EXPERIMENT NO.1

TITLE: Implementation and Demonstration of LAN

OBJECTIVE: Demonstrating client-server and peer to peer mode of configuration

AIM: Implementation of LAN using suitable multiuser Windows operating System and demonstrating client-server and peer to peer mode of configuration.

SOFTWARE USED: command Prompt, Cisco packet Tracer 6.2

Theory:

A LAN is a computer network that consists of access points, cables, routers, and switches that enabledevices to connect to web servers and internal servers within a single building, campus, or home network, and to other LANs via Wide Area Networks (WAN) or Metropolitan Area Network (MAN). Devices on a LAN, typically personal computers and workstations, can share files and be accessed by each other over a single Internet connection.
A router assigns IP addresses to each device on the network and facilitates a shared Internet connection between all the connected devices. A network switch connects to the router and facilitates communication between connected devices, but does not handle Local Area Network IP configuration or sharing Internet connections. Switches are ideal tools for increasing the number of LAN ports available on the network.

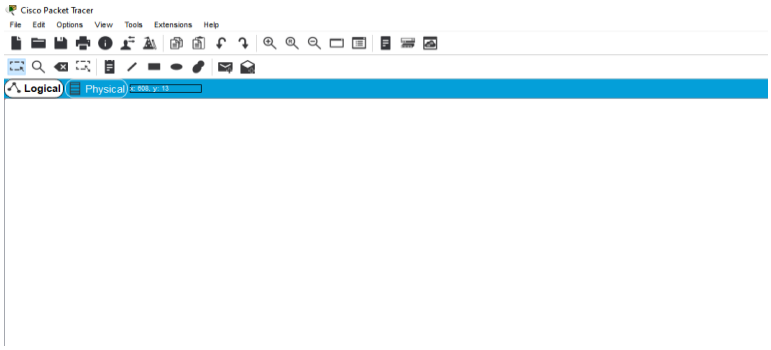
PART 1

Procedure: Peer to peer connection

Follow the below steps to initiate the setup for the connection:

Step 1: Download Cisco Packet Tracer.
Step 2: Run and install the setup (You can be requested to log in to your Cisco NetworkingAcademy Account or you can also log in as a guest).
Step 3: After the installation procedure has completed this display (below) will appear when you run the Cisco Packet Tracer-Start the

application.



Cisco Packet Tracer

Implementation:

Follow the below steps to implement the connection:

Step 1: From the bottom toolbar, click on **'End Devices'** and select **'PC'** and then click on thescreen (for two PC's do this step twice).

Bottom toolbar->End devices->PC

This is how it will appear on the screen

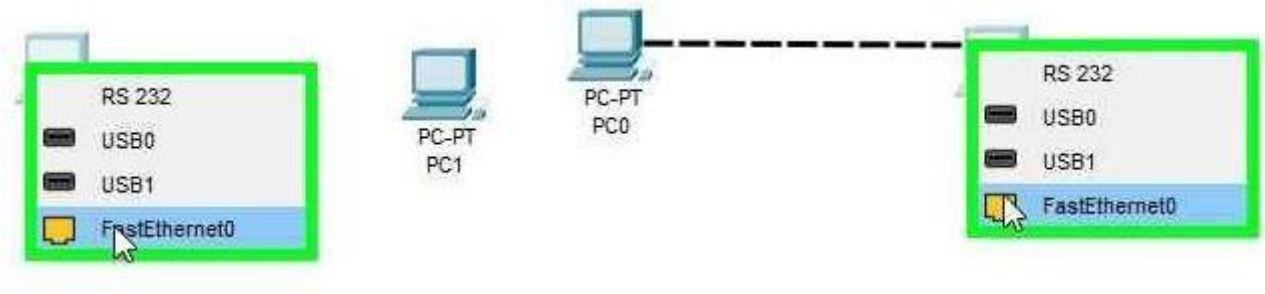
Step 2: Now to connect the PC's, we require a wire; we use cross-over wire to connect similardevices.

Select Connections from the bottom toolbar, and select cross-over wire (that is the fourth wire).

A **Cross-Over Wire** is largely used to connect the computing gadgets, additionally, cross wirecables are used to connect devices of equal type.

Step 3: After selecting the wire click on the computer on the screen(here PC0) and selectFastEthernet0.

Then, drag the wire to the other pc (here PC1) and do the same.

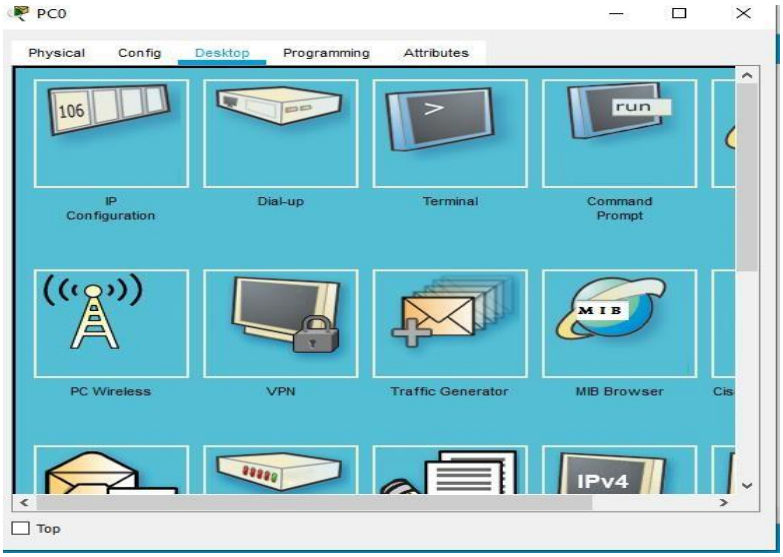


Step 4: Now, we will assign the IP address to both the PCs (PC0 & PC1).

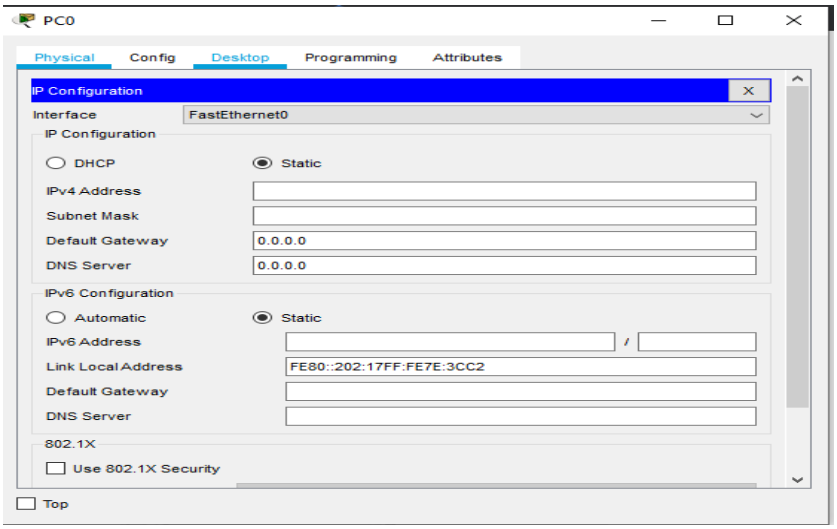
An **IP address** (Internet Protocol) is nothing but the numerical designation of the devicesconnected to the network that uses the Internet protocol as a communication medium.

Click on PC0. A dialog box will appear on the screen, select Desktop and

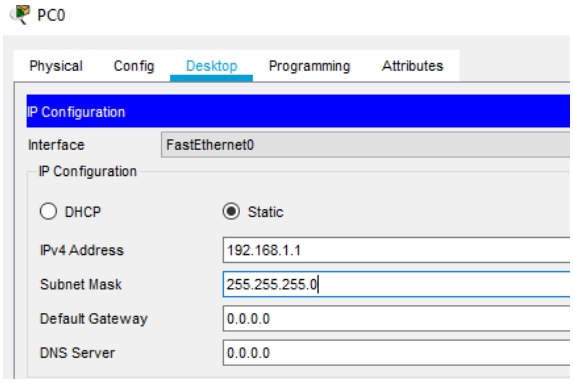
then select IPconfiguration:



After clicking on IP configuration this is what will appear



Now in IPv4 Address, write 192.168.1.1, Subnet mask will be 255.255.2550



Similarly, assign 192.168.1.2 to PC1

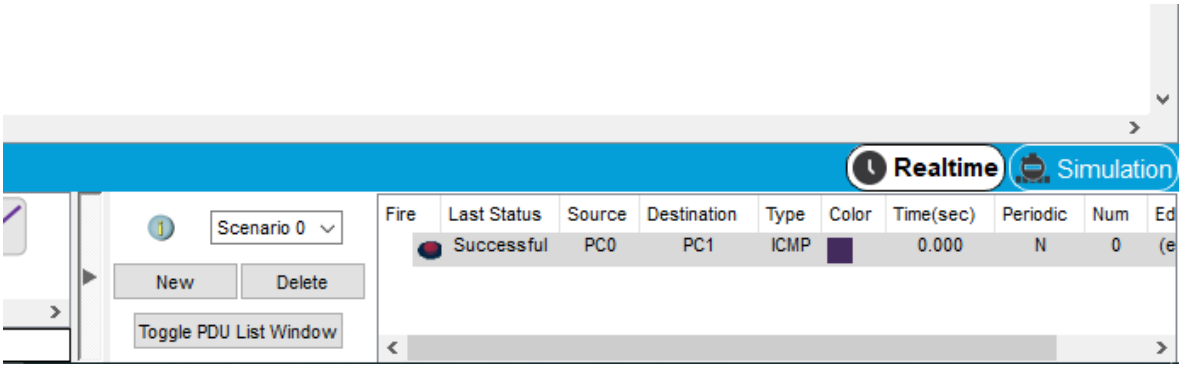
We have successfully connected two computers.

Now to check this, we will transfer data from one computer to another and check whether thetransfer is successful or not. To do so follow the below steps:

Step 1: From the Secondary Toolbar at the top, select 'Add sample PDU' that is the second lasticon.

Step 2: Now click on PC0 and then PC1.

Now if in Realtime box- PDU list window it shows successful, that means all the connections arecorrect and the data transfer is successful



You have successfully connected two computers(peer to peer) , using the virtual program CiscoPacket Tracer.

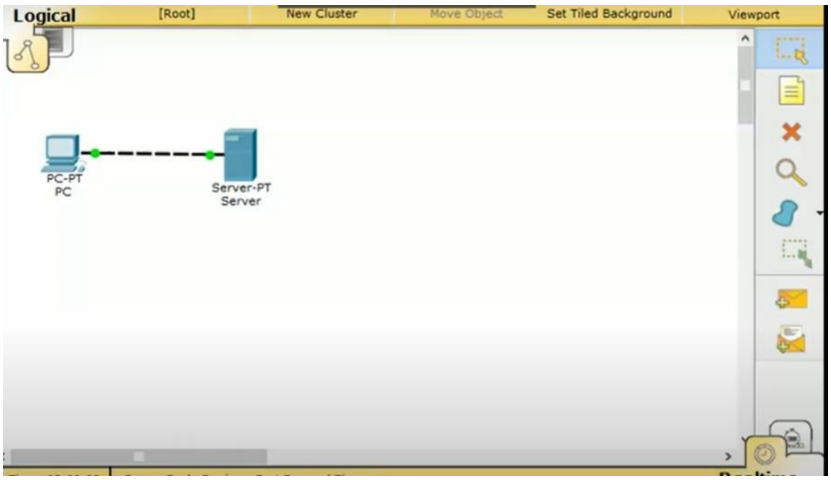
PART 2:

Simple server-client configuration

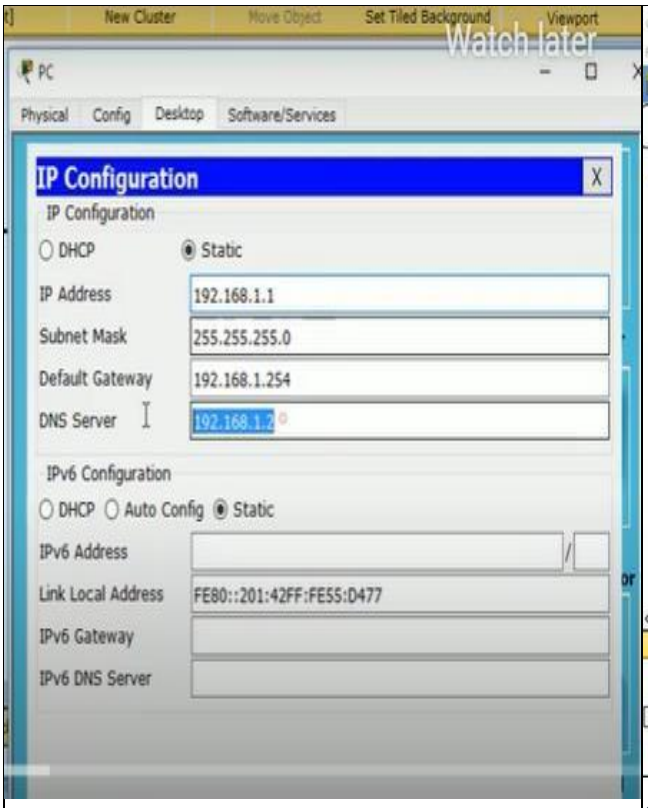
In a small network generally, there stays one server and one and more clients. Desktop PCs orLaptop PCs can be used as client. The client generally request service from the server and server serves that request as per requested. The server can be different types like web server, mail server, DNS server, File

transfer server or file server, or file server etc. types. Here In this we'll just learn about the simple client and server network by building them and analyze the client server architecture

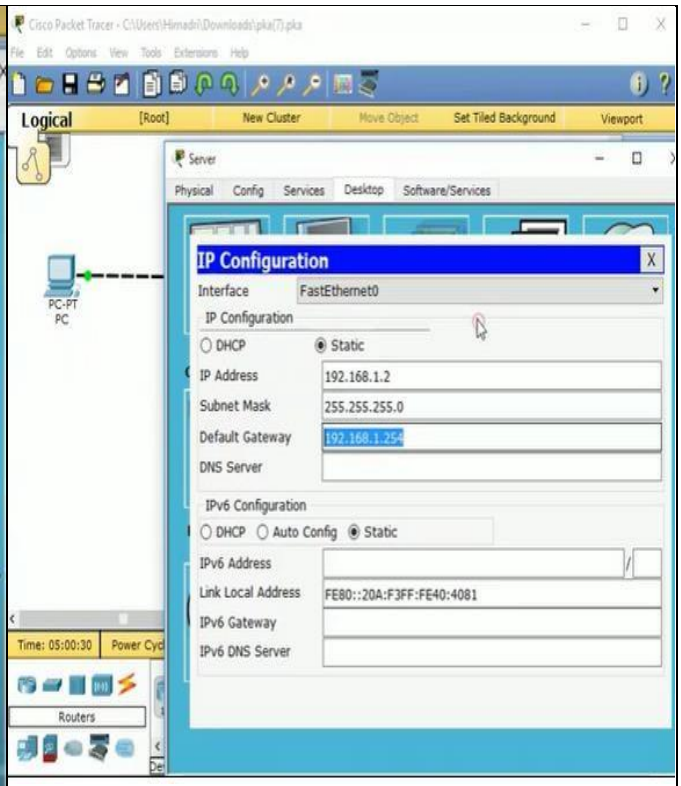
Step 1 : Open the Packet tracer Drag a PC(Desktop/Laptop) and Server. Connect them usingproper cable. Generally, the connecting cable is used as copper cross-over cable.



Step 2 : After that implement IP address in both of them.For PC you can enter the following IP address:
IP address: 192.168.1.1
Subnet mask: 255.255.255.0
Default Gateway: 192.168.1.254
DNS server: 192.168.1.2
Now for server you we here we've used this configuration:IP address: 192.168.1.2
Subnet mask: 255.255.255.0
Default gateway: 192.168.1.254



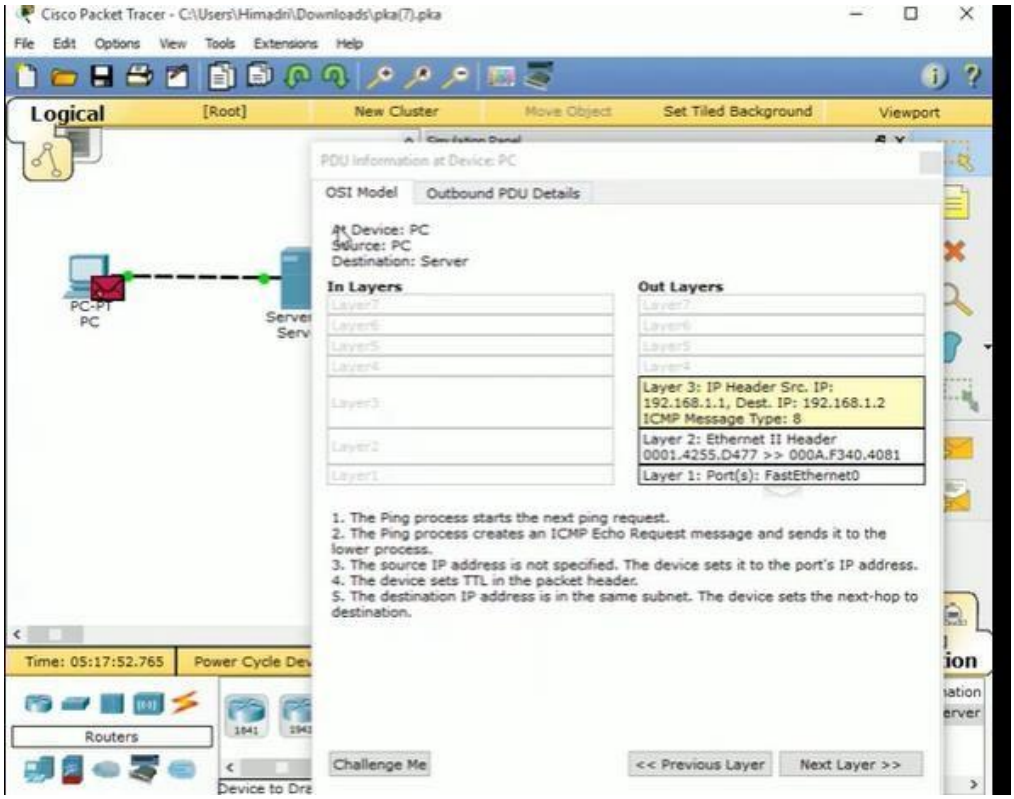
PC

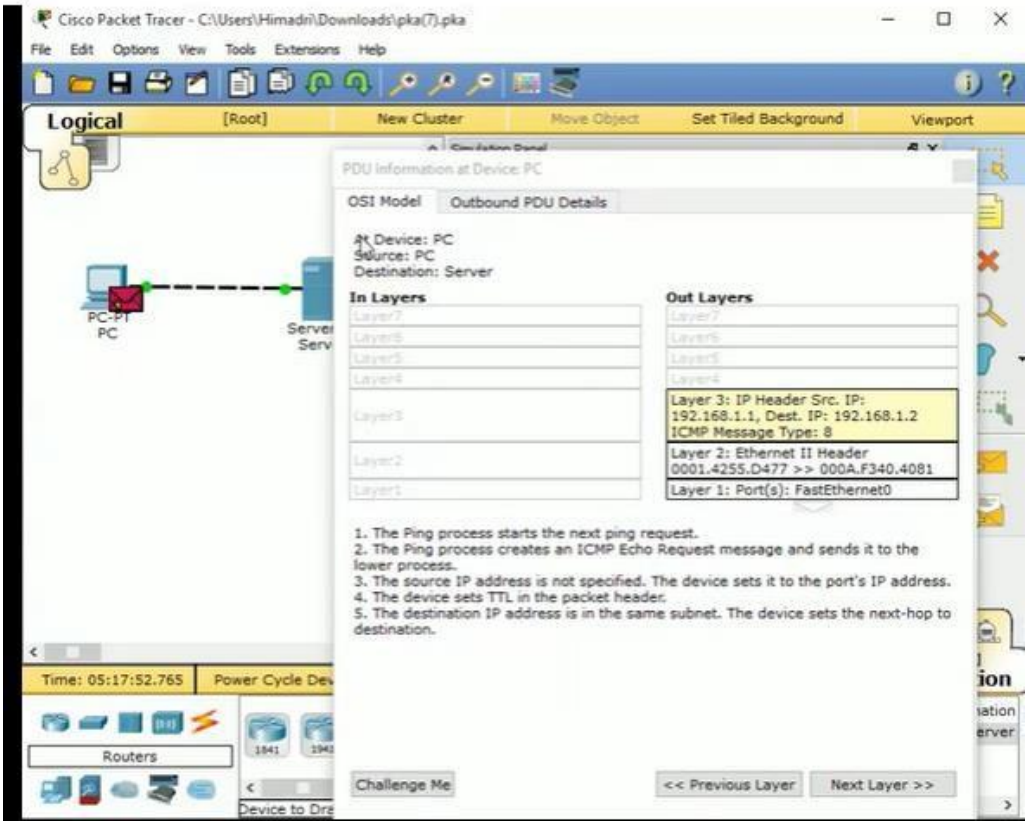


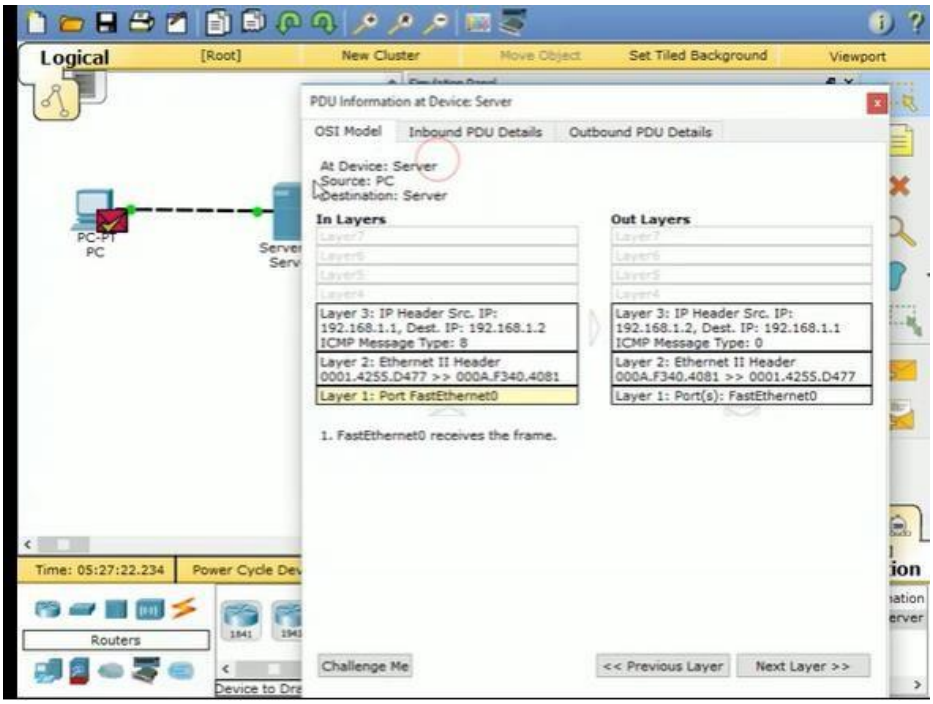
Server

Now just take a message envelope (close message envelope) and check the connection, by placing the message in two PC. If there is successful message shown the right bottom of the PC, then you can say that, the connection is OK. Also, you can ping IP address from PC to check the connection.

Now we'll look for the packet at the real time simulation mode. It is necessary to realize the client server network architecture. Just click the "Simulation", which stays by the side of "Realtime". After the click the option Auto Capture / Play. Now through this you cab be able to see the packet traveling path of this network server.


