NAME : J. P. SAI SRAVANTHI
CLASS : III CSE 'A'
ROLL NO: 220701902

S.NO	DATE	TITLE	REMARKS
1.	13.7.24	Study of various Network commands used in Linux and Windows.	9
2.	27.7.24	Study of Network cables.	10
3.	30.7.24	Study of CISCO Packet Tracer.	10
4.	17.8.24	Set up and configure a LAN using a Switch and Ethernet cable in your lab	
5.	17.8.24	Experiments on Packet Capture Tool: Wireshark.	
6.	6.9.24	Error Correction at Data Link Layer.	
7.	18.9.24	Flow Control at Data Link Layer.	
8. a	20.10.24	a) Simulation of Virtual LAN configuration	
9. 8b	20.10.	b) Configuration of wireless LAN	
	~~20.10~~ 24.10.24	Simulation and implementation of subnetting	
10. a	26.10	Internet working with router	
b)	26.10	Internetwork with wireless router DHCP server and cloud.	
11.	29.10	a) Simulation of Static Routing configuration	
	29.10	b) Simulate RIP using CISCO Packet Tracer	
12.	31.10	a) Implementation of client server connection using TCP	
	31.10	b) Implementation of chat server.	
13.	31.10	Implement ping program	
14.	31.10	Code using Raw socket to implement packet sniffing	
15.	1.12.24	Analyse using webalyser.	

completed

19/12

13.7.24 EXPERIMENT - 1

AIM: To study various Network commands used in Linux and Windows.

Basic Networking Commands:

1. arp -a :

OUTPUT:

Interface : 172.16.75.51 --- 0x19

Internet Address	Physical Address	Type
172.16.72.1	7c-5a-1c-cf-be-41	dynamic
172.16.72.133	4c-ae-a3-65-97-f3	dynamic
172.16.72.195	4c-ae-a3-64-fc-50	dynamic
224.0.0.2	01-00-5e-00-00-02	static

2. hostname :

OUTPUT:

DESKTOP-CO1BH7D

3. ipconfig /all :

OUTPUT:

Windows IP Configuration

Host Name . . . . . : DESKTOP-CO1BH7D
Primary Dns Suffix . . . . . :
Node Type . . . . . : Hybrid
IP Routing Enabled . . . . . : NO
WINS Proxy Enabled . . . . . : No

Ethernet adapter Ethernet 3 :

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . . . . . :
Description . . . . . : Intel(R) Ethernet Connection (17) I219-LM
Physical Address . . . . . : 20-88-10-86-7A-AC
DHCP Enabled . . . . . : Yes
Autoconfiguration Enabled . . . . . : Yes

Wireless LAN adapter Local Area Connection* 13:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . . . . :
Description . . . . . : Microsoft Wi-Fi Direct Virtual Adapter

Physical Address . . . . . : 4E-82-A9-78-88-49
DHCP Enabled . . . . . : Yes
Autoconfiguration Enabled . . . . . : Yes.

Wireless LAN adapter Local Area Connection * 14:

Media State . . . . . : Media disconnected
Connection specific DNS Suffix . . . :
Description . . . . . : Microsoft Wi-fi Direct Virtual Adapter #6
Physical Address : 42-82-A9-78-8B-49
DHCP Enabled : No
Autoconfiguration Enabled : Yes

Wireless LAN adapter Wi-Fi 8:

Connection specific DNS Suffix . :
Description . . . . . : Realtek RTL8852BE Wifi 6 802.11ax PCIe Adapter
Physical Address . . . . . : 4C-82-A9-78-8B-49
DHCP Enabled . . . . . : No
Autoconfiguration Enabled . . . : Yes
Link-local IPv6 address . . . . . : fe80::f196:9925:afba:9f58:25 (Preferred)
IPv4 Address . . . . . : 172.16.75.51 (Preferred)
Subnet Mask . . . . . : 255.255.248.0
Default Gateway . . . . . : 172.16.72.1
DHCPv6 IAID : 374112937
DHCPv6 Client DUID . . . . . : 00-01-00-01-2D-61-C9-B9-74-5D-22-E8-59-3D
DNS Servers . . . . . : 172.16.72.1
NetBIOS over Tcpip . . . . . : Enabled

4. a. nbtstat -a

OUTPUT:

Displays protocol statistics and current TCP/IP connections using NBT (NetBIOS over TCP/IP).

NBTSTAT [[-a RemoteName][-A IP address][-c][-n][-r][-R][-RR][-s][-S][interval]]

-a (adapter status) Lists remote machine's name table given its name.

-A (adapter status) Lists remote machine's name table given its IP address

5. netstat :

OUTPUT:

Active Connections

Proto	Local Address	Foreign Address	State
TCP	127.0.0.1:49678	DESKTOP-CO1BH7O:49679	ESTABLISHED
TCP	127.0.0.1:49679	DESKTOP-CO1BH7O:49678	ESTABLISHED
TCP	172.16.75.51:49945	rio09s03-in-f3:https	TIME_WAIT

⋮

6. nslookup :

OUTPUT:

Default sower : Unknown
Address : 172.16.72.1
> www.google.com
Server: unknown
Address : 172.16.72.1
Non authoritative answer:
Name: www.google.com
Addresses : 2404:6800:4007:81e::2004
142.250.183.228

7. pathping :

OUTPUT :

Usage : pathping [-g host list] [-h maximum hops] [-i address] [-n]
[-p period] [-q num queries] [-w timeout]
[-4][-6] target_name

Options

-g host list Loose source route along host-list.
-h maximum hops Maximum number of hops to search for target.
-i address Use the specified source address

⋮

8. ping

OUTPUT:

Usage : ping [-t] [-a][-n count] [-l size] [-f] [-i TTL] [-v TOS]
[-r count] [-s count] [[-j host list | [-k host-list]]
[-w timeout] [-R] [-S srcaddr] [-c compartment] [-p]
[-4] [-6] target-name.

Options

-t ping the specified host until stopped

-a Resolve addresses to hostname

-n count number of echo requests to send.

⋮

9. route :

OUTPUT:

ROUTE [-f][-p][-4|-6] command [destination]
[MASK netmask][gateway][METRIC metric][IF interface]

-f clears the routing tables of all gateway entries

-p when used with ADD command makes a route persistant across boots of the system.

-4 Force using IPv4

-6 Force using IPv6

Linux Networking Commands.

1. ip :

ip <options> <object> <command>

#ip address show

1. lo : <LOOPBACK> mtu 65536 noqueue state DOWN group default qlen 100.

2. enp520 : <BROADCAST, MULTICAST-UP> mtu 1500 qdisc fq_model state UP group default qlen 100.

#ip address add 192.168.1.254/24 dev ~~eth~~ enp2s0 (enp503)
(assign an IP to an interface)

#ip address del 192.168.1.254/24 dev lo [enp503]
(after the status of interface by bringing eth0 online).

#ip link set eth0 up
after the status of interface by bringing interface eth0 online.

#ip link set eth0 down
after the status of interface eth0 online

ip link eth0 promisc on
after the status of interface by enabling promiscous mode for eth0.

ip route add default via 192.168.1.254 dev enp230:
ip address add 192.168.1.254 dev enp260.
Add a default route via the local gateway 192.168.1.254 that can be reached on device enp2s0.

ip route add 192.168.1.0/24 via 192.168.1.254
add route to 192.168.1.0/24 via the gateway at 192.168.1.254.

ip route add 192.168.1.01/24 dev enp2s0:
add a route to 192.168.1.01/24 that can be reached on

ip route delete 192.168.1.0124 via 192.168.1.254
to delete route for 192.168.1.0124 via the gateway at 192.168.1.254.

Display: → [ ] # ip route get 10.10.1.4
10.10.1.4 dev ens160 SRC 192.168.1.254
usd 1000
cache.

2. ifconfig:
ens160: flags = 4113 <UP, BROADCAST, RUNNING, MULTICAST>
mtn 1500
inet 192.168.22.128 netmask 255.255.255.0 broadcast 192.168.22.255

3. mtn:
mtn <options> hostname IIP

mtn google.com
Keys: Help Display mode restart statistics order of fields quit

	Packets			Pings			
Host	Loss %	snt	Last	Avg	Best	Wrst	StDev
- gateway	0.0%	163	1.2	1.0	0.3	6.1	0.6

mtr -g google.com

-F, --filename FILE read hostname(s) from a file

-4 use IPV4 only

-6 use IP46 only

-n, --nap use UDP instead of ICMP echo.

mtr -b google.com

Keys: Help Display mode Restart Statistics Order of fields

	Packets			Pings			
Host	Loss %	sent	Last	Avg	Best	Wors	stDev
-gateway (	0.0%	391	1	1.1	0.2	10.2	0.8

mtr -c google.com

Keys: Help Display Mode Restart Statistics Order of fields quit

	Packets			Pings			
Host	Loss %	sent	Last	Avg	Best	words	stDev
-gateway	0.0%	4	0.7	7	0.4	1.3	0.3

4. tcpdump:

dnf install -y tcpdump

to install tcpdump

tcpdump -D

1. ens140 [Up, Running, Connected]
2. any (Pseudo-device that captures on all interfaces) [Up, Running]

tcpdump -i eth0

dropped privs to tcpdump

tcpdump: verbose output suppressed, use -v[v]... for full protocol decode.

tcpdump -i eth0 -c 10

10 packets captured

20 packets received by filter

0 packets dropped by kernel

tcpdump -i eth0 -c 10 host 8.8.8.8
dropped privis to tcpdump
tcpdump: verbose output suppressed, use -v[v]
for full protocol decode listening on 10 unit. type
EN10MB (ethernet) snapshot length 262144 bytes.

tcpdump -i eth0 src host 8.8.8.8
dropped privs to tcpdump
tcpdump: verbose output suppressed use -v[v].
for full protocol decode listening on 10.

tcpdump -i eth0 dst host 8.8.8.8
dropped pairs to tcpdump
tcpdump: verbose output suppressed, use -v[v].
for full protocol decode listening on 10.

tcpdump -i eth0 net 10.1.0.0 mask 255.255.255.0
to capture traffic and to form a specific
network using the command.

tcpdump -i eth0 net 10.1.0.0124
to capture traffic and to form a specific network
using the command.

tcpdump -i eth0 port 53
to capture only DNS port 53 traffic

tcpdump -i eth0 host 8.8.8.8 and port 53.
this is for a specific host.

tcpdump -i eth0 -c 10 host www.google.com and port 443
to capture only HTTPS traffic

tcpdump -i eth0 port not 53 and not 25
to capture all port except 53 and 25.

