RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CS19541- COMPUTER NETWORKS LABORATORY

LAB MANUAL

THIRD YEAR

FIFTH SEMESTER

2024- 2025

ODD SEMESTER

CS19541-COMPUTER NETWORKS-LAB MANUAL

List of Experiments

List of Experiments		
1.	**Study of various Network commands used in Linux and Windows:** Hands-on practice of various network commands.	[4]
2.	**Study of Network cables.** 1. Understand different types of Network cables. **2.** Make a cross-wired cable and straight through cable using clamping/crimping tool.	[4]
3.	**Experiments on CISCO PACKET TRACER (Simulation Tool):** **a)** To understand environment of CISCO PACKET TRACER to design simple network. **b)** Analyse the behaviour of network devices using CISCO PACKET TRACER simulator. Design a simple network with multiple nodes and connect via networking devices available in library. Perform simulation and trace communication behaviour of specified network devices. 1: Use only HUB to design a small network having 4 to 6 hosts 2: Use only a Switch to design a small network with 4 to 6 hosts. 3: Use both the device (HUB and SWITCH) for a network and find out functioning difference between switch and hub. **Find out the network topology implemented in your college and draw and label that topology in your observation book.**	[2] [2]
4.	**a) Setup and configure a LAN (Local area network) using a Switch and Ethernet cables in your lab.** 1. Connect 3-4 host machines to a switch. 2. Assign ip addresses to each host machine. 3. Check the connectivity between the machines by using ping command. 4. Share and access files and folder across the machines of the LAN.	[2]

5.	**Experiments on Packet capture tool: Wireshark** To understand the features of wireshark as a packet capture tool and understand **encapsulation of information at various layers of a Protocol stack**.	[4]
6.	**Error Correction at Data Link Layer:** Write a program to implement error detection and correction using HAMMING code concept. Make a test run to input data stream and verify error correction feature.	[4]
7.	**Flow control at Data Link Layer:** Write a program to implement flow control at data link layer using SLIDING WINDOW PROTOCOL. Simulate the flow of frames from one node to another.	[4]
8.	**Virtual LAN**: a) Simulate Virtual LAN configuration using CISCO Packet Tracer Simulation. b) There are 10 faculty in Robotics department sitting in 3 different blocks. Design and configure a Virtual LAN for Robotics department (using switch and Ethernet cables) so that all the faculty are logically in the same LAN. **Wireless LAN:** c) Configuration of Wireless LAN using CISCO Packet Tracer.	[4] [2]
9.	**Implementation of SUBNETTING in CISCO PACKET TRACER simulator.** a) Design multiple subnet with suitable number of hosts. b) Assign static IP address across all subnet and connect the subnets via Router. c) Simulate packet transmission across the subnets and observe the results:- a. When subnets are connected via a router. b. When subnets are not connected without a router.	[4]

10.	**Internetworking with routers in CISCO PACKET TRACER simulator.** **a) Design and configure a simple internetwork using a router.** 1. Design different networks (with 3 to 4 hosts) and connect via Router. 2. Allot static ip address to machines and router interfaces. 3. Perform simulation and trace how routing is done in packet transmission. **b) Design and configure an internetwork using wireless router DHCP server and internet cloud.** **c) Design and configure an inter-network in your lab using switch, router and Ethernet cables.**	[4] [2]
11.	**Routing at Network Layer:** a) Simulate Static Routing Protocol Configuration using CISCO Packet Tracer. b) Simulate RIP using CISCO Packet Tracer.	[4]
12.	**End –End Communication at Transport Layer** a) Implement echo client server using TCP/UDP sockets. b) Implement a chat program using socket programming.	[4]
13.	Implement your own ping program.	[2]
14.	Write a code using RAW sockets to implement packet sniffing.	[4]
15.	Analyse various types of servers using Webalizer tool.	[4]
	Total	**60 hours**

	Additional programs for practice	
1.	**Data Link Layer (Frame Generation):** Write a program to read a stream of data from data file (Having Characters) to create BSC frames by implementing character stuffing concept and inserting control characters. The receiving program must execute on other computer and decode received bytes and write to a file.	
2.	Demonstrate Configuration of Network Address Translation (NAT) and Port Address Translation (PAT) using CISCO Packet Tracer simulation.	
3.	Implement a static routing protocol which also displays the routing table details after every update.	
4.	Implement a dynamic routing protocol which also displays the routing table after every updates.	
5.	Implement FTP server using socket programming.	

CS19541-COMPUTER NETWORKS-LAB MANUAL

Practical -1

AIM: - Study of various Network commands used in Linux and Windows:

BASIC NETWORKING COMMANDS:

arp –a:- ARP is short form of address resolution protocol, It will show the IP address of your computer along with the IP address and MAC address of your router.
hostname: This is the simplest of all TCP/IP commands. It simply displays the name of your computer.
ipconfig /all: This command displays detailed configuration information about your TCP/IP connection including Router, Gateway, DNS, DHCP, and type of Ethernet adapter in your system
nbtstat –a: This command helps solve problems with NetBIOS name resolution. (Nbt stands for NetBIOS over TCP/IP)
netstat: (network statistics) netstat displays a variety of statistics about a computers active TCP/IP connections. It is a command line tool for monitoring network connections both incoming and outgoing as well as viewing routing tables, interface statistics etc. e.g.:- netstat -r
nslookup: (name server lookup) is a tool used to perform DNS lookups in Linux. It is used to display DNS details, such as the IP address of a particular computer, the MX records for a domain or the NS servers of a domain. nslookup can operate in two modes: interactive and non-interactive. e.g.:- nslookup www.google.com
pathping: Pathping is unique to Window's, and is basically a combination of the Ping and Tracert commands. Pathping traces the route to the destination address then launches a 25 second test of each router along the way, gathering statistics on the rate of data loss along each hop.
ping: (Packet INternet Groper) command is the best way to test connectivity between two nodes. Ping use ICMP (Internet Control Message Protocol) to communicate to other devices. 1. #ping hostname(ping localhost) 2. #ping ip address (ping 4.2.2.2) 3. #ping fully qualified domain name(ping www.facebook.com
Route: route command is used to show/manipulate the IP routing table. It is primarily used to setup static routes to specific host or networks via an interface.

Some important Linux networking commands

1. ip

The ip command is one of the basic commands every administrator will need in daily work, from setting up new systems and assigning IPs to troubleshooting existing systems. The ip command can show address information, manipulate routing, plus display network various devices, interfaces, and tunnels.

ip <OPTIONS> <OBJECT> <COMMAND>

Here are some common use cases for the ip command.

a. To show the IP addresses assigned to an interface on your server:
   a. [root@server ~]# *ip address show*

b. To assign an IP to an interface, for example, **enps03**:
   a. [root@server ~]# ip address add 192.168.1.254/24 dev enps03

c. To delete an IP on an interface:
   a. [root@server ~]# ip address del 192.168.1.254/24 dev enps03

d. Alter the status of the interface by bringing the interface **eth0** online:
   [root@server ~]# ip link set eth0 up

e. Alter the status of the interface by bringing the interface **eth0** offline:
   [root@server ~]# ip link set eth0 down

f. Alter the status of the interface by enabling promiscuous mode for **eth0**:
   [root@server ~]# ip link set eth0 promisc on

g. Add a default route (for all addresses) via the local gateway 192.168.1.254 that can be reached on device **eth0**:
   [root@server ~]# ip route add default via 192.168.1.254 dev eth0

h. Add a route to 192.168.1.0/24 via the gateway at 192.168.1.254:
   [root@server ~]# ip route add 192.168.1.0/24 via 192.168.1.254

i. Add a route to 192.168.1.0/24 that can be reached on device **eth0**:
   [root@server ~]# ip route add 192.168.1.0/24 dev eth0

j. Delete the route for 192.168.1.0/24 via the gateway at 192.168.1.254:
   [root@server ~]# ip route delete 192.168.1.0/24 via 192.168.1.254

k. Display the route taken for IP 10.10.1.4:
   [root@server ~]# ip route get 10.10.1.4

2. **ifconfig**

The ifconfig command was/is a staple in many sysadmin's tool belt for configuring and troubleshooting networks. It has since been replaced by the ip command discussed above.

3. **mtr**

MTR (Matt's traceroute) is a program with a command-line interface that serves as a network diagnostic and troubleshooting tool. This command combines the functionality of the ping and traceroute commands. Just like a traceroute, the mtr command will show the route from a computer to a specified host. mtr provides a lot of statistics about each hop, such as response time and percentage. With the mtr command, you will get more information about the route and be able to see problematic devices along the way. If you see a sudden increase in response time or packet loss, then obviously, there is a bad link somewhere.
The syntax of the command is as follows:

mtr <options> hostname/IP

Let's look at some common use cases.

a. The basic mtr command shows you the statistics, including each hop (hostnames) with time and loss%:

[root@server ~]# *mtr google.com*

b. Show numeric IP addresses (if you use -g, you will get IP addresses (numbers) instead of hostnames):

[root@server ~]# mtr -g google.com

c. Show the numeric IP addresses and hostnames, too:

[root@server ~]# mtr -b google.com

d. Set the number of pings that you want to send:

[root@server ~]# mtr -c 10 google.com

4. tcpdump

The tcpdump command is designed for capturing and displaying packets.

You can install tcpdump with the command below:
[root@server ~]# *dnf install -y tcpdump*

Before starting any capture, you need to know which interfaces tcpdump can use. You will need to use sudo or have root access in this case.
[root@server ~]# *tcpdump -D*

If you want to capture traffic on **eth0**, you can initiate that with tcpdump -i eth0 sample output:
[root@server ~]# *tcpdump -i eth0*

[root@server ~]# *tcpdump -i eth0 -c 10*

Capture traffic to and from one host

You can filter out traffic coming from a specific host. For example, to find traffic coming from and going to 8.8.8.8, use the command:

[root@server ~]# *tcpdump -i eth0 -c 10 host 8.8.8.8*

For traffic coming from 8.8.8.8, use:

[root@server ~]# *tcpdump -i eth0 src host 8.8.8.8*

For outbound traffic going to 8.8.8.8, use:

[root@server ~]# *tcpdump -i eth0 dst host 8.8.8.8*

Capture traffic to and from a network

You can also capture traffic to and from a specific network using the command below:

[root@server ~]# *tcpdump -i eth0 net 10.1.0.0 mask 255.255.255.0*

or:

[root@server ~]# *tcpdump -i eth0 net 10.1.0.0/24*

Capture traffic to and from port numbers

Capture only DNS port 53 traffic:

[root@server ~]# *tcpdump -i eth0 port 53*

For a specific host,

[root@server ~]# *tcpdump -i eth0 host 8.8.8.8 and port 53*

To capture only HTTPS traffic,

[root@server ~]# *tcpdump -i eth0 -c 10 host www.google.com and port 443*

To capture all port except port 80 and 25,

[root@server ~]# *tcpdump -i eth0 port not 53 and not 25*

5. ping

Ping is a tool that verifies IP-level connectivity to another TCP/IP computer by sending Internet Control Message Protocol (ICMP) Echo Request messages. The receipt of corresponding Echo Reply messages is displayed, along with round-trip times. Ping is the primary TCP/IP command used to troubleshoot connectivity, reachability, and name resolution.

```
[root@server ~]# ping google.com
PING google.com (216.58.206.174) 56(84) bytes of data.
64 bytes from sof02s27-in-f14.1e100.net (216.58.206.174): icmp_seq=1 ttl=56 time=10.7 ms
64 bytes from sof02s27-in-f14.1e100.net (216.58.206.174): icmp_seq=2 ttl=56 time=10.2 ms
64 bytes from sof02s27-in-f14.1e100.net (216.58.206.174): icmp_seq=3 ttl=56 time=10.4 ms
^C
```

You need to stop the ping command by pressing **CTRL+C**. Otherwise, it will ping until you stop it.

If you want to ping a host ten times, use the following command:
[root@server ~]*# ping -c 10 google.com*

While pinging a host, you'll find different output from the ping results, including the following three examples.

Destination Host Unreachable

The possible best reason is there is no route from the local host system and to the destination desired destination host, or a remote router reports that it has no route to the destination host.

Request timed out

This result means that no Echo Reply messages were received within the default time of one second or the time that you set while you are pinging that host. This can be due to many different causes; the most common include network congestion, failure of the ARP request, packet filtering/firewall, etc.

Unknown host/Ping Request Could Not Find Host

Maybe you misspelled the hostname or the host does not exist at all in the network.

You must have 0% packet loss for every ping result with a good latency or lower response time. Depending on which transmission medium (UTP, fibre optics cable, Wi-Fi) you're using, your latency will differ.

Configuring an Ethernet connection by using nmcli

If you connect a host to the network over Ethernet, you can manage the connection's settings on the command line by using the **nmcli** utility.

Procedure

1. List the NetworkManager connection profiles:

   # **nmcli connection show**

NAME	UUID	TYPE	DEVICE
Wired connection 1	a5eb6490-cc20-3668-81f8-0314a27f3f75	ethernet	enp1s0

2. # **nmcli connection add con-name <*connection-name*> ifname <*device-name*> type ethernet**
   Skip this step to modify an existing profile.

3. Optional: Rename the connection profile:

   # **nmcli connection modify "Wired connection 1"**
   Here, "Wired connection 1" is the name of the connection
4. Display the current settings of the connection profile:
   # **nmcli connection show**

   connection.interface-name: enp1s0
   connection.autoconnect: yes
   ipv4.method: auto
   ipv6.method: auto
   ...
5. Configure the IPv4 settings:
   - To use DHCP, enter:

     # **nmcli connection modify "Wired connection 1" ipv4.method auto**
     Skip this step if ipv4.method is already set to auto (default).

   - To set a static IPv4 address, network mask, default gateway, DNS servers, and search domain, enter:

     # **nmcli connection modify "Wired connection 1" ipv4.method manual ipv4.addresses 192.0.2.1/24 ipv4.gateway 192.0.2.254 ipv4.dns 192.0.2.200 ipv4.dns-search example.com**

6. Configure the IPv6 settings:
   - To use stateless address autoconfiguration (SLAAC), enter:

     # **nmcli connection modify "Wired connection 1" ipv6.method auto**
     Skip this step if ipv6.method is already set to auto (default).

- To set a static IPv6 address, network mask, default gateway, DNS servers, and search domain, enter:

  # **nmcli connection modify "Wired connection 1" ipv6.method manual ipv6.addresses 2001:db8:1::fffe/64 ipv6.gateway 2001:db8:1::fffe ipv6.dns 2001:db8:1::ffbb ipv6.dns-search example.com**

7. Activate the profile:

   # **nmcli connection up Internal-LAN**

   Verification

1. Display the IP settings of the NIC:

   # **ip address show enp1s0**
   enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
   link/ether 52:54:00:17:b8:b6 brd ff:ff:ff:ff:ff:ff
   inet 192.0.2.1/24 brd 192.0.2.255 scope global noprefixroute enp1s0
   valid_lft forever preferred_lft forever
   inet6 2001:db8:1::fffe/64 scope global noprefixroute
   valid_lft forever preferred_lft forever

2. Display the IPv4 default gateway:

   # **ip route show default**

   default via 192.0.2.254 dev enp1s0 proto static metric 102

3. Display the IPv6 default gateway:

   # **ip -6 route show default**
   default via 2001:db8:1::ffee dev enp1s0 proto static metric 102 pref medium

4. Display the DNS settings:

   # **cat /etc/resolv.conf**
   search example.com
   nameserver 192.0.2.200
   nameserver 2001:db8:1::ffbb
   If multiple connection profiles are active at the same time, the order of nameserver entries depend on the DNS priority values in these profile and the connection types.

5. Use the ping utility to verify that this host can send packets to other hosts:

   # **ping <*host-name-or-IP-address*>**

Troubleshooting

- Verify that the network cable is plugged-in to the host and a switch.
- Check whether the link failure exists only on this host or also on other hosts connected to the same switch.
- Verify that the network cable and the network interface are working as expected. Perform hardware diagnosis steps and replace defect cables and network interface cards.
- If the configuration on the disk does not match the configuration on the device, starting or restarting NetworkManager creates an in-memory connection that reflects the configuration of the device.

Student Observation:

1. **Which command is used to find the reachability of a host machine from your device?**
2. **Which command will be give the details of hops taken by a packet to reach its destination?**
3. **Which commands displays the ip configuration of your machine.**
4. **Which command displays the TCP port status in your machine?**
5. **Write the modify the ip configuration in a Linux machine.**

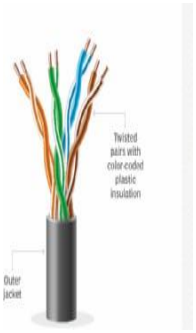
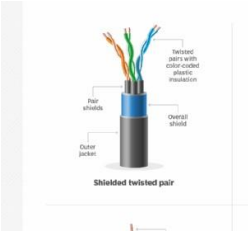
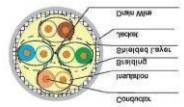
Practical-2

Aim: Study of different types of Network cables.

a) Understand different types of network cable.

Different type of cables used in networking are:

1. Unshielded Twisted Pair (UTP) Cable
2. Shielded Twisted Pair (STP) Cable
3. Coaxial Cable
4. Fibre Optic Cable

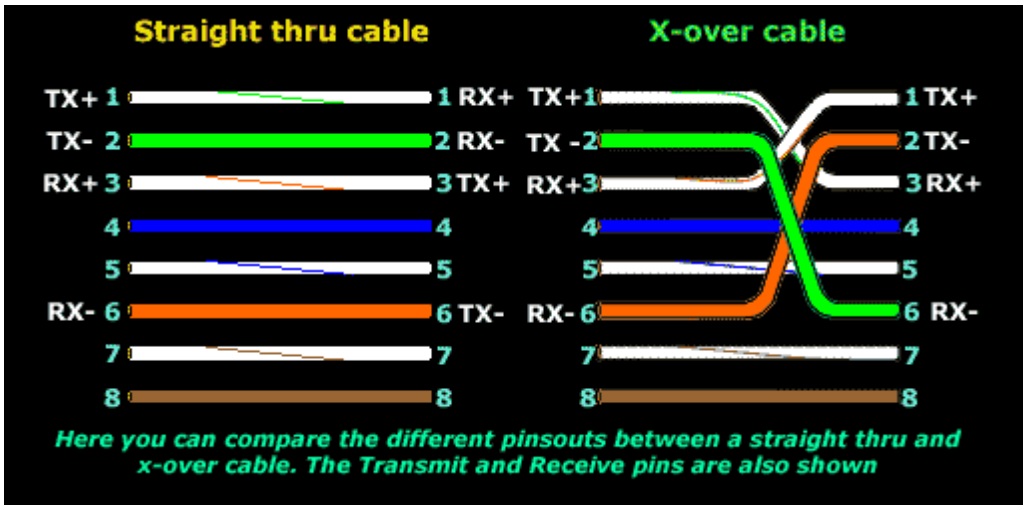
Cable type	Category	Maximum Data Transmission	Advantages/ Disadvantages	Application/Use	Image
UTP	Category 3	10 bps	**Advantages** • Cheaper in cost • Easy to install as they have a smaller overall diameter. **Disadvantages** • More prone to (EMI) Electromagnetic interference and noise	10Base-T Ethernet	
	Category 5	Up to 100 Mbps		Fast Ethernet, Gigabit Ethernet	
	Category 5e	1Gbps		Fast Ethernet, Gigabit Ethernet	
STP	Category6,6a	10Gbps	**Advantages** • Shielded. • Faster than UTP. • Less susceptible to noise and interference **Disadvantages** • Expensive • Greater installation effort	Gigabit Ethernet, 10G Ethernet (55m) Widely used in data centres	
SSTP	Category 7	10Gbps		Gigabit Ethernet, 10G Ethernet (100m)	 

Coaxial cable	RG-6 RG-59 RG-11	10-100Mbps	• High bandwidth • Immune to interference • Low loss bandwidth • Versatile • **Disadvantages** • Limited distance • Cost • Size is bulky	Speed of signal is 500m Television network High speed internet connections	
fibre optics cable	Single mode Multi mode	100Gbps	**Advantages** • High speed • High bandwidth • High security • Long distance **Disadvantages** • Expensive • Requires skilled installers	• Maximum distance of fibre optics cable is around 100meters	

b) Make Your Own Ethernet Cross-Over Cable/ Straight cable

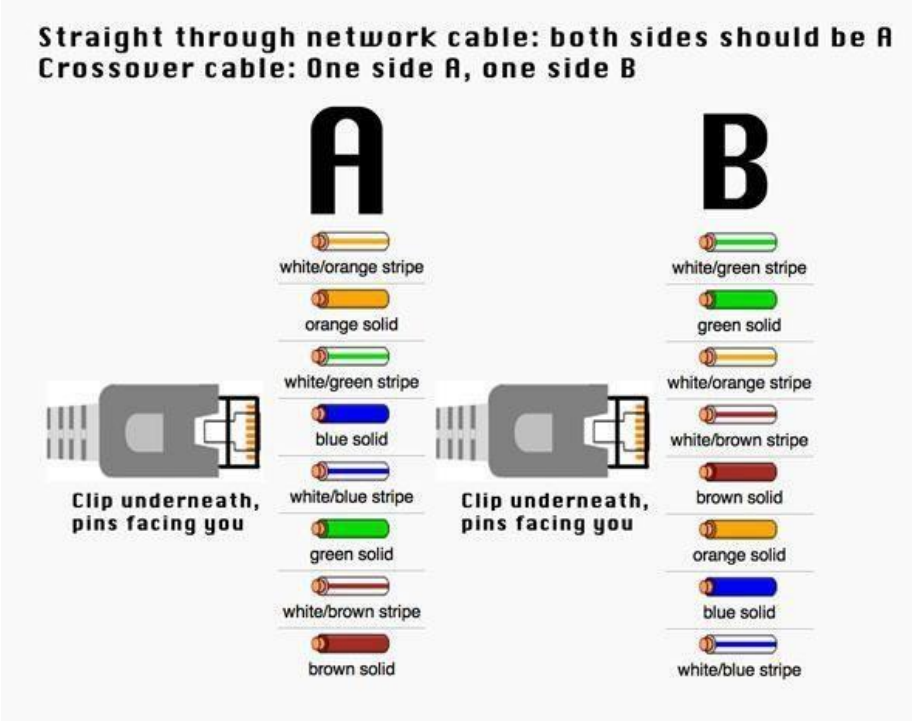
Tools and parts needed:

- Ethernet cabling. CAT5e is certified for gigabit support, but CAT5 cabling works as well, just over shorter distances.
- A crimping tool. This is an all-in-one networking tool shaped to push down the pins in the plug and strip and cut the shielding off the cables.
- Two RJ45 plugs.
- Optional two plug shields.

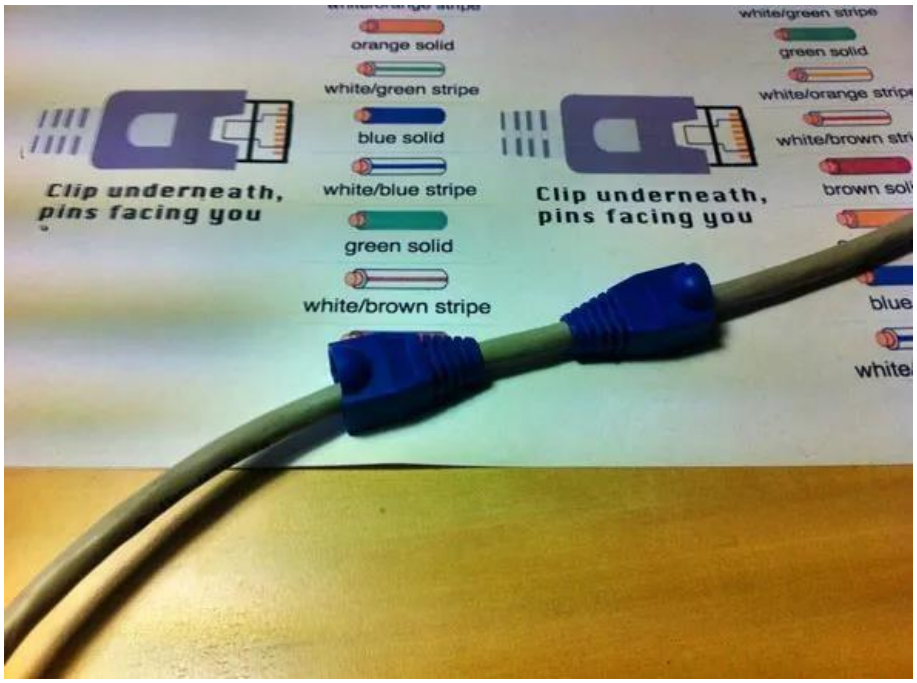


Difference between crossover cable and straight cable

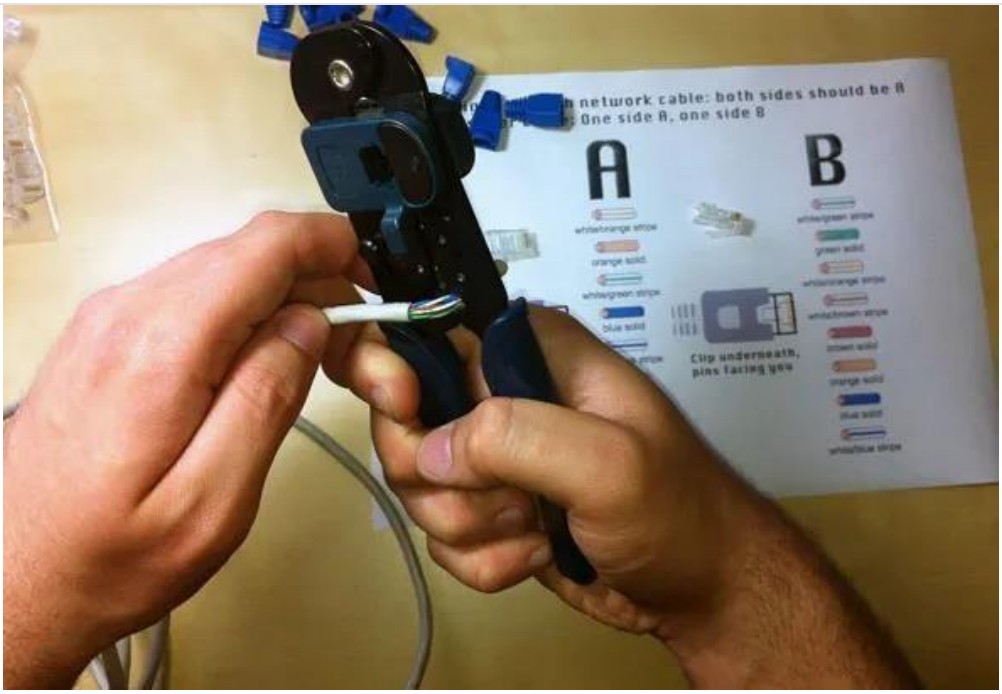
Take a print out the diagram below or have it handy as a reference



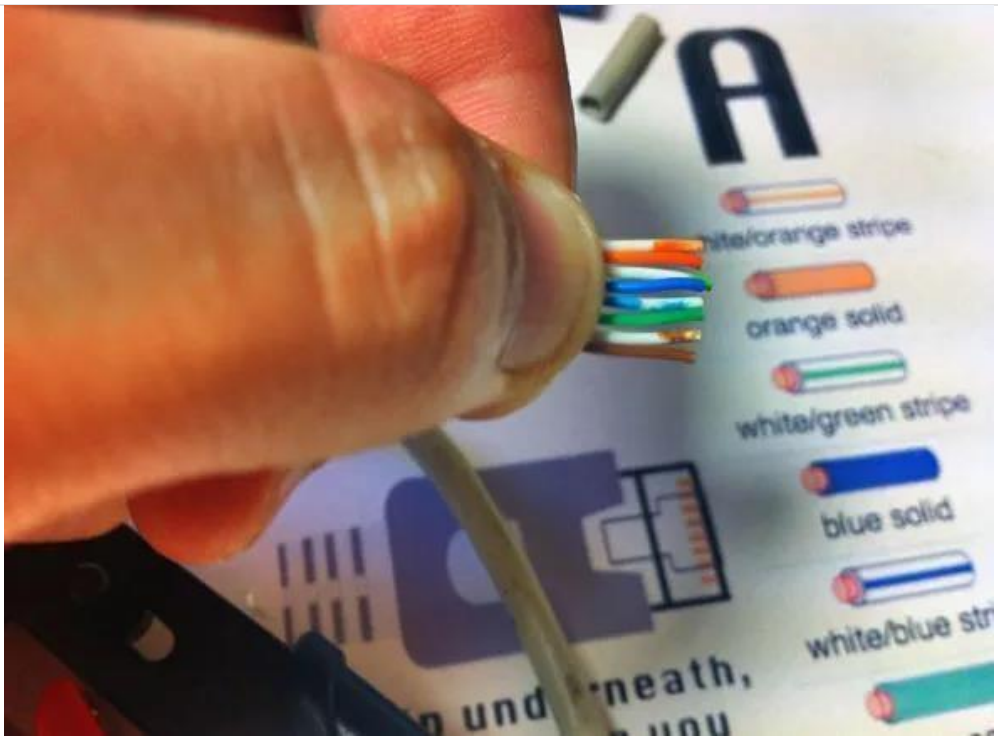
Step 1: To start construction of the device, begin by threading shields onto the cable.



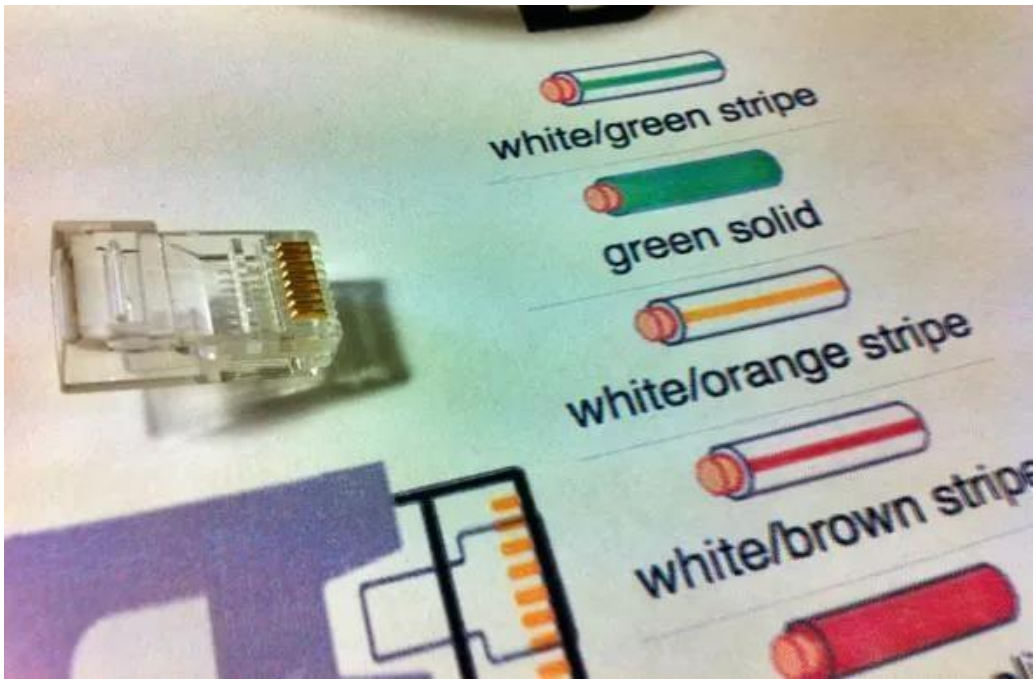
Step 2: Next, strip approximately 1.5 cm of cable shielding from both ends. The crimping tool has a round area to complete this task.



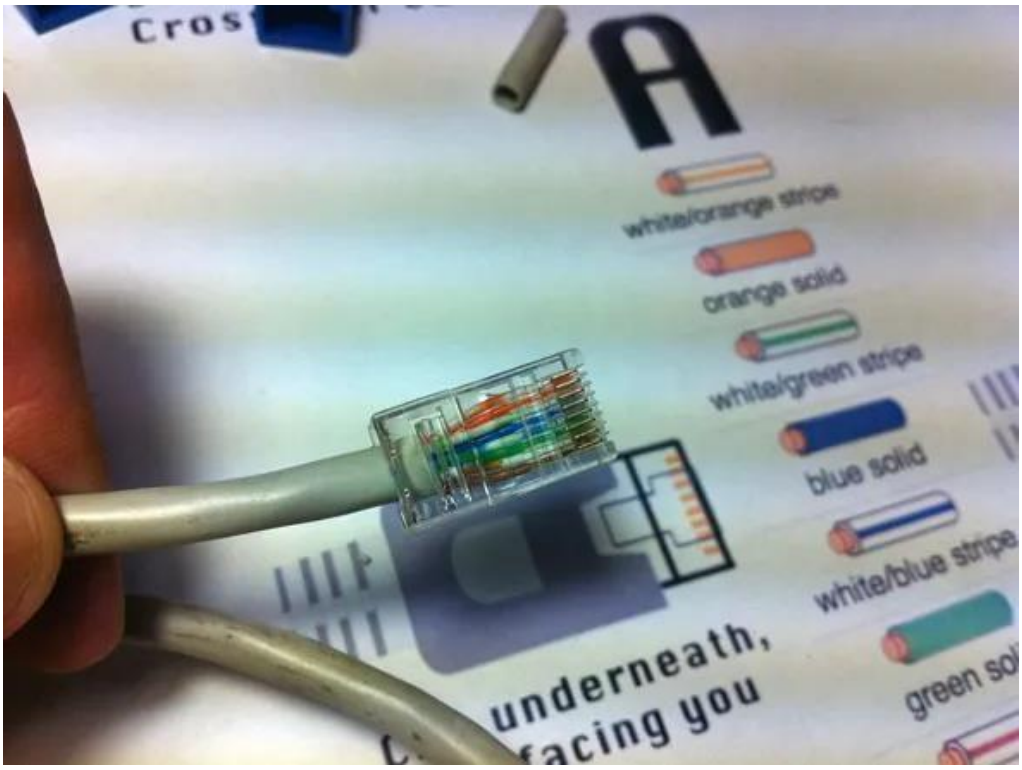
Step 3: After, you will need to untangle the wires; there should be four "twisted pairs." Referencing back to the sheet, arrange them from top to bottom. One end should be in arrangement A and the other in B.



Step 4: Once the order is correct, bunch them together in a line, and if there are any that stick out farther than others, snip them back to create an even level. The difficult aspect is placing these into the RJ45 plug without messing up the order. To do so, hold the plug with the clip side facing away from you and have the gold pins facing toward you, as shown.



Step 5: Next, push the cable right in. The notch at the end of the plug needs to be just over the cable shielding, and if it isn't, that means that you stripped off too much shielding. Simply snip the cables back a little more.



Step 6: After the wires are securely sitting inside the plug, insert it into the crimping tool and push down.
It should be shaped correctly, but pushing too hard can crack the fragile plastic plug.

Step 7: Lastly, repeat for the other end using diagram B (to make a crossover cables)/ using diagram A (to make a straight through cable)

To test it, plug it in and attempt to connect two devices directly.

Student observation:-

1. **What is the difference between cross cable and straight cable?**
2. **Which type of cable is used to connect two PC?(straight/Cross cable)**
3. **Which type cable is used to connect a router/switch to your PC? (straight/Cross cable)**
4. **Find out the category of twisted pair cable used in your la to connect the PC to the network socket.**
5. **Write down your understanding, challenges faced and output received while making a twisted pair cross/straight cable.**

Practical -3

AIM: To study the Packet tracer tool Installation and User Interface Overview

c) To understand environment of CISCO PACKET TRACER to design simple network.

INTRODUCTION:

A simulator, as the name suggests, simulates network devices and its environment. Packet Tracer is an exciting network design, simulation and modelling tool.

1. It allows you to model complex systems without the need for dedicated equipment.
2. It helps you to practice your network configuration and troubleshooting skills via computer or an Android or iOS based mobile device.
3. It is available for both the Linux and Windows desktop environments.
4. Protocols in Packet Tracer are coded to work and behave in the same way as they would on real hardware.