Cisco Packet Tracer - C:\Users\Himadri\Downloads\pka(7).pka
File Edit Options View Tools Extensions Help
Logical
[Root]
New Cluster
Move Object
Set Tiled Background
Viewport
PDU Information at Device: PC
OSI Model
Outbound PDU Details
At Device: PC
Source: PC
Destination: Server
In Layers
Out Layers
Layer 3: IP Header Src. IP: 192.168.1.1, Dest. IP: 192.168.1.2 ICMP Message Type: 8
Layer 2: Ethernet II Header 0001.4255.D477 >> 000A.F340.4081
Layer 1: Port(s): FastEthernet0
1. The Ping process starts the next ping request.
2. The Ping process creates an ICMP Echo Request message and sends it to the lower process.
3. The source IP address is not specified. The device sets it to the port's IP address.
4. The device sets TTL in the packet header.
5. The destination IP address is in the same subnet. The device sets the next-hop to destination.
Challenge Me
<< Previous Layer
Next Layer >>
Time: 05:17:52.765
Power Cycle Dev
Routers
PC-PT PC


Cisco Packet Tracer - C:\Users\Himadri\Downloads\pka(7).pka
File Edit Options View Tools Extensions Help
Logical
[Root]
New Cluster
Move Object
Set Tiled Background
Viewport
PDU Information at Device: PC
OSI Model
Outbound PDU Details
At Device: PC
Source: PC
Destination: Server
In Layers
Out Layers
Layer 3: IP Header Src. IP: 192.168.1.1, Dest. IP: 192.168.1.2 ICMP Message Type: 8
Layer 2: Ethernet II Header 0001.4255.D477 >> 000A.F340.4081
Layer 1: Port(s): FastEthernet0
1. The Ping process starts the next ping request.
2. The Ping process creates an ICMP Echo Request message and sends it to the lower process.
3. The source IP address is not specified. The device sets it to the port's IP address.
4. The device sets TTL in the packet header.
5. The destination IP address is in the same subnet. The device sets the next-hop to destination.
Challenge Me
<< Previous Layer
Next Layer >>
Time: 05:17:52.765
Power Cycle Dev
Routers
PC-PT PC


Logical
[Root]
New Cluster
Move Object
Set Tiled Background
Viewport
PDU Information at Device: Server
OSI Model
Inbound PDU Details
Outbound PDU Details
At Device: Server
Source: PC
Destination: Server
In Layers
Out Layers
Layer 3: IP Header Src. IP: 192.168.1.1, Dest. IP: 192.168.1.2 ICMP Message Type: 8
Layer 2: Ethernet II Header 0001.4255.D477 >> 000A.F340.4081
Layer 1: Port FastEthernet0
Layer 3: IP Header Src. IP: 192.168.1.2, Dest. IP: 192.168.1.1 ICMP Message Type: 0
Layer 2: Ethernet II Header 000A.F340.4081 >> 0001.4255.D477
Layer 1: Port(s): FastEthernet0
1. FastEthernet0 receives the frame.
Challenge Me
<< Previous Layer
Next Layer >>
Time: 05:27:22.234
Routers

CONCLUSION:

EXPERIMENT NO.2

TITLE: Simulating various Networks (LAN, WAN).

OBJECTIVE: To Simulate and test basic command line operation with Linux operating system andnetwork configuration, testing and verification.

AIM: To test Networking commands a) Ping b) ipconfig / ifconfig c) Host name d) Netstat f) Tracert /Traceroute / Tracepath g) NSlookup.

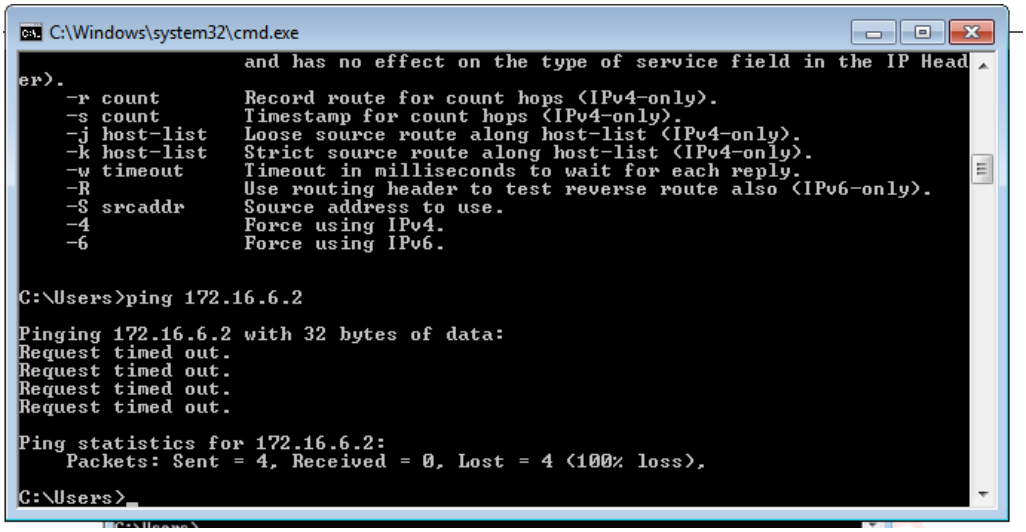
SOFTWARE USED: command Prompt, Cisco packet Tracer 6.2

Theory:

1. Ping

The ping command sends an echo request to a host available on the network. Using this command, you can check if your remote host is responding well or not. Tracking and isolating hardware and software problems. Determining the status of the network and various foreign hosts. The ping command is usually used as a simple way to verify that a computer can communicate over the network with another computer or network device. The ping command operates by sending Internet Control Message Protocol (ICMP) Echo Request messages to the destination computer and waiting for a response

ping172.16.6.2



2. IPConfig

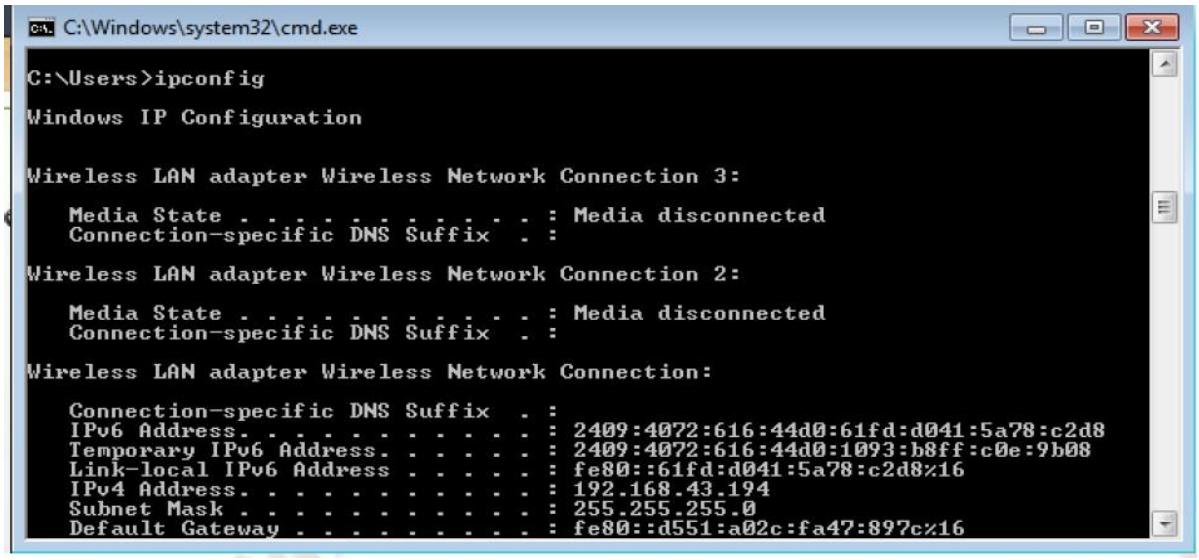
The IPConfig command is one of the more useful basic Windows network commands everyone should know and use to troubleshoot problems. The IPConfig command displays basic IP address configuration information for the Windows device you are working on. In fact, the command will display information for every network adapter that has ever been installed on your Windows 10 computer.

To run the basic command, at the prompt type:

ipconfig

The general information includes IP Addresses for both IPv4 and IPv6, the Default Gateway, and theSubnet Mask. Adding the parameter /all to the command will display DNS Server information and details concerning IP Address leases.

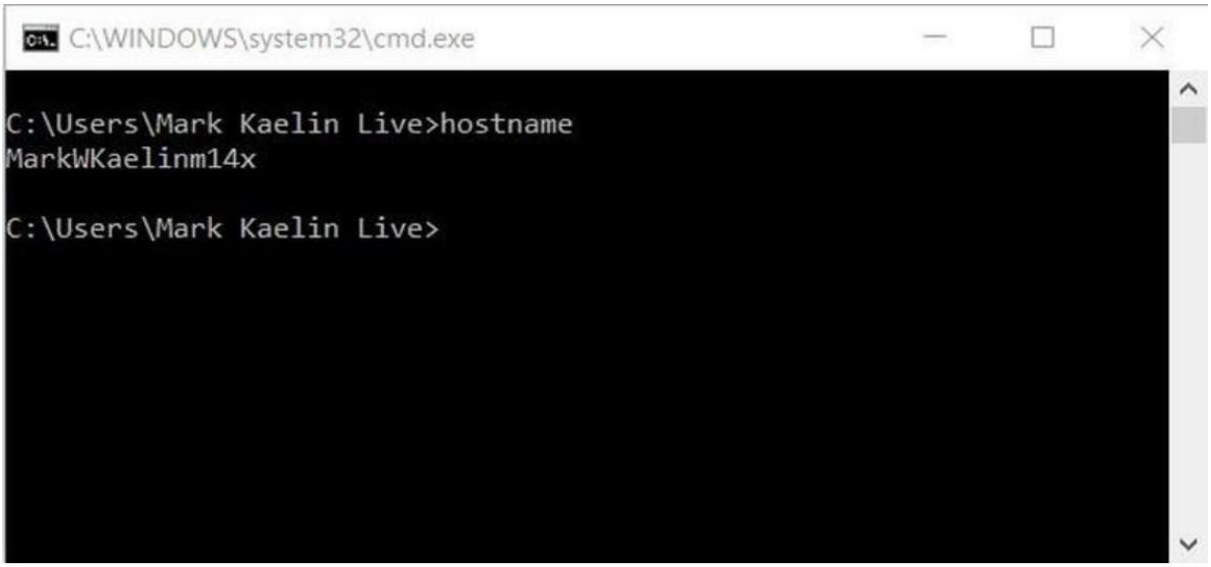
Check out Microsoft Docs for a more advanced look at the IP Config command and its variables andswitches.



3. HostName

The Windows 10 HostName network command will simply display the current name of yourWindows 10 computer (**Figure B**). This is the name your computer uses to identify itself to the otherdevices and servers on your local network. You can find this name in the System information screen in the GUI, but this command is quicker.

Figure B



To run the basic command, at the prompt type:

5. Trace route:

Traceroute uses Internet Control Message Protocol (ICMP) echo packets with variable time to live (TTL) values. The response time of each hop is calculated. To guarantee accuracy, each hop is queried multiple times (usually three times) to better measure the response of that particular hop.

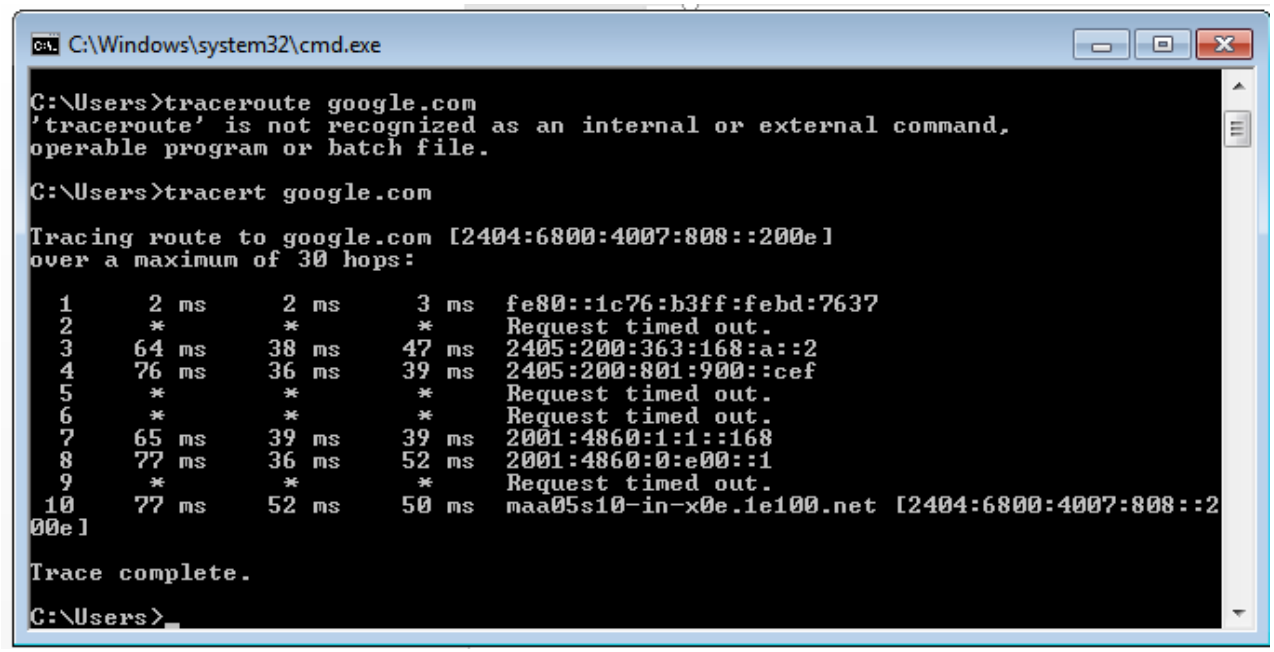
Traceroute is a network diagnostic tool used to track the pathway taken by a packet on an IP networkfrom source to destination. Traceroute also records the time taken for each hop the packet makes during its route to the destination. Traceroute uses Internet Control Message Protocol (ICMP) echo packets with variable time to live (TTL) values.

The response time of each hop is calculated. To guarantee accuracy, each hop is queried multiple times (usually three times) to better measure the response of that particular hop. Traceroute sends packets with TTL values that gradually increase

from packet to packet, starting with TTL value of

one. Routers decrement TTL values of packets by one when routing and discard packets whose TTL value has reached zero, returning the ICMP error message ICMP Time Exceeded. For the first set of packets, the first router receives the packet, decrements the TTL value and drops the packet because it then has TTL value zero. The router sends an ICMP Time Exceeded message back to the source. The next set of packets are given a TTL value of two, so the first router forwards the packets, but thesecond router drops them and replies with ICMP Time Exceeded.

Proceeding in this way, traceroute uses the returned ICMP Time Exceeded messages to build a list ofrouters that packets traverse, until the destination is reached and returns an ICMP Echo
Reply message. With the tracert command shown above, we're asking tracert to show us the pathfrom the local computer all the way to the network device with the hostname
www.google.co
m. #tracert
google.com

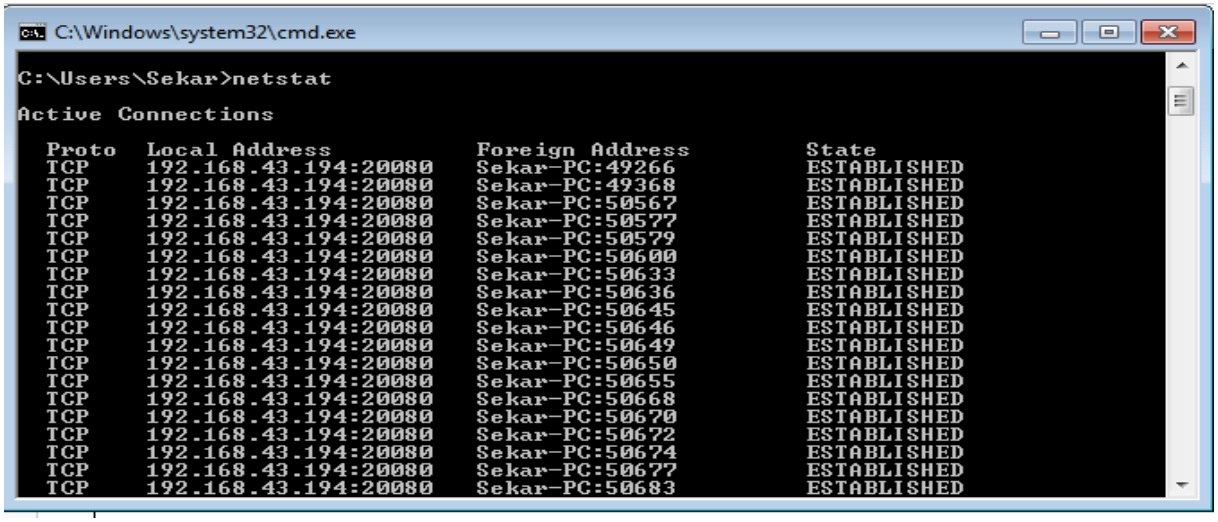


6.Netstat

Netstat is a common command line TCP/IP networking available in most versions of

Windows, Linux, UNIX and other operating systems. Netstat provides information and statistics about protocols in use and current TCP/IP network connections. The Windows help screen (analogous to a Linux or UNIX for netstat reads as follows:

displays protocol statistics and current TCP/IP network connections.#netstat

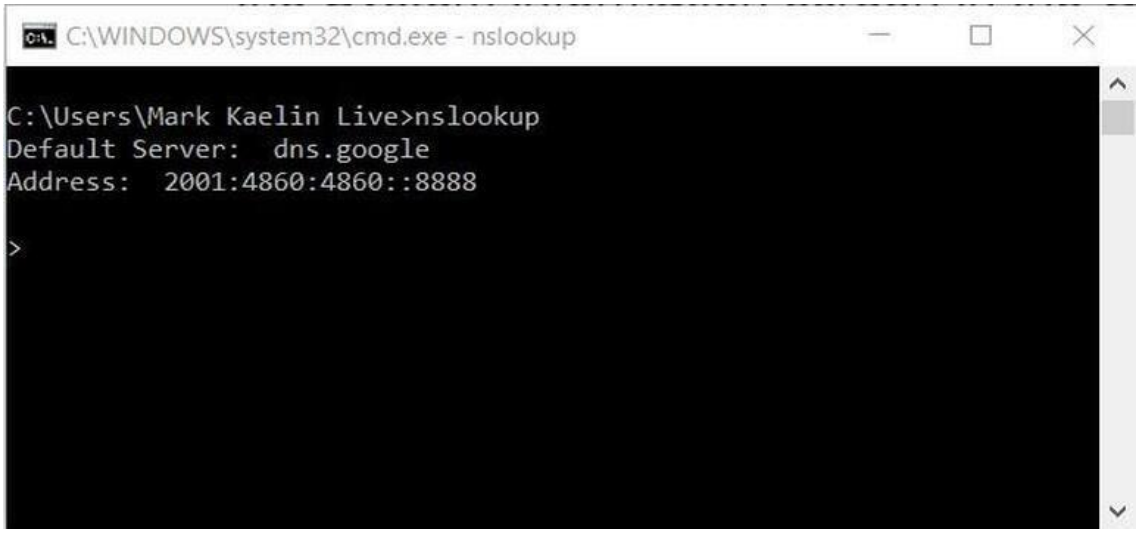


7. nslookup

The **nslookup** (which stands for *name server lookup*) command is a network utility program used to obtain information about internet servers. It finds name server information for domains by querying the Domain Name System.

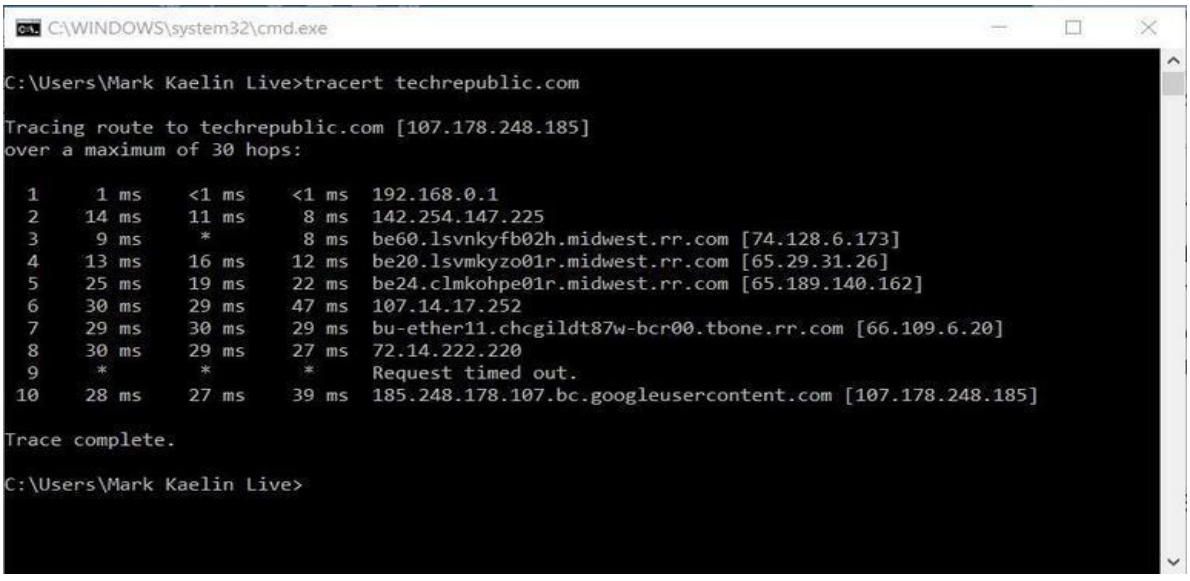
The nslookup command is a powerful tool for diagnosing DNS problems. You know you'reexperiencing a DNS problem when you can access a resource by specifying its IP address but not its DNS name.

#nslookup



8. Tracert

Another handy tool for troubleshooting network connections in Windows 10 is the Tracert command. This command will trace the route a data packet takes before reaching its destination, displaying information on each hop along the route. Each hop of the route will display the latency between your device and that particular hop and the IP address of the hop.



Conclusion:

EXPERIMENT NO. 3

Title: Capturing Packet in Live Traffic

Objective: To Capture the packets in live traffic using tools like Wireshark

AIM: Observe and note the details of the live type of traffic (ARP, Frame analysis, ethernet) from interface using packet capture and analysis tool

THEORY:

Requirements:

Wireshark: Wireshark software tool to capture and examine a packet trace.

arp: "arp" command-line utility to inspect and clear the cache used by the ARP protocol on your computer.

ifconfig / ipconfig: "ipconfig" (Windows) command-line utility to inspect the state of your computer's network interface.

route / netstat: "route" or "netstat" command-line utility to inspect the routes used by your computer.

Network Setup

ARP protocol in action is to be observed. ARP is used to find the Ethernet address that corresponds to a local IP address to which a computer wants to send a packet. A typical example of a local IP address is that of the local router or default gateway that connects a computer to the rest of the Internet. The defined computer caches these translations in an ARP cache so that the ARP protocol need only be used occasionally to do the translation. The setup from the viewpoint of your computer is as shown in the example below.

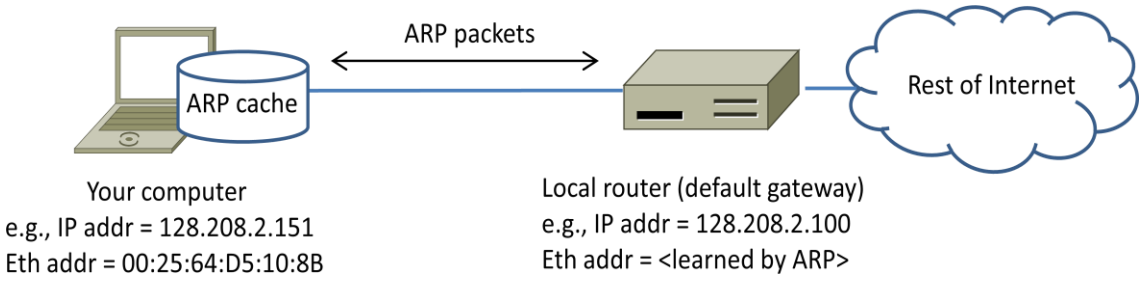


Figure 1: Network setup under which we will study ARP in second part

How ARP Works

When an incoming packet destined for a host machine on a particular local area network arrives ata gateway, the gateway asks the ARP program to find a physical host or MAC address that matches the IP address. The ARP program looks in the ARP cache and, if it finds the address, provides it so that the packet can be converted to the right packet length and format and sent to the machine. If no entry is found for the IP address, ARP broadcasts a request packet in a special format to all the machines on the LAN to see if one machine knows that it has that IP address associated with it.