5. ping:

usage: ping [-t] [-a] [-n count] [-l size] [-f] [-i][-L]
[-v TOS] [-r count] [-s count] [[-]host-list][-k host-list]
[-w timeout] [-R] [-S srcaddr] [-c compartment]
[-p] [-4] [-6] target_name.

~~#pinggk~~

ping google.com

PING google.com (216.58.200.142) 56(84) bytes of data
64 bytes from maa05s10-in-f14.1e100.net (216.58.200.142)
: icmp-seq = 40th

ping -c 10 www.google.com
PING google.com (142.250.193.100) 56(84) bytes of data
-- google.com ping statistics --
10 pckts transmitted, 0 received, +10 errors, 100% packet
10 ms, time 9197 ms pipe 3.

Configuring an Ethernet connection by using nmcli

1. # nmcli connection show

NAME	UUID	TYPE	DEVICE
wired connection1	95e66490-cc20-3868-81fd-031	ethernet	enp1s0.

2. # nmcli connection modify "wired connection1".
Rename the connection profile

3. # nmcli connection show
Display current settings
Connection.interface-sthname : enp1s0.
connection-autoconnect : yes.
ip4.method : auto
ipV6.method : auto.

4. To configure IP44 settings
#nmcli connection modify "wired connection1" ipV4 method

5. #nmcli connection modify "wired connection
ipV4.method-dns 192.0.2.200 ipV4-dns search example.com

6. #nmcli connection up Internal-LAN
Activate the profile.

RESULT: Thus the various networking commands in Linux and Windows were executed successfully.

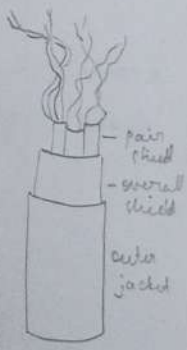
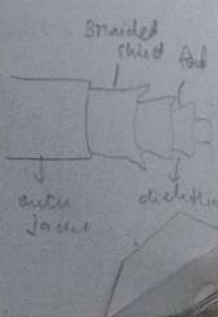
EXPERIMENT 2:

AIM: Study of different types of Network cables.

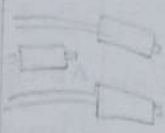
a) understand different types of network cable.

Different types of cables used in networking are:

1. unshielded Twisted pair (UTP) cable.
2. shielded Twisted pair (STP) cable.
3. Coaxial cable.
4. Fibre Optics cable.

Cable Type	Category	Maximum Data Transmission	Advantages Disadvantages	Application	Image.
UTP	Category 3 and 5	10 bps upto 100Mbps	A: • Cheaper in cost • Easy to install due to smaller diameter D: More prone to EMI (Electromagnetic Interference) and noise	10 Base-T Ethernet Fast Ethernet gigabit Ethernet Fast Ethernet gigabit Ethernet.	
	Category 5e	1Gbps.			
STP	category 6, 6a	10 Gbps	A: • Shielded • Faster than UTP • Less susceptible to noise D: Expensive Greater installation effort	gigabit Ethernet 10 G Ethernet 55m widely used in data centres gigabit ethernet 10 G Ethernet 100m.	
SSTP	category 7	10 Gbps.			
Coaxial cable	RG-6 RG-59 RG-11	10 - 100Mbps	A: High bandwidth immune to interface Low cost bandwith Versatile D: Limited distance cost size is bulky	speed of signal is 500M Television n/w High speed internet connections	

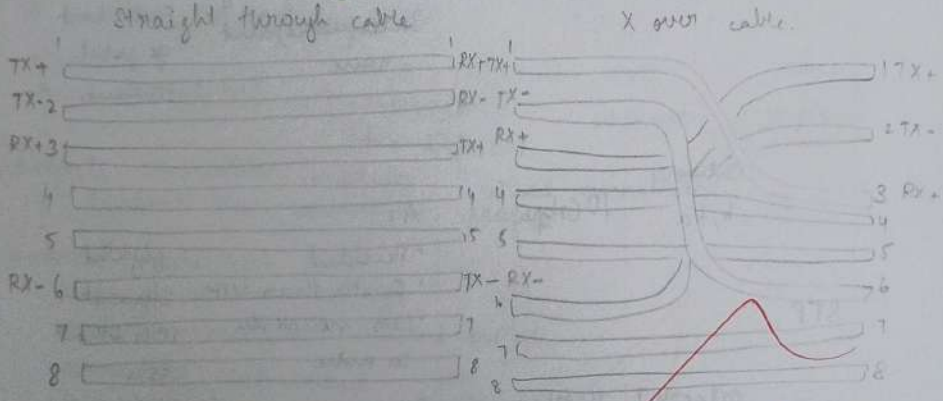
fibre optics cable	single mode multi mode	100 Gbps	A: High speed High bandwidth High security Long distance D: Expensive Requires skilled installation	maximum dist of fibre optics cable is around 100 m



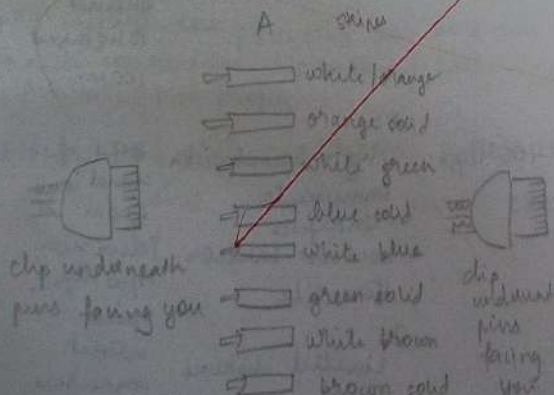
b) Make your own ethernet cross-over cable / straight cable.

Tools and parts needed:

- Ethernet cabling CAT5C 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 pens in the plug and strip and cut the shielding off the cables.
- 2(Two) RJ45 plugs
- optional two plug shields.



Difference between crossover cable and straight cable
Straight through network cable: both sides should be A
cross over cable: one side A one side B



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

Step 2: Next, strip approximately 1.5cm 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 towards you.

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 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.

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).

4/6/18

RESULT: Thus the study of various networking cables was done and executed.

30.7.24 EXPERIMENT - 3

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

To understand the environment of CISCO PACKET TRACKER to design a simple network

INTRODUCTION:

A simulator, as the name suggests simulates network devices and its environment. Packet tracker 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 movie device.
3. It is available for both the Linux and Windows desktop environments
4. Protocols in Packet Tracker are coded to work and behave in the same way as they would on real hardware.

INSTALLING PACKET TRACKER:

To download Packet Tracker go to https://netacad.com and log in with your Cisco Networking Academy credentials. Then click on the Packet Tracker graphic and download the package appropriate for your operating system.

Windows:

Installation is pretty simple. the setup comes in a single file named Packettracer-setup6.0.1.exe. Open file to begin set up accept licence agreement, choose location & start installation.

Linux:

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

```
chmod + x Packet Tracer601_i386_installer_rpm_bin
./Packet Tracer 601_i386_installer_rpm_bin
```

USER INTERFACE OVERVIEW:

The layout is divided. The components are as follows:

1. Menubar: common menu in all ~~op~~ software applications; used to open, save, print, change preferences etc.
2. Main toolbar: provides shortcut icons to menu options that are commonly used (open, save, zoom, undo etc) the right hand side has an icon for entering network information for the current network
3. Logical / Physical workspace tabs - allow you to toggle b/w logical and Physical work areas.
4. Workspace - topologies are created and simulations are displayed.
5. Common tools bar - provides controls for manipulating topologies, such as select, move, place, note, delete etc.
6. Real time / simulation tabs - These tabs are used to toggle b/w real & simulation modes. Buttons are provided to control time & capture packets.
7. Network component box: contains all of n/w & end dev's available with Pckt Racer. divided into
   → Device type selection box: contains device categories
   → Device specific selection box: different device models are displayed.
8. User-created packet box: Users can create highly-customised packets to test their topology from this area and the results are displayed as a list.

b) ANALYSE THE BEHAVIOUR OF NETWORK DEVICES USING CISCO PACKET TRACER:

1. From the network component box, drag
   a) 4 Generic PC's and one HUB
   b) 4 Generic PC's and one Switch.

2. Click on connections
   a) Click on copper straight through cable.
   b) Select one of the PC & connect to HUB. The link LED should glow green. ~~so~~ Repeat
   c) Similarly do that with 4 PCs and switch

3. Click on the PC's connected to hub, go to desktop tab. click on IP configuration, and enter IP address and subnet mask. The default getaway and DNS server info" is not needed as there are only two end devices.

   Click on PDU (msg icon)
   a) Drag & drop it on one of the PC and the drop on another PC connected to HUB.

4. Observe the flow of PDU from source to destination by selecting Realtime.
5. Repeat for switch.
6. Observe and write observation.

6/8

RESULT: Thus the CISCO Packet Tracer was successfully installed and configured.

Student's Observation:

1.1. Which command is used to find the reachability of a host machine from your device.

ping <hostname or IP address>

1.2. Which command will be given the details of hops taken by a packet to reach its destination?

traceroute <hostname or IP address>

1.3. Which command to displays the IP config of your machine

ifconfig (Linux/unix, older)
ipaddr
ip config

1.4. Which command displays the TCP port states in your machine

netstat -tuln
ss -tuln

1.5. Write the modify the ipconfiguration in a unix machine

temporary:
wing config
wing ip
sudo ip addr add 192.168.1.10/24 dev eth0
sudo ip route add default via 192.168.1.1

2.1. What is the difference between cross cable and straight cable

Straight cable	Cross cable
All the wires are on the same order on both ends of the cable.	The transmit & receive wires are swapped on each end.
Used to connect diff types of devices	used to connect similar devices directly.

2.2. Which type of cable is used to connect a router/switch to your PC?

Straight cable: PC → router/switch

2.3 Which cable is used to connect two PC?

Cross cable: PC → PC

2.4. Find out the category of twisted pair cable your lab to connect the PC to the network socket

Cat 6 (category 6): supports up to 10Gbps for short distances.

2.5. Write down your understanding, challenges faced and output received while making a twisted pair cross straight cable.

- Straight cable:
  - ⊙ used to connect diff types of network devices.
  - ⊙ Both ends have same wiring order.
- cross table:
  - ⊙ used to connect similar types of network devices directly
  - ⊙ one end follows EIA/EIA-568-B) and other TIA/EIA-568-A standard.

Challenges.
- crimping
- wire order
- Testing.