INSTALLING PACKET TRACER:

To download Packet Tracer, go to **https://www.netacad.com** and log in with your Cisco Networking Academy credentials; then, click on the Packet Tracer graphic and download the package appropriate for your operating system. (Can be used to download in your laptop).

Windows

Installation in Windows is pretty simple and straightforward; the setup comes in a single file named Packettracer_Setup6.0.1.exe. Open this file to begin the setup wizard, accept the license agreement, choose a location, and start the installation.

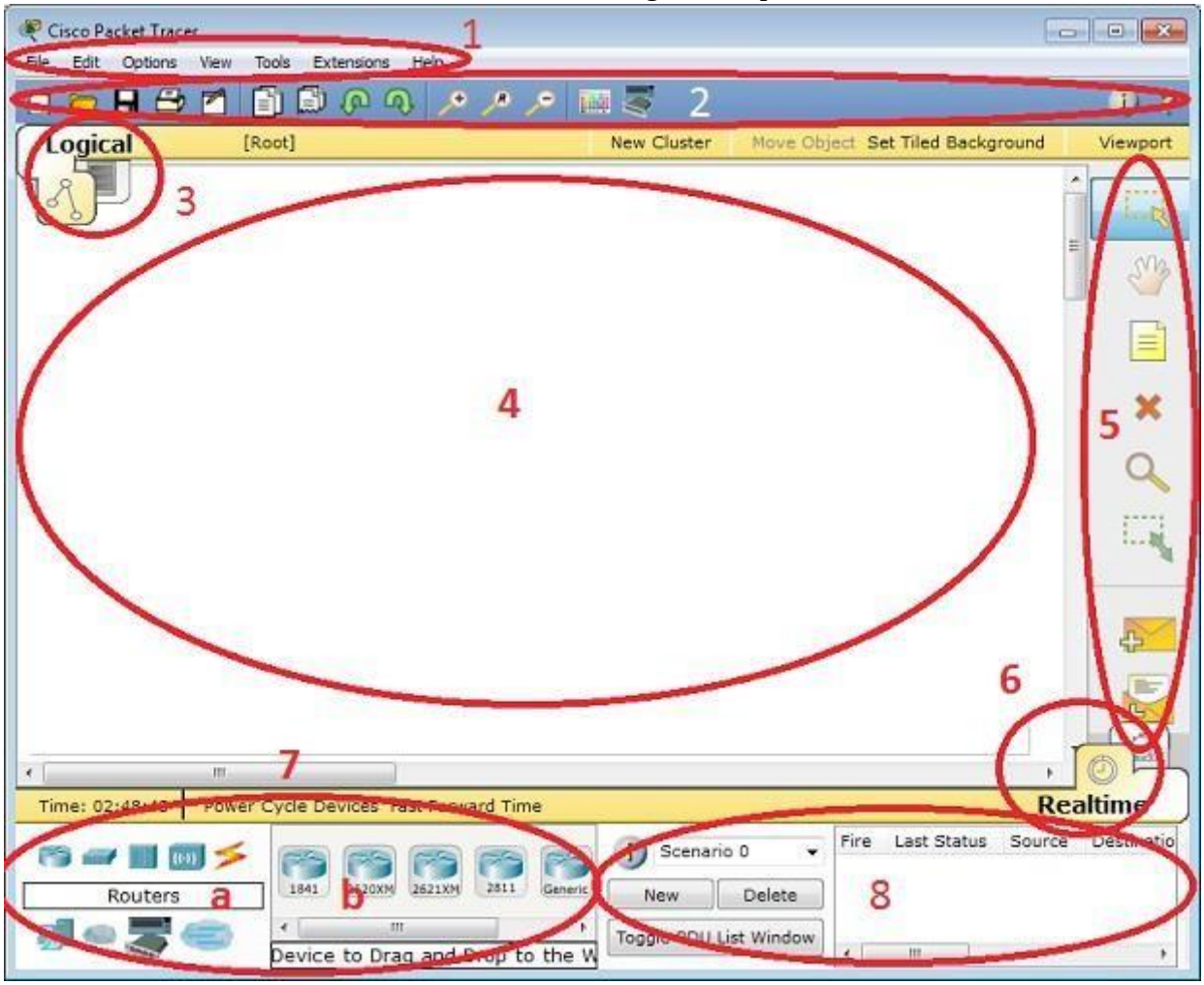
Linux

Linux users with an Ubuntu/Debian distribution should download the file for Ubuntu, and those using Fedora/Redhat/CentOS must download the file for Fedora. Grant executable permission to this file by using chmod, and execute it to begin the installation.

```
chmod +x PacketTracer601_i386_installer-rpm.bin
        ./PacketTracer601_i386_installer-rpm.bin
```

USER INTERFACE OVERVIEW:

The layout of Packet Tracer is divided into several components. The components of the Packet Tracer interface are as follows: match the numbering with explanations.



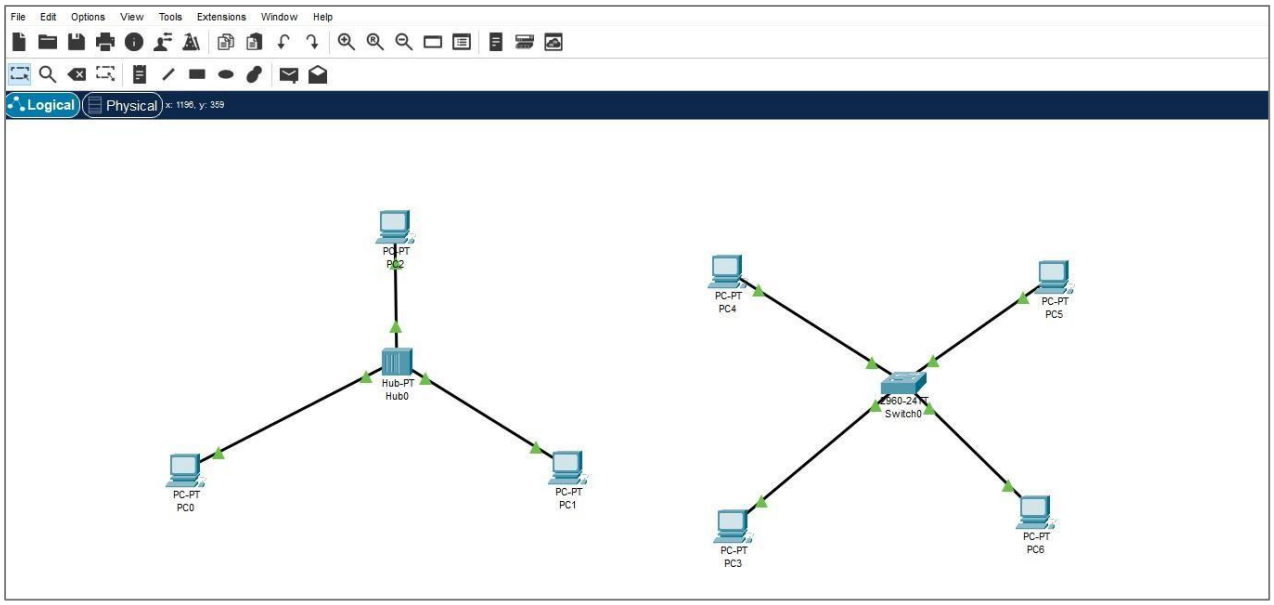
1. Menu bar – This is a common menu found in all software applications; it is used to open, save, print, change preferences, and so on.
2. Main toolbar – This bar provides shortcut icons to menu options that are commonly accessed, such as open, save, zoom, undo, and redo, and on the right-hand side is an icon for entering network information for the current network.
3. Logical/Physical workspace tabs – These tabs allow you to toggle between the Logical and Physical work areas.
4. Workspace – This is the area where topologies are created and simulations are displayed.
5. Common tools bar – This toolbar provides controls for manipulating topologies, such as select, move layout, place note, delete, inspect, resize shape, and add simple/complex PDU.
6. Real-time/Simulation tabs – These tabs are used to toggle between the real and simulation modes. Buttons are also provided to control the time, and to capture the packets.
7. Network component box – This component contains all of the network and end devices available with Packet Tracer, and is further divided into two areas: Area 7a: Device-type selection box – This area contains device categories Area 7b: Device-specific selection box – When a device category is selected, this selection box displays the different device models within that category

8. User-created packet box – Users can create highly-customized packets to test their topology from this area, and the results are displayed as a list.

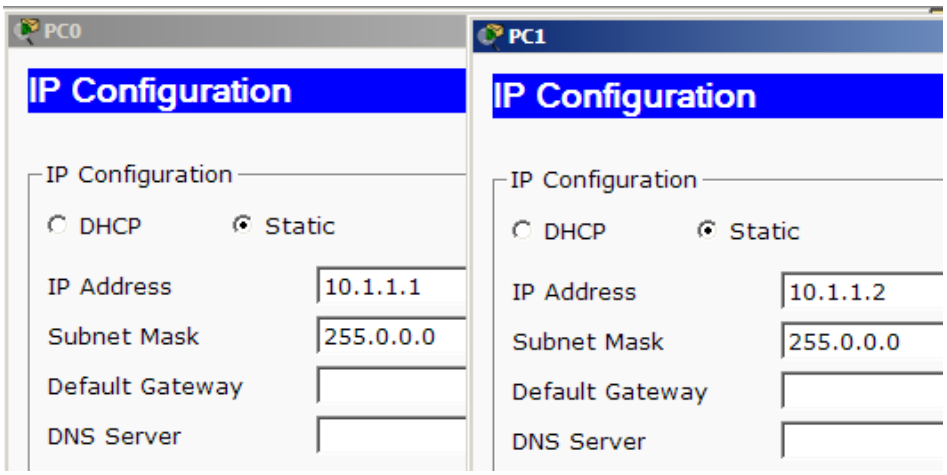
d) Analyse the behaviour of network devices using CISCO PACKET TRACER simulator.

1. From the network component box, click and drag-and-drop the below components:
    a. 4 Generic PCs and One HUB
    b. 4 Generic PCs and One switch

2. Click on Connections:
    a. Click on Copper Straight-Through cable,
    b. Select one of the PC and connect it to HUB using the cable. The link LED should glow in green, indicating that the link is up. Similarly connect remaining 3 PCs to the HUB.
    c. Similarly connect 4 PCs to the switch using copper straight-through cable.



3. Click on the PCs connected to hub, go to the Desktop tab, click on IP Configuration, and enter an IP address and subnet mask. Here, the default gateway and DNS server information is not needed as there are only two end devices in the network.



Click on the PDU (message icon) from the common tool bar,

a. Drag and drop it on one of PC (source machine) and then drop it on another PC (destination machine) connected to the HUB.

4. Observe the flow of PDU from source PC to destination PC by selecting the Realtime mode of simulation.

5. Repeat step #3 to step #5 for the PCs connected to the switch.

6. Observe how HUB and switch are forwarding the PDU and write your observation and conclusion about the behaviors of Switch and HUB.

Student observation:

a. From your observation write down the behavior of Switch and HUB in terms of forwarding the packets received by them.

b. Find out the network topology implemented in your college and draw and label that topology in your observation book.

Practical -4

AIM: Setup and configure a LAN (Local area network) using a Switch and Ethernet cables in your lab.

What is a LAN?

A Local Area Network (LAN) refers to a network that connects devices within a limited area, such as an office building, school, or home. It enables users to share resources, including data, printers, and internet access. LAN connects devices to promote collaboration and transfer information between users, such as computers, printers, servers, and switches. A local area network (LAN) switch serves as the primary connecting device, managing and directing communications within the local network. Each connected device on a LAN switch can communicate directly with each other, allowing for fast and secure data transfer.

How to set up a LAN

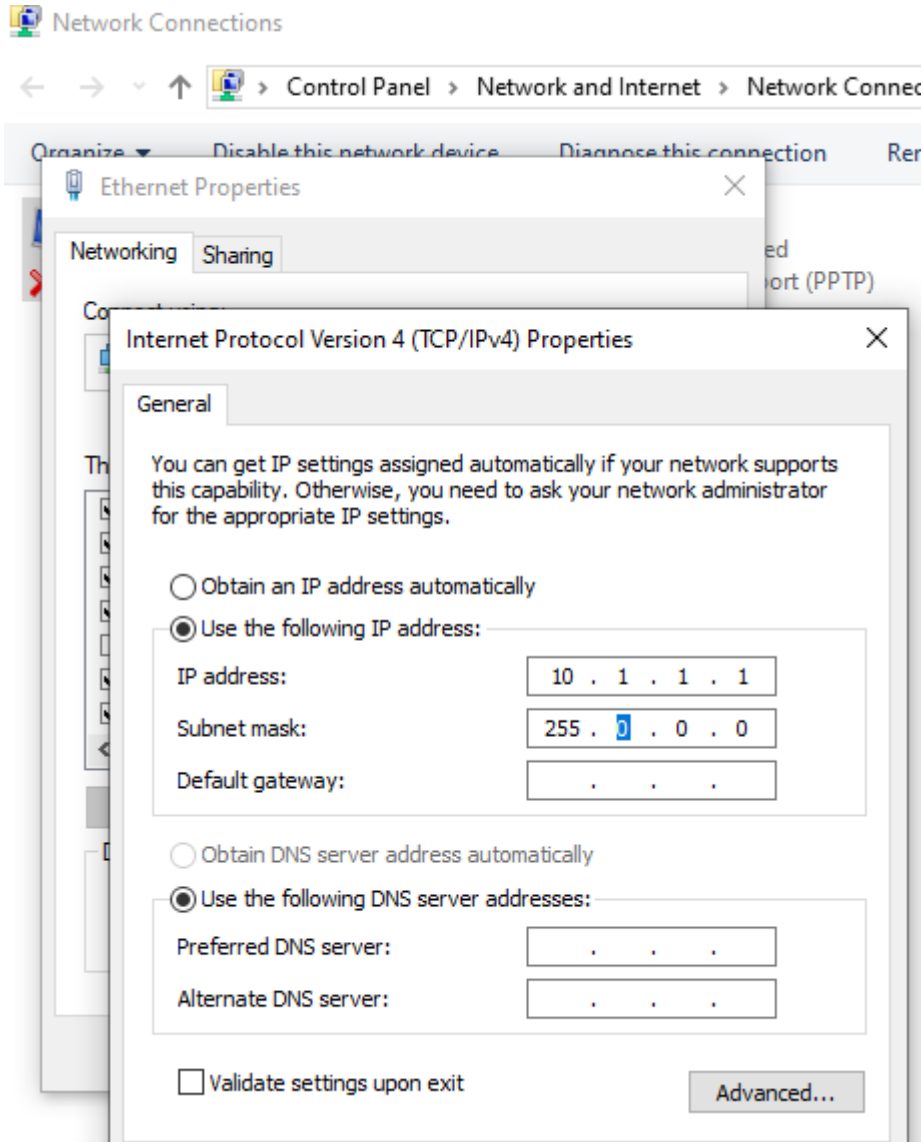
Step 1. Plan and Design an appropriate network topology taking into account network requirements and equipment location.

Step 2. You can take 4 Computers, a Switch with 8, 16, or 24 ports which is sufficient for networks of these sizes, and 4 Ethernet cables.

Step3: Connect your computers to network switch via an Ethernet cable, which is as simple as plugging one end of the Ethernet cable into your computer and the other end into your network switch.

Step4: Assign IP address to your PCs

1. Log on to the client computer as Administrator or as Owner.
2. Click Network and Internet Connections.
3. Right Click Local Area Connection/Ethernet->Go to Properties->Select Internet Protocol (TCP/IPv4)->Click on Properties->Select use the following ip address option and assign ipaddress.



Similarly assign IP address to all the PCS connected to switch.
PC1-IP address: 10.1.1.1, subnet mask 255.0.0.0
PC2-IP address-10.1.1.2, subnet mask 255.0.0.0
PC3-IP address 10.1.1.3, subnet mask 255.0.0.0.
PC4-IP address 10.1.1.4, subnet mask 255.0.0.0.

Step 5:- Configure a network switch:

1. Connect your computer to the switch: To access the switch's web interface, you will need to connect your computer to the switch using an Ethernet cable.
2. Log in to the web interface: Open a web browser and enter the IP address of the switch in the address bar. This should bring up the login page for the switch's web interface. Enter the username and password to log in.
3. Configure basic settings: Once you're logged in, you will be able to configure basic settings for the switch,
4. Assign IP address as: 10.1.1.5, subnet mask 255.0.0.0.

Step 6:- Check the connectivity between switch and other machine by using ping command in the command prompt of the device.

Step 7: Select a folder, ->go to properties-> click Sharing tab->share it with everyone on the same LAN.

Step 8. Try to access the shared folder from others Computers of the network.

Student observation:

Draw a neat diagram of the LAN in the configuration observation book. that you have implemented in your lab. Write the ip configuration of each and every device. Write the outcome and challenges faced while configuring the LAN.

Practical-5

AIM Experiments on Packet capture tool: Wireshark

Packet Sniffer

- Sniffs messages being sent/received from/by your computer
- Store and display the contents of the various protocol fields in the messages
- Passive program
  - never sends packets itself
  - no packets addressed to it
  - receives a copy of all packets (sent/received)

Packet Sniffer Structure Diagnostic Tools

- Tcpdump
  - E.g. tcpdump -enx host 10.129.41.2 -w exe3.out
- Wireshark
  - wireshark -r exe3.out

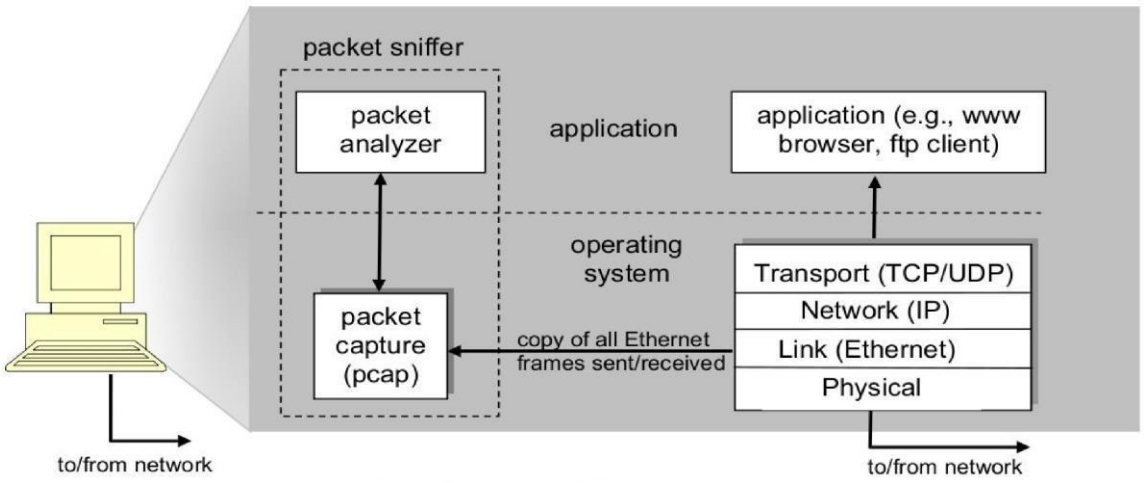


Figure 1: Packet sniffer structure

DESCRIPTION:

WIRESHARK

Wireshark, a network analysis tool formerly known as Ethereal, captures packets in real time and display them in human-readable format. Wireshark includes filters, color coding, and other features that let you dig deep into network traffic and inspect individual packets. You can use Wireshark to inspect a suspicious program's network traffic, analyze the traffic flow on your network, or troubleshoot network problems.

What we can do with Wireshark:

- Capture network traffic
- Decode packet protocols using dissectors
- Define filters – capture and display
- Watch smart statistics
- Analyze problems
- Interactively browse that traffic

Wireshark used for:

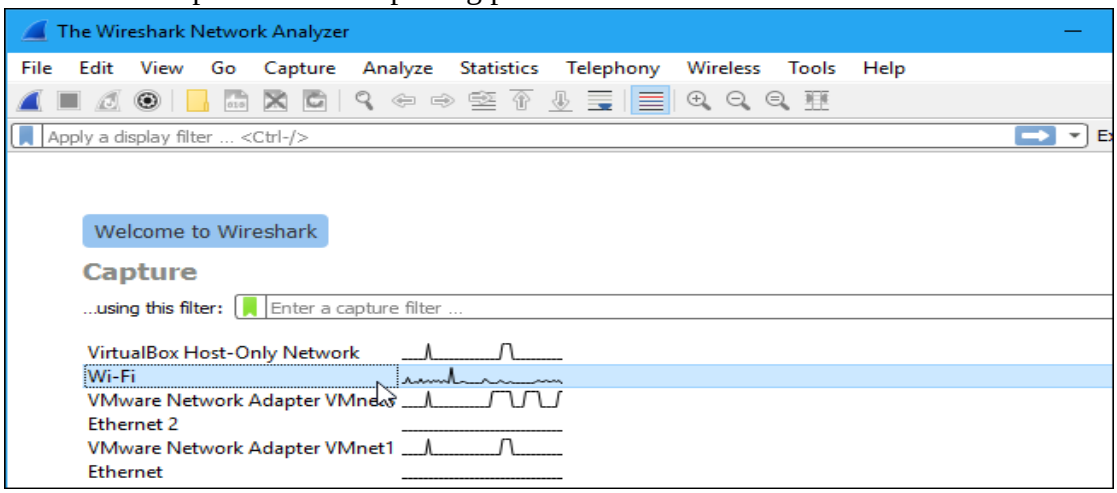
- Network administrators: troubleshoot network problems
- Network security engineers: examine security problems
- Developers: debug protocol implementations
- People: learn **network protocol internals**

Getting Wireshark

Wireshark can be downloaded for Windows or macOS from its official website. For Linux or another UNIX-like system, Wireshark will be found in its package repositories. For Ubuntu, Wireshark will be found in the Ubuntu Software Center.

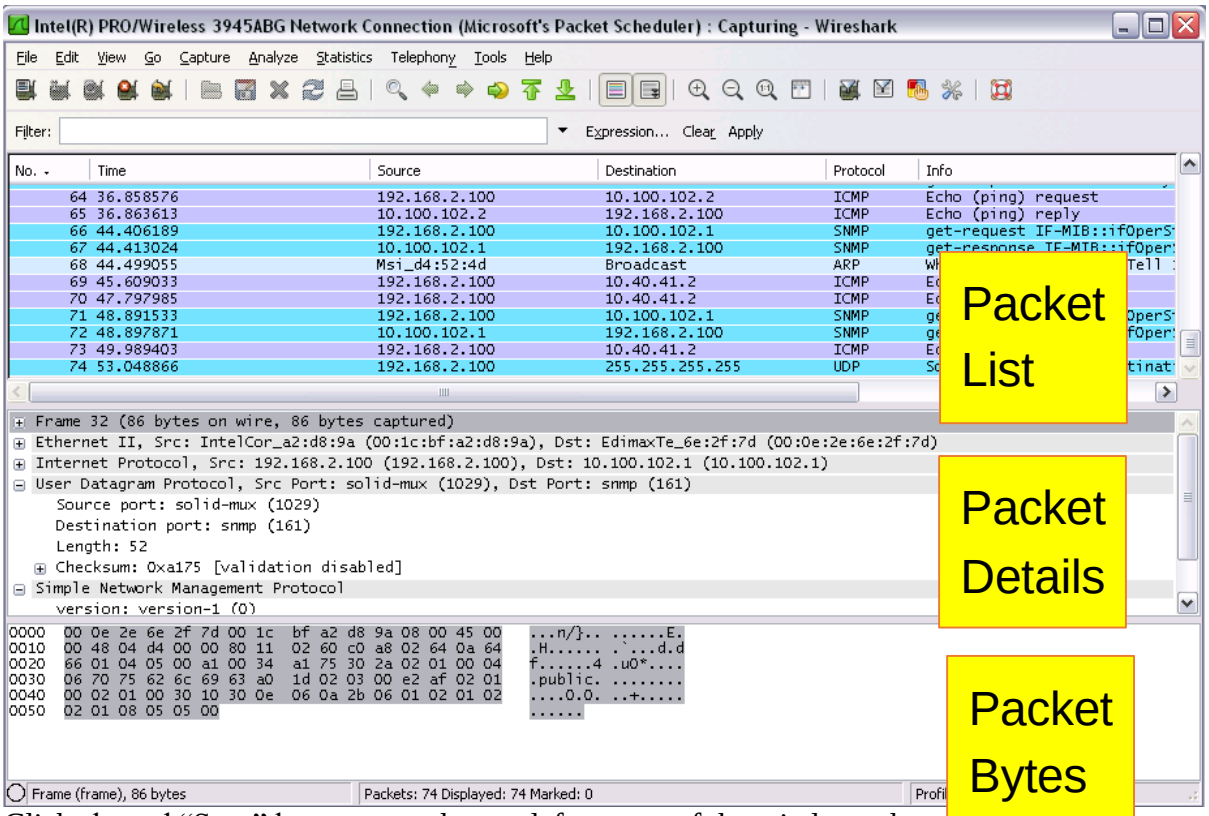
Capturing Packets

After downloading and installing Wireshark, launch it and double-click the name of a network interface under Capture to start capturing packets on that interface



As soon as you click the interface's name, you'll see the packets start to appear in real time. Wireshark captures each packet sent to or from your system.

If you have promiscuous mode enabled—it's enabled by default—you'll also see all the other packets on the network instead of only packets addressed to your network adapter. To check if promiscuous mode is enabled, click Capture > Options and verify the "Enable promiscuous mode on all interfaces" checkbox is activated at the bottom of this window.



Click the red "Stop" button near the top left corner of the window when you want to stop capturing traffic.

The "Packet List" Pane

The packet list pane displays all the packets in the current capture file. The "Packet List" pane Each line in the packet list corresponds to one packet in the capture file. If you select a line in this pane, more details will be displayed in the "Packet Details" and "Packet Bytes" panes.

The "Packet Details" Pane

The packet details pane shows the current packet (selected in the "Packet List" pane) in a more detailed form. This pane shows the protocols and protocol fields of the packet selected in the "Packet List" pane. The protocols and fields of the packet shown in a tree which can be expanded and collapsed.

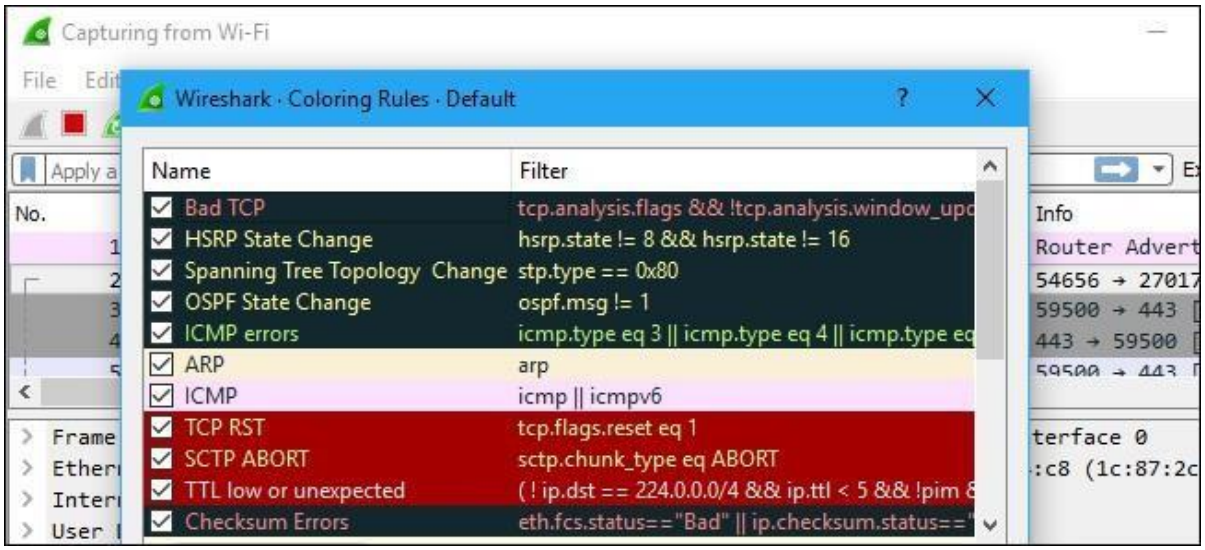
The "Packet Bytes" Pane

The packet bytes pane shows the data of the current packet (selected in the "Packet List" pane) in a hexdump style.

Color Coding

You'll probably see packets highlighted in a variety of different colors. Wireshark uses colors to help you identify the types of traffic at a glance. By default, light purple is TCP traffic, light blue is UDP traffic, and black identifies packets with errors—for example, they could have been delivered out of order.

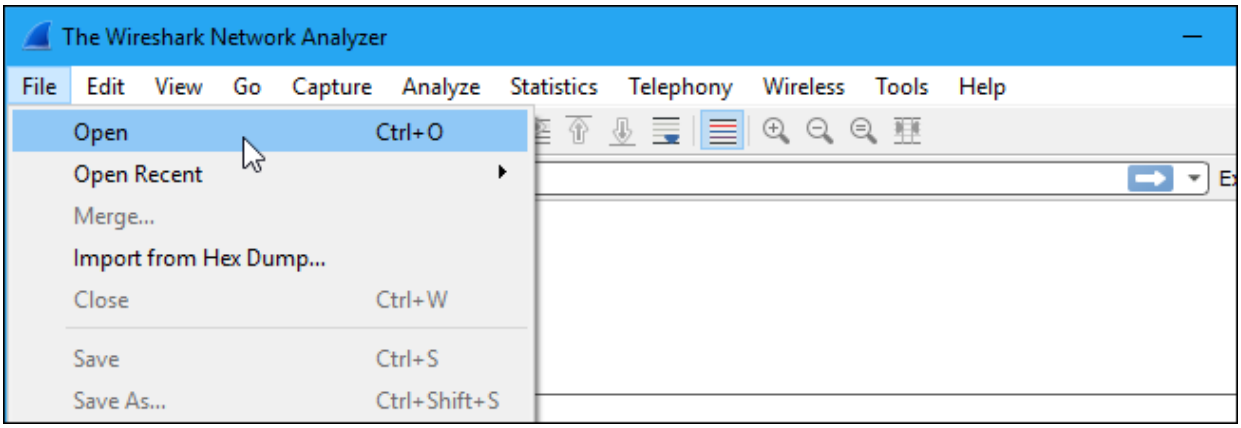
To view exactly what the color codes mean, click View > Coloring Rules. You can also customize and modify the coloring rules from here, if you like.



Sample Captures

If there's nothing interesting on your own network to inspect, Wireshark's wiki has you covered. The wiki contains a page of sample capture files that you can load and inspect. Click File > Open in Wireshark and browse for your downloaded file to open one.

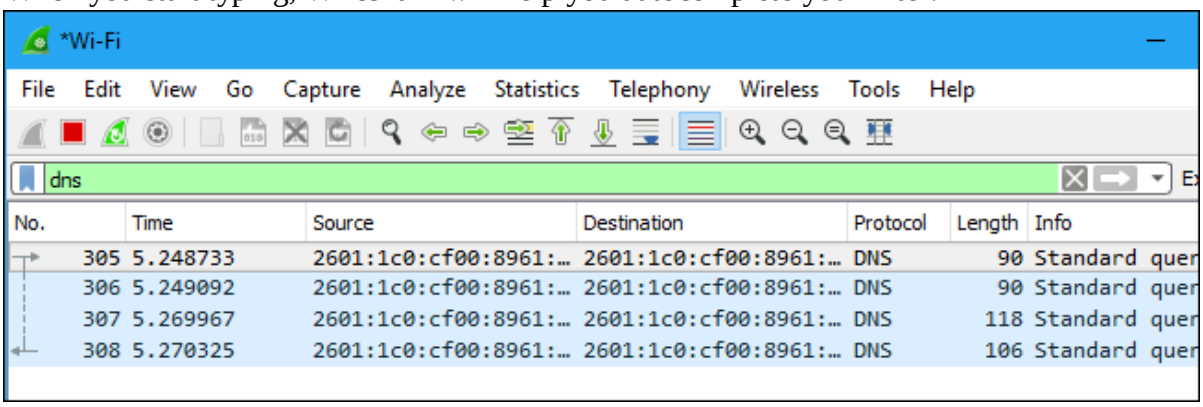
You can also save your own captures in Wireshark and open them later. Click File > Save to save your captured packets.



Filtering Packets

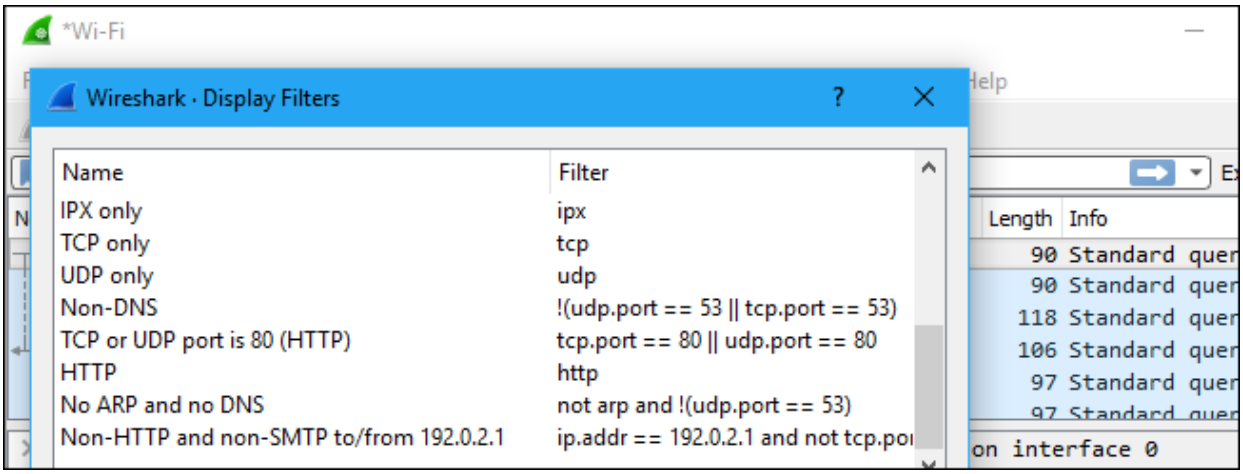
If you're trying to inspect something specific, such as the traffic a program sends when phoning home, it helps to close down all other applications using the network so you can narrow down the traffic. Still, you'll likely have a large amount of packets to sift through. That's where Wireshark's filters come in.

The most basic way to apply a filter is by typing it into the filter box at the top of the window and clicking Apply (or pressing Enter). For example, type "dns" and you'll see only DNS packets. When you start typing, Wireshark will help you autocomplete your filter.