A machine that recognizes the IP address as its own returns a reply so indicating. ARP updates the ARP cache for future reference and then sends the packet to the MAC address that replied. There is a Reverse ARP (RARP) for host machines that don't know their IP address. RARP enables them to request their IP address from the gateway's ARP cache.

Step 1: Finding your IP address and Gateway address

1. Open a command prompt as an administrator as follows:

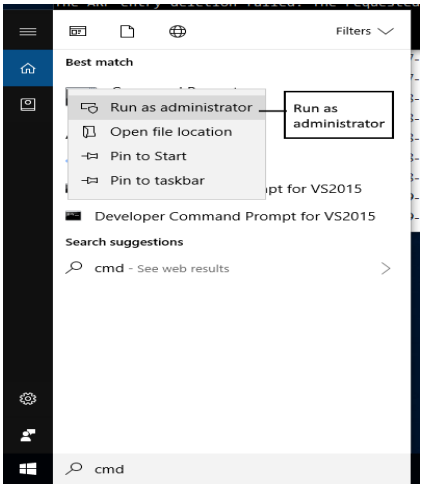


Figure 2: Finding the computer's Ethernet address with ipconfig (Windows)

2. Find the **Ethernet** address of the main network interface OR

the **wireless** address withthe ipconfig command. On Windows, bring up a command-line shell and type “ipconfig /all”. Common names for the interface are “eth0” or “Ethernet adapter”. An ex ample is shown below in figure 2, with added highlighting.

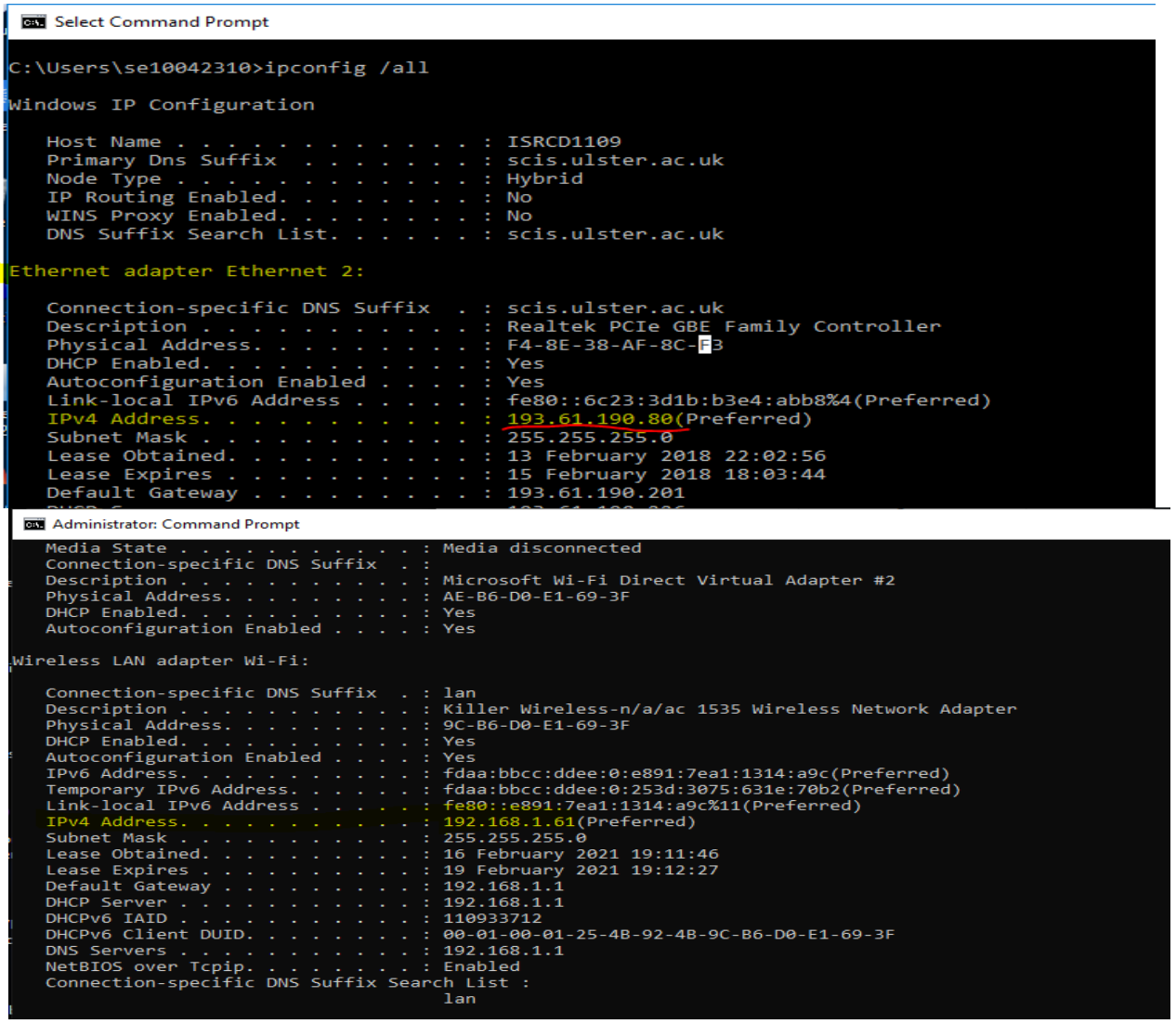


Figure 3: Finding the computer's WiFi IP address with ipconfig (Windows)

3. Find the IP address of the local router or default gateway that your computer uses to reach the rest of the Internet using the netstat / route command. You should be able touse the netstat-r command on Windows.

   Alternatively, you can use the route command ("route print" on Windows). In either case you are looking for the gateway IP address that corresponds to the destination of default or 0.0.0.0. An example is shown in figure 3 for netstat, with added highlighting.

```
C:\Users\se10042310>netstat -r
===========================================================================
Interface List
  4...f4 8e 38 af 8c f3 ......Realtek PCIe GBE Family Controller
  3...00 50 56 c0 00 01 ......VMware Virtual Ethernet Adapter for VMnet1
  6...00 50 56 c0 00 08 ......VMware Virtual Ethernet Adapter for VMnet8
  1...........................Software Loopback Interface 1
===========================================================================

IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0   193.61.190.201    193.61.190.80     25
        127.0.0.0        255.0.0.0         On-link         127.0.0.1    331
        127.0.0.1  255.255.255.255         On-link         127.0.0.1    331
  127.255.255.255  255.255.255.255         On-link         127.0.0.1    331
    192.168.139.0    255.255.255.0         On-link     192.168.139.1    291
    192.168.139.1  255.255.255.255         On-link     192.168.139.1    291
  192.168.139.255  255.255.255.255         On-link     192.168.139.1    291
    192.168.159.0    255.255.255.0         On-link     192.168.159.1    291
    192.168.159.1  255.255.255.255         On-link     192.168.159.1    291
  192.168.159.255  255.255.255.255         On-link     192.168.159.1    291
```

Figure 4: Finding the default gateway IP address with netstat(Windows)

4. Now **run Wireshark** by typing "*wireshark*" in the bottom left search box inWindows

5. You should see the main Wireshark interface. **Click on the Ethernet OR Wireless interface** to start traffic analysis on that interface.

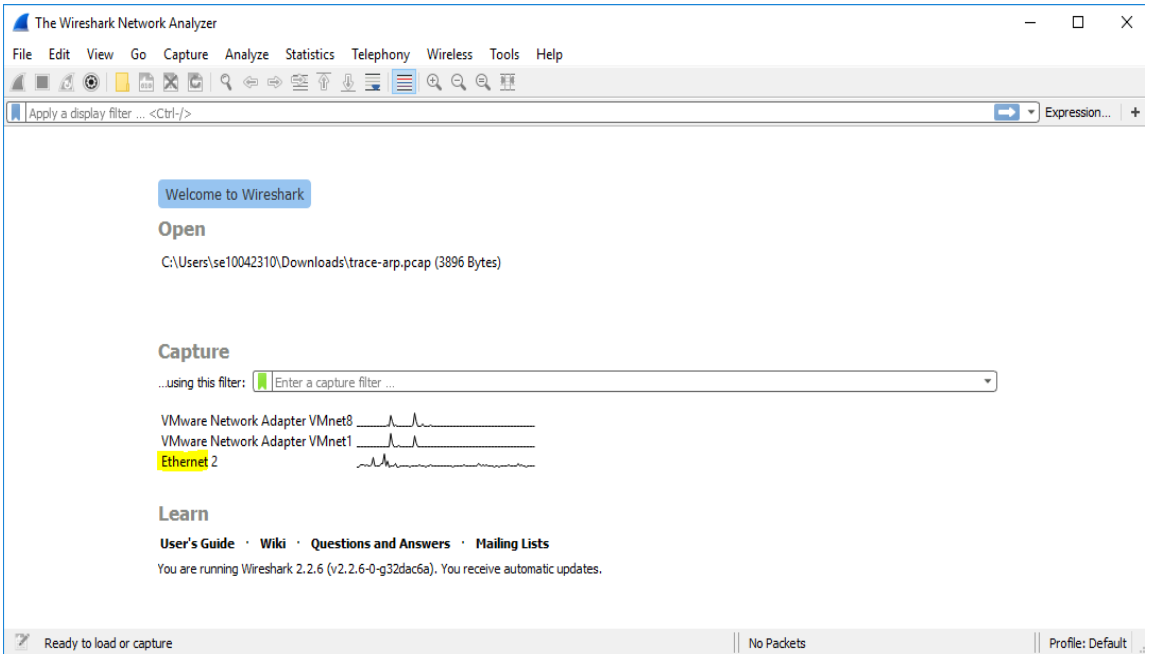


Figure 5: Applying a filter

6. Add a filter of "arp".

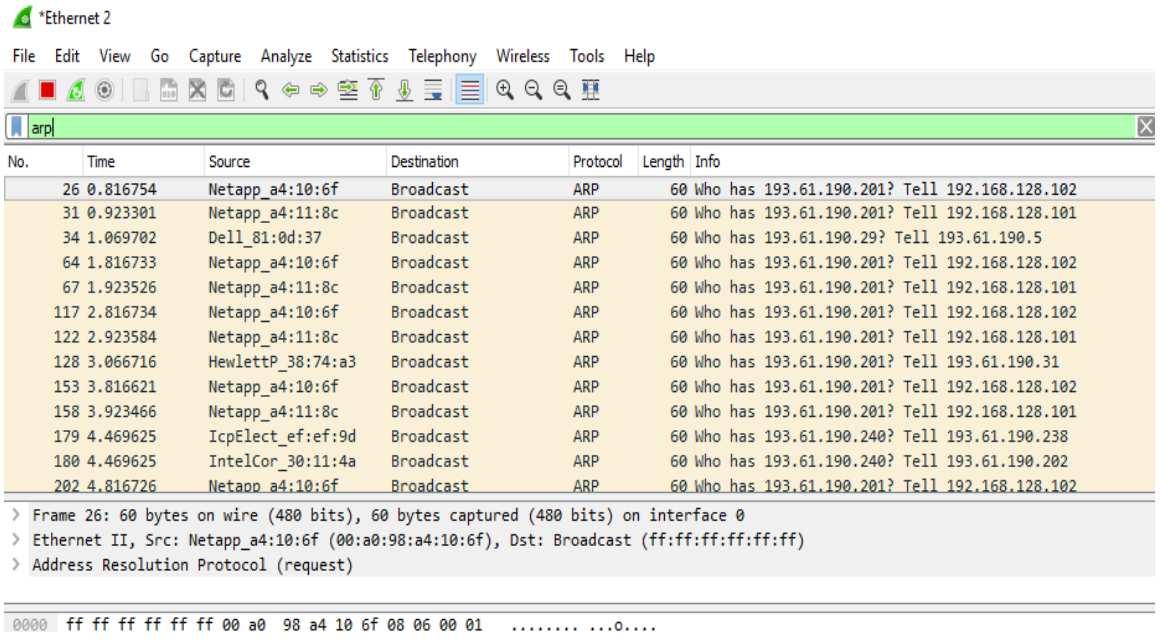
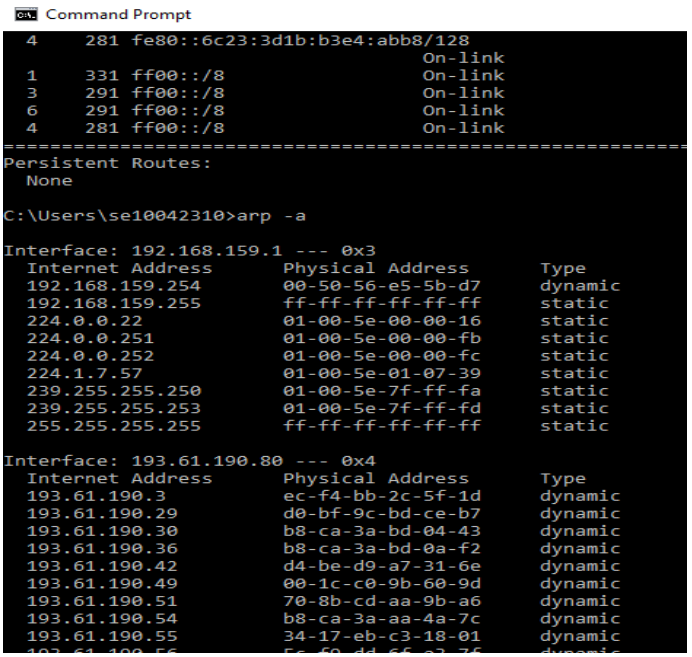
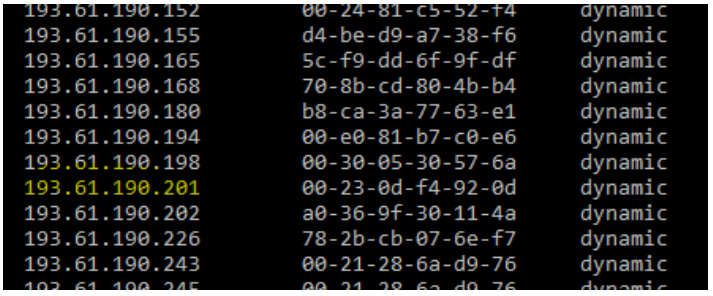


Figure 6: Setting up the capture options

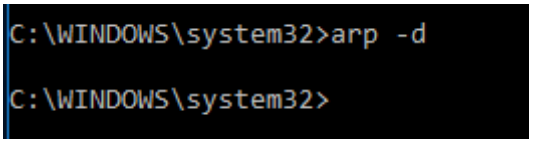
7. When the capture is started, use the "arp" command to clear the default gateway from the ARPcache. Using the command "arp –a" will show you the contents of the ARP cacheasa check that you can run "arp". Go to command prompt and type **arp -a** as shown below.


Command Prompt
4 281 fe80::6c23:3d1b:b3e4:abb8/128
On-link
1 331 ff00::/8 On-link
3 291 ff00::/8 On-link
6 291 ff00::/8 On-link
4 281 ff00::/8 On-link
Persistent Routes:
None
C:\Users\se10042310>arp -a
Interface: 192.168.159.1 --- 0x3
Internet Address Physical Address Type
192.168.159.254 00-50-56-e5-5b-d7 dynamic
192.168.159.255 ff-ff-ff-ff-ff-ff static
224.0.0.22 01-00-5e-00-00-16 static
224.0.0.251 01-00-5e-00-00-fb static
224.0.0.252 01-00-5e-00-00-fc static
224.1.7.57 01-00-5e-01-07-39 static
239.255.255.250 01-00-5e-7f-ff-fa static
239.255.255.253 01-00-5e-7f-ff-fd static
255.255.255.255 ff-ff-ff-ff-ff-ff static
Interface: 193.61.190.80 --- 0x4
Internet Address Physical Address Type
193.61.190.3 ec-f4-bb-2c-5f-1d dynamic
193.61.190.29 d0-bf-9c-bd-ce-b7 dynamic
193.61.190.30 b8-ca-3a-bd-04-43 dynamic
193.61.190.36 b8-ca-3a-bd-0a-f2 dynamic
193.61.190.42 d4-be-d9-a7-31-6e dynamic
193.61.190.49 00-1c-c0-9b-60-9d dynamic
193.61.190.51 70-8b-cd-aa-9b-a6 dynamic
193.61.190.54 b8-ca-3a-aa-4a-7c dynamic
193.61.190.55 34-17-eb-c3-18-01 dynamic

You should see an entry for the IP address of the default gateway as shown in image below. In this case it is 193.61.190.201 which is the default gateway on my office PC.



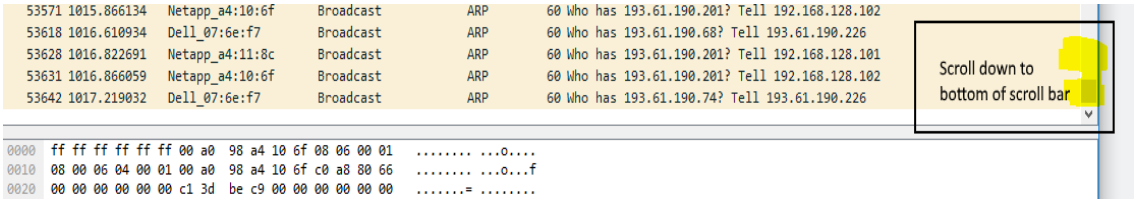
8. To clear this entry, use the arp command with different arguments ("arp –d" on Windows) as follows. Type **arp -d** in the command prompt.



Note: This usage of arp will need administrator privileges to run, so you have to run as a privi- leged user on Windows which is what you should have done in step 1. The command should run without error, but the ARP entry may not appear to be cleared if you check with "arp –a". This is because your computer will send ARP packets to repopulate this entry as soon as you need to send a packet to a remote IP address, and that can happen very quickly due to background activity on the computer.

Now that you have cleared your ARP cache, **fetch a remote page with your Web browser**. This will cause ARP to find the Ethernet address of the default gateway so that the packets can be sent.

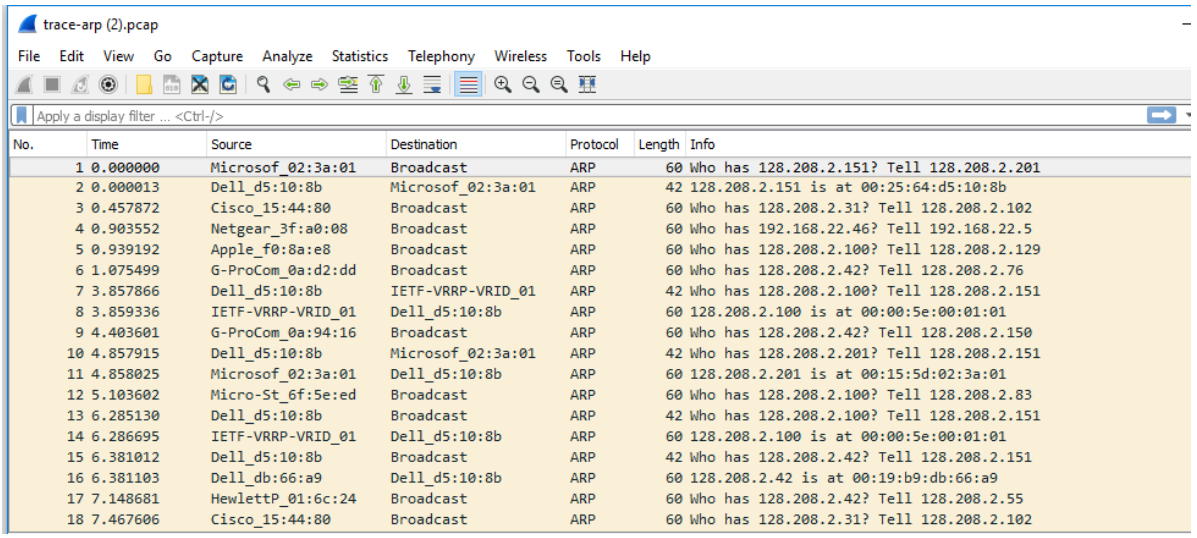
9. You will see these packets flowing through your computer by scrolling down in the Wireshark window to the bottom as shown below.



10. These ARP packets will be captured by Wireshark. You might clear the ARP cache and fetch a document a couple of times. Hopefully there will also be other ARP packets sent by other computers on the local network that will be captured. These packets are likely to be present if there are other computers on your local network. In fact, if you have a busy computer and extensive local network then you may capture many ARP packets. The ARP traffic of other computers will be captured when the ARP packets are sent to the broadcast address, since in this case they are destined for all computers including the one on which you are running Wireshark. Because ARP activity happens slowly, you may need to wait up to 30 seconds to observe some of this back- ground ARP traffic.
11. Once you have captured some ARP traffic, stop the capture. You will need the trace, plus the Ethernet address of your computer and the IP address of the default gateway for the next steps.

Step 2: Inspect the supplied ARP Trace

1. **Close** Wireshark.
2. Once Wireshark is closed, **open** the ARP trace here:
3. You should see a screen as shown below.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Microsof_02:3a:01	Broadcast	ARP	60	Who has 128.208.2.151? Tell 128.208.2.201
2	0.000013	Dell_d5:10:8b	Microsof_02:3a:01	ARP	42	128.208.2.151 is at 00:25:64:d5:10:8b
3	0.457872	Cisco_15:44:80	Broadcast	ARP	60	Who has 128.208.2.31? Tell 128.208.2.102
4	0.903552	Netgear_3f:a0:08	Broadcast	ARP	60	Who has 192.168.22.46? Tell 192.168.22.5
5	0.939192	Apple_f0:8a:e8	Broadcast	ARP	60	Who has 128.208.2.100? Tell 128.208.2.129
6	1.075499	G-ProCom_0a:d2:dd	Broadcast	ARP	60	Who has 128.208.2.42? Tell 128.208.2.76
7	3.857866	Dell_d5:10:8b	IETF-VRRP-VRID_01	ARP	42	Who has 128.208.2.100? Tell 128.208.2.151
8	3.859336	IETF-VRRP-VRID_01	Dell_d5:10:8b	ARP	60	128.208.2.100 is at 00:00:5e:00:01:01
9	4.403601	G-ProCom_0a:94:16	Broadcast	ARP	60	Who has 128.208.2.42? Tell 128.208.2.150
10	4.857915	Dell_d5:10:8b	Microsof_02:3a:01	ARP	42	Who has 128.208.2.201? Tell 128.208.2.151
11	4.858025	Microsof_02:3a:01	Dell_d5:10:8b	ARP	60	128.208.2.201 is at 00:15:5d:02:3a:01
12	5.103602	Micro-St_6f:5e:ed	Broadcast	ARP	60	Who has 128.208.2.100? Tell 128.208.2.83
13	6.285130	Dell_d5:10:8b	Broadcast	ARP	42	Who has 128.208.2.100? Tell 128.208.2.151
14	6.286695	IETF-VRRP-VRID_01	Dell_d5:10:8b	ARP	60	128.208.2.100 is at 00:00:5e:00:01:01
15	6.381012	Dell_d5:10:8b	Broadcast	ARP	42	Who has 128.208.2.42? Tell 128.208.2.151
16	6.381103	Dell_db:66:a9	Dell_d5:10:8b	ARP	60	128.208.2.42 is at 00:19:b9:db:66:a9
17	7.148681	HewlettP_01:6c:24	Broadcast	ARP	60	Who has 128.208.2.42? Tell 128.208.2.55
18	7.467606	Cisco_15:44:80	Broadcast	ARP	60	Who has 128.208.2.31? Tell 128.208.2.102

The setup from the viewpoint of your computer from this trace is shown in the example below.