Output received
Successfully made cable.

3.1. From your observation write down the behaviour of switch and HUB in terms of forwarding the packets received by them.

HUB:

Broadcasting:

A hub is a basic networking device that operates at the physical layer.

When a HUB receives a packet of one of its ports it broadcasts the packet to all other ports, regardless of destination

This behaviour results in all devices connected to the HUB receiving the packets even if they are not the intended recipient

Collision Domain:

All devices connected to HUB are in the same collision domains.

SWITCH:

Intelligent Forwarding :

A switch operates at the data link layer of OSI when a switch receives a packet it examines the MAC address of the destination devices.

It forwards the packet only to the port associates with the destination MAC address, thus reduce unnecessary traffic and improving network efficiency.

Collision Domain

Each port on a switch represents a separate collision domain.

This isolation significantly reduces the likelihood of collision compared to a hub as packets are only forwarded to the intended recipients.

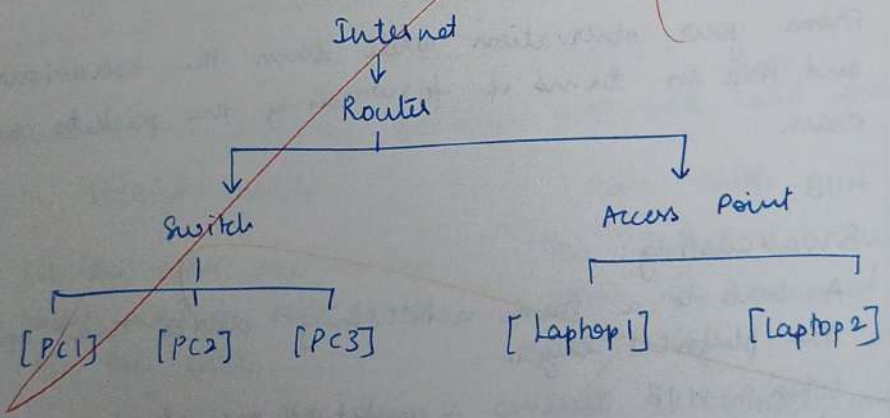
3.2. Find out the network topology implemented in your college and draw & label that topology.

→ STAR TOPOLOGY

All devices are connected to central Switch & Hub

most commonly & widely used.

Simple and efficient



EXPERIMENT - 4

AIM: Setup and configure a LAN (Local Area Network) using a switch and Ethernet cables in your lab

What is 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 connect devices to promote collaboration and transfer information between users such as computers, printers, servers. A local area network switch serves as the primary connecting devices.

How to set up a LAN:

Step 1: Plan and Design an appropriate, network topology taking into account network requirement.

Step 2: You can take 4 computers a switch with 8, 16, 24 ports.

Step 3: Connect your computers to network switch an ethernet cable, which is as simple as plugging one end of the ethernet cable.

Step 4: Assign IP address to your PCs.

1. Log on the client computer as administrator.
2. Click Network and internet.
3. Right click local area connection → Go to properties → set internet protocol → click on properties → select ip address option.

Step 5: Configure a network switch

1. Connect your computer to switch: To access the switch's web interface, you will need to connect your computer.
2. Log in to the web interface: Open a web browser and enter the IP address of the switch in the address bar.

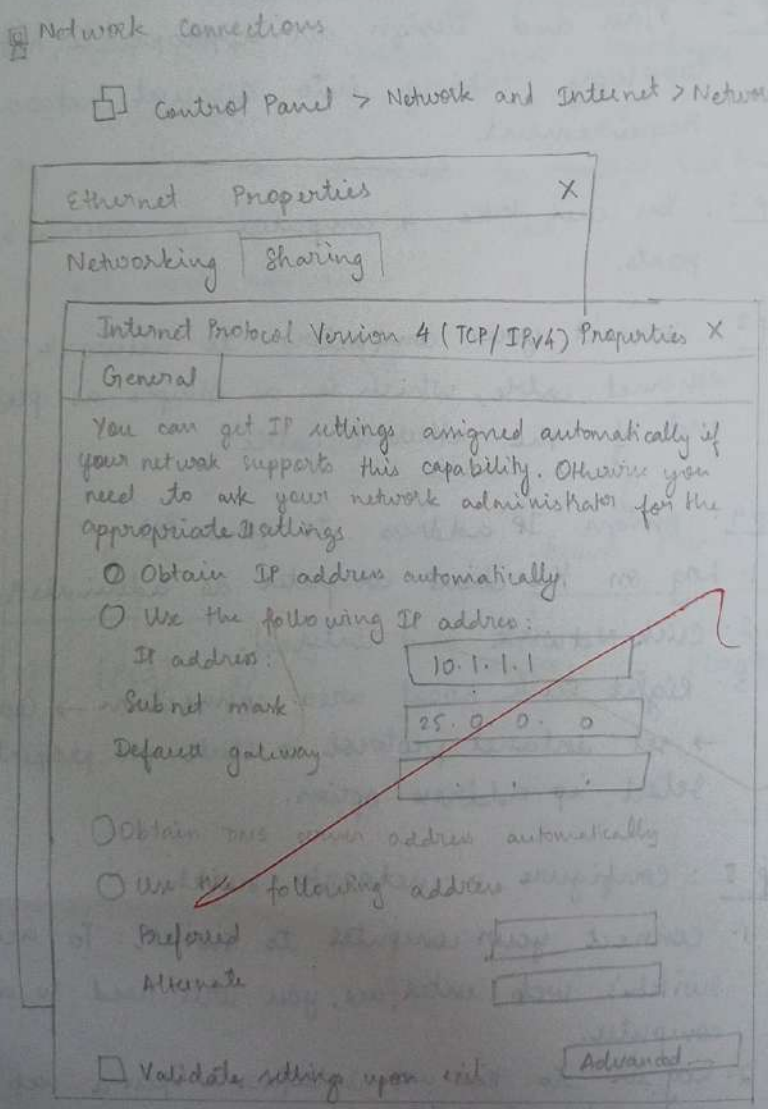
3. Configure basic settings: Once you're logged in, you will be able to configure basic settings for the switch

4. Assign IP address 10.1.1.5 subnet mask 255.0.0.

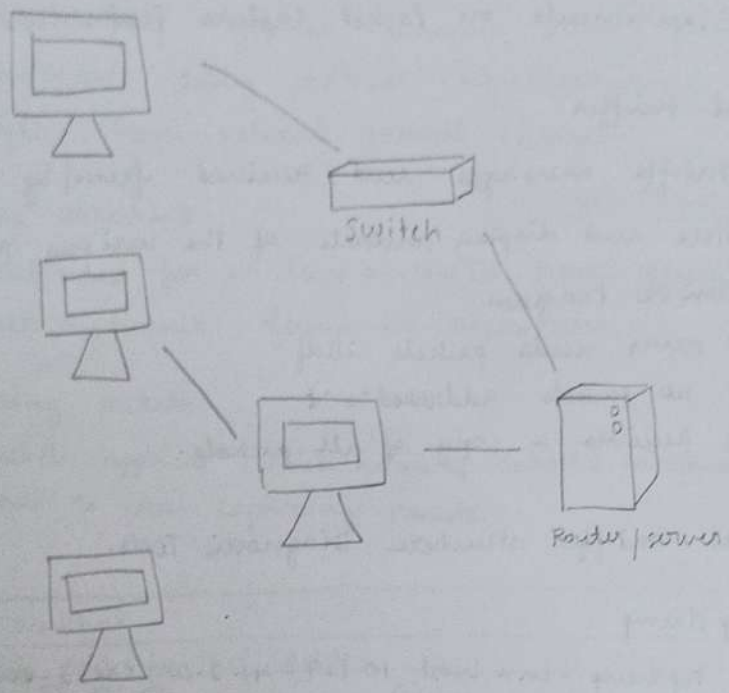
Step 6: Check the connectivity between switch and other machine by using ping command in the command prompt device.

Step 7: Select a folder → go to properties → click sharing tab → share it with everyone on the same LAN.

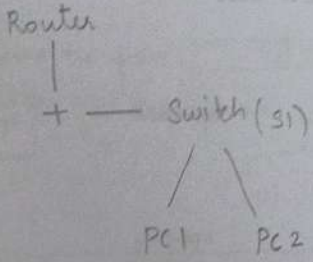
Step 8: Try to access the shared folder from other computers of the network.



4.1.



or



IP configuration:

PC1: IP: 10.1.1.1, subnet mask : 255.0.0.0

PC2: IP: 10.1.1.2 subnet mask : 25.0.0.0

PC3: IP: 10.1.1.3 subnet mask : 25.0.0.0.

PC4: IP: 10.1.1.4 subnet mask : 25.0.0.0.

RESULT: The LAN set up was configured successfully. using a switch and Ethernet cable is done and executed successfully.

17/8/24

EXPERIMENT - 5

AIM: Experiments on Packet Capture Tool: Wireshark

Packet Sniffer

→ Sniffs messages sent / received from/ by your comp
→ Store and display contents of the various protocol fields
→ Passive Program
  → never sends packets itself
  → no packets addressed to it
  → receives a copy of all packets.

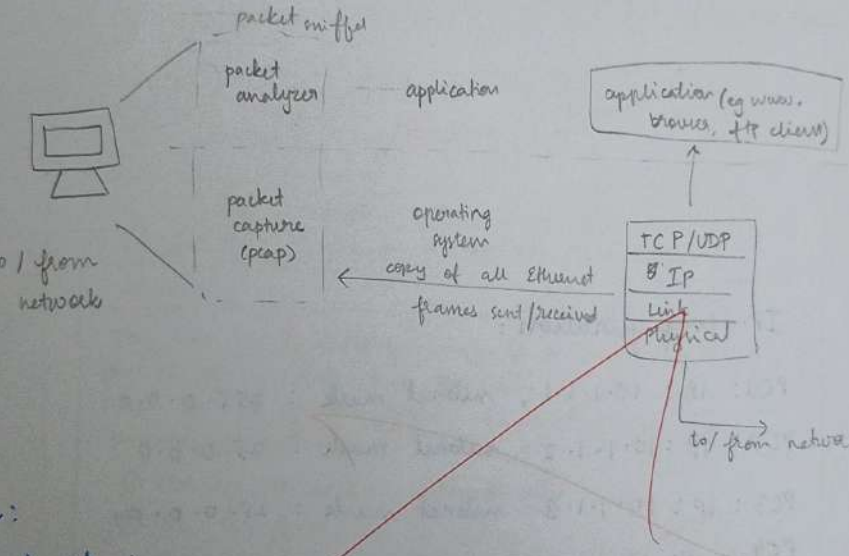
Packet sniffer structure Diagnostic Tools.

