

NETWORK PROGRAMMING

IT432 - NETWORK PROGRAMMING

DR. ALI HUSSEIN AHMED

ALI.HUSSEIN@AUN.EDU.EG

IT DEPT - OFFICE NO. 312

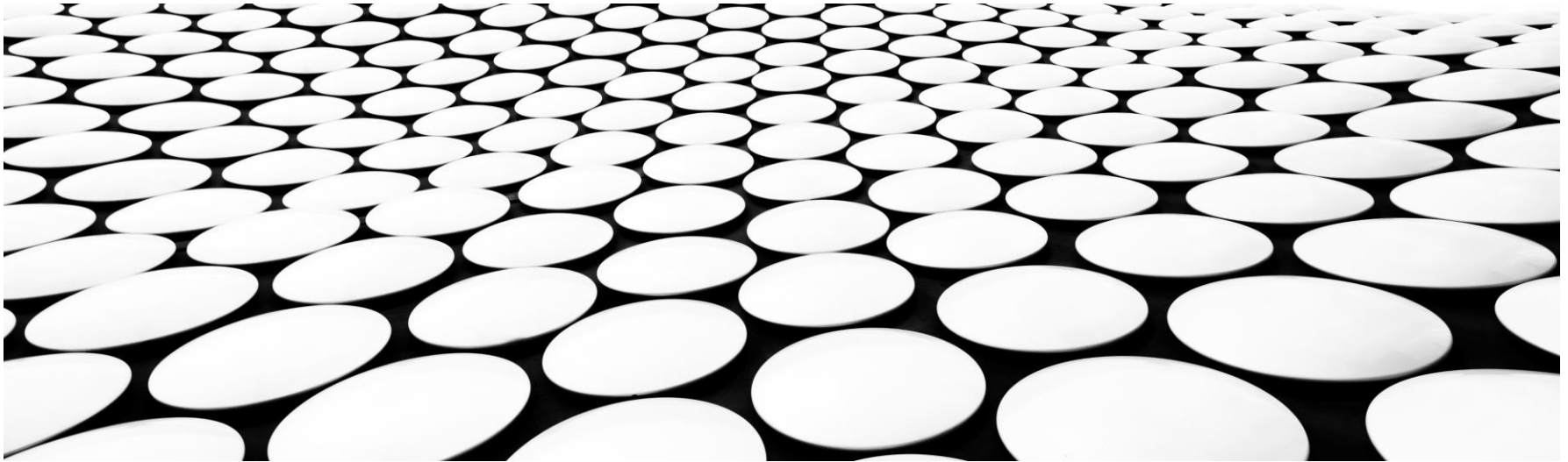
2022



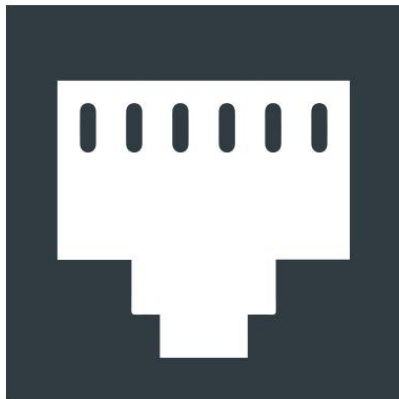
كلية معتمدة من الهيئة القومية
لضمان الجودة والاعتماد

LECTURE 3

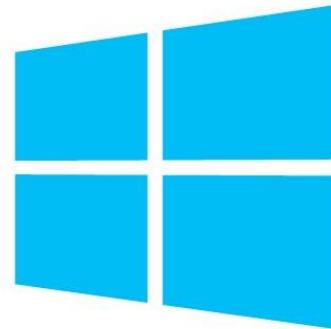
THE BASICS . . .



SOCKETS



SOCKETS



WHAT IS A SOCKET?