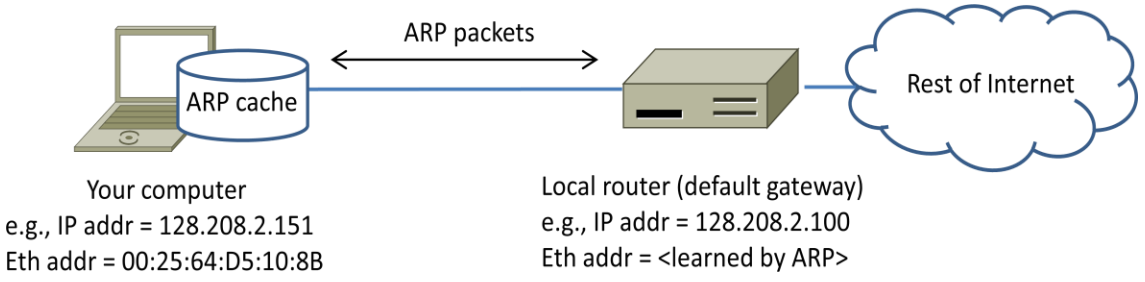


Figure 7: Network setup under which we will study ARP in thispart Note: **Ethernet address** of computer: 00:25:64:d5:10:8b and IP address of **gateway**:128.208.2.100

4. Now we can look at an ARP exchange. Since there may be many ARP packets in your trace, we'll first narrow our view to only the ARP packets that are sent directly from or to your computer.

Set a display filter for packets with the Ethernet address of your computer which is this case is

00:25:64:d5:10:8b.

You can do this by entering an expression in the blank "Filter:" box near the top of the Wireshark window and clicking "Apply" or Enter. After applying this filter your capture should look something like the figure below, in which we have expanded the ARP protocol details.

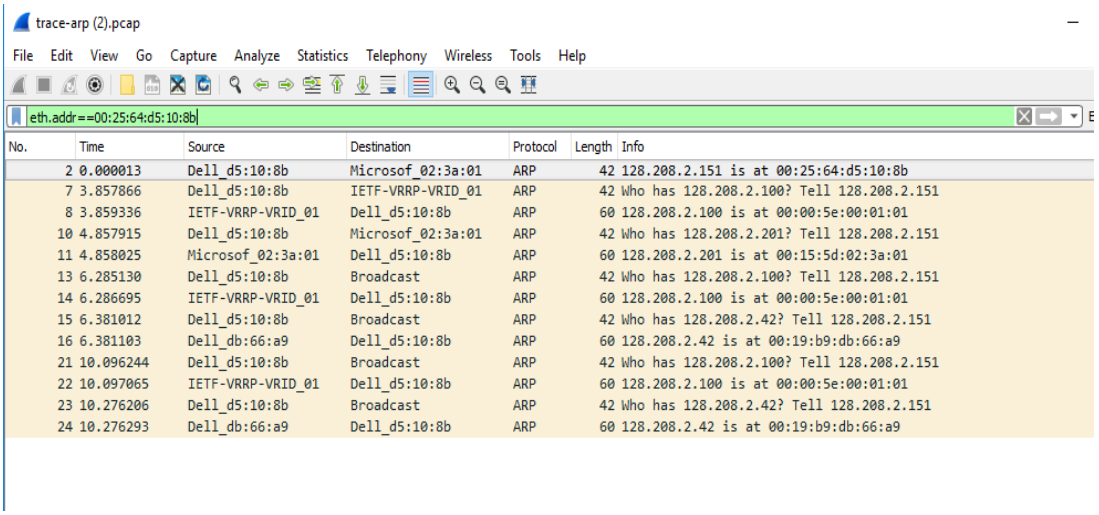


Figure 8: Capture of ARP packets, showing details of a request

Find and select an ARP request for the default gateway and examine its fields. There are two kinds of ARP packets, a request and a reply, Next come the four key fields, the sender MAC (Ethernet) and IP and the target MAC (Ethernet) and IP. These fields are filled in as much as possible. For a request, the sender knows their MAC and IP address and fills them in. The sender also knows the target IP address – it is the IP address for which an Ethernet address is wanted. But the sender does not know the target MAC

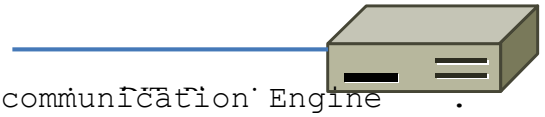
Step 3: Details of ARP over Ethernet

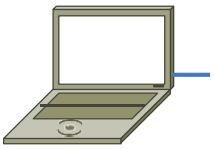
ARP packets are carried in Ethernet frames, and the values of the Ethernet header fields are chosen to support ARP. For instance, you may wonder how an ARP request packet is delivered to the target computer so that it can reply and tell the requestor its MAC address. The answer is that the ARP request is (normally) broadcast at the Ethernet layer so that it is received by all computers on the local network including the target. Look specifically at the destination Ethernet address of a request: it is set to ff:ff:ff:ff:ff:ff, the broadcast address. So, the target receives the request and recognizes that it is the in- tended recipient of the message; other computers that receive the request know that it is not meant for them. Only the target responds with a reply. However, anyone who receives an ARP packet can learn a mapping from it: the sender MAC and sender IP pair. The ARPheader for a request and a reply is 28 bytes for both the request and reply for IPv4.

Answers to Step 3: ARP request and reply

Your computer Sender MAC=00:25:64:D5:10:8BSender IP=128.208.2.151

Target MAC=00:00:00:00:00:00 Target IP=128.208.2.1





Default gateway

Reply

Sender MAC=00:00:5E:00:00:01

Sender
IP=128.208.
2.10 0
Target
MAC=00:25:64:D5
:1 0:
8B Target

IP=128.208.2.151

Figure 9: Details of the ARP request and reply to resolve the default gateway

There are several features to note:

- On the request, the target MAC is not known so it is usually filled in as 00:00:00:00:00:00.
- On the reply, the request target becomes the reply sender and vice versa.
- On the reply, the sender MAC returns the answer that is sought; it ishighlighted.
- All of the fields that are shown are ARP header fields

CONCLUSION:

EXPERIMENT NO.4

TITLE: Using a Network Simulator (e.g., packet tracer) Configure router using RIP

OBJECTIVE: To Configure and analyze the performance of the Routing Information Protocol (RIP) .

AIM: Installation and configuration of RIP using Cisco packet tracer.

SOFTWARE USED: Cisco packet Tracer

HARDWARE USED:

- Operating System: Windows XP/Vista/7/8,10
- Memory (RAM): 512MB of RAM required.
- Hard Disk Space: 300MB of free space required.
- Processor: Intel Pentium 4 or later.

THEORY:

A router in the network needs to be able to look at the destination address in the packet and then determine which one of the output ports is the best choice to get the packet to that address. The router makes this decision by consulting a forwarding table. The fundamental problem of routing is: How do routers acquire the information in their forwarding tables? Routing algorithms are required to build the routing tables and, hence, forwarding tables. The basic problem of routing is to find the lowest-cost path between any two nodes, where the cost

of a path equals the sum of the costs of all the edges that make up the path. Routing is achieved in most practical networks by running routing protocols among the nodes. The protocols provide a distributed, dynamic way to solve the problem of finding the lowest-cost path in the presence of link and node failures and changing edge costs. One of the main classes of routing algorithms is the distance-vector algorithm. Each node constructs a vector containing the distances (costs) to all other nodes and distributes that vector to its immediate neighbors. RIP is the canonical example of a routing protocol built on the distance-vector algorithm. Routers running RIP send their advertisements regularly (e.g., every 30 s). A router also sends an update message whenever a triggered update from another router causes it to change its routing table. The Internet Control Message Protocol (ICMP) can be utilized to analyze the performance of the created routes. It can be used to model traffic between routers without the need for running applications in an end node. In this lab, you will set up a network that utilizes RIP as its routing protocol. You will analyze the routing tables generated in the routers, and you will observe how RIP is affected by link failures.

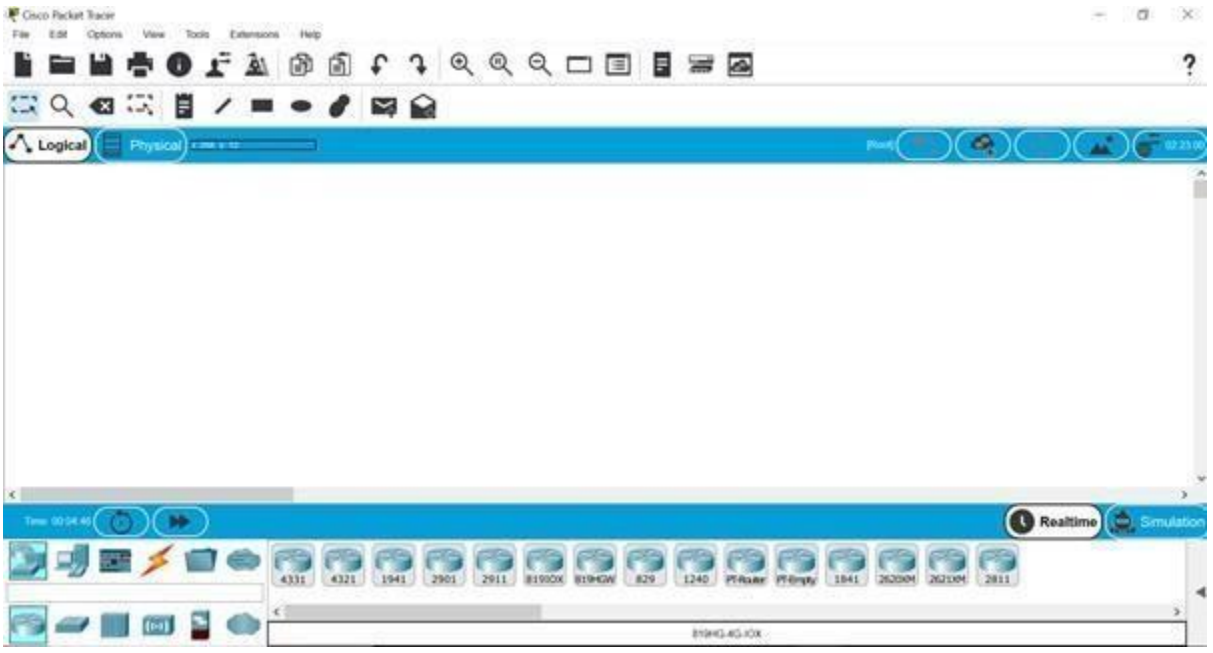
RIP (Routing Information Protocol) RIP is a standardized Distance Vector protocol, designed for use on smaller networks. RIP was one of the first true Distance Vector routing protocols, and is supported on a wide variety of systems.

RIP adheres to the following Distance Vector characteristics:

- RIP sends out periodic routing updates (every 30 seconds)
- RIP sends out the full routing table every periodic update
- RIP uses a form of distance as its metric (in this case, hop count)
- RIP uses the Bellman-Ford Distance Vector algorithm to determine the best “path” to a particular destination.

PROCEDURE:

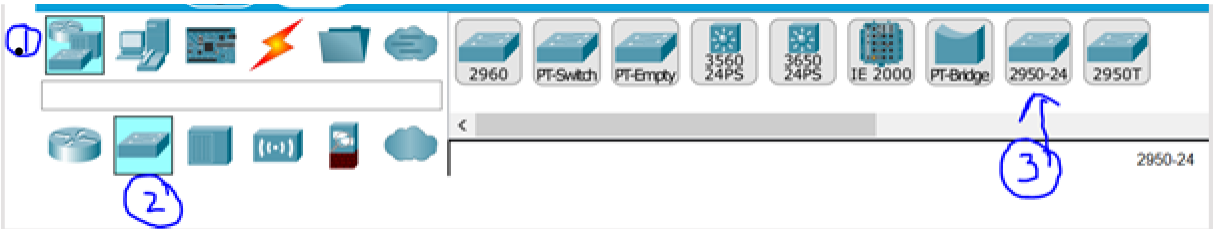
STEP 1: OPEN CISCO PACKET TRACER



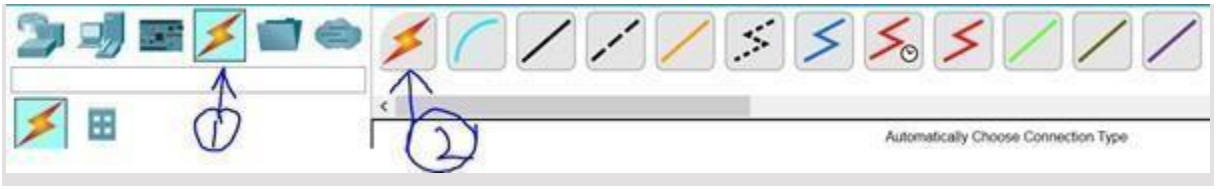
Blank Cisco Packet Tracer

STEP 2: MAKE CONNECTIONS: As shown in the figure below, go to (1) End Devices and select (2) PC and then finally drag and drop PC on Screen.

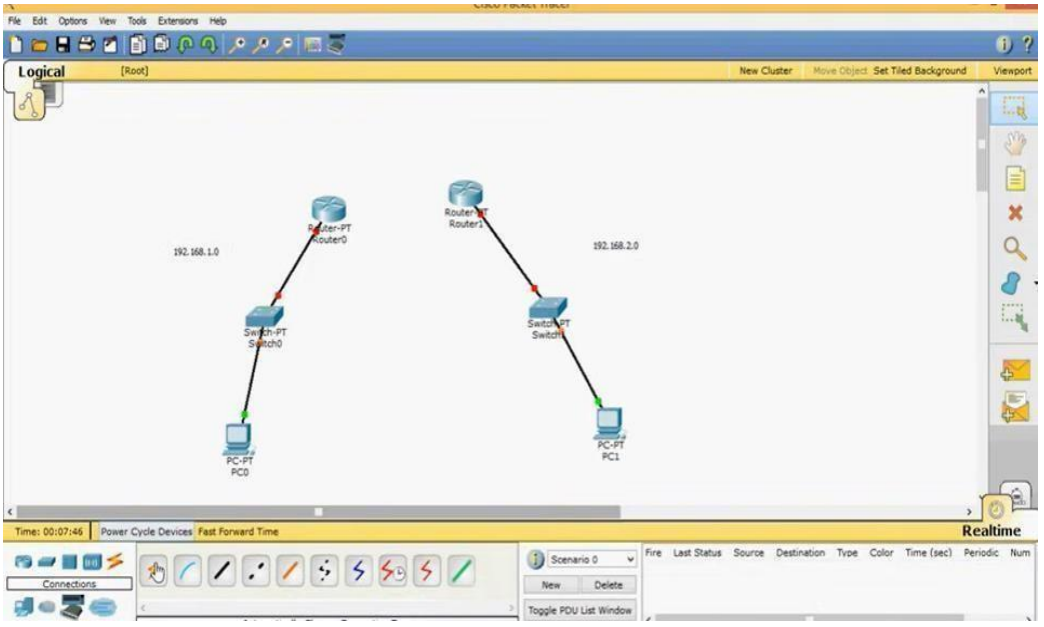
Then select a switch [I have taken Switch named as 2950-24]. Go to (1)Network device ->(2) switches ->(3) 2950-24 as shown in the

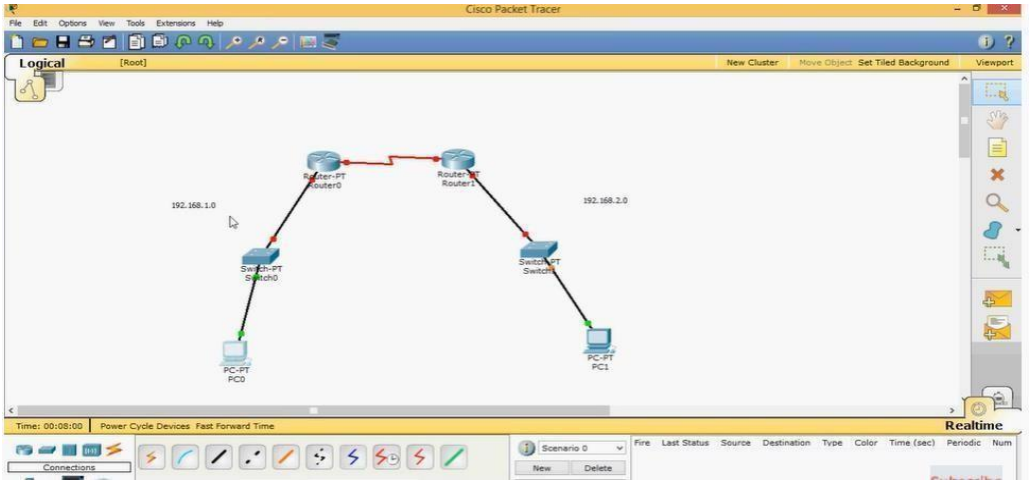


Then finally take a wire and connect PC to switch or you can click first option in connection as automatically choose connection type as shown below no .2 (use copper straight through

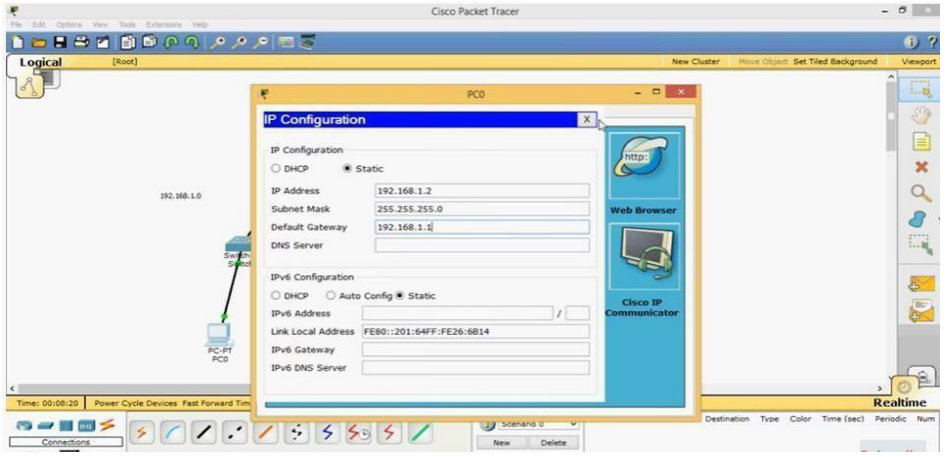


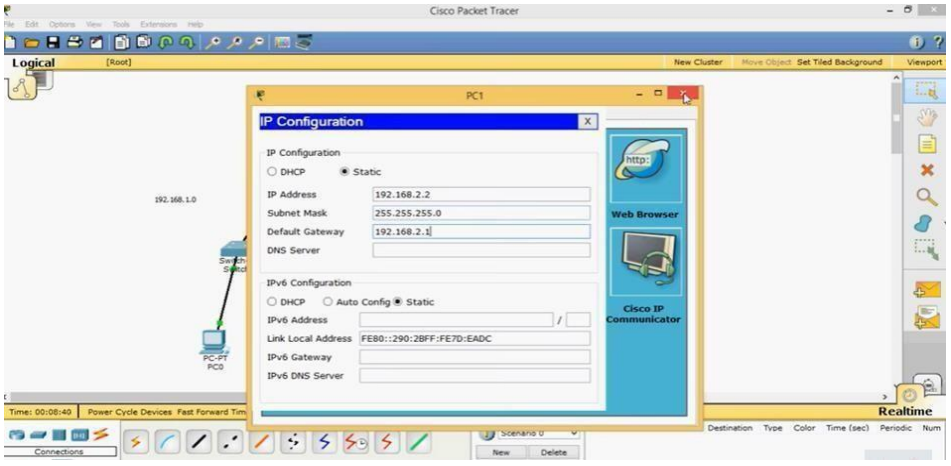
Connect Router0 and Router 1



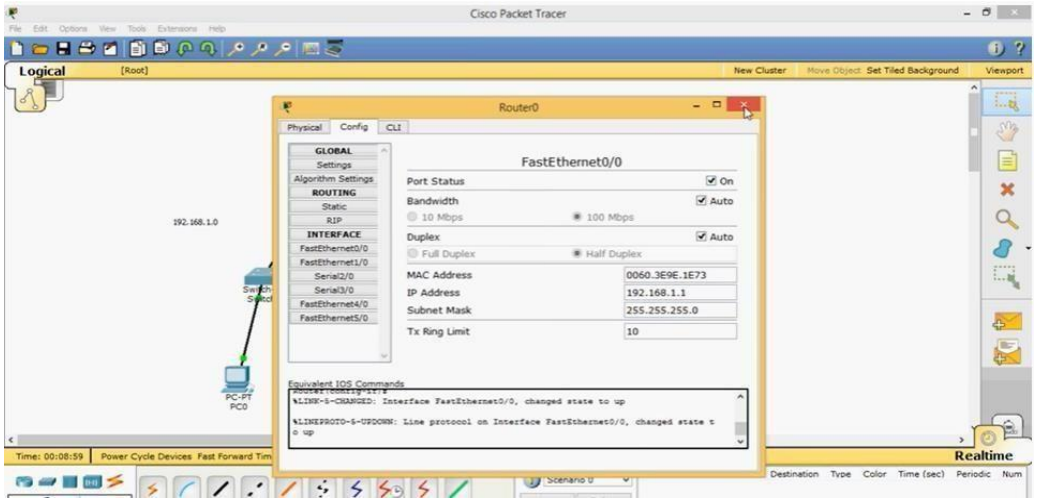


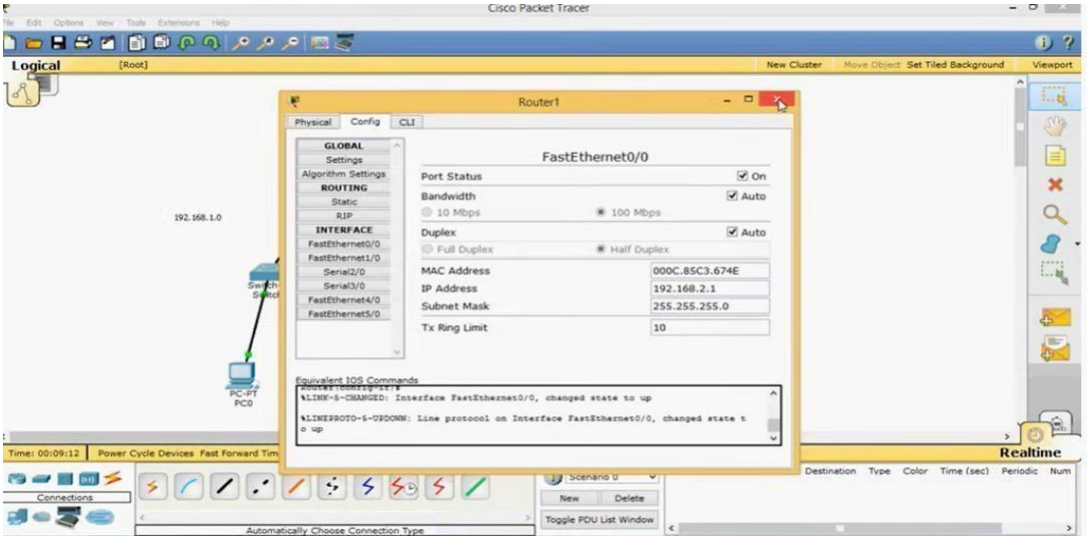
STEP 3: Configuration of IP for PC 0 &PC1