→ Tcp dump

```
tcpdump -cnx host 10.129.41.2.cw exe.3.out
```

→ Wireshark

```
- wireshark -r exe3.out
```



Desc:

Wireshark :

→ network analysis tool formerly known as Ethereal. captures packets in real time and display them in human readable format.
→ includes filters, colour coding
→ can be used to inspect a suspicious program's network traffic

Uses:

→ captures network traffic
→ Decode packet protocols using dissectors
→ define filters.
→ smart statistics
→ Analyse problems
→ Interactively browse traffic

Used for:

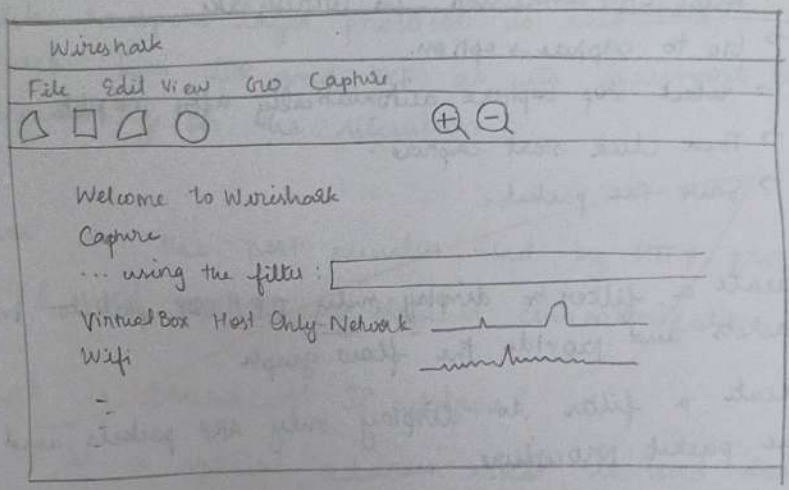
→ N/w administration: Troubleshoot n/w problems.
→ N/w security engineers: examine security problems.
→ Developers: debug protocol implementations.
→ People: learn network protocol internals.

Getting Wireshark:

downloaded for Windows or MacOs from official website.
Linux and Unix: found in repositories.

Capturing packets:

→ launch applicatn, click name of network interface under capture to start capturing packets.



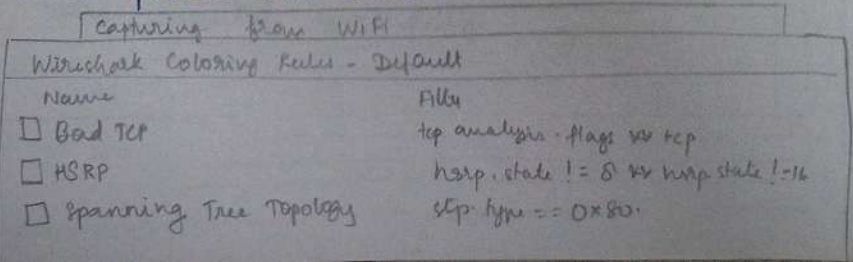
The Packet list pane: displays the current capture file as packets. The packet list pane each line in the packet lists corresponds to one packet in the capture file.

The Packet Details Pane: The selected protocols and protocol fields in the 'packet list' pane is shown in this pane.

The Packet Bytes Pane: data of current packet is shown in hexdump style.

Colour coding:

light purple - TCP traffic, light blue is UDP and black - packets with errors.



Sample captures:

Wireshark's wiki contains a page of sample captures files that you can load and inspect. Click File > open to browse for your downloaded file to open one.

Click file > save to save your captured packets.

~~Filtering Packets~~:

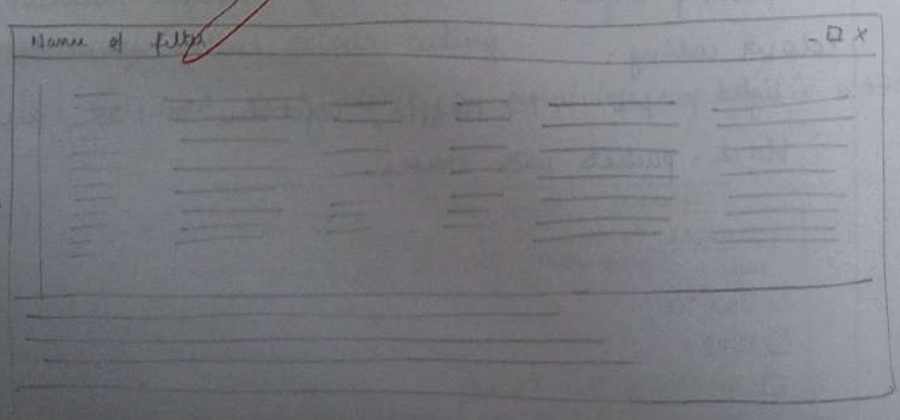
Capturing and Analysing Packets using Wireshark Tool

To filter, capture, view, packets in Wireshark Tool capture 100 pckts from the ethernet : IEEE 802.3 CAN interface & Save it.

Procedure:

→ Select LAN connection in Wireshark
→ Go to capture → option.
→ Select stop capture automatically after 100 pckts
→ Then click start capture.
→ Save the packet.

1. Create a filter to display only TCP / UDP packets. Inspect the packets and provide the flow graph
2. Create a filter to display only ARP packets and inspect the packet procedure.
3. Create a filter to display only DNS packets and provide the flow graph.
4. Create a filter to display only HTTP packets and inspect the packets.
5. Create a filter to display only RPI ICMP packets and inspect the packets.
6. Create a filter to display only DHCP packets and inspect the packets.



5.1. What is promiscuous mode?

↳ is a configuration n/w interface card (NIC) or network device that allows it to capture and process all packets on the network segment to which it attached rather than only the packets addressed to it

5.2. Does ARP packets have transport layer header? Explain.

No, ARP packets do not have a transport layer header. ARP is a protocol used to map an IP address to a physical MAC address on a local network. Its' part of the n/w layer of the OSI Model,

5.3. Which transport layer protocol is used in DNS?

↳ uses both UDP and TCP as its transport layer protocols depending on the situation.

5.4. What is the port number used by HTTP protocol?

HTTP used port 80 by default communication.

5.5. What is broadcast IP address?

↳ is a special address used to send data to all devices on a particular n/w segment.

3/18

RESULT: Experiments on packet capture tool with Wireshark were executed successfully.

EXPERIMENT - 6

AIM: To 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.

$2^r \geq d + r + 1$

Error Correction:

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.

Algorithm:

① finding r

d = len(data)

~~for (i=1; i<=d; i++)~~ d = len(data)

for r in range(1, d):

if 2**r >= d+r+1:

break.

② finding redundant bit:

```
d= len(data)
r= finding_r(data)
j=0
k=1
ham=[];
for i in range(1, m+r+1):
    if i==2**j
        ham.append(0)
        j+=1
    else
        ham.append(data[-k])
        k+=1
ham.reverse()
return ham
```

③ Calculate parity values:

```
cal_par(ham):
    d = len(ham)
    r = find_r(ham)
    for i in range(r):
        p_pos = 2**i
        p_sum = 0          ——— even parity.
        for j in range(1, d+1):
            if j & p_pos == p_pos:
                p_sum ^= ham[-j]
    ham[-p_pos] = p_sum
    return ham
```

④ generate ham_code:

```
ham = find_red_bit(ham)
ham = cal_par(ham)
return ham
```

⑤ detect error (introduced statically / dynamically)

```
d = len(ham)
r = find_red_bit(ham)
error = 0
for i in range(r):
    p_pos = 2**i
    p_sum = 0
    for j in range(1, r+1):
        if j & p_pos == p_pos:
            p_sum ^= ham[-j]
    if p_sum != 0
        error += p_pos
return error
```

⑥ correct error (hamm)

```
    error = detect_error(hamm)
    if error != 0
        hamm[-error] ^= 1
        print(f"error at {error}")
    else
        print("No error")
    return hamm
```

⑦ convert string to binary. (input)

```
return ''.join(format(ord(char), '08b') for
                char in input)
```

CODE :

```
def string_to_bits(s):
    return ''.join(format(ord(c), '08b') for c in s)
def calculate_redundant_bits(m):
    r = 0
    while (2**r) < (m + r + 1):
        r += 1
    return r
def insert_redundant_bits(data_bits):
    m = len(data_bits)
    r = calculate_redundant_bits(m)
    encoded_bits = ['0'] * (m + r)
    j = 0
    for i in range(1, len(encoded_bits) + 1):
        if (i & (i-1)) == 0:
            continue
        encoded_bits[i-1] = data_bits[j]
        j += 1
    for i in range(r):
        parity_pos = 2**i
        parity = 0
        for j in range(parity_pos - 1, len(encoded_bits),
                       2 * parity_pos):
            for k in range(parity_pos):
                if j + k < len(encoded_bits):
                    parity ^= int(encoded_bits[j+k])
        encoded_bits[parity_pos - 1] = str(parity)
    return ''.join(encoded_bits)
def detect_and_correct(encoded_bits):
    r = 0
    while (2**r) <= len(encoded_bits):
        r += 1
    error_pos = 0
    for i in range(r):
        parity_pos = 2**i
        parity = 0
        for j in range(parity_pos - 1, len(encoded_bits), 2*parity_pos):
            for k in range(parity_pos):
                if j + k < len(encoded_bits):
                    parity ^= int(encoded_bits[j+k])
```

```
    if parity != 0:
        error_pos += parity_pos
    return error_pos

def main():
    input_str = input("Enter string to encode: ")
    data_bits = string_to_bits(input_str)
    print("Original data bits: {}".format(data_bits))
    r = calculate_redundant_bits(len(data_bits))
    print("Number of redundant bits needed: {}".format(r))
    encoded_bits = insert_redundant_bits(data_bits)
    print("Encoded bits redundant bits: {}".format(encoded_bits))
    user_input = input("Enter the bit position (1-based index)
        to change (or 0 to skip): ")
    if user_input != '0':
    bit_position = int(user_input) - 1
    if 0 <= bit_position < len(encoded_bits):
        encoded_bits_list = list(encoded_bits)
        encoded_bits_list[bit_position] = '1' if encoded_bits_list[bit_pos
        encoded_bits = ''.join(encoded_bits_list)
        print("Bit at position {} changed. Updated encoded bits:
            {}".format(bit_position+1, encoded_bits))
    else
        print("Invalid bit position entered")

    detected_error_pos = detect_and_correct(encoded_bits)
    if detected_error_pos:
        print("Detected error at position: {}".format(detected
            error_pos))
        encoded_bits_list = list(encoded_bits)
        encoded_bits_list[detected_error_pos-1]
            = '1' if encoded_bits_list[detected_error_pos-1]=='0' else
        print("corrected encoded bits: {}".format(encoded_bits
    else
        print("No errors detected.")
```

```
corrected_data_bits = [encoded_bits[i] for i in range(len(encoded_bits))
                       if (i+1) & (i) != 0]
corrected_data_bits = ''.join(corrected_data_bits)
print("corrected data bits: {}".format(corrected_data_bits))
original_message = ''.join(chr(int(corrected_data_bits[i:i+8],2))
                   for i in range(0, len(corrected_data_bits))
print("original message: {}".format(original_message))
if __name__ == "__main__":
    main()
```