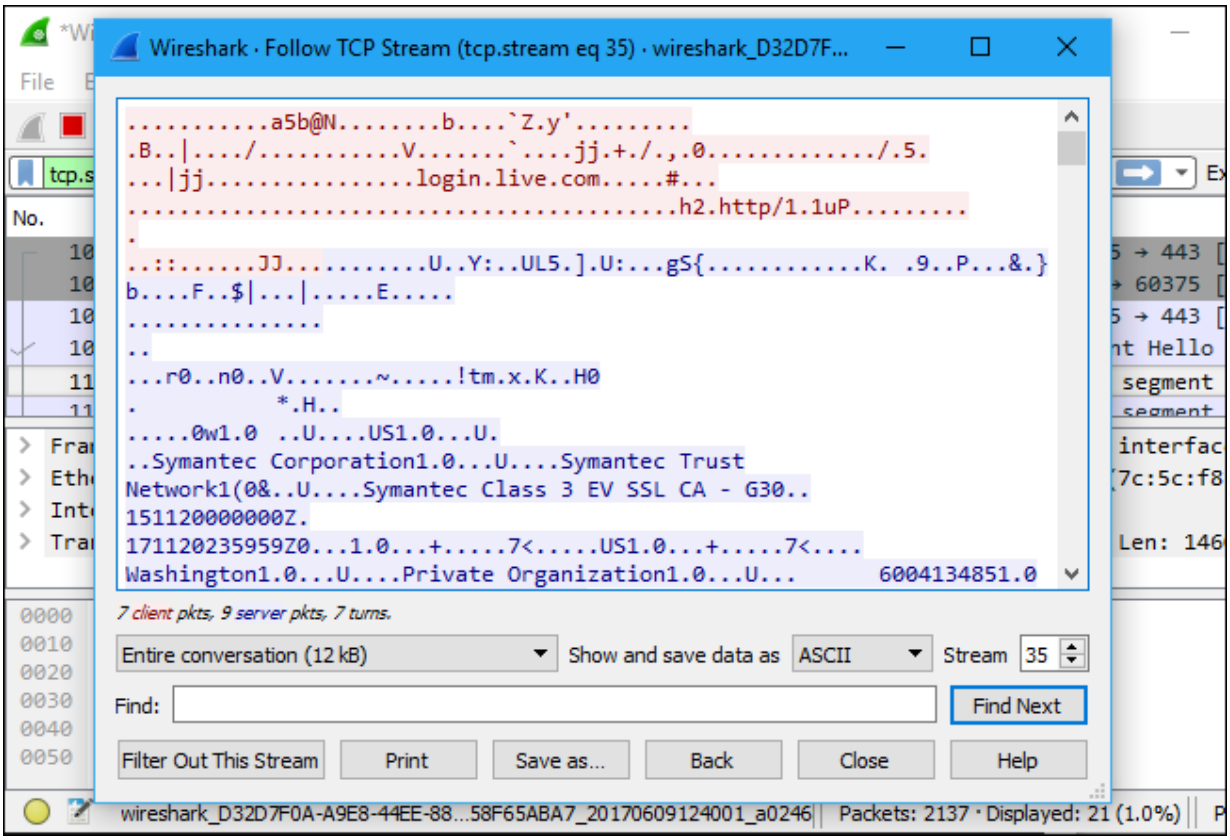


You can also click Analyze > Display Filters to choose a filter from among the default filters included in Wireshark. From here, you can add your own custom filters and save them to easily access them in the future.

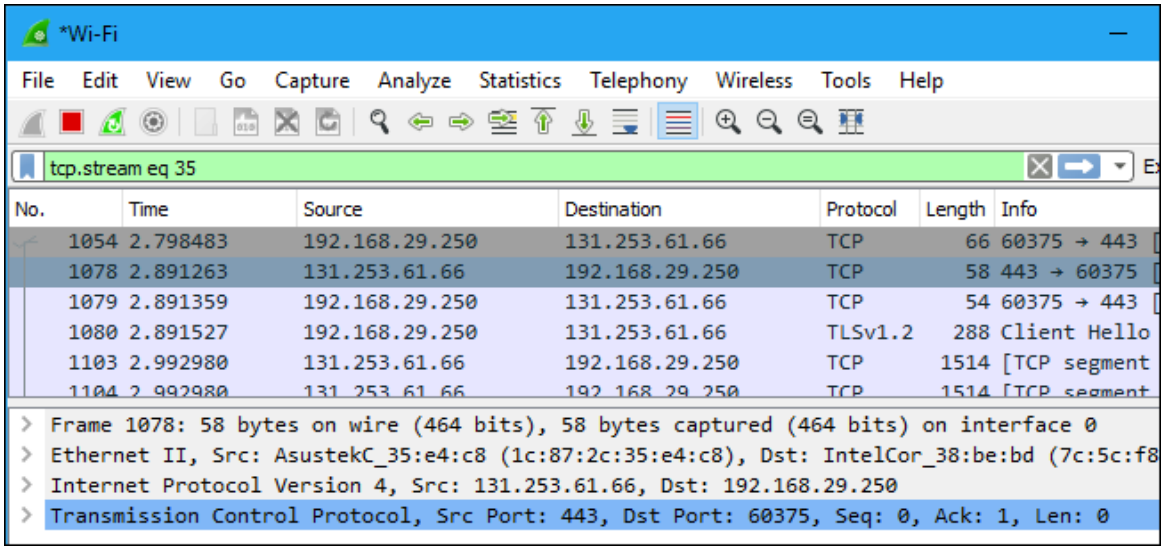
For more information on Wireshark's display filtering language, read the Building display filter expressions page in the official Wireshark documentation.



Another interesting thing you can do is right-click a packet and select Follow > TCP Stream. You'll see the full TCP conversation between the client and the server. You can also click other protocols in the Follow menu to see the full conversations for other protocols, if applicable.

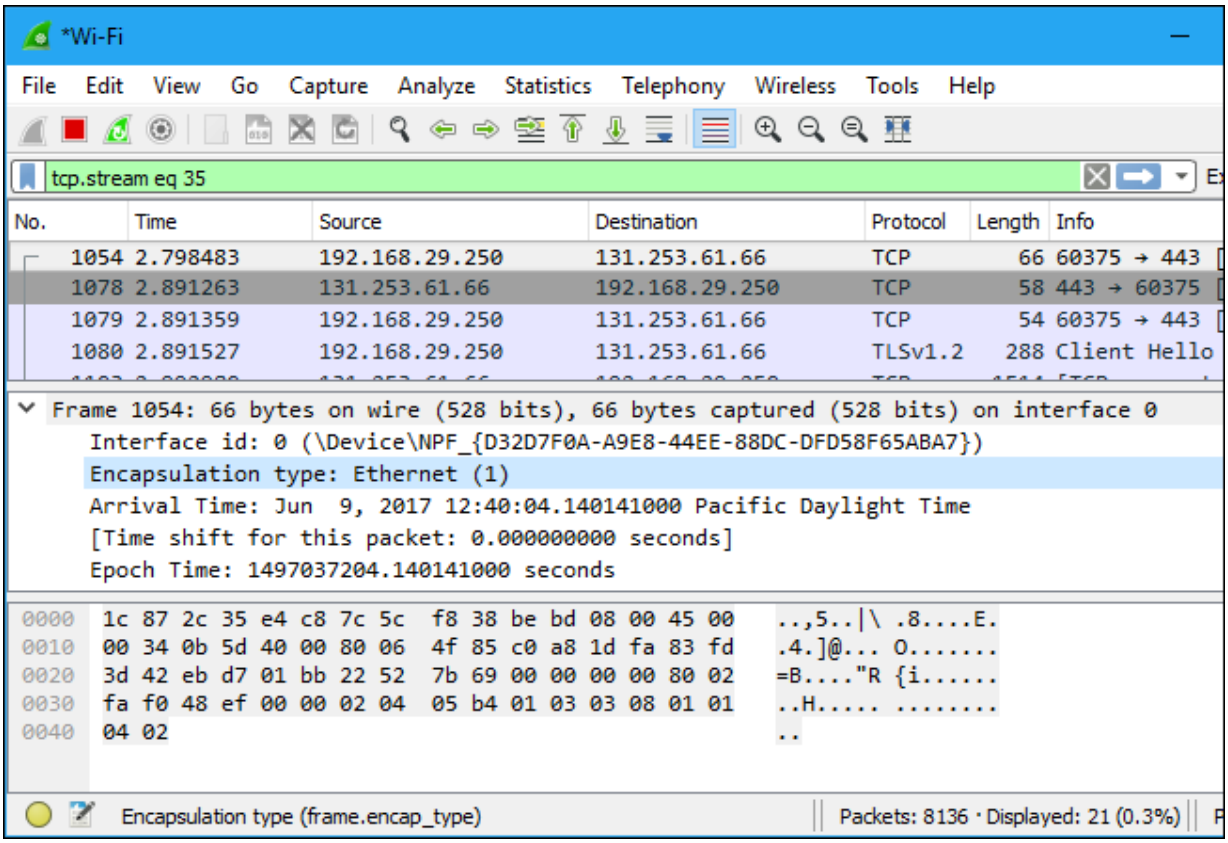


Close the window and you'll find a filter has been applied automatically. Wireshark is showing you the packets that make up the conversation.

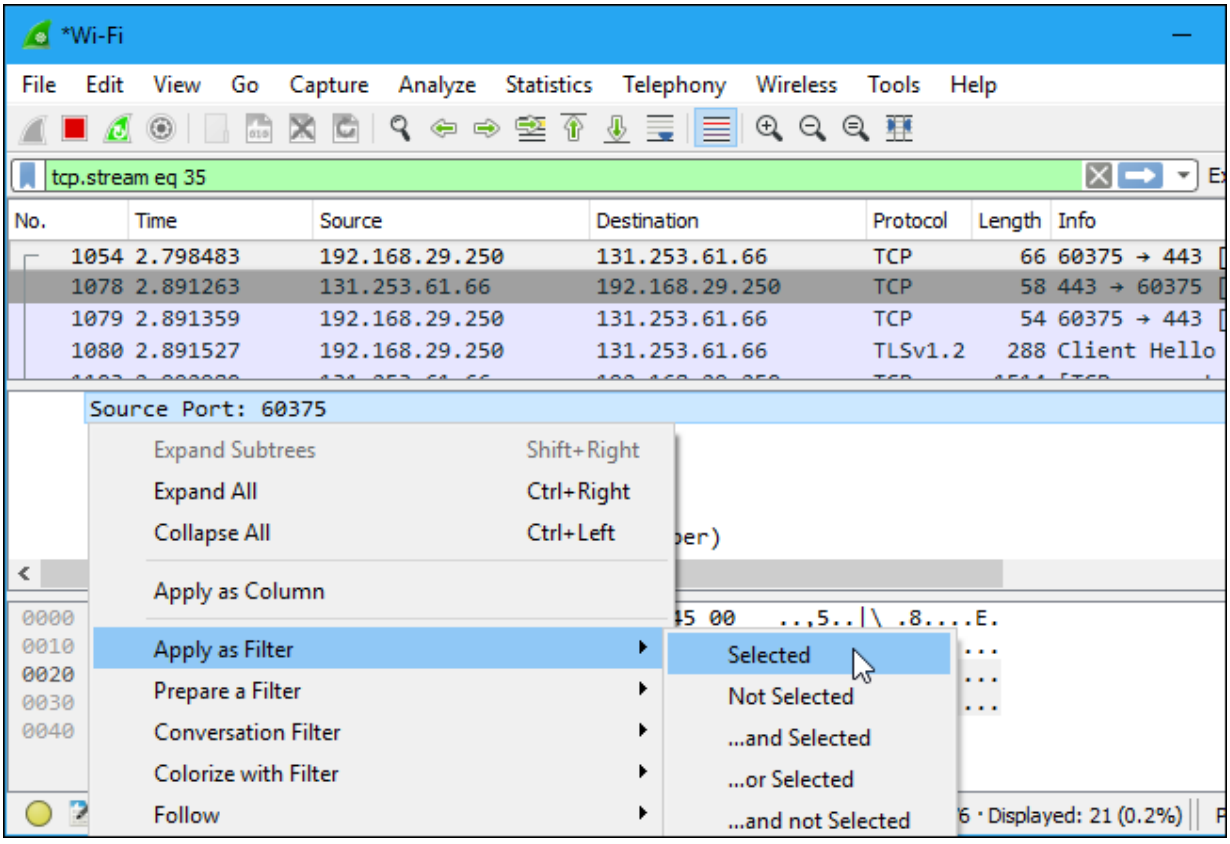


Inspecting Packets

Click a packet to select it and you can dig down to view its details.

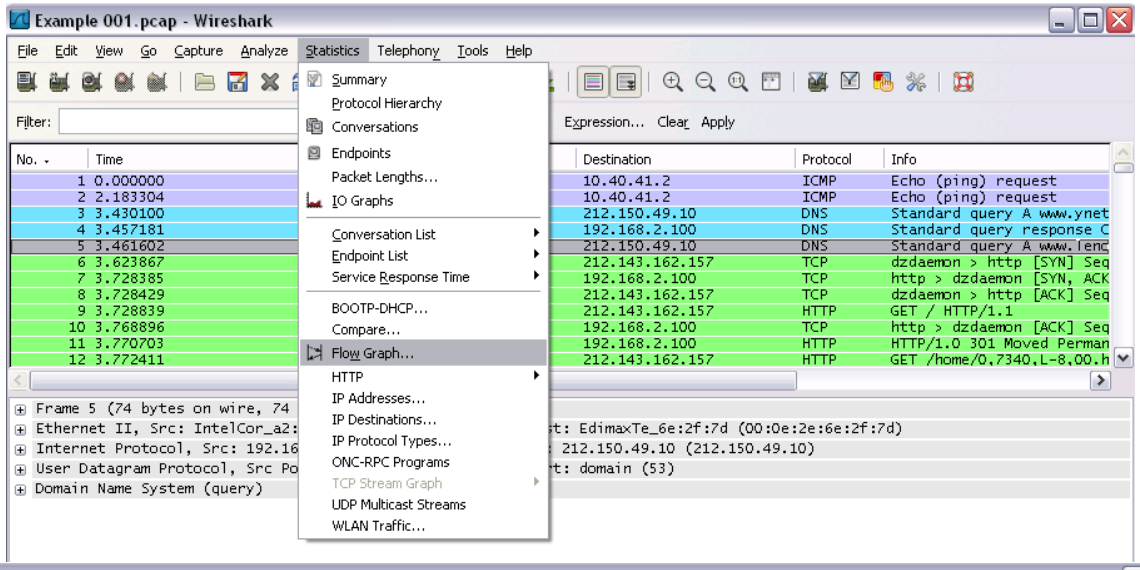


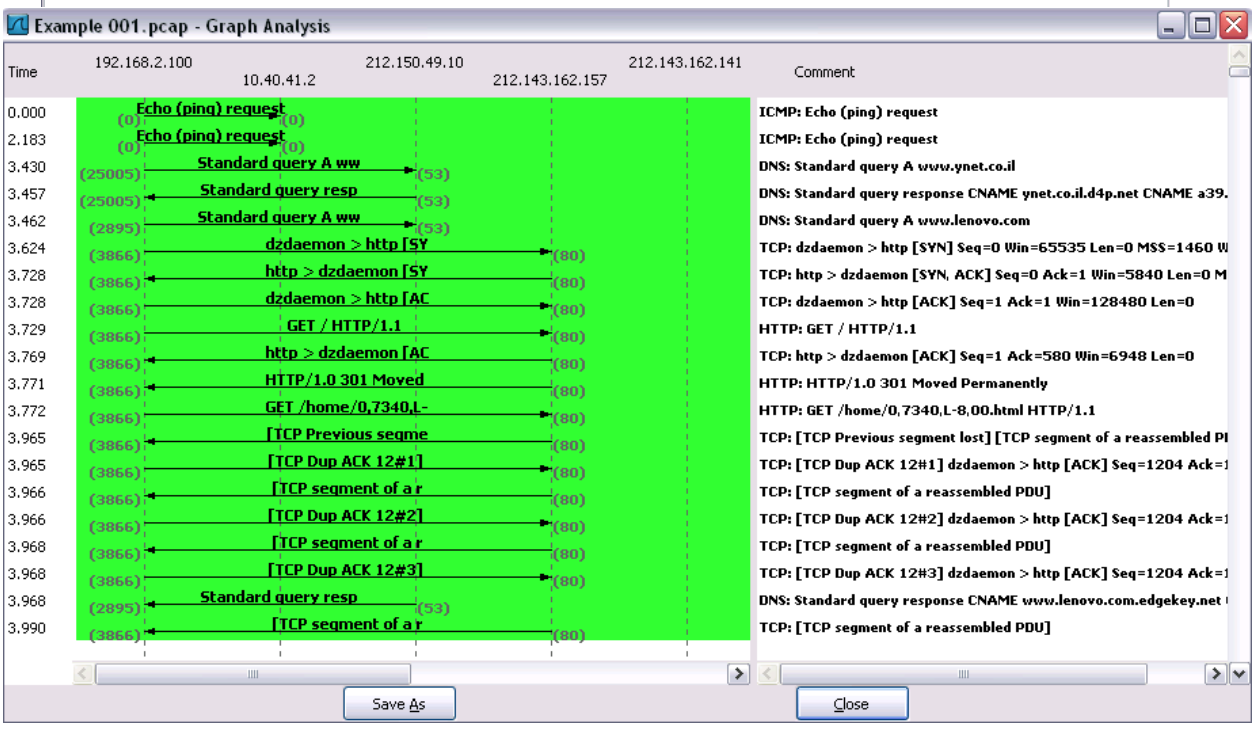
You can also create filters from here — just right-click one of the details and use the Apply as Filter submenu to create a filter based on it.



Wireshark is an extremely powerful tool, and this tutorial is just scratching the surface of what you can do with it. Professionals use it to debug network protocol implementations, examine security problems and inspect network protocol internals.

Flow Graph: Gives a better understanding of what we see.





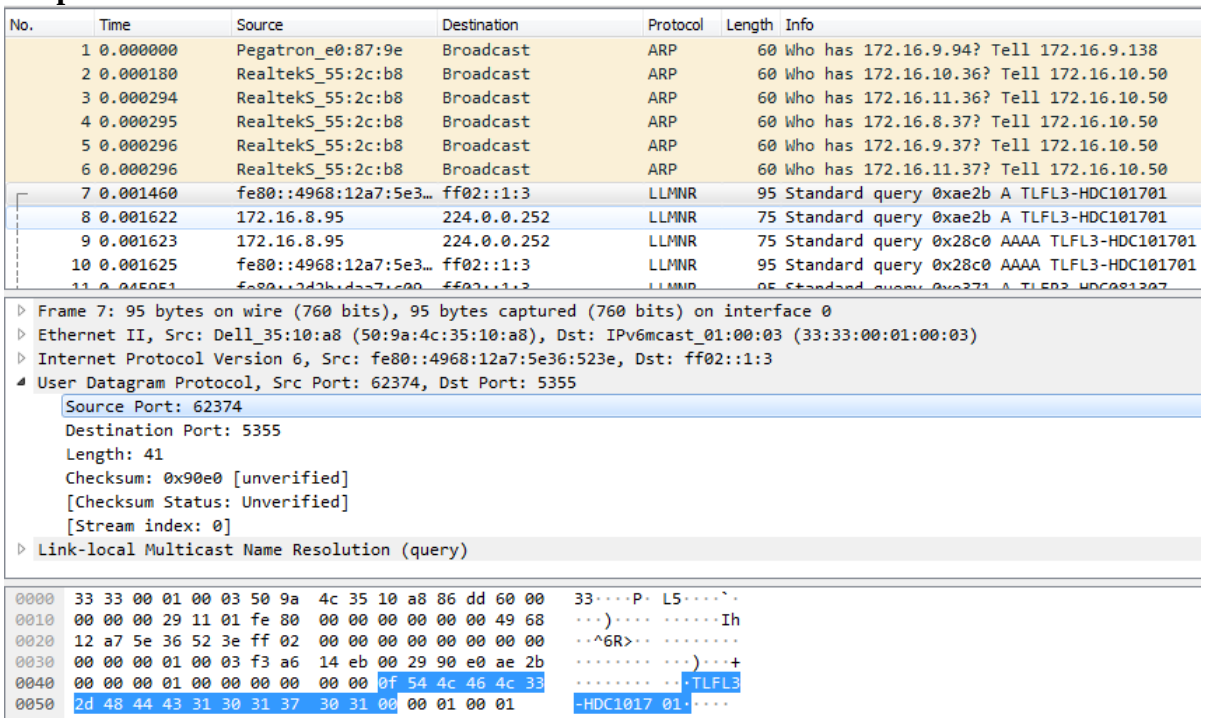
CAPTURING AND ANALYSING PACKETS USING WIRESHARK TOOL

To filter, capture, view, packets in Wireshark Tool.
Capture 100 packets from the Ethernet: IEEE 802.3 LAN Interface and save it.

Procedure

- Select Local Area Connection in Wireshark.
- Go to capture → option
- Select stop capture automatically after 100 packets.
- Then click Start capture.
- Save the packets.

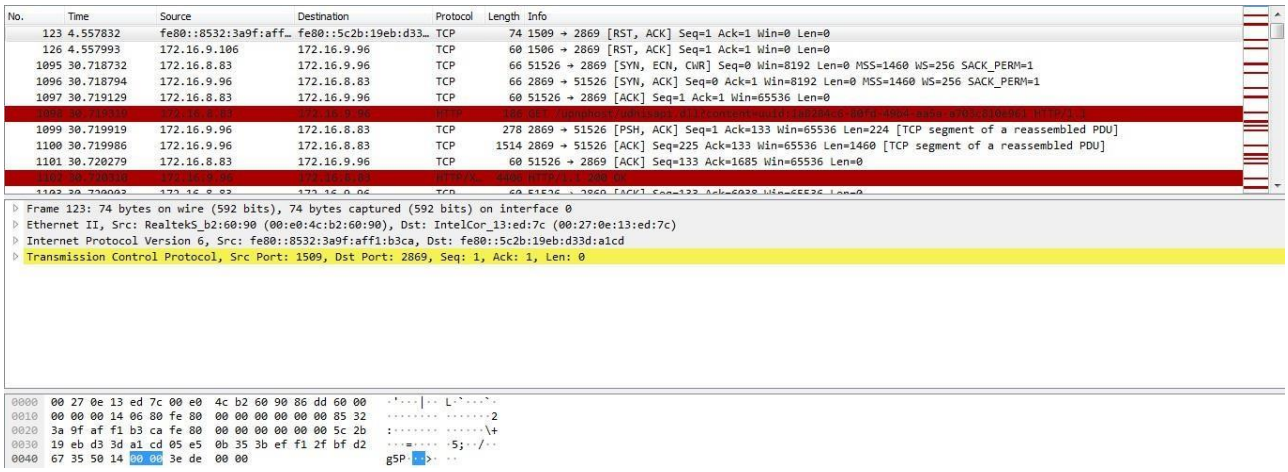
Output



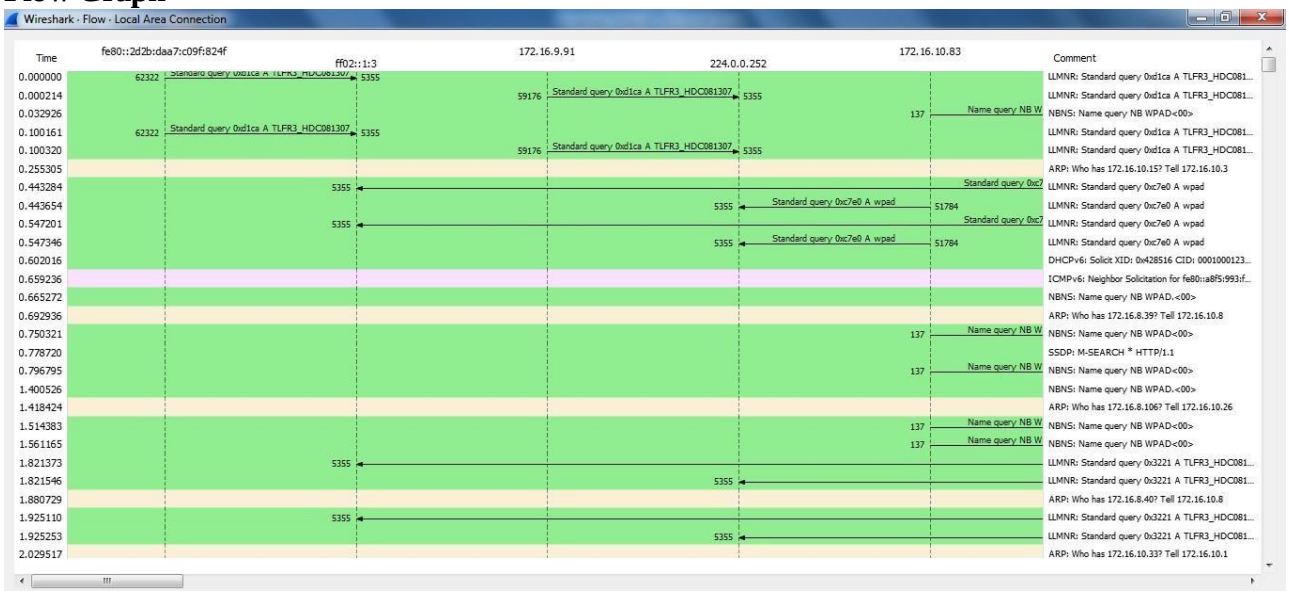
1. **Create a Filter to display only TCP/UDP packets, inspect the packets and provide the flow graph**

Procedure

- Select Local Area Connection in Wireshark.
- Go to capture → option
- Select stop capture automatically after 100 packets.
- Then click Start capture.
- Search TCP packets in search bar.
- To see flow graph click Statistics→Flow graph.
- Save the packets.



Flow Graph

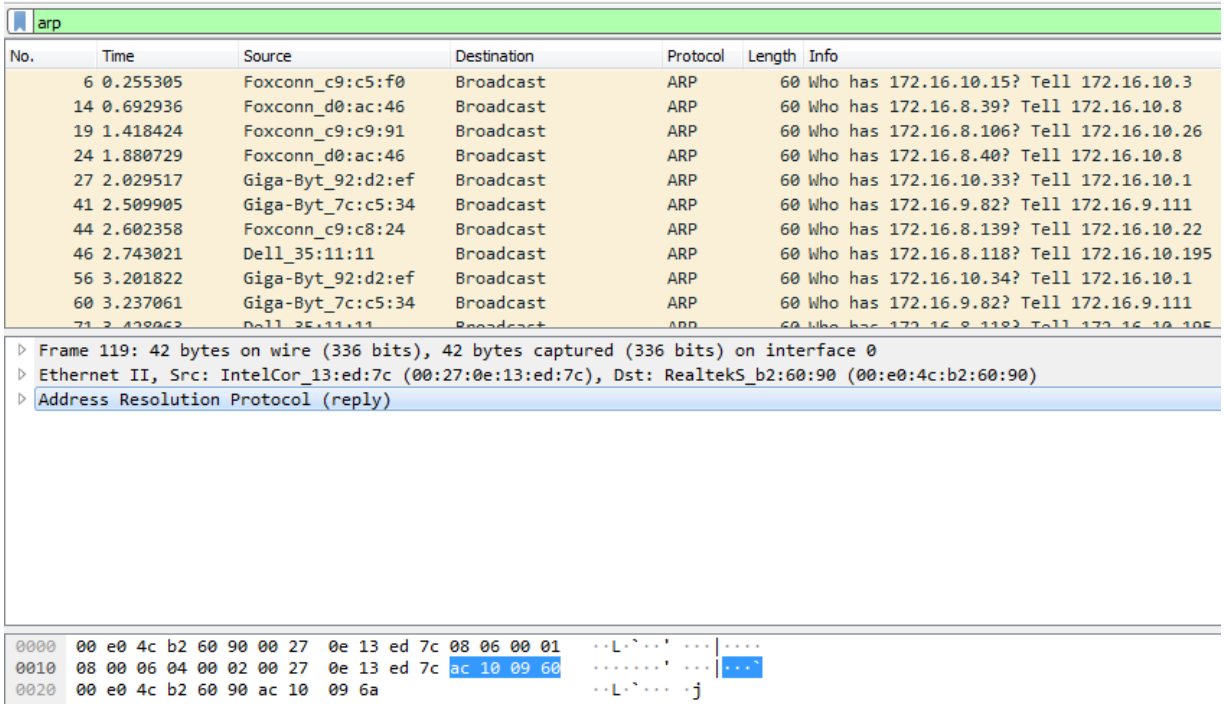


2. Create a Filter to display only ARP packets and inspect the packets.

Procedure

- Go to capture → option
- Select stop capture automatically after 100 packets.
- Then click Start capture.
- Search ARP packets in search bar.
- Save the packets.

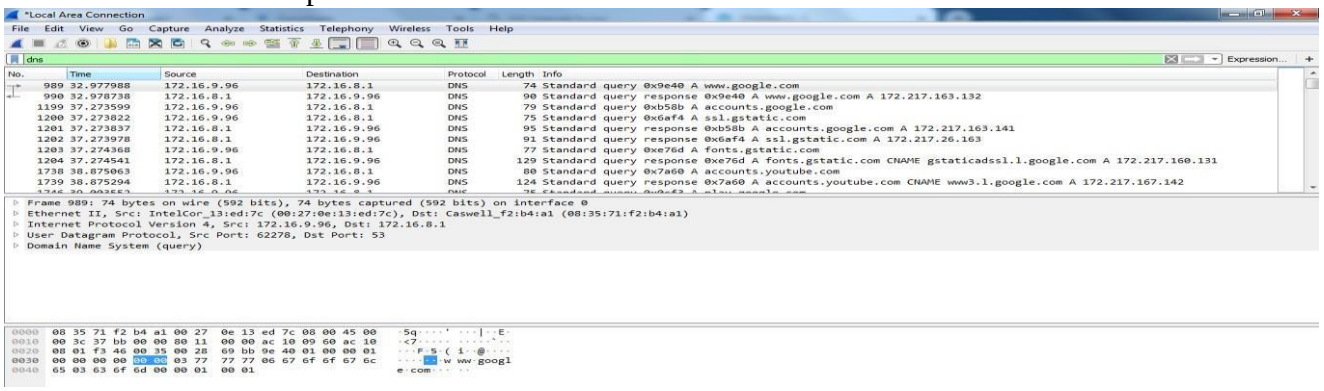
Output

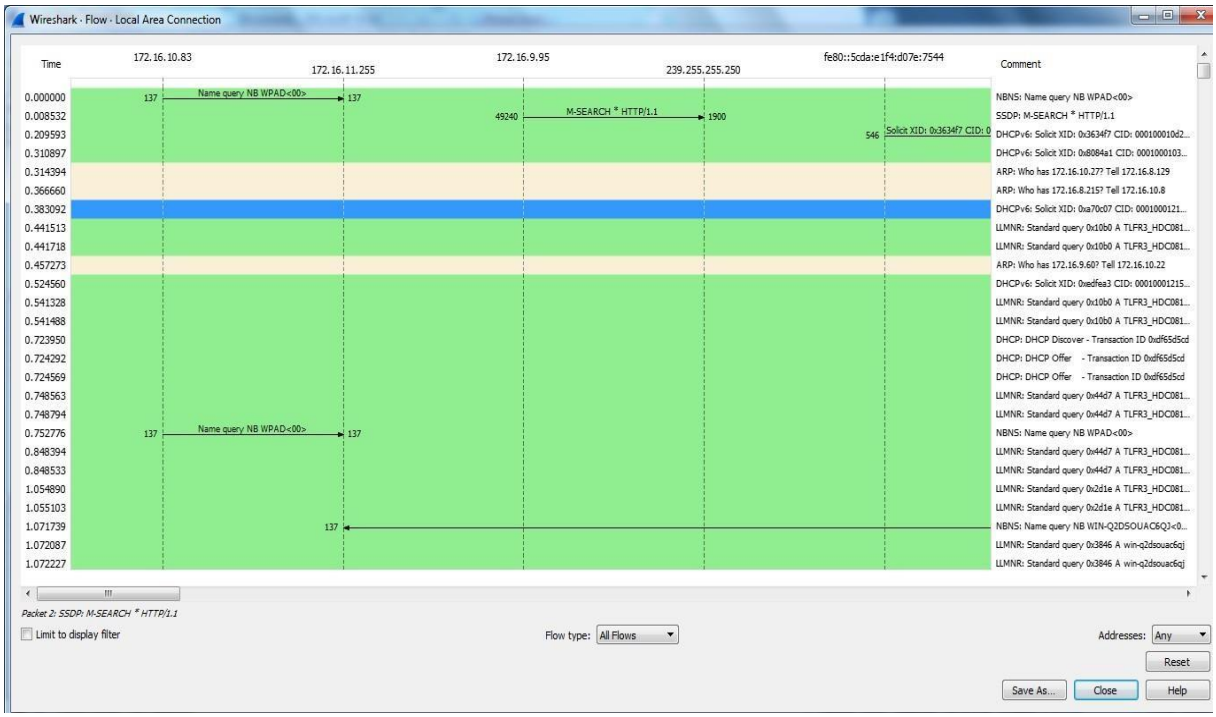


3. Create a Filter to display only DNS packets and provide the flow graph.

Procedure

- Go to capture → option
- Select stop capture automatically after 100 packets.
- Then click Start capture.
- Search DNS packets in search bar.
- To see flow graph click Statistics→Flow graph.
- Save the packets.

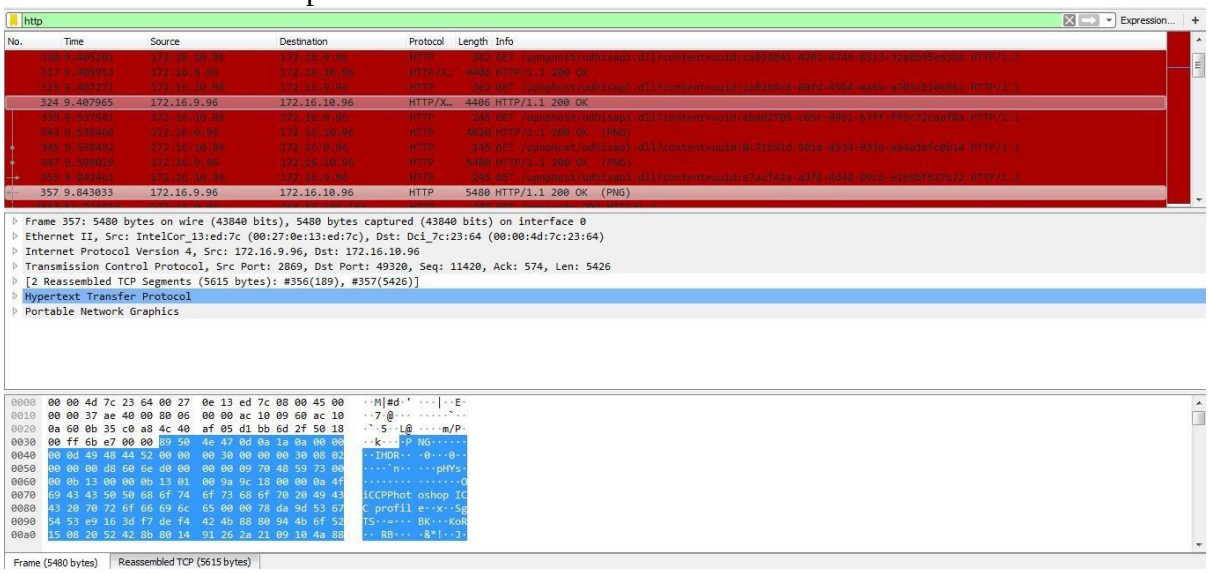




4. Create a Filter to display only HTTP packets and inspect the packets

Procedure

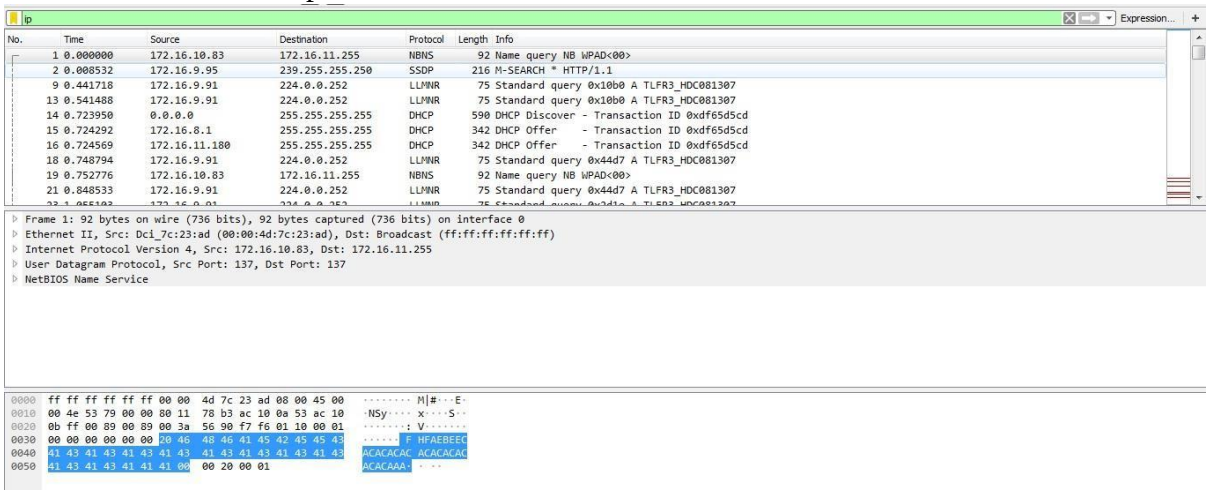
- Select Local Area Connection in Wireshark.
- Go to capture → option
- Select stop capture automatically after 100 packets.
- Then click Start capture.
- Search HTTP packets in search bar.
- Save the packets.