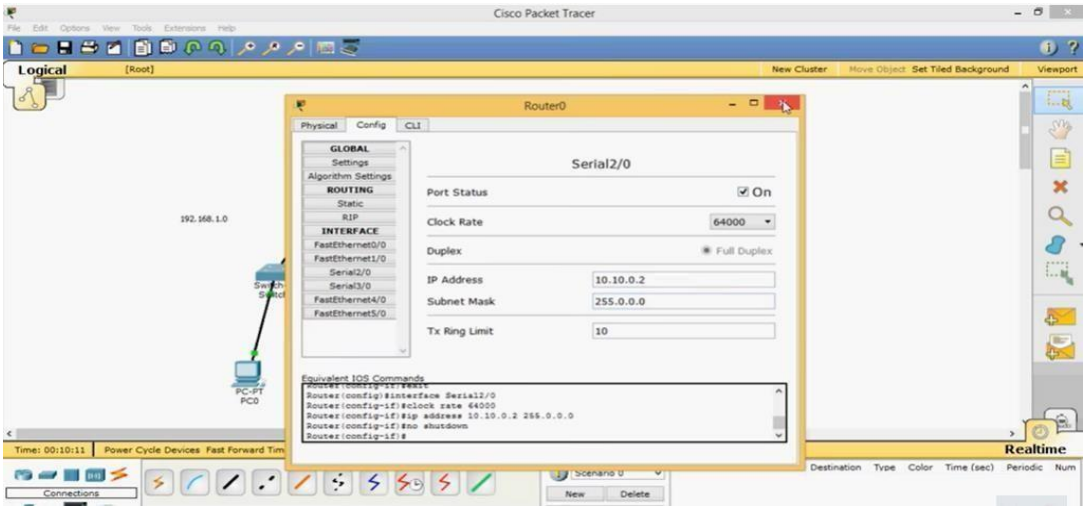


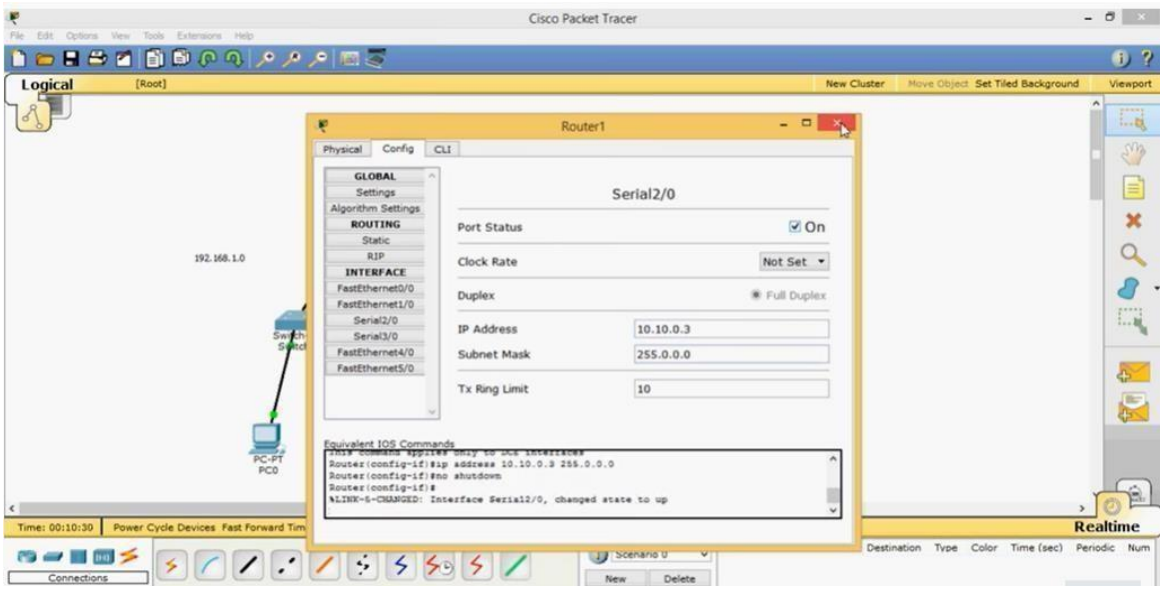
STEP 4: Configuration of Router 0 & Router 1





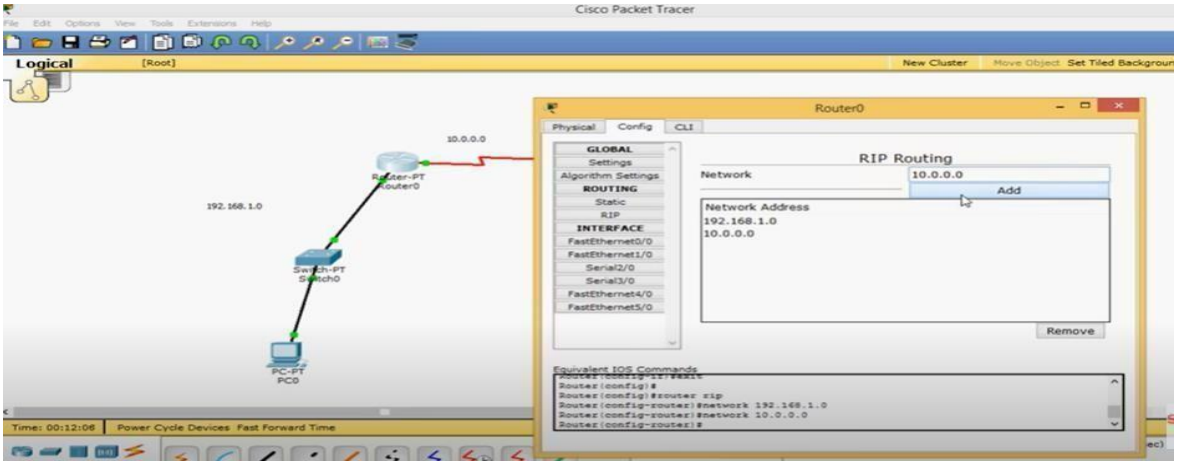
The Network between two router as IP address 10.0.0.0, We have connected a serial cable to the serial interface of the router0 &router 1, determine the interface is DTE or DCE. Since clocking is required to enable the interface, one of the two routers should function as DCE and DTE should provide clocking. Now configure serial2/0 of router 0 and router 1 as shown


Cisco Packet Tracer
Logical
[Root]
Router0
Physical
Config
CLI
GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
INTERFACE
FastEthernet0/0
FastEthernet1/0
Serial2/0
Serial3/0
FastEthernet4/0
FastEthernet5/0
Serial2/0
Port Status
On
Clock Rate
64000
Duplex
Full Duplex
IP Address
10.10.0.2
Subnet Mask
255.0.0.0
Tx Ring Limit
10
Equivalent IOS Commands
Router(config)#interface Serial2/0
Router(config-if)#clock rate 64000
Router(config-if)#ip address 10.10.0.2 255.0.0.0
Router(config-if)#no shutdown
Router(config-if)#
192.168.1.0
PC-PT
PC0
Realtime

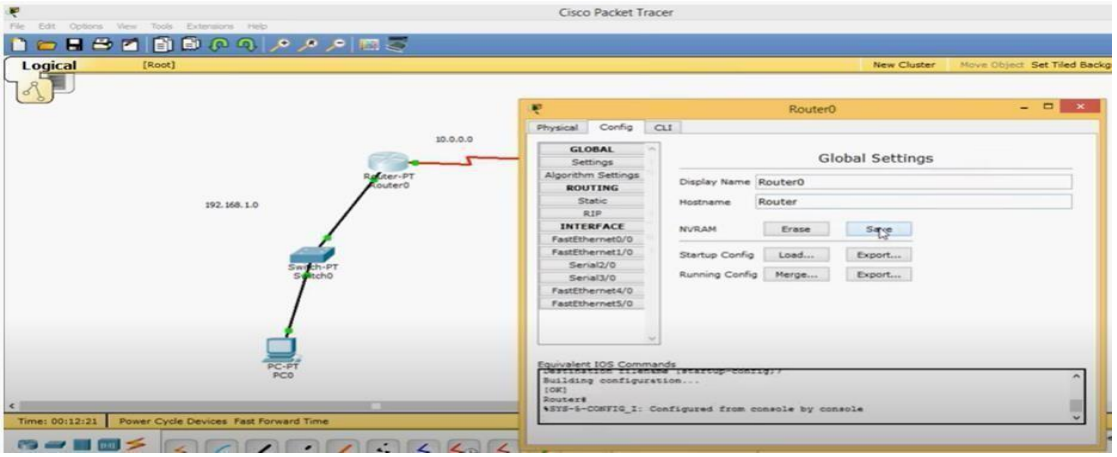

Cisco Packet Tracer
Logical
[Root]
Router1
Physical
Config
CLI
GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
INTERFACE
FastEthernet0/0
FastEthernet1/0
Serial2/0
Serial3/0
FastEthernet4/0
FastEthernet5/0
Serial2/0
Port Status
On
Clock Rate
Not Set
Duplex
Full Duplex
IP Address
10.10.0.3
Subnet Mask
255.0.0.0
Tx Ring Limit
10
Equivalent IOS Commands
Router(config-if)#ip address 10.10.0.3 255.0.0.0
Router(config-if)#no shutdown
Router(config-if)#
%LINK-5-CHANGED: Interface Serial2/0, changed state to up
192.168.1.0
PC-PT
PC0
Realtime

Configure Routing information in to router 0:

add Network IP 192.168.1.0 and 10.0.0.0



Save the global settings router 0



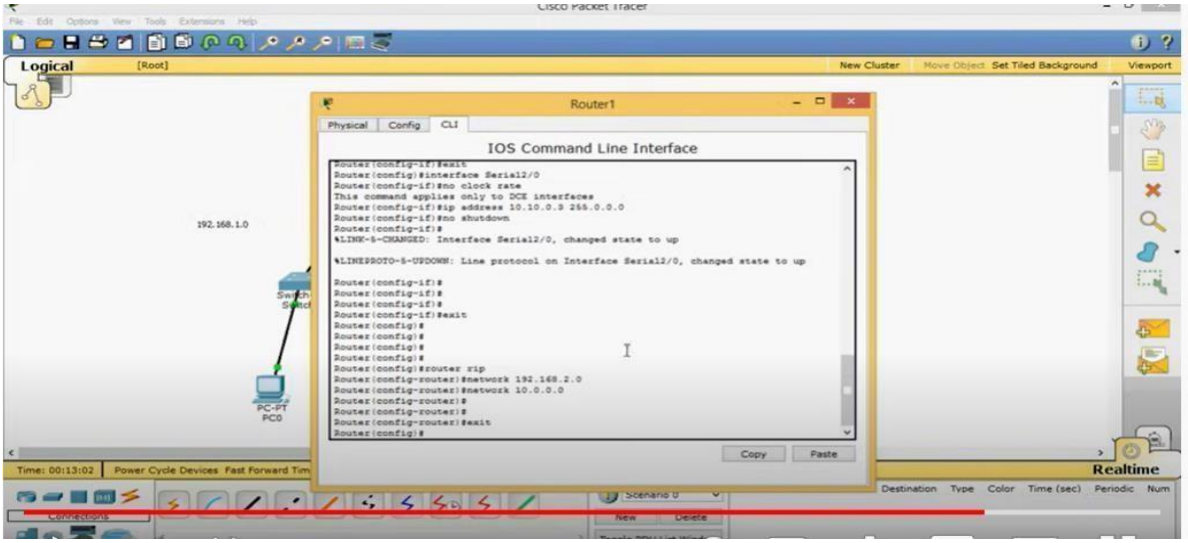
Or configure using CLI of router 1 and save global setting

On Router1, execute the following commands to configure RIP routing. Router1(config)#router rip

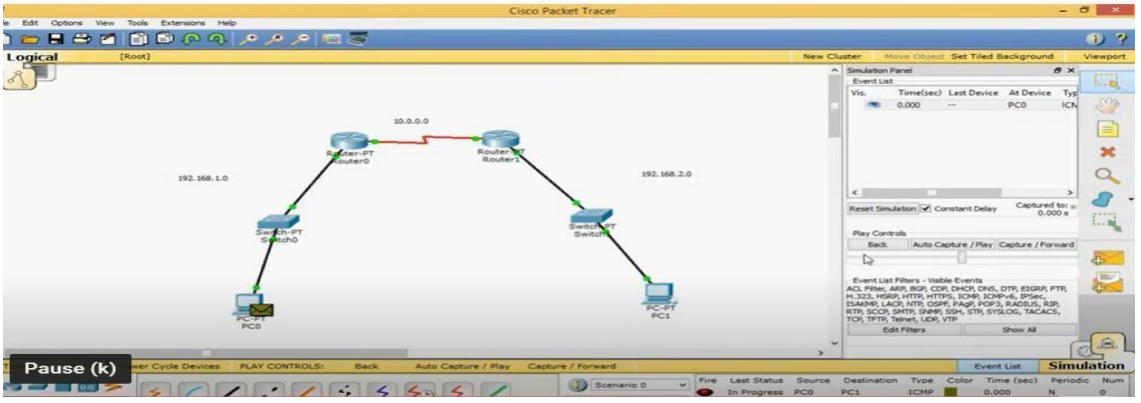
Router1(config-router)#network 192.168.1.0

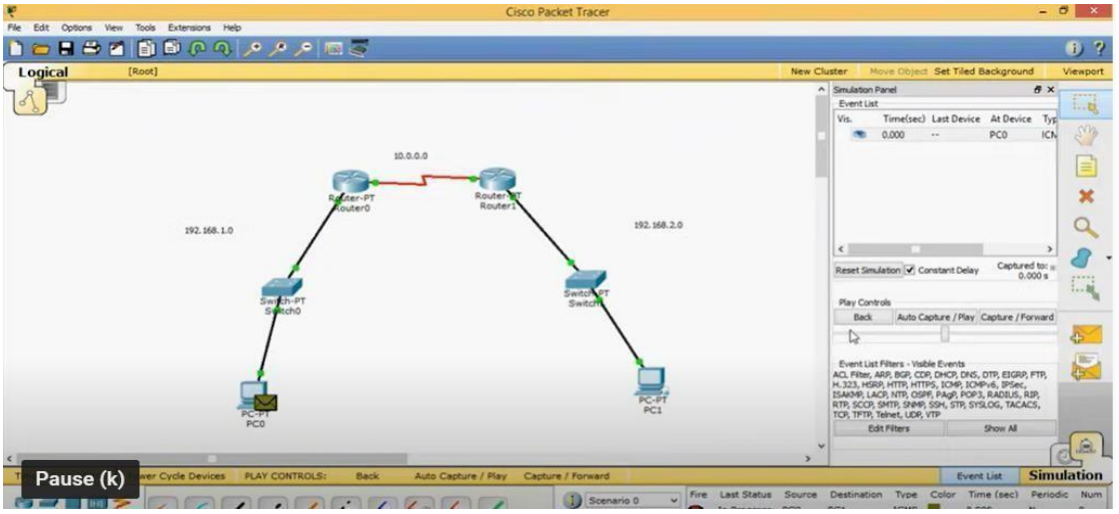
Router1(config-router)#network 10.0.0.0

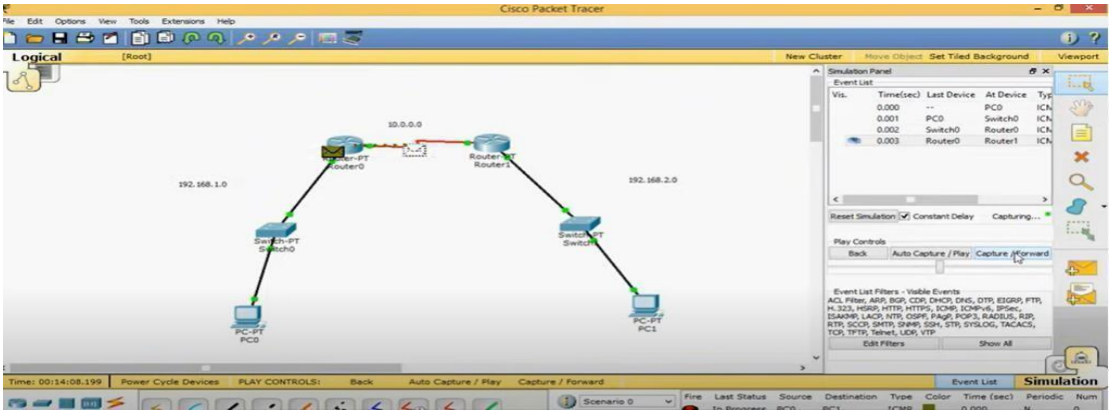
Router1(config-router)#exit

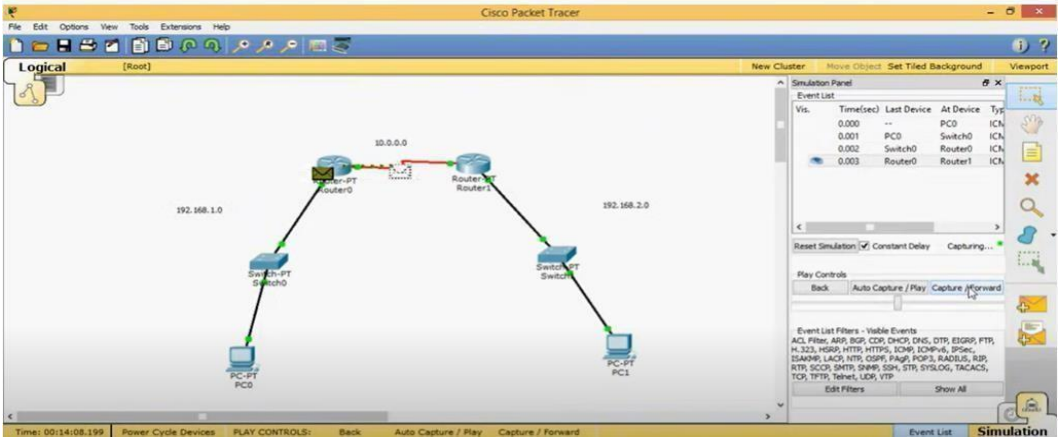


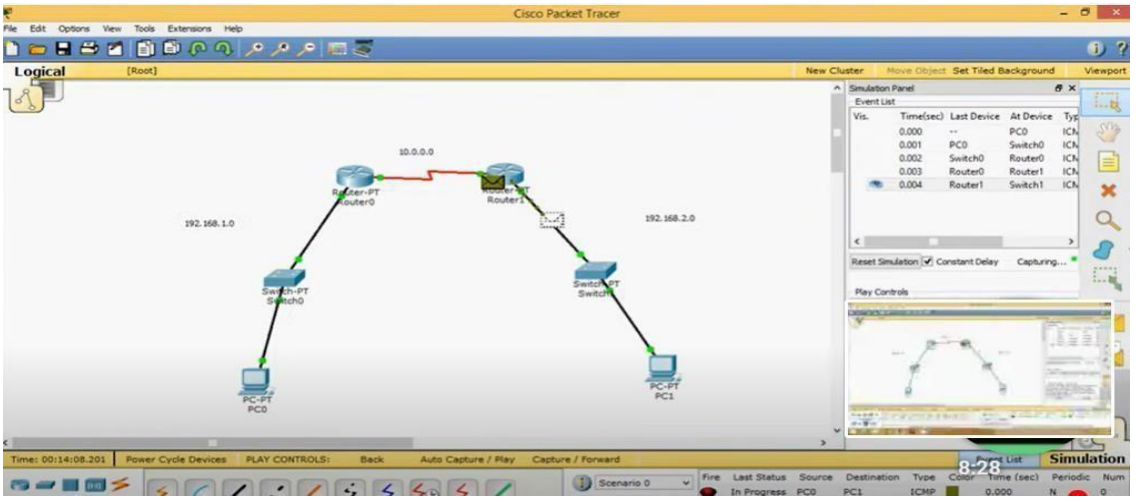
Observe the simulation by adding packet PC0& PC 1

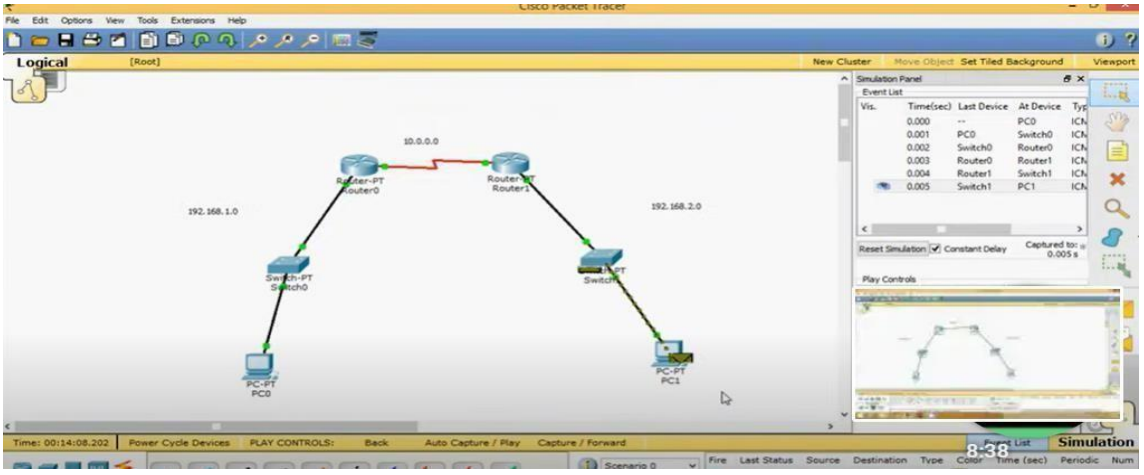



Cisco Packet Tracer
Logical
[Root]
10.0.0.0
192.168.1.0
192.168.2.0
Router0
Router1
Switch0
Switch1
PC0
PC1
Pause (k)
Simulation


Cisco Packet Tracer
Logical
[Root]
10.0.0.0
192.168.1.0
192.168.2.0
Router0
Router1
Switch0
Switch1
PC0
PC1
Simulation


Cisco Packet Tracer
Logical
[Root]
10.0.0.0
192.168.1.0
192.168.2.0
Router0
Router1
Switch0
Switch1
PC0
PC1
Simulation





OUTPUT:

CONCLUSION:

Experiment no – 5

Title- Implementation of data link layer protocol

Objective- Configuring HDLC protocol for data communication

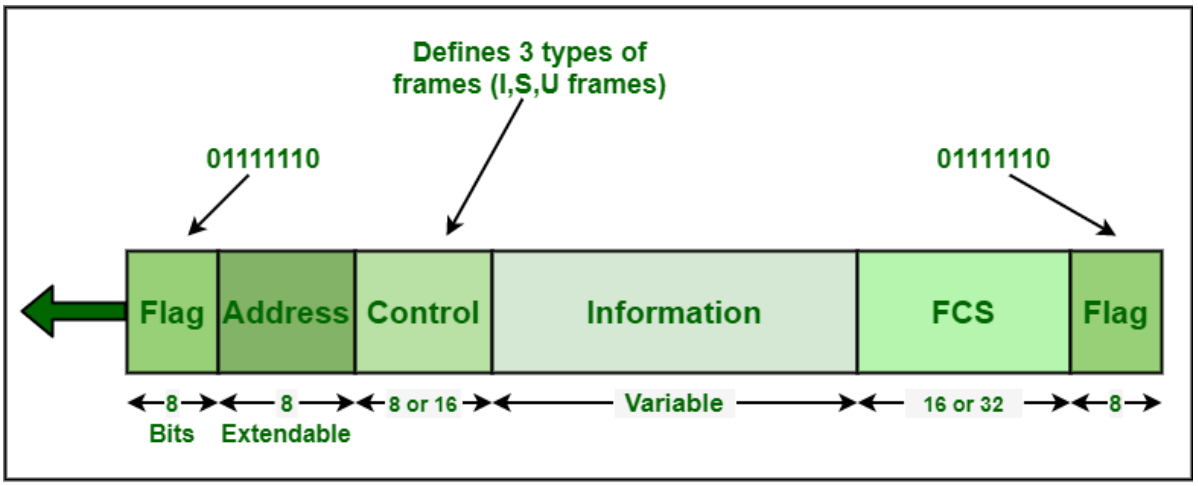
Aim- Implementing HDLC protocol and configuring it for demonstrating data link layers communication

Software used- Cisco Packet Tracer

Theory-

High-Level Data Link Control (HDLC) generally uses term "frame" to indicate and represent an entity of data or a protocol of data unit often transmitted or transferred from one station to another station. Each and every frame on link should begin and end with Flag Sequence Field (F). Each of frames in HDLC includes mainly six fields. It begins with a flag field, an address field, a control field, an information field, an frame check sequence (FCS) field, and an ending flag field. The ending flag field of one frame can serve as beginning flag field of the next frame in multiple-frame transmissions.

The basic frame structure of HDLC protocol is shown below:



Basic Frame Structure

Let us understand these fields in details:

Flag Field –

The flag field is generally responsible for initiation and termination of error checking. In HDLC protocol, there is no start and stop bits. So, the flag field is basically using delimiter 0x7e to simply indicate beginning and end of frame.

It is an 8-bit sequence with a bit pattern 01111110 that basically helps in identifying both starting and end of a frame. This bit pattern also serves as a synchronization pattern for receiver. This bit pattern is also not allowed to occur anywhere else inside a complete frame.