SAMPLE OUTPUT:

```
Enter string to encode: SAI
Original data bits : 01010011010000010100 1001
Number of redundant bits needed: 5 [1, 2, 4, 8, 16]
Encoded bits with redundant bits: [0, 1, 1, 0, 0]
Enter the bit position (1-based index) to change (or 0 to skip): 3
Bit at position 3 changed : Updated : 01111010011010000010100 1001
Detected error at position : 3
Corrected encoded bits: 01011011001101000001010 01001
Corrected data bits: 01010011010000010100 1001
Original message : SAI
```

RESULT:

Thus a python program was written and executed successfully to implement HAMMING CODE

EXPERIMENT-7

AIM: To 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.

ALGORITHM:

Sender:
→ Input from user to get window size
→ Input from user to get message
→ Consider one character per frame
→ Frame with fields: [Frame no, Data]
→ Send frame
→ Wait for acknowledgement from receiver
→ Reader a file called receiver_buffer.
→ Check ACK for ack. number.
→ If ack is expected send new frames, else NACK and resend frames.

Receiver:
→ Reader a file called Sender_Buffer
→ Check frame no.
→ If frame no as expected write appropriate ACK no else write NACK.

CODE:

```
import time
import random

class Frame:
    def __init__(self, frame_no, data):
        self.frame_no = frame_no
        self.data = data
        self.acknowledged = False

def send_frames(frames, window_size):
    print("\n--- Sending Frames ---")
    for i in range(window_size):
        if i < len(frames) and not frames[i].acknowledged:
            print(f"Sent Frame {frames[i].frame_no}: {frames[i].data}")
    print("Frames sent, waiting for acknowledgements...\n")

def received_frames(frames, window_size):
    print("\n--Receiving Frames--")
    for i in range(window_size):
        if i < len(frames) and not frames[i].acknowledged:
            if random.random() < 0.2:
                print(f"Received Frame {frames[i].frame_no}:
                      {frames[i].data} [ERROR]")
                frames[i].acknowledged = False
            else:
                print(f"Received Frame {frames[i].frame_no}: {frames[i].data}
                      [RECEIVED]")
                frames[i].acknowledged = True

def sliding_window_protocol():
    window_size = int(input("Enter window size: "))
    message = input("Enter a message to send: ")
    frames = [Frame(i, message[i]) for i in range(len(message))]
    base = 0
    while base < len(frames):
```

```
        send_frames(frames[base:], window_size)
        time.sleep(2)
        receive_frames(frames[base:], window_size)
        while base < len(frames) and frames[base].acknowledged:
            base += 1
        if base < len(frames):
            print("\nResending unacknowledged frames...\n")
            time.sleep(2)
    print("\nAll frames sent and acknowledged!")

if __name__ == "__main__":
    sliding_window_protocol()
```

Sample Output:

```
Enter window size: 3
Enter a message to send: SHRAVANTHI

---Sending Frames---
Sent Frame 0: S
Sent Frame 1: H
Sent Frame 2: R
Frames sent, waiting for acknowledgements...

---Receiving Frames---
Received Frame 0: S [RECEIVED]
Received Frame 1: H [RECEIVED]
Received Frame 2: R [ERROR]

Resending unacknowledged frames...

---Sending Frames---
Sent Frames 2: R
Sent Frame 3: A
Sent Frame 4: V
Frames sent, waiting for acknowledgements...
```

--- Receiving Frames ---
Received Frame 2 : R [RECEIVED]
Received Frame 3 : A [RECEIVED]
Received Frame 4 : V [RECEIVED]

Resending unacknowledged frames...

--- Sending Frames ---
Sent Frame 5 : A
Sent Frames 6 : N
Sent Frames 7 : T
Frames sent, waiting for acknowledgements...

--Receiving Frames---
Received Frame 5 : A [ERROR]
Received Frame 6 : N [RECEIVED]
Received Frame 7 : T [RECEIVED]

Resending unacknowledged frames...

--- Sending Frames ---
Sent Frame 5 : A
Frames sent, waiting for acknowledgments

--- Receiving Frames ---
Received Frame 5 : A [RECEIVED]

Resending unacknowledged frames...

--- Sending Frames ---
Sent Frames 8 : H
Sent Frames 9 : I
Frames sent, waiting for acknowledgements...

--- Receiving Frames ---
Received Frame 8 : H [RECENED]
Received Frame 9 : I [RECEIVED]

All frames sent and acknowledgement.

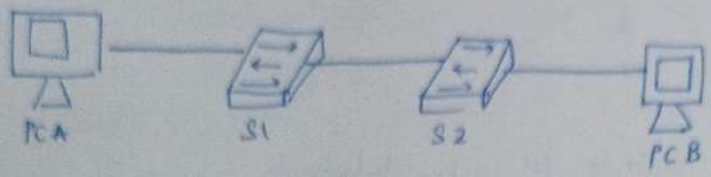
22/10/21.

RESULT : Thus a python program was written and executed to implement sliding Window Protocol successfully

EXPERIMENT-8

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

Packet Tracer - configure VLAN's and Tracking - Physical mode topology



Addressing Table

Device	Interface	IP Address	Subnet Mask	Default Gateway
S1	VLAN1	172.168.1.11	255.255.255.0	N/A
S2	VLAN1	172.168.1.12	255.255.255.0	N/A
PC-A	NIC	192.168.10.3	255.255.255.0	192.168.10.4
PC-B	NIC	192.168.10.4	255.255.255.0	192.168.10.4

Instructions

Part 1: Build the n/w & configure.

Step 1: Build the N/W as shown in topology
Attach the devices as shown in the topology diagram and cable as necessary.

Step 2: Configure basic settings
From desktop IOS on each PC use the terminal to console to each switch
Enter configuration mode.

Part 2: Create VLANs and assign switch ports

Step 1: Create VLANs on the switches
a) Create VLANs on the switches
b) Create VLANS on S1
c) Create VLANS on S2
c) Issue the show VLAN brief command

Step 2: Assign VLANs to correct switch interface
Assigns VLANs to interfaces on S1
Assign PC-A to the operation VLAN S1 (config) interface fa 0/6.

Part 3: Maintain VLAN Root Assignments
Remove a VLAN ID from VLAN db.

a) Add VLAN 30 to interface Fo/24 without using the global VLAN command.

b) Verify the new VLAN

c) Use the no vlan 30 command to remove VLAN 30

Part 4: configure an 802.1Q Trunk between switches.

Step 1: Use DTP to initiate trunking on Fo/1

To default DTP mode of a2960 switch port

Step 2: Manually configure trunk interface Fo/1

a) On interface Fo/1 change the switch apart mode

Open configuration using down

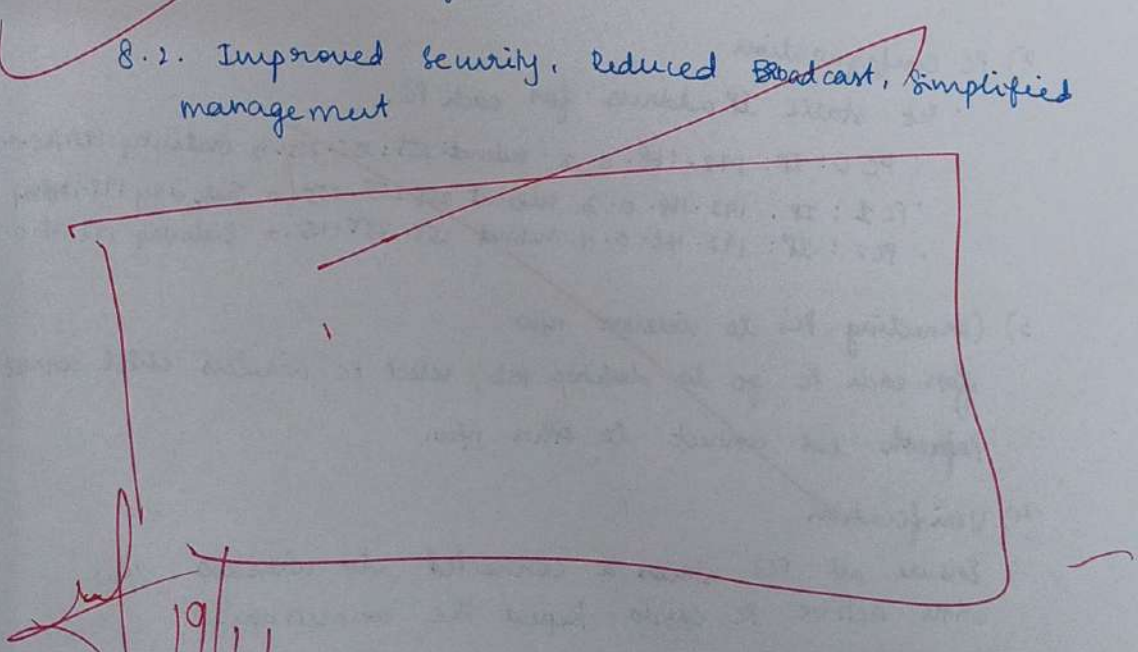
S1(config) : interface fo/1

S1(config-if) : switch mode trunk

Student's Observation

8.1. Router or Layer 3 switch.

8.2. Improved security, Reduced Broadcast, Simplified management

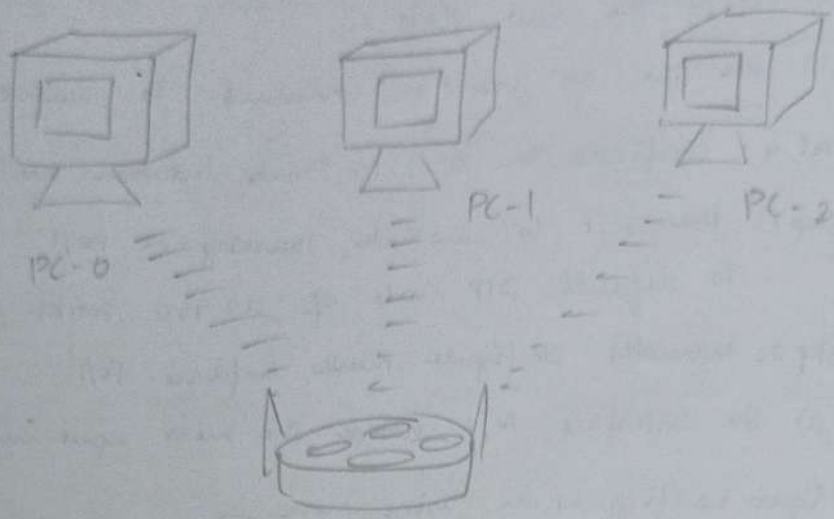


19/2/

RESULT: Thus the simulation Virtual LAN config using CISCO packet Tracer Simulation successfully created & verified

AIM: b) Configuration of wireless LAN using CISCO Packet Tracer

Design a topology with 3 PCs



1) Router Configuration:
   - Access the administration tab on the wireless router.
   - Set the username and passwd to admin
   - Navigate to wireless and change SSID to mother n/w
   - Set security mode to WEP & config key.