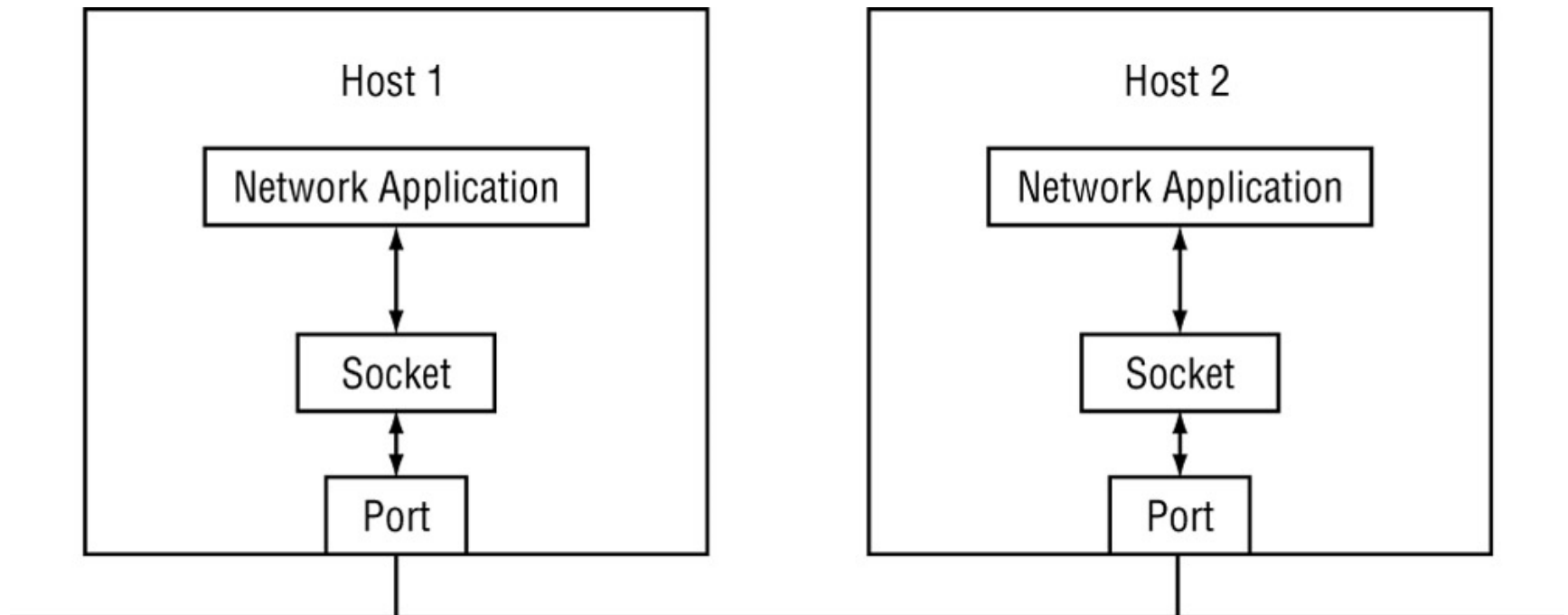
- In socket-based network programming, you do not directly access the network interface device to send and receive packets.
- Instead, an intermediary file descriptor is created to handle the programming interface to the network.
- operating system handles the details of determining which network interface device will be used to send out the data and how.



WHAT DID THE SOCKET DEFINE?

- A specific **communication domain**, such as a network.
- A specific **communication type**, such as stream or datagram
- A **specific protocol**, such as TCP or UDP

THE SOCKET INTERFACE





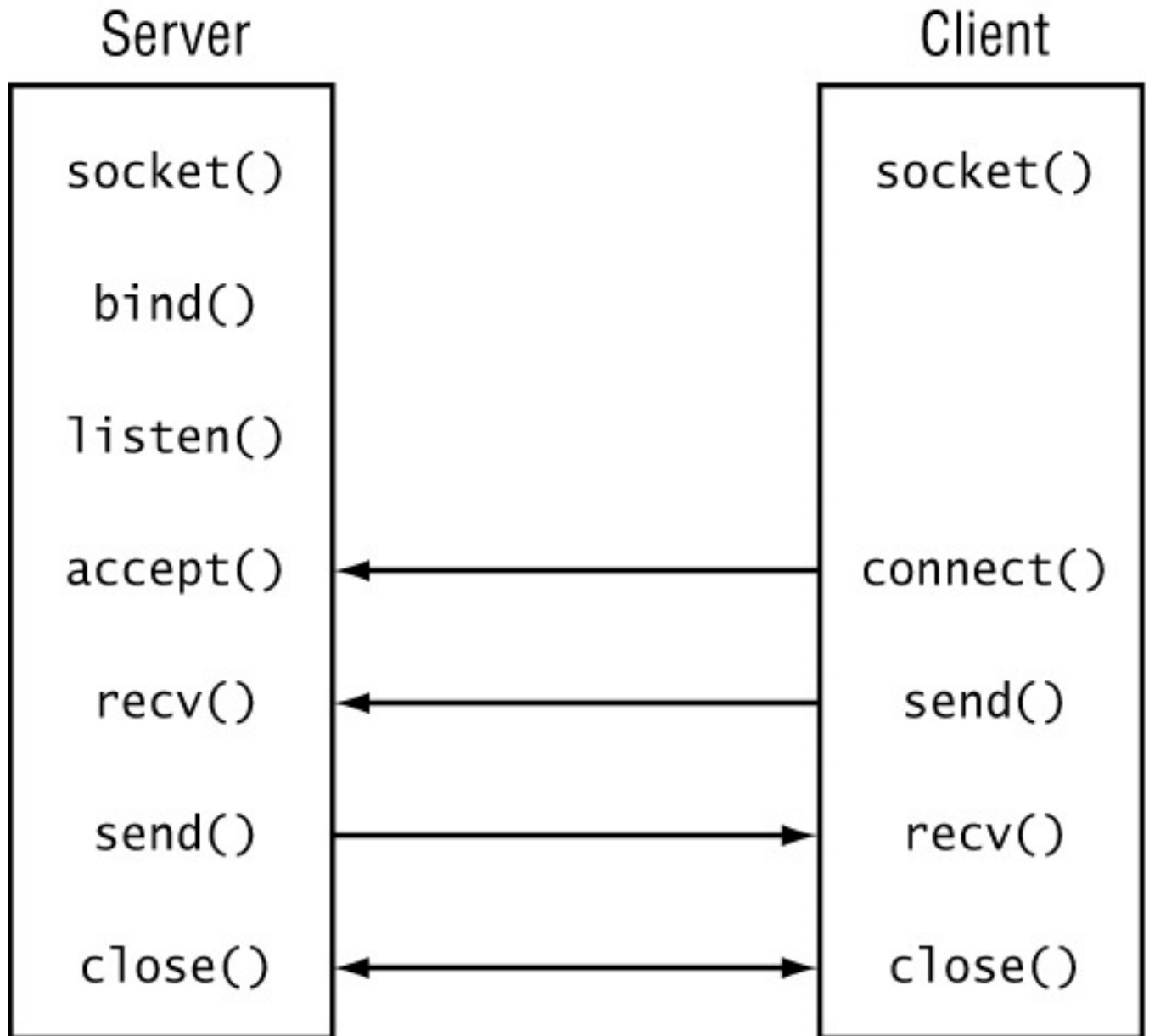
SOCKET TYPES

- **Connection-Oriented Sockets**
- **Connectionless Sockets**

CONNECTION-ORIENTED SOCKETS

- In a connection-oriented socket (stream socket) the TCP protocol is used to establish a session (connection) between two IP address endpoints.
- There is a fair amount of overhead involved with establishing the connection, but once it is established, data can be **reliably** transferred between the devices.
- To create a connection-oriented socket, separate sequences of functions must be used for server programs and for client programs (see the next slide)

STEPS



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

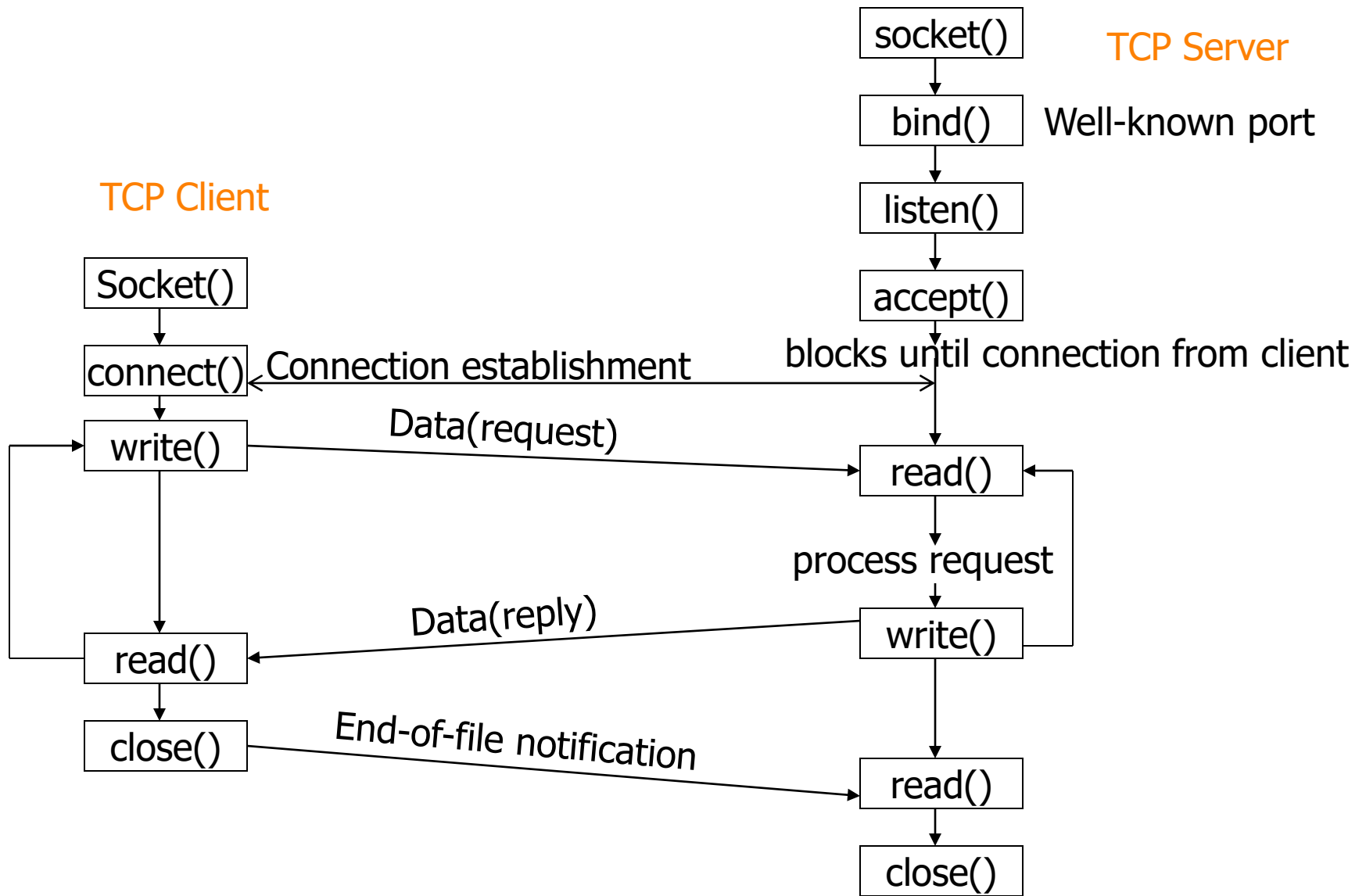
Create a socket

Setup the server address

Connect to the server

Read/write data

Shutdown connection



THE SERVER FUNCTIONS

- **For the server program**, the created socket must be bound to a local IP address and port number that will be used for the TCP communication.
- After the socket is bound to an address and port, the server program must be ready to accept connections from remote clients. This is a two-step process: first, the program looks for an incoming connection, next it sends and receives data.
- The program must first use the `listen()` function to "listen" to the network for an incoming connection. Next it must use the `accept()` function to accept connection attempts from clients
- After the `listen()` function, the `accept()` function must be called to wait for incoming connections. The format of the `accept()` function
- Once the connection has been accepted, the server can send and receive data from the client using the send and receive