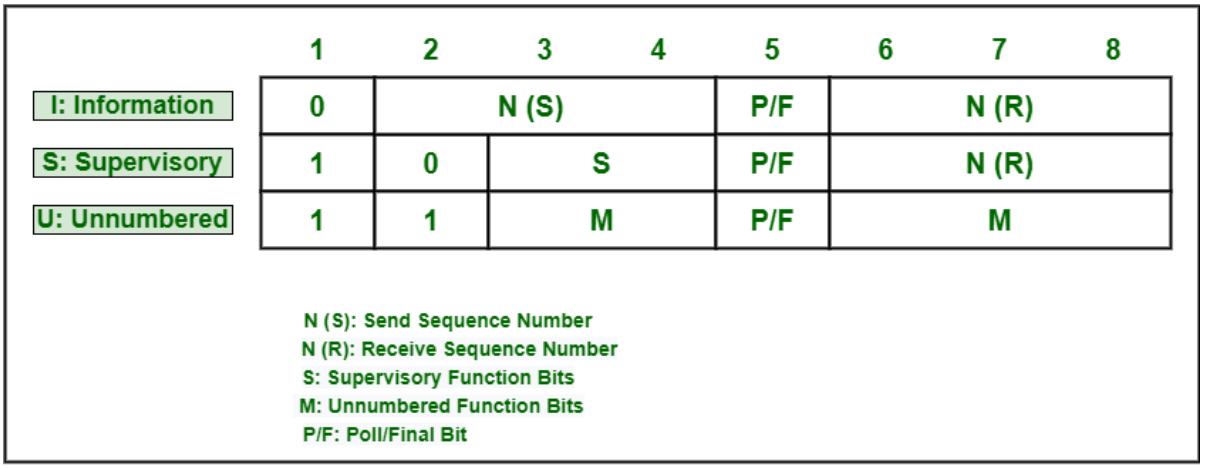
Address Field –

The address field generally includes HDLC address of secondary station. It helps to identify secondary station will sent or receive data frame. This field also generally consists of 8 bits therefore it is capable of addressing 256 addresses. This field can be of 1 byte or several bytes long, it depends upon requirements of network. Each byte can identify up to 128 stations.

This address might include a particular address, a group address, or a broadcast address. A primary address can either be a source of communication or a destination that eliminates requirement of including address of primary.

Control Field –

HDLC generally uses this field to determine how to control process of communication. The control field is different for different types of frames in HDLC protocol. The types of frames can be Information frame (I-frame), Supervisory frame (S-frame), and Unnumbered frame (U- frame).



Control Field Format

This field is a 1-2-byte segment of frame generally requires for flow and error control. This field basically consists of 8 bits but it can be extended to 16 bits. In this field, interpretation of bits usually depends upon the type of frame.

Information Field –

This field usually contains data or information of users sender is transmitting to receiver in an I-frame and network layer or management information in U-frame. It also consists of user's data and is fully transparent. The length of this field might vary from one network to another network.

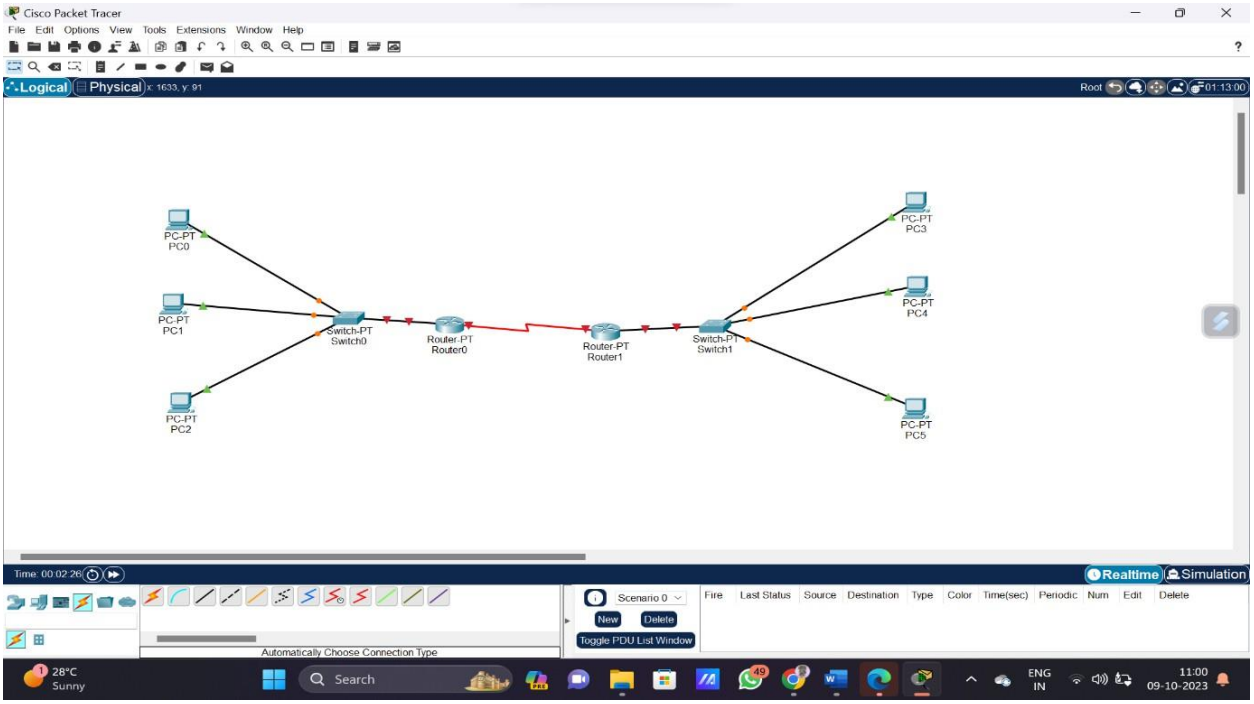
Information field is not always present in an HDLC frame.

Frame Check Sequence (FCS) –

FCS is generally used for identification of errors i.e., HDLC error detection. In FCS, CRC16 (16-bit Cyclic Redundancy Check) or CRC32 (32-bit Cyclic Redundancy Check) code is basically used for error detection. CRC calculation is done again in receiver. If somehow result differs even slightly from value in original frame, an error is assumed.
This field can either contain 2 byte or 4 bytes. This field is a total 16 bit that is required for error detection in address field, control field, and information field. FCS is basically calculated by sender and receiver both of a data frame. FCS is used to confirm and ensure that data frame was not corrupted by medium that is used to transfer frame from sender to receiver.

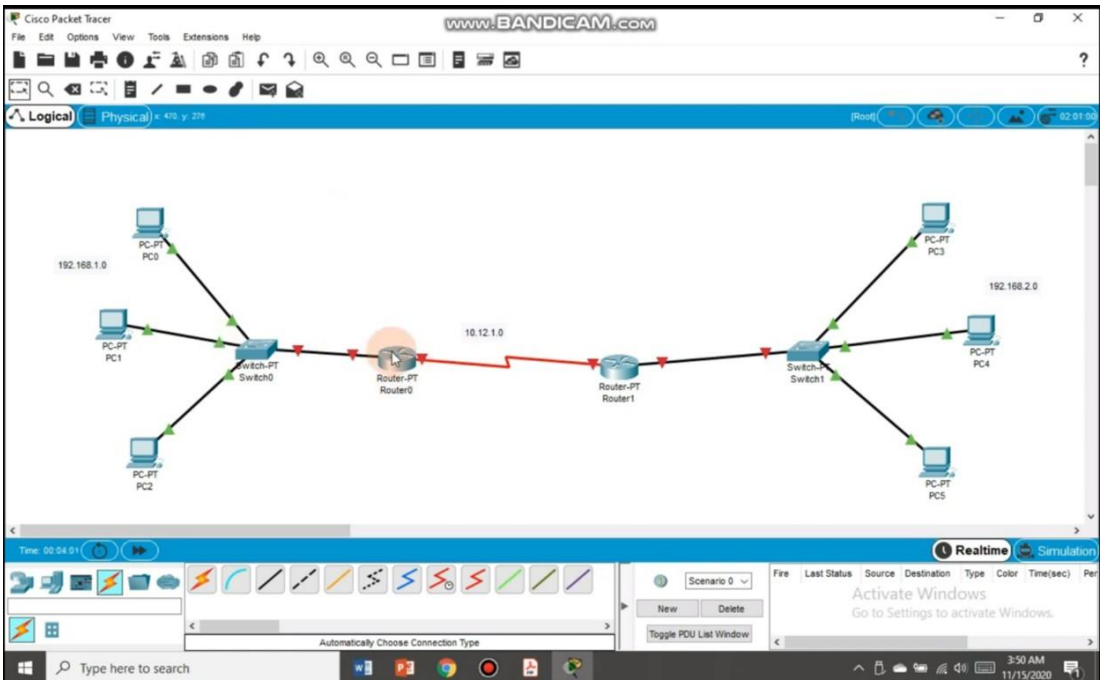
Procedure Step I:
Make a network using 3 end devices and connect them using switch and connect switch to router

Similarly create another network and connect both networks using serial wires.



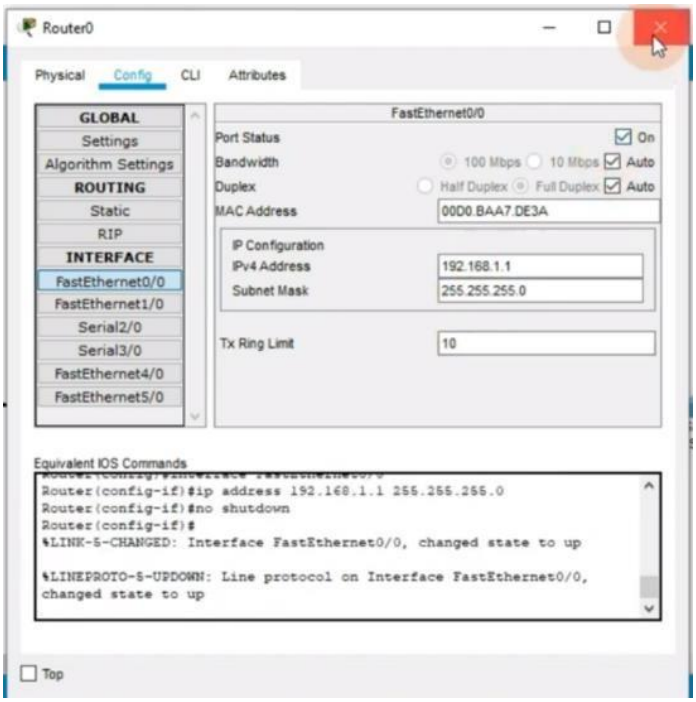
Step2-

Configure each end device with their respective network IP addresses



Step3-

Configure the router in fast ethernet section with network gateway IP address

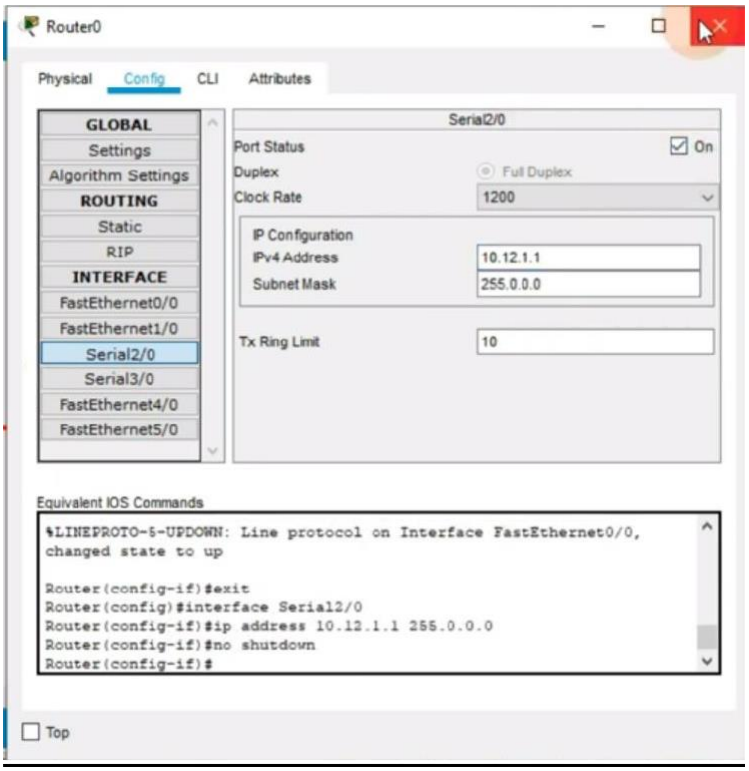


Similarly configure router 2 with its network gateway IP address

Step4-

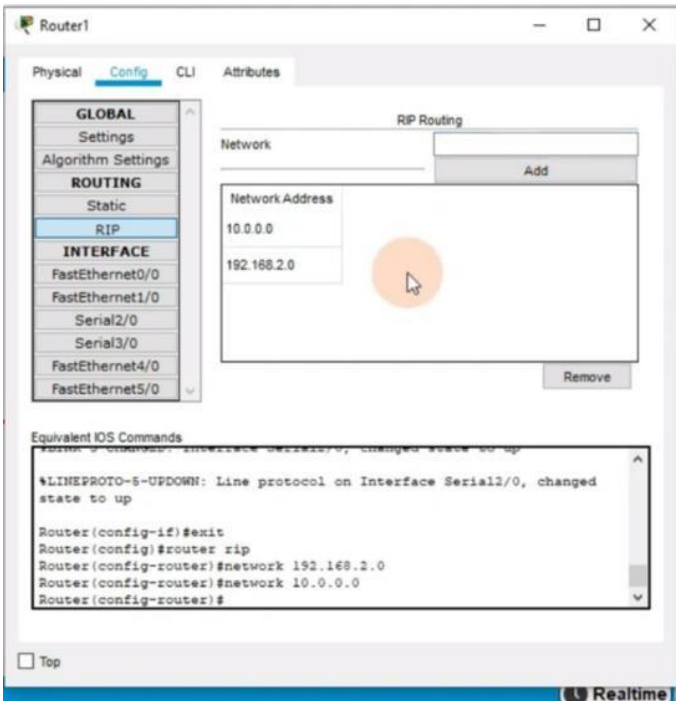
Now in serial 2/0 section configure router to router network ip address and select appropriate clock rate

And make port status on



Similarly do it for router 1

Step5-Configure Routing information in to router 0: add Network IP 192.168.1.0 and 10.0.0.0

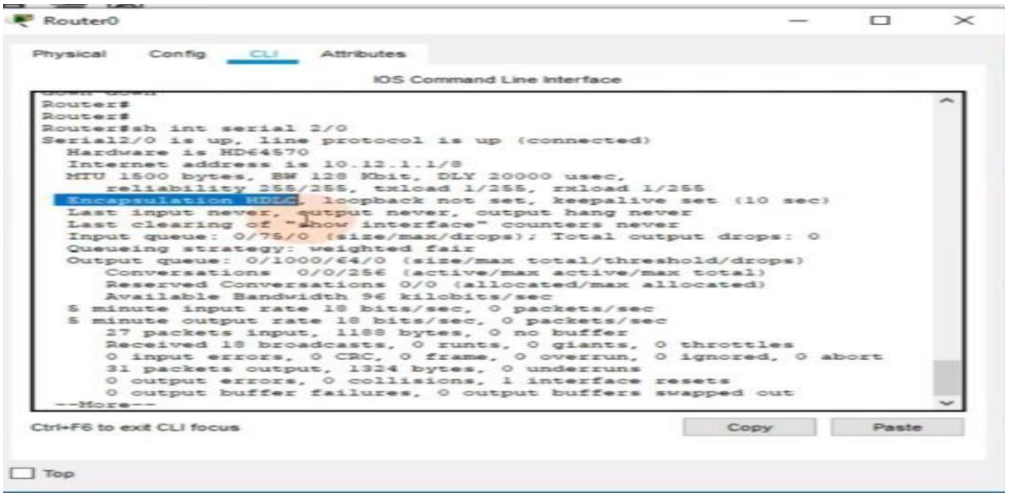


Similarly do it for router 1Network IP-192.168.1.0 and 10.0.0.0

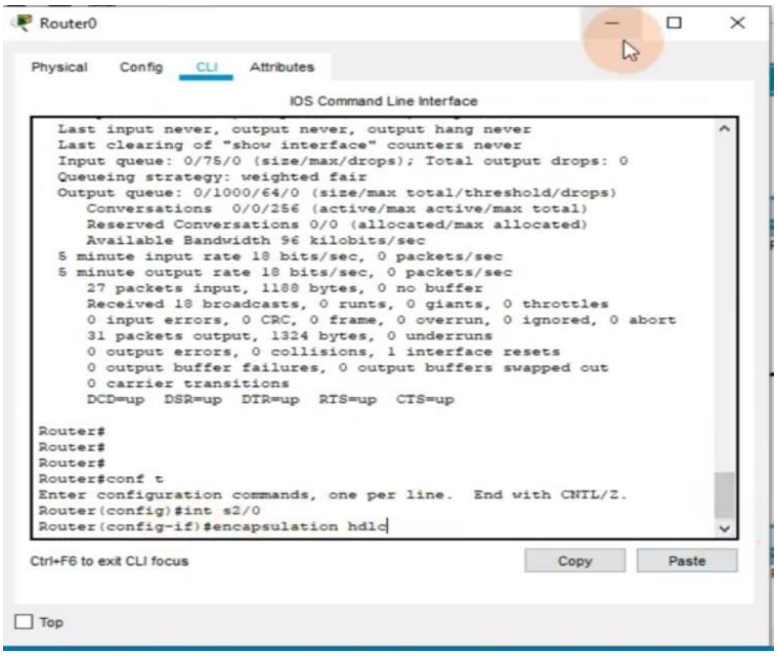
Step6-

Check weather HDLC encapsulation is active in both the router using command line interface of routers

Use command Sh int serial 2/0



Step 7-



In case HDLC encapsulation is not set in your router run configuration commands on CLI window
Conf t
Int S2/0
Encapsulation HDLC
Note do not run if already configured
Similarly do it for router 1

Conclusion:

EXPERIMENT NO.6

TITLE: Socket Programming in python(C/C++) on TCP Client, TCP Server.

OBJECTIVE: To study configuration of socket programming

AIM: To practice the communication between client and server.

SOFTWARE USED: Python 3.6 version

THEORY:

Python provides two levels of access to network services. At a low level, you can access the basic socket support in the underlying operating system, which allows you to implement clients and servers for both connection-oriented and connectionless protocols.

Python also has libraries that provide higher-level access to specific application-level network protocols, such as FTP, HTTP, and so on.

This chapter gives you understanding on most famous concept in Networking - Socket Programming.

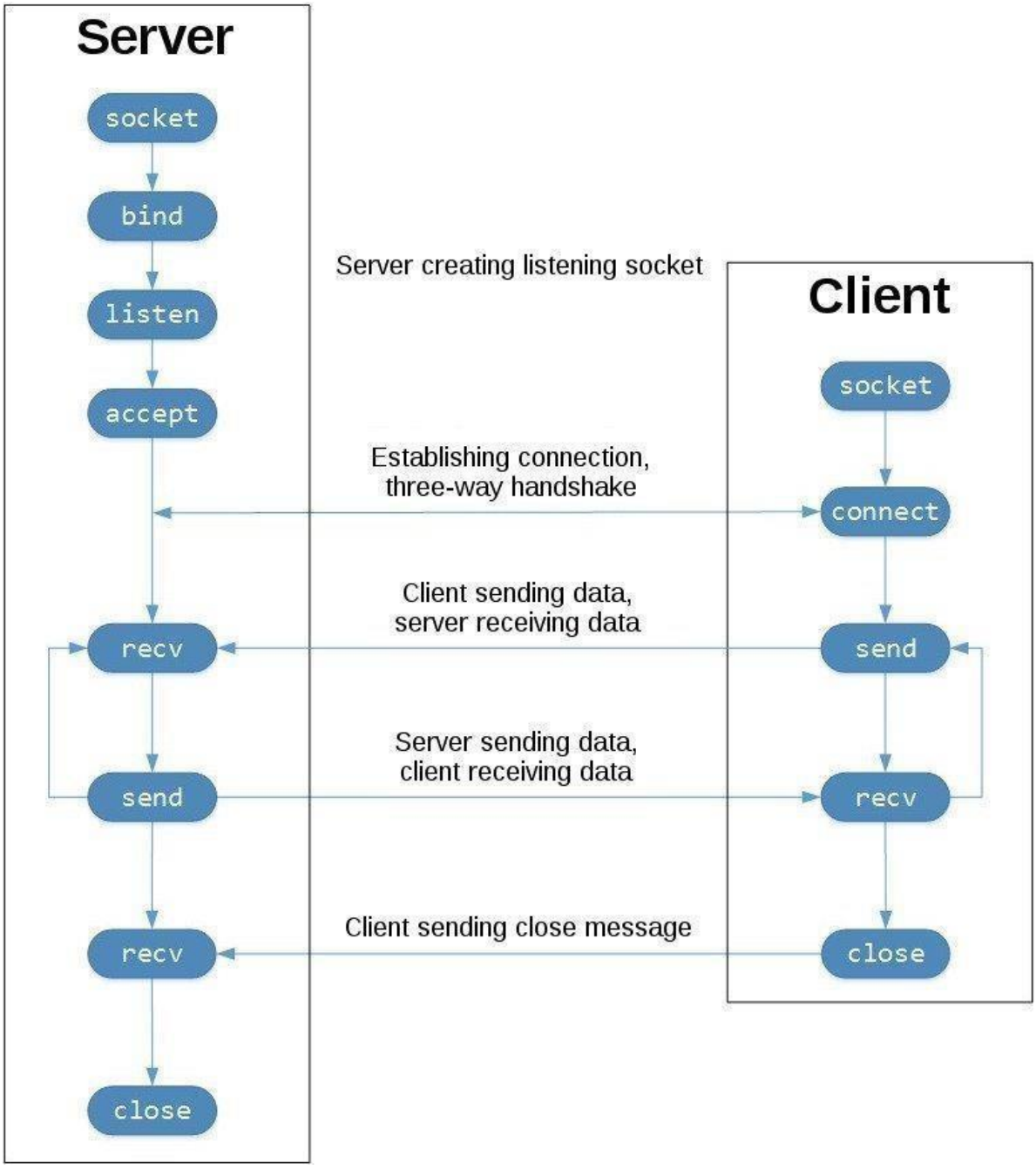
What is Sockets?

Sockets are the endpoints of a bidirectional communications channel. Sockets may communicate within a process, between processes on the same machine, or between processes on different continents.

Sockets may be implemented over a number of different channel types: Unix domain sockets, TCP, UDP, and so on. The *socket* library provides specific classes for handling the common transports as well as a generic interface for handling the rest.

TCP Sockets

In the diagram below, let's look at the sequence of socket API calls and data flow for TCP:



TCP Socket Flow

The left-hand column represents the server. On the right-hand side is the client.

- Starting in the top left-hand column, note the API calls the server makes to setup a “listening” socket:
- socket()

- bind()
- listen()
- accept()

A listening socket does just what it sounds like. It listens for connections from clients. When a client connects, the server calls accept() to accept, or complete, the connection.

The client calls connect() to establish a connection to the server and initiate the three-way handshake. The handshake step is important since it ensures that each side of the connection is reachable in the network, in other words that the client can reach the server and vice-versa. It may be that only one host, client or server, can reach the other.

In the middle is the round-trip section, where data is exchanged between the client and server using calls to send() and recv().

At the bottom, the client and server close() their respective sockets. Sockets have their own vocabulary –

Sr.No.	Term & Description
1	**Domain** The family of protocols that is used as the transport mechanism. These values are constants such as AF_INET, PF_INET, PF_UNIX, PF_X25, and so on.
2	**type** The type of communications between the two endpoints, typically SOCK_STREAM for connection-oriented protocols and SOCK_DGRAM for connectionless protocols.
3	**protocol** Typically zero, this may be used to identify a variant of a protocol within a domain and type.

4	**hostname** The identifier of a network interface − • A string, which can be a host name, a dotted-quad address, or an IPV6 address in colon (and possibly dot) notation • A string "<broadcast>", which specifies an INADDR_BROADCAST address. • A zero-length string, which specifies INADDR_ANY, or • An Integer, interpreted as a binary address in host byte order.
5	**port** Each server listens for clients calling on one or more ports. A port may be a Fixnum port number, a string containing a port number, or the name of a service.