2) PC configuration
   - Set static IP address for each PC.
     - PC 0: IP: 192.168.0.2 subnet: 255.255.255.0 Gateway 192.168.0
     - PC 1: IP: 192.168.0.3 subnet 255.255.255.0 Gateway 192.168.0
     - PC 2: IP: 192.168.0.4 subnet 255.255.255.0 Gateway 192.168.0

3) Connecting PCs to wireless n/w
   For each PC go to desktop tab, select PC wireless click connect
   Refresh list connect to above n/w.

4) Verification
   Ensure all PCs shows a connected to wireless n/w with active PC cards. Repeat the connections.

Student Observation:

1) What is SSD of a wireless router

The SSD Service set identifier is name of a wireless n/w that allows devices to identify and connect to it.

2) What is a security key in windows router?

The security key (or wifi password) is a code used to authenticate user and encrypt data transmitted over the wireless n/w ensuring secure access.

3) Configure a simple wireless LAN in your LAB using a real access point and write down the configuration.

To configure

- Connect to the access point
- Access the configuration page
- Login / set SSID / choose security type
- set security key, save settings.

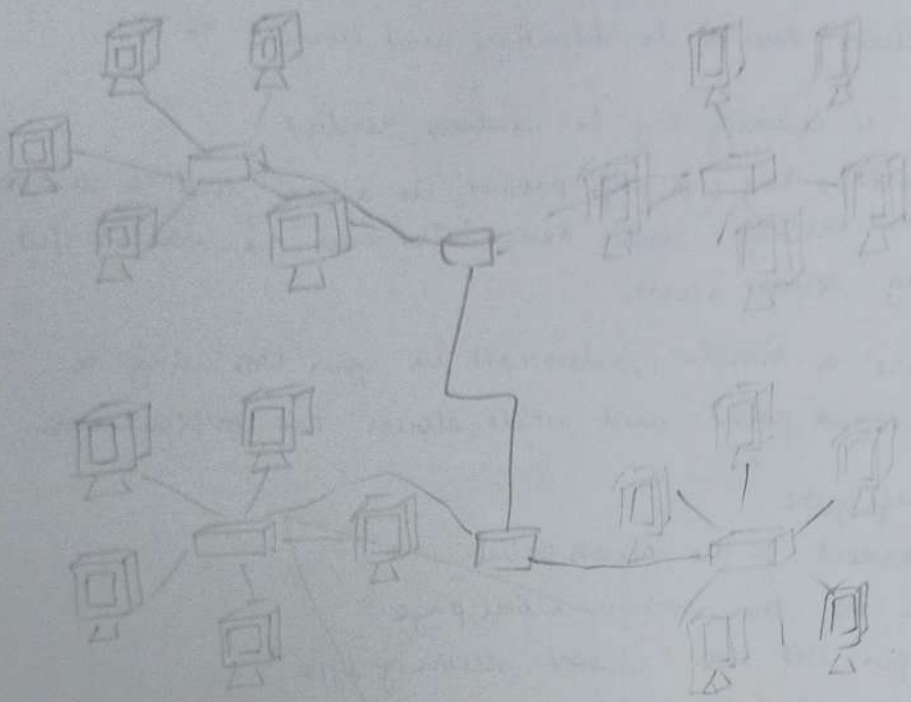
19/11

RESULT:

Thus the configuration for a wireless lan using CISCO pkt tracer was done successfully

EXPERIMENT - 9

AIM: Implementation of Subnetting in CISCO packet Tracer Simulator



IP addressing

Router R1

gigabit ethernet 0/0: 192.168.1.1

Gigabit ethernet 0/1: 192.168.2.1

switch s1

fast-ethernet 0/1 : 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

fast ethernet 0/2 : 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:

fast ethernet 0/0 192.168.3.1

fast ethernet 0/1 192.168.4.1

switch s2

fast ethernet 11: 192.168.30127

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

fast ethernet 0/2 : 192.168.4.0127

PC 1 : 192.168.4.11

PC 2 : 192.168.4.12

PC3 : 192.168.4.13

PC4 : 192.168.4.14

PC5 : 192.168.4.15

classless IP subnetting allows for efficient use of IP address only enabling variable length subnet masks, which helps divide an IP address space into smaller subnet for better management

Create a n/w Topology :

- open packet Tracer, select "new", then "Network" & "Generic" to start a blank topology.

Add devices :

- Drag and drop routers, switches, PCs from devices menu onto topology.
- Connect devices using enable tool

Subnetting

- For the n/w address 192.168.1.0/24 use /27 subnet mask to provide at least 5 addresses for end device the switches and router
- The setup create 8 subnet each with usable host address.

Network Configuration

Router Configuration

Use the CLI to set up the router interface
fast ethernet 0/0 : connect to switch
fast ethernet 0/1 : connect to PC

Switch configuration

Access CLI
enable
configure terminal
interface fast ethernet 0/1
switchport mode access
exit
interface fastethernet 0/2
switchport mode access
exit

PC configuration

- IP address Default gateway
Subnet mask DNS server

Ensure IP address & subnet mask match subnet of routers fast ethernet 0/1

Gigabit Ethernet configuration

Use the CLI to configure the Gigabit ethernet 0/0 interface with desired IP address and subnet mask then enable the interface.

Student Observation:

a) Write down your understanding of subnetting:
Subnetting is a process of diving a larger network into smaller managable sub network. (subnets)
Each subnet operates within a defined range of IP address

b) What is advantage of implementing subnetting within a network?

Improved network — Efficient IP address management
Enhanced Security — Simplified Network management

c) Find out whether subnetting is implemented your college if yes, draw and list down the subnet used.

d) To determine if subnetting is implemented in your college check with IT dept or n/w administration

subnet 1: 192.168.0.0124
subnet 2: 192.168.1.0.124
subnet 3: 192.168.2.0124

19/11

RESULT: Implementing subnet in CISCO packet Race simulation successfully & executed.

EXPERIMENT - 10

AIM: Interconnect working with routers in CISCO Packet Tracer Simulator.

Configure the network

Configure router1

1) open router CLI
2) Activate privileged mode by typing enable.
3) Enter global configuration mode with config t.
4) Configure fast Ethernet 0/0.
   - interface fastethernet 0/0
   - ip address 192.168.10.1 255.255.255.0
   - no shutdown.
5) Configure fast Ethernet 0/1:

Configure PCs

Assign IP address to each PC.

PC0 : IP address : 192.168.10.2
Subnet : 255.255.255.0
Gateway : 192.168.10.1

Connecting PC's to Routers

1) PC0 to Router1
2) PC1 to Router2

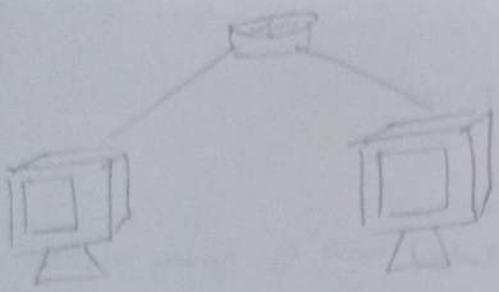
Router configuration Table:

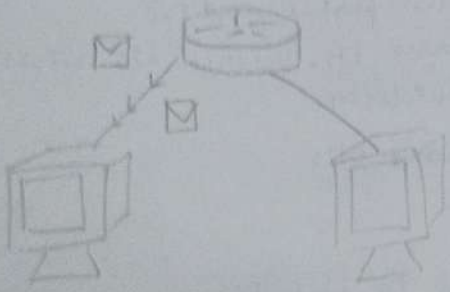
Device Name	IP address	Subnet mask
Router1	192.168.10.1	255.255.255.0
	192.168.20.1	255.255.255.0

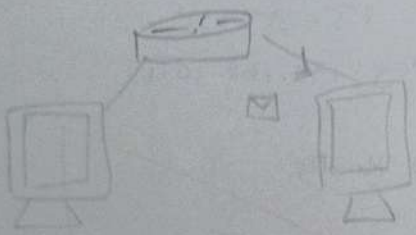
PC configuration Table:

Device Name	IP address	subnet mask	Category
PC0	192.168.10.2	255.255.255.0	192.168.10.1
PC1	192.168.10.2	255.255.255.0	192.168.20.1

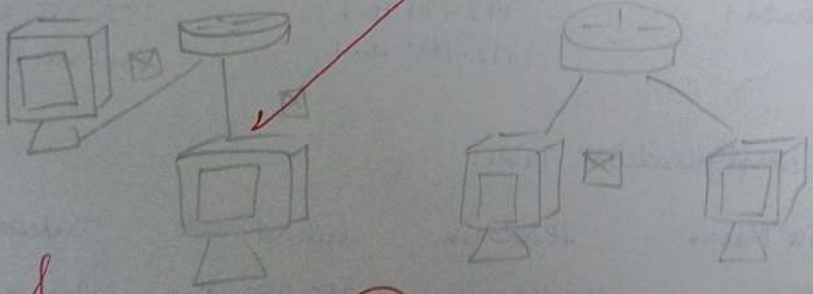
Testing: To verify connectivity perform PDU transfer from PC0-PC1 if successfully the n/w is correctly configured.