5. Create a Filter to display only IP/ICMP packets and inspect the packets.

Procedure

- Select Local Area Connection in Wireshark.
- Go to capture → option
- Select stop capture automatically after 100 packets.
- Then click Start capture.
- Search ICMP/IP packets in search bar.
- Save the packets

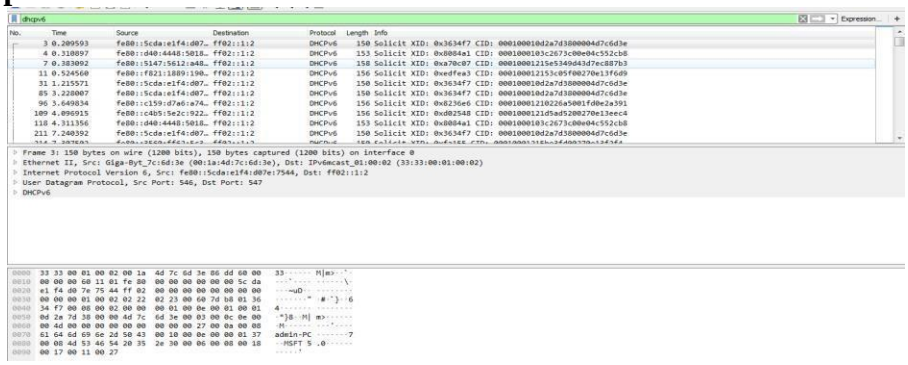


6. Create a Filter to display only DHCP packets and inspect the packets.

Procedure

- Select Local Area Connection in Wireshark.
- Go to capture → option
- Select stop capture automatically after 100 packets.
- Then click Start capture.
- Search DHCP packets in search bar.
- Save the packets

Output



Student observation:

1. **What is promiscuous mode?**
2. **Does ARP packets has transport layer header? Explain.**
3. **Which transport layer protocol is used by DNS?**
4. **What is the port number used by http protocol?**
5. **What is a broadcast ip address?**

Practical-6

AIM: Write a program to implement error detection and correction using HAMMING code concept. Make a test run to input data stream and verify error correction feature.

Error Correction at Data Link Layer:
Hamming code is a set of error-correction codes that can be used to detect and correct the errors that can occur when the data is transmitted from the sender to the receiver. It is a technique developed by R.W. Hamming for error correction.

Create sender program with below features.

1. Input to sender file should be a text of any length. Program should convert the text to binary.
2. Apply hamming code concept on the binary data and add redundant bits to it.
3. Save this output in a file called channel.

Create a receiver program with below features

1. Receiver program should read the input from Channel file.
2. Apply hamming code on the binary data to check for errors.
3. If there is an error, display the position of the error.
4. Else remove the redundant bits and convert the binary data to ascii and display the output.

Student observation:-

Write the code here:

```
import numpy as np

# Function to convert text to binary
def text_to_binary(text):
return ''.join(format(ord(char), '08b') for char in text)

# Function to convert binary to text
def binary_to_text(binary_data):
chars = [chr(int(binary_data[i:i+8], 2)) for i in range(0, len(binary_data), 8)]
return ''.join(chars)

# Function to calculate redundant bits needed for Hamming Code
def calculate_redundant_bits(m):
r = 0
while (2 ** r) < (m + r + 1):
r += 1
return r

# Function to apply Hamming Code to add redundant bits to binary data
def apply_hamming_code(data):
m = len(data)
r = calculate_redundant_bits(m)
```

Practical-6

```
hamming_code = ['0'] * (m + r)
j = 0
k = 1
for i in range(1, m + r + 1):
if i == 2 ** j:
j += 1
else:
hamming_code[i - 1] = data[-k]
k += 1

for i in range(r):
x = 2 ** i
one_count = 0
for j in range(1, m + r + 1):
if j & x and hamming_code[j - 1] == '1':
one_count += 1
hamming_code[x - 1] = '1' if one_count % 2 != 0 else '0'

return ''.join(hamming_code[::-1])

# Function to detect and correct errors in Hamming Code
def detect_and_correct_hamming_code(data):
n = len(data)
r = calculate_redundant_bits(n)
error_position = 0
for i in range(r):
x = 2 ** i
one_count = 0
for j in range(1, n + 1):
if j & x and data[-j] == '1':
one_count += 1
if one_count % 2 != 0:
error_position += x

if error_position > 0:
print(f"Error detected at position: {error_position}")
data = list(data)
data[-error_position] = '1' if data[-error_position] == '0' else '0'
data = ''.join(data)
else:
print("No error detected.")

# Removing redundant bits
corrected_data = []
j = 0
for i in range(1, n + 1):
if i != 2 ** j:
corrected_data.append(data[-i])
else:
```

Practical-6

```
j += 1

return ''.join(corrected_data[::-1])

# Sender Program
def sender(text):
binary_data = text_to_binary(text)
encoded_data = apply_hamming_code(binary_data)
with open("channel.txt", "w") as f:
f.write(encoded_data)
print("Data has been sent to channel.txt.")

# Receiver Program
def receiver():
with open("channel.txt", "r") as f:
encoded_data = f.read()
print(f"Received encoded data: {encoded_data}")
corrected_data = detect_and_correct_hamming_code(encoded_data)
decoded_text = binary_to_text(corrected_data)
print(f"Decoded text after error correction: {decoded_text}")

# Example Usage
# Enter text to send
text_to_send = "Hello"  # Modify the text to test
sender(text_to_send)
receiver()
```

Input:-
Text to send: "Hello"

Output:

```
Received encoded data: 01001000011001101011011000110101000110011101 00
No error detected.
Decoded text after error correction: Hello
```

Practical-7

AIM: Write a program to implement flow control at data link layer using SLIDING WINDOW PROTOCOL. Simulate the flow of frames from one node to another.

Program should achieve at least below given requirements. You can make it a bidirectional program wherein receiver is sending its data frames with acknowledgement (Piggybacking).

Create a sender program with following features:-

1. Input Window size from the user.
2. Input a Text message from the user.
3. Consider 1 character per frame.
4. Create a frame with following fields [Frame no., DATA].
5. Send the frames. [Print the output on screen and save it in a file called Sender_Buffer.]
6. Wait for the acknowledgement from the Receiver. [Induce delay in the program]
7. Reader a file called Receiver_Buffer.
8. Check ACK field for the Acknowledgement number.
9. If the Acknowledgement number is as expected, send new set of frames accordingly, [overwrite the Sender_Buffer file with new frames] Else if NACK is received, resend the frames accordingly. [Overwrite the Sender_Buffer with old frame].

Create a receiver file with following features

1. Reader a file called Sender_Buffer.
2. Check the Frame no.
3. If the Fame no. are as expected, write the appropriate ACK no. in the Receiver_ Buffer file. Else write NACK no. in the Receiver_Buffer file.

NOTE: Induce error and verify the behaviour of the program. Manually Change the Frame no and Ack no in the files].

Student observation:

Write the code here:

```
import random

# Constants for ACK and NACK
ACK = "ACK"
NACK = "NACK"

# Sliding window protocol implementation
def sliding_window_protocol(window_size, message):
# Split the message into frames (1 character per frame)
frames = [(i, message[i]) for i in range(len(message))]
current_window_start = 0
current_window_end = min(window_size, len(frames))
expected_ack = current_window_start

while current_window_start < len(frames):
print("\n--- SENDING WINDOW ---")
```

Practical-7

```
# Send frames in the current window
for i in range(current_window_start, current_window_end):
frame_no, data = frames[i]
print(f"Sent frame {frame_no} with data '{data}'.")

# Simulate Receiver side: Acknowledge frames
print("\n--- RECEIVER WINDOW ---")
for i in range(current_window_start, current_window_end):
frame_no, data = frames[i]

# Randomly decide to send ACK or NACK (to simulate error)
ack_type = ACK if random.random() > 0.2 else NACK  # 20% chance of NACK
if ack_type == ACK:
print(f"Frame {frame_no} received correctly. Sending {ACK}.")
else:
print(f"Error in frame {frame_no}. Sending {NACK}.")
expected_ack = frame_no  # Reset to the frame needing retransmission
break
else:
# If no errors in the window, move to the next set of frames
expected_ack = current_window_end

# Update window based on the acknowledgments received
if expected_ack == current_window_end:
print("All frames in the window acknowledged correctly. Moving window forward.")
current_window_start += window_size
current_window_end = min(current_window_start + window_size, len(frames))
else:
print(f"Resending frames from frame {expected_ack} due to error.")
current_window_start = expected_ack
current_window_end = min(current_window_start + window_size, len(frames))

print("\nAll frames sent and acknowledged successfully.")

# User input for window size and message
window_size = int(input("Enter window size: "))
message = input("Enter the message to send: ")

# Run the sliding window protocol
sliding_window_protocol(window_size, message)
```

Practical-7

Input:

Enter window size: 3
Enter the message to send: HELLO

Output:

--- SENDING WINDOW ---
Sent frame 0 with data 'H'.
Sent frame 1 with data 'E'.
Sent frame 2 with data 'L'.
--- RECEIVER WINDOW ---
Error in frame 0.
Sending NACK.
Resending frames from frame 0 due to error.
--- SENDING WINDOW ---
Sent frame 0 with data 'H'.
Sent frame 1 with data 'E'.
Sent frame 2 with data 'L'.
--- RECEIVER WINDOW ---
Frame 0 received correctly.
Sending ACK.
Frame 1 received correctly.
Sending ACK.
Frame 2 received correctly.
Sending ACK.
All frames in the window acknowledged correctly.
Moving window forward.
--- SENDING WINDOW ---
Sent frame 3 with data 'L'.
Sent frame 4 with data 'O'.
--- RECEIVER WINDOW ---
Error in frame 3.
Sending NACK.
Resending frames from frame 3 due to error.
--- SENDING WINDOW ---
Sent frame 3 with data 'L'.
Sent frame 4 with data 'O'.
--- RECEIVER WINDOW ---
Frame 3 received correctly.
Sending ACK. Error in frame 4.
Sending NACK.
Resending frames from frame 4 due to error.

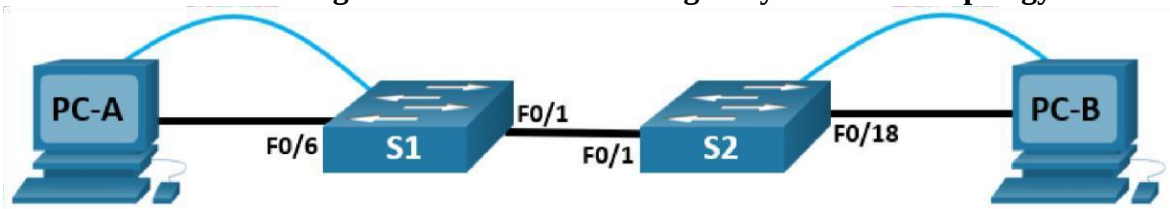
Practical-7

--- SENDING WINDOW ---
Sent frame 4 with data 'O'.
--- RECEIVER WINDOW ---
Frame 4 received correctly.
Sending ACK.
All frames in the window acknowledged correctly.
Moving window forward.
All frames sent and acknowledged successfully.

Practical-8

AIM: - a) Simulate Virtual LAN configuration using CISCO Packet Tracer Simulation.

Packet Tracer - Configure VLANs and Trunking - Physical Mode Topology



Addressing Table

Device	Interface	IP Address	Subnet Mask	Default Gateway
S1	VLAN 1	192.168.1.11	255.255.255.0	N/A
S2	VLAN 1	192.168.1.12	255.255.255.0	N/A
PC-A	NIC	192.168.10.3	255.255.255.0	192.168.10.1
PC-B	NIC	192.168.10.4	255.255.255.0	192.168.10.1

Blank Line - no additional information

Objectives

Part 1: Build the Network and Configure Basic Device Settings
Part 2: Create VLANs and Assign Switch Ports
Part 3: Maintain VLAN Port Assignments and the VLAN Database Part 4: Configure an 802.1Q Trunk between the Switches

Background / Scenario

Modern switches use virtual local-area networks (VLANs) to improve network performance by separating large Layer 2 broadcast domains into smaller ones. VLANs can also be used as a security measure by controlling which hosts can communicate. In general, VLANs make it easier to design a network to support the goals of an organization.
VLAN trunks are used to span VLANs across multiple devices. Trunks allow the traffic from multiple VLANS to travel over a single link, while keeping the VLAN identification and segmentation intact.
In this Packet Tracer Physical Mode (PTPM) activity, you will create VLANs on both switches in the topology, assign VLANs to switch access ports, and verify that VLANs are working as expected. You will then create a VLAN trunk between the two switches to allow hosts in the same VLAN to communicate through the trunk, regardless of which switch to which the host is attached.

Instructions

Part 1: Build the Network and Configure Basic Device Settings

Step 1: Build the network as shown in the topology.

Attach the devices as shown in the topology diagram, and cable as necessary. a. Click and drag both switch **S1** and **S2** to the **Rack**.

b. Click and drag both **PC-A** and **PC-B** to the **Table** and use the power button to turn them on.

c. Provide network connectivity by connecting **Copper Straight-through** cables, as shown in the topology.
d. Connect **Console Cable** from device **PC-A** to **S1** and from device **PC-B** to **S2**.

Step 2: Configure basic settings for each switch.

a. From the **Desktop Tab** on each PC, use the **Terminal** to console into each switch and enable privileged EXEC mode.
Open configuration window
b. Enter configuration mode.
c. Assign a device name to each switch.
d. Assign **class** as the privileged EXEC encrypted password.
e. Assign **cisco** as the console password and enable login.
f. Assign **cisco** as the vty password and enable login.
g. Encrypt the plaintext passwords.
h. Create a banner that warns anyone accessing the device that unauthorized access is prohibited.
i. Configure the IP address listed in the Addressing Table for VLAN 1 on the switch.
Note: The VLAN 1 address is not grade because you will remove it later in the activity. However, you will need VLAN 1 to test connectivity later in this Part.
j. Shut down all interfaces that will not be used.
k. Set the clock on each switch.
Note: The clock setting cannot be graded in Packet Tracer.
l. Save the running configuration to the startup configuration file.
Close configuration window

Step 3: Configure PC hosts.

From the **Desktop** tab on each **PC**, click IP Configuration and enter the addressing information as displayed in the Addressing Table.

Step 4: Test connectivity.

Test network connectivity by attempting to ping between each of the cabled devices.

Questions:

Can PC-A ping PC-B?
Can PC-A ping S1?
Can PC-B ping S2?
Can S1 ping S2?

Close configuration window

Part 2: Create VLANs and Assign Switch Ports

In Part 2, you will create Management, Operations, Parking Lot, and Native VLANs on both switches. You will then assign the VLANs to the appropriate interface. The **show vlan** command is used to verify your configuration settings.

Step 1: Create VLANs on the switches.

From the **Desktop Tab** on each **PC**, use Terminal to continue configuring both network switches.

Open configuration window

a. Create the VLANs on **S1**.

```
S1(config)# vlan 10
S1(config-vlan)# name Operations
S1(config-vlan)# vlan 20
S1(config-vlan)# name Parking_Lot
S1(config-vlan)# vlan 99
S1(config-vlan)# name Management
S1(config-vlan)# vlan 1000
S1(config-vlan)# name Native
S1(config-vlan)# end
```

b. Create the same VLANs on S2.

c. Issue the **show vlan brief** command to view the list of VLANs on **S1**.

```
S1# show vlan brief

VLAN Name                              Status    Ports
---- --------------------------------- --------- -------------------------------
1    default                           active    Fa0/1, Fa0/2, Fa0/3, Fa0/4
                                                 Fa0/5, Fa0/6, Fa0/7, Fa0/8
                                                 Fa0/9, Fa0/10, Fa0/11, Fa0/12
                                                 Fa0/13, Fa0/14, Fa0/15, Fa0/16
                                                 Fa0/17, Fa0/18, Fa0/19, Fa0/20
                                                 Fa0/21, Fa0/22, Fa0/23, Fa0/24
                                                 Gi0/1, Gi0/2
10   Operations                        active
20   Parking_Lot                       active
99   Management                        active
1000 Native                            active
1002 fddi-default                      active
1003 token-ring-default                active
1004 fddinet-default                   active
1005 trnet-default                     active
```

Questions:

What is the default VLAN?

What ports are assigned to the default VLAN?

Step 2: Assign VLANs to the correct switch interfaces.

a. Assign VLANs to the interfaces on **S1**.
   1) Assign PC-A to the Operation VLAN.
      S1(config)# **interface f0/6**
      S1(config-if)# **switchport mode access**
      S1(config-if)# **switchport access vlan 10**
   2) From VLAN 1, remove the management IP address and configure it on VLAN 99.
      S1(config)# **interface vlan 1**
      S1(config-if)# **no ip address**
      S1(config-if)# **interface vlan 99**
      S1(config-if)# **ip address 192.168.1.11 255.255.255.0**
      S1(config-if)# **end**

b. Issue the **show vlan brief** command and verify that the VLANs are assigned to the correct interfaces. c. Issue the **show ip interface brief** command.
   Question:
   What is the status of VLAN 99? Explain.

d. Assign **PC-B** to the Operations VLAN on **S2**.
e. From VLAN 1, remove the management IP address and configure it on VLAN 99 according to the Addressing Table.
f. Use the **show vlan brief** command to verify that the VLANs are assigned to the correct interfaces.
   Questions:
   Is S1 able to ping S2? Explain.
   Is PC-A able to ping PC-B? Explain.

Part 3: Maintain VLAN Port Assignments and the VLAN Database

In Part 3, you will change port VLAN assignments and remove VLANs from the VLAN database.

Step 1: Assign a VLAN to multiple interfaces.

From the **Desktop Tab** on each **PC**, use **Terminal** to continue configuring both network switches.

Open configuration window

a. On S1, assign interfaces F0/11 – 24 to VLAN99.
   S1(config)# **interface range f0/11-24**
   S1(config-if-range)# **switchport mode access**
   S1(config-if-range)# **switchport access vlan 99**
   S1(config-if-range)# **end**
b. Issue the **show vlan brief** command to verify VLAN assignments.
c. Reassign F0/11 and F0/21 to VLAN 10.
d. Verify that VLAN assignments are correct.

Step 2: Remove a VLAN assignment from an interface.

a. Use the **no switchport access vlan** command to remove the VLAN 99 assignment to F0/24.
   S1(config)# **interface f0/24**
   S1(config-if)# **no switchport access vlan**
   S1(config-if)# **end**
b. Verify that the VLAN change was made.
   Question:

Which VLAN is F0/24 now associated with?

Step 3: Remove a VLAN ID from the VLAN database.

a. Add VLAN 30 to interface F0/24 without issuing the global VLAN command.
S1(config)# **interface f0/24**
S1(config-if)# **switchport access vlan 30**
% Access VLAN does not exist. Creating vlan 30
Note: Current switch technology no longer requires that the **vlan** command be issued to add a VLAN to the database. By assigning an unknown VLAN to a port, the VLAN will be created and added to the VLAN database.

b. Verify that the new VLAN is displayed in the VLAN table.
Question:
What is the default name of VLAN 30?

c. Use the **no vlan 30** command to remove VLAN 30 from the VLAN database.
S1(config)# **no vlan 30**
S1(config)# **end**

d. Issue the **show vlan brief** command. F0/24 was assigned to VLAN 30.
Question:
After deleting VLAN 30 from the VLAN database, why is F0/24 no longer displayed in the output of the **show vlan brief** command? What VLAN is port F0/24 now assigned to? What happens to the traffic destined to the host that is attached to F0/24?

e. On interface F0/24, issue the **no switchport access vlan** command.

f. Issue the **show vlan brief** command to determine the VLAN assignment for F0/24.
Questions:
To which VLAN is F0/24 assigned?

Note: Before removing a VLAN from the database, it is recommended that you reassign all the ports assigned to that VLAN.

Why should you reassign a port to another VLAN before removing the VLAN from the VLAN database?

Close configuration window.

Part 4: Configure an 802.1Q Trunk Between the Switches

In Part 4, you will configure interface F0/1 to use the Dynamic Trunking Protocol (DTP) to allow it to negotiate the trunk mode. After this has been accomplished and verified, you will disable DTP on interface F0/1 and manually configure it as a trunk.

Step 1: Use DTP to initiate trunking on F0/1.

The default DTP mode of a 2960 switch port is dynamic auto. This allows the interface to convert the link to a trunk if the neighboring interface is set to trunk or dynamic desirable mode.

Open configuration window

a. On **S1**, set F0/1 to negotiate trunk mode.

S1(config)# **interface f0/1**
S1(config-if)# **switchport mode dynamic desirable**

Sep 19 02:51:47.257: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/1, changed state to up
Sep 19 02:51:47.291: %LINEPROTO-5-UPDOWN: Line protocol on Interface Vlan99, changed state to up
You should also receive link status messages on S2.
S2#
Sep 19 02:42:19.424: %LINK-3-UPDOWN: Interface FastEthernet0/1, changed state to up
Sep 19 02:42:21.454: %LINEPROTO-5-UPDOWN: Line protocol on Interface Vlan99, changed state to up

Sep 19 02:42:22.419: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/1, changed state to up

b. On **S1** and **S2**, issue the **show vlan brief** command. Interface F0/1 is no longer assigned to VLAN 1. Trunked interfaces are not listed in the VLAN table.
c. Issue the **show interfaces trunk** command to view trunked interfaces. Notice that the mode on **S1** is set to desirable, and the mode on **S2** is set to auto.
S1# **show interfaces trunk**

S2# **show interfaces trunk**
Note: By default, all VLANs are allowed on a trunk. The **switchport trunk** command allows you to control what VLANs have access to the trunk. For this activity, keep the default settings. This allows all VLANs to traverse F0/1.

Close configuration window

d. Verify that VLAN traffic is traveling over trunk interface F0/1.
Questions:
Can S1 ping S2?
Can PC-A ping PC-B?
Can PC-A ping S1?
Can PC-B ping S2?

Step 2: Manually configure trunk interface F0/1.

The **switchport mode trunk** command is used to manually configure a port as a trunk. This command should be issued on both ends of the link.

a. On interface F0/1, change the switchport mode to force trunking. Make sure to do this on both switches.

Open configuration window

S1(config)# **interface f0/1**
S1(config-if)# **switchport mode trunk**

b. Issue the **show interfaces trunk** command to view the trunk mode. Notice that the mode changed from **desirable** to **on**.
S1# **show interfaces trunk**
c. Modify the trunk configuration on both switches by changing the native VLAN from VLAN 1 to VLAN 1000.
S1(config)# **interface f0/1**
S1(config-if)# **switchport trunk native vlan 1000**
d. Issue the **show interfaces trunk** command to view the trunk. Notice the Native VLAN information is updated.
S2# **show interfaces trunk**
Questions:
Why might you want to manually configure an interface to trunk mode instead of using DTP?
Why might you want to change the native VLAN on a trunk?

Close configuration window

Reflection Questions

1. What is needed to allow hosts on VLAN 10 to communicate to hosts on VLAN 99?
2. What are some primary benefits that an organization can receive through effective use of VLANs?

_____

b) Design and configure a VLAN for the below given scenario.

There are 10 faculty in Robotics department, sitting in 3 different blocks. Design and configure a Virtual LAN for Robotics department (using switch and Ethernet cables) so that all the faculty are logically in the same LAN.

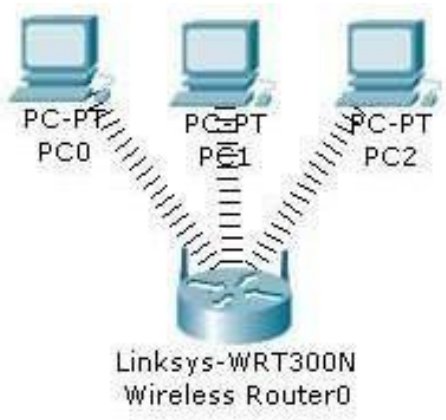
Student observation:-

a) Draw and Label the VLAN for Qb).
b) Show the ip configuration for each device.
c) Write the commands used for VLAN configuration in switch.

Practical-8

AIM:-b) Configuration of Wireless LAN using CISCO Packet Tracer.

Design a topology with three PCs connected from Linksys Wireless routers.



Perform following configuration:-

- Configure Static IP on PC and Wireless Router
- Set SSID to MotherNetwork
- Set IP address of router to 192.168.0.1, PC0 to 192.168.0.2, PC1 to 192.168.0.3 and PC2 to 192.168.0.4.
- Secure your network by configuring WAP key on Router
- Connect PC by using WAP key

To complete these tasks follow these step by step instructions:-

Step1:- Click on wireless router,

- Select Administration tab from top Menu, set username and password to admin and click on Save Setting.

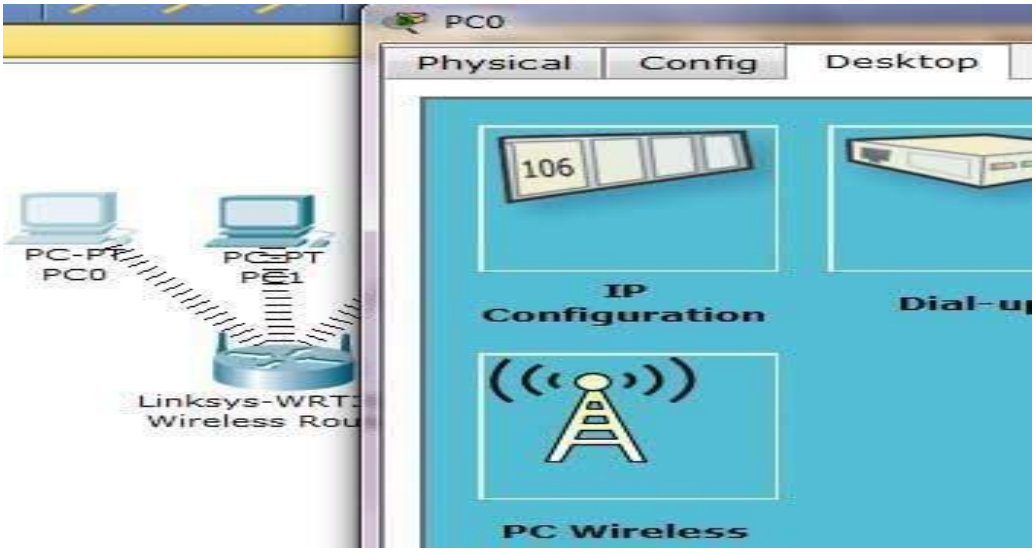
- Next click on wireless tab and set default SSID to MotherNetwork.
- Now Select wireless security and change Security Mode to WEP

- Set Key1 to 0123456789

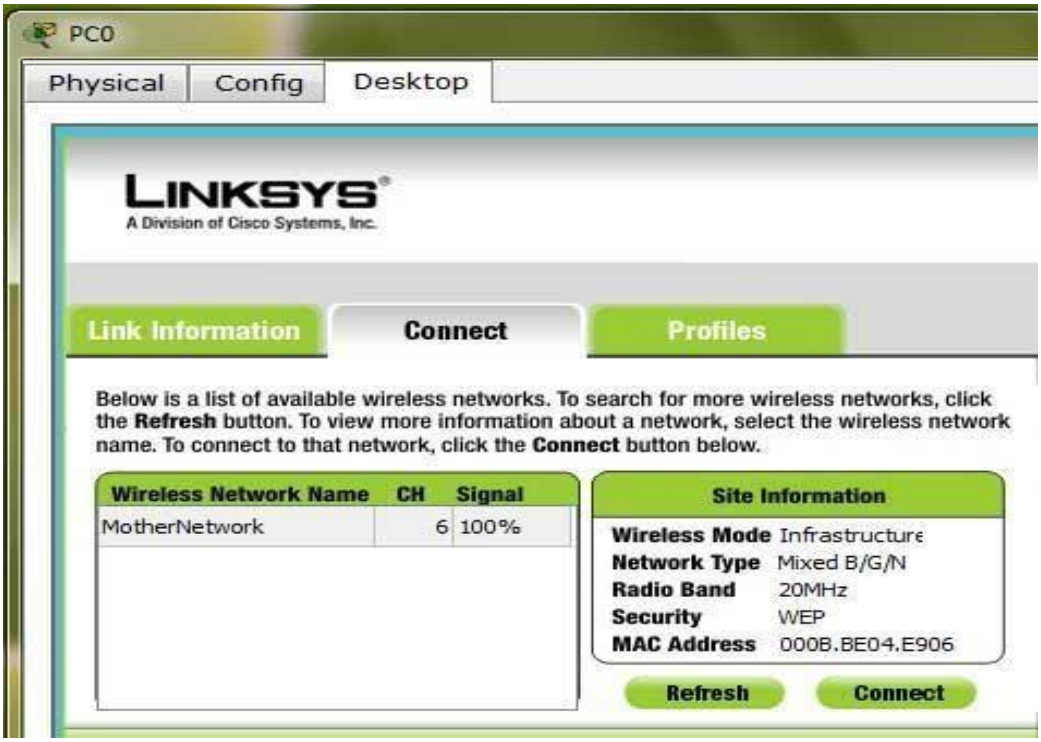
- Again go in the end of page and Click on Save Setting
- Now we have completed all given task on Wireless router. Now configure the static IP on all three PC's
- Double click on pc select Desktop tab click on IP configuration select Static IP and set IP as given below

PC	IP	Subnet Mask	Default Gateway
PC0	192.168.0.2	255.255.255.0	192.168.0.1
PC1	192.168.0.3	255.255.255.0	192.168.0.1
PC2	192.168.0.4	255.255.255.0	192.168.0.1

- Now it's time to connect PC's from Wireless router. To do so click PC select Desktop click on PC Wireless

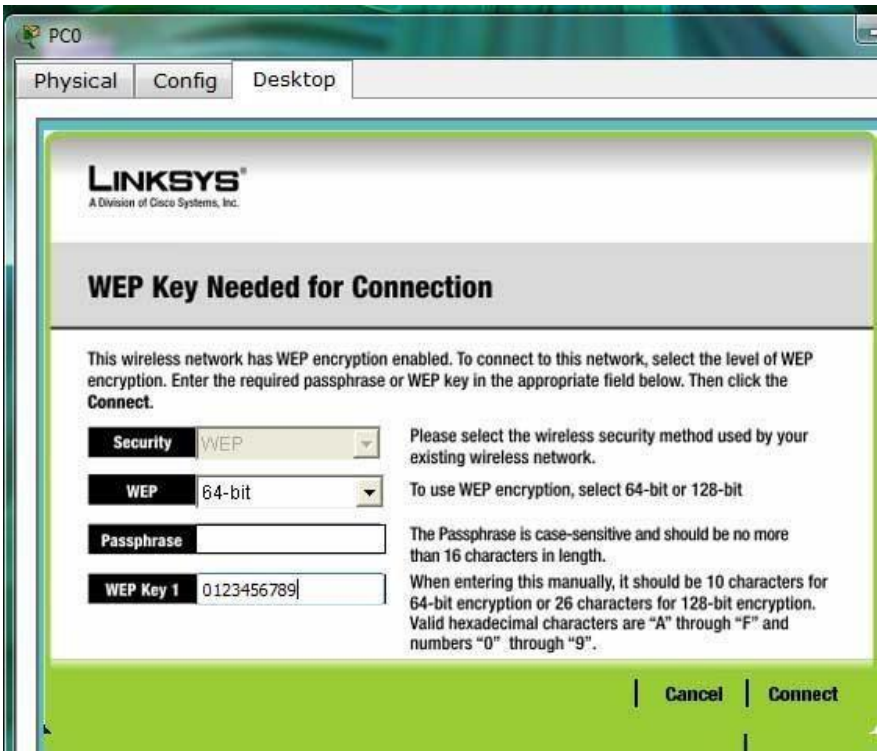


- Click on connect tab and click on Refresh button



As you can see in image that Wireless device is accessing MotherNetwork on CH 6 and signal strength is 100%. In left side you can see that WEP security is configured in network. Click on connect button to connect MotherNetwork

- It will ask for WAP key insert 0123456789 and click connect



It will connect you with wireless router.

As you can see in image below that system is connected. And PCI card is active.

- Repeat same process on PC1 and PC2.

Student observation:

c) **What is SSID of a wireless router?**
d) **What is a security key in wireless router?**
e) **Configure a simple Wireless LAN in your lab using a real access point and write down the configurations in your notebook.**

Practical-9

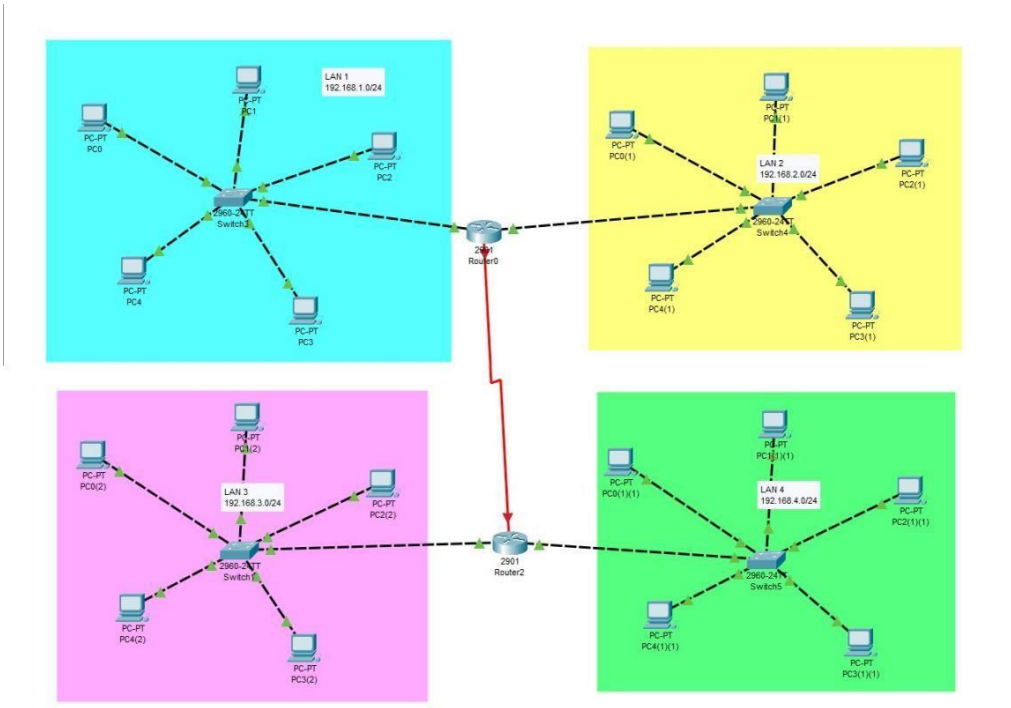
AIM:-Implementation of SUBNETTING in CISCO PACKET TRACER simulator.

Classless IP subnetting is a technique that allows for more efficient use of IP addresses by allowing for subnet masks that are not just the default masks for each IP class. This means that we can divide our IP address space into smaller subnets, which can be useful when we have a limited number of IP addresses but need to create multiple networks.

CREATING A NETWORK TOPOLOGY:
The first step in implementing classless IP subnetting is to create a network topology in Packet Tracer. To create a network topology in Packet Tracer, select the "New" button in the top left corner, then select "Network" and "Generic". This will create a blank network topology that we can use to add devices.

ADDING THE DEVICES:
Once we have created our network topology, we can add devices to it. Here, we will be adding routers, witches, and PCs. To add a device, select the device from the bottom left corner and drag it onto the network topology. Then, connect the devices by dragging a cable from one device's port to another device's port.

SUBNETTING:
To subnet the network address of 192.168.1.0/24 to provide enough space for at least 5 addresses for end devices, the switch, and the router, we can use a /27 subnet mask. This will give us 8 subnets with 30 host addresses each.



