INDEX

Name : K. Chithra Sub : CN

Std : Div : Roll No :

Telephone No : E-mail ID :

Blood Group : Birth Day :

	Sr. No.	Title	Page No.	Sign. / Remarks
1.	13.7.24	Study of various commands used in Linux and windows		
2.	20.7.24	Study of network cables		
3.	30.7.24	Experiments on CISCO PACKET TRACER		
4.	10.8.24	Setup and configure a LAN		
5.	13.8.24	Packet capture tool: Wireshark		
6.	19.8.24	Hamming code		
7.	26.8.24	Sliding Window Protocol.		
8.a)		Virtual LAN Configuration		
8.b)		Wireless LAN Configuration		
9.		SUBNETTING		
10.a)		Net work with Routers		
10. b)		Wireless Routers DHCP.		
11.a)		Static Routing Configuration		
11.b)		RIP		
12)a)		echo client-server		
12)b)		chat client-server		
13)		Ping program		
14)		Packet sniffer		
15)		webalizer		

2024/11/20 14:14

Completed.

13/7/24

Practical-1.

Aim:

Study for various network commands used in linux and windows.

Basic networking commands:

1. arp -a

Interface: 172.16.75.36 --- 0x12

Internet Address	Physical Address	Type
172.16.72.1	7C-5a-1C-Cf-6e-41	dynamic
224.0.0.2	01-00-5e-00-00-02	Static
224.0.0.22	01-00-5e-00-00-16	static
224.2.2.2	01-00-5e-02-02-02	static

2. hostname

DESKTOP-C01BH7D

3. ipconfig /all

Windows Ip Configuration

Host name. /all. . . . . : DESKTOP-C01BH7D
Primary Dns Suffix. . . . . :
Node type. . . . . : Hybrid
IP Routing Enabled. . . . : No
WINS Proxy Enabled. . . . : No

4. Nbtstat -a

Active Connections

Proto	Local Address	Foreign Address	State
TCP	127.0.0.1:49678	DESKTOP-C01BH7D:49679	ESTABLISHED
TCP	127.0.0.1:49679	DESKTOP-C01BH7D:49678	ESTABLISHED
TCP	172.16.75.36:49525	20.198.118.190:https	ESTABLISHED
TCP	172.16.75.36:49567	14.98.16.11:http	ESTABLISHED
TCP	172.16.75.36:59925	maa05s24-in-f10:https	ESTABLISHED

5. netstat -a

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

NBTSTAT
-a (adapter Status) Lists the remote machine name table per its name

2024-07-20 14:14

6. nslookup

nslookup www.google.com
Server: Unknown
Address: 172.16.72.1

Non-authoritative answer:
Name: www.goog[redacted]
Addresses: 2404:6800:4001:81c::2004

7. pathping

usage : pathping [-g host-list][-h maximum hops] [-i address] [-n]

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

1. #ping hostname

2. #ping ip address

3. # ping fully qualified domain name

usage : ping [-t][-a] [-n count][-l size][-f][-i TTL][-v TOS]
[-r count][-s count][[-j host-list]|[-k host-list]]

options:

-t ping the specified host until stopped.
To see statistics and continue -type control-Break;
-a Resolve addresses to hostnames
-n count number of echo requests to send.
-l size send buffer size.
-i TTL Time to Live

1) Ping DESKTOP-C01BH7D

Pinging DESKTOP-C01BH7D [fe80::71f9:4f11:4275:cc63%18] with 32 bytes:
Reply from fe80::71f9:4f11:4275:cc63%18: time<1ms
Ping statistics for fe80::71f9:4f11:4275:cc63%18:
Packets: Sent = 4, Received = 4, Lost = 0.

2) ping 172.16.75.36

Pinging 172.16.75.36 with 32 bytes of data:
Reply from 172.16.75.36: bytes=32 time<1ms TTL=128

3) ping www.facebook.com

Pinging star-mini.c10r.facebook.com [157.240.192.35] with 32 bytes of data:
Reply from 157.240.192.35: bytes=32 time=6ms TTL=59

Some important Linux networking

9. Route

Manipulates network routing tables

Route [-f][-p][-4|6] Command [destination]

-f clears the routing tables of all gateway entries.
-p when used with the ADD command.

2024/11/20 14:15

Some important Linux networking commands.

1) ip

usage: ip [options] OBJECT {command | help}
ip [-force] -batch filename

where OBJECT := { link | address | addrlabel | route | rule | neigh | ntable | tunnel | tuntap | maddress | mroute | mrule | monitor | xfrm | netns | l2tp | fou | macsec | tcp_metrics | token | netconf | ila | vrf }

ip <options> <object> <commands>

a) 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
2. enp2s0: <BROADCAST, MULTICAST, UP, LOWER_UP> mtu 1500 qdisc fq_codel state up group default qlen 1000 link/ether 50:9a:4c:35:0f:75 brd ff:ff:ff:ff:ff:ff

b) ip address add 192.168.1.254/24 dev enp2s0
To assign an Ip to an interface

c) ip address del 192.168.1.254/24 dev enp2s0
after the status of interface by bringing etho online

d) ip link set ~~eth0~~ do up
after the status of interface by bringing interface etho online

e) ip link set lo down
after the status of interface etho online

f) ip link set do promisc on
after the status of interface by enabling promiscuous mode for etho

g) ip route add default via 192.168.1.254 dev enp2s0
ip address add 192.168.1.254 dev enp2s0
add a default route via the local gateway

h) ip route add 192.168.1.0/24 via 192.168.1.254
add route to 192.168.1.0/24 that can be reached or device etho via gateway 192.[redacted]

i) ip route add 192.168.1.0/24 dev enp...
Add a route to 192.168.1.0/24 that can be reached on device etho

j) ip route del 192.168.1.0/24 via 192.168.1.254
Delete route for 192.168.1.0/24 via the gateway 192.168.1.254

k) ip route get 10.10.1.4

10.10.1.4 dev enp2s0 src 192.168.1.254 uid 0
cache.

2) ifconfig

enp2s0: flags=4163<UP, BROADCAST, RUNNING, MULTICAST> mtu 1500 inet 172.16.10.228 netmask 255.255.252.0 broadcast 172.16.11.255 inet6 fe8::d7e4:f38f:9a5:d315 prefixlen 64 scopeid 0x20<link> ether 50:9a:4c:35:0f:75 txqueuelen 1000

lo: flags=73<UP, LOOPBACK, RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local loopback)

3) mtr

Host	Packets		Pings				
	Loss %	Snt	Last	Avg	Best	Wrst	StDev
1. ::1	0.0%	117	0.1	0.1	0.1	0.1	0.0

a) mtr google.com

localhost.localdomain (0.0.0.0)

Keys: Help Displaymode Restart Order of fields quit

2024/11/20 14:16

Host			
1.	172.16.8.1		
2.	static-41.229.429.49-tataidc.co.in		
3.	142.250.171.162		

b) mtr -g google.com

c) mtr -b google.com

localhost.localdomain

Keys: Help · Display mode Restart Statistic order of quit

Packets Pings

	Loss%	Snt	Last	Avg	Best	Wrst	StD
1. 172.16.8.1	0.0%	163	0.2	0.1	0.1	0.4	0
2. 142.250.171.262							
3. 142.25.227.217							

d) mtr -c 10 google.com

localhost.localdomain

Keys: Help display mode Restart Statistic order of fields quit

Host

1. 172.16.18.1

2. 142.250.171.162

4) tcpdump

tcpdump: verbose output suppressed, use -v or -vv for full protocol decode

09:14:22.071728 IP 172.16.8.215.mdns > mdns.mcast.net.mdns:

09:14:22.071758 IP6 fe80::7cfc:d02f:f1a4:3c72.mdns

09:14:22.071849 IP 172.16.8.215.mdns > mdns.mcast.net.mdns

a) dnf install -y tcpdump

Package tcpdump-14:4.9.0-7.fc26.i686 is already installed, skipping.

Nothing to do

Complete!

2024/11/20 14:16

b) tcpdump -D

1. enp2s0 [up, Running]
2. any (Pseudo-device that captures on all interfaces)
3. lo [up, Running, loopback]
4. wlp3s0 [up]
5. bluetooth0 (Bluetooth adapter number 0)
6. bluetooth-monitor
7. usbmon0
8. nflog

c) tcpdump -i lo

verbose output suppressed, use -v or -vv for full protocol decode

09:15:44.541275 IP localhost.localdomain > localhost.localdomain: ICMP host www.time.pl.india.org unreachable

09:15:45.693167 IP localhost.localdomain > localhost.localdomain: ICMP host maa03s45-in-f14.1e100.net, length 68

d) tcpdump -i lo -c 10

verbose output suppressed, use -v or -vv

listening on lo, link-type EN10MB, capture size 262144

09:17.46.013128 IP localhost.localdomain > localhost.localdomain: ICMP host maa05s22-in-f14.1e100.net unreachable, length 68

09:17.46.013141 IP localhost.localdomain > localhost.localdomain: ICMP host maa05s22-in-f14.1e100.net unreachable, length 68

e) tcpdump -i eth0 -C 10 host 8.8.8.8

verbose output suppressed, use -v or -vv for full protocol decode listening on lo, link-type EN10MB,

2024/11/20 14:17

capture size 262144 bytes

f) tcpdump -i lo dst host 8.8.8.8
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode listening on lo, link type EN10MB, Capture size 262144 bytes

g) tcpdump -i etho not 10.1.00/24
tcpdump: Verbose output Suppressed, use -v or -vv for full protocol decode listening on lo, link type EN10MB, Capture Size 262144 bytes.

h) tcpdump -i etho port 53
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode listening on lo, link type EN10MB, capture size 262144 bytes.

i) tcpdump -i etho host 8.8.8.8 and port 53
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode listening on lo

j) tcpdump -i lo -c 10 host www.google.com and port 443
To Capture only HTTPS traffic.

k) tcpdump -i lo port not 53 and not 25
To capture all port except port 80 and 25.

5) Ping
usage: Ping [-aAbBdDfhLnOqrRUvV64] [-c count][-i interval]
[-I interface][-m mark][-M pmtudisc option]
[-l preload][-p pattern][-Q tos][-s packetsize]
[-S sndbuf][-t ttl][-T timestamp-option][-w deadline][-W timeout] destination.

2024/11/20 19:17

a) ping google.com

PING google.come (142.250.182.14) 56(84) bytes of data

64 bytes from maa05s18-in-f14.1e100.net (142.250.182.14):

icmp-seq=1 ttl=120 time=3.21 ms

64 bytes from maa05s18-in-f14.1e100.net (142.250.182.14):

icmp-seq=2 ttl=120 time=3.21 ms

64 bytes from maa05s18-in-f14.1e100.net (142.250.182.4):

icmp-seq=3 ttl=120 time=3.17 ms

b) ping -c 10 google.com

PING google.com (142.250.182.14) 56(84) bytes of data

64 bytes from maa05s18-in-f14.1e100.net (142.250.182.14):

icmp-seq=1 ttl=120 time=3.19 ms

64 bytes from maa05s18-in-f14.1e100.net (142.250.182.14):

icmp-seq=2 ttl=120 time=3.31 ms

64 bytes from maa05s18-in-f14.1e100.net (142.250.182.14):

icmp-seq=3 ttl=120 time=3.29 ms

Configuration an ethernet connection by using nmcli

1. nmcli Connection show

NAME	UUID	TYPE	DEVICE
wl	f8ac812f-cf23-4aa5-81ae	802-3-ethernet	enp2s0

2. nmcli connection add con-name RECAA ifname type ethernet

Successfully added.

