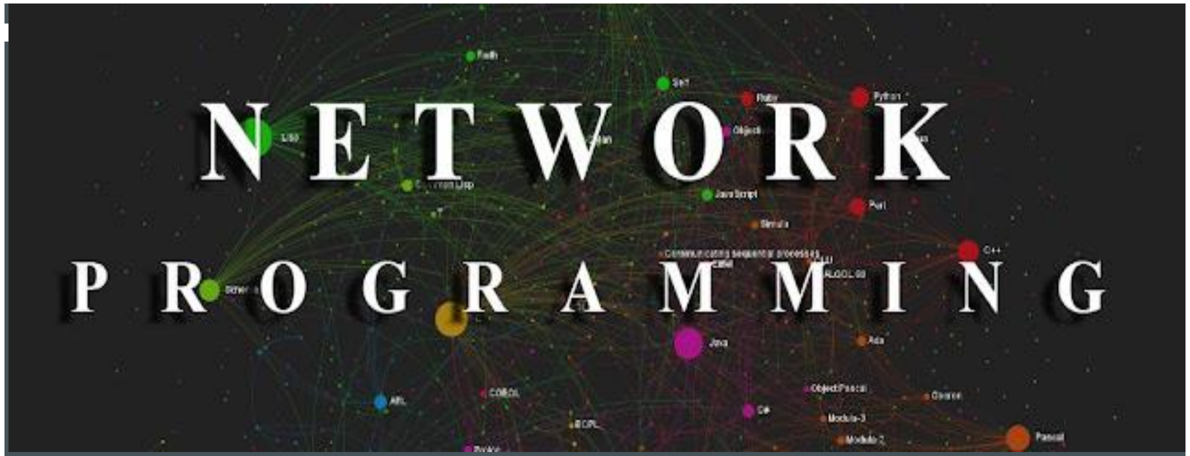


IT432 - NETWORK PROGRAMMING

DR. ALI HUSSEIN AHMED

ALI.HUSSEIN@AUN.EDU.EG

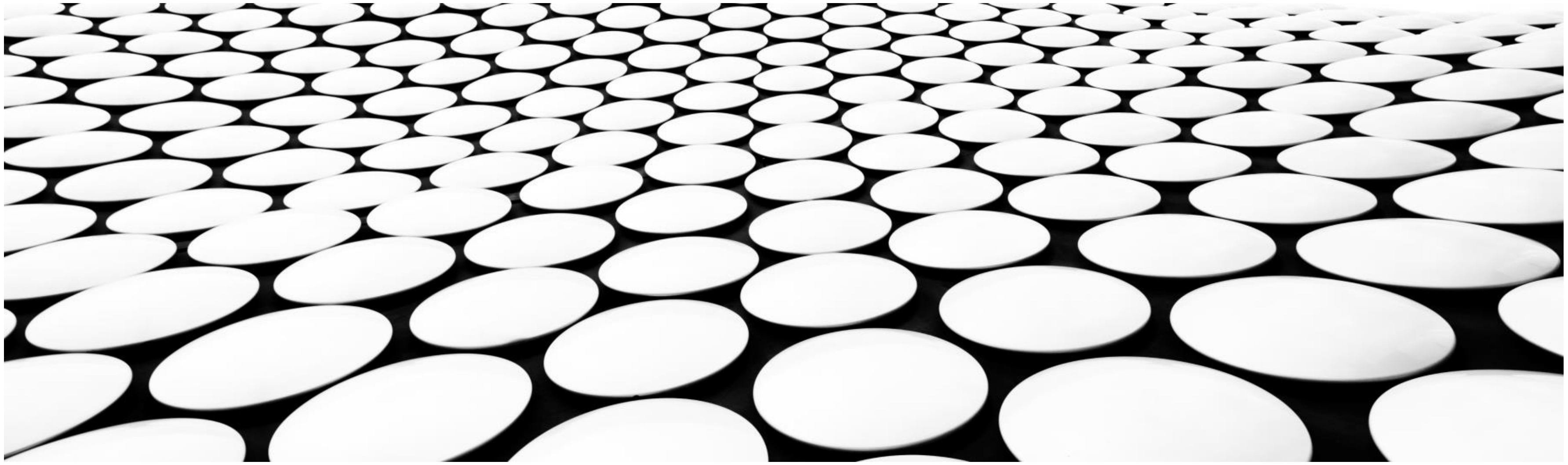
IT DEPT - OFFICE NO. 312

2022

كلية معتمدة من الهيئة القومية
لضمان الجودة والاعتماد

LECTURE 7

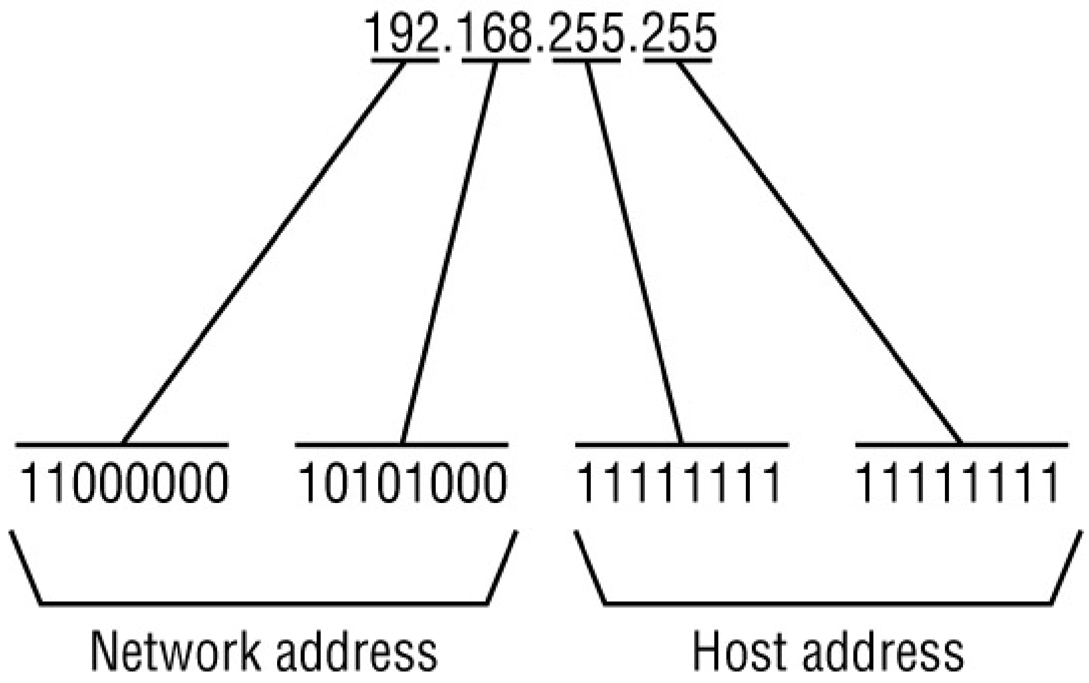
BROADCASTING



WHAT IS BROADCAST?

- To send a single packet of information that can be accessible by every device on the network.
- Two types: local and global
- The main goal is to limit the broadcasting network to developers' network and other are unaffected.
- Because TCP communication requires that two devices have a dedicated connection, it is not possible to send broadcast messages in a strictly TCP environment.
- Instead, UDP packets must be used because that protocol has the capability of sending messages without a specific connection being defined

LOCAL VERSUS GLOBAL BROADCASTS



The idea is to localize a broadcast message so that other networks are not affected by the broadcast.

SENDING BROADCAST PACKETS (THE BAD BROADCAST PROGRAM)

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class BadBroadcast
{
  public static void Main()
  {
   Socket sock = new Socket(AddressFamily.InterNetwork, SocketType.Dgram,
               ProtocolType.Udp);
   IPEndPoint iep = new IPEndPoint(IPAddress.Broadcast, 9050);
   byte[] data = Encoding.ASCII.GetBytes("This is a test message");
   sock.SendTo(data, iep);
   sock.Close();
  }
}
```

GUESS THE RESULT !!

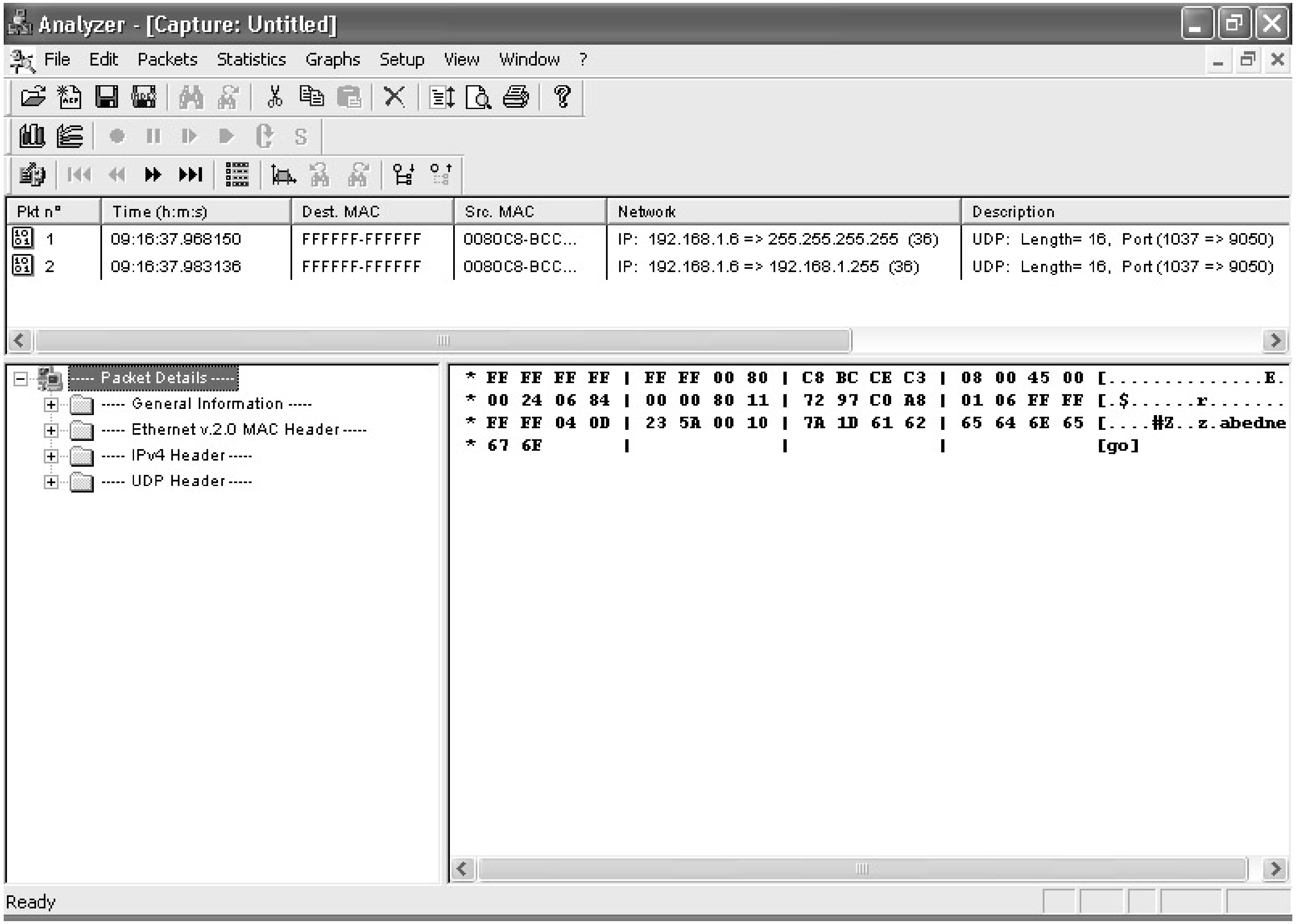
- **Unhandled Exception: System.Net.Sockets.SocketException: An attempt was made to access a socket in a way forbidden by its access permissions at System.Net.Sockets.Socket.SendTo(Byte[] buffer, Int32 offset, Int32 size, SocketFlags socketFlags, EndPoint remoteEP) at System.Net.Sockets.Socket.SendTo(Byte[] buffer, EndPoint remoteEP) at BadBroadcast.Main()**
- If you attempt to send a broadcast message from a default UDP socket, you will get a SocketException error.
- The broadcast socket option must be set on the created socket using the **SetSocketOption()** method of the Socket class
- sock.SetSocketOption(SocketOptionLevel.Socket, SocketOptionName.Broadcast, 1);

THE GOOD BROADCAST.CS

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class Broadcst{
  public static void Main()  {
   Socket sock = new Socket(AddressFamily.InterNetwork,
SocketType.Dgram,
                ProtocolType.Udp);
   IPEndPoint iep1 = new IPEndPoint(IPAddress.Broadcast, 9050);
   IPEndPoint iep2 = new
IPEndPoint(IPAddress.Parse("192.168.1.255"), 9050);
   string hostname = Dns.GetHostName();
   byte[] data = Encoding.ASCII.GetBytes(hostname);
   sock.SetSocketOption(SocketOptionLevel.Socket,
              SocketOptionName.Broadcast, 1);
   sock.SendTo(data, iep1);
   sock.SendTo(data, iep2);
   sock.Close();  } }
```