The IP addressing for the network shown in the topology can be as follows:

- Router R1:
- GigabitEthernet0/0: 192.168.1.1
- GigabitEthernet0/1: 192.168.2.1
- Switch S1:
- FastEthernet0/1: 192.168.1.0/27
- PC1: 192.168.1.11
- PC2: 192.168.1.12
- PC3: 192.168.1.13
- PC4: 192.168.1.14
- PC5: 192.168.1.15
- FastEthernet0/2: 192.168.2.0/27
- PC1: 192.168.2.11
- PC2: 192.168.2.12
- PC3: 192.168.2.13
- PC4: 192.168.2.14
- PC5: 192.168.2.15
- Router R2:
- FastEthernet0/0: 192.168.3.1
- FastEthernet0/1: 192.168.4.1
- Switch S2:
- FastEthernet0/1: 192.168.3.0/27
- PC1: 192.168.3.11
- PC2: 192.168.3.12
- PC3: 192.168.3.13
- PC4: 192.168.3.14
- PC5: 192.168.3.15
- FastEthernet0/2: 192.168.4.0/27
- PC1: 192.168.4.11
- PC2: 192.168.4.12
- PC3: 192.168.4.13
- PC4: 192.168.4.14
- PC5: 192.168.4.15

CONFIGURING THE DEVICES:

Now that we have added our devices and connected them, we can start configuring them. We will start by configuring the router. Right-click on the router and select "CLI". This will open the command-line interface (CLI) for the router. In the CLI, enter the following commands:

#enable
#configure terminal
#interface FastEthernet0/0
#ip address {IP address} {subnet mask}
#no shutdown
#exit

interface FastEthernet0/1
ip address {IP address} {subnet mask}

no shutdown
exit

Replace "{IP address}" and "{subnet mask}" with your desired IP address and subnet mask. The first interface, FastEthernet0/0, will be connected to the switch, while the second interface, FastEthernet0/1, will be connected to one of the PCs. These commands configure the router's interfaces with IP addresses and subnet masks.
Next, we will configure the switch. Right-click on the switch and select "CLI". In the CLI, enter the following commands:

```
enable
configure terminal
interface FastEthernet0/1
switchport mode access
exit

interface FastEthernet0/2
switchport mode access
exit
```

These commands configure the switch to operate in access mode on its two ports, which are connected to the two PCs.
Finally, we will configure the PCs. Right-click on each PC and select "Config". In the configuration window, enter the IP address, subnet mask, default gateway, and DNS server information. The IP address and subnet mask should be within the same subnet as the router's FastEthernet0/1 interface.

To configure the GigabitEthernet interface on the router, you can follow these steps:

1. Right-click on the router and select "CLI".
2. Enter the following commands:

```
enable
configure terminal
interface GigabitEthernet0/0
ip address {IP address} {subnet mask}
no shutdown
exit
```

Replace "{IP address}" and "{subnet mask}" with your desired IP address and subnet mask. These commands configure the GigabitEthernet interface with an IP address and subnet mask, and enable the interface.

TESTING THE NETWORK:

Now that our network topology is configured, we can test the network. Open a command prompt on each PC and try to ping the other PC. If the ping is successful, then the network is functioning properly. We can also use the "ping" command to test connectivity between the router and the PCs.

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num
	Successful	PC4(2)	Router2	ICMP		0.000	N	12
	Successful	PC4(2)	PC2(1)(1)	ICMP		0.000	N	13
	Successful	PC0	Router0	ICMP		0.000	N	14

Student observation:

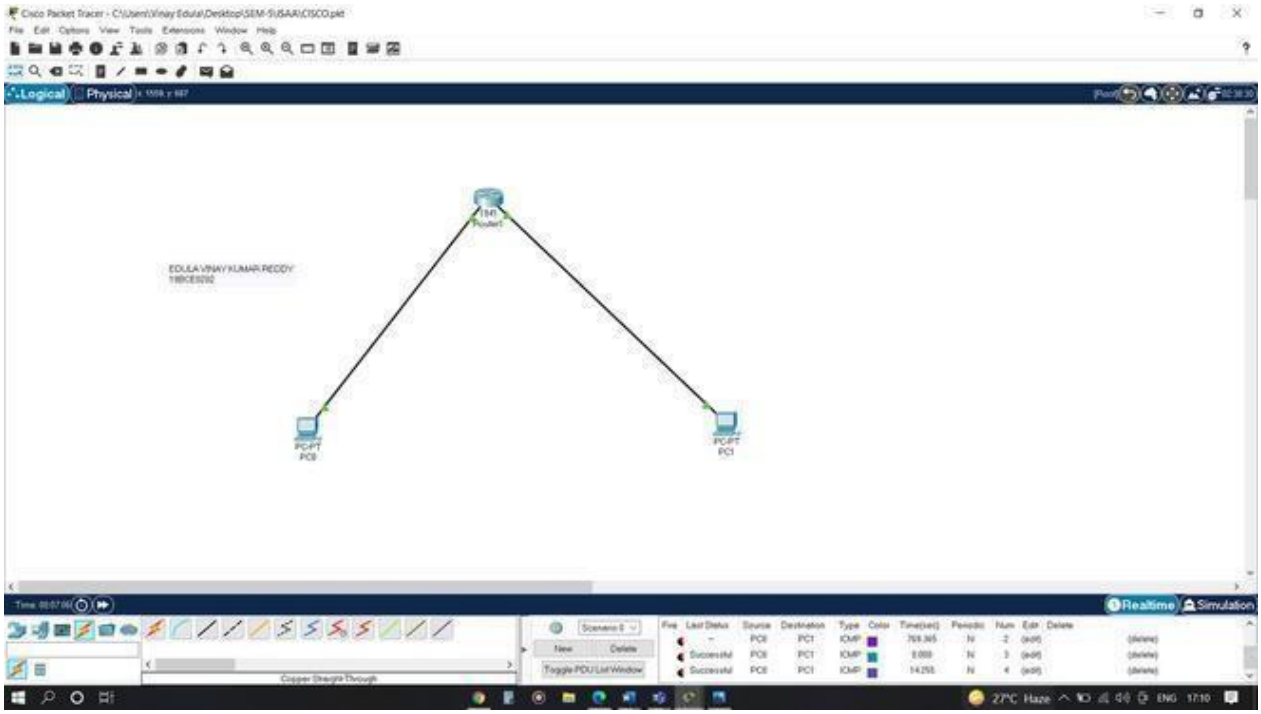
a) Write down your understanding of subnetting.
b) What is the advantage of implementing subnetting within a Network?
c) Find out whether subnetting is implemented in your college. If yes, draw and list down the subnets used with ip addresses.

Practical-10

AIM:-a) Internetworking with routers in CISCO PACKET TRACER simulator.

d) Design and configure a simple internetwork using a router.

In this network, a router and 2 PCs are used. Computers are connected with routers using a copper straight-through cable. After forming the network, to check network connectivity a simple PDU is transferred from PC0 to PC1.



Procedure:

Step-1(Configuring Router1):

1. Select the router and Open CLI.
2. Press ENTER to start configuring Router1.
3. Type enable to activate the privileged mode.

Router1 Command Line Interface:

Router>enable
Router#config t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface FastEthernet0/0
Router(config-if)#ip address 192.168.10.1 255.255.255.0
Router(config-if)#no shutdown
Router(config-if)#
%LINK-5-CHANGED: Interface GigabitEthernet0/0, changed state to up
Router(config-if)#interface FastEthernet0/1
Router(config-if)#ip address 192.168.20.1 255.255.255.0
Router(config-if)#no shutdown

Step-2(Configuring PCs):

1. Assign IP Addresses to every PC in the network.
2. Select the PC, Go to the desktop and select IP Configuration and assign an IP address, Default gateway, Subnet Mask
3. Assign the default gateway of PC0 as 192.168.10.1.
4. Assign the default gateway of PC1 as 192.168.20.1.

Step-3(Connecting PCs with Router):

1. Connect FastEthernet0 port of PC0 with FastEthernet0/0 port of Router1 using a copper straight-through cable.
2. Connect FastEthernet0 port of PC1 with FastEthernet0/1 port of Router1 using a copper straight-through cable.

Router Configuration Table:

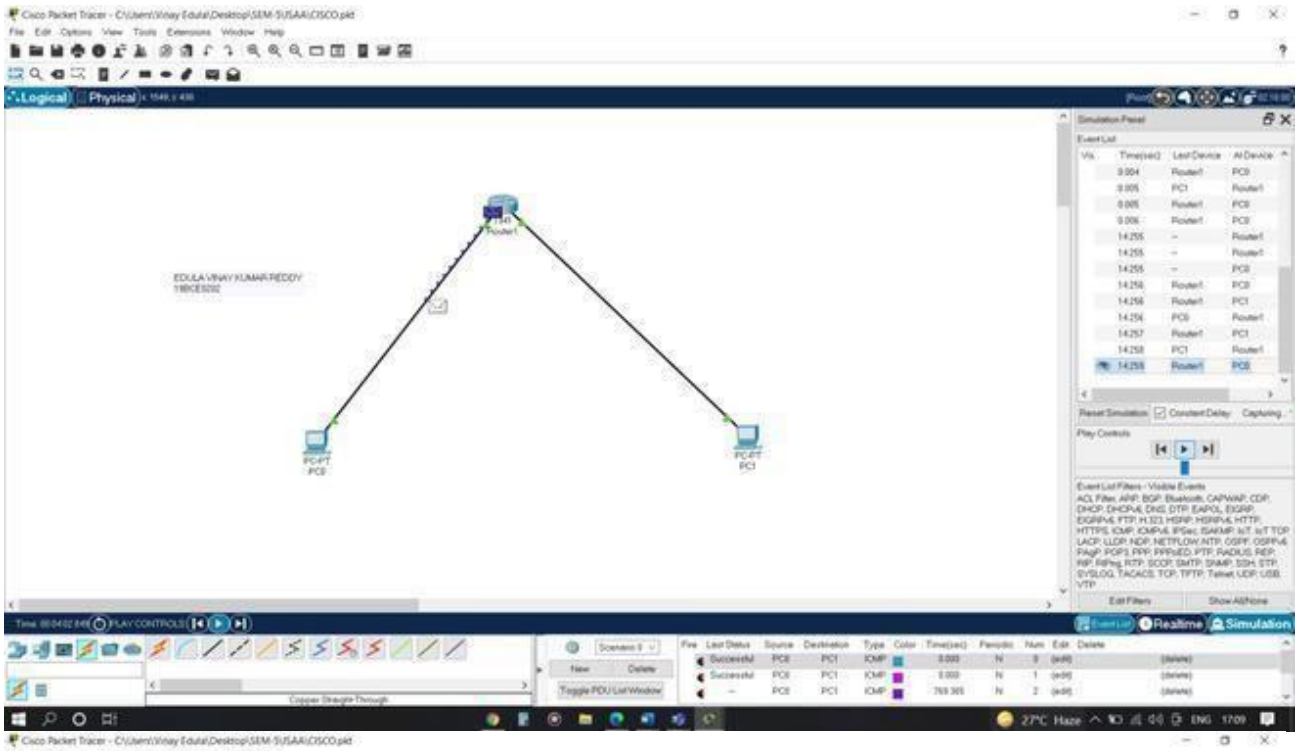
Device Name	IP address FastEthernet0 /0	Subnet Mask	IP Address FastEthernet0/1	Subnet Mask
Router1	192.168.10.1	255.255.255.0	192.168.20.1	255.255.255.0

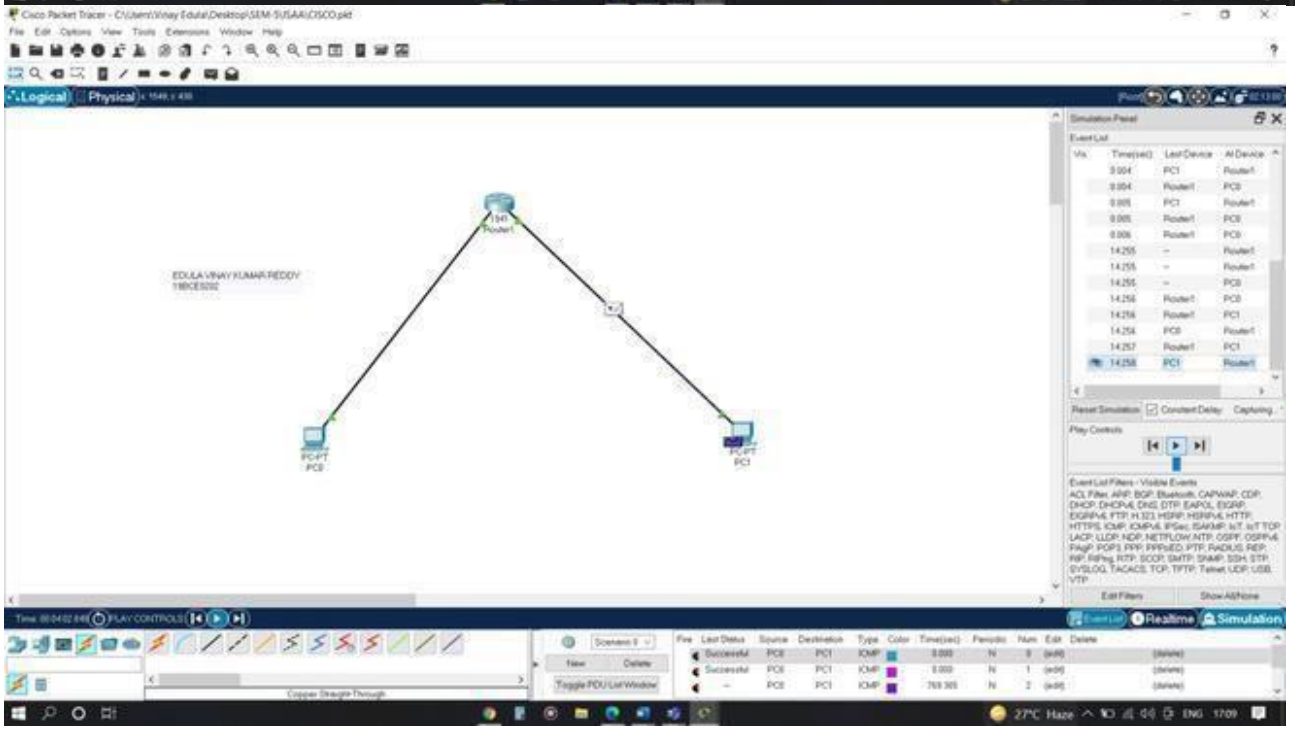
PC Configuration Table:

Device Name	IP address	Subnet Mask	Gateway
PC 0	192.168.10.2	255.255.255.0	192.168.10.1
PC 1	192.168.20.2	255.255.255.0	192.168.20.1

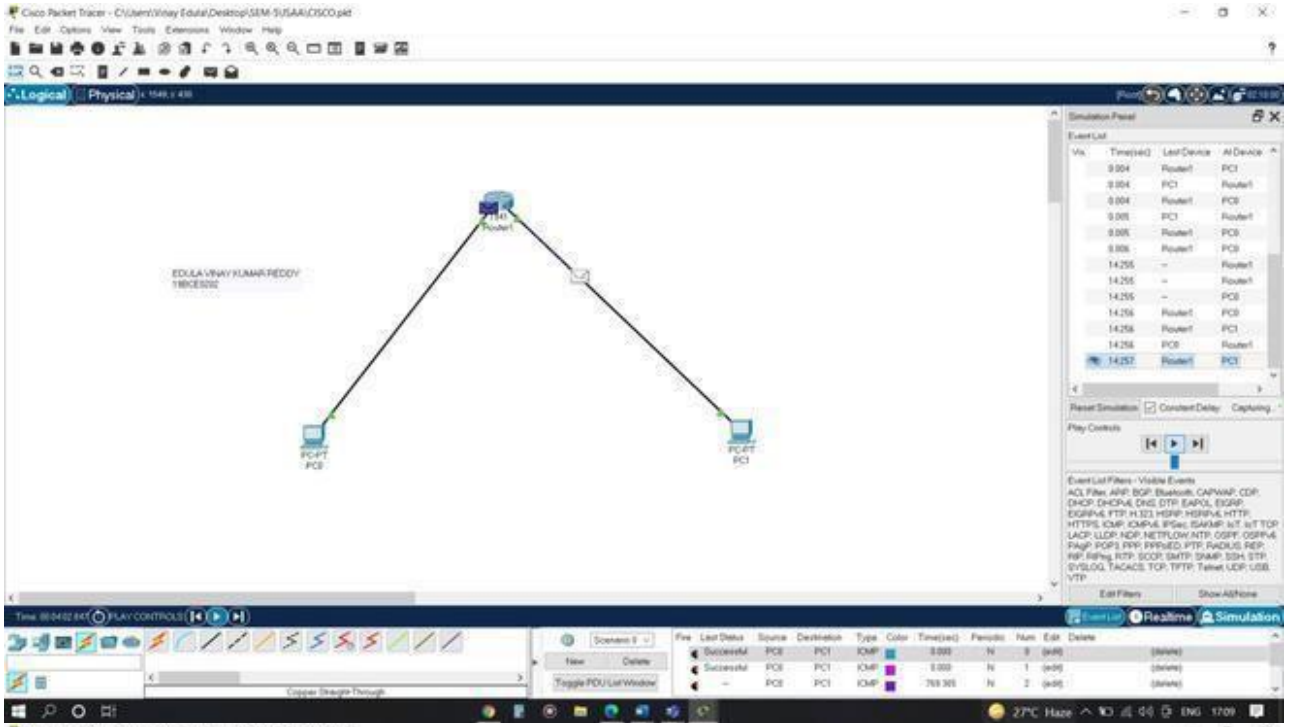
Designed Network topology:
Simulation of Designed Network Topology:

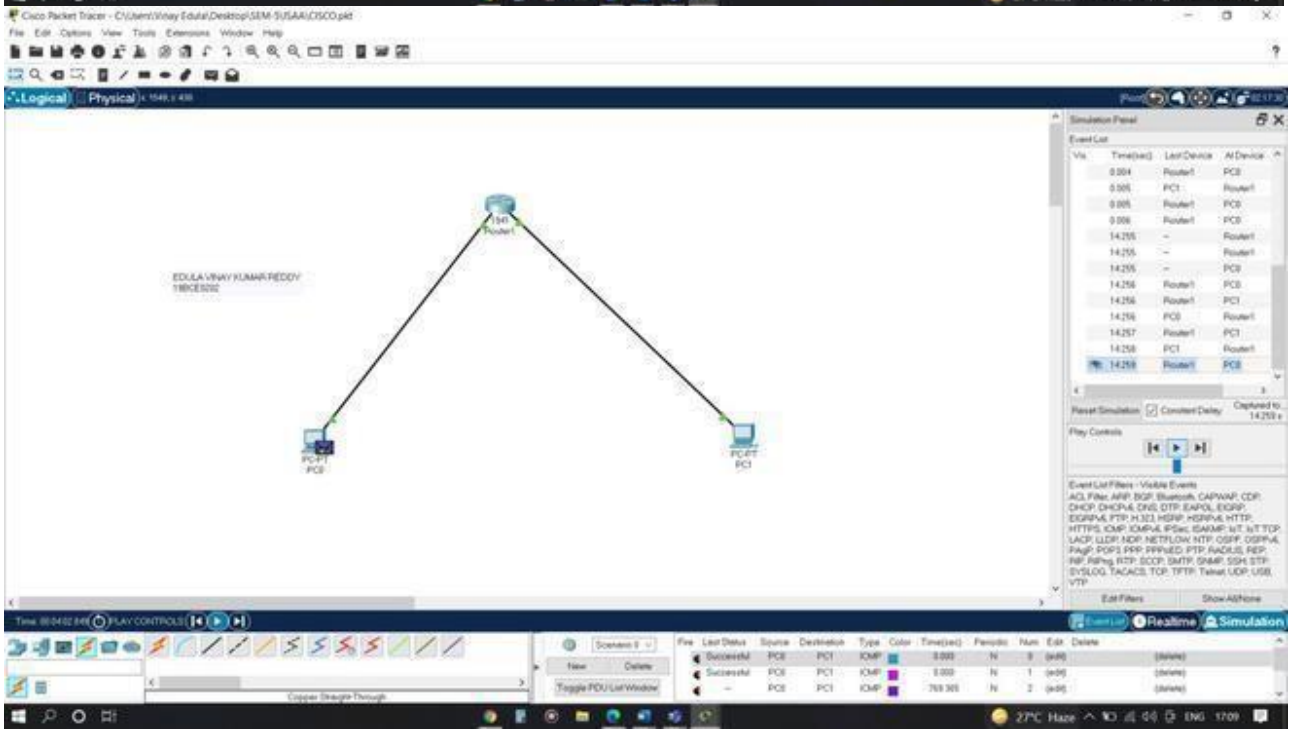
Sending a PDU From PC0 to PC1:





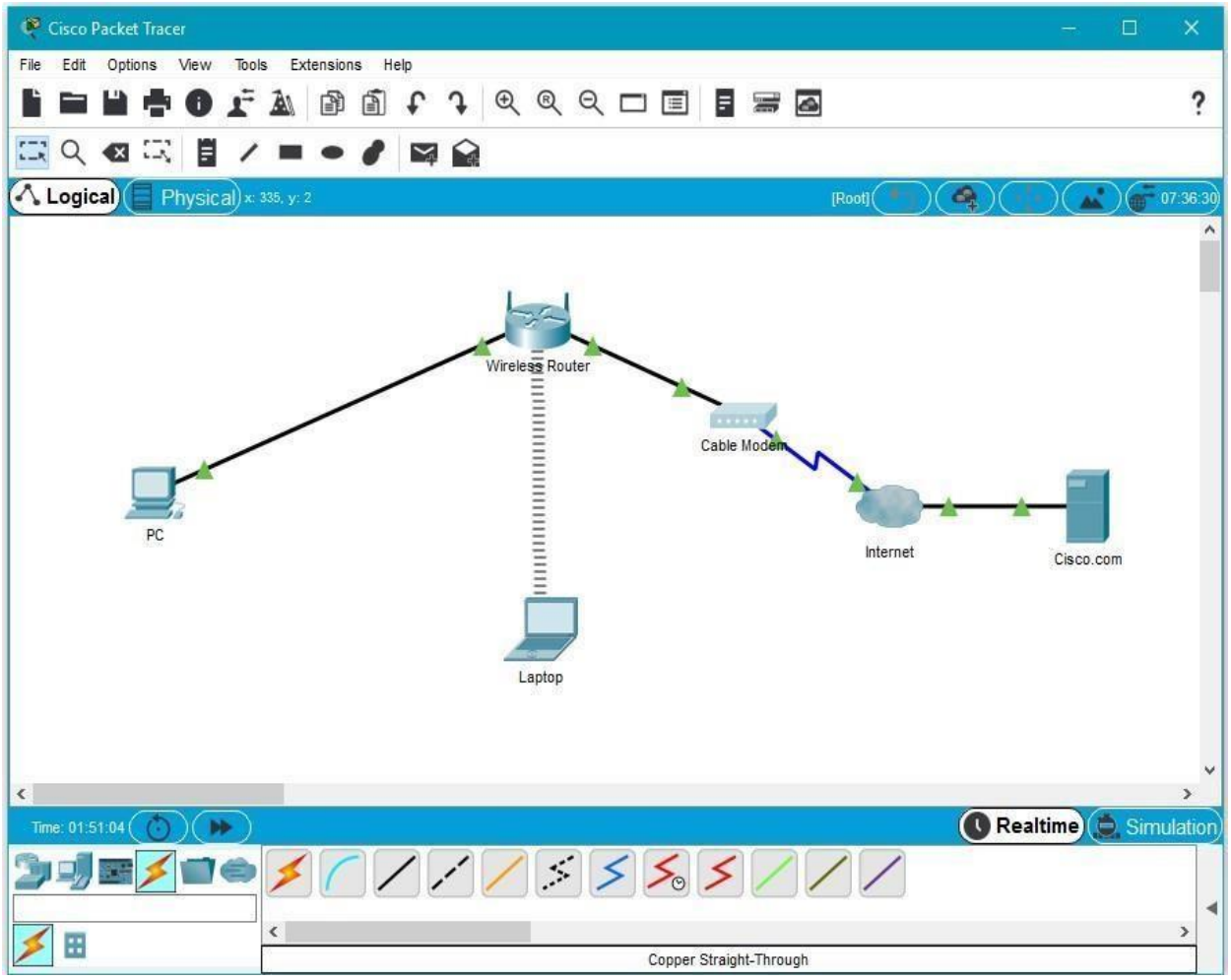
Acknowledgment From PC1 to PC0:





Practical 10

AIM:- b) Design and configure an internetwork using wireless router, DHCP server and internet cloud.



Addressing Table

Device	Interface	IP Address	Subnet Mask	Default Gateway
PC	Ethernet0	DHCP		192.168.0.1
Wireless Router	LAN	192.168.0.1	255.255.255.0	
Wireless Router	Internet	DHCP		
Cisco.com Server	Ethernet0	208.67.220.220	255.255.255.0	
Laptop	Wireless0	DHCP		

Objectives

Part 1: Build a Simple Network in the Logical Topology Workspace

Part 2: Configure the Network Devices
Part 3: Test Connectivity between Network Devices Part 4: Save the File and Close Packet Tracer

Part 1: Build a Simple Network in the Logical Topology Workspace

Step 1: Launch Packet Tracer.

Step 2: Build the topology

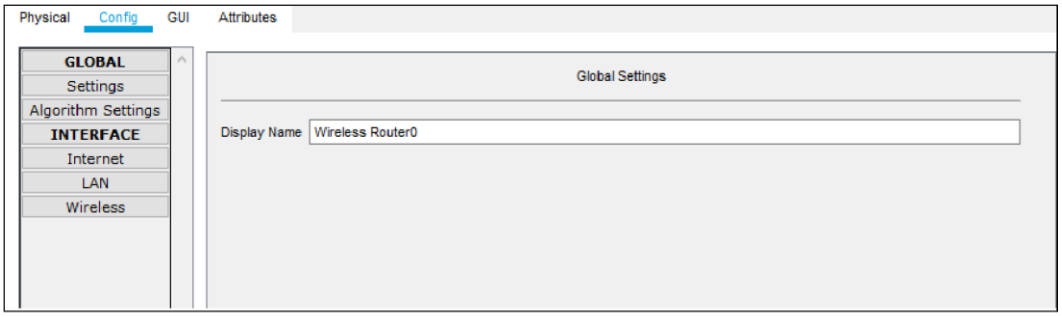
a. Add network devices to the workspace.

Using the device selection box, add the network devices to the workspace as shown in the topology diagram.

To place a device onto the workspace, first choose a device type from the **Device-Type Selection** box. Then, click on the desired device model from the **Device-Specific Selection** box. Finally, click on a location in the workspace to put your device in that location. If you want to cancel your selection, click the **Cancel** icon for that device. Alternatively, you can click and drag a device from the **Device-Specific Selection** box onto the workspace.

b. Change display names of the network devices.

To change the display names of the network devices click on the device icon on the Packet Tracer **Logical** workspace, then click on the **Config** tab in the device configuration window. Type the new name of the device into the **Display Name** box as show in the figure below.



c. Add the physical cabling between devices on the workspace

Using the device selection box, add the physical cabling between devices on the workspace as shown in the topology diagram.

The PC will need a copper straight-through cable to connect to the wireless router. Select the copper straight-through cable in the device selection box and attach it to the FastEthernet0 interface of the PC and the Ethernet 1 interface of the wireless router.

The wireless router will need a copper straight-through cable to connect to the cable modem. Select the copper straight-through cable in the device-selection box and attach it to the Internet interface of the wireless router and the Port 1 interface of the cable modem.

The cable modem will need a coaxial cable to connect to the Internet cloud. Select the coaxial cable in the device-selection box and attach it to the Port 0 interface of the cable modem and the coaxial interface of the Internet cloud.

The Interne cloud will need copper straight-through cable to connect to the Cisco.com server. Select the copper straight-through cable in the device-selection box and attach it to the Ethernet interface of the Internet cloud and the FastEthernet0 interface of the Cisco.com server.

Part 2: Configure the Network Devices

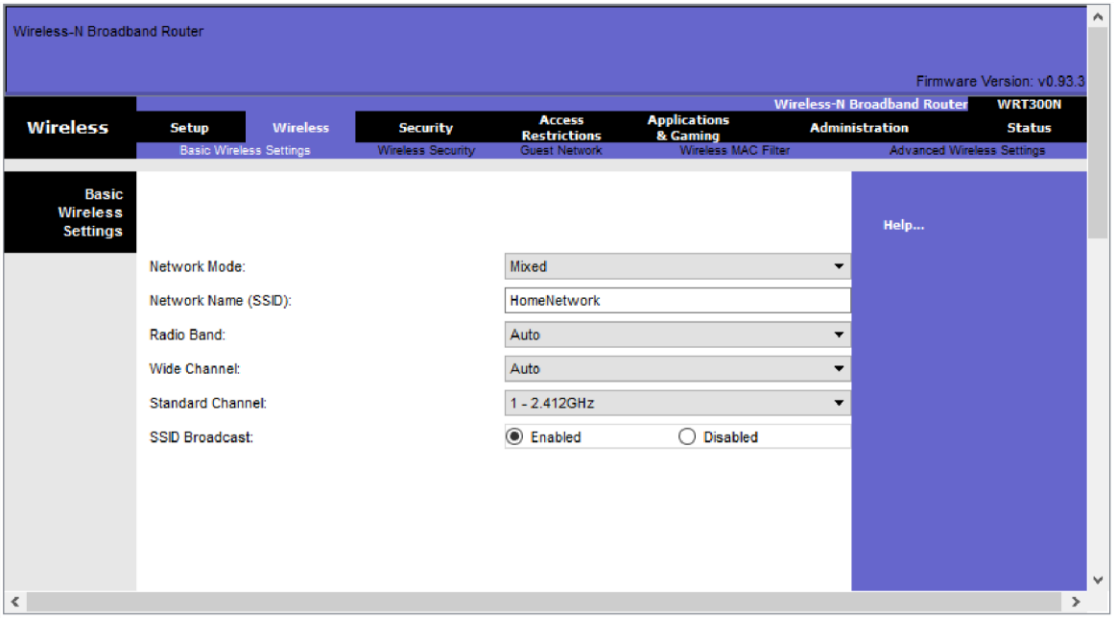
Step 1: Configure the wireless router

a. Create the wireless network on the wireless router

Click on the **Wireless Router** icon on the Packet Tracer **Logical** workspace to open the device configuration window.

In the wireless router configuration window, click on the **GUI** tab to view configuration options for the wireless router.

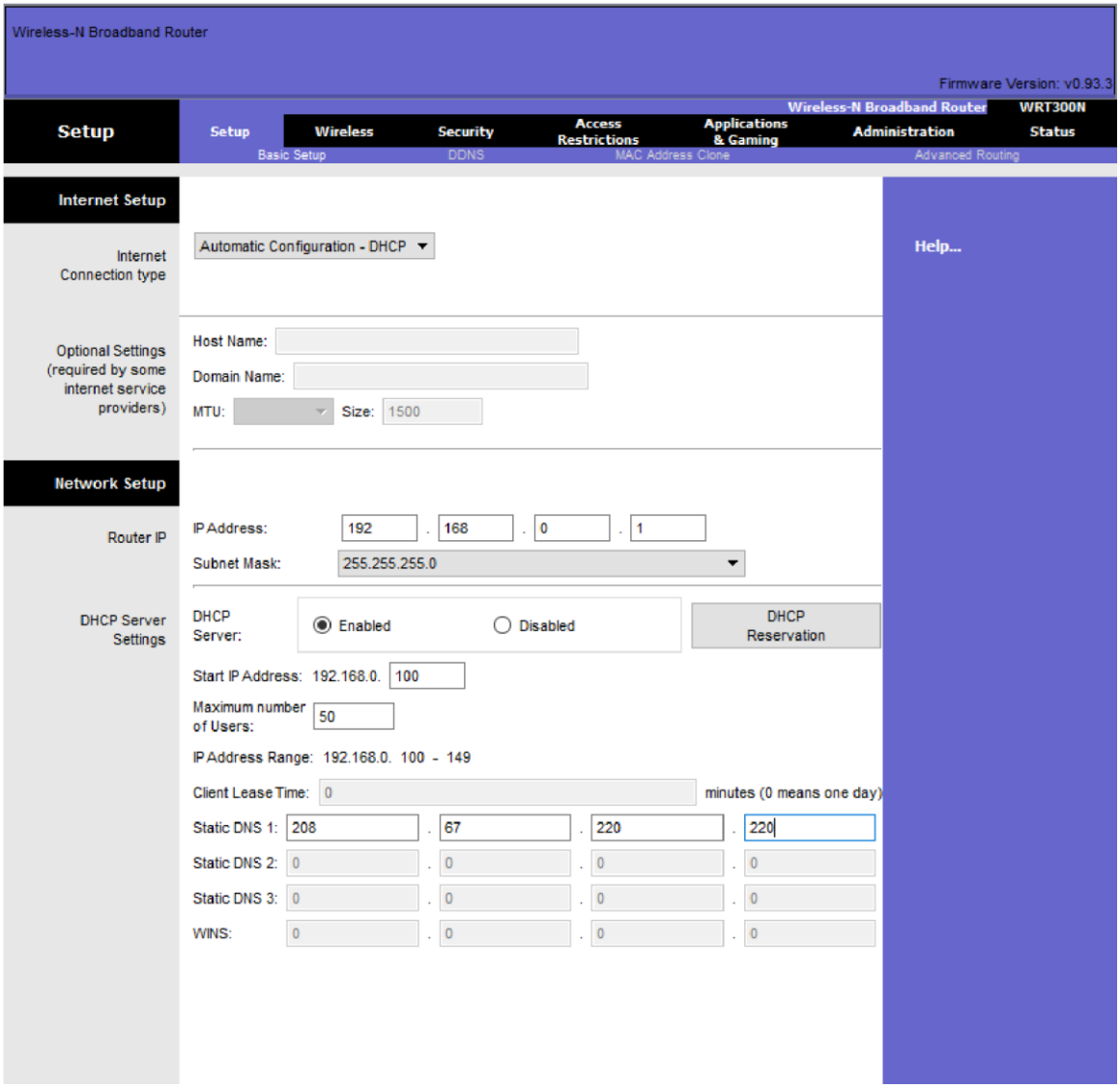
Next, click on the **Wireless** tab in the GUI to view the wireless settings. The only setting that needs to be changed from the defaults is the **Network Name (SSID)**. Here, type the name "HomeNetwork" as shown in the figure.



Configure the Internet connection on the wireless router
Click on the **Setup** tab in the wireless router GUI.

In the **DHCP Server** settings verify that the **Enabled** button is selected and configure the static IP address of the DNS server as 208.67.220.220 as shown in the figure.

b. Click on the **Save Settings** tab.