3. ip address show enp2s0

enp2s0: <BROADCAST, MULTICAST, UP, LOWER_UP> mtu 1500 qdisc fq_codel state up group default qlen 1000

4. ip route show default

default via 172.16.8.1 dev enp2s0 proto static metric 100

172.16.8.0/22 dev enp2s0 proto kernel scope link src 172.16.9.155

2024/11/20 14:18

Result:

Thus the study of various commands in Linux and windows are studied.

27.7.24

Practical-2

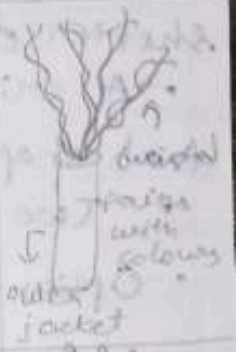
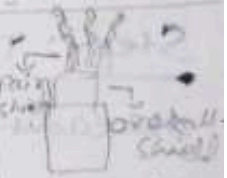
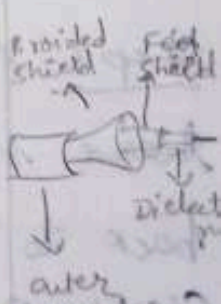
Aim:

Study of different types of Network cables.

a) Understand different types of network cables.

Different type of cables:

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

Cable type	Category	Maximum Data Transmission	Advantages / Disadvantages	Application / Use	Image
UTP	Category 3	10 bps	Advantage • Cheaper in cost • Easy to install Disadvantage • More prone to (EMI)	10 Base-T Ethernet	
	Category 5	up to 100 Mbps		Fast Ethernet, Gigabit	
	Category 5e	1Gbps		Fast & Gigabit Ethernet	
STP	Category 6 & 6a	100Gbps	Advantage • Shielded • Faster than UTP Disadvantage • Expensive • Greater installation effort	Gigabit Ethernet 10G Ethernet (55m)	
SSTP	Category 7	10Gbps		Gigabit Ethernet 10G Ethernet (55m)	
Coaxial Cable	RG-6 RG-59 RG-11	10-100 Mbps	• High Bandwidth • Immune to interference • Low Loss Disadvantage • Limited distance • Cost • Size is bulky	Speed of Signal is Boom Tele Vision network High speed internet Connections	

fibre optics Cable	single mode multi mode	100Gbps	Advantage • High speed • High security • Long distance Disadvantage • Expensive • Requires skilled installers	Maximum distance of fibre optics Cable is around 100 meters	

b) Make your own Ethernet Cross-Over cable / Straight cable-

Tools and parts.

- Ethernet Cabling, CAT5e is certified for gigabit support, but CAT5 Cabling works as well, just over shorter distances.
- A crimping tool, This is an all-in one networking tool shaped to push down the pins in the plug.
- Two RJ 45 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 Referening back to the sheet, arrange them top to bottom.

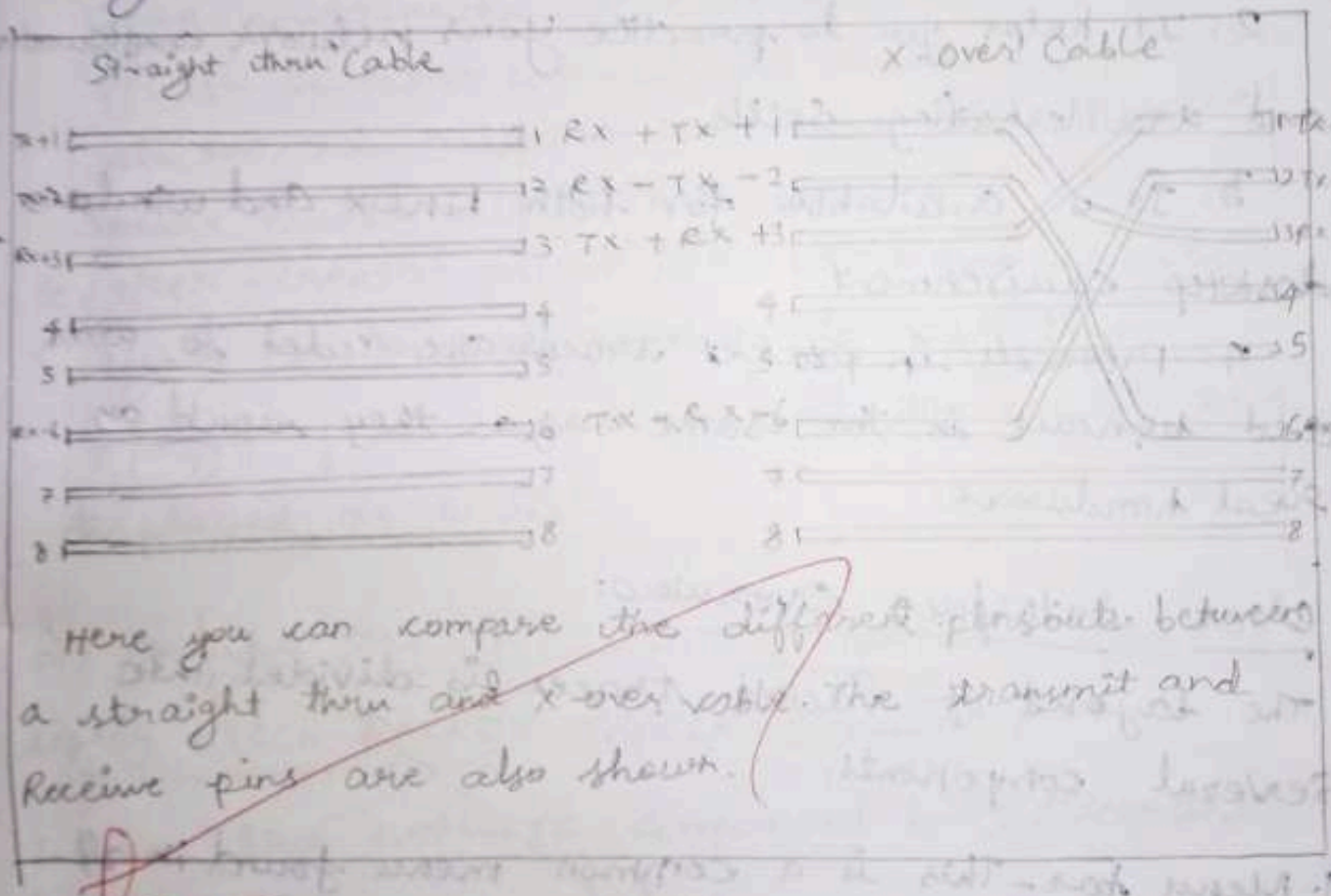
Step 4: Once the order is correct, bunch them together in a line and if there are any that stick out farther than others, snip them back to create an even level. The difficult aspect is placing these into the RJ45 plug without messing up

the order. To do so, hold the plug.

Steps: Next, push the cable right in. The notch at the end of the plug needs to be just over the cable shielding, and if it isn't, that means that you stripped off too much shielding.

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) or using diagram A.



Here you can compare the different pinouts between a straight thru and x-over cable. The transmit and Receive pins are also shown.

6/8

Result:

Thus types of network cables is studied.

2024/11/20 14:19

30.7.24 Practical -3.

Aim:

To study the Packet tracer tool Installation and User Interface Overview

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

INTRO:

1. It allows you to model complex systems with out the need for dedicated equipment.
2. It helps you to practice your network configuration and troubleshooting skills.
3. It is available for both linux and windows desktop environment.
4. Protocols in packet tracer are coded to work and behave in the same way as they would on real hardware

User Interface Overview:

The layout of Packet tracer is divided into several components.

1. Menu bar - This is a common menu found in all Software applications.
2. Main toolbar - This bar provides shortcut icons to menu options that are commonly accessed, such as open, save.
3. Logical / Physical workspace tabs - these tabs allow you to toggle between the logical and work areas.

4. Workspace - This is a area where topologies are created and simulation are displayed.
5. Common tools bar - This toolbar provides control for manipulating topologies, such as select, move layout, place, delete.
6. Real-time/Simulation mode - These tabs are used to toggle between real and simulation modes.
7. Network component box - this component contains all the network and end devices available with packet Tracer, and is further divided into two areas. Area 7a: Device type selection box - This area contains device categories. Area 7b: Device specific selection box - when a device category is selected, this selection box displays the different device models within that category.
8. User-created packet box - Users can create highly customized packets to test their topology from this area, and the results are displayed as a list.

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

1. From the network component box; click and drag and drop the below.
   a) 4 Generic PCs 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 and connect it

using the cable. The link LED should glow in green, indicating that the link is up.
Similarly, connect remaining 3Pcs to HUB.

c) Similarly connect 4 Pcs to switch using copper straight through cable.

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

Click on PDU (message icon) from common tool bar.

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

4) Observe the ~~following~~ flow of PDU from source PC to destination PC by selecting the Realtime mode of Stimulation.

5) Repeat step #3 to step #5 for PCs connected to Switch.

6) Observe how HUB and Switch are forwarding the PDU and write your observation & conclusion about the behaviour of Switch & HUB.

2024/11/20 14:20

Result: 6/8

Thus the experiments on CISCO PACKET TRACER (Stimulation Tool) is executed.

Student Observation - Practical 1

1. To find the reachability of a host machine from your device?

   - command: `ping <hostname_or_IP_address>`

2. To get the details of hops taken by a packet to reach its destination?

   Command: traceroute <hostname_or_IP_address> (on Linux) or tracert <hostname_or_IP_address> (on windows)

3. To display the IP configuration of your machine

   - Command (Linux): ifconfig or ip a
   - windows: ipconfig

4. To display the TCP port status of your machine

   - Linux: `netstat -tuln` or `ss -tuln`
   - Windows: `netstat -an`

5. To modify the IP configuration on a Linux machine?

   Temporary changes:

   using ifconfig (old method)

   using ip (Modern method)

   Sudo ip address add 192.168.1.10/24 dev eth0

   Sudo ip route add default via 192.168.1.1

2024/11/20 14:21

Student observation - Practical 2

1. Difference between cross cable and straight cable?

Straight cable:

- used to connect different types of devices.
- The wiring standard on both ends is the same (either T568A or T568B).
- ex: switch, hub or router.

Cross Cable:

- used to connect similar types of devices.
- The wiring standard on one end is T568A and on the other end is T568B.
- ex: Connecting two Computers

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

- Straight cable

3. which type cable is used to connect a two PC?

- cross cable

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

cat 6 (Category 6): Supports use to 10Gbps for short distance

5. write down your understanding while making twisted pair, Cross / straight Cable.