The socket *Module*

To create a socket, you must use the *socket.socket()* function available in *socket* module, which has the general syntax −

```
s = socket.socket (socket_family, socket_type, protocol=0)
```

Here is the description of the parameters −

- **socket_family** − This is either AF_UNIX or AF_INET, as explained earlier.
- **socket_type** − This is either SOCK_STREAM or SOCK_DGRAM.
- **protocol** − This is usually left out, defaulting to 0.

Once you have *socket* object, then you can use required functions to create your client or server program. Following is the list of functions required −

Server Socket Methods

Sr.No.	Method & Description
1	**s.bind()** This method binds address (hostname, port number pair) to socket.
2	**s.listen()** This method sets up and start TCP listener.
3	**s.accept()** This passively accept TCP client connection, waiting until connection arrives (blocking).

Client Socket Methods

Sr.No.	Method & Description
1	**s.connect()-** This method actively initiates TCP server connection.

General Socket Methods

Sr.No.	Method & Description
1	**s.recv()** This method receives TCP message
2	**s.send()** This method transmits TCP message

3	**s.recvfrom()** This method receives UDP message
4	**s.sendto()** This method transmits UDP message
5	**s.close()** This method closes socket
6	**socket.gethostname()** Returns the hostname.

A Simple Server

To write Internet servers, we use the **socket** function available in socket module to create a socket object. A socket object is then used to call other functions to setup a socket server.

Now call **bind(hostname, port)** function to specify a *port* for your service on the given host.Next, call the *accept* method of the returned object. This method waits until a client connects to the port you specified, and then returns a *connection* object that represents the connection to that client.

```
#!/usr/bin/python          # This is server.py file

import socket              # Import socket module

s = socket.socket()        # Create a socket object
```

```
host = socket.gethostname() # Get local machine name
port = 12345               # Reserve a port for your service.
s.bind((host, port))       # Bind to the port

s.listen(5)                # Now wait for client connection.
while True:
   c, addr = s.accept()    # Establish connection with client.
   print 'Got connection from', addr
   c.send('Thank you for connecting')
   c.close()               # Close the connection
```

A Simple Client

Let us write a very simple client program which opens a connection to a given port 12345 and given host. This is very simple to create a socket client using Python's socket module function.

The socket.connect (hosname, port) opens a TCP connection to hostname on the port. Once you have a socket open, you can read from it like any IO object. When done, remember to close it, as you would close a file.

The following code is a very simple client that connects to a given host and port, reads any available data from the socket, and then exits −

```
#!/usr/bin/python          # This is client.py file

import socket              # Import socket module

s = socket.socket()        # Create a socket object
host = socket.gethostname() # Get local machine name
port = 12345               # Reserve a port for your service.

s.connect((host, port))
print s.recv(1024)
s.close()                  # Close the socket when done
```

Now run this server.py in background and then run above client.py to see the result.

```
# Following would start a server in background.
$ python server.py &

# Once server is started run client as follows:
$ python client.py
```

```
Got connection from ('127.0.0.1', 48437)
Thank you for connecting
```

This would produce following result

Output :

Server.py file

```
import socket
s=socket.socket()
print('socket is created' )
s.bind(('localhost',9999))
s.listen(3)
print('waiting for connections')
while True:

    c,addr=s.accept()
    print("connected with",addr)
    c.send(bytes('welcome to AISSMS IOIT','Utf-8'))
    c.close()
```

```
# This is server.py file
import socket                          # Import socket module
s=socket.socket()                      #  Create  a  socket
object print('socket is created' )
s.bind(('localhost',9999))             # Bind to the port on the given host
s.listen(3)                            #  Now  wait  for  client
connection. print('waiting for connections')
while True:
  c,addr=s.accept()                    # Establish connection with client.
  print("connected with",addr)
  c.send(bytes('welcome to AISSMS IOIT','Utf-8'))  # Transmits TCP message to receiver
  c.close()                            # Close the connection
```

client.py

```
import socket
c =socket.socket()
c.connect(('localhost',9999))
print(c.recv(1024).decode())
```

```
# This is client.py file

import socket                  # Import socket module
c =socket.socket()             #  Create  a  socket  object
c.connect(('localhost',9999))  #  Initiates  TCP  server
connection. print(c.recv(1024).decode())
```

Now run this server.py in background and then run above client.py to see the result.

Output of server.py

C:\Users\sandh\pythonProject\venv\Scripts\python.exe
C:/Users/sandh/pythonProject/server.py socket is created
waiting for connections
connected with ('127.0.0.1', 63138)

Output of Client.py

C:\Users\sandh\pythonProject\venv\Scripts\python.exe
C:/Users/sandh/pythonProject/client.py welcome to AISSMS IOIT

Process finished with exit code 0.

Conclusion:

Experiment NO .7

TITLE-Congestion control using Leaky Bucket algorithm

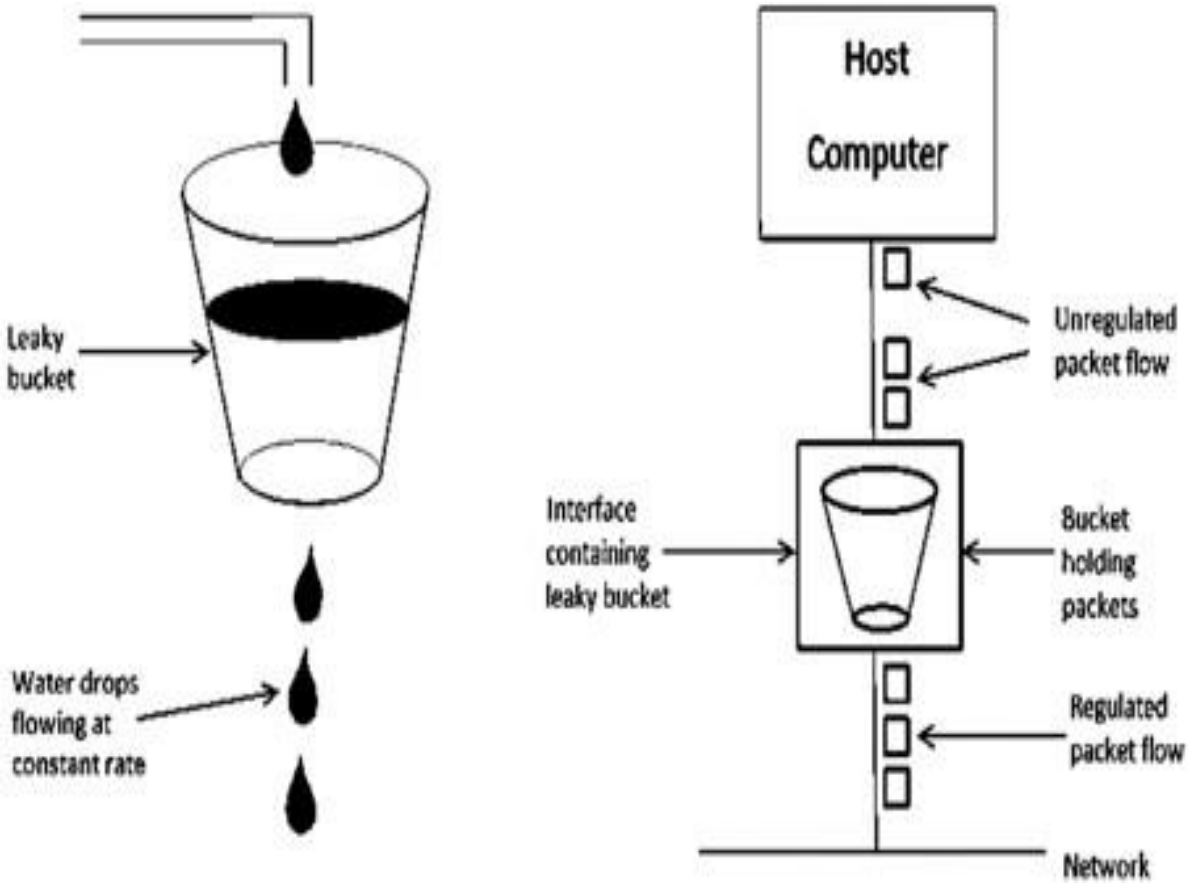
AIM-To write a program for congestion control using Leaky Bucket algorithm

SOFTWARE USED-

THEORY-

The congesting control algorithms are basically divided into two groups: open loop and closed loop. Open loop solutions attempt to solve the problem by good design, in essence, to make sure it does not occur in the first place. Once the system is up and running, midcourse corrections are not made. Open loop algorithms are further divided into ones that act at source versus ones that act at the destination. In contrast, closed loop solutions are based on the concept of a feedback loop if there is any congestion. Closed loop algorithms are also divided into two sub categories: explicit feedback and implicit feedback. In explicit feedback algorithms, packets are sent back from the point of congestion to warn the source. In implicit algorithm, the source deduces the existence of congestion by making local observation, such as the time needed for acknowledgment to come back. The presence of congestion means that the load is (temporarily) greater than the resources (in part of the system) can handle. For subnets that use virtual circuits internally, these methods can be used at the network layer. Another open loop method to help manage congestion is forcing the packet to be transmitted at a more predictable rate. This approach to congestion management is widely used in ATM networks and is called traffic shaping.

The other method is the leaky bucket algorithm. Each host is connected to the network by an interface containing a leaky bucket, that is, a finite internal queue. If a packet arrives at the queue when it is full, the packet is discarded. In other words, if one or more process are already queued, the new packet is unceremoniously discarded. This arrangement can be built into the hardware interface or simulate d by the host operating system. In fact it is nothing other than a single server queuing system with constant service time. The host is allowed to put one packet per clock tick onto the network. This mechanism turns an uneven flow of packet from the user process inside the host into an even flow of packet onto the network, smoothing out bursts and greatly reducing the chances of congestion.



Algorithm

1. Start
2. Set the bucket size or the buffer size,
3. Set the output rate.
4. Transmit the packets such that there is no overflow.
5. Repeat the process of transmission until all packets are transmitted. (Reject packets where its size is greater than the bucket size)
6. Stop

Code

main.c

Run

```c
#include<stdio.h>
int main(){
int incoming, outgoing, buck_size, n, store=0;
printf("Enter bucket size, outgoing rate and no of inputs: ");
scanf("%d %d %d", &buck_size, &outgoing, &n);
while (n!=0){
printf("Enter the incoming packet size: ");
scanf("%d", &incoming);
printf("Incoming packet size %d\n", incoming);
if (incoming <= (buck_size-store)){
store += incoming;
printf("bucket buffer size %d out of %d\n",store,buck_size);
}else{
printf("dropped %d no of packet\n",incoming-(buck_size-store));
printf("Bucket buffer size %d out of %d\n");
store=buck_size;
}
store=store-outgoing;
printf("after outgoing %d packet left out of %d in buffer \n", store, buck_size);
n--;
}
}
```

Output

Output

Clear

```
/tmp/kpEsF6NAF1.o
Enter bucket size, outgoing rate and no of inputs: 32 10 3
Enter the incoming packet size: 12
Incoming packet size 12
bucket buffer size 12 out of 32
after outgoing 2 packet left out of 32 in buffer
Enter the incoming packet size: 4
Incoming packet size 4
bucket buffer size 6 out of 32
after outgoing -4 packet left out of 32 in buffer
Enter the incoming packet size: 14
Incoming packet size 14
bucket buffer size 10 out of 32
after outgoing 0 packet left out of 32 in buffer
_
```

Conclusion:

Experiment NO: 8

TITLE-Analyzing working of protocols

AIM-To observe and note the working of protocols using PING / TRACEROUTE / PATHPING and capture packets in LAN using packet capture and analysis tools.

THEORY-

Traceroute, Ping, and PathPing are network tools or utilities that use the ICMP protocol to perform testing to diagnose issues on a network. Internet Control Message Protocol (ICMP) is an error reporting and diagnostic utility. ICMPs are used by routers, intermediary devices, or hosts to communicate updates or error information to other routers, intermediary devices, or hosts.

1. *Ping:*

The ping command sends an echo request to a host available on the network. Using this command, you can check if your remote host is responding well or not. Tracking and isolating hardware and software problems, determining the status of the network and various foreign hosts. The ping command is usually used as a simple way to verify that a computer can communicate over the network with another computer or network device. The ping command operates by sending Internet Control Message Protocol (ICMP) Echo Request messages to the destination computer and waiting for a response.

2. *Trace route:*

Traceroute uses Internet Control Message Protocol (ICMP) echo packets with variable time to live (TTL) values. The response time of each hop is calculated. To guarantee accuracy, each hop is queried multiple times (usually three times) to better measure the response of that particular hop. Traceroute is a network diagnostic tool used to track the pathway taken by a packet on an IP network from source to destination. Traceroute also records the time taken for each hop the packet makes during its route to the destination. Traceroute uses Internet Control Message Protocol (ICMP) echo packets with variable time to live (TTL) values.
The response time of each hop is calculated. To guarantee accuracy, each hop is

queried multiple times (usually three times) to better measure the response of that particular hop. Traceroute sends packets with TTL values that gradually increase from packet to packet. starting with TTL value of one. Routers decrement TTL values of packets by one when routing and discard packets whose TTL value has reached zero, returning the ICMP error message ICMP Time Exceeded. For the first set of packets, the first router receives the packet, decrements the TTL value and drops the packet because it then has TTL value zero. The router sends an ICMP Time Exceeded message back to the source. The next set of packets are given a TTL value of two, so the first router forwards the packets, but the second router drops them and replies with ICMP Time Exceeded. Proceeding in this way, traceroute uses the returned ICMP Time Exceeded messages to build a list of routers that packets traverse, until the destination is reached and returns an ICMP Echo Reply message. With the tracert command shown above, we're asking tracert to show us the path from the local computer all the way to the network device with the hostname.

3. *Pathping:*

This network utility is a more advanced version of the Ping tool, which performs a ping to cach hop along the route to the destination (unlike Ping, which just pings from the originating device to the destination device). It is extremely useful in diagnosing packet loss, and can help with diagnosing slow speed faults.

To Pathping a device, proceed as follows.

1. Open a Windows Command Prompt window.

2. At the command prompt, type, pathping <IP address/Website url>

4. ICMP:

ICMP or Internet Control Message Protocol is Internet or Network layer protocol; it is used to check the reachability of a host or router in a network.

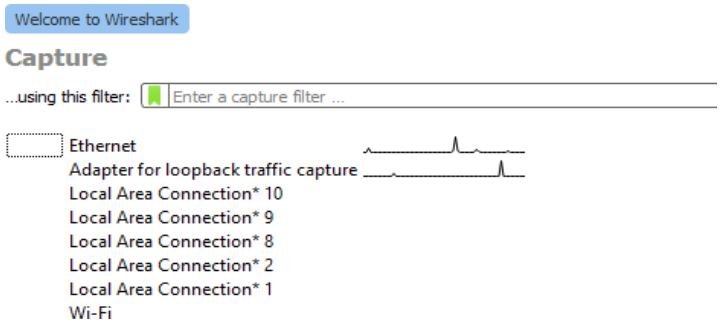
PROCEDURE:

1. Set up the LAN:

   •Connect multiple devices (e.g., computers, laptops) in a LAN network. Ensure that all devices are connected to the same LAN and can communicate with each other.

2. Launch Wireshark:

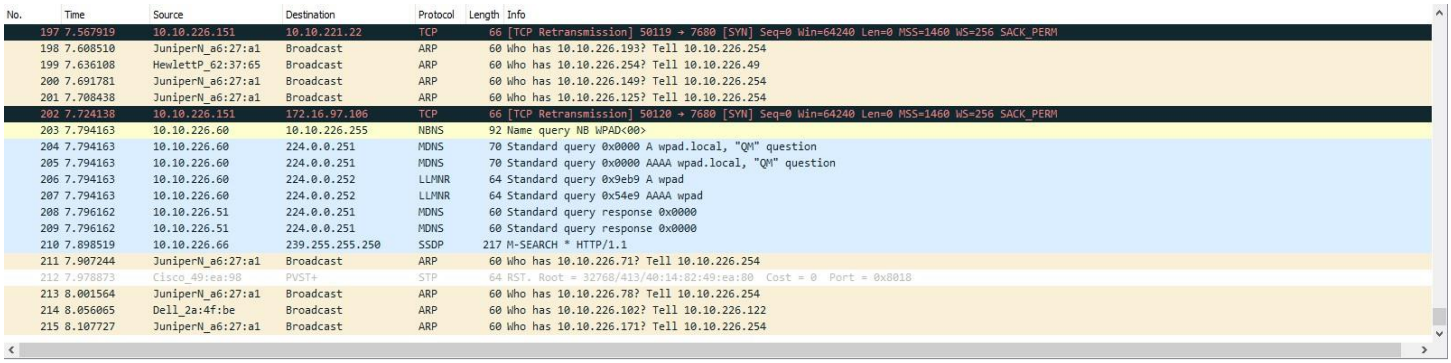
   •Install Wireshark on a computer within the LAN network.



   •Launch Wireshark and select the network interface that is connected to the LAN.

3. Start Packet Capture:

   •Click on the "Capture" button (the one with a shark fin icon) in Wireshark to start capturing packets on the selected interface.



No.	Time	Source	Destination	Protocol	Length	Info
197	7.567919	10.10.226.151	10.10.221.22	TCP	66	[TCP Retransmission] 50119 → 7680 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
198	7.608510	JuniperN_a6:27:a1	Broadcast	ARP	60	Who has 10.10.226.193? Tell 10.10.226.254
199	7.636108	HewlettP_62:37:65	Broadcast	ARP	60	Who has 10.10.226.254? Tell 10.10.226.49
200	7.691781	JuniperN_a6:27:a1	Broadcast	ARP	60	Who has 10.10.226.149? Tell 10.10.226.254
201	7.708438	JuniperN_a6:27:a1	Broadcast	ARP	60	Who has 10.10.226.125? Tell 10.10.226.254
202	7.724138	10.10.226.151	172.16.97.106	TCP	66	[TCP Retransmission] 50120 → 7680 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
203	7.794163	10.10.226.60	10.10.226.255	NBNS	92	Name query NB WPAD<00>
204	7.794163	10.10.226.60	224.0.0.251	MDNS	70	Standard query 0x0000 A wpad.local, "QM" question
205	7.794163	10.10.226.60	224.0.0.251	MDNS	70	Standard query 0x0000 AAAA wpad.local, "QM" question
206	7.794163	10.10.226.60	224.0.0.252	LLMNR	64	Standard query 0x9eb9 A wpad
207	7.794163	10.10.226.60	224.0.0.252	LLMNR	64	Standard query 0x54e9 AAAA wpad
208	7.796162	10.10.226.51	224.0.0.251	MDNS	60	Standard query response 0x0000
209	7.796162	10.10.226.51	224.0.0.251	MDNS	60	Standard query response 0x0000
210	7.898519	10.10.226.66	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
211	7.907244	JuniperN_a6:27:a1	Broadcast	ARP	60	Who has 10.10.226.71? Tell 10.10.226.254
212	7.978873	Cisco_49:ea:98	PVST+	STP	64	RST. Root = 32768/413/40:14:82:49:ea:80 Cost = 0 Port = 0x8018
213	8.001564	JuniperN_a6:27:a1	Broadcast	ARP	60	Who has 10.10.226.78? Tell 10.10.226.254
214	8.056065	Dell_2a:4f:be	Broadcast	ARP	60	Who has 10.10.226.102? Tell 10.10.226.122
215	8.107727	JuniperN_a6:27:a1	Broadcast	ARP	60	Who has 10.10.226.171? Tell 10.10.226.254

4. Capture Packets during PING:

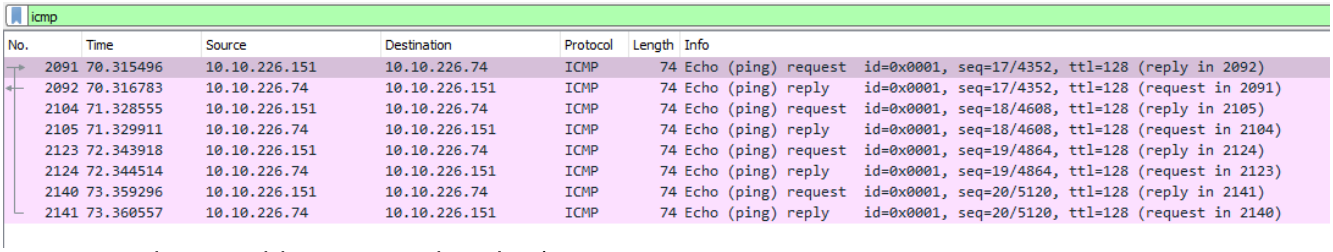
   •Open the command prompt/terminal on a device within the LAN.

   •Use the PING command to send ICMP Echo Request packets to another device in the LAN.

   •For example, on a Windows system, type: ping <IP_address> (replace <IP_address> with the IP address of the target device).

```
Command Prompt
Microsoft Windows [Version 10.0.19045.3570]
(c) Microsoft Corporation. All rights reserved.

C:\Users\admin>ping 10.10.226.74

Pinging 10.10.226.74 with 32 bytes of data:
Reply from 10.10.226.74: bytes=32 time=1ms TTL=128
Reply from 10.10.226.74: bytes=32 time=1ms TTL=128
Reply from 10.10.226.74: bytes=32 time<1ms TTL=128
Reply from 10.10.226.74: bytes=32 time=1ms TTL=128

Ping statistics for 10.10.226.74:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\Users\admin>
```

- While the PING command is running, Wireshark will capture the ICMP packets



exchanged between the devices.

5. Capture Packets during TRACEROUTE:

- Open the command prompt/terminal on a device within the LAN.
- Use the TRACEROUTE command to trace the route to another device in the LAN.
- For example, on a Windows system, type: tracert <IP_address> (replace

```
C:\Users\admin>tracert 10.10.226.74

Tracing route to DESKTOP-P52R3QV [10.10.226.74]
over a maximum of 30 hops:

  1     1 ms     1 ms     1 ms  DESKTOP-P52R3QV [10.10.226.74]

Trace complete.

C:\Users\admin>
```

<IP_address> with the IP address of the target device).