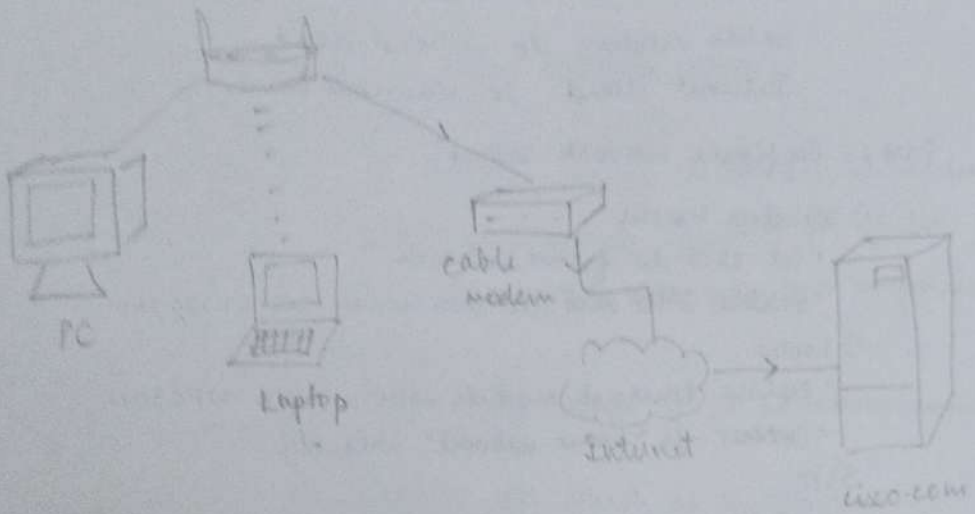
Acknowledgement PC1 to PC0



RESULT: Thus the configuration of Internet using working router was done successfully

b) Date

AIM: Design and configure an internetwork using wireless router DHCP server and internal cloud.



Addressing Table

Device	Interface	IP address	Subnet mask	Default gateway
PC	ethernet	DHCP		192.168.0.1
wireless router	LAN	192.168.0.1	255.255.255.0	
wireless router	Internet	DHCP		
cisco.com server	ethernet	208.67.220.220	255.255.255.0	
Laptop	wireless0	DHCP		

Part 1: Build a simple network

1) Launch Packet Tracer

2) Add Devices
   - Select device from Device Type > Device specific selection tons
   - Place devices in workspace

3) Rename Devices
   - Click on each device and change the display name in the config tab.

4) Connect Devices

- Using copper straight wire through and coaxial cable connect
  - PC wireless router
  - wireless router to cable modem
  - cable modem to internet cloud
  - Internet cloud to cisco.com server.

Part 2: Configure network Devices

1) Wireless router
   - Set SSID to frame network
   - Enable DHCP and set DNS server 208.67.220.220.0

2) Laptop
   - Replace Ethernet module with wireless WPC30N
   - Connect to "home network" wirelessly.

3) PC
   - Set to receive DHCP configuration for an IPV4 address

4) Internet Cloud
   - Install necessary network module
   - set up coaxial and ethernet connections

5) Cisco.com server.

Enable DHCP service with setting
- Port 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

Enable DNS service with:
- Name: cisco.com
- Type : A Record
- Address : 208.67.220.220

Set Global setting for Static configuration

Configure fast ethernet 0 with a static IP address: 208.67.220.220.

Part 3: Verify ~~one~~ connectivity

Refresh IPV4 setting on PC
- use ip config release and ipconfig /renew to obtain an IP address from DHCP

Test connectivity:

Ping the cisco.com server from PC using the command ping cisco.com Verify successfully Replies

Part 4: Save and close file.

RESULT: Thus the configuration of a internetwork using wireless router and DHCP server and internet cloud was done successfully

Student Observation

① Write down the key features of configuring wireless router and DHCP server.

Wireless Router Configuration

- SSID setup : create a unique network name
- Security Protocol : choose WPA3 or WPA2
- Channel selection : Set optional channels to reduce interference
- Guest network : Configure a separate network for visitors
- Firmware Update : Regularly update for security & performance.

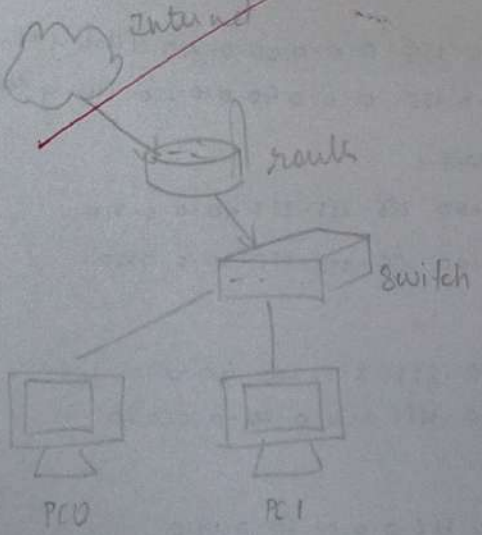
DHCP Server configuration :

- IP address Pool : Define the range of IP addresses available for assignment
- Lease Time : Set duration for which an Ip assigned.
- DHCP options : Configure default gateway, DNS server and other network settings.
- Static IP Reservation : Assign fixed IP to specific MAC address.

② What is the significance of DHCP server in internetworking?

Automatic IP assignment, Reduces conflicts, Ease of management Dynamic Update.

③ Design and configure an internetwork in your lab using switch, router and ethernet cable. Draw and label design your notebook. Also show IP address configuration



Router IP : 192.168.1.1

Switch

PC0 : IP : 192.168.1.2

Subnet mask : 255.255.255.0

Default Gateway : 192.168.1.1

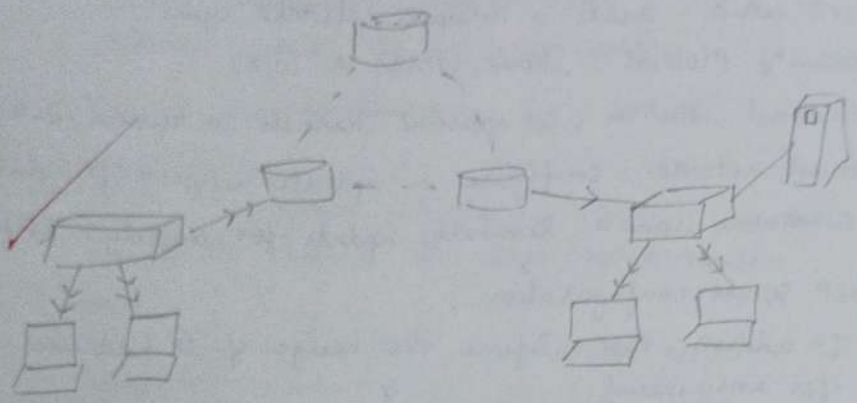
PC1 Ip 192.168.1.3

Subnet mask : 255.255.255.0

Default Gateway : 192.168.1.1

EXPERIMENT - 11

a) AIM: Stimulate Static Routing Configuration using CISCO packet tracer.



Lab Setup:

Network Design Create a lab with router and network as per specified requirement

Connection networks

Router 0 10.0.0.0/8, 40.0.0.0/8, 3.0.0.0/8, 5.0.0.0/8

Router 1: 20.0.0.0/8, 30.0.0.0/8, 6.0.0.0/8, 10.0.0.0/8, 40.0.0.0/8

Router 2: 40.0.0.0/8, 50.0.0.0/8, 20.0.0.0/8, 30.0.0.0/8

Router Configuration:

→ Router for 30.0.0.0/8

Main: Ip route 30.0.0.0 255.0.0.0 20.0.0.210 (using Router1)

Backup: Ip route 30.0.0.0 255.0.0.0 40.0.0.220 (using Router2)

→ Host route for 30.0.0.100/8

Main: Ip route 30.0.0.100 255.255.255 40.0.0.210

Backup: Ip route 30.0.0.100 255.255.255.20.0.0.220.

→ Route 50.0.0.0/8

Main: Ip route 50.0.0.0 255.255.255 40.0.0.210

Backup: Ip route 50.0.0.0 255.0.0.0.20.0.0.220

→ Route 10.0.0.0/8:

Main: Ip route 10.0.0.0 255.0.0.0. 20.0.0110

Backup: Ip route 10.0.0.0 255.0.0.0.50.0.0.1

→ Route 40.0.0.0/8

Main: ip route 40.0.0.0 255.0.0.0 20.0.110

Backup: iproute 40.0.0.0 255.0.0.0 50.0.0120

Router 2 configuration:

→ Route 10.0.0.018

ip route 10.0.0.0 255.0.0.0 40.0.0.1

→ Route 30.0.0.018

ip route 30.0.0.0 255.0.0.0 50.0.0.2

Verifying Configuration:

- Use show ip route static to view the routing table
- Testing routes
  - Ping or use tracert from PC's in connected network to verify routing paths.

Simulating Route failure

- To test back up routes, disconnect the primary route and recheck connectivity to the target network.

Deleting Static routes:

1) use show ip address to view all routes
2) Identify the route to delete
3) use no ip route [destination] [mask] [nextstop] to remove the route.

19/17

RESULT: Thus the experiment to simulate static routing configuration using CISCO pkt Racer was done succesfully.

b) AIM: Simulate RIP using CISCO Packet Tracer

network Configuration:

- Devices and IP Configuration

PC0: 10.0.0.2/8 (Connected to Router Fa 0/1)

Router 0 Fa0/1 : 10.0.0.1/8

Router S0/0/1 : 192.168.1.254/30 (Connected to Router 3 S0/0/1)

Router S0/0/0 : 192.168.1.249/30 (Connected to Router1 S0/0/0)

Router S0/0/0 : 192.168.1.250/30

Router S0/0/1 : 192.168.1.253/30

PC1: 20.0.0.2/30 (Connected to Router Fa0/1)

Router 2 100/1: 20.0.0.1/30.

Steps to configure:

Assign IP address to Router

↳ Access each router using CLI

↳ Use configure terminal to enter global config mode

Configure serial interface

↳ Set clock rate and bandwidth for DCE ends of serial connections.

Implement RIP Protocol

↳ Enable RIP on each router with router rep

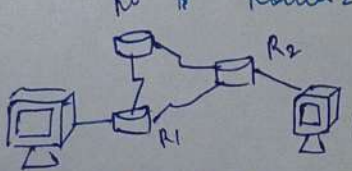
↳ Specify network to advertise using network command.