HOW TO RECEIVE THE BROADCAST MESSAGES?

Method 1


Analyzer - [Capture: Untitled]
File Edit Packets Statistics Graphs Setup View Window ?

Pkt n°	Time (h:m:s)	Dest. MAC	Src. MAC	Network	Description
1	09:16:37.968150	FFFFFF-FFFFFF	0080C8-BCC...	IP: 192.168.1.6 => 255.255.255.255 (36)	UDP: Length= 16, Port (1037 => 9050)
2	09:16:37.983136	FFFFFF-FFFFFF	0080C8-BCC...	IP: 192.168.1.6 => 192.168.1.255 (36)	UDP: Length= 16, Port (1037 => 9050)

----- Packet Details -----
- ----- General Information -----
- ----- Ethernet v.2.0 MAC Header -----
- ----- IPv4 Header -----
- ----- UDP Header -----

```
* FF FF FF FF | FF FF 00 80 | C8 BC CE C3 | 08 00 45 00 [..............E.
* 00 24 06 84 | 00 00 80 11 | 72 97 C0 A8 | 01 06 FF FF [.$......r.......
* FF FF 04 0D | 23 5A 00 10 | 7A 1D 61 62 | 65 64 6E 65 [....#Z..z.abedne
* 67 6F       |             |             |             [go]
```