Straight cable: - used to connect different types of network devices.
- both ends have same wiring order.

Cross cable: - used to connect similar types of network devices.
- one end follows TIA/EIA-568-B and the other follows TIA/EIA-568-A Standard.

Challenges faced: crimping issues, wire order, testing

output received: Successfully made cable.

Student observation – Practical 3.

a) Behaviour of switch and hub in forwarding Packets.

Switch:

- Packet Forwarding: uses MAC addresses to forward packets only to the intended recipient
- Efficiency: More efficient, as it reduces unnecessary traffic on the network
- Collision Domains: Each port on a switch is its own collision domain, reducing collisions and improving performance

Hub:

- Packet Forwarding: Broadcasts packets to all connected devices regardless of destination.
- Efficiency: Less efficient, as it increase unnecessary traffic.
- Collision Domains: All port on a hub share the same collision domain, leading to potential network slowdown.

b) Network Topology implemented in your college?

- observation: Look at how devices like computers, switches, routers, and hubs are interconnected.
- Documentation: Ask your college's IT department for a network diagram or description of network layout.
- Drawing: Draw the topology in your observation book based on your findings.

Common network topologies:

- Star: All devices are connected to central hub
- Bus: All devices share a single communication line
- Ring: Devices are connected in circular fashion.
- Mesh: Devices are interconnected with multiple paths between them.

2024/11/20 14:22

10/8/24 Practical-4:

Aim:

Setup and configure a LAN (Local area network) using a switch and Ethernet cable 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.

How to setup LAN.

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

② - take 4 computers, a switch with 8, 16, or 24 ports which is sufficient for networks of these sizes and 4 Ethernet cables.

③ - Connect your computers to network switch via an ethernet cable.

④ - Assign IP address to your PCs

-> log on to the client computer as administrator or as owner.

-> Right click local area connection Ethernet -> Go to properties -> select Internet protocol (TCP IPv4)

-> Click on properties -> select on the following ip address option and assign ip address

2024/11/20 14:22

⑤ - Configure a network switch:

-> Connect your computer to the switch to

Switch's web interface, you will need to connect your computer to the switch using an Ethernet cable.

c> log in to the web interface

c> configure basic settings

c> Assign IP address as 10.1.1.5, subnet mask 255.0.0.0

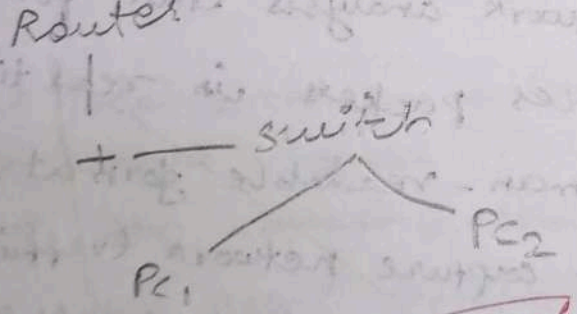
⑥ - Check the connectivity between switch and other machine by using ping command in the command prompt of the device.

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

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

Student observation:

Draw a neat diagram of the LAN in the configuration observation book.



PC_1 - IP address: 10.1.1.1, Subnet mask 255.0.0.0

PC_2 - IP address: 10.1.1.2, Subnet mask 255.0.0.0

PC_3 - IP address: 10.1.1.3, Subnet mask 255.0.0.0

Result: Thus the setup and configuration of LAN is successfully completed.

2024/11/20 14:23

13/8/24 Practical-5.

Aim:

Experiments on packet capture tool, wireshark.

Packet Sniffer:

↳ Sniffs message being sent / received from/by from computers.

↳ Share and display the contents of the various protocol field in messages.

- never sends packet itself
- no packet addressed to it
- receives a copy of all packets

Packet Sniffer Structure Diagnostic tools

↳ Tcpdump (eg: tcpdump -enrhost 10.129.41.2 -w

↳ wireshark (wireshark -r exe.out).

Wireshark:

wireshark, a network analysis tool formerly known as ethernet, captures packets in real time and display them in human-readable format.

→ what we can do - capture network traffic
- Decode packets protocols using dissecto vs
- Analyse problems

→ used for - people: learn network protocol internals intern network administrators: trouble shoot network problems.

2024/11/20 14:23

Getting wireshark: wireshark can be download for windows or mac os from the official website.

Capturing packets: After downloading and installing wireshark launch it and double check the name of a network interface under capture to start capturing packets.

The "packet list" pane: displays all packets in current capture

The "packet details" pane: Shows the current packet in a more detailed form

The "packet Bytes" pane: shows data of current packet

Colouring, Sample packets, Filtering packets, Inspecting packets, Flow Graph: Gives a better understanding of what we see.

Capturing and analysing packets using wireshark tool.

Procedure:

Select Local Area Connection in wireshark
go to capture -> option
select stop capture automatically after 100 packets
Then click start capture
Save the packets

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 packets.

2024/11/20 14:24

3. create a filter to display only DNS packets and inspect the provide flow graph.

4. create a filter to display only HTTP packets and inspect the packets.

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

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

Color coding:

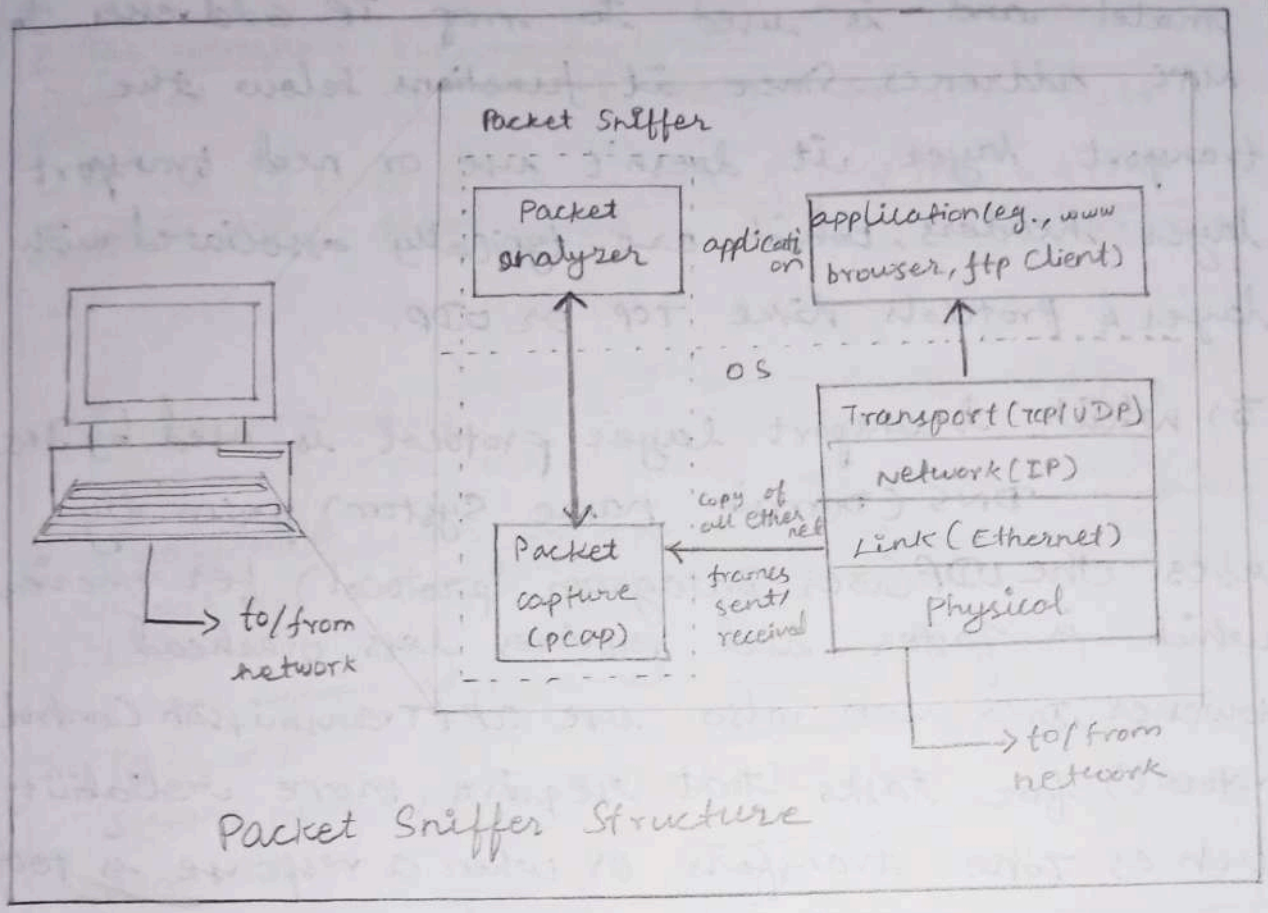
You'll probably see packets highlighted in a variety of different colours. Wireshark uses colors to help you identify the types of traffic at a glance. By default, light purple is TCP traffic, light blue is UDP traffic, and black identifies packets with errors.

Sample captures:

If there's nothing interesting on your own network to inspect, Wireshark's wiki has you covered. The wiki contains a page of Sample capture files that you can load and inspect.

Filtering Packets:

If you're trying to inspect something specific, such as the traffic a program sends when phoning home, it helps to close down all other application using the network so you can narrow down the traffic.



Packet Sniffer Structure

Student observation:

1. What is promiscuous mode?

Promiscuous mode is a network interface mode where the network interface card (NIC) passes all traffic it receives to the CPU rather than just the frames addressed to it. This mode is commonly used in network monitoring tools to capture and analyse all packets on a network segment.

2) Does ARP packets have a transport layer header? Explain.

No, ARP (Address Resolution Protocol) packets do not have a transport layer header. ARP operates at the data link layer (Layer 2) of the OSI

2024/11/20 14:25

model and is used to map IP addresses to MAC addresses. Since it functions below the transport layer, it doesn't use or need transport layer headers, which are typically associated with layer 4 protocols like TCP or UDP.

3) Which transport layer protocol is used by DNS?

DNS (Domain name System) primarily uses the UDP (User Datagram protocol) for queries, which is faster and requires less overhead. However, DNS can also use TCP (Transmission Control protocol) for tasks that require more reliability, such as zone transfers or when a response is too large for a single UDP packet.