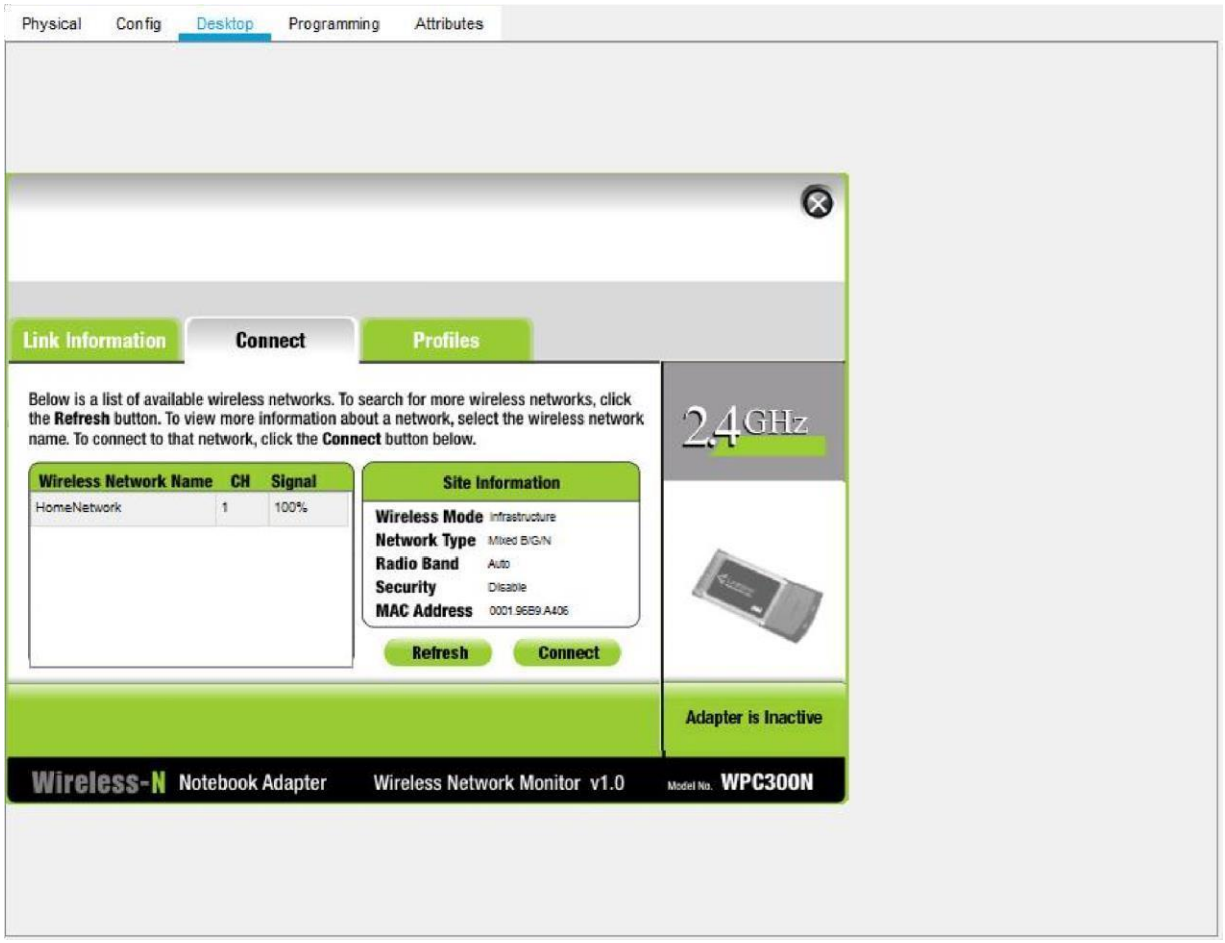
Step 2: Configure the laptop

a. Configure the Laptop to access the wireless network
Click on the Laptop icon on the Packet Tracer **Logical** workspace and in the laptop configuration windows select the **Physical** tab.
In the **Physical** tab you will need to remove the Ethernet copper module and replace it with the Wireless WPC300N module.
To do this, you first power the Laptop off by clicking the power button on the side of the laptop. Then remove the currently installed Ethernet copper module by clicking on the module on the side of the laptop and dragging it to the **MODULES** pane on the left of the laptop window. Then install the Wireless WPC300N module by clicking on it in the **MODULES** pane and dragging it to the empty module port on the side of the laptop. Power the laptop back on by clicking on the Laptop power button again.

With the wireless module installed, the next task is to connect the laptop to the wireless network.

Click on the **Desktop** tab at the top of the Laptop configuration window and select the **PC Wireless** icon.

Once the Wireless-N Notebook Adapter settings are visible, select the **Connect** tab. The wireless network "HomeNetwork" should be visible in the list of wireless networks as shown in the figure.
Select the network, and click on the **Connect** tab found below the **Site Information pane.**

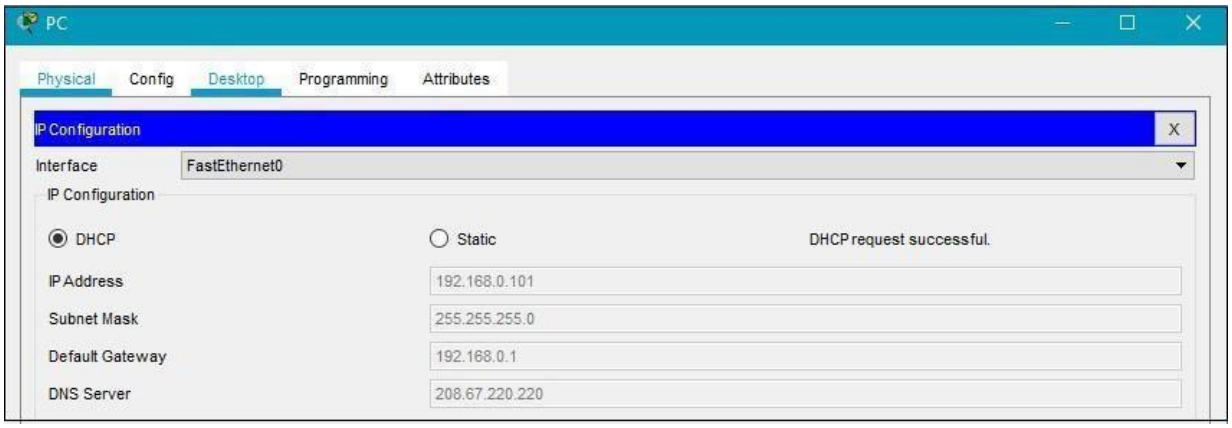


Step 3: Configure the PC

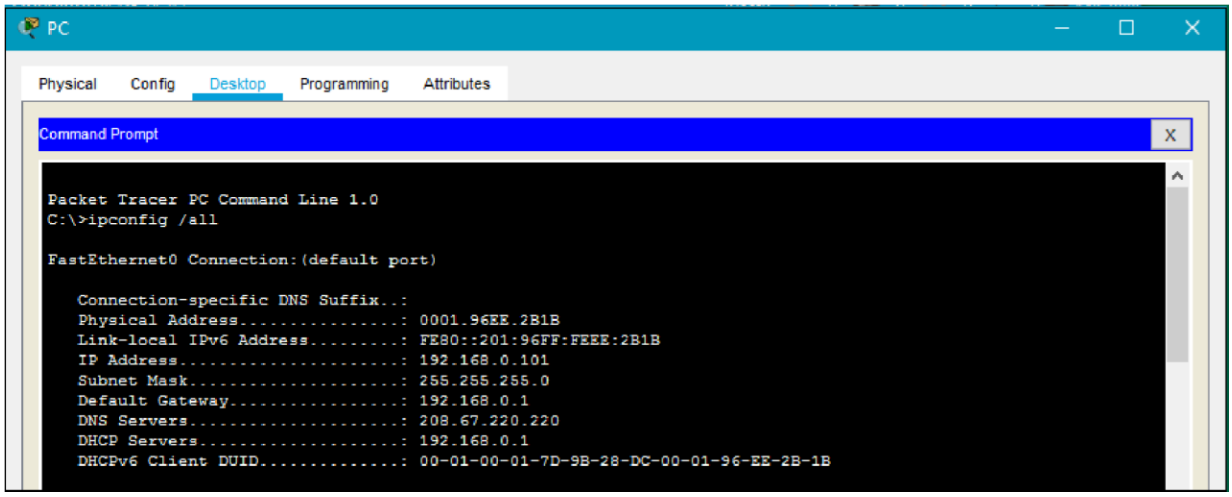
a. Configure the PC for the wired network

Click on the **PC** icon on the Packet Tracer **Logical** workspace and select the **Desktop** tab and then the **IP Configuration** icon.

In the IP Configuration window, select the **DCHP** radio button as shown in the figure so that the PC will use DCHP to receive an IPv4 address from the wireless router. Close the IP Configuration window.



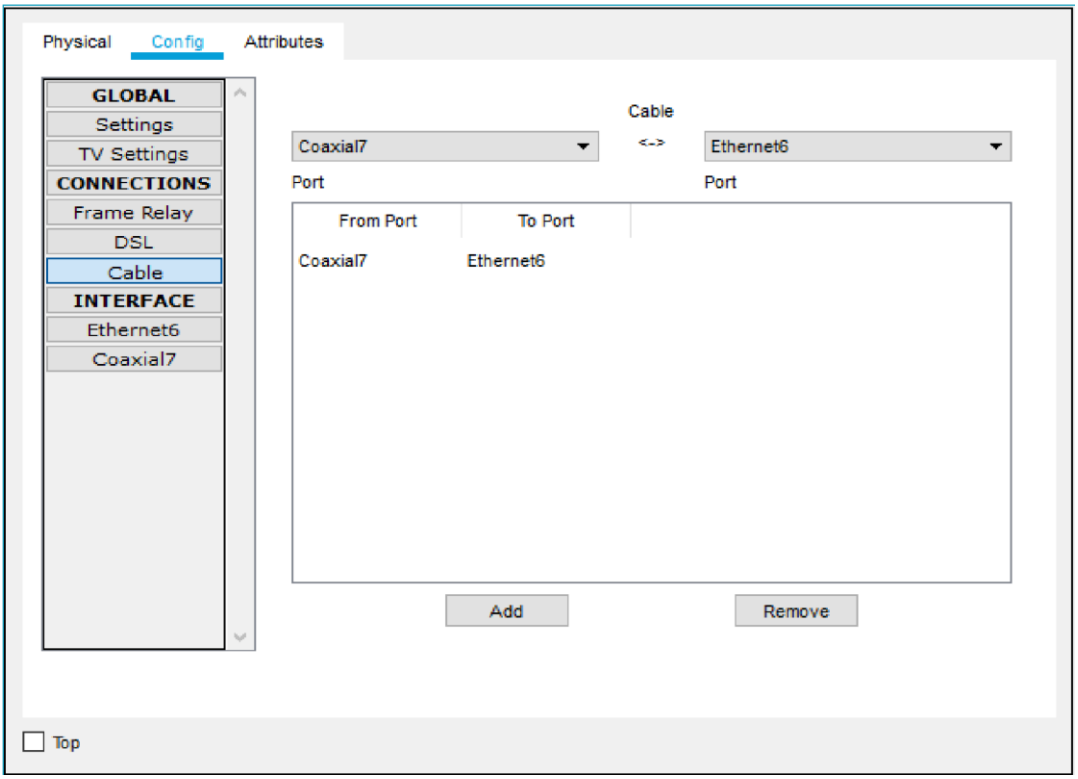
Click on the Command Prompt icon. Verify that the PC has received an IPv4 address by issuing the **ipconfig /all** command from the command prompt as shown in the figure. The PC should receive an IPv4 address in the 192.168.0.x range.



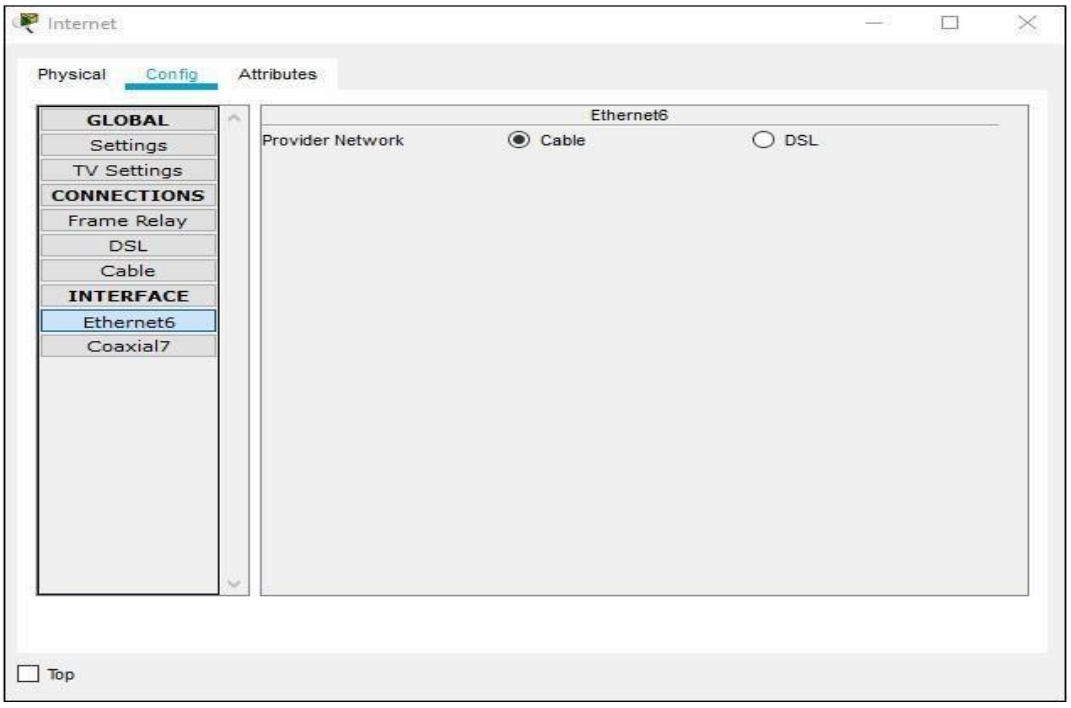
Step 4: Configure the Internet cloud

a. Install network modules if necessary
Click on the **Internet Cloud** icon on the Packet Tracer **Logical** workspace and then click on the **Physical** tab. The cloud device will need two modules if they are not already installed. The PT-CLOUD-NM-1CX which is for the cable modem service connection and the PT-CLOUD-NM-1CFE which is for a copper Ethernet cable connection. If these modules are missing, power off the physical cloud devices by clicking on the power button and drag each module to an empty module port on the device and then power the device back on.

b. Identify the From and To Ports
Click on the **Config** tab in the Cloud device window. In the left pane click on **Cable** under **CONNECTIONS**. In the first drop down box choose Coaxial and in the second drop down box choose
Ethernet then click the **Add** button to add these as the **From Port** and **To Port** as shown in the figure.



c. Identify the type of provider
While still in the **Config** tab click Ethernet under **INTERFACE** in the left pane. In the Ethernet configuration window select **Cable** as the Provider Network as shown in the figure.



Step 5: Configure the Cisco.com server

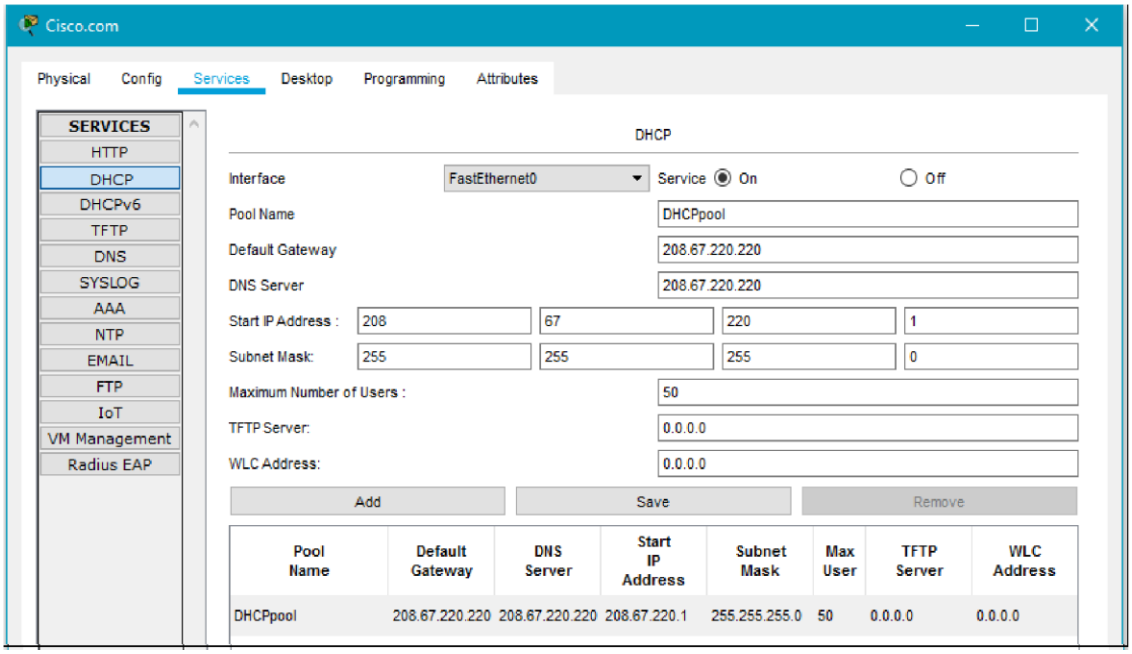
a. Configure the Cisco.com server as a DHCP server

Click on the Cisco.com server icon on the Packet Tracer **Logical** workspace and select the **Services** tab. Select **DHCP** from the **SERVICES** list in the left pane.

In the DHCP configuration window, configure a DHCP as shown in the figure with the following settings.

- Click **On** to turn the DCHP service on
- Pool name: DHCPpool
- Default Gateway: 208.67.220.220
- DNS Server: 208.67.220.220
- Starting IP Address: 208.67.220.1
- Subnet Mask 255.255.255.0
- Maximum number of Users: 50

Click **Add** to add the pool



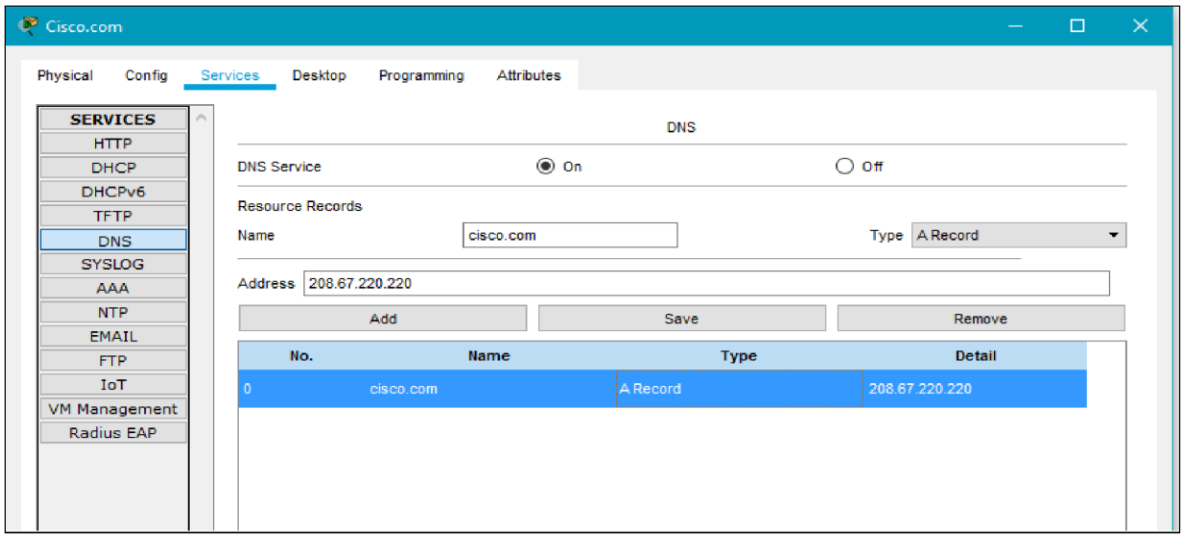
b. Configure the Cisco.com server as a DNS server to provide domain name to IPv4 address resolution.

While still in the **Services** tab, select **DNS** from the **SERVICES** listed in the left pane.

Configure the DNS service using the following settings as shown in the figure.

- Click **On** to turn the DNS service on
- Name: Cisco.com
- Type: A Record
- Address: 208.67.220.220

Click **Add** to add the DNS service settings



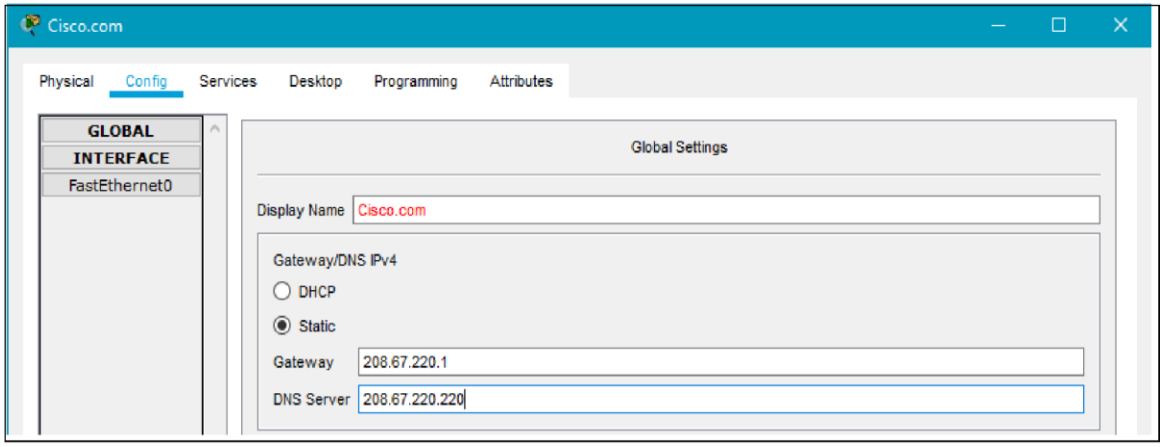
c. Configure the Cisco.com server Global settings.

Select the **Config** tab.

Click on **Settings** in left pane.

Configure the Global settings of the server as follows:

- Select **Static**
- Gateway: 208.67.220.1
- DNS Server: 208.67.220.220

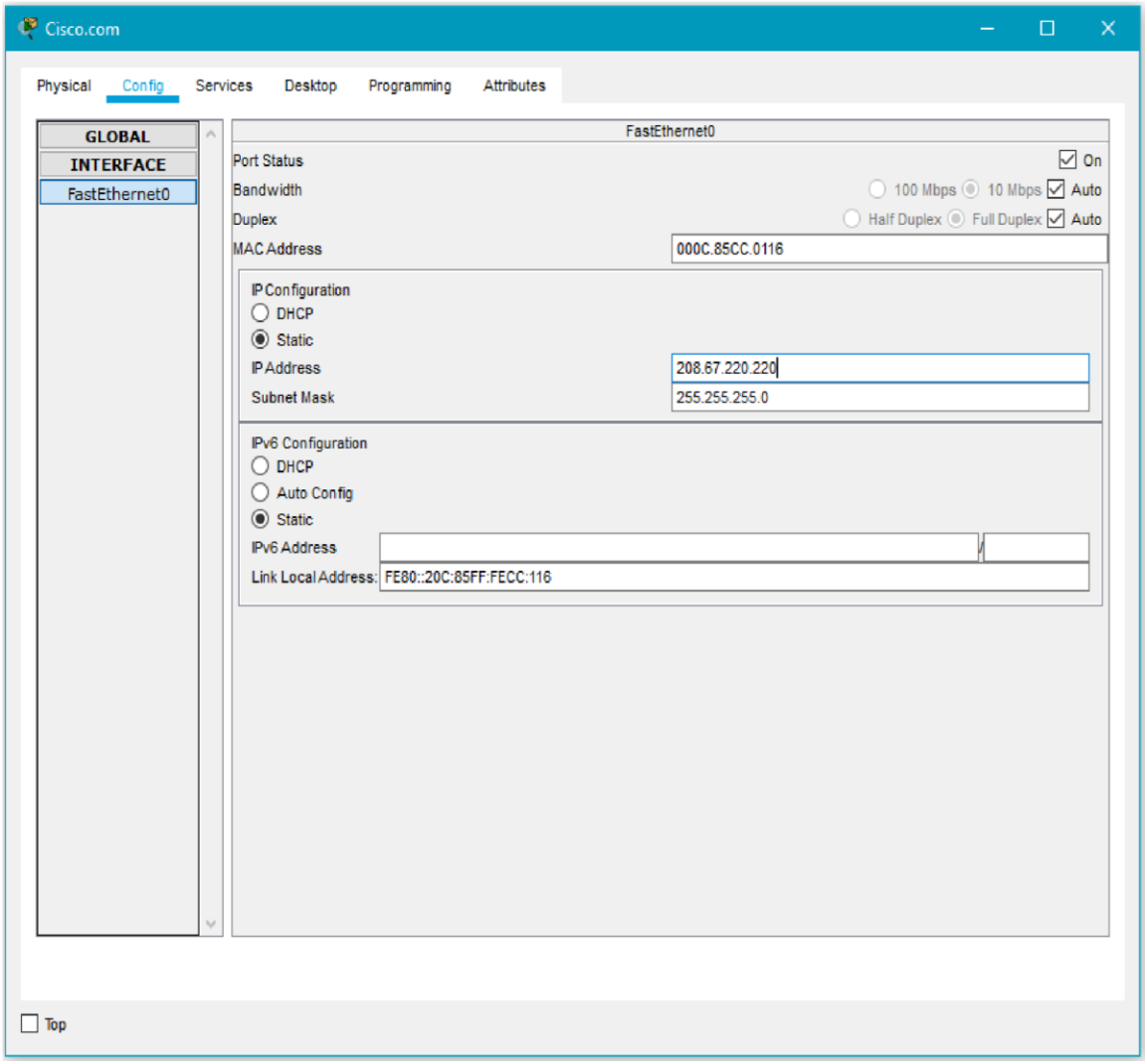


d. Configure the Cisco.com server FastEthernet0 Interface settings.

Click on **Fast Ethernet** in left pane of the **Config** tab

Configure the Fast Ethernet Interface settings of the server as follows:

- Select **Static** under IP Configuration
- IP Address: 208.67.220.220
- Subnet Mask: 255.255.255.0



Part 3: Verify Connectivity

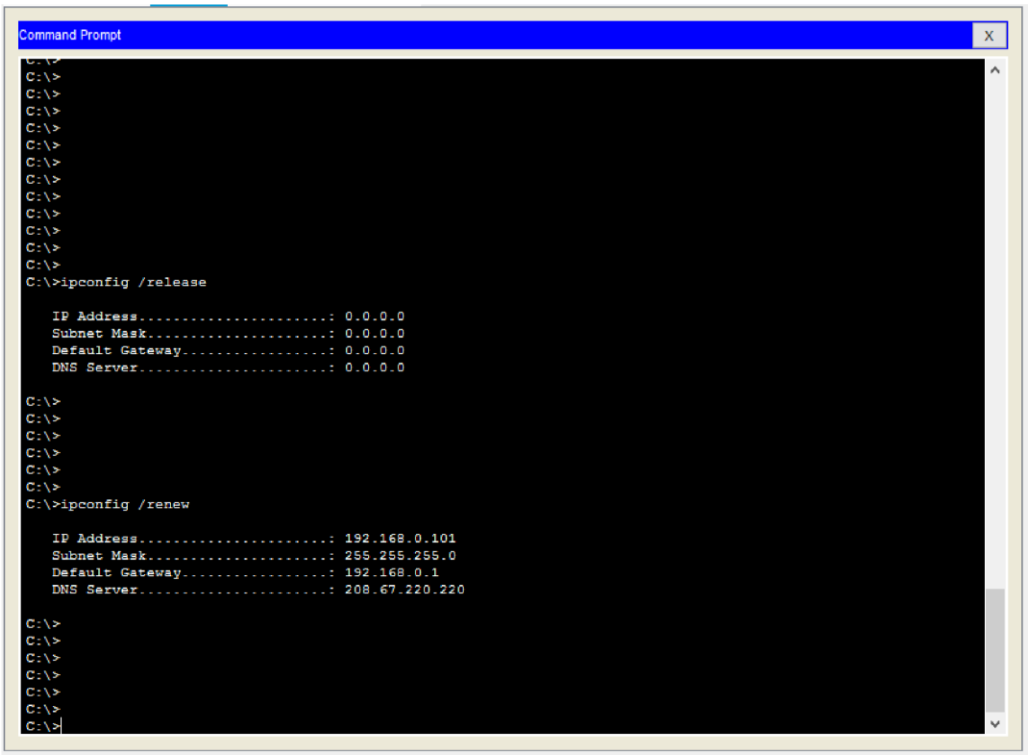
Step 1: Refresh the IPv4 settings on the PC

a) Verify that the PC is receiving IPv4 configuration information from DHCP.

Click on the **PC** on the Packet Tracer **Logical** workspace and then the select the **Desktop** tab of the PC configuration window.

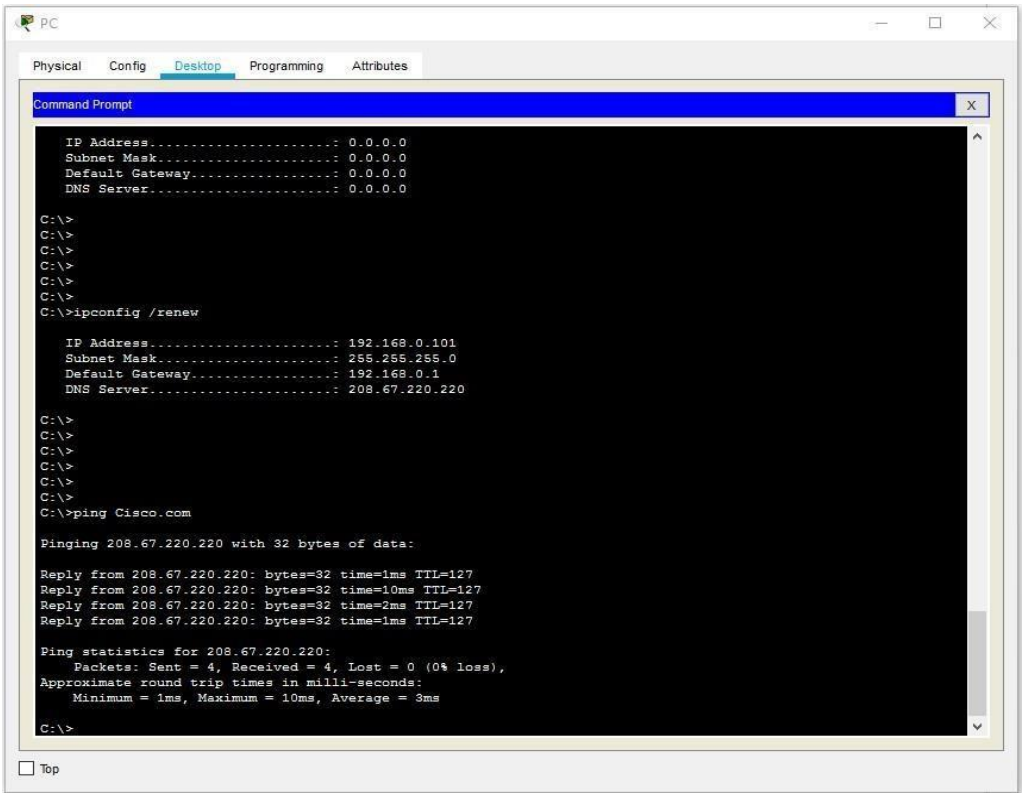
Click on the **Command Prompt** icon

In the command prompt refresh the IP settings by issuing the commands **ipconfig /release** and then **ipconfig /renew.** The output should show that the PC has an IP address in the 192.168.0.x range, a subnet mask, a default gateway, and DNS server address as shown in the figure.



b) Test connectivity to the Cisco.com server from the PC

From the command prompt, issue the command **ping Cisco.com.** It may take a few seconds for the ping to return. Four replies should be received as shown in the figure.



Student observation:

1. **Write down the key features of configuring Wireless router and DHCP server.**
2. **What is the significance of DHCP sever in internetworking.**
3. **Design and configure an inter-network in your lab using switch, router and Ethernet cables. Draw and label the design in your notebook. Also, show the ip address configuration of each and every device.**

Practical 11

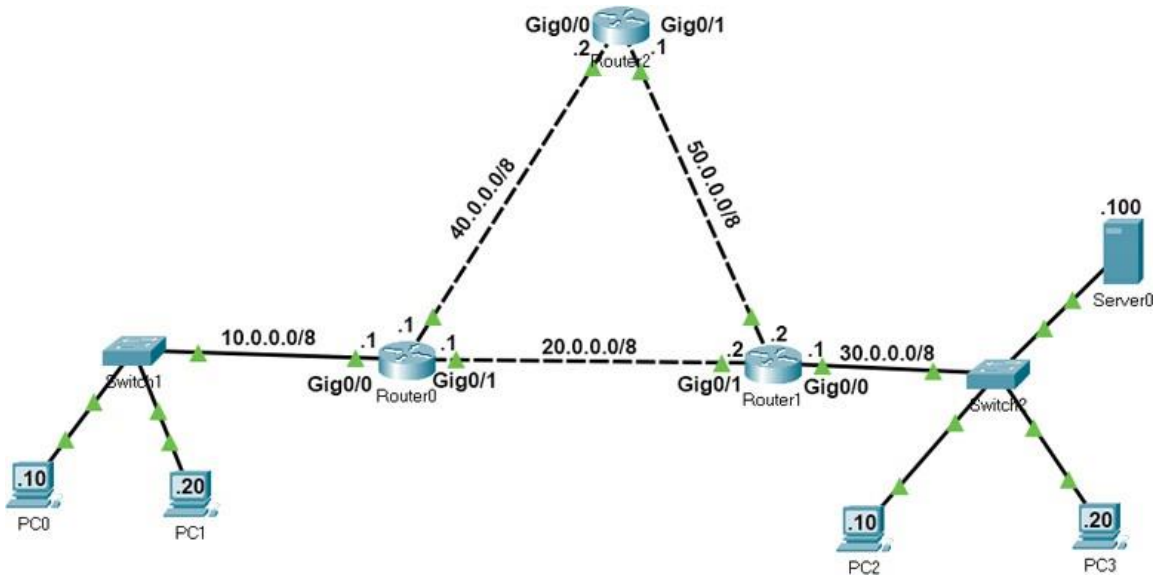
AIM:- a)Simulate Static Routing Configuration using CISCO Packet Tracer

Static routes are the routes you manually add to the router's routing table. The process of adding static routes to the routing table is known as static routing. Let's take a packet tracer example to understand how to use static routing to create and add a static route to the routing table.

Setting up a practice lab

Create a packet tracer lab as shown in the following image or download the following pre-created lab and load it on Packet Tracer.

Packet Tracer Lab with Initial IP Configuration



In this lab, each network has two routes to reach. We will configure one route as the main route and another route as the backup route. If the link bandwidth of all routes is the same, we use the route that has the least number of routers as the main route. If the link bandwidth and the number of routers are the same, we can use any route as the main route and another route as the backup route.

If we specify two routes for the same destination, the router automatically selects the best route for the destination and adds the route to the routing table. If you manually want to select a route that the router should add to the routing table, you have to set the AD value of the route lower than other routes. For example, if you use the following commands to create two static routes for network 30.0.0/8, the route will place the first route to the routing table.

```
#ip route 30.0.0.0 255.0.0.0 20.0.0.2 10
#ip route 30.0.0.0 255.0.0.0 40.0.0.2 20
```

If the first route fails, the router automatically adds the second route to the routing table.

Creating, adding, verifying static routes

Routers automatically learn their connected networks. We only need to add routes for the networks that are not available on the router's interfaces. For example, network 10.0.0.0/8, 20.0.0.0/8 and 40.0.0.0/8 are directly connected to Router0. Thus, we don't need to configure routes for these

networks. Network 30.0.0.0/8 and network 50.0.0.0/8 are not available on Router0. We have to create and add routes only for these networks.

The following table lists the connected networks of each router.

Router	Available networks on local interfaces	Networks available on other routers' interfaces
Router0	10.0.0.0/8, 20.0.0.0/8, 40.0.0.0/8	30.0.0.0/8, 50.0.0.0/8
Router1	20.0.0.0/8, 30.0.0.0/8, 50.0.0.0/8	10.0.0.0/8, 40.0.0.0/8
Router2	40.0.0.0/8, 50.0.0.0/8	10.0.0.0/8, 20.0.0.0/8, 30.0.0.0/8

Let's create static routes on each router for networks that are not available on the router.