PORTS

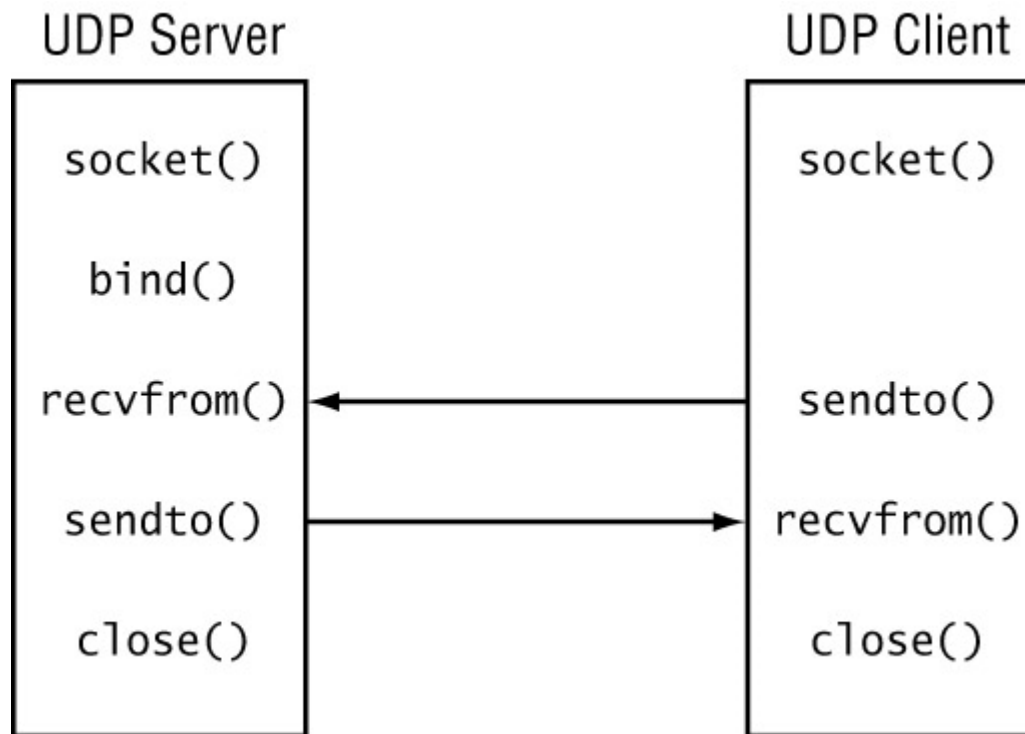
- Well Known Ports are in the range of 0 to 1023 only the operating system or an Administrator of the system can access
- Registered Ports are in the range of 1024 to 49151.
- Dynamic and/or Private Ports are those from 49152 through 65535 and are open for use without restriction



THE CLIENT FUNCTIONS

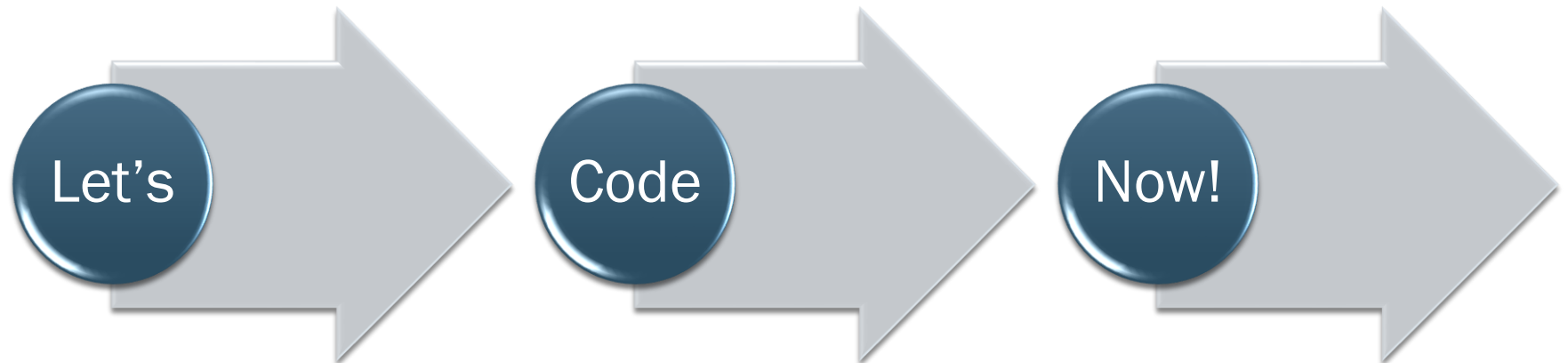
- In a connection-oriented socket, the client must bind to the specific host address and port for the application.
- For client programs, the `connect()` function is used instead of the `listen()` function:
- Once the `connect()` function succeeds, the client is connected to the server and can use the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