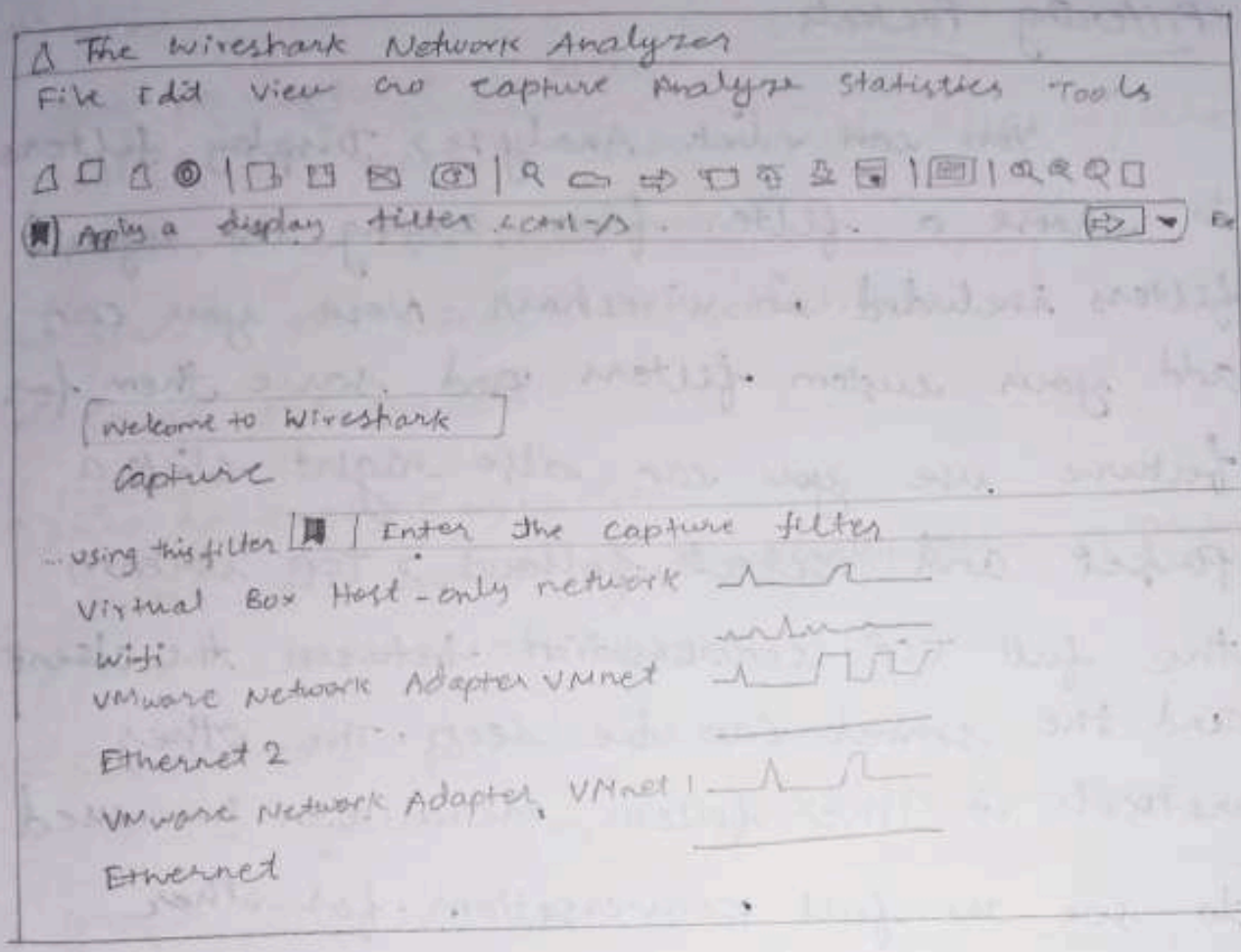
4) What is the port number used by HTTP protocol?

The HTTP (Hypertext Transfer Protocol) typically uses port 80.

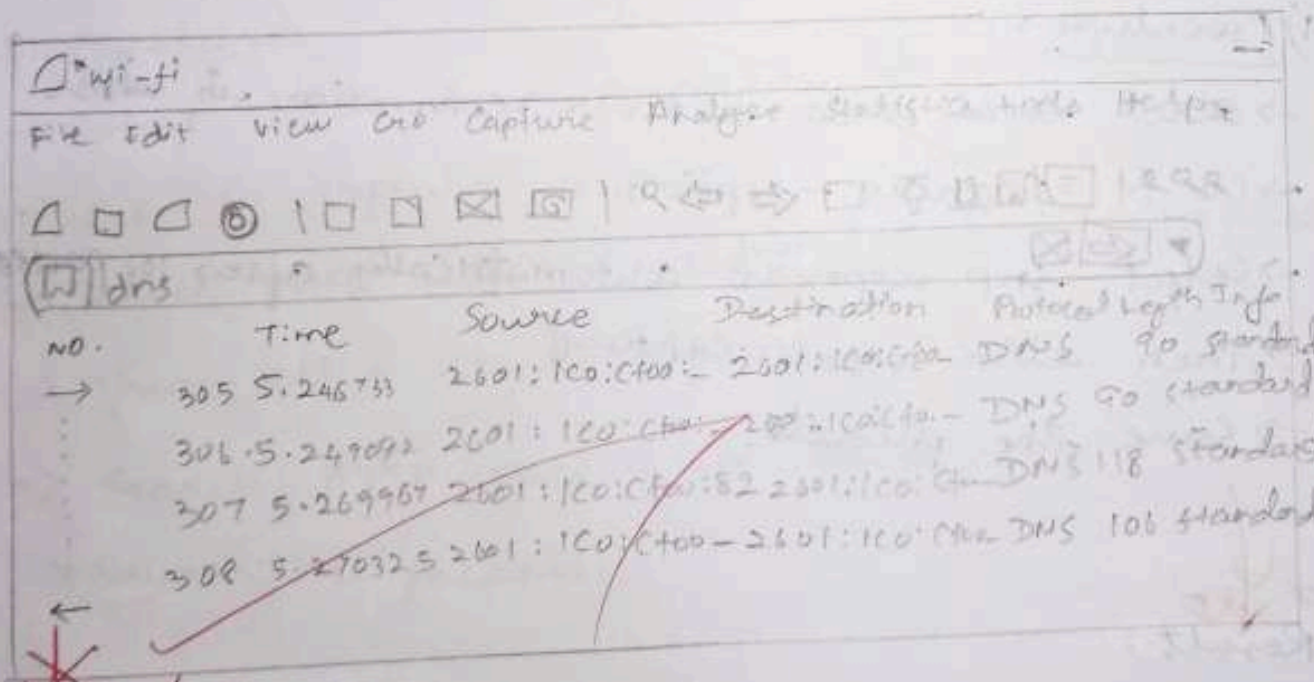
5) What is a broadcast IP address?

A broadcast IP address is a special address used to send data to all hosts on a specific network segment. In IPV4, the broadcast address for a subnet is the last address in the subnet's IP range.

2024/11/20 14:25



Filtering Packets:



Result:

The experiments on packet capture tool : wireshark is successfully completed.

2024/11/20 14:25

Filtering Packets:

You can click Analyze > Display filters to choose a filter from among the default filters included in wireshark. Now you can add your custom filters and save them for future use you can also right click a packet and select follows > TCP stream the full TCP conversation between the client and the server can be seen. The other protocols in the follow menu can be used to see the full conversations. for other protocols if applicable.

1) Procedure:

-> select local area network connection in wires
-> Go to capture -> option
-> Select stop capture automatically after 100 packets
-> Then click start capture
-> save the packets.

2024/11/20 14:26

2) Procedure:

-> Go to capture -> option
-> select top capture automatically after 100 packets
-> Then click start capture
-> search ARP packets in search bar
-> save the packets.

3) Procedure
-> Go to capture -> option
-> select stop capture automatically after 100 packets
-> Then click start capture.
->Search DNS packets in search bar.
->To see follow graph click start -> Flow graph
->save the packets.

4) Procedure
->select local Area Connection in wireshark
-> Go to capture -> option
-> select stop capture automatically after 100 packets
-> Then click start capture
-> Search HTTP packets in search bar.
->Save the packets.

2024/11/20 14:26

5) Procedure

-> Select local area connection in wireshark.
-> Go to capture -> option
-> Select stop capture automatically after 100 packets
-> Then click start capture
-> Search ICMP / IP packets in search bar
-> Save the packets.

6) Procedure:

-> Select local area connections in wireshark.
-> Go to capture -> option
-> Select stop capture automatically after 100 packets
-> Then click start capture
-> Search DNS packets in search bar.
-> Save the packets.

Result:

The experiments on packet capture tool: wireshark is successfully completed.

2024/11/20 14:27

19/8/24 Practical-6

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

Error Correction at Data Link Layer:

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

Create sender program with below features.

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

Create receiver program with below features.

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

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)
```

2024/11/20 14:28

```
    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 = 0
    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)
        if res <= n:
            data[res-1] = '0' if data[res-1] == '1'
            else '1'
            print(f"Error corrected at position: {res}")
```

2024/11/20 14:28

```
    else:
        print("Error position out of range. No
              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)
```

2024/11/20 14:28

Sender program:

```
def sender(text):
    binary_data = text_to_binary(text)
    m = len(binary_data)
    r = calc_redundant_bits(m)
    arr = pos_redundant_bits(binary_data, r)
    arr = calc_parity_bits(arr, r)
    print(f"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(f"Decoded text: {ascii_output}")
```

Main program