- While the TRACEROUTE command is running, Wireshark will capture the

ICMP packets with increasing TTL values as they traverse through the network.

icmp

No.	Time	Source	Destination	Protocol	Length	Info
349	23.347962	10.10.226.151	10.4.122.117	ICMP	106	Echo (ping) request id=0x0001, seq=138/35328, ttl=1 (no response found!)
350	23.348994	10.10.226.254	10.10.226.151	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
351	23.349356	10.10.226.151	10.4.122.117	ICMP	106	Echo (ping) request id=0x0001, seq=139/35584, ttl=1 (no response found!)
352	23.353785	10.10.226.254	10.10.226.151	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
353	23.354108	10.10.226.151	10.4.122.117	ICMP	106	Echo (ping) request id=0x0001, seq=140/35840, ttl=1 (no response found!)
354	23.355022	10.10.226.254	10.10.226.151	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
365	23.530986	10.10.226.254	10.10.226.151	ICMP	70	Destination unreachable (Port unreachable)
392	25.045861	10.10.226.254	10.10.226.151	ICMP	70	Destination unreachable (Port unreachable)
408	26.558631	10.10.226.254	10.10.226.151	ICMP	70	Destination unreachable (Port unreachable)
448	29.087688	10.10.226.151	10.4.122.117	ICMP	106	Echo (ping) request id=0x0001, seq=141/36096, ttl=2 (no response found!)
449	29.088559	10.13.46.169	10.10.226.151	ICMP	134	Time-to-live exceeded (Time to live exceeded in transit)
450	29.089987	10.10.226.151	10.4.122.117	ICMP	106	Echo (ping) request id=0x0001, seq=142/36352, ttl=2 (no response found!)
451	29.090801	10.13.46.169	10.10.226.151	ICMP	134	Time-to-live exceeded (Time to live exceeded in transit)
452	29.091472	10.10.226.151	10.4.122.117	ICMP	106	Echo (ping) request id=0x0001, seq=143/36608, ttl=2 (no response found!)
453	29.092308	10.13.46.169	10.10.226.151	ICMP	134	Time-to-live exceeded (Time to live exceeded in transit)
587	34.805097	10.10.226.151	10.4.122.117	ICMP	106	Echo (ping) request id=0x0001, seq=144/36864, ttl=3 (no response found!)
588	34.817066	10.13.46.218	10.10.226.151	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
589	34.818187	10.10.226.151	10.4.122.117	ICMP	106	Echo (ping) request id=0x0001, seq=145/37120, ttl=3 (no response found!)
590	34.825892	10.13.46.218	10.10.226.151	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)

1. Capture Packets during PATHPING:

- Open the command prompt/terminal on a device within the LAN.
- Use the PATHPING command to trace the route and measure the round-trip time to another device in the LAN.

```
C:\Users\admin>pathping 10.10.226.74

Tracing route to DESKTOP-P52R3QV [10.10.226.74]
over a maximum of 30 hops:
  0  DESKTOP-MTVU9JE.Unitech.local [10.10.226.151]
  1  DESKTOP-P52R3QV [10.10.226.74]

Computing statistics for 25 seconds...
            Source to Here   This Node/Link
Hop  RTT    Lost/Sent = Pct  Lost/Sent = Pct  Address
  0                                           DESKTOP-MTVU9JE.Unitech.local [10.10.226.151]
                                    0/ 100 =  0%   |
  1    1ms     0/ 100 =  0%     0/ 100 =  0%  DESKTOP-P52R3QV [10.10.226.74]

Trace complete.
```

- For example, on a Windows system, type: pathping <IP_address> (replace <IP_address> with the IP address of the target device).

icmp

No.	Time	Source	Destination	Protocol	Length	Info
161	9.360087	10.10.226.151	10.10.226.74	ICMP	106	Echo (ping) request id=0x0001, seq=37/9472, ttl=1 (reply in 162)
162	9.361737	10.10.226.74	10.10.226.151	ICMP	106	Echo (ping) reply id=0x0001, seq=37/9472, ttl=128 (request in 161)
169	9.364819	10.10.226.151	10.10.226.74	ICMP	106	Echo (ping) request id=0x0001, seq=38/9728, ttl=1 (reply in 170)
170	9.366376	10.10.226.74	10.10.226.151	ICMP	106	Echo (ping) reply id=0x0001, seq=38/9728, ttl=128 (request in 169)
174	9.619023	10.10.226.151	10.10.226.74	ICMP	106	Echo (ping) request id=0x0001, seq=39/9984, ttl=1 (reply in 175)
175	9.620545	10.10.226.74	10.10.226.151	ICMP	106	Echo (ping) reply id=0x0001, seq=39/9984, ttl=128 (request in 174)
180	9.884743	10.10.226.151	10.10.226.74	ICMP	106	Echo (ping) request id=0x0001, seq=40/10240, ttl=1 (reply in 181)
181	9.885866	10.10.226.74	10.10.226.151	ICMP	106	Echo (ping) reply id=0x0001, seq=40/10240, ttl=128 (request in 180)
187	10.150136	10.10.226.151	10.10.226.74	ICMP	106	Echo (ping) request id=0x0001, seq=41/10496, ttl=1 (reply in 188)
188	10.151629	10.10.226.74	10.10.226.151	ICMP	106	Echo (ping) reply id=0x0001, seq=41/10496, ttl=128 (request in 187)
193	10.415777	10.10.226.151	10.10.226.74	ICMP	106	Echo (ping) request id=0x0001, seq=42/10752, ttl=1 (reply in 194)
194	10.416438	10.10.226.74	10.10.226.151	ICMP	106	Echo (ping) reply id=0x0001, seq=42/10752, ttl=128 (request in 193)
200	10.681201	10.10.226.151	10.10.226.74	ICMP	106	Echo (ping) request id=0x0001, seq=43/11008, ttl=1 (reply in 201)
201	10.681715	10.10.226.74	10.10.226.151	ICMP	106	Echo (ping) reply id=0x0001, seq=43/11008, ttl=128 (request in 200)
206	10.946815	10.10.226.151	10.10.226.74	ICMP	106	Echo (ping) request id=0x0001, seq=44/11264, ttl=1 (reply in 207)
207	10.947419	10.10.226.74	10.10.226.151	ICMP	106	Echo (ping) reply id=0x0001, seq=44/11264, ttl=128 (request in 206)
211	11.212483	10.10.226.151	10.10.226.74	ICMP	106	Echo (ping) request id=0x0001, seq=45/11520, ttl=1 (reply in 212)
212	11.213015	10.10.226.74	10.10.226.151	ICMP	106	Echo (ping) reply id=0x0001, seq=45/11520, ttl=128 (request in 211)
219	11.478080	10.10.226.151	10.10.226.74	ICMP	106	Echo (ping) request id=0x0001, seq=46/11776, ttl=1 (reply in 220)

CONCLUSION:

Experiment NO 9

TITLE: Executing proxy server using simulator

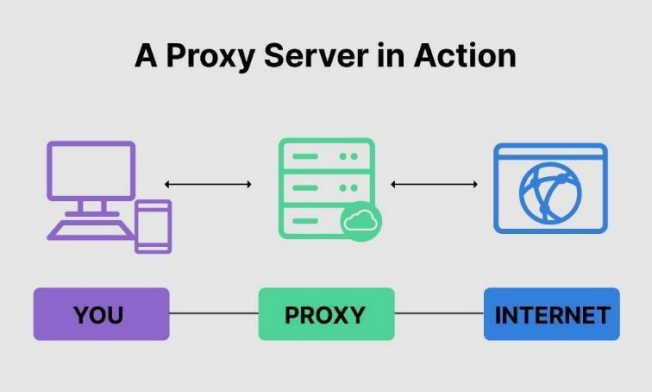
AIM : Designing and simulating proxy server on cisco packet tracer and observing its working

SOFTWARE USED: Cisco packet tracer

THEORY:

Proxy Server Overview:

A proxy server is an intermediary server that plays a critical role in the transmission of data between a client (e.g., a computer, smartphone) and a destination server (e.g., a web server). It functions as a gateway, receiving requests from clients and forwarding those requests to the target server. The target server processes the request and sends a response back to the proxy server, which, in turn, delivers the response to the client.



Purposes of Proxy Servers:

Caching: Proxy servers can store copies of frequently requested resources like web pages, images, and files. By doing so, they reduce the load on the destination server and accelerate content delivery to clients. This is especially useful in networks with limited bandwidth.

Security: Proxy servers enhance network security by serving as a barrier between the client and the internet. They can inspect, filter, or block network traffic, thereby providing an extra layer of protection against malicious content, viruses, and cyber threats.

Anonymity and Privacy: Proxy servers can mask the client's IP address, making it difficult for websites or services to trace a user's identity or location. This feature is valuable for individuals who want to browse the internet anonymously.

Content Filtering: Proxy servers are commonly used in corporate or educational settings to filter and control access to specific websites or types of content. Administrators can implement content policies to restrict or allow access based on organizational rules.

Load Balancing: In the case of reverse proxy servers, they distribute incoming client requests across multiple backend servers. This load balancing improves system reliability and ensures even distribution of workloads.

Types of Proxy Servers:

Forward Proxy: A forward proxy is used by clients to access the internet indirectly. It's often employed in corporate networks to control and monitor outbound traffic and enforce security policies.

Reverse Proxy: Reverse proxies are placed in front of one or more destination servers.

They act as a gateway for incoming client requests and can handle tasks such as load

balancing, SSL termination, and serving as a single entry point to multiple backend servers.

Transparent Proxy: Transparent proxies intercept network traffic without the client's knowledge. They are frequently used for caching and content filtering in network infrastructure.

Anonymous Proxy: These proxies conceal the client's IP address from the destination server, providing a level of anonymity while browsing the web.

Configuration and Operation:

Proxy servers are configured to route and filter traffic based on specific rules and policies. They mainly operate at the application layer (Layer 7) of the OSI model, allowing them to inspect and modify the content of HTTP requests and responses. Clients can be configured to route their traffic through the proxy server.

Advantages and Disadvantages:

Proxy servers offer several advantages, such as improved security, privacy, and network performance. However, they can also introduce complexity, potential single points of failure, and require proper configuration and maintenance.

Understanding the theory of proxy servers is essential for effectively utilizing their capabilities in different network scenarios, from boosting performance to enhancing security and privacy. Network administrators and IT professionals typically configure and manage proxy servers to meet specific network requirements.

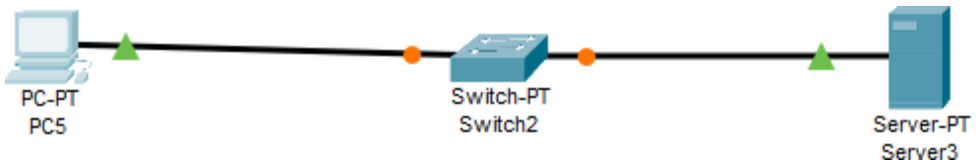
PROCEDURE:

In Packet Tracer, a network simulation tool developed by Cisco, you can create a simple web server using a generic server device and configure it as a proxy server. Here's a step-by-step guide on how to do it:

1. Open Packet Tracer: If you don't have Packet Tracer installed, you can download it from the Cisco Networking Academy website.

2. Set up the network: Create a network topology by dragging and dropping devices from the device list onto the workspace. For this example, you will need at least three devices: a PC, a server (generic server), and a switch to connect them.

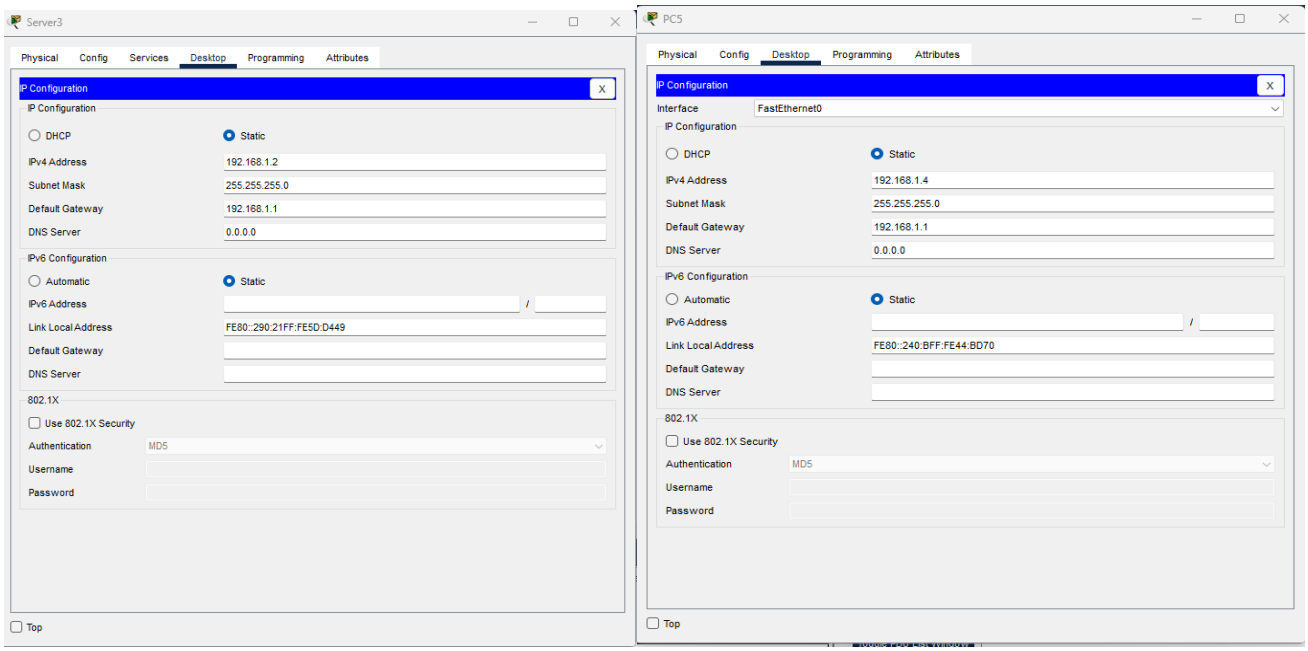
3. Connect the devices: Use the copper straight-through cable to connect the PC to one of the router's LAN ports. Connect the server to another LAN port of the router using the same cable type.



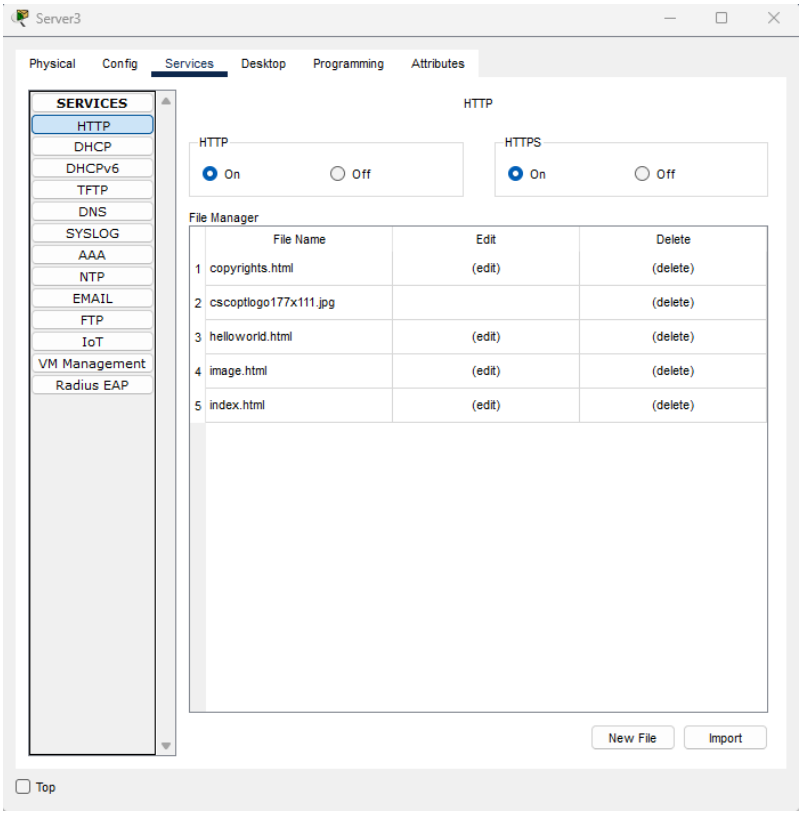
1. Configure IP addresses: Right-click on each device and select "Configure." Set IP addresses on the interfaces connected to the router's LAN ports. For example:

   • PC: IP address 192.168.1.4, subnet mask 255.255.255.0, default gateway 192.168.1.1

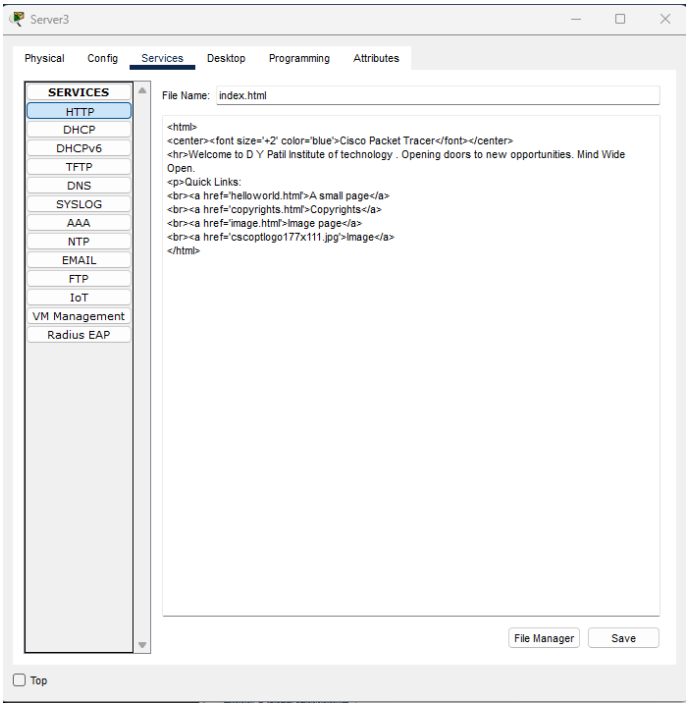
   • Server: IP address 192.168.1.2, subnet mask 255.255.255.0, default gateway 192.168.1.1



2. Set up HTTP service on the server: Double-click on the server to access its configuration. Go to the "Services" tab and enable the HTTP service. This will allow the server to respond to HTTP requests and act as a web server.

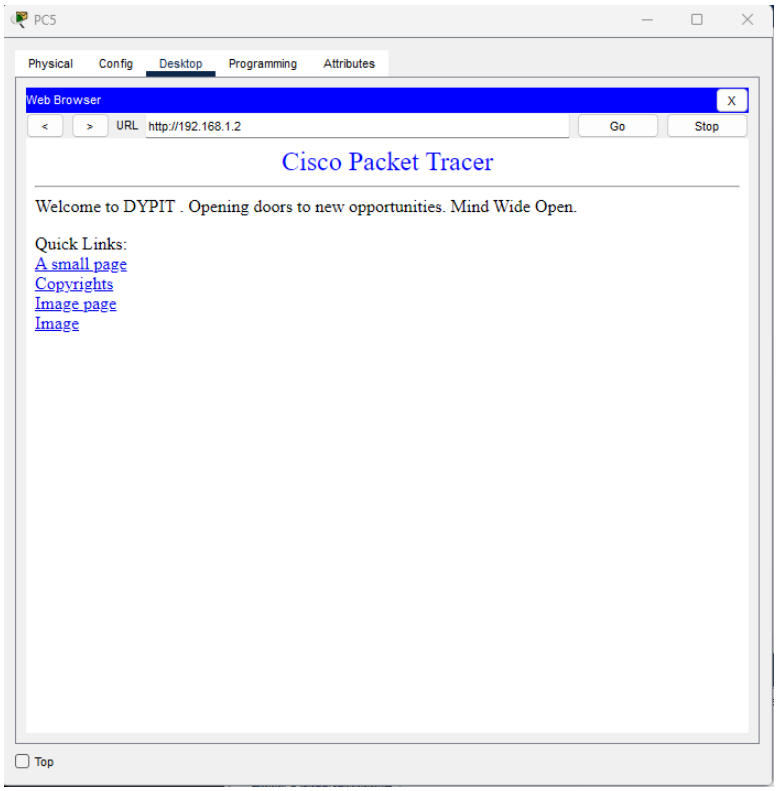


3. You can also configure data on the website by changing scripts of index.html file by clicking on edit option



4. Configure the PC's web browser proxy settings: Open the PC's web browser. Set the HTTP proxy to the IP address of the server (192.168.1.2).

5. Test the setup: Open the web browser on the PC and try to access a website. The PC will send the HTTP requests to the server acting as a proxy. The server will forward the requests to the Internet and return the responses back to the PC through the proxy.


PC5
Physical
Config
Desktop
Programming
Attributes
Web Browser
X
<
>
URL
http://192.168.1.2
Go
Stop
Cisco Packet Tracer
Welcome to DYPIT . Opening doors to new opportunities. Mind Wide Open.
Quick Links:
A small page
Copyrights
Image page
Image
Top

CONCLUSION:

Experiment NO :10

TITLE-Cabling of Network

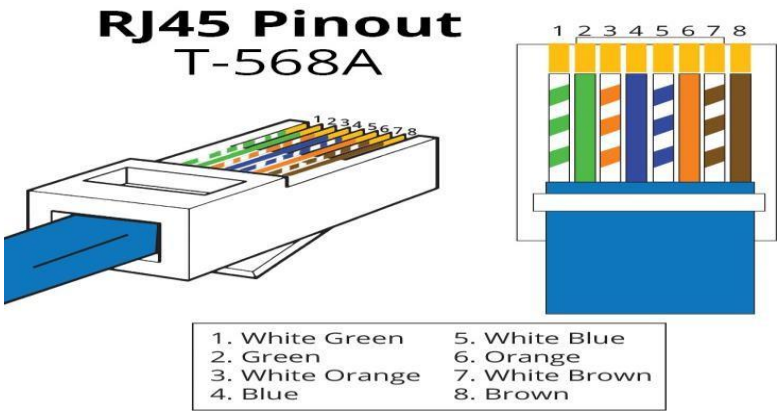
AIM-To do cabling of Network and learn various steps of cabling

1. Cable Crimping
2. Straight Cabling
3. Cross Cabling
4. IO connector crimping
5. Testing crimped cable using a cable tester

HARDWARE- RJ-45 connector, 10 Connector, Crimping Tool, Twisted pair Cable, Cable Tester. Standard Cabling: 1. 10BaseT and 100BaseT, UTP category-5 cable for both modes. A straight cable is used to connect a computer to a hub

RJ45 Pin # (END 1)	Wire Color	Diagram End #1	RJ45 Pin # (END 2)	Wire Color	Diagram End #2
1	White/Orange		1	White/Green	
2	Orange		2	Green	
3	White/Green		3	White/Orange	
4	Blue		4	White/Brown	
5	White/Blue		5	Brown	
6	Green		6	Orange	
7	White/Brown		7	Blue	
8	Brown		8	White/Blue	

Cross Cabling: A cross cable is used to connect 2 computers directly (with ONLY the UTPcable). It is also used to connect 2 hubs with a normal port on both hubs



PROCEDURE:

Cable Crimping:

1. Remove the outmost vinyl shield for 12mm at one end of the cable.

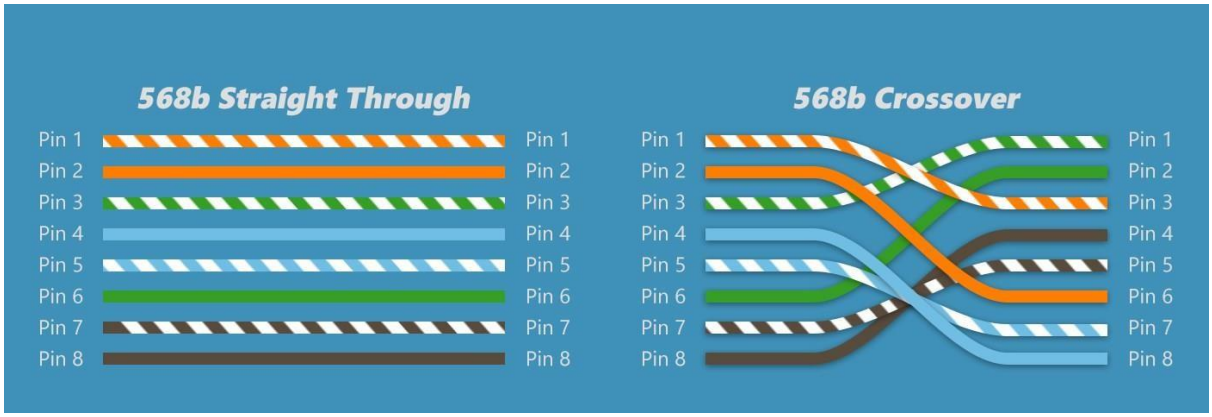
2. Arrange the metal wires in parallel

3. Insert the metal wires into RJ45 connector on keeping the metal wire arrangement.

4. Set the RJ45 connector (with the cable) on the pliers, and squeeze it tightly.

5. Make the other side of the cable in the same way.

6. Once done, no need to take care of the direction of the cable.



I0 connector crimping:

Run the full length of Ethernet cable in place, from endpoint to endpoint, making sure to leave excess. At one end, cut the wire to length leaving enough length to work, but not too much excess. Strip off about 2 inches of the Ethernet cable sheath. Align each of the colored wires according to the layout of the jack. Use the punch down tool to insert each wire into the jack. Repeat the above steps for the second RJ45 jack.

Testing the crimped cable using a cable tester:

1. Skin off the cable jacket 3.0 cm long cable stripper up to cable.
2. Untwist each pair and straighten each wire 190 0.1.5 cm long.
3. Cut all the wires.
4. Insert the wires into the RJ45 connector right white orange left brown the pins facing up.
5. Place the connector into a crimping tool, and squeeze hard so that the handle reaches its full swing.
6. Use a cable tester to test for proper continuity.


3cm
1. Get your RJ45 plug, cable cap, Ethernet cable ready
2. Insert the cable cap
3. Put the Ethernet cable into crimping tool
4. Cut the outer skin
5. Separate the wires
T568B Line order: orange+white | orange | green+white | blue | blue+white | green | brown+white | brown
6. keep the wires neat, T568B is recommended
Reserve a length of about 2.5cm
7. Cut the wires in obluque angle
8. Wear the core wire into the RJ45 plug
push
9. Insert the cable cap into the RJ45 plug
10. Complete as shown
11. Crimp plug
12. Complete

Conclusion: