INDEX

NAME: CHARAN JEETH R.M. ROLL NO: 52

SUBJECT: COMPUTER NETWORK OBSERVATION STD: CSE DIV: "A"

SCHOOL: REC

SR.NO.	DATE	TITLE	PAGE NO.	SIGNATURE
1	13/7/24	Basic Networking commands		
2	27/7/24	Study of Network cables		
3	30/7/24	Exp. on Cisco packet tracer		
4	17/8/24	Setup and configure LAN using Switch & ethernet cable.		
5	17/8/24	Packet Capture tool Wireshark		
6		Sliding window protocol		
7a)		VLAN configuration using Cisco Packet tracer		
7b)		wireless LAN using ciscopacket tracer		
8.		Hamming Code		
9.		Subnetting in Cisco packet tracer		
10.	a)	Internetworking with routers in Ciscopacket tracer.		
11.	b) a2)	DHCP-is ciscopacket tracer Static routing configuration		
12.	b) a)	using ciscopacket tracer		
13.	b) chat	Echo client server in TCP/UDP Client server in TCP/UDP Ping program		
14.		RAWsocket to implement		
15.		Packet sniffing Analyze different types of each log using wireshark tool		

completed

13/7/24

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

BASIC NETWORKING COMMANDS:

arp -a: Interface : 172.16.75.44 --- 0x13

internet address	physical address	TYPE
172.16.72.1	7c-5a-1c-cf-ne-41	dynamic
172.16.72.133	4c-ac-a3-65-97-f3	dynamic
172.16.72.195	4c-ac-a3-64-fc-5b	dynamic

host name:

C:\Users\Lenovo>hostname

DESKTOP-C01BH7D

ipconfig /all

Host name . . . . . : DESKTOP-C01BH7D
Primary Dns suffix . . . . . :
Node type . . . . . :
IP Routing enabled . . . . . : No
WINS Proxy enabled . . . . . : No

nbtstat -a: Lists the remote machine's name table given its name

-A Lists the remote machine's nametable given its IP address

-n lists local NetBios names.

nestat:

Active Connections

Proto	Local Address	Foreign address	state
TCP	127.0.0.1:49679	DESKTOP-C01BH7D:49680	est.
TCP	127.0.0.1:49680	DESKTOP-C01BH7D:49679	est.
TCP	127.0.0.1:49682	DESKTOP-C01BH7D:49683	est.

nslookup:

Default server: Unknown

Address: 172.16.72.1

Pathping:

server: Unknown

address: 172.16.72.1

[-g = host-list] [-i address] [-n]

[-h maximum-hops] [-q num-queries] [-w ti

Ping:

server: Unknown

address: 172.16.72.1

[-t] [-a] [-n count] [-l size] [-f] [-i TTL

[-v TOS] [-r count] [-s count]

[-S Srcaddr]

Route: manipulates network routing tables

[-f] [-p] [-4 | -6] command [destinat

Some important Linux networking commands:

ip [OPTIONS] OBJECT {COMMAND | help}
ip [-force] -batch filename

#ip address show:
1. lo : <LOOPBACK, UP, LOWER_UP> mtu 65536 qdisc noqueue
state UNKNOWN group default qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
#ipaddress add 192.168.1.254/24 dev enp503

Added Successfully

#ipaddress del 192.168.1.254/24 dev enp503

Deleted Successfully.

#ip link set eth0 up

Link Set up successfully.

e. # ip link set eth0 down

Link set down successfully.

f. ip link set eth0 promisc on

link set promisc on is successful

g. # ip route add default via 192.168.1.254 dev eth0

route added to default successfully

h. # ip route add 192.168.1.0/24 via 192.168.1.254

route added successfully.

i. # ip route add 192.168.1.0/24 dev eth0

Route added successfully.

j. # ip route delete 192.168.1.0/24 via 192.168.1.

IP route deleted.

k. # ip route get 10.10.14

10.10.14.0 via 172.16.8.1 dev emp2so src

172.16.8.108 uid 1000

Cache

2. ip config

eth250: flags=4163<UP,BROADCAST,RUNNING,
MULTICAST> mtu1500

inet 172.16.8.68 netmask 255.255.252.0
broadcast 172.16.11.255 ether
inet6 fe80::e743:e5fc:1686:5e9
prefixlen 64 scopeid 0x20<link>

3. MTR

host	packets Loss%	Snt	Last	pings avg	best	wrst	stdev
1...::1	0.0%	42	0.1	0.1	0.1	0.1	0.0

.1 # mtr google.com

host	packets Loss%	Snt	Last	pings avg	best	wrst	Stdev
12.16.8.1	0.0%	73	0.4	0.5	0.2	11.2	1.1
42.250.171.162	0.0%	73	2.1	3.7	1.7	10.3	3.0
42.251.227.215	0.0%	73	2.7	3.7	2.3	38.6	5.6

b) # mtr -g google.com

NO GUI SUPPORT. SORRY.

c) # mtr -b google.com

host	Packets Loss%	Snt	pings Last	Avg	Best	Wrst	Stdev
172.16.8.1	0.0	58	0.2	0.3	0.2	0.5	0.0
static-41.229.249.49-tetraide	0.0	58	4.9	2.8	2.2	9.8	0.8
142.250.171.162	0.0	58	2.1	2.5	1.7	8.0	1.0

d) # mtr -c 10 google.com

host	packets loss	Snt	Last	pings Avg	Best	Wrst	Stdev
172.16.8.1	0.0%	8	0.4	0.3	0.3	0.4	0.0
142.250.171.162	0.0%	7	2.7	2.9	2.7	3.0	0.0
142.251.227.217	0.0%	7	2.4	2.6	2.2	3.2	0.0

5. Ping:

Usage: ping [-aAbBdDfhLnOqrRUvV64] [-c count]
[-i interval]

~~Ping google.com~~

ping google.com

64 bytes from sof02s27-in-f14.1e100.net
(216.58.206.174): icmp-seq
=1 ttl=56 ~~=~~
time=10.7

Ping -c 10 google.com

64 bytes from maa05s10-in-f14.1e100.net (216.58.200.
142):
icmp-seq=1 ttl=120 time=2.86 ms

~~6~~64 bytes from maa05s10-in-f14.1e100.net (216.58.200.
142):
icmp-seq=2 ttl=120 time=3.00 ms

2 packets transmitted, 2 received,
0% packet loss, time 1001 ms

rtt min/avg/max/mdev = 2.861/2.930/3.000/0.088
ms

TCP Dump:

#dnf install -y tcpdump

Installed successfully.

#tcpdump -D

1. enp2s0 [up, running]
2. any (pseudo-device that captures on all interfaces)
3. lo [up, running, loopback
4. wlp3s0 [up]

#tcpdump -i eth0

verbose output suppressed, use -v or -v for full protocol decode listening on lo

Stopped

#tcpdump -i eth0 -c 10

Link type EN10MB (ethernet) capture size 262144 bytes

0 packets captured

0 packets received by filter

0 packets dropped by kernel

tcp dump -i eth0 net 10.1.0.0 mask 255.255.255.0

non-network address not in "10.10.10.10 mask 255.255.255.0"

tcp dump -i eth0 net 10.1.0.0/24

link-type EN10MB (Ethernet) capture size 262144 bytes

#tcpdump -i 10 port 53

To capture only DNS port 53 Traffic

tcp dump -i eth0 host 8.8.8.8 and port 53

This is for a specific host

NMCLI

"VMware VMXNET3"
ethernet (vmnet3), 00:0C:29:6C:E0:PP, ...
ip4 default
inet4 192.168.22.128/24
route4 192.168.22.0/24 metric 100

nmcli connection show

NAME	UUID	TYPE	DEVICE
wired connection	70ca0ac8-7156-3516-a9e0-ab2a02b8ac9	ethernet	ens160

#nmcli connection modify "wired connection" "wired connection 1" is name of connection.

#nmcli connection show
connection.interface-name: ens160
connection.autoconnect: yes
ipv4.method: auto
ipv6.method: auto

#ip address show ens160
ens160: <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 ens160
inet6 2001:db8:1:0:ff:e164 scope global noprefixroute

RESULT:
Thus the study of various network commands used in Linux & windows is done and executed successfully.

PRACTICAL - 2

Aim: Study of different types of Network Cables

a) Understand different types of network cables.

1. Unshielded Twisted pair cable
2. Shielded Twisted pair cable
3. Coaxial cable
4. Fibre optic cable

Cable Type	Category	Maximum Data Transmission	Advantages / Disadvantages	Application / use	Image
UTP	Category 3	10Mbps	Adv: • cheaper in cost • Easy to install as they have a smaller diameter Disadv: • More prone to EMI and noise	10Base-T ethernet	
	Category 5	up to 100mbps		Fast ethernet, Gigabit Ethernet	
	Category 5e	1Gbps		Fast ethernet, Gigabit ethernet	
STP	Category 6, 6a	10Gbps	Adv: • Shielded • Faster than UTP • Less susceptible to noise and interference	Gigabit Ethernet, 10G ethernet widely used in data centres	
SSTP	Category 7	10Gbps	Disadv: • Expensive • greater installation efforts	Gigabit ethernet, 10G ethernet (100m)	

coaxial cable	RG-6 RG-59 RG-11	10-100 mbps	• High bandwidth • Low loss bandwidth • versatile Disadv: • Cost • Size is bulky	speed of signal is 500m Tv network High speed internet
Fibre optic cable	single mode multiple mode	100gbps	Adv: • High speed • High bandwidth • High security Disadv: • Expensive • Requires skilled installers	• Maximum distance of fibre optic cable is around 100 m

b) Make your own Ethernet cross-over cable/ Straight cable

Tools and parts needed:

- Ethernet cabling. CAT 5e 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.
- Two RJ45 plugs
- Optional two plug shields.

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 sheet, arrange them from top to bottom.

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. To do so, hold the plug with the clip side facing away from you have the gold pins facing towards you, as shown

Step 5:

Next, push the cable right in. The notch at end of plug need to be just over the cable shielding, that means that you stripped off too much shielding.

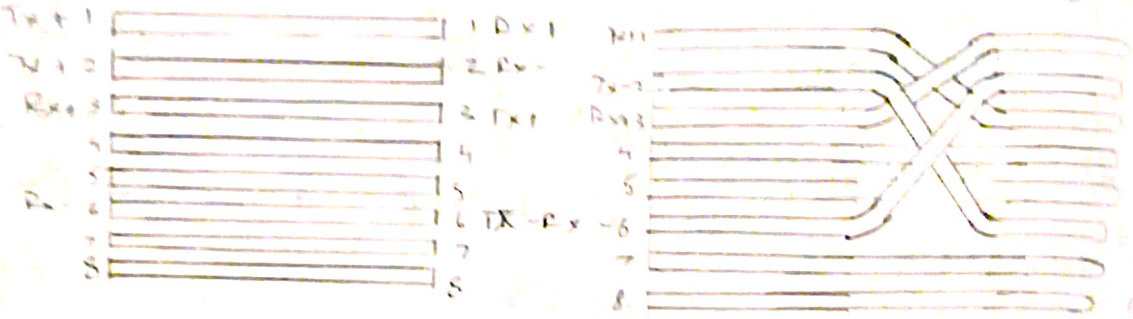
Step 6 :

After the wires are securely sitting inside the plug, insert into the crimping tool & push down.

Step 7:

Lastly, repeat for the other end using diagram B / using diagram A

b) Make your own Ethernet cross-over cable / straight cable



°Ethernet cabling, CAT 5e is ~~not~~ 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 pins in the plug & strip & cut the shielding of the cables.

°Two RJ45

° optional two plug shields.

RESULT :

Thus, the study of different type of networking cables has been executed successfully.

PRACTICAL - 3

AIM: To study the Packet tracer tool Installation and user interface Overview.

c) To understand environment of CISCO PACKETS TRACER to design Simple network.

INTRODUCTION:

A simulator, as the name suggests, simulates network devices and its environment.

1. It allows you to model complex systems without 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 Linux and Windows desktop environments.

4. protocols in packet tracer are coded to work and behave in same way as they would on real hardware

INSTALLING PACKET TRACER:

To download packet tracer, go to http://www.netacad.com and log in with your Cisco networking Academy credentials

WINDOWS:

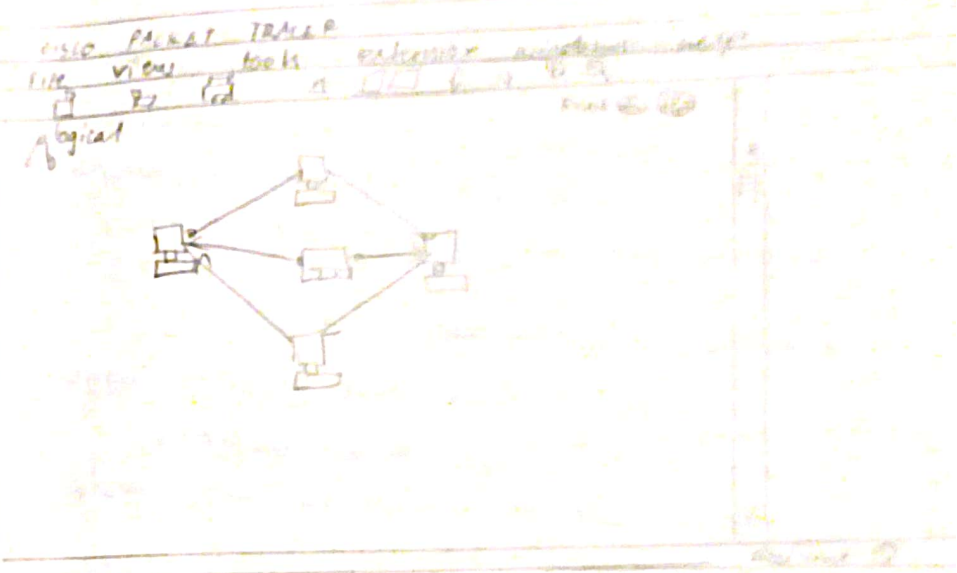
In windows is pretty simple and straight forward; the setup comes in a single file named packettracer-setup 6.0.1.exe open this file to begin the setup wizard accept the license agreement, choose a location, and start the installation.

LINUX:

Users with an Ubuntu/Debian distribution should download 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 installation

USER INTERFACE OVERVIEW:

The layout of packet tracer is divided into several components. The components of packet tracer interface are as follows



MENUBAR:

It is a menu found in all software applications, it is used to open save print, change Preference and so on.

MAIN TOOLBAR:

It provides shortcut icons to menu option that are commonly accessed such as open save, zoom, undo and redo, and on right-hand side is an icon for entering network information for current network

LOGICAL/PHYSICAL WORKSPACE TABS:

These tabs allow you to toggle between the logical & Physical work

WORKSPACE:

It is the are where topologies are created and simulations are displayed.

COMMON TOOLS BAR:

Toolbar provides controls for manipulating topologies, such as select, move layout, place, note, delete, inspect and add simple/complex PDU.

REALTIME / SIMULATION TABS:

There tabs are used to toggle between the real and simulation modes, multons are also provided to control time

NETWORK COMPONENT BOX:

This component contains all four Networks and end devices available with Packet Tracer.

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 network component box, click and drag-add-drop the below components:
   a. 4 generic PC's and one HUB
   b. 4 generic PC's and one Switch

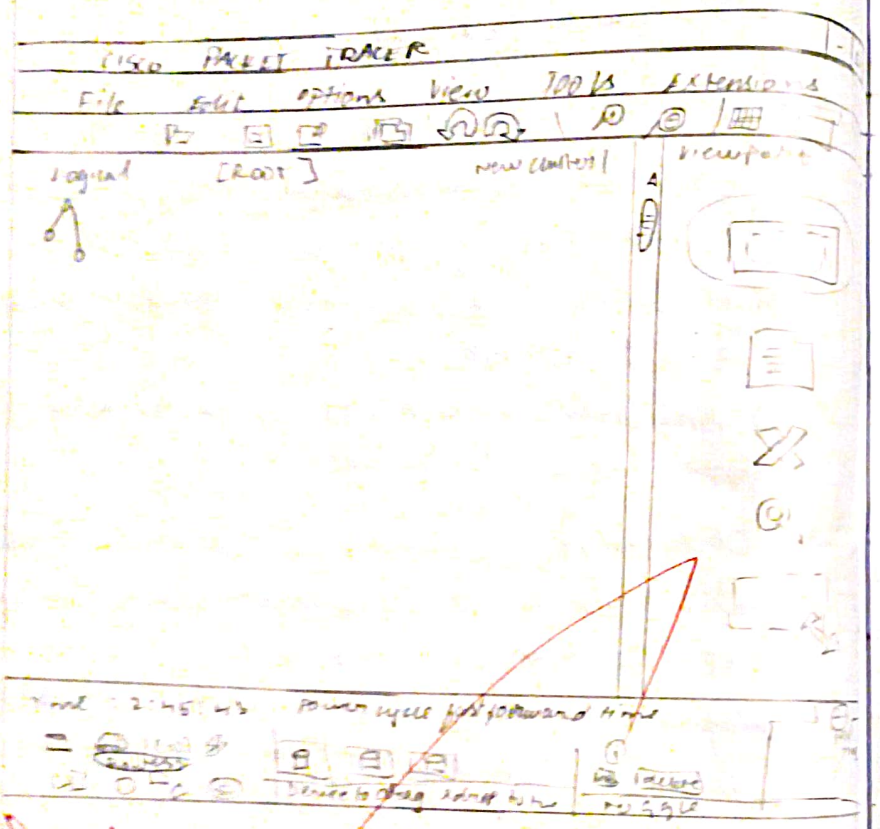
2. Click on Connections:
   a) Click on copper straight-through cable
   b) Select one of PC and connect it to HUB using cable. Similarly connect remaining 3 PC's to HUB.
   c) similarly connect 4PCs to switch using copper straight-through cable

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

4. Observe the flow of PDU from source PC to destination PC by selecting realtime mode of Simulation.

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

6. Observe how HUB & switch are forwarding PDU & write your observation & conclusion about behaviours of Switch & HUB



RESULT:

Thus, to study the packet Tracer tool installation & user interface view ~~too~~ is executed successfully.

STUDENT OBSERVATION:

PRACTICAL -1

1) ping <hostname or ipaddress>

2) traceout <hostname or ipaddress>

3) ifconfig or ip addr show

4) netstat -tuln

5) ifconfig eth0 192.168.1.100 netmask 255.255.255.0 up

STUDENT OBSERVATION

PRACTICAL - 2

1) Straight cable :
- Used to connect different types of devices
- Typically used for connecting a computer to network switch, hub, or router.

Crosscable:
- Used to connect similar type of devices.
- Used for directly connecting 2 devices

2) A cross cable is typically used to connect to a PC to ~~router or~~ PC ~~switch~~ without intermediate network device.

3) A straight cable is used ~~for~~ to connect a PC to router or switch

4)
- Cat5
- Cat5e
- Cat6

5) Understanding :

Straight : Easier to make because the same colour is used to on both ends of a cable

Cross cable : Slightly more complex as you need to follow diff. wiring pattern

output Received :

Straight cable : If all the lights on the tester line up correctly, the straight cable was made successfully.

~~Staff~~

Cross cable: Testing the crosscable with a tester should show that the wires are correctly crossed.

PRACTICAL - 3

STUDENT OBSERVATION:

a) Switch : o more advanced than hub.
- when a switch receives a packet, it checks the destination mac address.
- Each port on a switch represents a seperate collision domain

~~Hub~~ :

Hub:
- It is a smaller device with no intelligence
- All ~~host~~ devices connected to a hub share same collision domain, which leads to collisions

b) Star topology, Bus topology, Ring topology, mesh topology.

PRACTICAL - 4

AIM: Setup and configure a LAN using a Switch and ethernet cables in your lab.

What is a LAN?

It refers to a network that connects devices within a limited area, such as an office building, school or home. A LAN switch serves as primary connecting device, managing & directing communication within local.

HOW TO SETUP A LAN?

1. Plan and design an appropriate network topology taking into account network requirments and equipment location.

2. You can take 4 computers, a switch with 8, 16 or 24 ports which is sufficient for network of sizes & 4 ethernet cables.

3. Connect your computers to network switch via ethernet cable into your computer & other end in network switch.

Step4:

1. Log on to the client computer as administrator or as owner.

2. Click network & Internet conne

3. Local area connection / Ethernet -> Lo Properties -> select internet protocol (TCP/ Click on properties -> select use follow IP address option & assign ip address

Step 5:

1. Connect your computer to switch
2. Log into web interface
3. configure basic settings
4. Assign IP address as: 10.1.1.5, su mask 255.0.0.0.

Step 6: Check connectivity c/w switch other machine ing using ping command in commando prompt of device

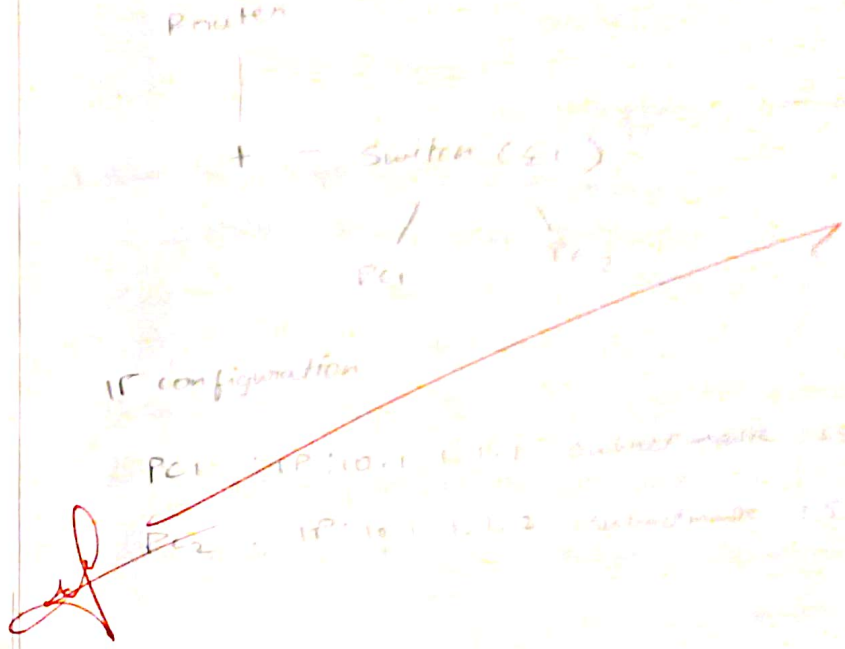
Step 7: Select a folder -> go to proper Click sharing tab -> share it with everyo on the same LAN.

Step 8: Try to access shared folder fro others computer of network.

RESULT:

Thus to setup and configure a LAN using a Switch and Ethernet cable in your lab has been executed successfully.

STUDENT OBSERVATION:



PRACTICAL - 5

AIM:

Experiment on packet capture tool wireshark

Packet Sniffer:

- Stores and displays the contents of various protocall fields in messages
  - never sends packets itself
  - no packet addressed to it
  - receives a copy of all packets.

Packet Sniffer structure diagnostic tools.

↳ tcpdump

↳ wireshark

WIRESHARK:

A network analysis tool formerly known as Ethereal, captures packet in real time and display them in human-readable format.

→ What we can do: ◦ Capture network traffic
◦ Decode packets Protocols using dissectors.

→ used for:

People: learn network protocol internals network administrators: Trouble shoot network problems.

Getting Wireshark:

It can be download for windows or macos from its official website.

Capture Packets:

After downloading & installing wireshark launch it & double-click the name of a network interface under capture to start capturing packets.

"Packet List": display all packets in current capture.

"Packet Bytes": Shows data of current packet, Colour coding, filtering packet, Flow graph.

PROCEDURES:

1. Create a filter to display only TCP/UDP packets, inspect the packet & Provide flow graph.
2. Create a filter to display only ARP packets and inspect packets.
3. Create a filter to display only

4. Create a filter to display only HTTP packets and inspect packets

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

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

Student Observation:

1. What is Promiscuous mode?

It is a configuration network interface card (NIC) or device that allows to capture & process packets on network.

2. No, ARP packet do not have a transport layer header. It is a protocol used to map an IP address to physical mac address.

3. DNS uses both UDP & TCP as its transport layer protocol.

4. It is a special address used to send packet to all devices on a specific network or subnet.

RESULT:

Thus to experiment on packet capture tool wireshark has been executed successfully.

PRACTICAL - 6 : HAMMING CODE

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

Error Correction at data link layer:

It is a set of code that can be used to detect & correct the errors that occur when data is transmitted from send to receiver.

Create a sender program with below features

1. Input to send file should be a text of any length. program should convert text to binary.
2. Apply hamming code concept on binary da & add redundant bits.
3. Save this output in a file called chan

Create Receiver program with below features

1. Receiver program should read input from file.
2. Apply hamming code on binary data to Check error.
3. If error, display position of error.
4. Else, remove redundant bits & convert binary data to ASCII & display output.

Student

CODE:

```
import numpy as np
def text_to_binary(text):
return ''.join(format(ord(char), '08b') for
    char in text)

def binary_to_text(binary):
    chars = [binary[i:i+8] for i in range
        (0, len(binary), 8)]
return ''.join([chr(int(char,2)) for char
    in chars])

def calc_redundant_bits(m):
    r=0;
    while(2**r<m+r+1):
        r+=1;
    return r

def pos_redundant_bits(data, r):
    j=0
    k=0
    m=len(data)
    res=''
for i in range(1, m+r+1)
    if i==2**j:
        res=res+'0'
        j+=1
    else: res=res+data[k]
        k+=1
    return res
```

```
def calc_parity_bits(arr,r):
    n=len(arr)
    arr=list(arr)

for i in range(r):
    Parity=0
    Position=2**i
for j in range(1,n+1):
    if j & position:
    Parity^=int(arr[j-1]).
    arr[position-1]=str(parity)
    return ''.join(arr)

def detect_and_correct(data, r):
    n=len(data)
    res=100
for i in range(r):
    Parity=0
    position=2**i

for j in range(1,n+1):
        if j & position
            parity^=int(data[j-1])
    if parity!=0:
        res+=position.

if res!=0:
print(f"Error detected at position: {res}")
    data=list(data)
```

```
print (f"Error corrected at ...")
else
    print ("Error position out of range
      not correction performed"),

corrected_data = ''.join(data)
return corrected_data
else
    print ("No error detected")
    return data

def remove_redundant_bits(data, r):
    j = 0
    original_data = ''
    for i in range(1, len(data)+1)
        if i == 2**j :
            j += 1
        else
            original_data += data[i-1]
    return original_data

def introduce_error(data, position):
    if position < 1 or position > len(data):
        print("Error position is out of range")
        return data
    data = list(data)
    data[position-1] = '0' if data[position-1]
        == '1' else '1'
    print(f"Introduced error at position {position}")
    return ''.join(data)
```

```
# Sender program

def sender(text)
    binary_data = text_to_binary(text)
    m = len(binary_data)
    r = calc_redundant_bits(m)
    arr = pos_redundant_bit
    arr = calc_parity_bits(arr, r)
    print ("Sender output (binary with
                redundant bits): {arr}")
    return arr

# Receiver program

def receiver(data):
    r = calc_redundant_bits(len(data))
    corrected_data = detect_and_correct(data, r)
    original_data = remove_redundant_bits(
                                corrected_data, r)
    ascii_output = binary_to_text(original_data)
    print ("Decoded text:", ascii_output)

if __name__ == "__main__":
    input_text = input("Enter text to be encoded")
    channel_data = sender(input_text)
    corrupted_data = introduce_error(channel_data
    receiver(corrupted_data)
```

O/P :
Enter text to encode: hello
Sender O/P : 0110100101100 ...
Introduced error at position 2
Error detected at position 2
Decoded text: hello

O/P :

Enter text to be encoded : Hello

Parity :

Sender o/p :

1000100 110000 11100101 01 1011000 1101 1 0001101111

Introduced error at position : 2

Error detected at position : 2

Decoded text : Hello

CODE EXECUTED SUCCESSFULLY.

Enter text to encode : Charan

Encoded binary data : 0110000 11011 0100000 110000 1110010011000010 1

No. of redu : 6

Placing redundant bit at position : 1 2 4 8 16 3

No. of redundant bits : 7

Parity bit : [Position 1:0] [Position 2:0] [position 4:0] [Position 8:0] [position 16:0] [position 32:1]

Sender Output : 0000110000 1101100 100000 11 000011 1100100 110000101 10111

Introduced error at position : 2

Binary with error : 010011 000011011 0010000 11000 1111 00100 1100001011 01110

Error detected at position : 2

Error corrected at position : 2

Binary after error correction : 0000 110000 11 011 0011 11 0000 101111 0 0100110000 10110

Decoded text : Charan

PRACTICAL - 7: SLIDING WINDOW PROTOCOL

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.

Create a Sender program with following features:

1. Input window size from user.
2. Input a text message from user.
3. consider 1 character per frame
4. Create a frame with following fields.
5. Send the frames.
6. Wait for acknowledgement from receiver.
7. Reader a file ~~re~~ called Receiver_buffer.
8. Check ACK field for acknowledgement number.
9. If the acknowledgement number is as expected, send new set of frames accordingly; Else if NACK is received, resend the frames accordingly.

Create a receiver file with

1. Reades a file (called sender-buffer
2. Check the frame no.
3. If the Frame no. are as expected, write the appropriate Ack no. in Receiver. buffer file.
Else write Nack no. in Receiver - buffer file.

STUDENT OBSERVATION

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].acknowledge
  Print(f"Sent Frame {frames[i].frame_no}
                    {frames[i].data}")
Print("Frames sent, waiting for acknowledgement..
  def receive_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}
                    [CORRUPT]")
      frames[i].acknowledged = False
```

```
print(f"Received Frame{frames[i].frame_no}
                {frames[i].data}[OK]")
    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("\n Resending unacknowledged
        frames...\n")
time.sleep(2)
print("\n All frames sent and acknowledged")
if __name__ == "__main__":
    sliding_window_protocol()
```

OUTPUT

```
Enter window size: 3
enter a message to send: Protocol
  --- sending frames ---
Sent Frame 0: P
Sent Frame 1: R
Sent Frame 2: O
Frames sent, waiting for acknowledgements...
---- Receiving Frames...
Received frame 0: P [OK]
Received frame 1: R [ERROR]
Received frame 2: O [OK]
Resending unacknowledged frames
--- sending frames ---
sent frame 1: R
sent frame 3: T
Frames sent, waiting for acknowledgements
--- receiving frames ---
Received frames 1: R [ERROR]
Received frames 3: T [ERROR]
Resending unacknowledged frames
    Sending frames---
sent frame 4: O
sent frame 5: C
Sent frame 6: O
frames sent waiting for acknowledgement
--- Receiving Frames---
Received frame 4: O [ERROR]
Received frame 5: C [OK]
Received frame 6: O [OK]
```

Resending unacknowledged frames....

---sending frames---

sent frame 4:0

Frames sent, waiting for acknowledgement

--- Receiving frames ---

recieved frame 4:0 [Ok]

Resending unacknowledged frames....

--- sending frames---

sent frame 7:1

frames sent, waiting for acknowledgement

---Receiving frames---

Received frame 7:1 [Ok]

All frames sent and acknowledged.

RESULT:

Thus, the program ~~to~~ implement sliding window protocol is executed successfully.

29/11.

EX.NO:08

AIM: To simulate a virtual LAN (VLAN) configuration using cisco packet tracer, follow these step by step instruction.

S-1: Setup topology:

1. open cisco packet tracker
2. drag & drop switches (ciscoz960) 2 PCs into workspace
3. maintain VLAN port assignment & the VLAN database
4. configure an 802.10 trunk b/w switches

INSTRUCTIONS

PART-1: Build network & configure Basic device settings.

STEP 1: Build the network as shown in topology.

a. Click & drag both switch S1 & S2 to rack
b. Click & drag both PC-A & PC-B to table & use the power button to turn them down.
c. Provide network connectivity
d. Connect console cable from device

STEP 2: Configure basic setting for each switch.

a. From desktop tab on each PC, use terminal to console into each switch
b. Enter configuration mode
c. Assign class as Priviliged EXEC encrypted.

e. assign cisco as console

f. Assign cisco as vty.

g. Encrypt plaintext passwords.

h. Create a banner that warns anyone accessing the device

i. configure the IP address listed in address table from VLAN1 on switch

j. Shut down all interfaces that will not be used

k. set the clock on each switch

l. Close configuration window

Step 3 Configure PC hosts

Step 4 Test connectivity.

Part 2: Create VLANS & assign switch ports.

Step 1: create VLANS on switches

a. create VLANS on S1

b. create the same VLANS on S2

c. Issue the show VLAN brief command to view list of VLANS on S1

Step 2:

a. Assign VLANS to correct switch interface

1) assign PC-A to operation VLAN

2) From VLAN1, remove management IP address & configure it on VLAN 99

c) Issue the ip interface

d) assign PC-B to operations VLAN on S2

e) From VLAN1, remove management IP address & configure it on VLAN 99.

f) use show VLAN brief command to verify that the VLANS assigned to one correct interface

PART-3: maintain VLAN port assignment & VLAN

S-1 Assign a VLAN to multiple interface from desktop tab on each PC, use terminal to continue config both network switches.

S-2 Remove a VLAN ID from VLAN database

a. Add VLAN 30 to interface Fo/24 without issuing global VLAN

b. Verify that the new VLAN is displayed in VLAN table

c. use no VLAN 30 command to remove VLAN 30

PART 4 Configure an 802.1Q trunk b/w switches

Step 1. use DTP to initiate trunking on Fo/1

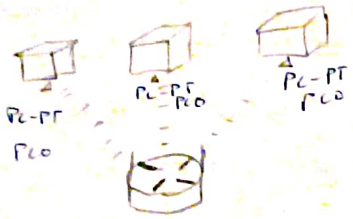
Step 2 manually configure trunk interface Fo/1

2011

RESULT:

Thus, VLAN configuration using Cisco packet tracer is executed successfully.

b) Configuration of wireless LAN using Cisco packet tracer.



Step 1: Click on a wireless router.

- Get username & password to admin & click on same settings.
- Next Click on wireless tab & set default SSID to mother network
- Again go in end of page & click on save settings

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

- Click on connect you with wireless router.
- PC1 card is active
- ~~Request from~~ Repeat same process.

STUDENT OBSERVATION:

c) SSID is name of wireless network. When you connect to wifi network, you typically see a list of available SSID's, which are names of wireless network being broadcast as nearby routers or access points.

d) Security key:

It is a password that is used to protected a wireless network. It is essential for preventing unauthorized access, securing data, transmission & protecting the network from potential attacks.

e) - gather equipment
- Connect to access point
- Access router configuration page
- Configure SSID & security settings
- Set up IP address & DHCP settings
- Save & reboot.

20/4

RESULT: Thus, the configuration of wireless LAN using Cisco packet tracer is executed successfully.

~~Aim~~

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 address ing allowing for subnet mask that are not just default IP masks. Which can be useful when we have a limited number of IP addresses that need to create multiple network.

Creating network topology:

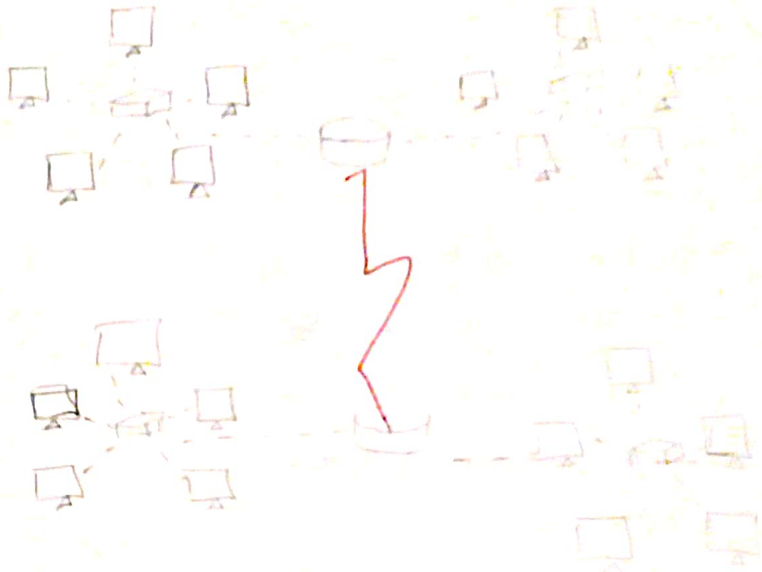
Implementing Classless IP subnetting is to create a network topology.

Adding devices:

ONCE we have created our network topology, we will be adding routers, switches & PC's.

Subnetting:

To provide enough space for end devices, switch, routers, This will give us 8 subnets.



Configuration of Devices.

enable
configure terminal
interface fast ethernet 0/0
ip address {IP address} {subnet mask}
no shutdown
exit.

interface fast ethernet 0/1
ip address {IP address} {subnet mask}
no shutdown
exit.

Next, we will configure switch. Right click on the switch & select "CLI", enter following commands.

enable
configure terminal
interface fastethernet 0/1
switchport mode access
exit
interface fastethernet 0/2
switchport mode access
exit

Gigabit ethernet interface on the router:

1. Right click on router & select "CLI".
2. Enter the following command.

enable
~~Exe~~
Interface gigabit ethernet 0/0
ip address [IP address] [subnet mask]
no shutdown
exit

Testing network.

Open a command prompt on each PC & try to ping other PC. We can use the "ping" command to test connectivity between router & the PCs.

Student Observation

a) It is process of dividing a large network into smaller, more manageable subnetworks.

In subnetting, we modify default subnet mask to allocate IP address to different network segment.

b) • Improved network management
• Enhanced security
• Efficient IP address utilization
• Subnetting implementation in college
• Reduced network traffic.

c) • Research mapping
• Subnet diagram & IP address list

RESULT:
Thus, to implement subnetting in CISCO PACKET TRACER simulator

PRACTICAL-10

AIM:

a) Internetworking with routers in CISCO Packet Tracer simulator.

In the network, a router & 2 PC's are used. After forming the network, to check network connectivity a simple PDU is transfered from PC0 to PC1

PROCEDURE:

STEP-1:

1. Select router & open CLI
2. press enter to start configuring router1
3. Type enable to activate Privileged mode.

STEP-2:

1) Assign IP to every PC in network
2) select PC, go to desktop & select IP configuration & assign an IP address.
3) Assign the default gateway of PC0 as 192.168.10.1
4) Assign the default gateway of PC1 as 192.168.10.1

STEP-3:

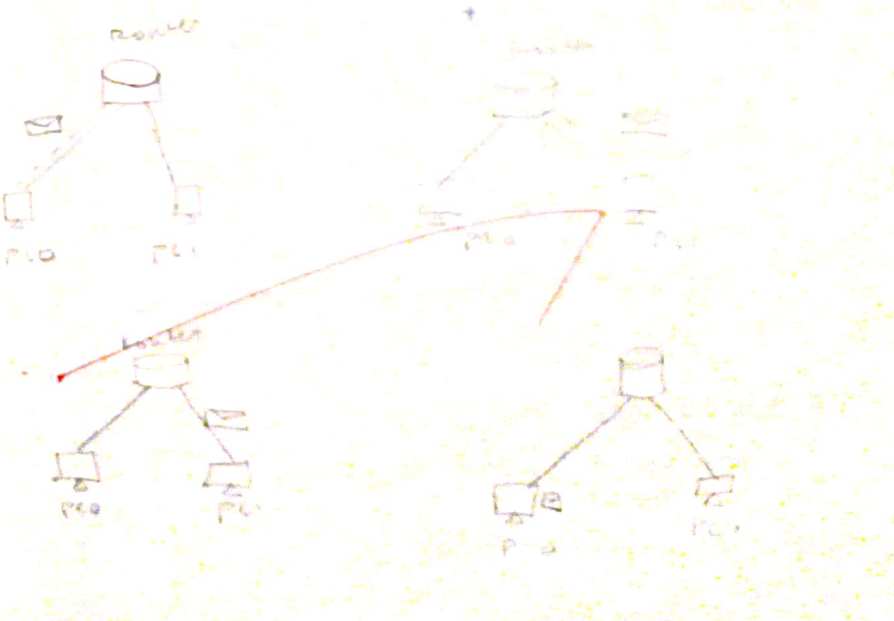
1. Fast ethernet 0/0 port of Router1
2. Fast ethernet 0/1 port of router1

Router Configuration Table:

Device Name	IP address 0/0	Subnet mask	address of 1	subnet mask
Router 1	192.168.0.1	255.255.255.0	192.168.0.1	255.255.255.0

PC configuration Table.

Device name	IP address	Subnet mask	Gateway
PC0	192.168.10.2	255.255.255.0	192.168.10.1
PC1	192.168.20.2	255.255.255.0	192.168.10.1



Addressing Table:

Device	Interface	IP address	Subnet mask	Default gateway
PC	ethernet 0	DHCP		192.168.0.1
Wireless Router	LAN	192.168.0.1	255.255.255.0	
wireless router	Internet	DHCP		
CISCO comserver	ethernet 0	208.67.220.210	255.255.255.0	
Laptop	wireless	DHCP		

PART 1 : Build a simple network.

STEP 1 : Launch Packet tracer.

STEP 2 : Build the topology

a) To place a device onto the workspace, first choose a device type from Device-type section box.

b) Change display names of network devices

c) Add the physical cabling between devices on workspace

Part 2:

STEP-1: Configure wireless network
Create a wireless network on wireless router

STEP-2 : Configure the laptop
Configure laptop to access the wireless network

STEP-3: Configure the PC
Configure PC for wired network

STEP-4: Configure Internet Cloud
- Identify the form A to ports
- Identify the type of provider

STEP-5: Configure Cisco.com server.

a. Configure cisco.com server as DHCP server

b. Configure cisco.com server as DNS server

c. Configure cisco.com server global settings

PART-3:

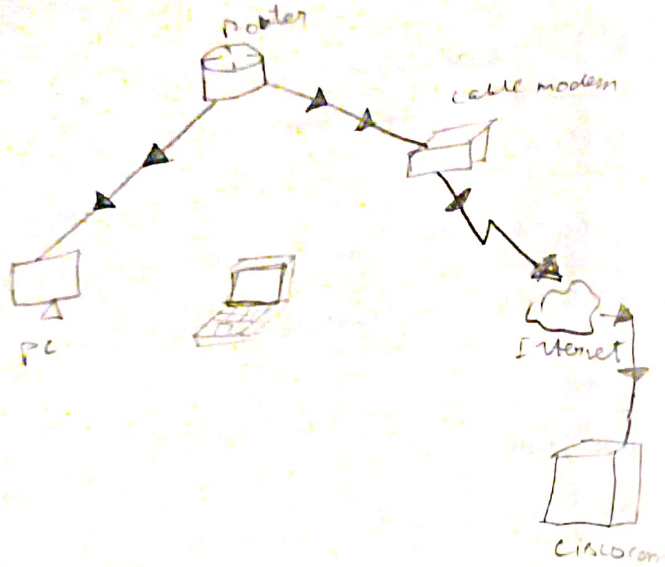
STEP-1. Refresh IPV4 setting on PC

a) Verify the PC is receiving IPV4 configuration information from DHCP

Result:

Thus, Implementation of routers in CISCO packet tracer has been executed successfully.

a)



- DHCP Server Configuration:

Define the IP address range that the DHCP will assign to devices.

Specify the subnet mask to define network boundaries.

Set duration for which an IP address is assigned to a device

- Significance of DHCP

The DHCP is ~~so~~ crucial in internetworking because it simplifies & automates the process of assigning IP addresses to device in network.

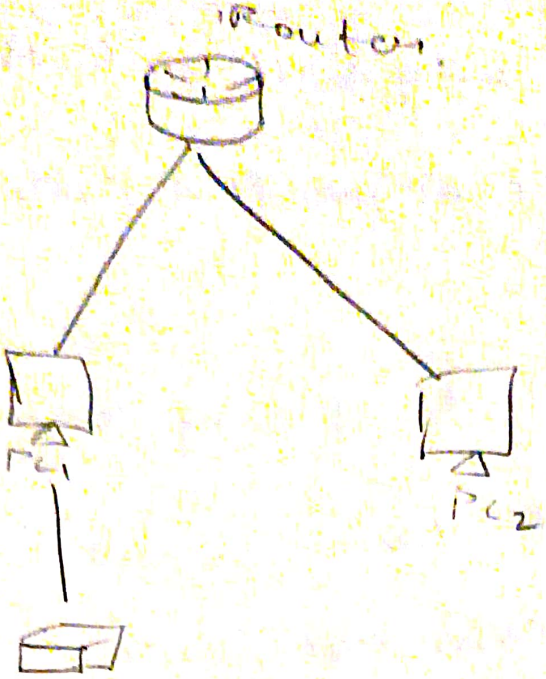
DHCP dynamically assigns IP addresses, reducing manual configaration & preventing IP conflicts.

- DHCP server make it easy to add & manage multiple devices across large networks, an essential feature in growing environments

3 Design & configuration of an inter-network in a lab:

1. Hardware requirements
   - one switch
   - One Router
   - Ethernet labels
2. Network layout
   - Router
   - Switch
   - Devices
3. configuration steps

(i) connect to router.
(ii) configure router's interface with an IP address
(iii) configure DHCP server on router
(iv) Connect PC's to switch



2014

RESULT:

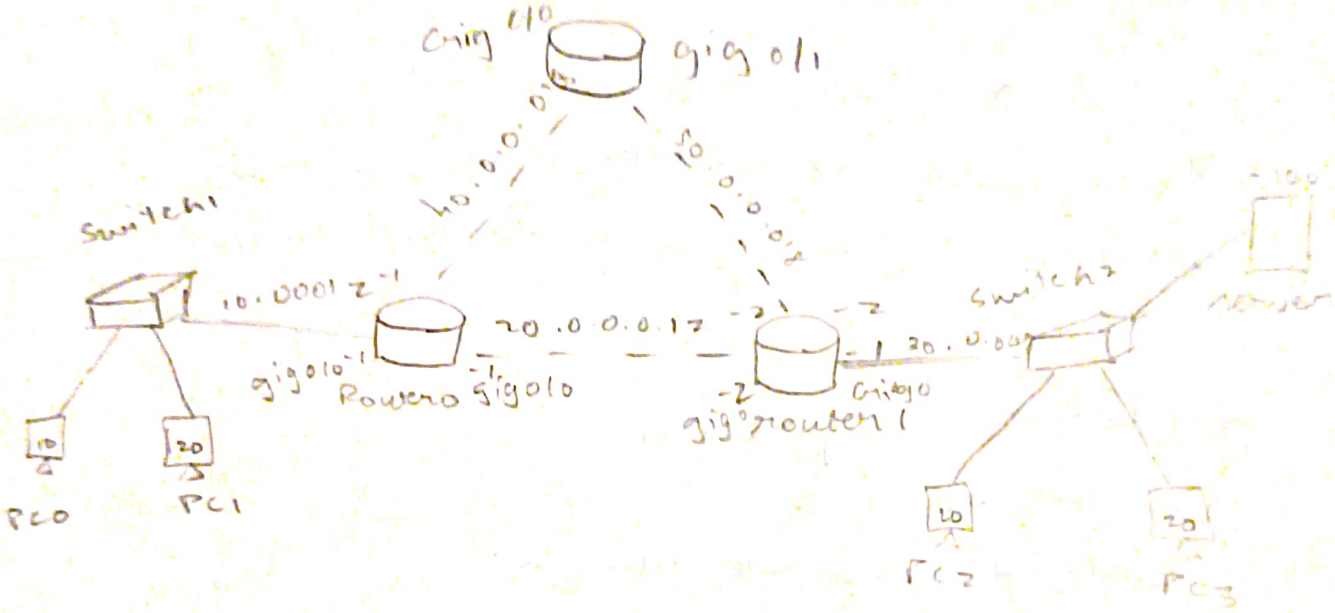
Thus, the implementation of interv

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 routing table is known as static routing.



Creating, adding, verifying static Routes

Routes automatically learn their connected networks, we only need to add routes for network.

Router 0

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

Router 1 — 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

Router 2 — 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

Router 0 requirement

- Create 2 routes for networks 30.0.0.0/8
  & configure the first route as main route & the second route as backup route

Router 0 configuration

Router > enable

router # configure terminal

Enter configuration command

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 20.0.0.0 20

Router # show ip route static

S 30.0.0.0/8 [1/0] via 20.0.0.2

S 30.0.0.0/32 [10/0] via 40.0.0.2

Router #

Routers requirements

- Create two routers for network 10.0.0.0/8 & configure
- Create two routers for network 40.0.0.0/8 & configure

Routers Configuration

Router enable

Router # configure terminal

Enter configuration command, one per line

Router (config)# ip route 10.0.0.0 255.0.0.0 30.0.0.1 10

Router (config)# ip route 10.0.0.0 255.0.0.0 50.0.0.1 20

Router (config)# exit

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 #

Router 2 requirements:

Create static routers for network 10.0.0.0/8

& network 30.0.0.0/8 & verify routers

Router 2 Configuration

Router > enable

Router # configure terminal

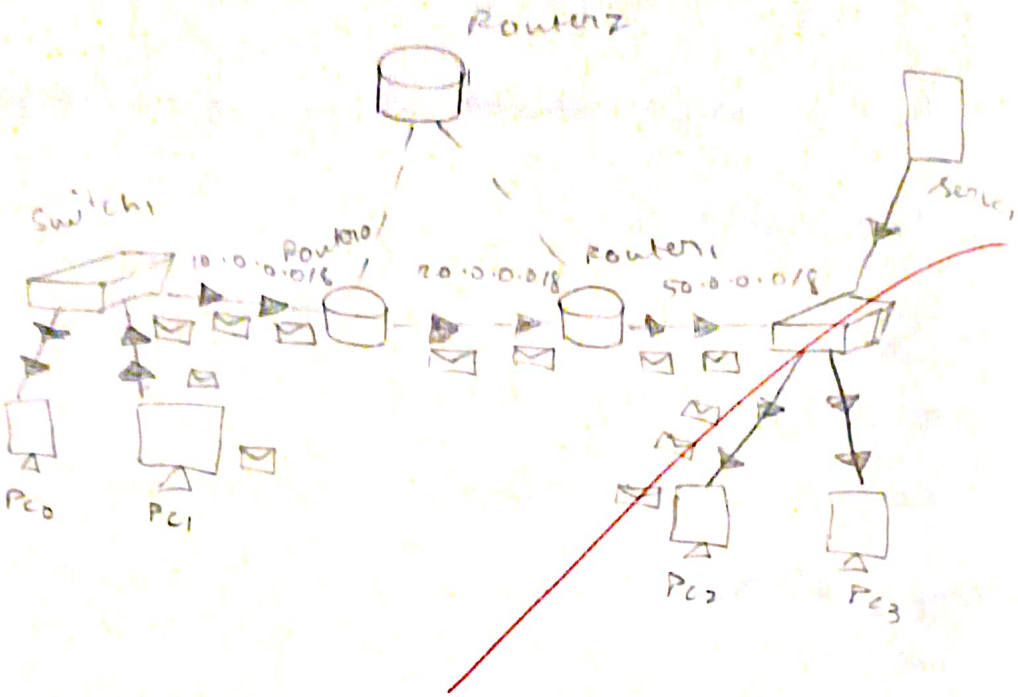
Router (config)# ip route 10.0.0.0 255.0.0.0 40.0.0.1

Router (config)# ip route 30.0.0.0 255.0.0.0 50.0.0.2

Router (config)# exit

Router # show ip route static

Verify static routing:



RESULT: 20/11/

Thus the connection to simulate static routing configuration using CISCO PACKET TRACER is configured.

PRACTICAL - 12

AIM:

Implement Echo client server using TCP/UDP sockets.

TCP echo client - server algorithm.

SERVERS:

1. Create a TCP socket
2. Connect socket to a local address & port
3. Listen for incoming client connections
4. Accept a client connect
5. loop
   - receive data from client
   - If data is received, send it back to the client.
6. Close connection.

CLIENT:

1. Create a TCP socket
2. Connect to the server using specified address & port
3. send a message to server
4. Receive the echo message from server
5. Display the received message
6. Close socket.

TCP server.py

```
import socket
def tcp-server()
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(("localhost",12345))
server_socket.listen(1)
print("TCP server is waiting for connection")
connect, client_address = server_socket.accept()
Print(f"connected to {client_address}")
try:
  while true:
    data = connection.recv(1024)
    if data:
    print(f"received: {data.decode()}")
  else:
      break
  finally:
      connection.close()
```

TCP client.py.

```
import socket
def tcp-client():
  client_socket = socket.socket(socket.AF_INET)
  Client_socket.connect(("localhost",12345))
try:
message=input("Enter a message to send")
  client_socket.sendall(message.encode())
    data=client.socket.recv(1024)
    Print(f"Received from server: {data.decode()}")
finally: Client_server.close()
if __name__=
```

tcp-client()

OUTPUT:

```
> Python tcp-client.py
  Enter a message to send: Hi, I am Roxanne
  Received from server: Hi, I am Roxanne

> Python tcp-server.py
  TCP server is waiting for a connection
    Connected to ('127.0.0.1', 56893)
  Received: Hi I am Roxanne
```

RESULT:

Thus, the program to implement echo Client server using TCP is executed successfully.

```
chat_client.py

import socket
import threading
def receive_message(client_socket):
while True:
    try:
        message = client_socket.recv(1024).decode('utf-8')
        if message:
            print(message)
    except exception as e:
        print(f"An error occurred: {e}")
        break
```

```
chat_server.py

import socket
import threading
def handle_client(client_socket):
    while True:
        try:
            message = client_socket.recv(1024).decode('utf-8')
            if message:
                print(f"Received message: {message}")
                client_socket.send("Message received".encode('utf-8'))
```

```
print(f" an error occured : {e}")
finally:
    server.close()
def start_chat_server():
    server = socket.socket(socket.AF_INET,
        socket.socket.sock_stream)
    server.bind(('127.0.0.1', 12345))
    server.listen(5)
print("chat server has started on
    127.0.0.1  12345")
while true:
    client_socket, addr = server.accept()
print(f"new connection from {addr}")
client_handler = threading.thread(target
handle_client, handle = start()
        = (client_socket)
if __name__ == "__main__"
        start_server()
```

OUTPUT:

```
>Python chat_server.py
 chat server started on 127.0.0.1:12345
new connection from('127.0.0.1', 57226)
 Received from client: Hi
type from message to client : received
>Python chat_client.py
    connected to chat server
    You : Hi there
    You server: received
```

RESULT: Thus, the program to implement the client server using tcp is executed successfully.

Aim:

Implement your own ping program

Algorithm:

- Open a raw socket to send ICMP request
- Create icmp echo request packet including a header & data
- Calculate time
- Show response

```
server_script.py
import socket
def start_server(host='127.0.0.1', port=12345)
 with socket.socket(socket.AF_INET)
    socket.SOCK_DGRAM) as s
    s.bind(("host, port)
print(f"UDP server running on {host} {port}
 while true:
     data, addr = s.recvfrom(1024)
  print(f"Received message from {client}:
    {data.decode()}")
     s.sendto(b'pong', addr)
 if __name__ == "__main__":
    start_server()
```

```
Client_script.py
import socket
def start_server(host='127.0.0.1', port=12345)
  with socket.socket(socket.AF_INET,
     socket.SOCK_DGRAM) as s:
try:
     s.print((host, port))
     print(f"UDP server running on {host}:{port
```

while true:

data = addr = s.recvfrom(1024)

print(f"UDP server running on

{host}: {

s.sendto(b'ping', addr)

except socket.timeout:

print("Request timed out")

OUTPUT:

>python ping-server.py

UDP server running on 127.0.0.1 : 12345

Received message from(127.0.0.1, 5734)

>python ping-client

Received ping- from ('127.0.0.1', 12345) in

0.00seconds

Result:

Thus, the program to implement the program is executed successfully.

Ex: 14

Aim:

Write a code using RAW socket to implement packet sniffing

Algorithm:

- Create a raw socket
- Continously capture Unopened packet using recvfrom()
- parse & display information like the source & destination IP address & protocol type
- Close the socket after

CODE:

```
from scapy.all import sniff
from scapy.layers.inet import IP, TCP, UDP, ICMP
def packet_callback(packet):
    if IP in packet:
        IP_layer = packet[IP]
        protocol = IP_layer.proto
        src_IP = IP_layer.src
        dest_IP = IP_layer.dst
    protocol_name = ""
    if protocol == 1:
        protocol_name = "ICMP"
    elif protocol == 6:
        protocolname = UTCP
    elif protocol == 17:
        protocol_name = "UDP"
```

```
else:
protocol_name = "Unknown protocol"
print(f"protocol : {protocol_name}")
print(f"Source IP : {Source_IP}")
print(f"Destination IP : {dest_IP}")
print("-"*50)

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

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

OUTPUT:

```
Protocol : TCP
Source IP : 192.168.1.2
Destination IP : 93.184.216.34
-----------------------------------
Protocol : TCP
Source IP : TCP 192.168.1.2
Destination IP : 172.217.14.206
-----------------------------------
Protocol : TCP
Source IP : 192.168.1.2
Destination : 172.217.14.206
```

RESULT

Thus, the using of RAW socket to implement packet sniffing is executed successfully.

Ex: 15

AIM:

To analyze the difference types of each Logs using webalizer tool

Procedure:

S1: Run webalizer window version

S2: Input weblog file

S3: Press run webalizer.

choose logfiles	logfile	view	settings	additional HTML code	HTML

Input:

Logfiles:

C:\user\TCS1\download\access.log

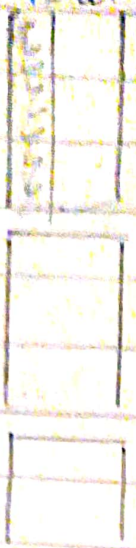
Target directories:

C:\user\TCS1

Clear existing directory

delete all files selected target directory.

17 Daily usage for



29

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21

Day	Hits	Files	Pages	visitors	Site	K bytes
3	69.17%	36 1000	1110 0071	110000.1	10000·/·	135·100·0

Monthly Status for October 2024

Total hits		
Total files		99
Total pages		98
Total visits		97
Total K bytes		18
Total unique reference		3
Total unique URL's		7
Hits per hour		3
hits per day	Avg	ne
visits per day	2	14
K bytes per day	4·9	247
	·0	1
	181	90

RESULT: 30/11

Thus the procedure to analyse the different type of web logs using website