USING CONNECTIONLESS SOCKETS



USING CONNECTIONLESS SOCKETS

- sockets use the UDP protocol, no connection information is required to be sent between network devices.
- Because of this, it is often difficult to determine which device is acting as a “server”, and which is acting as a “client”.
- If a device is initially waiting for data from a remote device, the socket must be bound to a local address/port pair using the `bind()` function.
- Once this is done the device can send data out from the socket, or receive incoming data from the socket.
- Because the client device does not create a connection to a specific server address, the `connect()` function need not be used for the UDP client program





OUTLINE

- IP Addresses in C#
- Using C# Sockets
- C# Socket Exceptions



IP ADDRESSES IN C#

- .NET defines two classes in the System.Net namespace to handle various types of IP address information:
 - IPAddress
 - IPEndPoint

IPADDRESS

- used to represent a single IP address.
- This value can then be used in the various socket methods to represent the IP address.
- `public IPAddress(long address)`
- `IPAddress newaddress = IPAddress.Parse("192.168.1.1");`

TYPES OF IP ADDRESSES

- **Any** Used to represent any IP address available on the local system
 - Provides an IP address that indicates that the server must listen for client activity on **all network interfaces**. This field is read-only.
- **Broadcast** Used to represent the IP broadcast address for the local network
- **Loopback** Used to represent the loopback address of the system
- **None** Used to represent no network interface on the system
 - Don't listen for user activity

TRY

```
using System;
using System.Net;
class AddressSample
{
    public static void Main()
    {
        IPAddress test1 = IPAddress.Parse("192.168.1.1");
        IPAddress test2 = IPAddress.Loopback;
        IPAddress test3 = IPAddress.Broadcast;
        IPAddress test4 = IPAddress.Any;
        IPAddress test5 = IPAddress.None;
        IPHostEntry ihe = Dns.GetHostByName(Dns.GetHostName());
        IPAddress myself = ihe.AddressList[0];
    }
}
```

```
if (IPAddress.IsLoopback(test2))
    Console.WriteLine("The Loopback address is: {0}",
        test2.ToString());
else
    Console.WriteLine("Error obtaining the loopback
address");
    Console.WriteLine("The Local IP address is: {0}\n",
        myself.ToString());
    if (myself == test2)
        Console.WriteLine("The loopback address is the same as
local address.\n");
    else
        Console.WriteLine("The loopback address is not the local
address.\n");
    Console.WriteLine("The test address is: {0}",
        test1.ToString());
    Console.WriteLine("Broadcast address: {0}",
        test3.ToString());
    Console.WriteLine("The ANY address is: {0}",
        test4.ToString());
    Console.WriteLine("The NONE address is: {0}",
        test5.ToString());
}}
```



```
Microsoft Visual Studio Debug Console

The Loopback address is: 127.0.0.1
The Local IP address is: fe80::3c0c:effd:153e:6c52%14

The loopback address is not the local address.

The test address is: 192.168.1.1
Broadcast address: 255.255.255.255
The ANY address is: 0.0.0.0
The NONE address is: 255.255.255.255

C:\Users\Dr Ali\source\repos\ConsoleApp1\ConsoleApp1\bin\Debug\netcoreapp3.1\ConsoleApp1.exe (process 2380) exited with
code 0.
Press any key to close this window . . .
```

OUTPUT

IPENDPOINT

- Two constructors are used to create IPEndPoint instances:
 - IPEndPoint(long address, int port)
 - IPEndPoint(IPAddress address, int port)
- Both constructors use two parameters: an IP address value, represented as either a long value or an IPAddress object;
- and the integer port number.

```
using System;
using System.Net;
class IPEndPointSample {
    public static void Main() {
        IPAddress test1 = IPAddress.Parse("192.168.1.1");
        IPEndPoint ie = new IPEndPoint(test1, 8000);
        Console.WriteLine("The IPEndPoint is: {0}", ie.ToString());
        Console.WriteLine("The AddressFamily is: {0}",
ie.AddressFamily);
        Console.WriteLine("The address is: {0}, and the port is:
{1}\n", ie.Address, ie.Port);
        Console.WriteLine("The min port number is: {0}",
IPEndPoint.MinPort);
        Console.WriteLine("The max port number is: {0}\n",
IPEndPoint.MaxPort);
        ie.Port = 80;
        Console.WriteLine("The changed IPEndPoint value is: {0} ",
ie.ToString());
        SocketAddress sa = ie.Serialize();
        Console.WriteLine("The SocketAddress is: {0}",
sa.ToString());    } }
```

```
Microsoft Visual Studio Debug Console

The IPEndPoint is: 192.168.1.1:8000
The AddressFamily is: InterNetwork
The address is: 192.168.1.1, and the port is: 8000

The min port number is: 0
The max port number is: 65535

The changed IPEndPoint value is: 192.168.1.1:80
The SocketAddress is: InterNetwork:16:{0,80,192,168,1,1,0,0,0,0,0,0,0,0}

C:\Users\Dr Ali\source\repos\ConsoleApp1\ConsoleApp1\bin\Debug\netcoreapp3.1\ConsoleApp1.exe (process 2952) exited with
code 0.
Press any key to close this window . . .
```

OUTPUT

**LET'S USE
THE
SOCKETS**



USING C# SOCKETS

The Socket class constructor is as follows:

`Socket(AddressFamily af, SocketType st, ProtocolType pt)`

It uses three parameters to define the type of socket to create:

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

SocketType	Protocoltype	Description
Dgram	Udp	Connectionless communication
Stream	Tcp	Connection-oriented communication
Raw	Icmp	Internet Control Message Protocol
Raw	Raw	Plain IP packet communication

IP SOCKET DEFINITION COMBINATIONS

SOCKET PROPS.