```
if __name__ == "__main__":
    input_text = input("Enter the text to be encoded:")
    channel_data = sender(input_text)
    corrupted_data = introduce_error(channel_data, 2)
    receiver(corrupted_data)
```

2024/11/20 14:29

Output:

Chithra

Enter the text to be encoded: Hello

position of redundant bits:: 1 2 4 8 16 32

parity bit in position 1:0 parity bit in position 2:0

parity bit in position 4:1 parity bit in position 8:0

parity bit in position 16:1 parity bit in position 32:1

Sender output (binary with redundant bits): 000

00011100001101110100001101001011110100011010000111001100001

Enter the bit position to introduce error: 6

Introduced error at position: 6

Binary data after introducing error: 00011000001101110100001

01101001011110100011010000111001100001

Error detected at position: 6

Error corrected at position: 6

Binary data after error correction:

000111000011011101000011010010111101000110100001110011000 01.

3/9/24

Result:

The program for hamming code is executed succesfully.

2024/11/20 14:29

26/8/24 Practical - 7

Aim:

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

create a sender program with following features:

1. Input windows size from the user.
2. Input a text message from user.
3. consider 1 character per frame.
4. create a frame with following fields [frame no, data]
5. Send the frames [print output on screen & save it in a file called send-buffer]
6. wait for the acknowledgement from receiver.
7. Reader a file called Receiver-Buffer.
8. Check ACK field from acknowledge number.

create a receiver file called with the following features.

1. Reader a file called sender-Buffer
2. check frame-no
3. If the frame-no are as expected, write the following appropriate ACK no in receiver-Buffer file. Else write NACK no in the receiver-Buffer file.

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" Send Frame {frame[i].frame_no}.
                {frames[i].data}")
    print("Frames sent, waiting for acknowledgment")
def receive_frames(frames, window_size):
    print("\n Receiving Frames")
    for i is range(window_size):
        if i < len(frames) and not frames[i].
                                acknowleged:
            If random.random() < 0.2:
            print("Received frame{frame[i].frame_no}.
                {frames[i].data}{2024-09-20 14:30
            frames[i].acknowledged = False
```

```
else:
    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)
        receiver_frame(frames[base:], window_size)
        while base < len(frames) and frames[base].
                                        acknowledged:

            base += 1
        if base < len(frames):
            print("\n Resending Acknowledged
                                        frames")

            time.sleep(2)
    print("\n All the frames sent & Acknowledged")

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

2024/11/20 14:31

Output:

Enter window size: 3
Enter message to send: Chithra
- Sending frames
Sent frame 0: C
Sent frame 1: h
Sent frame 2: i
frames sent waiting for acknowledgements

Receiving frames
Received frame 0: C [Received]
Received frame 1: h [Received]
Received frame 2: i [Received]

Resending unacknowledged frames.
Sending frames
Sent Frame 3: t Sent frame 4: h Sent frame 5: r
Frames sent, waiting for acknowledgements.

Receiving Frames.
Received frame 3: t [Received] Received frame 5: r
Received frame 4: h [Received] [Received.

Resending unacknowledged frames ...
Sent frame 6: a
frames sent, waiting for ack.

Received Frame 6: a [Received]
All frames sent and acknowledged!

Result:

Thus the program for sliding window protocol is executed successfully.

2024/11/20 14:31

Practical-8

a)

Aim: To simulate a virtual LAN configuration using cisco Packet tracer, follow these step-by step instruction.

Step 1: setup the topology

1. open cisco packet tracer
2. Drag and Drop switches (CISCO 2960) and 2 Pcs into the workspace
3. User copper Straight-through cables to connect the PCs to the switch using their Fastethernet ports.

Step 2: Assign IP address to PCs

1. Click on each PC, go to the Desktop tab and click IP Configuration.
2. Assign IP address to each PC.

- PC_1 : IP = 192.168.1.10, Subnet mask = 255.255.255.0
- PC_2 : IP = 192.168.2.10, Subnet mask = 255.255.255.0

Step 3: VLAN Configuration on the switch

1. Click on the switch and open the CLI.
2. enter the following commands to create VLAN's and assign the two ports to different VLAN's.

2024/11/20 14:31

Create VLAN's:

Switch> enable
Switch# configure terminal
Switch # vlan 10
Switch # (Config-vlan)# name
VLAN_10
Switch(Config-vlan)# exit
Switch(Config)# vlan 20
switch(Config-vlan)# name VLAN-20
Switch (Config-vlan)# exit

Assign switch ports to VLAN's:

- For PC, connected to Fast ethernet 0/1, assign it to VLAN 10:

switch(config)# interface
fast ethernet 0/1
Switch(config-if)# switch port
mode access
switch (config-if)# switch port
access vlan 10
Switch (config-if)# exit

- For PC2 connected to Fast ethernet 0/2, assign it to VLAN's 20

Switch(config)# interface
fast ethernet 0/2
Switch(config-if)# switch port
mode access
Switch (config-if)# switch port
access vlan 20
Switch (Config-if)# exit.

2024/11/20 14:32

Step 4: Verify VLAN Configuration.

1. Run the following command to confirm VLAN's are properly configured.

Switch # show vlan brief