Verification

ping test: use the ping command from PC to test connectivity to PC0

Route Management: RIP dynamically manages routes automatically switching to alternatively routes if one goes down.

Simulate link feilure: Disconnect a cable between Router 0 R0 & Router 2, then use Tracer to observe how RIP reroutes traffic



RESULT: Simulating RIP using CISCO packet Tracer Implemented successfully.

19/11

EXPERIMENT - 12

a) AIM: Implement echo client server using TCP socket

ALGORITHM
AND CODE :

~~TCP Echo Server~~: Import socket module
Create TCP socket
Bind the socket to specific IP addr & port
listen for incoming connections
Accept a client connection
Loop: Receive data from client
Send it back (echo)
no connection (close)
Close the server socket

CODE:

TCP Echo Server:

```
import socket
def server(host='127.0.0.1', port=65432):
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as server_socket:
        server_socket.bind((host, port))
        server_socket.listen()
        print(f'tcp echo server listening on {host}:{port}')
        while True:
            conn, addr = server_socket.accept()
            with conn:
                print(f'connected by {addr}')
                while True:
                    data = conn.recv(1024)
                    if not data:
                        break
                    conn.sendall(data)

if __name__ == "__main__":
    server()
```

TCP client :

```
def start_client():
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_address = ('localhost', 12345)
    client_socket.connect(server_address)
    print(f"Connected to server at {server_address}")

    while True:
        message = input("Enter message to send (or exit to quit): ")
        if message.lower() == 'exit':
            print("Exiting...")
            break

        client_socket.sendall(message.encode())
        response = client_socket.recv(1024)
        print(f"Received from server: {response.decode()}")

    client_socket.close()

if __name__ == "__main__":
    start_client()
```

Output :

Server : Enter message to send : hello India - client
Received from server : hello India - server

Client : server connection established
Received message: hello

connected to server
Enter message to send : hello.

4/9/24

RESULT: Thus the program to show client server connection using TCP socket was done successfully

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

ALGORITHM:

- Create a socket using TCP (sock_stream)
- Connect to server's IP address
- Take input from the user
- Print receive a respon from server
- Print server response on client side.

CODE:

Server:

```
import socket

server_host = '127.0.0.1'
server_port = 12345

server_socket = socket.socket(socket.AF_NET)
server_socket.bind((server_host, server_port))

while True:
    message = client_socket.recv(1024).decode().
    if message.lower() :"quit"
        print("client has ended the chat")
        break
    print(f"Client" {message}")
    response = Input("server...")
    client_socket.send(response.encode())
    if response.lower() : 'quit':
        print("server has ended the chat")
        break

client_socket.close()
server_socket.close()
```

client :

```
import socket
server_host = '127.0.0.1'
server_port = 12345

client_socket = socket.socket(socket.AF_INET)
client_socket.connect((server_host, server_port))

While True:
    message = input("Client")
    client_socket.send(message.encode())

if message.lower == 'quit':
    print("client ended chat")
    break
response = client_socket.recv(1024).decode()
print(f"Server: {response}")
client_socket.close()
```

Output:

client Enter message ~~from~~ server = "Hello server"

server Reply = "Hello client"

Server listening on ("localhost', 12345) connection established Received message : hello	connected to server at (localhost, 12345) Enter message to send : hello
~~client~~ Server : hello client : hi server : how	server : hello client : hi server : how

9/11

RESULT: Thus the implementation of chat client server using TCP was successful and verified.

EXPERIMENT - 13

AIM: Implement your own ping program

ALGORITHM:

- Open a raw socket to send ICMP req
- Create the ICMP Echo request packet including a header and data
- Send Packet. Send the ICMP request to target host
- Calculate the time
- Show response

CODE:

```
import os
import socket
import struct
import time
import select

ICMP_Echo_request = 8
ICMP_Echo_Reply = 0

def checksum(data)
    Sum = 0
    countto = (len(data)/2) * 2
    count = 0
    while count < countto:
        this_val = data[count + 1] 255 + data[count]
        sum = sum + this_val

def create_packet(id)
    header = struct.pack('bbHHh', ICMP_Echo_req, 0, 0, id, 1)
    checksum_val = checksum(header)
    header = struct.pack(ICMP_echo_response, 0, checksum_valid, 1)
```

```
def ping (host):
    try
    ICMP_ socket = socket.socket (socket.AF_NET)
    dest_addr = socket.gethostbyname(host)
    packet_id = os.getpid()
    packet = create_packet(packet_id)

    icmp_socket.sendto(packet,(dest_addr,1))
    start_time = time.time()

    if ready[0]
        rec_packet, addr = icmp_socked_recvfr(1024, end_time=time
                                              .time
```

```
if __name__ == "__main__":
    host = input("Enter host to ping :) google.com
    ping(host)
```

Output

Enter host to ping: www.google.com

Reply : time = 21.45ms

19/11

RESULT :

Thus the ping program is successfully executed and the output was verified successfully

EXPERIMENT - 14

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

ALGORITHM

Import scapy modules
Set network layer to capture IP Packets
Define a function to process packets
    Check if packet has IP layer
    Get source IP, destination IP and protocol type
    Identify the Protocol
    Print the protocol and IP address
Capture packets using the sniff function
Run the main function if the script is executed directly.

CODE:

```
from scapy.all import sniff
from scapy.layers.inet import IP,TCP,UDP,ICMP
from scapy.config import conf

conf.L3socket = conf.L3socket6

def packet_callback(packet):
    if IP in packet:
        ip_layer = packet[IP]
        protocol = ip_layer.proto
        src_ip = ip_layer.src
        dst_ip = ip_layer.dst

        if protocol == 1:
            protocol_name = "ICMP"
        elif protocol == 6:
            protocol_name = "TCP"
        elif protocol == 17:
            protocol_name = "UDP"
        else
            protocol_name = "Unknown"
```

```
        print(f" Protocol : {protocol_name}")
        print(f" Source IP: {src_ip}")
        print(f" Destination IP : {dst_ip}")
        print ("-" *50)

def main()
    sniff (prn = packet_callback, filter="ip", store=0)

if __name__ == "__main__":
    main()
```

Output :

```
Protocol : TCP
Source IP : 140.82.112.25
Destination IP : 192.168.1.8
- - - - - -
Protocol : TCP
Source IP : 192.168.1.8
Destination IP : 140.82.112.25
- - - - - -
Protocol : UDP
Source IP : 192.168.1.8
Destination IP : 20.42.65.91
```

RESULT : Thus the program was executed successfully to show the implementation of packet sniffer.

EXPERIMENT - 15

AIM: To analyze the different types of web logs using webalizer tool

PROCEDURE:

Step 1: Run webalizer windows version

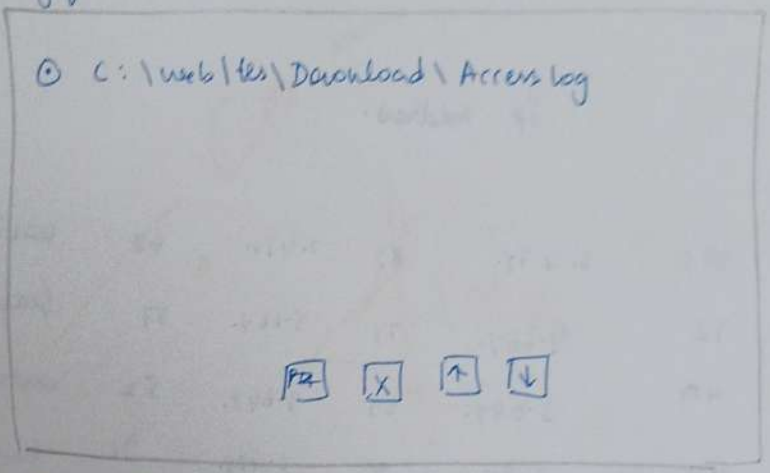
Step 2: Input web log file (download from web)

Step 3: Press Run webalizer.

INPUT:

Choose log files using Additional HTML code.
hide | glh | ignore | include graph config file

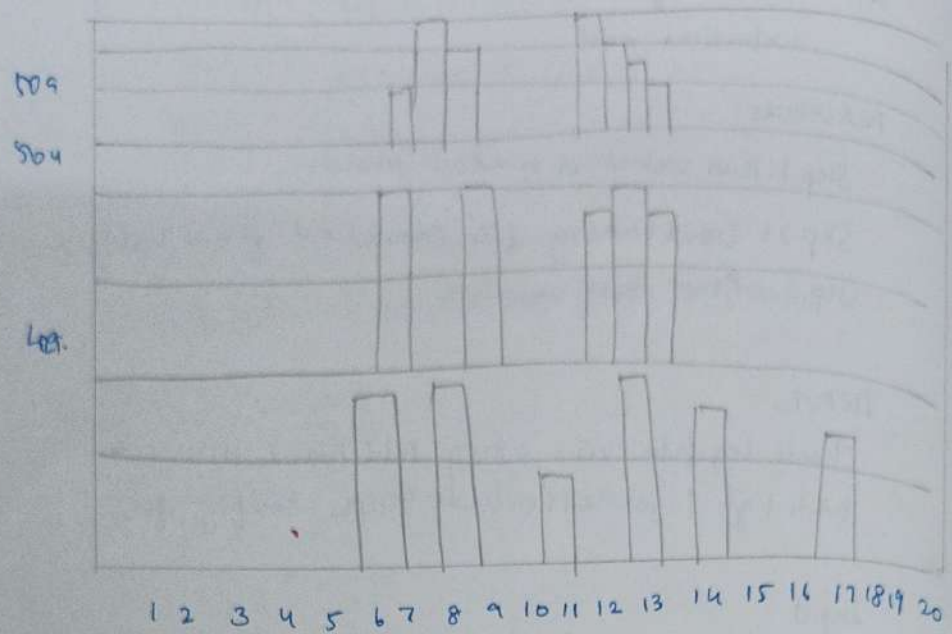
Input
log files:



Target Directory

D lear existing directory.

Monthly Statistics



Ip Address.

1	100	6.47%	83	7.47%	62	searchable.net	
2	72	4.66%	71	5.66%	51	search.net	
3	49	3.04%	27	3.64%	52	outlook.com	
4	44	2.95%	81	3.45%	91	learn.com	
5	35	2.5%	63	2.95%	87	bc.ca	
6	29	3.95%	41	4.95%	69	pam.com	

RESULT:

Thus the experiment to analyse web logs was done successfully.

completed

19/11