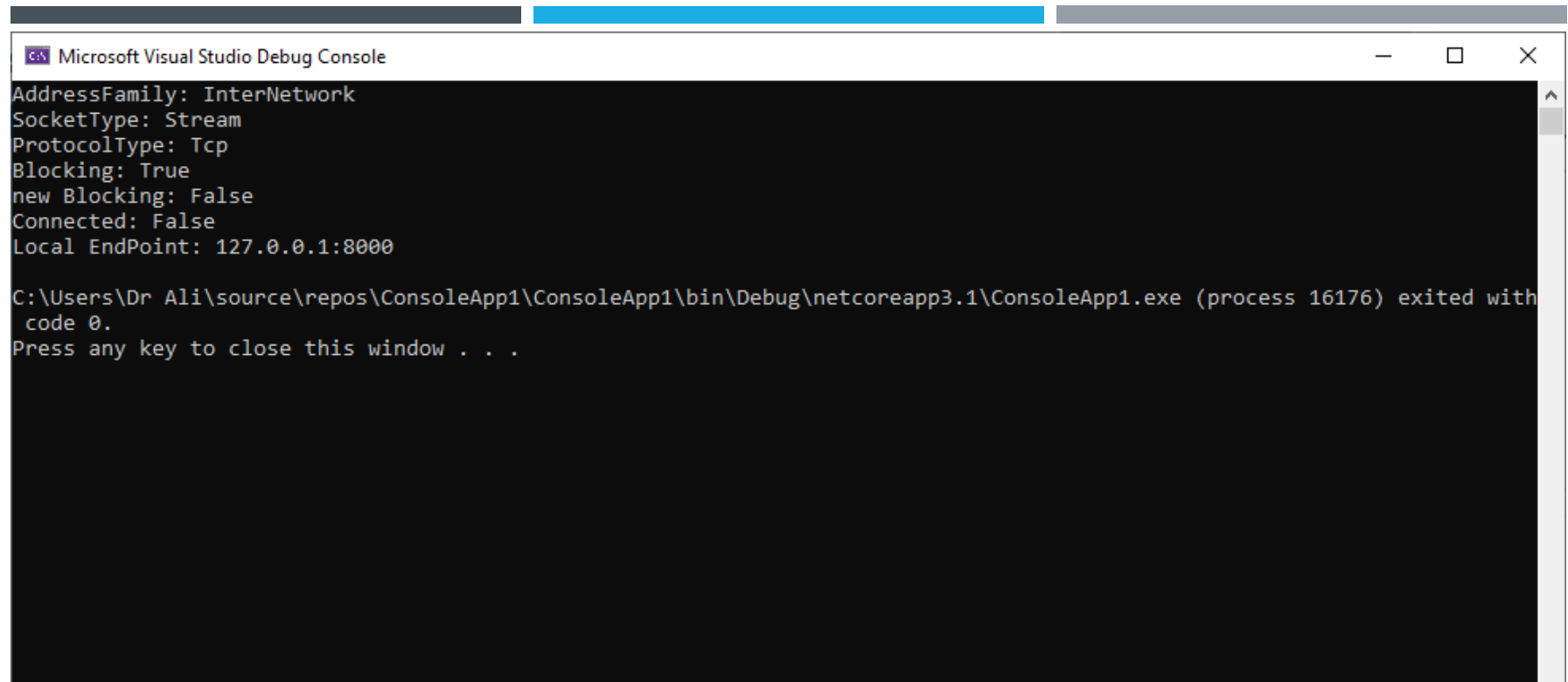
```
Socket newsock = Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
```

Socket Properties

Property	Description
AddressFamily	Gets the address family of the Socket
Available	Gets the amount of data that is ready to be read
Blocking	Gets or sets whether the Socket is in blocking mode
Connected	Gets a value that indicates if the Socket is connected to a remote device
Handle	Gets the operating system handle for the Socket
LocalEndPoint	Gets the local EndPoint object for the Socket
ProtocolType	Gets the protocol type of the Socket
RemoteEndPoint	Gets the remote EndPoint information for the Socket
SocketType	Gets the type of the Socket



CAN'T WAIT LET'S CODE !



