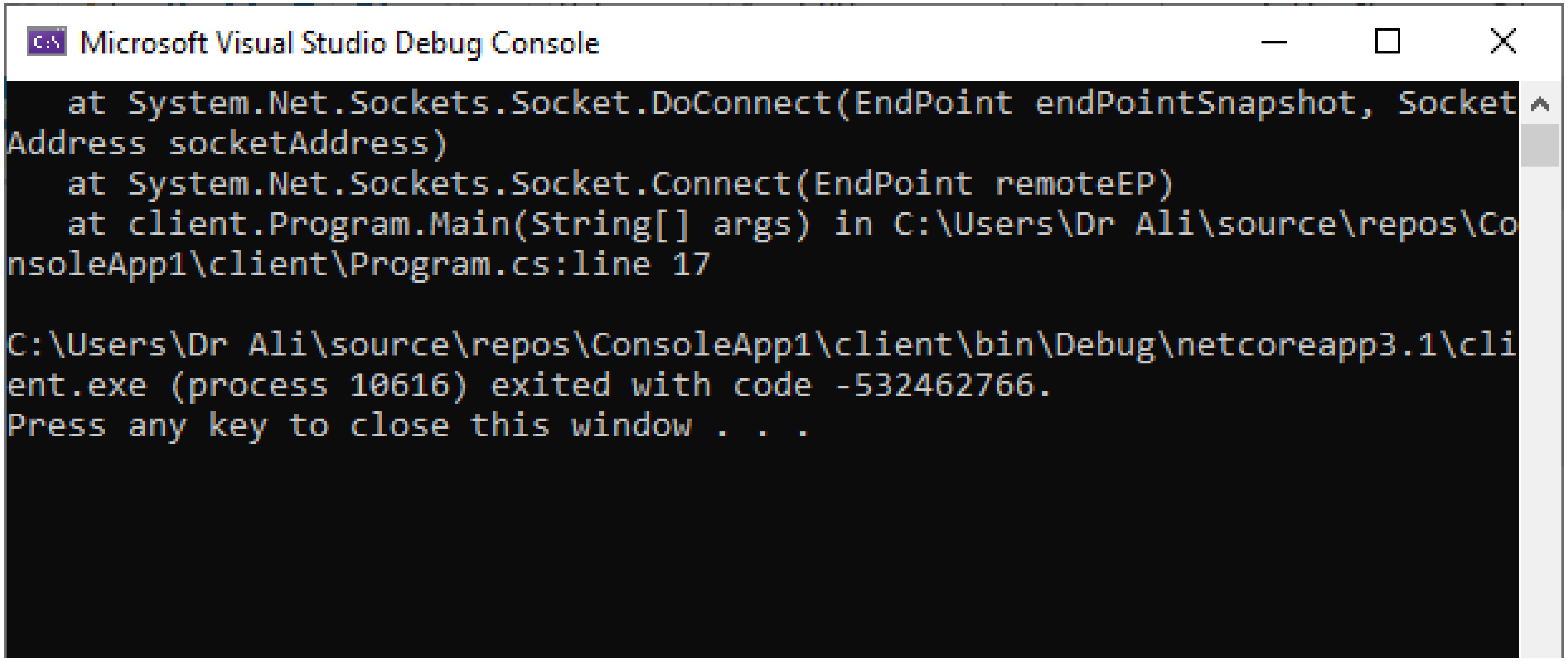
```
 byte[] msg = Encoding.ASCII.GetBytes(Console.ReadLine());

                    // Send the data through the socket.
                    int bytesSent = sock.Send(msg);

                    // Receive the response from the remote device.
                    int bytesRec = sock.Receive(bytes);
                    Console.WriteLine("Echoed test =
{0}",Encoding.ASCII.GetString(bytes, 0, bytesRec));
                sock.Close();



            }
        }
    }
```