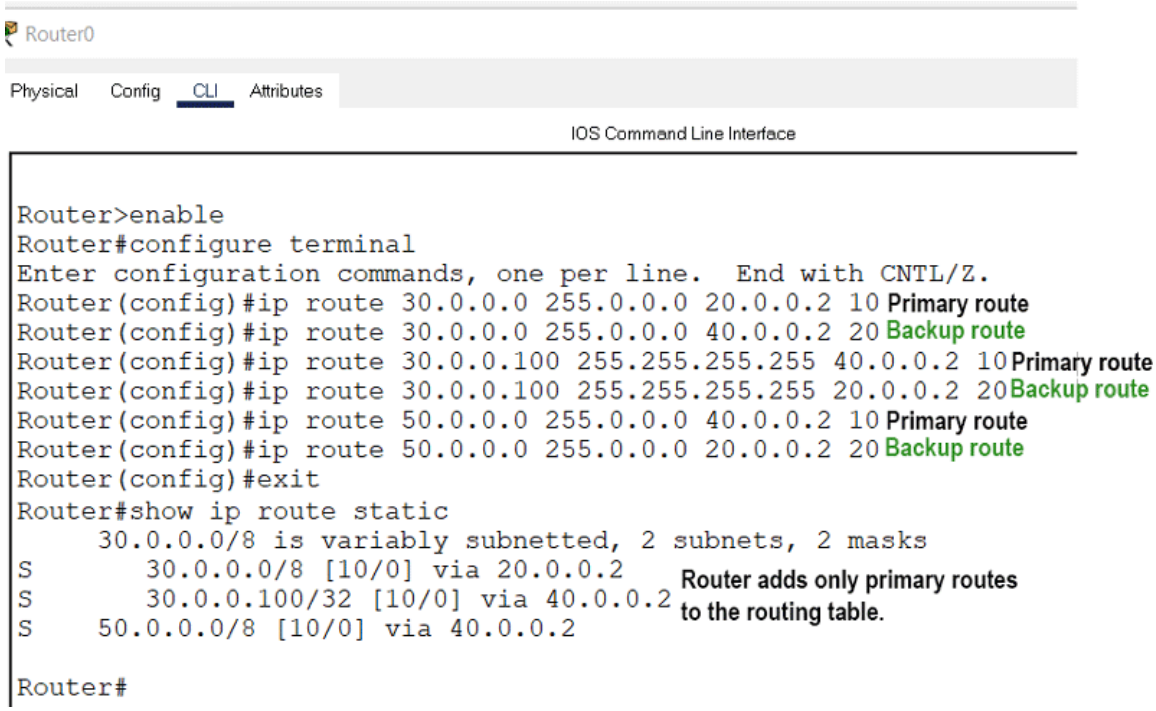
Router0 requirements

- Create two routes for network 30.0.0.0/8 and configure the first route (via -Router1) as the main route and the second route (via-Router2) as a backup route.
- Create two routes for the host 30.0.0.100/8 and configure the first route (via -Router2) as the main route and the second route (via-Router1) as a backup route.
- Create two routes for network 50.0.0.0/8 and configure the first route (via -Router2) as the main route and the second route (via-Router1) as a backup route.
- Verify the router adds only main routes to the routing table.

Router0 configuration

Access the CLI prompt of Router0 and run the following commands.

```
Router>enable
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#ip route 30.0.0.0 255.0.0.0 20.0.0.2 10
Router(config)#ip route 30.0.0.0 255.0.0.0 40.0.0.2 20
Router(config)#ip route 30.0.0.100 255.255.255.255 40.0.0.2 10
Router(config)#ip route 30.0.0.100 255.255.255.255 20.0.0.2 20
Router(config)#ip route 50.0.0.0 255.0.0.0 40.0.0.2 10
Router(config)#ip route 50.0.0.0 255.0.0.0 20.0.0.2 20
Router(config)#exit
Router#show ip route static
30.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
S 30.0.0.0/8 [10/0] via 20.0.0.2
S 30.0.0.100/32 [10/0] via 40.0.0.2
S 50.0.0.0/8 [10/0] via 40.0.0.2
Router#
```

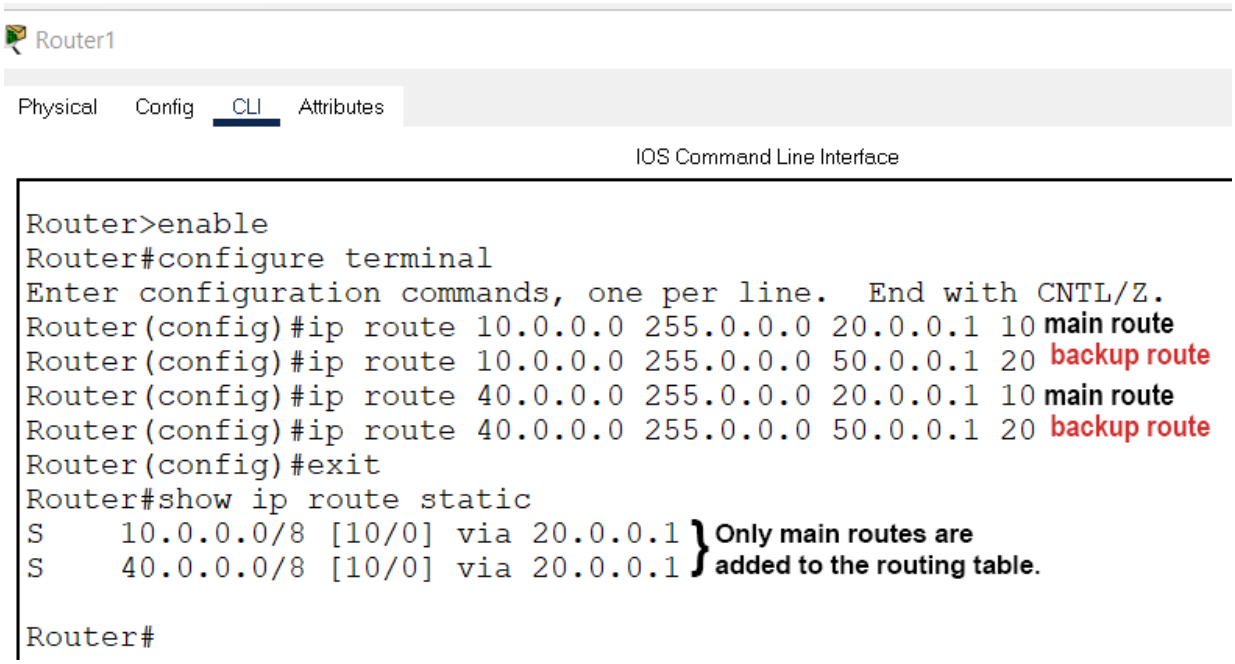


Router1 requirements

- Create two routes for network 10.0.0.0/8 and configure the first route (via -Router0) as the main route and the second route (via-Router1) as a backup route.
- Create two routes for network 40.0.0.0/8 and configure the first route (via -Router0) as the main route and the second route (via-Router2) as a backup route.
- Verify the router adds only main routes to the routing table.

Router1 configuration

```
Router>enable
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#ip route 10.0.0.0 255.0.0.0 20.0.0.1 10
Router(config)#ip route 10.0.0.0 255.0.0.0 50.0.0.1 20
Router(config)#ip route 40.0.0.0 255.0.0.0 20.0.0.1 10
Router(config)#ip route 40.0.0.0 255.0.0.0 50.0.0.1 20
Router(config)#exit
Router#show ip route static
S 10.0.0.0/8 [10/0] via 20.0.0.1
S 40.0.0.0/8 [10/0] via 20.0.0.1
Router#
```

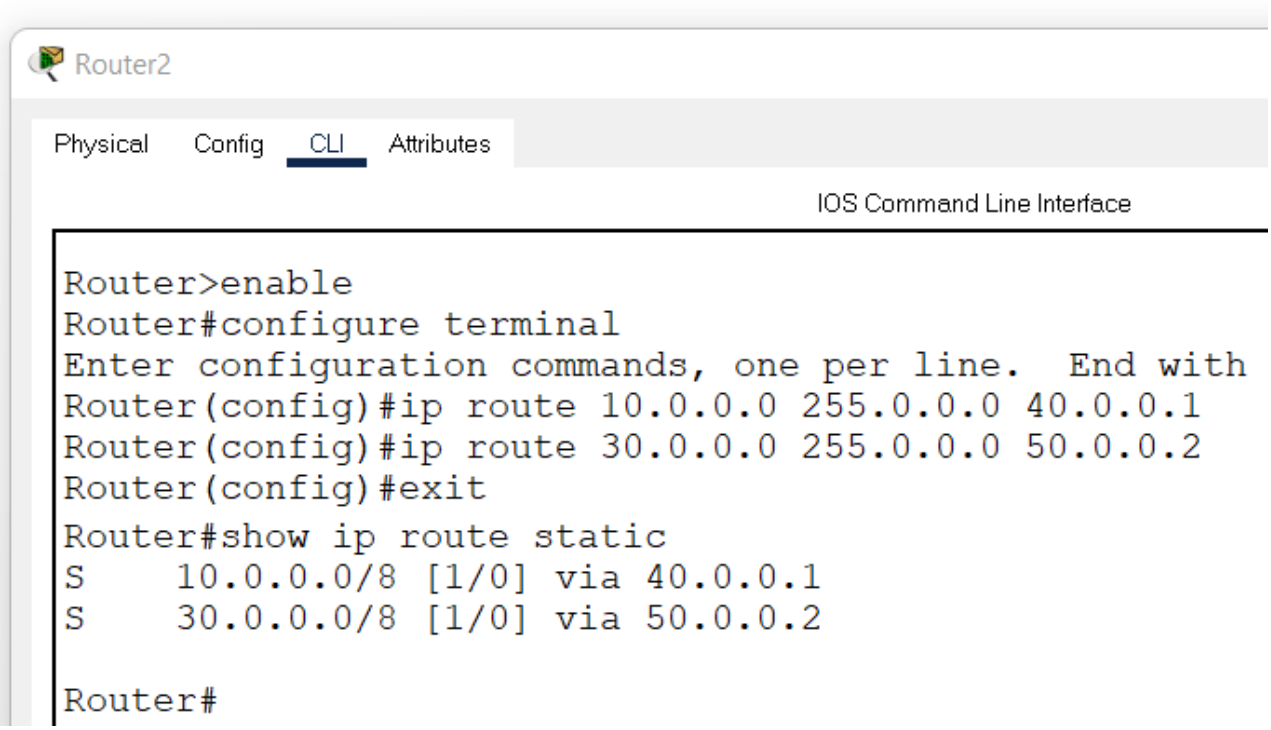


Router2 requirements

Create static routes for network 10.0.0.0/8 and network 30.0.0.0/8 and verify the router adds both routes to the routing table.

Router2 configuration

```
Router>enable
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#ip route 10.0.0.0 255.0.0.0 40.0.0.1
Router(config)#ip route 30.0.0.0 255.0.0.0 50.0.0.2
Router(config)#exit
Router#show ip route static
S 10.0.0.0/8 [1/0] via 40.0.0.1
S 30.0.0.0/8 [1/0] via 50.0.0.2
Router#
```



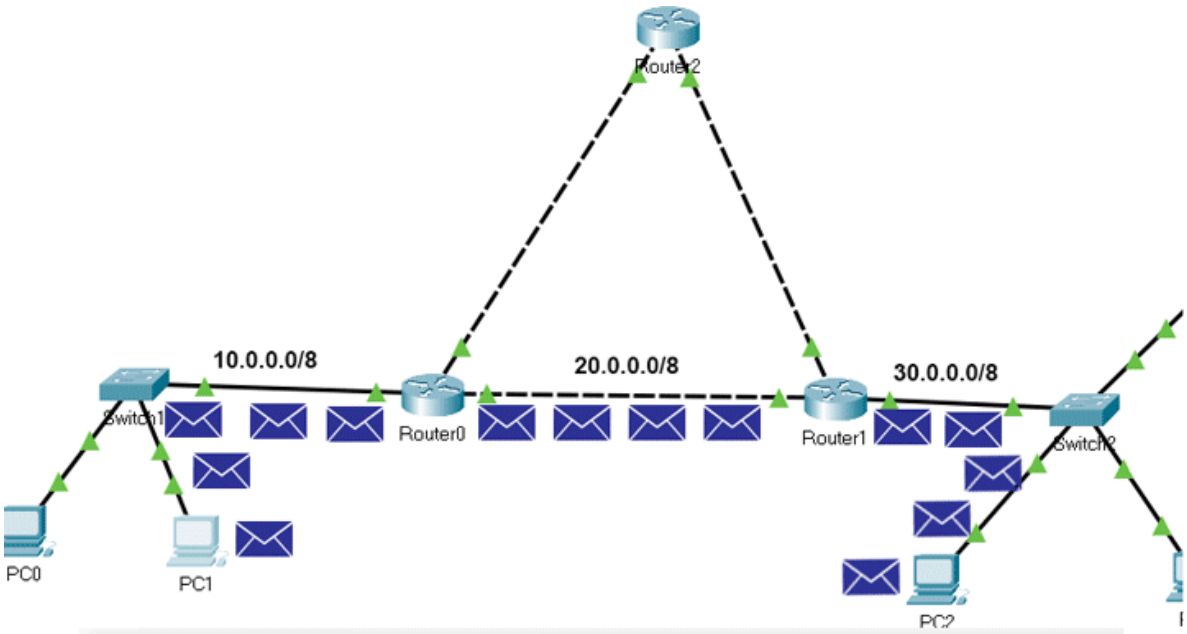
Verifying static routing

On Router0, we configured two routes for network 30.0.0.0/8. These routes are via Router1 and via Router2. We set the first route (via-Router1) as the main route and the second route as the backup route. We can verify this configuration in two ways.

By sending ping requests to a PC of network 30.0.0.0/8 and tracing the path they take to reach the network 30.0.0.0/8. For this, you can use **'tracert'** command on a PC of network 10.0.0.0/8. The **'tracert'** command sends ping requests to the destination host and tracks the path they take to reach the destination.

By listing the routing table entries on Router0. Since a router uses the routing table to forward data packets, you can check the routing table to figure out the route the router uses to forward data packets for each destination.

The following image shows the above testing.



Router0

Physical Config CLI Attributes

IOS Command Line Interface

```
Router>enable
Router#show ip route static
     30.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
S       30.0.0.0/8 [10/0] via 20.0.0.2
S       30.0.0.100/32 [10/0] via 40.0.0.2
S    50.0.0.0/8 [10/0] via 40.0.0.2

Router#
```

PC1

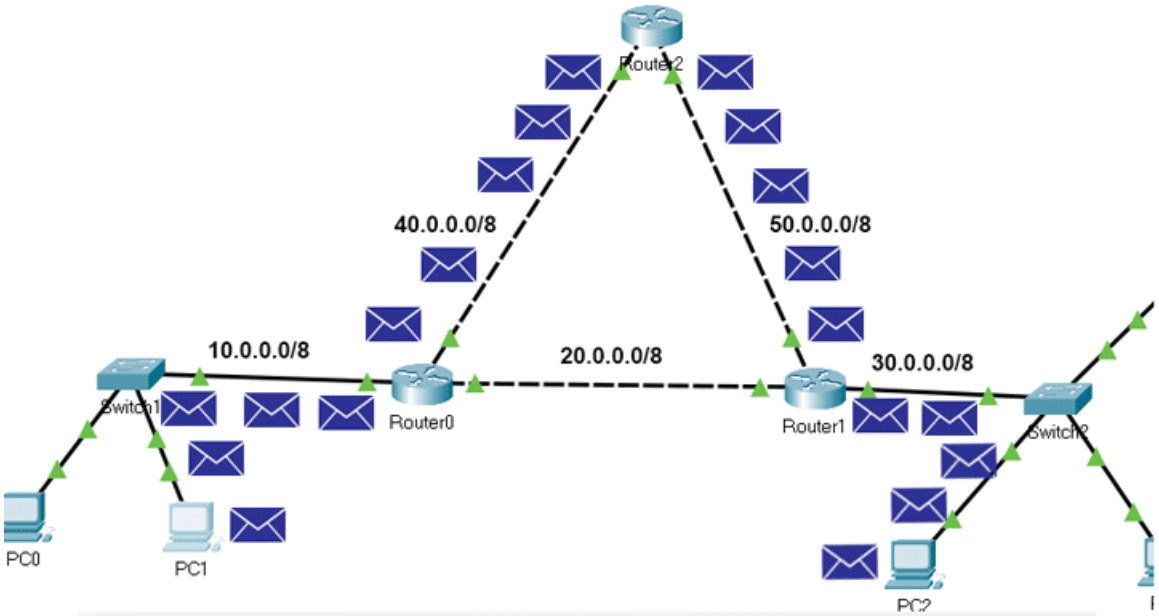
Physical Config Desktop Programming Attributes

Command Prompt

```
C:\>tracert 30.0.0.20

Tracing route to 30.0.0.20 over a maximum of 30
hops:

  1    0 ms      0 ms      0 ms      10.0.0.1
  2    0 ms      0 ms      0 ms      20.0.0.2
  3    *         0 ms      0 ms      30.0.0.20

Trace complete.

C:\>
```

We also configured a separate static host route for the host 30.0.0.100/8. The router must use this route to forward data packets to the host 30.0.0.100/8. To verify this, you can do the same testing for the host 30.0.0.100/8.

The following image shows this testing.



Router0

Physical Config CLI Attributes

IOS Command Line Interface

```
Router>enable
Router#show ip route static
     30.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
S       30.0.0.0/8 [10/0] via 20.0.0.2
S       30.0.0.100/32 [10/0] via 40.0.0.2
S    50.0.0.0/8 [10/0] via 40.0.0.2

Router#
```

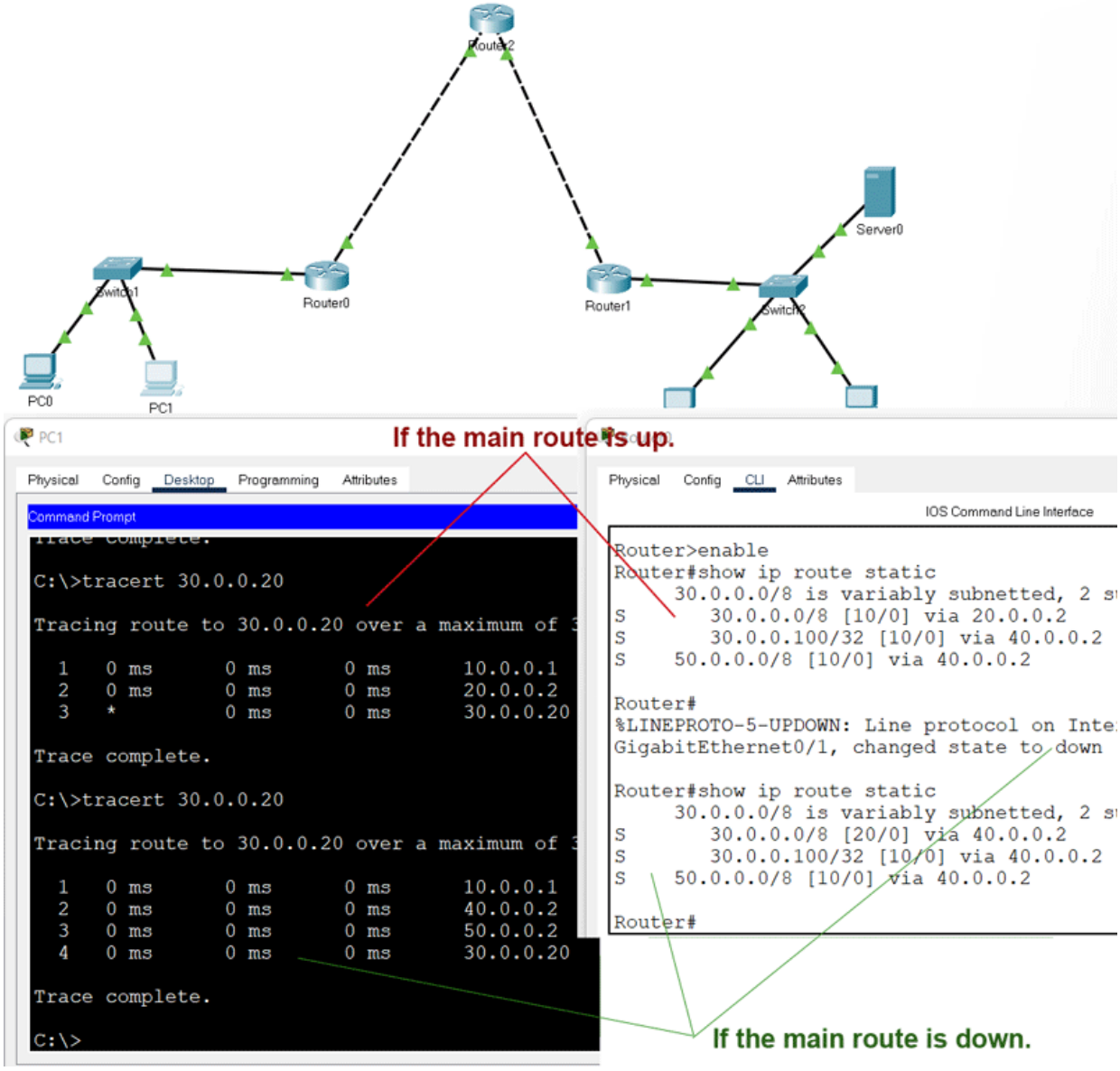
PC1

Physical Config Desktop Programming Attributes

```
Packet Tracer PC Command Line 1.0
C:\>tracert 30.0.0.100

Tracing route to 30.0.0.100 over a maximum of 30
hops:

  1    1 ms       0 ms       0 ms       10.0.0.1
  2    *          0 ms       0 ms       40.0.0.2
  3    *          *          0 ms       20.0.0.2
  4    *          0 ms       0 ms       30.0.0.100

Trace complete.

C:\>
```

We also configured a backup route for network 30.0.0.0/8. The router must put the backup route to the routing table and use it to forward data packets to network 30.0.0.0/8 when the main route fails. To verify this, we have to simulate the failure of the main route.

To simulate the failure of the main route, you can delete the link between Router0 and Router1. After deleting the link, do the same testing again for the network 30.0.0.0/8.



The following link provides the configured packet tracer lab of the above example.
Packet Tracer Lab with Static Routing Configuration

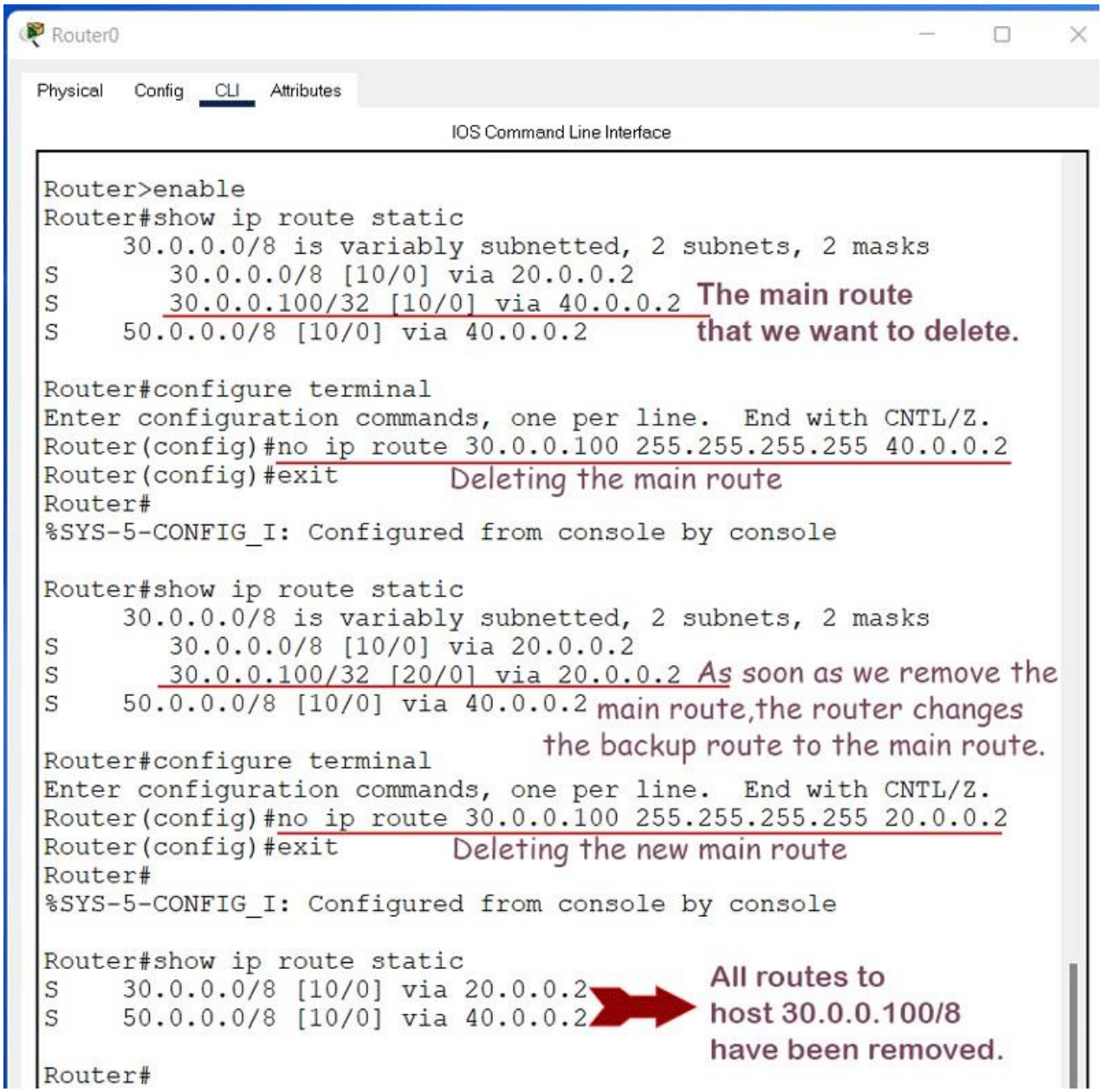
Deleting a static route

To delete a static route, use the following steps.

- Use the '**show ip route static**' command to print all static routes.
- Note down the route you want to delete.
- Use the '**no ip route**' command to delete the route.

If you have a backup route, the backup route becomes the main route when you delete the main route.

In our example, we have a backup route and a main route for the host 30.0.0.100/8. The following image shows how to delete both routes.


Router0
Physical
Config
CLI
Attributes
IOS Command Line Interface
Router>enable
Router#show ip route static
30.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
S 30.0.0.0/8 [10/0] via 20.0.0.2
S 30.0.0.100/32 [10/0] via 40.0.0.2
S 50.0.0.0/8 [10/0] via 40.0.0.2
The main route that we want to delete.
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#no ip route 30.0.0.100 255.255.255.255 40.0.0.2
Router(config)#exit
Deleting the main route
Router#
%SYS-5-CONFIG_I: Configured from console by console
Router#show ip route static
30.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
S 30.0.0.0/8 [10/0] via 20.0.0.2
S 30.0.0.100/32 [20/0] via 20.0.0.2
S 50.0.0.0/8 [10/0] via 40.0.0.2
As soon as we remove the main route,the router changes the backup route to the main route.
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#no ip route 30.0.0.100 255.255.255.255 20.0.0.2
Router(config)#exit
Deleting the new main route
Router#
%SYS-5-CONFIG_I: Configured from console by console
Router#show ip route static
S 30.0.0.0/8 [10/0] via 20.0.0.2
S 50.0.0.0/8 [10/0] via 40.0.0.2
All routes to host 30.0.0.100/8 have been removed.
Router#

Practical 11

AIM:- b)Simulate RIP using CISCO Packet Tracer

Initial IP configuration

Device	Interface	IP Configuration	Connected with
PC0	Fast Ethernet	10.0.0.2/8	Router0's Fa0/1
Router0	Fa0/1	10.0.0.1/8	PC0's Fast Ethernet
Router0	S0/0/1	192.168.1.254/30	Router2's S0/0/1
Router0	S0/0/0	192.168.1.249/30	Router1's S0/0/0
Router1	S0/0/0	192.168.1.250/30	Router0's S0/0/0
Router1	S0/0/1	192.168.1.246/30	Router2's S0/0/0
Router2	S0/0/0	192.168.1.245/30	Router1's S0/0/1
Router2	S0/0/1	192.168.1.253/30	Router0's S0/0/1
Router2	Fa0/1	20.0.0.1/30	PC1's Fast Ethernet
PC1	Fast Ethernet	20.0.0.2/30	Router2's Fa0/1

Assign IP address to PCs

Double click **PCs** and click **Desktop** menu item and click **IP Configuration**. Assign IP address referring the above table.

Assign IP address to interfaces of routers

Double click **Router0** and click **CLI** and press **Enter key** to access the command prompt of **Router0**.

We need to configure IP address and other parameters on interfaces before we could actually use them for routing. Interface mode is used to assign IP address and other parameters. Interface mode can be accessed from global configuration mode. Following commands are used to access the global configuration mode.

```
Router>enable
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#
```

From global configuration mode we can enter in interface mode. From there we can configure the interface. Following commands will assign IP address on FastEthernet0/0.

```
Router(config)#interface fastEthernet 0/0
Router(config-if)#ip address 10.0.0.1 255.0.0.0
Router(config-if)#no shutdown
Router(config-if)#exit
Router(config)#
```

interface *fastEthernet 0/0* command is used to enter in interface mode.
ip address *10.0.0.1 255.0.0.0* command will assign IP address to interface.

no shutdown command will bring the interface up.
exit command is used to return in global configuration mode.
Serial interface needs two additional parameters **clock rate** and **bandwidth**. Every serial cable has two ends DTE and DCE. These parameters are always configured at DCE end.
We can use **show controllers interface** command from privilege mode to check the cable's end.

```
Router#show controllers serial 0/0/0
Interface Serial0/0/0
Hardware is PowerQUICC MPC860
DCE V.35, clock rate 2000000
[Output omitted]
```

Fourth line of output confirms that DCE end of serial cable is attached. If you see DTE here instead of DCE skip these parameters.
Now we have necessary information let's assign IP address to serial interface.

```
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#interface serial 0/0/0
Router(config-if)#ip address 192.168.1.249 255.255.255.252
Router(config-if)#clock rate 64000
Router(config-if)#bandwidth 64
Router(config-if)#no shutdown
Router(config-if)#exit
Router(config)#interface serial 0/0/1
Router(config-if)#ip address 192.168.1.254 255.255.255.252
Router(config-if)#clock rate 64000
Router(config-if)#bandwidth 64
Router(config-if)#no shutdown
Router(config-if)#exit
Router(config)#
```

Router#configure terminal Command is used to enter in global configuration mode.
Router(config)#interface *serial 0/0/0* Command is used to enter in interface mode.
Router(config-if)#ip address *192.168.1.249 255.255.255.252* Command assigns IP address to interface. For serial link we usually use IP address from /30 subnet.
Router(config-if)#clock rate *64000* And **Router(config-if)#bandwidth *64*** In real life environment these parameters control the data flow between serial links and need to be set at service providers end. In lab environment we need not to worry about these values. We can use these values.
Router(config-if)#no shutdown Command brings interface up.
Router(config-if)#exit Command is used to return in global configuration mode.
We will use same commands to assign IP addresses on interfaces of remaining routers. We need to provided clock rate and bandwidth only on DCE side of serial interface. Following command will assign IP addresses on interface of Router1.

Router1

```
Router>enable
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#interface serial 0/0/0
Router(config-if)#ip address 192.168.1.250 255.255.255.252
```

```
Router(config-if)#no shutdown
Router(config-if)#exit
Router(config)#interface serial 0/0/1
Router(config-if)#ip address 192.168.1.246 255.255.255.252
Router(config-if)#clock rate 64000
Router(config-if)#bandwidth  64
Router(config-if)#no shutdown
Router(config-if)#exit
```

Use same commands to assign IP addresses on interfaces of Router2.

Router2

```
Router>enable
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#interface fastEthernet 0/0
Router(config-if)#ip address 20.0.0.1 255.0.0.0
Router(config-if)#no shutdown
Router(config-if)#exit
Router(config)#interface serial 0/0/0
Router(config-if)#ip address 192.168.1.245 255.255.255.252
Router(config-if)#no shutdown
Router(config-if)#exit
Router(config)#interface serial 0/0/1
Router(config-if)#ip address 192.168.1.253 255.255.255.252
Router(config-if)#no shutdown
Router(config-if)#exit
```

Now routers have information about the networks that they have on their own interfaces. Routers will not exchange this information between them on their own. We need to implement RIP routing protocol that will insist them to share this information.

Configure RIP routing protocol

Configuration of RIP protocol is much easier than you think. It requires only two steps to configure the RIP routing.

- Enable RIP routing protocol from global configuration mode.
- Tell RIP routing protocol which networks you want to advertise.

Let's configure it in Router0

Router0

```
Router0(config)#router rip
Router0(config-router)# network 10.0.0.0
Router0(config-router)# network 192.168.1.252
Router0(config-router)# network 192.168.1.248
```

router rip command tell router to enable the RIP routing protocol.

network command allows us to specify the networks which we want to advertise. We only need to specify the networks which are directly connected with the router.

That's all we need to configure the RIP. Follow same steps on remaining routers.

Router1

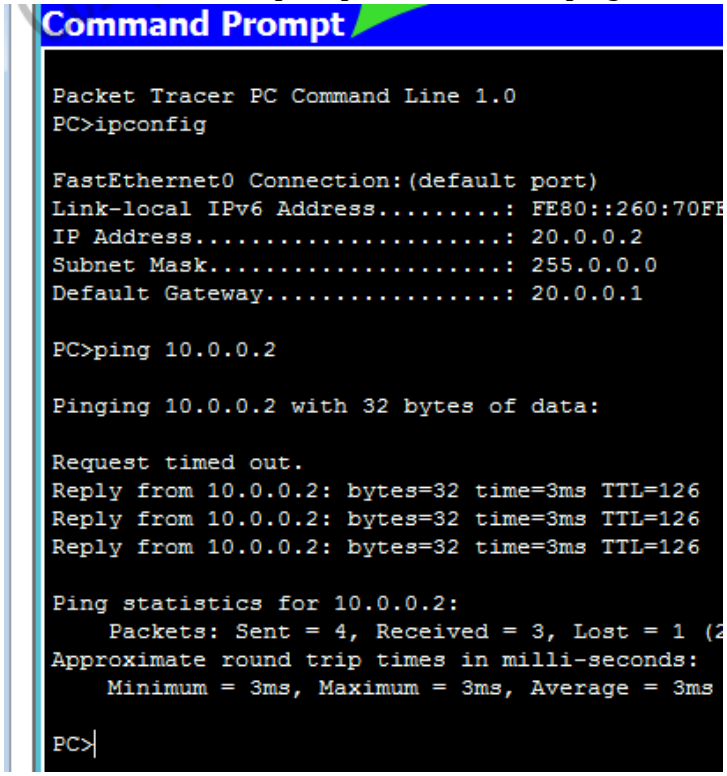
```
Router1(config)#router rip
Router1(config-router)# network 192.168.1.244
Router1(config-router)# network 192.168.1.248
```

Router2

```
Router2(config)#router rip
```

```
Router2(config-router)# network 20.0.0.0
Router2(config-router)# network 192.168.1.252
Router2(config-router)# network 192.168.1.244
```

That's it. Our network is ready to take the advantage of RIP routing. To verify the setup we will use ping command. ping command is used to test the connectivity between two devices.
Access the command prompt of **PC1** and use *ping* command to test the connectivity from **PC0**.



RIP protocol automatically manage all routes for us. If one route goes down, it automatically switches to another available. To explain this process more clearly we have added one more route in our network.
Currently there are two routes between PC0 and PC1.

Route 1

PC0 [Source / destination – 10.0.0.2] <==> Router0 [FastEthernet0/1 – 10.0.0.1] <==> Router0 [Serial0/0/1 – 192.168.1.254] <==> Router2 [Serial 0/0/1 – 192.168.1.253] <==> Router2 [FastEthernet0/0 – 20.0.0.1] <==> PC1 [Destination /source – 20.0.0.2]

Route 2

PC0 [Source / destination – 10.0.0.2] <==> Router0 [FastEthernet0/1 – 10.0.0.1] <==> Router0 [Serial0/0/0 – 192.168.1.249] <==> Router1 [Serial 0/0/0 – 192.168.1.250] <==> Router1 [Serial 0/0/1 – 192.168.1.246] <==> Router2 [Serial 0/0/0 – 192.168.1.245] <==> Router2 [FastEthernet0/0 – 20.0.0.1] <==> PC1 [Destination /source – 20.0.0.2]
By default RIP will use the route that has low hops counts between source and destination. In our network route1 has low hops counts, so it will be selected. We can use *tracert* command to verify it.
Now suppose route1 is down. We can simulate this situation by removing the cable attached between **Router0 [s0/0/1]** and **Router2 [s0/0/1]**.