Step 5: Test connectivity

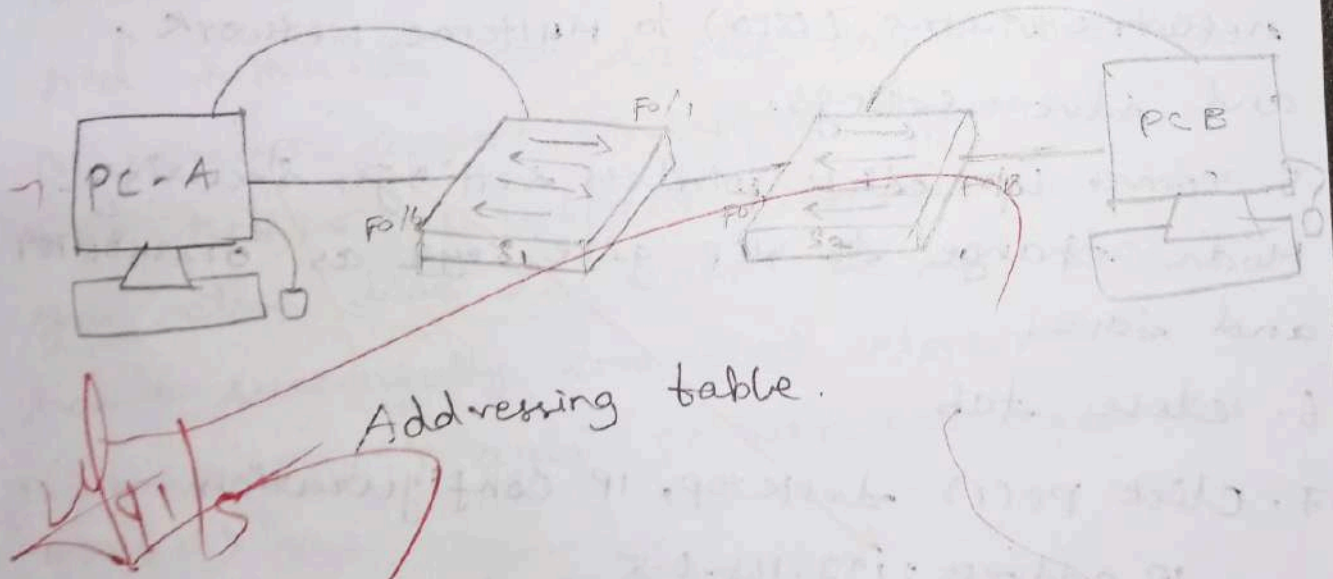
1. Go to PC_1 open the command prompt and ping PC_2's IP address (192.168.2.10)

2. Since PC_1 and PC_2 are in different VLANs, they should not be able to communicate with each other.

Step 6: Save the configuration.

Finally, save the configuration using

Switch # Copy running-config startup-config



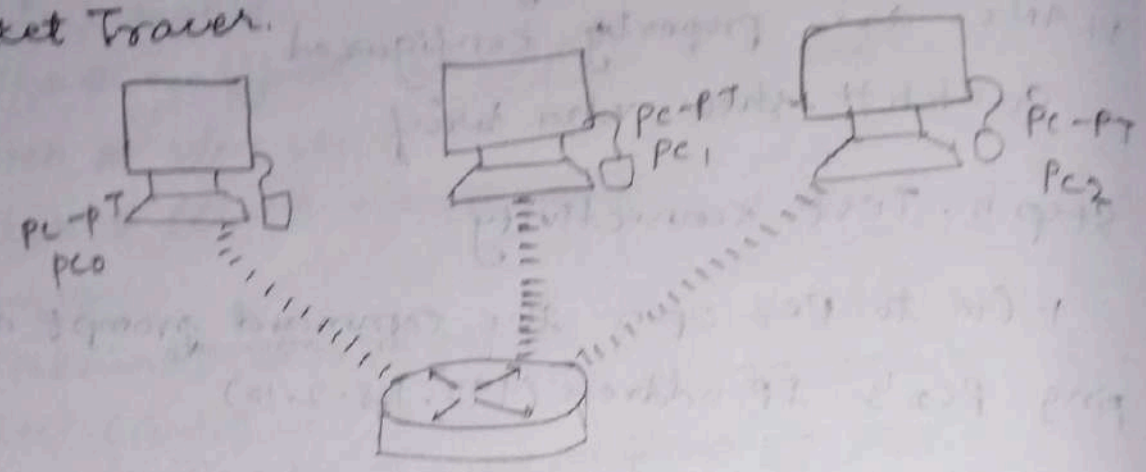
Addressing table.

Result:

Thus to simulate a virtual LAN Configuration is executed successfully.

2024/11/20 14:32

b) Configuration of wireless LAN using Cisco Packet Tracer.



1. Click network devices down click wireless devices select wireless Router in center.
2. Add 3 pc's, click End devices to add.
3. click router GUI tab check given IP address is 192.168.0.1, click disabled and save settings.
4. come up again click wireless setting, change network name (SSID) to MyHome Network, and save settings.
5. comes up, click wireless settings, Security Mode change to WEP give key1 as 0123456789 and save.
6. close tab.
7. Click PC(1) desktop, IP Configuration.

   IP address: 192.168.0.5

   Subnet mask: 255.255.255.0

   Default gateway: 192.168.0.1 close tab.

8. Go to next PC(2) desktop, IP Configuration

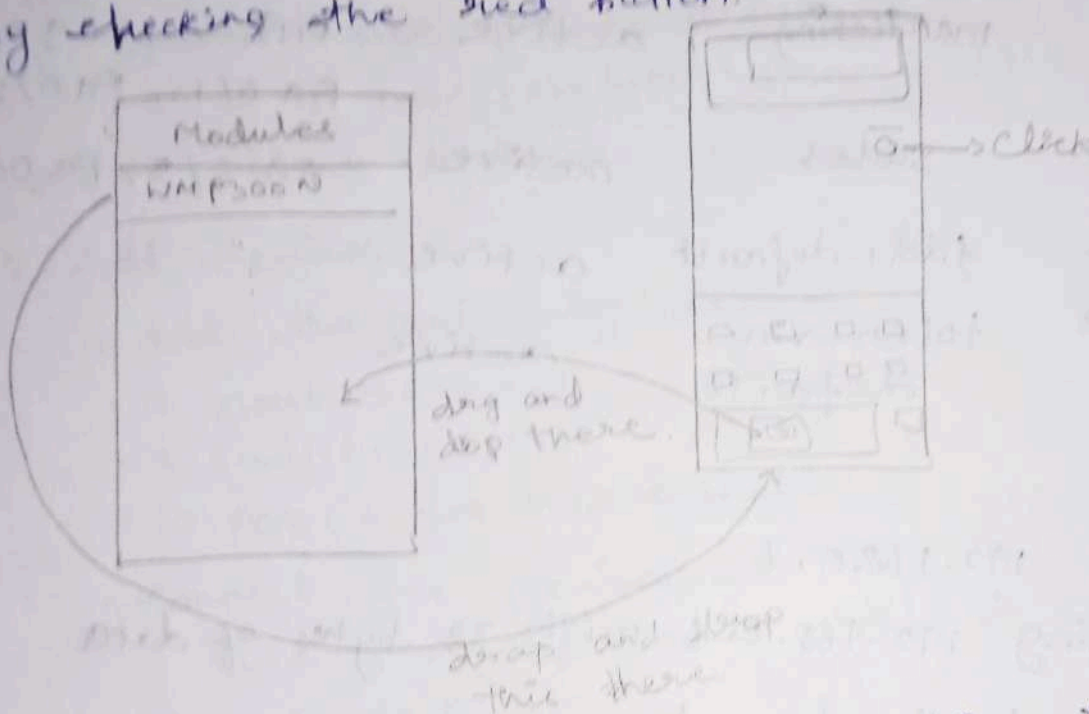
   IP address: 192.168.0.6

2024/11/20 14:33

subnet mask : 255.255.255.0
Default gateway : 192.168.0.1 close tab.

10. Again click pc(1), PC wireless -> Error will come, close it.

11. On top, go to physical tab, turn off the PC by checking the red button.



12. Now turn on the PC, by clicking the red button. Go to desktop tab on the top, PC wireless, connect, wireless router, connect, WEP key -> 0123456789, connect now when you click the information it will show you have successfully connected, close tab.

13. Now PC(2) again PC wireless (Repeat step 10, 11, 12)

14. PC(3) -> (Repeat step 10, 11, 12)

15. To check the connectivity whether all is connected click any PC, command prompt any IP address given during this process is

ping 192.168.0.1 enter.

2024/11/20 14:33

Output:

a)

VLAN	NAME	Status	Ports
1	default	active	Fa 0/5, Fa 0/6, Fa 0/7, Fa 0/8, Fa 0/24
10	marketing	active	Gig 0/1, Gig 0/2 Fa 0/1, Fa 0/2
20	sales	active	Fa 0/3, Fa 0/4
1002	fddi-default	active	
1003	token-ring-default.	active	

b)

Ping 192.168.0.6

Pinging 192.168.0.6 with 32 bytes of data

Request timed out

ping statistics for 192.168.0.6

Packets: Sent = 4, Received = 0, Lost = 4 (100% loss)

Result:

Thus to simulate Virtual LAN configuration using cisco packet Tracer and configuration of wireless LAN using cisco packet Tracer is done and executed successfully.

2024/11/20 14:33

Practical - 9

Aim: Implementation of SUBNETTING in cisco packet tracer simulator.

Steps

1. create a Network topology
   - open a cisco packet tracer
   - click on new > network > Generic to create a blank topology.

2. Add devices
   - add the following devices
   - 2 routers (R_1, R_2)
   - 2 switches (S_1, S_2)
   - 10 PCs (5 for each subnet).
   - Connect the devices:
     - Use the appropriate cables to connect
       - R_1 to S_1
       - R_1 to R_2
       - S_2 to R_2

3. Subnetting configuration
   - Network address: 192.168.1.0/24
   - Subnet mask: 127 (provides 8 subnets host addresses each).

4. IP Addressing scheme:
   - Router R_1
     - Gigabit Ethernet 0/0: 192.168.1.1
     - Gigabit Ethernet 0/1: 192.168.2.1

2024/11/20 14:35

   - Switch S_1
     Fast Ethernet 0/1: 192.168.1.0/27

- PCs
  - 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

- switch S2:

  Fast Ethernet 0/1 : 192.168.1.0/24

  - PCs
    - 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

5. Configuring the devices :
   - Router configuration
     - open CLI on Router R1 and enter

enable
configure terminal
interface Gigabit Ethernet 0/0
ip address 192.168.2.1, 255.255.255.224
no shutdown
Exit

- Switch configuration
  - open CLI on switch S1 and enter

enable
configure terminal
interface fastethernet 0/1
interface mode access
switch port mode access
exit.

2024/11/20 14:35

interface fast ethernet 0/2
switch port mode access
exit

- Pc configuration
  - Right click on each port and select configure

Enter:
IP addresses, subnet mask (255.255.255.224), default gateway (router IP).

6. Testing the Network:
   - Open the command prompt on each PC
   - Use the ping command to check connectivity between PCs and the router.

Ping 192.168.1.X (for PCs in the first subnet)
Ping 192.168.3.X (for PCs in the second subnet)

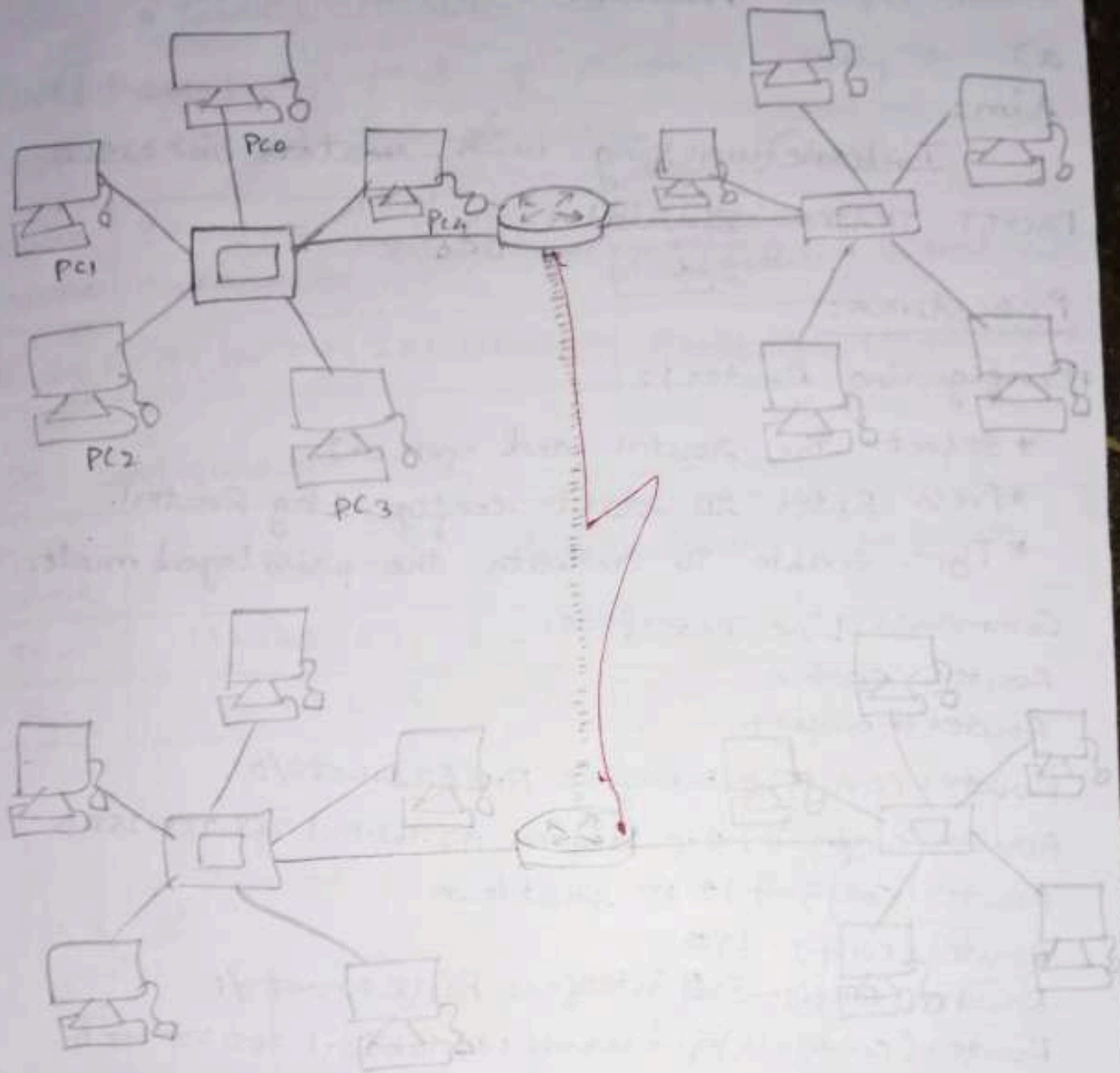
7. Conclusion

If all pings are successful your subnetting and network configuration in cisco packet tracer is functioning correctly.

Observation:

1) Write down your understanding of subnetting.

Subnetting is the practice of dividing a larger network into smaller, manageable sub-networks (subnets) to improve performance and security. It uses a subnet mask to define the network and host portions of an IP address.



8/11

Result:

Thus the implementation of subnetting in CISCO packet tracer is executed successfully and configured.

2024/11/20 14:36

Practical-10

a)

Aim:

Internetworking with routers in CISCO PACKET TRACER Simulator.

Procedure:

1) Configuring Router1:

* Select the router and open CLI.
* Press Enter to start configuring Router1.
* Type enable to activate the privileged mode.

Commands Line Interface:

```
Router> enable
Router# config t
Router(config)# interface FastEthernet0/0
Router(config-if)#ip address 192.168.10.1 255.255.255.0
Router(config-if)# no shutdown
Router(config-if)#
Router(config-if)# interface FastEthernet0/1
Router(config-if)#ip address 192.168.20.1 255.255.255.0
Router(config-if)# no shutdown
```

2) Configuring PCs:

* Assign IP Address to every PC in the network.
* Select the PC, Clo to the desktop and select IP Configuration and assign an ip address Default gateway
* Assign the default gateway of PC0 as 192.168.10.1
* Assign the default gateway of PC1 as 192.168.20.1

3) Connecting PCs with Router:

* Connect FastEthernet() port of PC0 with FastEthernet 0/0 port of Router1 using a copper straight through cable.

2024/11/20 14:36

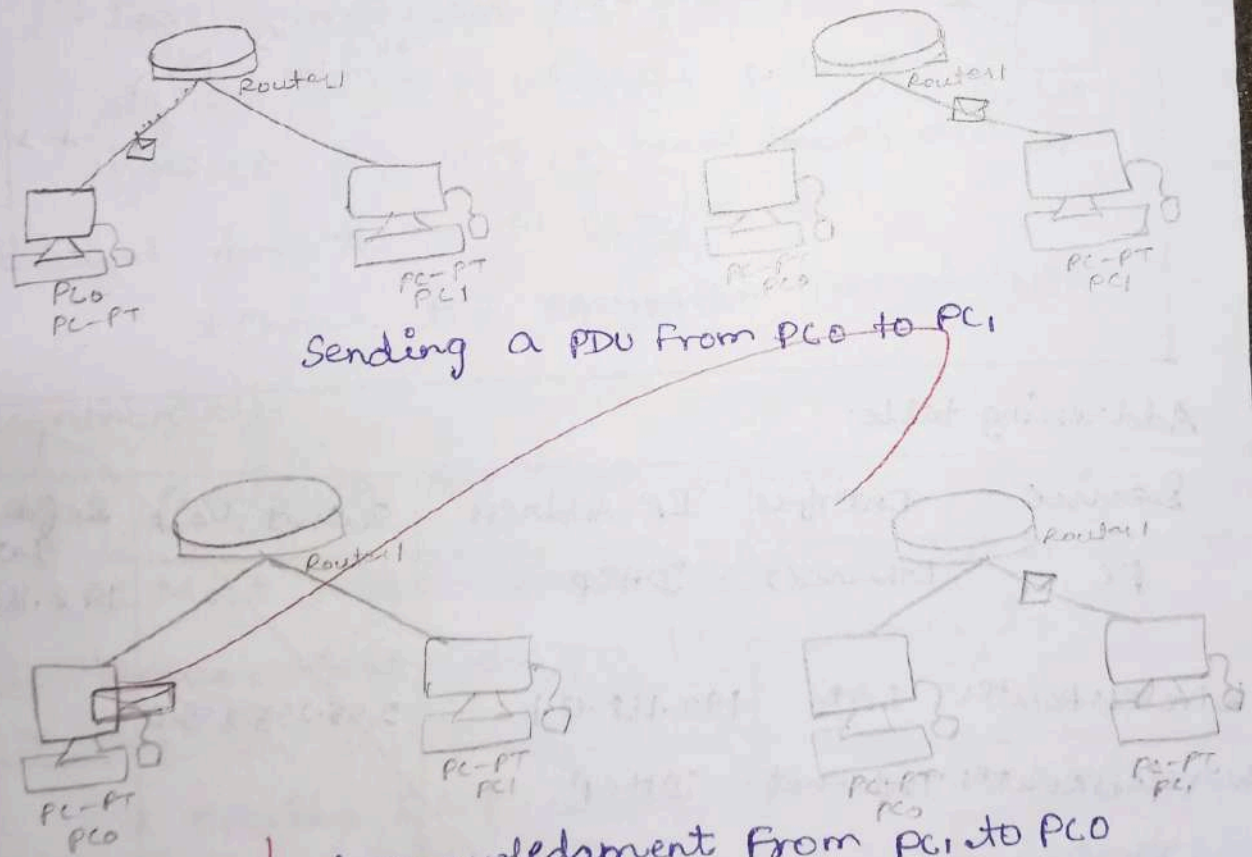
* Connect FastEthernet0 port of PC1 with FastEthernet 0/1 port of Router) using a copper straight through cable.

Router table

Device name	IP address FastEthernet 0/0	Subnet Mask	IP address FastEthernet0/1	Subnet Mask
Router1	192.168.10.1	255.255.255.0	192.168.20.1	255.255.255.0

PC configuration table

Device name	IP Address	Subnet Mask	Gateway
PC0	192.168.10.2	255.255.255.0	192.168.10.1
PC1	192.168.20.2	255.255.255.0	192.168.20.1



Result:

Thus the internetworking with routers is executed successfully.

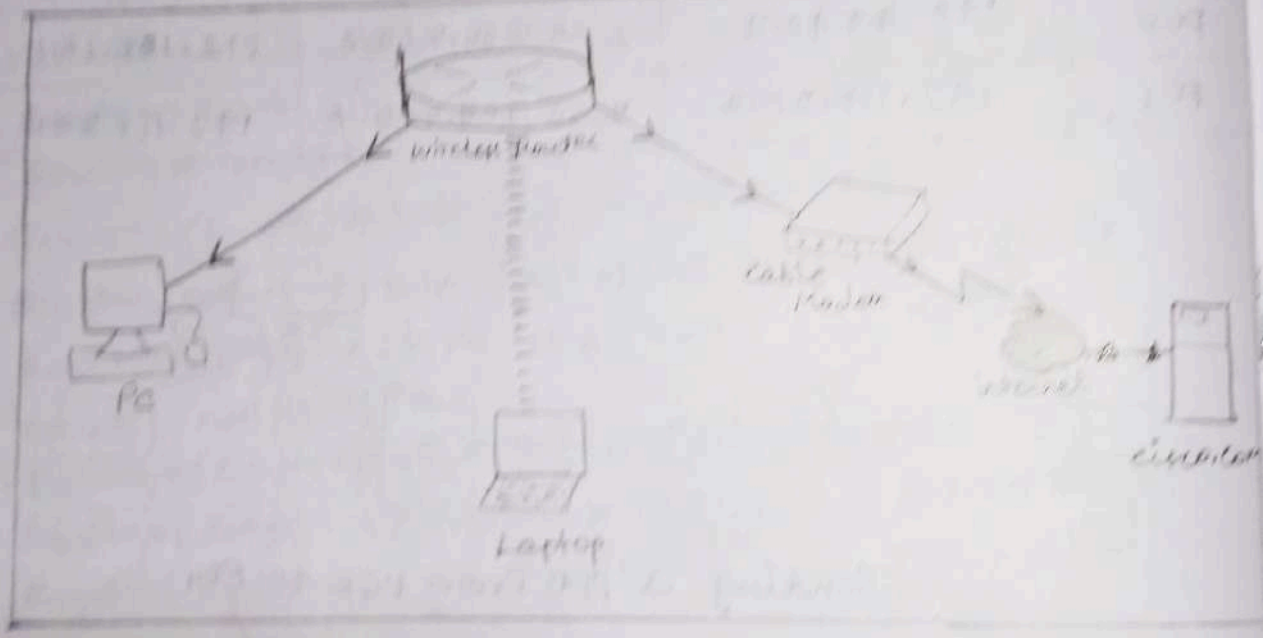
2024/11/20 14:36

Practical-10

b) Design and configure an Internetwork using wireless router, DHCP Server and Internet cloud.

Step 1: Components Required

- Wireless Router
- DHCP Server
- Internet Cloud
- End devices
- Ethernet cables



Addressing table:

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

2024/11/20 14:37

Step by step configuration:

step 1: setup the wireless Router

• Physical Setup:

* Connect the internet source (ISP) to the router's WAN port.

* Connect any wired devices via LAN ports on the Router.

• Access Router Admin Panel:

* Connect a device to the router.

* Open a web browser and type the default IP address of the router (commonly 192.168.1.1 or 192.168.0.1)

* Login using the default credentials.

• Enable wireless Networking:

* Go to the wireless settings

* set the SSID (network name) and password for the Wi-Fi Network.

* Choose the encryption method (WPA2 is recommended).

Step 2: Configure the DHCP server

* Most modern routers have a built-in DHCP server that assigns IP addresses to devices.

* Enable DHCP under the LAN settings.

* set the IP address range (for example, from 192.168.1.2 to 192.168.1.100)

2024/11/20 14:37

* Specify the lease time for the IP addresses.

* optionally, configure DHCP reservations for specific devices if needed.

Step 3: Internet cloud setup

* The internet cloud represents your connection to the wider internet. If you are simulating the network, you can use cloud devices in simulation software like Cisco Packet Tracer or GNS3.

* For a real world setup, make sure your router is connected to the ISP modem or directly to the Internet provider.

* Verify the WAN connection type in the router settings (it may be PPPoE, DHCP, or static IP depending on your ISP).

Step 4: Configure Devices

* For wireless devices (e.g., laptops, phones), connect them to the wireless network using the SSID and password you set earlier.

* The devices should receive an IP address automatically from the DHCP server.

Step 5: Connectivity Test

* Ping test: From a device, open the command prompt (Windows) or terminal (Linux/Mac) and ping an external IP to verify internet connectivity.

* Check IP Address: Run ipconfig or ifconfig to check if the device has received a correct IP address from the DHCP server

Step 6: Advanced options.

* Port Forwarding: If certain services (a web server) need to be accessed from the internet, you may need to configure port forwarding in the router settings.

* Firewall Rules: Most routers have a built in firewall. Configure it to allow or block specific traffic as needed.

Considerations for the network Design

* Security: Ensure the WiFi network is secured with WPA2 encryption and use a strong password.

* Backup DHCP Server: In larger networks, a dedicated DHCP server might be needed for reliability.

* Scalability: Depending on the number of devices, switches or additional access points might be required to extend the network.

Result:

8/11

Thus the interworking with Cisco and internetwork using wireless router, DHCP server and internet cloud. is configured and executed successfully.

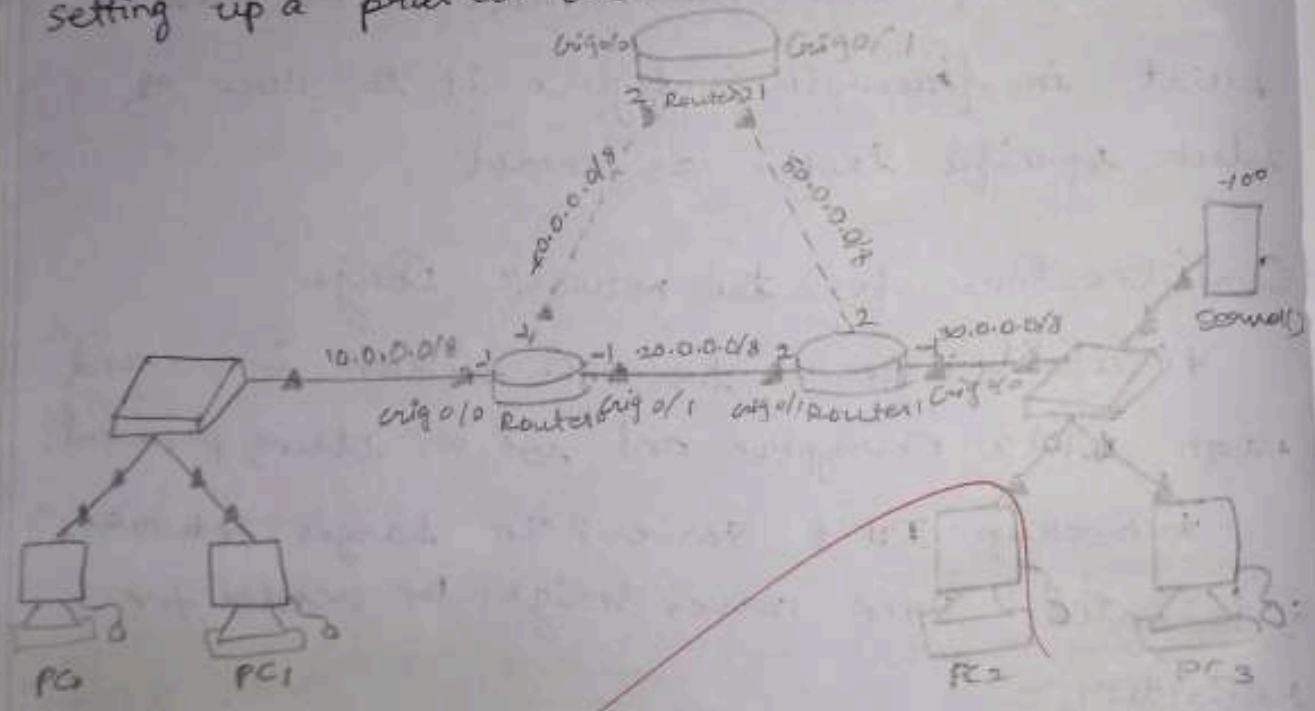
Practical-11

a)

Aim: Simulate static Routing configuration using Cisco Packet Tracer.

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

setting up a pratical lab



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

2024/11/20 14:38

If we specify two routes for the same destination, the router automatically selects the best route for the destination and adds the route to the routing table.

If you manually want to select a route that router should add to the routing table, you have to set the AD value of the route lower than other routes.

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

creating, adding, verifying static routes

Routers automatically learn their connected networks. We only need to add routes for the networks that are not available on the router's interfaces. We have to create and add routes only for these networks.

Router0 requirements

* Create two routes for network 30.0.0.0/8 and configure the first route as the main route and the second route as a backup route.
* Create two routes for the host 30.0.0.100/8 and configure the first route (via-router 2) as the main route and the second route as backup route.
* Create two routes for network 50.0.0.0/8 network and configure the first route as the main route and second route as a backup route.
* verify the router adds only main routes to the routing table.

2024/11/20 14:39

Router 1 Requirements

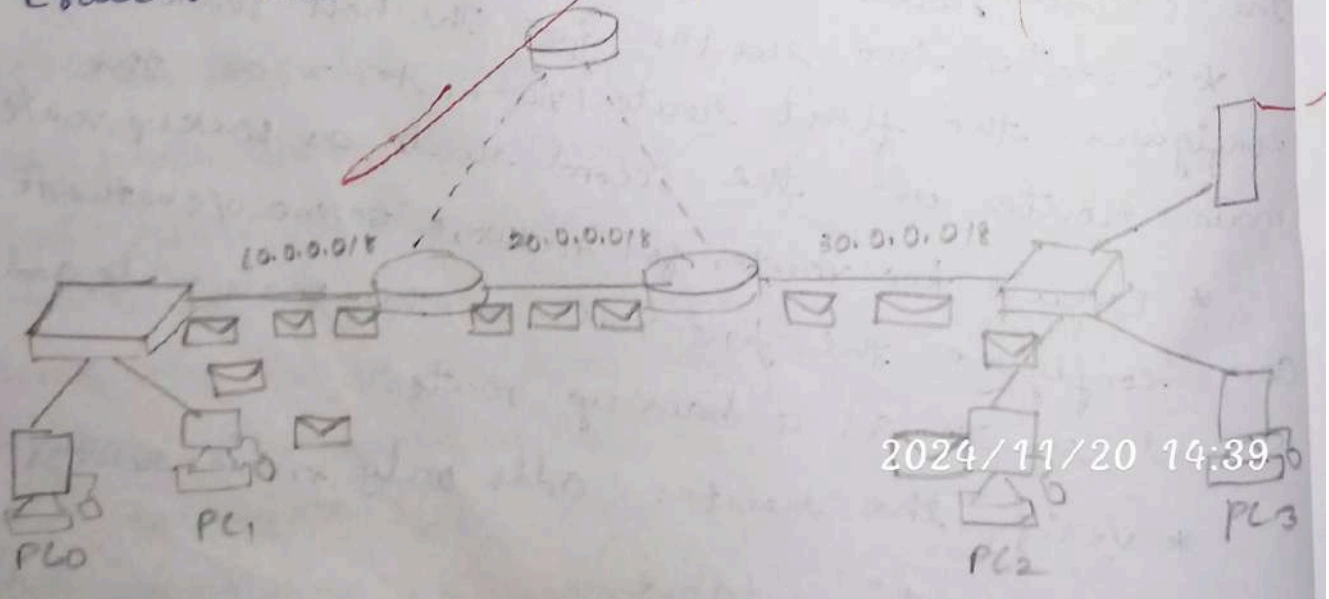
* create two routes for network 10.0.0.0/8 and configure the first route (via Router 1) as the main route and second route as a backup route.
* create two routes for network 40.0.0.0/8 and configure the first route as the main route and second routes as backup route.
* verify the router adds only main routes to the routing table.

Router 2 Requirements

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

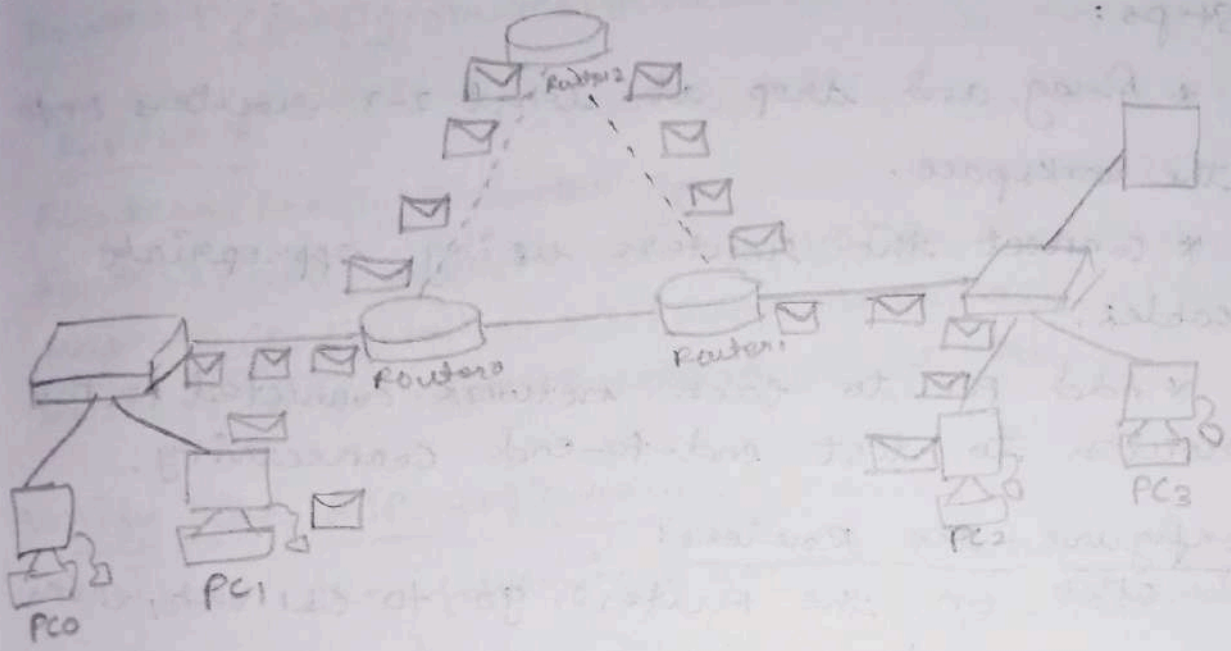
verifying static routing:

On router0, we configured two routes for network 30.0.0.0/8. These routes are via Router 1 and Router2. By sending ping requests to a PC of network 30.0.0.0/8 and tracing the path they take to reach the network 30.0.0.0/8. For this use 'tracert' command on PC of network.



2024/11/20 14:39

We also configured a separate static host route for the host 30.0.0.100/8. The router must use this route to forward data packets to the host 30.0.0.100/8.



Router0