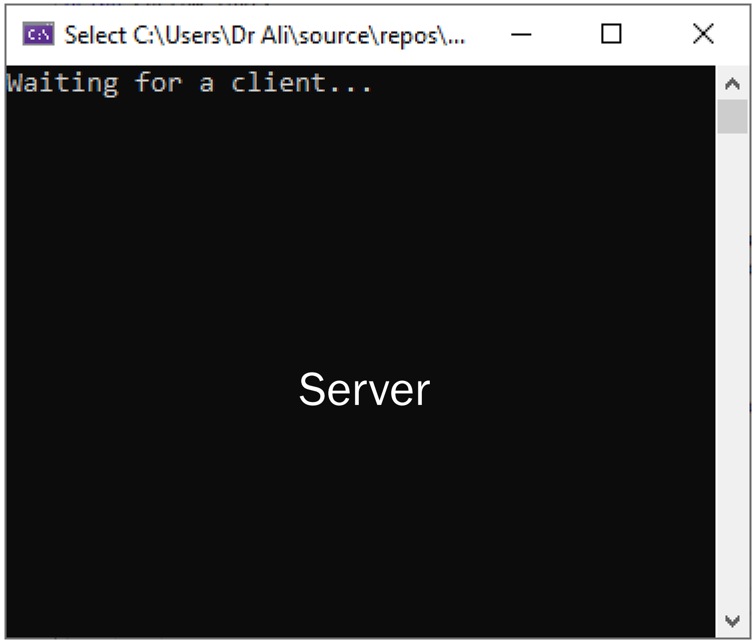
```
   at System.Net.Sockets.Socket.DoConnect(EndPoint endPointSnapshot, Socket
Address socketAddress)
   at System.Net.Sockets.Socket.Connect(EndPoint remoteEP)
   at client.Program.Main(String[] args) in C:\Users\Dr Ali\source\repos\Co
nsoleApp1\client\Program.cs:line 17

C:\Users\Dr Ali\source\repos\ConsoleApp1\client\bin\Debug\netcoreapp3.1\cli
ent.exe (process 10616) exited with code -532462766.
Press any key to close this window . . .
```

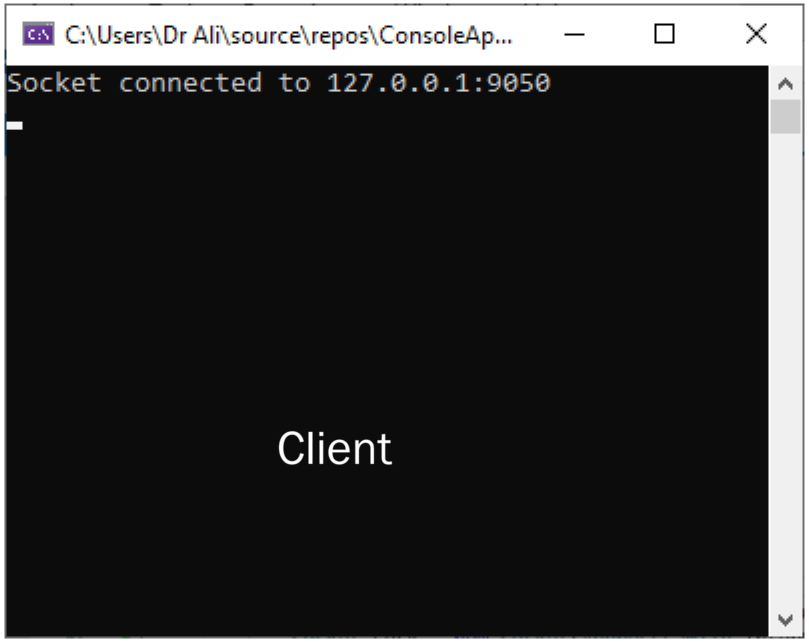
RUNNING THE CLIENT FIRST, WHY?

YOU HAVE TO ..

- Run the server program, it must be always on.
- Run the client ..