Ready

LET'S CODE THE RECEIVER

Method 2

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class RecvBroadcst {
  public static void Main()   {
   Socket sock = new Socket(AddressFamily.InterNetwork,SocketType.Dgram, ProtocolType.Udp);
   IPEndPoint iep = new IPEndPoint(IPAddress.Any, 9050);
   sock.Bind(iep);
   EndPoint ep = (EndPoint)iep;
   Console.WriteLine("Ready to receive…");
   byte[] data = new byte[1024];
   int recv = sock.ReceiveFrom(data, ref ep);
   string stringData = Encoding.ASCII.GetString(data, 0, recv);
   Console.WriteLine("received: {0} from: {1}",stringData, ep.ToString());
   data = new byte[1024];
   recv = sock.ReceiveFrom(data, ref ep);
   stringData = Encoding.ASCII.GetString(data, 0, recv);
   Console.WriteLine("received: {0} from: {1}", stringData, ep.ToString());
   sock.Close();   } }
```

USING BROADCAST PACKETS TO ADVERTISE A SERVER

- One of the most common uses for broadcast packets is to advertise the presence of a server on the network.
- By repeatedly sending out a broadcast packet containing information regarding the server at a predetermined time interval, you make it possible for clients to easily detect the presence of the server on the local subnet.
- This technique is used for many types of applications, from Windows application servers to network printers.

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;
class Advertise {
  public static void Main()
  {
   Advertise server = new Advertise();
  }
   public Advertise()  {
   Thread advert = new Thread(new ThreadStart(sendPackets));
   advert.IsBackground = true;
   advert.Start();
   Console.Write("Press Enter to stop");
   string data = Console.ReadLine();
  }
void sendPackets()
  {
   Socket sock = new Socket(AddressFamily.InterNetwork,
           SocketType.Dgram, ProtocolType.Udp);
   sock.SetSocketOption(SocketOptionLevel.Socket,
              SocketOptionName.Broadcast, 1);
   IPEndPoint iep = new IPEndPoint(IPAddress.Broadcast, 9050);
   string hostname = Dns.GetHostName();
   byte[] data = Encoding.ASCII.GetBytes(hostname);
   while (true)
   {
     sock.SendTo(data, iep);
     Thread.Sleep(60000);
   }
  }
}
```

- The Advertise.cs program creates a new Thread object using the sendPackets() method, which generates a Socket object for the broadcast address and broadcasts the hostname every 60 seconds.
- The Thread.Sleep() method puts the thread in idle mode between broadcasts.
- The broadcast thread is placed in background mode by means of the IsBackground property of the Thread class.
  - Foreground threads are those threads that keep running even after the application exits or quits. ...
  - Background Threads are those threads that will quit if our main application quits.

- Did you tried to run server and got and error indicating that the port is already used by another process? Justify the matter taking into issues of foreground threads.
  - The answer is that there are a thread (foreground) still running on a port (even if your program is closed)
- Because the main part of the application does nothing but create the background thread, it must wait, or the background thread will be terminated.
- This is set up using the Console.ReadLine() method to wait for input from the console before terminating the main program.

- A TcpChat program may have one major drawback: for it to communicate properly, you had to manually compile the address of the remote host into the program—obviously not a good way to do business in a production application.
- With broadcast messages at your disposal, you can fix the TcpChat program to be more user-friendly.
- There are two pieces that need to be added to the original TcpChat program to accomplish this goal:
  - A background thread that broadcasts an identification message on the subnet
  - A way to listen for broadcast messages and detect another chat application running on the subnet
- As seen in the Advertise.cs program, the first part is easy; you just create a background thread using the Thread class to broadcast a message using a UDP port that's separate from the one used for the chat communication.
- The second piece—the listener—is a little trickier to implement.

- First, the program must listen for broadcast messages from possible chat clients on the subnet. This can be done by another background thread that listens for broadcasts on the network.
- As broadcast messages from other chat clients are detected, their information will be listed for the customer's selection.
- A problem with creating this list is that each client is sending its information out every minute.
- You cannot simply add any message detected to the list, or a single chat client will appear multiple times. A production-quality application must keep track of each client added to the list and not duplicate it.
- For this simple example, you can handle that issue by having the program get only the first advertising chat program it sees and place its information in the ListBox.
- The second part of this process is to enable the customer's selection of the desired client information from the list box (which for this example will be only one entry).
- Then the customer needs to click the Connect button to chat with the selected client. The hostname and address for the remote client are taken from the information gathered from the broadcast packet, and the chat connection is created

AN EXERCISE

- Design a multithreaded chat server that allows the following features
  - Clients text-chat to each other
  - Client send images to each other ✓
  - Your server will log all the messages sent between clients
  - Your server will broadcast its ip address and port number. ✓
  - Features will be added incrementally during this course
  - Complete the code in the course teams
  - [Server]()
  - [Client]()

THANK U