The image shows a screenshot of the Microsoft Visual Studio Debug Console window. The window has a title bar with the Visual Studio logo and the text "Microsoft Visual Studio Debug Console". The main area is a black console with white text. The output shows network-related information: "AddressFamily: InterNetwork", "SocketType: Stream", "ProtocolType: Tcp", "Blocking: True", "new Blocking: False", "Connected: False", and "Local EndPoint: 127.0.0.1:8000". Below this, a message states: "C:\Users\Dr Ali\source\repos\ConsoleApp1\ConsoleApp1\bin\Debug\netcoreapp3.1\ConsoleApp1.exe (process 16176) exited with code 0." followed by "Press any key to close this window . . .".

```
Microsoft Visual Studio Debug Console

AddressFamily: InterNetwork
SocketType: Stream
ProtocolType: Tcp
Blocking: True
new Blocking: False
Connected: False
Local EndPoint: 127.0.0.1:8000

C:\Users\Dr Ali\source\repos\ConsoleApp1\ConsoleApp1\bin\Debug\netcoreapp3.1\ConsoleApp1.exe (process 16176) exited with
code 0.
Press any key to close this window . . .
```

OUTPUT

C# SOCKET EXCEPTIONS

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class SocketExcept
{
    public static void Main()
    {
        IPAddress host = IPAddress.Parse("195.168.1.1");
        IPEndPoint hostep = new IPEndPoint(host, 8000);
        Socket sock = new
Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);
        try{
            sock.Connect(hostep);
        }
        catch (SocketException e){
            Console.WriteLine("Problem connecting to host");
            Console.WriteLine(e.ToString());
            sock.Close();
            return;
        }
    }
}
```

```
try{
    sock.Send(Encoding.ASCII.GetBytes("testing"));
}
catch (SocketException e){
    Console.WriteLine("Problem sending data");
    Console.WriteLine(e.ToString());
    sock.Close();
    return;
}
sock.Close();
}
```

```
Microsoft Visual Studio Debug Console

Problem connecting to host
System.Net.Internals.SocketExceptionFactory+ExtendedSocketException (10061): No connection could be made because the target machine actively refused it. 195.168.1.1:8000
    at System.Net.Sockets.Socket.DoConnect(EndPoint endPointSnapshot, SocketAddress socketAddress)
    at System.Net.Sockets.Socket.Connect(EndPoint remoteEP)
    at SocketExcept.Main() in C:\Users\Dr Ali\source\repos\ConsoleApp1\ConsoleApp1\Program.cs:line 13

C:\Users\Dr Ali\source\repos\ConsoleApp1\ConsoleApp1\bin\Debug\netcoreapp3.1\ConsoleApp1.exe (process 15608) exited with code 0.
Press any key to close this window . . .
```

OUTPUT

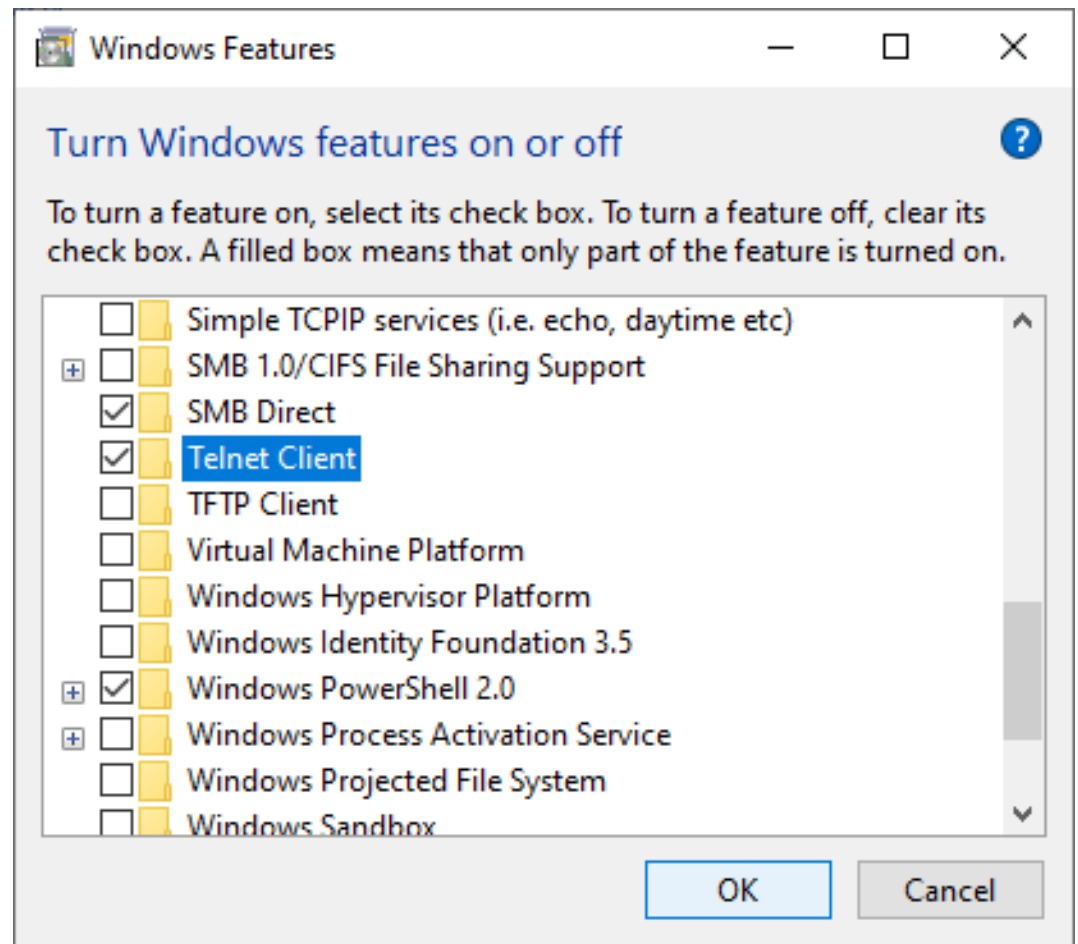
ECHO SERVER