Select C:\Users\Dr Ali\source\repos\...

```
Waiting for a client...
```


C:\Users\Dr Ali\source\repos\ConsoleAp...

```
Socket connected to 127.0.0.1:9050
```

THE C# NETWORK STREAMS

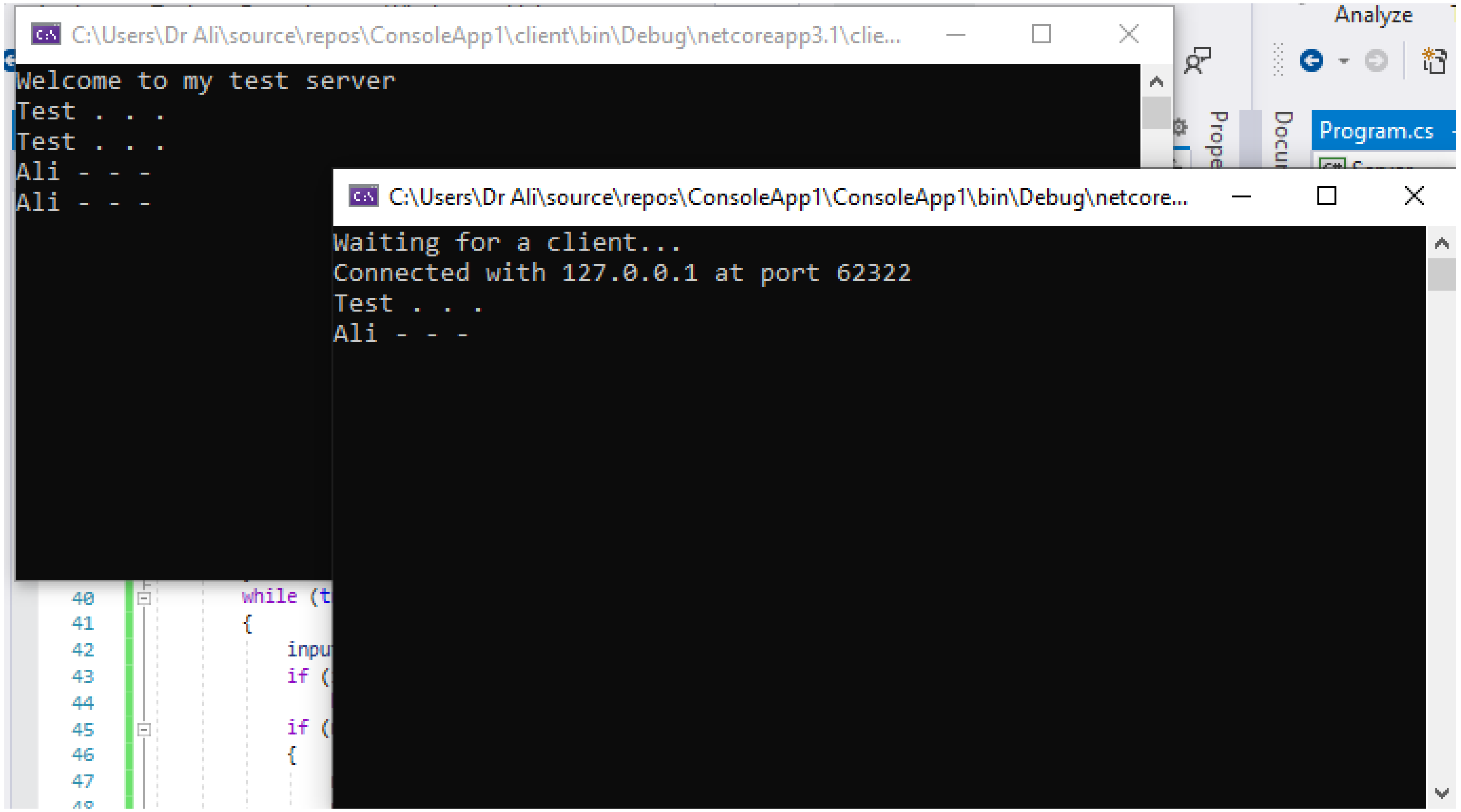
- The .NET Framework supplies some extra classes to help out.
- This slides describes the NetworkStream class, which provides a stream interface for sockets, as well as two additional stream classes, StreamReader and StreamWriter, that can be used to send and receive text messages using TCP.