What will happen now? There is no need to worry. RIP will automatically reroute the traffic.
Use ***tracert*** command again to see the magic of dynamic routing.

Practical 12

AIM: - a) Implement echo client server using TCP/UDP sockets.

Algorithm:-

- **TCP Echo Server and Client**:

- The **server** listens on port 12345 and waits for a client connection. Upon receiving a message, it echoes it back.
- The **client** connects to the server, sends a message, and prints the echoed message received from the server.

- **UDP Echo Server and Client**:

- The **server** listens for UDP packets on port 12345 and echoes the received packet back to the client.
- The **client** sends a UDP packet to the server and prints the echoed response.

Code:

TCP Echo Server:

```
// tcp_echo_server.c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 12345
#define BUF_SIZE 1024

int main() {
int server_fd, new_sock;
struct sockaddr_in server_addr, client_addr;
socklen_t addr_len = sizeof(client_addr);
char buffer[BUF_SIZE];

// Create TCP socket
server_fd = socket(AF_INET, SOCK_STREAM, 0);
if (server_fd < 0) {
perror("Socket failed");
exit(1);
}

server_addr.sin_family = AF_INET;
server_addr.sin_addr.s_addr = INADDR_ANY;
```

```
server_addr.sin_port = htons(PORT);

// Bind socket
if (bind(server_fd, (struct sockaddr*)&server_addr, sizeof(server_addr)) < 0) {
perror("Bind failed");
exit(1);
}

// Listen for connections
if (listen(server_fd, 5) < 0) {
perror("Listen failed");
exit(1);
}

printf("TCP Echo Server listening on port %d...\n", PORT);

// Accept a client connection
new_sock = accept(server_fd, (struct sockaddr*)&client_addr, &addr_len);
if (new_sock < 0) {
perror("Accept failed");
exit(1);
}

// Echo loop
while (1) {
memset(buffer, 0, BUF_SIZE);
int len = recv(new_sock, buffer, BUF_SIZE, 0);
if (len <= 0) break;  // Connection closed or error
send(new_sock, buffer, len, 0);  // Echo back
}

close(new_sock);
close(server_fd);
return 0;
}
```

TCP Echo Client:

```
// tcp_echo_client.c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 12345
#define BUF_SIZE 1024

int main() {
   int sock;
   struct sockaddr_in server_addr;
   char buffer[BUF_SIZE];
```

```
    // Create TCP socket
    sock = socket(AF_INET, SOCK_STREAM, 0);
    if (sock < 0) {
        perror("Socket failed");
        exit(1);
    }

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

    // Connect to server
    if (connect(sock, (struct sockaddr*)&server_addr, sizeof(server_addr)) < 0) {
        perror("Connection failed");
        exit(1);
    }

    // Send data to server and receive echo
    printf("Enter message: ");
    fgets(buffer, BUF_SIZE, stdin);
    send(sock, buffer, strlen(buffer), 0);

    int len = recv(sock, buffer, BUF_SIZE, 0);
    buffer[len] = '\0';  // Null-terminate the string
    printf("Received echo: %s", buffer);

    close(sock);
    return 0;
}
```

Output:-

Enter message: Hello, Server!
Received echo: Hello, Serv

Practical 12

AIM: - b) Implement chat client server using TCP/UDP sockets.

Algorithm:-

- TCP Chat Server:
- This server accepts multiple client connections. Each client is handled in a separate thread. When a client sends a message, it is broadcast to all other connected clients.
- TCP Chat Client:
- The client connects to the server, sends messages, and listens for incoming messages from the server (other clients).
- UDP Chat Server:
- The UDP server listens for incoming messages from any client. Upon receiving a message, it broadcasts it to all other connected clients.
- UDP Chat Client:
- The client sends messages to the UDP server and receives messages broadcasted by other clients.

Code:

TCP Chat Server:

```
// tcp_chat_server.c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <pthread.h>

#define PORT 12345
#define BUF_SIZE 1024
#define MAX_CLIENTS 10

int client_sockets[MAX_CLIENTS];

void *handle_client(void *arg) {
int client_socket = *(int *)arg;
char buffer[BUF_SIZE];
int bytes_received;

while (1) {
memset(buffer, 0, BUF_SIZE);
bytes_received = recv(client_socket, buffer, BUF_SIZE, 0);
if (bytes_received <= 0) {
printf("Client disconnected.\n");
close(client_socket);
return NULL;
}

printf("Received: %s", buffer);
```

Practical 12

```
// Send message to all clients
for (int i = 0; i < MAX_CLIENTS; i++) {
if (client_sockets[i] != 0 && client_sockets[i] != client_socket) {
send(client_sockets[i], buffer, bytes_received, 0);
}
}
}
}

int main() {
int server_fd, client_socket, addr_len;
struct sockaddr_in server_addr, client_addr;
pthread_t thread_id;

// Create socket
server_fd = socket(AF_INET, SOCK_STREAM, 0);
if (server_fd == -1) {
perror("Socket creation failed");
exit(1);
}

server_addr.sin_family = AF_INET;
server_addr.sin_addr.s_addr = INADDR_ANY;
server_addr.sin_port = htons(PORT);

// Bind the socket
if (bind(server_fd, (struct sockaddr *)&server_addr, sizeof(server_addr)) == -1) {
perror("Bind failed");
exit(1);
}

// Listen for connections
if (listen(server_fd, MAX_CLIENTS) == -1) {
perror("Listen failed");
exit(1);
}

printf("TCP Chat Server listening on port %d...\n", PORT);

while (1) {
addr_len = sizeof(client_addr);
client_socket = accept(server_fd, (struct sockaddr *)&client_addr, (socklen_t *)&addr_len);
if (client_socket < 0) {
perror("Accept failed");
continue;
}

// Add client to the list of client sockets
for (int i = 0; i < MAX_CLIENTS; i++) {
```

```
if (client_sockets[i] == 0) {
client_sockets[i] = client_socket;
break;
}
}

// Create a new thread to handle the client
pthread_create(&thread_id, NULL, handle_client, (void *)&client_socket);
}

close(server_fd);
return 0;
}
```

UDP Chat Client:

```
// udp_chat_client.c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 12345
#define BUF_SIZE 1024

int main() {
    int sock;
    struct sockaddr_in server_addr;
    char buffer[BUF_SIZE];

    // Create UDP socket
    sock = socket(AF_INET, SOCK_DGRAM, 0);
    if (sock < 0) {
        perror("Socket creation failed");
        exit(1);
    }

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

    printf("Connected to chat server. Type messages and press Enter.\n");

    // Chat loop
    while (1) {
        printf("You: ");
        fgets(buffer, BUF_SIZE, stdin);
        sendto(sock, buffer, strlen(buffer), 0, (struct sockaddr *)&server_addr, sizeof(server_addr));
```

Practical 12

```
        // Receive message from the server (other clients' messages)
        int len = recvfrom(sock, buffer, BUF_SIZE, 0, NULL, NULL);
        buffer[len] = '\0';  // Null-terminate the string
        printf("Other: %s", buffer);
    }

    close(sock);
    return 0;
}
```

Output:-

```
$ ./tcp_chat_client
Connected to chat server. Type messages and press Enter.
You: Hi, Client 1!
Other: Hi, Client 1!
You: What are you up to today?
Other: What are you up to today?
```

```
$ ./tcp_chat_server
TCP Chat Server listening on port 12345...
Received: Hello, everyone!
Received: How's everyone doing?
Received: Hi, Client 1!
Received: What are you up to today?
```

Practical 14

AIM: - Implement your own ping program

Algorithm:

- **Create a raw socket with the protocol type IPPROTO_ICMP.**
- **Construct the ICMP echo request packet.**
- **Send the ICMP echo request to the destination host.**
- **Wait for the ICMP echo reply.**
- **Measure the round-trip time.**
- **Repeat the process and display the results**

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <netinet/ip.h>
#include <netinet/in.h>
#include <netinet/ip_icmp.h>
#include <time.h>

#define ICMP_HEADER_LEN 8
#define PACKET_SIZE 64
#define TIMEOUT 1    // Timeout for receiving a response in seconds

// Checksum function for ICMP header
unsigned short checksum(void *b, int len) {
   unsigned short *buf = b;
   unsigned int sum = 0;
   unsigned short result;

   for (sum = 0; len > 1; len -= 2) {
      sum += *buf++;
   }

   if (len == 1) {
      sum += *(unsigned char *)buf;
   }

   sum = (sum >> 16) + (sum & 0xFFFF);
   sum += (sum >> 16);
   result = ~sum;
   return result;
```

```
}

// ICMP Echo Request Packet Construction
void send_ping(int sockfd, struct sockaddr_in *dest_addr, int pid) {
    char packet[PACKET_SIZE];
    struct icmp *icmp_hdr = (struct icmp *)packet;
    struct timeval *timestamp = (struct timeval *)(packet + ICMP_HEADER_LEN);

    // Set up ICMP header
    icmp_hdr->icmp_type = ICMP_ECHO;
    icmp_hdr->icmp_code = 0;
    icmp_hdr->icmp_id = pid;
    icmp_hdr->icmp_seq = 1;  // Sequence number is 1 (you can increment it if needed)
    gettimeofday(timestamp, NULL);  // Set timestamp
    icmp_hdr->icmp_cksum = 0;  // Initially set checksum to 0
    icmp_hdr->icmp_cksum = checksum(packet, PACKET_SIZE);

    // Send the packet
    ssize_t bytes_sent = sendto(sockfd, packet, PACKET_SIZE, 0, (struct sockaddr *)dest_addr,
sizeof(*dest_addr));
    if (bytes_sent < 0) {
        perror("Failed to send ICMP packet");
        exit(1);
    }
}

// Receive ICMP Echo Reply
void receive_ping(int sockfd, struct sockaddr_in *dest_addr, int pid) {
    char packet[PACKET_SIZE];
    struct sockaddr_in recv_addr;
    socklen_t addr_len = sizeof(recv_addr);
    struct ip *ip_hdr;
    struct icmp *icmp_hdr;
    struct timeval *recv_timestamp;
    struct timeval start_time, end_time;
    int bytes_received;

    // Record start time for round-trip calculation
    gettimeofday(&start_time, NULL);

    // Receive reply
    bytes_received = recvfrom(sockfd, packet, PACKET_SIZE, 0, (struct sockaddr *)&recv_addr,
&addr_len);
    if (bytes_received < 0) {
        perror("Failed to receive ICMP packet");
        return;
    }

    // Get ICMP and IP headers
```

Practical 14

```
    ip_hdr = (struct ip *)packet;
    icmp_hdr = (struct icmp *)(packet + (ip_hdr->ip_hl << 2));  // IP header length
    recv_timestamp = (struct timeval *)(packet + ICMP_HEADER_LEN);

    // Check if the reply is from the correct host and matches the sequence number
    if (icmp_hdr->icmp_type == ICMP_ECHOREPLY && icmp_hdr->icmp_id == pid) {
        gettimeofday(&end_time, NULL);

        // Calculate round-trip time
        double rtt = ((end_time.tv_sec - start_time.tv_sec) * 1000.0) + ((end_time.tv_usec -
start_time.tv_usec) / 1000.0);
        printf("Received reply from %s: icmp_seq=%d time=%.3f ms\n",
inet_ntoa(recv_addr.sin_addr), icmp_hdr->icmp_seq, rtt);
    } else {
        printf("Received unexpected ICMP reply\n");
    }
}

// Main Ping Program
int main(int argc, char *argv[]) {
    if (argc != 2) {
        fprintf(stderr, "Usage: %s <hostname>\n", argv[0]);
        exit(1);
    }

    const char *hostname = argv[1];
    struct sockaddr_in dest_addr;
    struct hostent *host;
    int sockfd, pid;

    // Get destination address
    host = gethostbyname(hostname);
    if (host == NULL) {
        perror("Failed to resolve hostname");
        exit(1);
    }

    memset(&dest_addr, 0, sizeof(dest_addr));
    dest_addr.sin_family = AF_INET;
    dest_addr.sin_port = 0;
    dest_addr.sin_addr = *((struct in_addr *)host->h_addr_list[0]);

    // Create raw socket for ICMP
    sockfd = socket(AF_INET, SOCK_RAW, IPPROTO_ICMP);
    if (sockfd < 0) {
        perror("Failed to create raw socket");
        exit(1);
    }
```

Practical 14

```
    // Get process ID for unique ICMP identification
    pid = getpid() & 0xFFFF;

    printf("PING %s (%s): %d bytes of data.\n", hostname, inet_ntoa(dest_addr.sin_addr),
PACKET_SIZE);

    while (1) {
        send_ping(sockfd, &dest_addr, pid);
        receive_ping(sockfd, &dest_addr, pid);
        sleep(1);  // Send ping every second
    }

    close(sockfd);
    return 0;
}
```

Output:-

PING google.com (142.250.192.78): 64 bytes of data.
Received reply from 142.250.192.78: icmp_seq=1 time=12.345 ms
Received reply from 142.250.192.78: icmp_seq=1 time=10.567 ms
Received reply from 142.250.192.78: icmp_seq=1 time=11.456 ms

Practical 14

AIM: - Write a code using RAW sockets to implement packet sniffing.

Algorithm:

Raw Socket Creation:

- The program uses socket(AF_INET, SOCK_RAW, IPPROTO_IP) to create a raw socket that captures all IP packets. IPPROTO_IP specifies that we want to capture all IP packets, regardless of the protocol.

Packet Reception:

- The program uses recvfrom() to receive raw packets from the network. It stores the captured data in a buffer.
- Each packet is parsed and the IP header is extracted.

Handling Different Protocols:

- Depending on the protocol type (TCP, UDP, ICMP), the program calls different functions to parse and print the corresponding header fields.
- If the packet is TCP, the program prints the TCP header; for UDP, the UDP header is printed; and for ICMP, the ICMP header is printed.

Printing Header Information:

- For each protocol, we print useful header fields (like IP addresses, ports, sequence numbers, etc.) to provide insights into the packet's structure.

Infinite Sniffing Loop:

- The program continuously receives packets in an infinite loop and prints the details of each packet. The loop only stops if there is an error or the program is terminated.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <netinet/ip.h>
#include <netinet/in.h>
#include <netinet/udp.h>
#include <netinet/tcp.h>
#include <netinet/ip_icmp.h>
#include <netdb.h>

// Define a buffer size for the packet
#define BUFFER_SIZE 65536

// Function to print the IP header
void print_ip_header(struct ip *ip_header) {
    printf("IP Header:\n");
```

Practical 14

```
    printf("   |-IP Version        : %d\n", ip_header->ip_v);
    printf("   |-IP Header Length  : %d bytes\n", ip_header->ip_hl * 4);
    printf("   |-Type Of Service   : %d\n", ip_header->ip_tos);
    printf("   |-IP Total Length   : %d bytes\n", ntohs(ip_header->ip_len));
    printf("   |-Identification    : %d\n", ntohs(ip_header->ip_id));
    printf("   |-IP Fragment Offset: %d\n", ntohs(ip_header->ip_off));
    printf("   |-Time To Live      : %d\n", ip_header->ip_ttl);
    printf("   |-Protocol          : %d\n", ip_header->ip_p);
    printf("   |-Checksum          : 0x%X\n", ntohs(ip_header->ip_sum));
    printf("   |-Source IP         : %s\n", inet_ntoa(ip_header->ip_src));
    printf("   |-Destination IP    : %s\n", inet_ntoa(ip_header->ip_dst));
}

// Function to print TCP header
void print_tcp_header(struct tcphdr *tcp_header) {
    printf("TCP Header:\n");
    printf("   |-Source Port      : %d\n", ntohs(tcp_header->th_sport));
    printf("   |-Destination Port : %d\n", ntohs(tcp_header->th_dport));
    printf("   |-Sequence Number  : %u\n", ntohl(tcp_header->th_seq));
    printf("   |-Acknowledgment   : %u\n", ntohl(tcp_header->th_ack));
    printf("   |-Data Offset      : %d\n", tcp_header->th_off);
    printf("   |-TCP Flags        : 0x%X\n", tcp_header->th_flags);
    printf("   |-Window Size      : %d\n", ntohs(tcp_header->th_win));
    printf("   |-Checksum         : 0x%X\n", ntohs(tcp_header->th_sum));
    printf("   |-Urgent Pointer   : %d\n", tcp_header->th_urp);
}

// Function to print UDP header
void print_udp_header(struct udphdr *udp_header) {
    printf("UDP Header:\n");
    printf("   |-Source Port      : %d\n", ntohs(udp_header->uh_sport));
    printf("   |-Destination Port : %d\n", ntohs(udp_header->uh_dport));
    printf("   |-UDP Length       : %d\n", ntohs(udp_header->uh_ulen));
    printf("   |-Checksum         : 0x%X\n", ntohs(udp_header->uh_sum));
}

// Function to print ICMP header
void print_icmp_header(struct icmphdr *icmp_header) {
    printf("ICMP Header:\n");
    printf("   |-Type     : %d\n", icmp_header->type);
    printf("   |-Code     : %d\n", icmp_header->code);
    printf("   |-Checksum : 0x%X\n", ntohs(icmp_header->checksum));
}

// Function to handle the sniffed packet
void handle_packet(char *buffer, int size) {
    struct ip *ip_header = (struct ip *)buffer;  // IP header
    int ip_header_len = ip_header->ip_hl * 4;
```

Practical 14

```
    // Print the IP header
    print_ip_header(ip_header);

    // Depending on the protocol type, we handle the packet accordingly
    switch (ip_header->ip_p) {
        case IPPROTO_TCP: {
            struct tcphdr *tcp_header = (struct tcphdr *)(buffer + ip_header_len);
            print_tcp_header(tcp_header);
            break;
        }
        case IPPROTO_UDP: {
            struct udphdr *udp_header = (struct udphdr *)(buffer + ip_header_len);
            print_udp_header(udp_header);
            break;
        }
        case IPPROTO_ICMP: {
            struct icmphdr *icmp_header = (struct icmphdr *)(buffer + ip_header_len);
            print_icmp_header(icmp_header);
            break;
        }
        default:
            printf("Other Protocol: %d\n", ip_header->ip_p);
            break;
    }

    printf("\n----------------------------------\n");
}

// Main packet sniffing function
int main() {
    int sockfd;
    char *buffer = (char *)malloc(BUFFER_SIZE);

    // Create a raw socket to capture all IP packets
    sockfd = socket(AF_INET, SOCK_RAW, IPPROTO_IP);
    if (sockfd < 0) {
        perror("Socket creation failed");
        return 1;
    }

    printf("Packet Sniffer Started...\n");

    // Infinite loop to sniff packets
    while (1) {
        struct sockaddr saddr;
        socklen_t saddr_len = sizeof(saddr);

        // Receive a packet
        int data_size = recvfrom(sockfd, buffer, BUFFER_SIZE, 0, &saddr, &saddr_len);
```

Practical 14

```
        if (data_size < 0) {
            perror("Failed to receive packet");
            return 1;
        }

        // Handle the received packet
        handle_packet(buffer, data_size);
    }

    // Close the socket
    close(sockfd);
    free(buffer);

    return 0;
}
```

Output:-

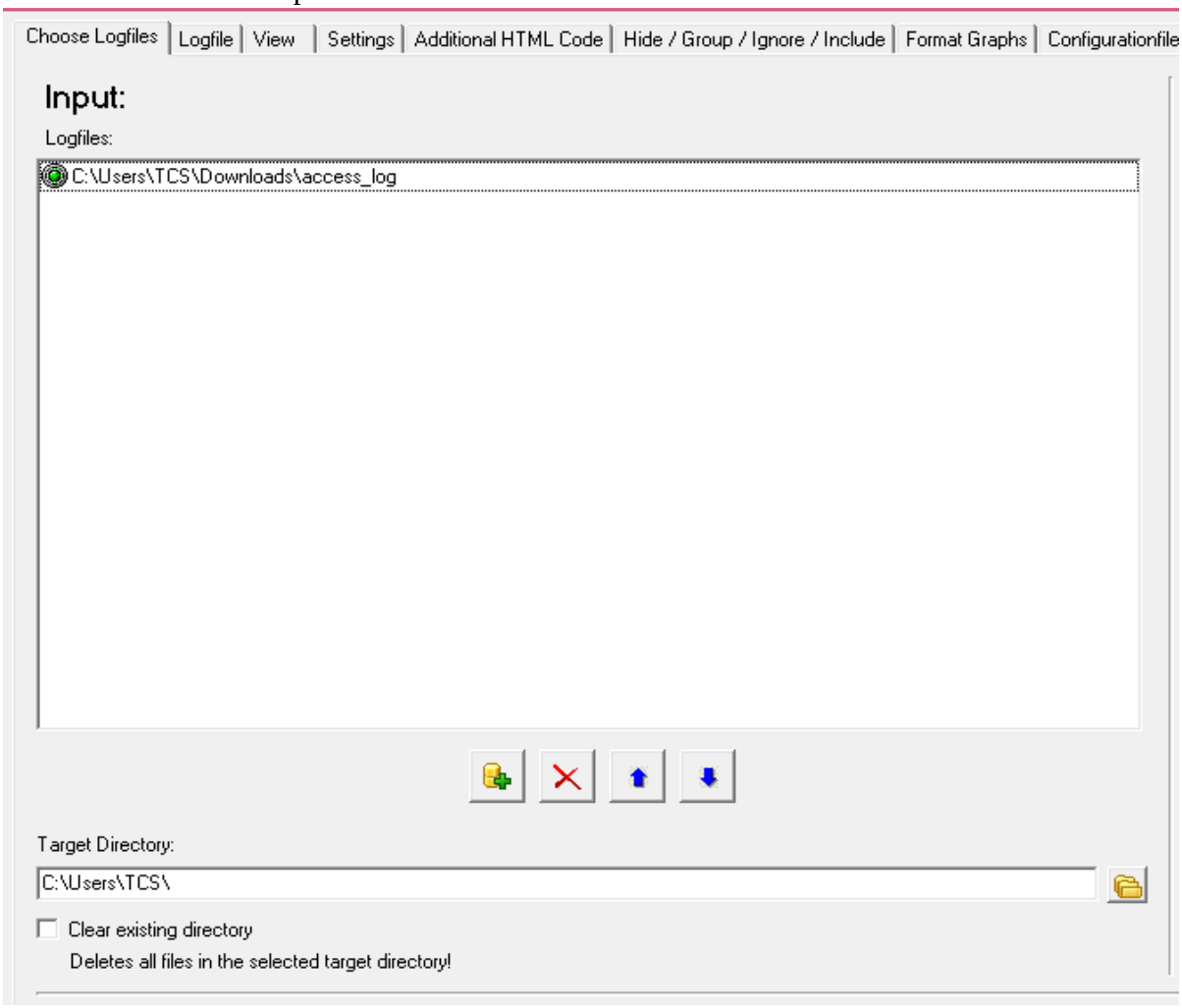
```
Packet Sniffer Started...
IP Header:
   |-IP Version        : 4
   |-IP Header Length  : 20 bytes
   |-Type Of Service   : 0
   |-IP Total Length   : 52 bytes
   |-Identification    : 12345
   |-IP Fragment Offset: 0
   |-Time To Live      : 64
   |-Protocol          : 6
   |-Checksum          : 0xB1E4
   |-Source IP         : 192.168.1.1
   |-Destination IP    : 192.168.1.2
TCP Header:
   |-Source Port       : 12345
   |-Destination Port  : 80
   |-Sequence Number   : 123456789
   |-Acknowledgment    : 987654321
   |-Data Offset       : 5
   |-TCP Flags         : 0x18
   |-Window Size       : 65535
   |-Checksum          : 0xB1E4
   |-Urgent Pointer    : 0
-----------------------------------
```

Practical 15

AIM:- To analyze the different types of web logs using Webalizer tool.

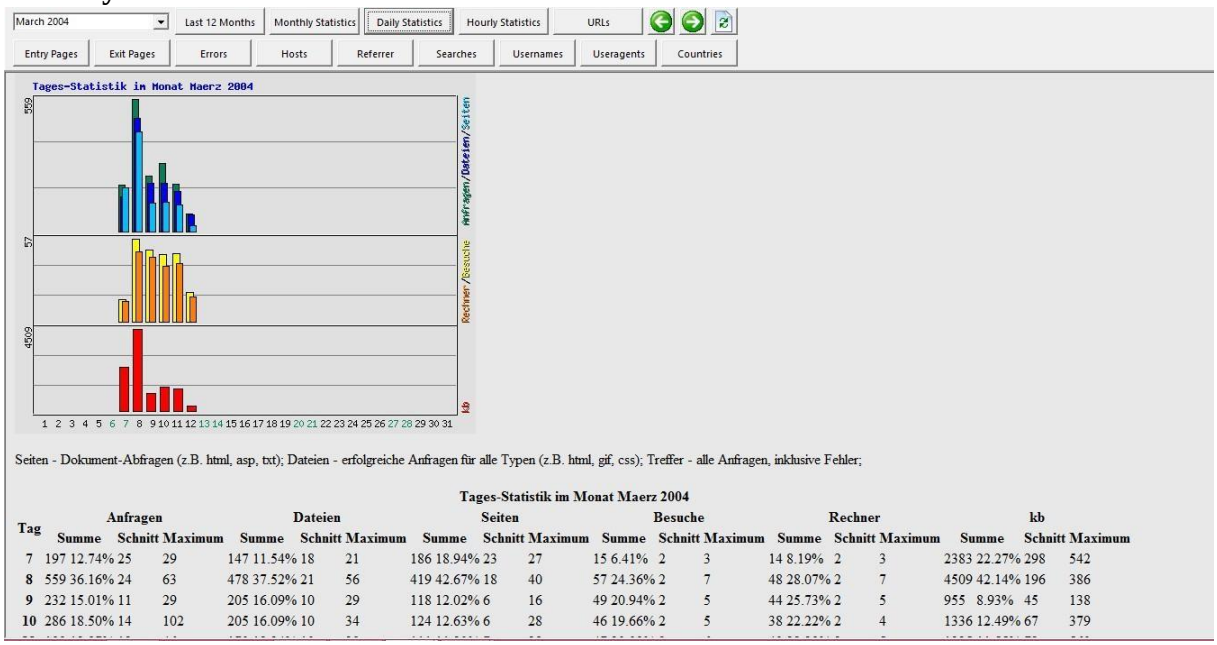
Procedure

Step1: Run webalizer windows version
Step2. Input web log file (down load from web)
Step3: Press Run webalizer



Output:

Monthly statistics

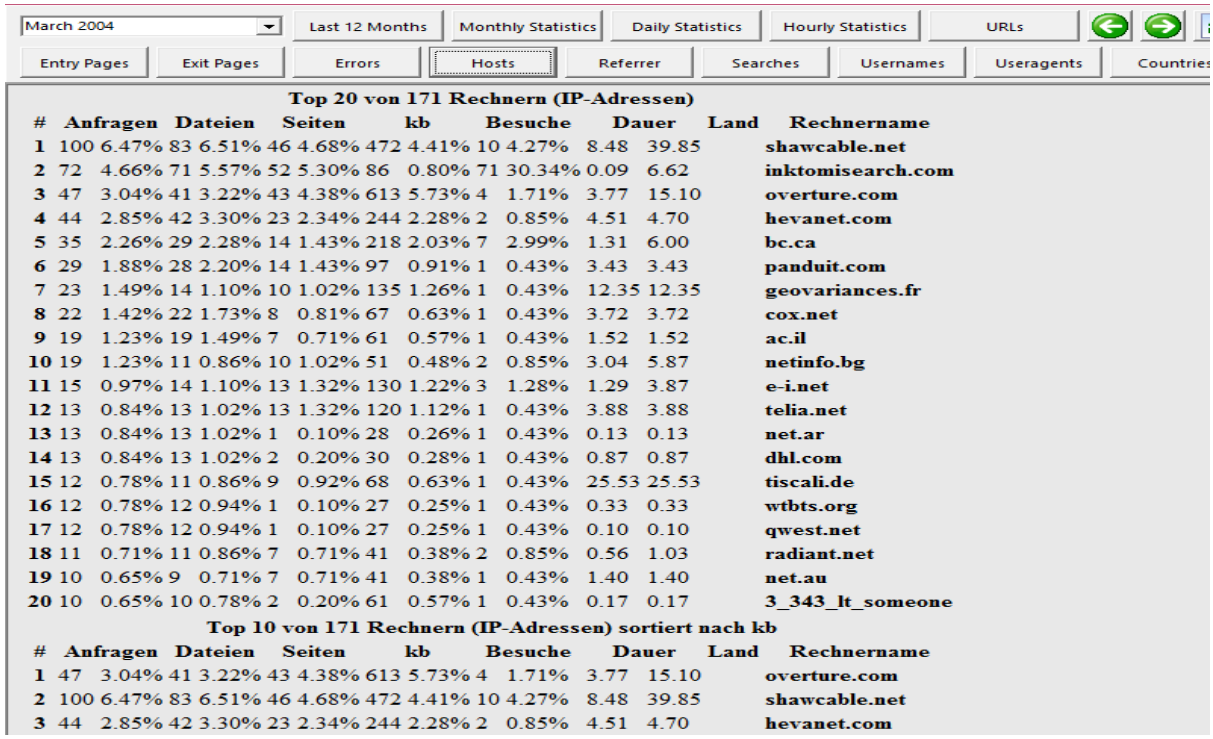


Seiten - Dokument-Abfragen (z.B. html, asp, txt); Dateien - erfolgreiche Anfragen für alle Typen (z.B. html, gif, css); Treffer - alle Anfragen, inklusive Fehler;

Tages-Statistik im Monat Maerz 2004

Tag	Anfragen			Dateien			Seiten			Besuche			Rechner			kb		
	Summe	Schnitt	Maximum	Summe	Schnitt	Maximum	Summe	Schnitt	Maximum	Summe	Schnitt	Maximum	Summe	Schnitt	Maximum	Summe	Schnitt	Maximum
7	197 12.74%	25	29	147 11.54%	18	21	186 18.94%	23	27	15 6.41%	2	3	14 8.19%	2	3	2383 22.27%	298	542
8	559 36.16%	24	63	478 37.52%	21	56	419 42.67%	18	40	57 24.36%	2	7	48 28.07%	2	7	4509 42.14%	196	386
9	232 15.01%	11	29	205 16.09%	10	29	118 12.02%	6	16	49 20.94%	2	5	44 25.73%	2	5	955 8.93%	45	138
10	286 18.50%	14	102	205 16.09%	10	34	124 12.63%	6	28	46 19.66%	2	5	38 22.22%	2	4	1336 12.49%	67	379

Hosts



Top 20 von 171 Rechnern (IP-Adressen)

#	Anfragen	Dateien	Seiten	kb	Besuche	Dauer	Land	Rechnername
1	100 6.47%	83 6.51%	46 4.68%	472 4.41%	10 4.27%	8.48	39.85	shawcable.net
2	72 4.66%	71 5.57%	52 5.30%	86 0.80%	71 30.34%	0.09	6.62	inktomisearch.com
3	47 3.04%	41 3.22%	43 4.38%	613 5.73%	4 1.71%	3.77	15.10	overture.com
4	44 2.85%	42 3.30%	23 2.34%	244 2.28%	2 0.85%	4.51	4.70	hevanet.com
5	35 2.26%	29 2.28%	14 1.43%	218 2.03%	7 2.99%	1.31	6.00	bc.ca
6	29 1.88%	28 2.20%	14 1.43%	97 0.91%	1 0.43%	3.43	3.43	panduit.com
7	23 1.49%	14 1.10%	10 1.02%	135 1.26%	1 0.43%	12.35	12.35	geovariances.fr
8	22 1.42%	22 1.73%	8 0.81%	67 0.63%	1 0.43%	3.72	3.72	cox.net
9	19 1.23%	19 1.49%	7 0.71%	61 0.57%	1 0.43%	1.52	1.52	ac.il
10	19 1.23%	11 0.86%	10 1.02%	51 0.48%	2 0.85%	3.04	5.87	netinfo.bg
11	15 0.97%	14 1.10%	13 1.32%	130 1.22%	3 1.28%	1.29	3.87	e-i.net
12	13 0.84%	13 1.02%	13 1.32%	120 1.12%	1 0.43%	3.88	3.88	telia.net
13	13 0.84%	13 1.02%	1 0.10%	28 0.26%	1 0.43%	0.13	0.13	net.ar
14	13 0.84%	13 1.02%	2 0.20%	30 0.28%	1 0.43%	0.87	0.87	dhl.com
15	12 0.78%	11 0.86%	9 0.92%	68 0.63%	1 0.43%	25.53	25.53	tiscali.de
16	12 0.78%	12 0.94%	1 0.10%	27 0.25%	1 0.43%	0.33	0.33	wtbts.org
17	12 0.78%	12 0.94%	1 0.10%	27 0.25%	1 0.43%	0.10	0.10	qwest.net
18	11 0.71%	11 0.86%	7 0.71%	41 0.38%	2 0.85%	0.56	1.03	radiant.net
19	10 0.65%	9 0.71%	7 0.71%	41 0.38%	1 0.43%	1.40	1.40	net.au
20	10 0.65%	10 0.78%	2 0.20%	61 0.57%	1 0.43%	0.17	0.17	3_343_lt_someone

Top 10 von 171 Rechnern (IP-Adressen) sortiert nach kb

#	Anfragen	Dateien	Seiten	kb	Besuche	Dauer	Land	Rechnername
1	47 3.04%	41 3.22%	43 4.38%	613 5.73%	4 1.71%	3.77	15.10	overture.com
2	100 6.47%	83 6.51%	46 4.68%	472 4.41%	10 4.27%	8.48	39.85	shawcable.net
3	44 2.85%	42 3.30%	23 2.34%	244 2.28%	2 0.85%	4.51	4.70	hevanet.com

User-agents

March 2004 | Last 12 Months | Monthly Statistics | Daily Statistics | Hourly Statistics | URLs

Entry Pages | Exit Pages | Errors | Hosts | Referrer | Searches | Usernames | Useragents | Countries

9	72	4.66%	71	5.57%	52	5.30%	86	0.80%	71	30.34%	0.09	6.62	inktomisearch.com
10	4	0.26%	4	0.31%	4	0.41%	74	0.69%	4	1.71%	0.00	0.00	rhyolite.com

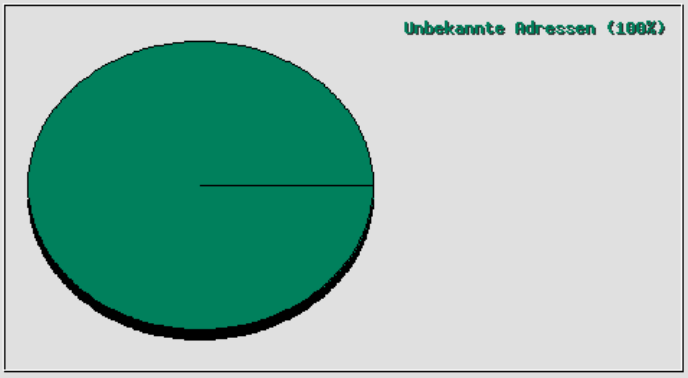
Top 1 von 1 Verweise

#	Anfragen		Besuche		Verweise
1	1546	100.00%	234	100.00%	- (Direkte Anfrage)

Top 1 von 1 Anwenderprogramme

#	Anfragen		kb		Besuche		Anwenderprogramm
1	1546	100.00%	10699	100.00%	234	100.00%	-

Anfragen aus Laendern in Monat Maerz 2004



Top 1 von 1 Ländern

#	Anfragen		Dateien		kb		Besuche		Land
1	1543	99.81%	1271	99.76%	10684	99.85%	231	100.00%	Unbekannte Adressen

100% MicroSoft free!

Stone Steps Webalizer (v3.10.2.5)