```
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();
}
}
```

ENABLE TELNET CLIENT

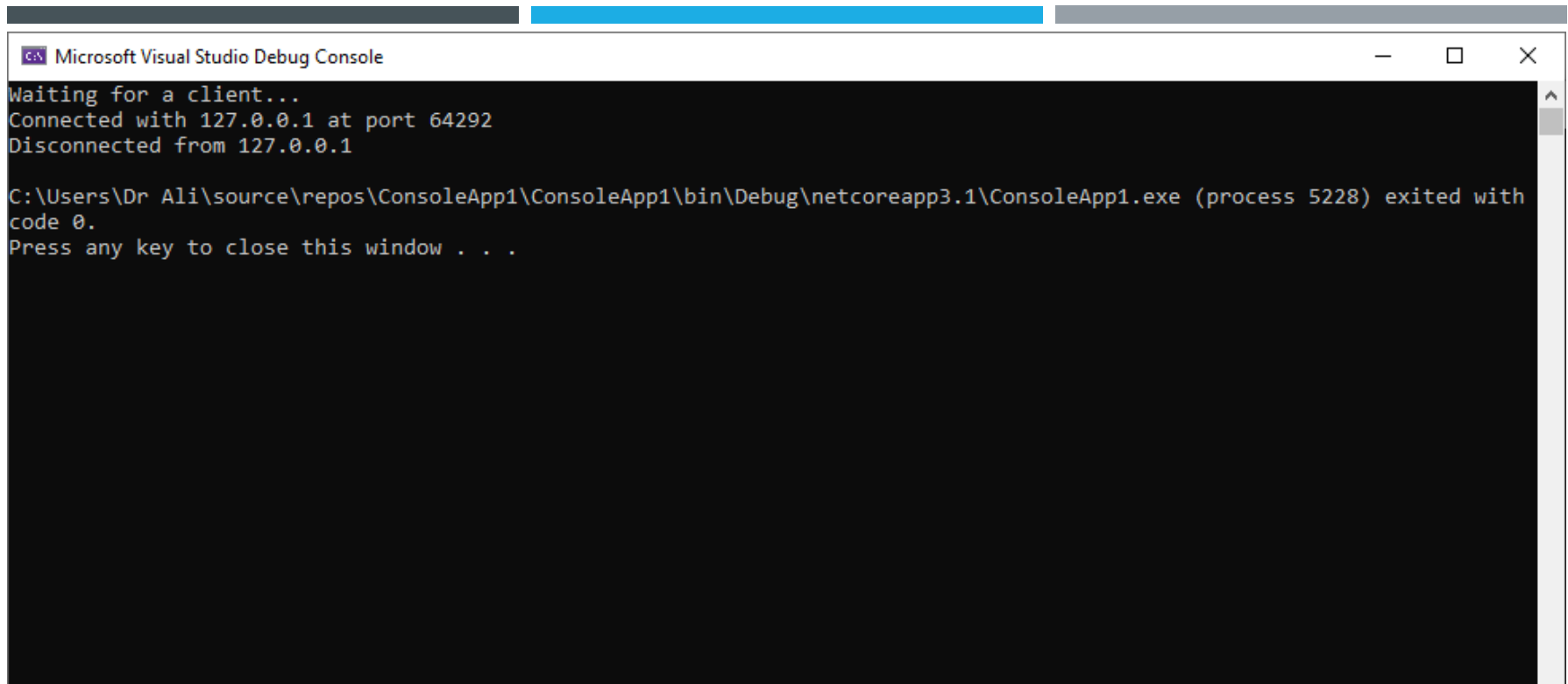
- How to Install Telnet in Windows 10
- Open “Control Panel”.
- Open “Programs”.
- Select the “Turn Windows features on or off ” option.
- Check the “Telnet Client” box.
- Click “OK“. A box will appear that says “Windows features” and “Searching for required files“. When complete, the Telnet client should be installed in Windows.





TELNET UR SERVER

- Open the CMD
- C:\>telnet 127.0.0.1 9050



The image shows a screenshot of the Microsoft Visual Studio Debug Console window. The window has a title bar with the Visual Studio logo and the text "Microsoft Visual Studio Debug Console". The main area is a black console with white text. The text shows the application waiting for a client, connecting to 127.0.0.1 at port 64292, and then disconnecting. It then shows the application path and that it exited with code 0. The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

```
Microsoft Visual Studio Debug Console
Waiting for a client...
Connected with 127.0.0.1 at port 64292
Disconnected from 127.0.0.1

C:\Users\Dr Ali\source\repos\ConsoleApp1\ConsoleApp1\bin\Debug\netcoreapp3.1\ConsoleApp1.exe (process 5228) exited with
code 0.
Press any key to close this window . . .
```

OUTPUT



THANK U

NEXT TOPICS

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