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

USING NETWORKSTREAM IN OUR CLIENT

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class NetworkStreamTcpClient {
    public static void Main()     {
        byte[] data = new byte[1024];
        string input, stringData;
        int recv;
        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);
if (ns.CanRead){
           recv = ns.Read(data, 0, data.Length);
           stringData = Encoding.ASCII.GetString(data, 0, recv);
           Console.WriteLine(stringData);
       }
       else {
           Console.WriteLine("Error: Can't read from this socket");
           ns.Close();
           server.Close();
           return;
       }
       while (true) {
           input = Console.ReadLine();
           if (input == "exit")
               break;
           if (ns.CanWrite)  {
               ns.Write(Encoding.ASCII.GetBytes(input), 0, input.Length);
               ns.Flush();
           }
           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.Shutdown(SocketShutdown.Both);
       server.Close();   }}
```

LET'S RUN THE ENHANCED CLIENT



C:\Users\Dr Ali\source\repos\ConsoleApp1\client\bin\Debug\netcoreapp3.1\clie...

```
Welcome to my test server
Test . . .
Test . . .
Ali - - -
Ali - - -
```

C:\Users\Dr Ali\source\repos\ConsoleApp1\ConsoleApp1\bin\Debug\netcore...

```
Waiting for a client...
Connected with 127.0.0.1 at port 62322
Test . . .
Ali - - -
```

```
while (t
{
    inpu
    if (

    if (
    {
```

THE STREAMREADER AND STREAMWRITER CLASSES

- The two helper classes can be used with any stream, say our network stream.
- The next few slides rewrites the server program to be more efficient.

STREAM SERVER

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using System.Text;
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();
    }
}
```

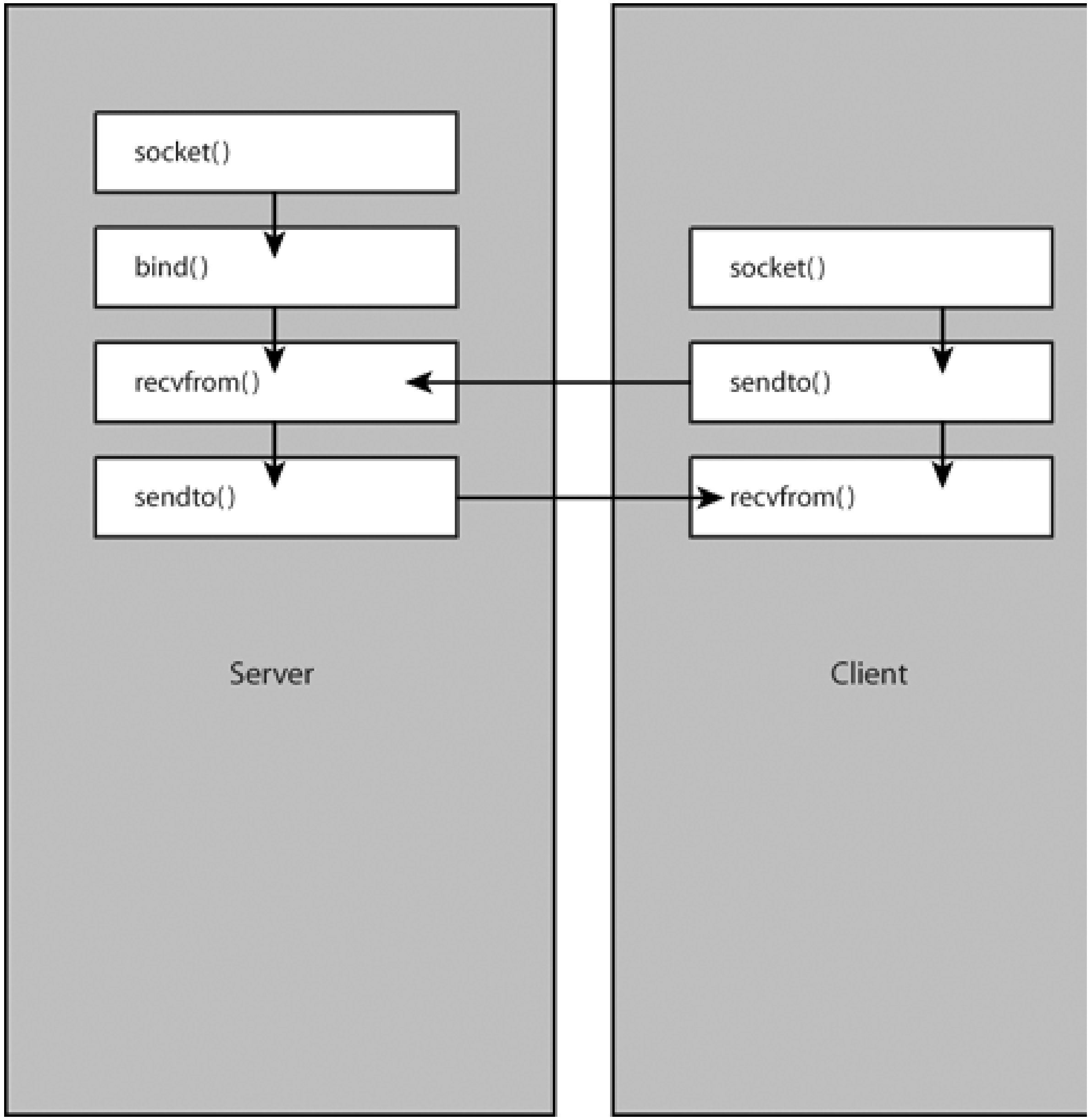
STREAM CLIENT

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using System.Text;
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();
   }}
```

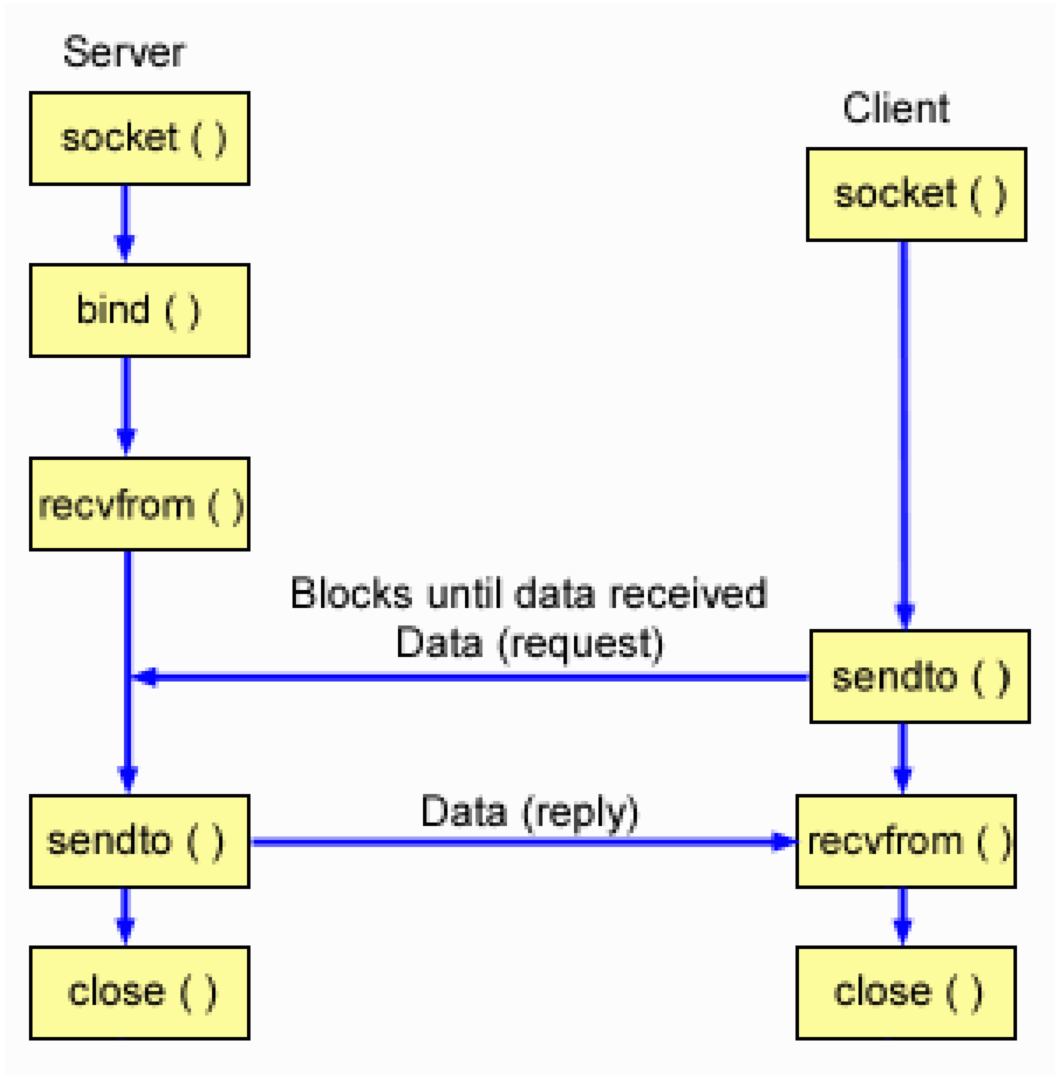
CONNECTIONLESS SOCKETS

- Connectionless sockets allow the sending of messages in self-contained packets.
- A single read method reads the entire message sent by a single sent method.
- This helps you avoid the hassle of trying to match message boundaries in packets.
- Unfortunately, UDP packets are not guaranteed to arrive at their destination.
- Many factors, such as busy networks, can prevent the packet from making it to its destination.

HOW IT RUNS


socket()
bind()
recvfrom()
sendto()
Server
socket()
sendto()
recvfrom()
Client

ITS NOT SERVER, JUST WHO ARE THE ONE ISSUING THE BIND AND THE FIRST RECEIVE



UDP SERVER

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class SimpleUdpSrvr
{
    public static void Main()
    {
        int recv;
        byte[] data = new byte[1024];
        IPEndPoint ipep = new IPEndPoint(IPAddress.Any, 9050);
        Socket newsock = new Socket(AddressFamily.InterNetwork,
                SocketType.Dgram, ProtocolType.Udp);
        newsock.Bind(ipep);
        Console.WriteLine("Waiting for a client...");
        IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);
        EndPoint Remote = (EndPoint)(sender);
        recv = newsock.ReceiveFrom(data, ref Remote);
Console.WriteLine("Message received from {0}:",
        Remote.ToString());
         Console.WriteLine(Encoding.ASCII.GetString(data, 0,
        recv));
         string welcome = "Welcome to my test server";
         data = Encoding.ASCII.GetBytes(welcome);
         newsock.SendTo(data, data.Length, SocketFlags.None,
Remote);
         while (true)
         {
              data = new byte[1024];
              recv = newsock.ReceiveFrom(data, ref Remote);

              Console.WriteLine(Encoding.ASCII.GetString(data, 0,
                 recv));
              newsock.SendTo(data, recv, SocketFlags.None, Remote);
         }
     }
}
```

UDP CLIENT

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class SimpleUdpClient
{
    public static void Main()
    {
        byte[] data = new byte[1024];
        string input, stringData;
        IPEndPoint ipep = new IPEndPoint(
                IPAddress.Parse("127.0.0.1"), 9050);
        Socket server = new Socket(AddressFamily.InterNetwork,
                SocketType.Dgram, ProtocolType.Udp);
        string welcome = "Hello, are you there?";
        data = Encoding.ASCII.GetBytes(welcome);
        server.SendTo(data, data.Length, SocketFlags.None, ipep);
        IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);
        EndPoint Remote = (EndPoint)sender;
        data = new byte[1024];
```

```
 int recv = server.ReceiveFrom(data, ref Remote);
        Console.WriteLine("Message received from {0}:",
Remote.ToString());
        Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));
        while (true)
        {
            input = Console.ReadLine();
            if (input == "exit")
                break;
            server.SendTo(Encoding.ASCII.GetBytes(input), Remote);
            data = new byte[1024];
            recv = server.ReceiveFrom(data, ref Remote);
            stringData = Encoding.ASCII.GetString(data, 0, recv);
            Console.WriteLine(stringData);
        }
        Console.WriteLine("Stopping client");
        server.Close();
    }
}
```

SECTION WORK THIS WEEK

- Create clean UI for the following programs:
  - Implement and run all the code snippets listed in this lecture.
  - Can you tune the last stream server and client to send a file!
    - Try to send a file using client server program
    - Your project should send the file from server to client.
    - Make use of file streams you learnt before.
  - Compress the files and submit them on the subject team (Eng Salma will create the Dir)
  - You will take 5 points /100 if U did this
  - Work at home, deliver & submit on section (grading will be on section)