```
Router>enable
Router# configure terminal
Router(config)# ip route 30.0.0.0 255.0.0.0 20.0.0.2 10
Router(config)# ip route 30.0.0.0 255.0.0.0 40.0.0.2 20
Router(config)# ip route 30.0.0.100 255.255.255.255 40.0.0.2 10
Router(config)# exit
```

Router1

```
Router>enable
Router# configure terminal
Router(config)# ip route 10.0.0.0 255.0.0.0 20.0.0.1 10
Router(config)# ip route 10.0.0.0 255.0.0.0 50.0.0.1 20
Router(config)# exit
```

Router2

2024/11/20 14:39

```
Router>enable
Router#configure terminal
Router(config)# ip route 10.0.0.0 255.0.0.0 via 40.0.0.1
Router(config)# exit
```

b) Simulate RIP using CISCO Packet Tracer.

Aim:
To simulate RIP using CISCO Packet Tracer

Steps:
* Drag and drop at least 2-3 routers onto the workspace.
* Connect the routers using appropriate cables.
* Add PCs to each network connected to the routers to test end-to-end connectivity.

Configure the Routers:
Click on the Router0, go to CLI tab, enter the commands.

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

Repeat the above configuration for all router interfaces connecting to other routers or networks.

Enable RIP on the Routers:
enter the following commands,

```
Router0(config)# router rip
Router0(config-router)# network 10.0.0.0
Router0(config-router)# network 192.168.1.252
Router0(config-router)# network 192.168.1.2[redacted]
```

2024/11/20 14:40

Router 1

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

Router 2

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

Verify the RIP configuration:

Use the following command to check if RIP is configured and routes are being advertised:

```
show ip route
show ip protocols
```

Test connectivity:

Ping from one pc to another across the network to ensure data can travel through the configured routes.

Result:

Thus the simulation of static routing configuration using cisco Packet Tracer is configured and executed successfully.

2024/11/20 14:40

Practical-12

a)

Aim:

Implement echo client server using TCP/UDP sockets

1. TCP echo client-server algorithm.

Server:

1. Create a TCP socket
2. Connect the socket to a local address & port.
3. Listen for incoming client connections
4. Accept a client connect
5. Loop
   - Receive data from the client
   - If data is received, sent it back to client
   - else, break the loop
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 the server
5. Display the received message
6. close Socket.

TCPServer.py

```
import Socket
def tcp-seaver():
    Server_Socket = Socket.Socket(
                    Socket.SOCK-STREAM)
    Server-Socket.bind(("localhost", 12345))
    Server-Socket.listen(1)
```

2024/11/20 14:41

```
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, socket.
                                  SOCK_STREAM)
    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_socket.close()
if __name__ == "__main__":
    tcp_client()
```

2024/11/20 14:41

output:

>python tcp-client.py

Enter a message to send : Hi, I am chithra

Received from server : Hi, I am chithra

>python tcp-server.py

TCP server is waiting for a connection

Connected to ('127.0.0.1', 56893)

Received : Hi, I am chithra

2. UDP echo client-server Algorithm.

Server:

1. create a UDP socket
2. Bind the socket to a local address & port.
3. Loop
   - Receive the data from client
   - Echo the received data back to the client
4. The server keeps running aand does not need to close the socket explicitly.

client:

1. create a UDP socket.
2. send a message to the server using the specified address & port.
3. Receive the echoed message from server.
4. Display the received message.
5. close socket.

2024/11/20 14:41

Output:

> python udp-client.py

Enter message to send: Hi, I am chithra

Received from server: Hi, I am Chithra

> python udp-server.py

UDP Server is listening

Received from ('127.0.0.1', 56621)

Received : Hi, I am chithra

Result: 28/11

The program for using echo client Server using TCP/UDP has 2024/11/20 14:42 executed successfully.

Practical-12

b)

Aim: To implement the chat client server using TCP / UDP server.

Algorithm:

chat server:

1. Start the server by creating a socket, bind to a specific address and port, listen for incoming connections.
2. When a new client connects add client to a list of a connected clients start a new process to talk to the clients.
3. For each connected clients start a new keep checking for new messages.
4. If a client disconnects remove that client from the list & stop talking to that client.
5. Keep running the process till the server stops.

Chat-client:

1. Connect to the server by creating a socket and connect it to server address & port.
2. Start a process by creating a ~~sock~~ to listen to messages from the server.
3. Keep asking for the new messages.
4. Keep running till the user decides to quit.

2024/11/20 14:42

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(f"server : {message}');
        except Exception as e:
                print(f"an error occured: {e}")
                break
def start_client():
    client_socket = socket.socket(socket.AEINET, Socket
                                  Sock_STREAM)
    host = '127.0.0.1'
    port = 12345
    client_socket.connect(host, port))
    print("Connected to chat server")
threading.Thread(target = receive:message, args =
                 (client_socket,), daemon=True).start()
while True:
    message = input("Yes:")
    client_socket.send(message.encode('utf-8')
if_name__=='__main__':
        Start_client()
```

2024/11/20 14:43

chat-server.py:

```
import socket
import threading
def handle_client(client_socket):
    while True:
        try:
            message = client_socket.recv(1024).decode('utf-8')
            if not message:
                break
            print(f"Received message from client {message}")
            client_socket.send(response.encode('utf-8')
        except Exception as e:
            print(f"An error has occured")
            break
    client_socket.close()
def start_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_handler.start()    args=(client_socket)
if __name__=="main__"
    start_server()
```

2024/11/20 14:43

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 : chithra

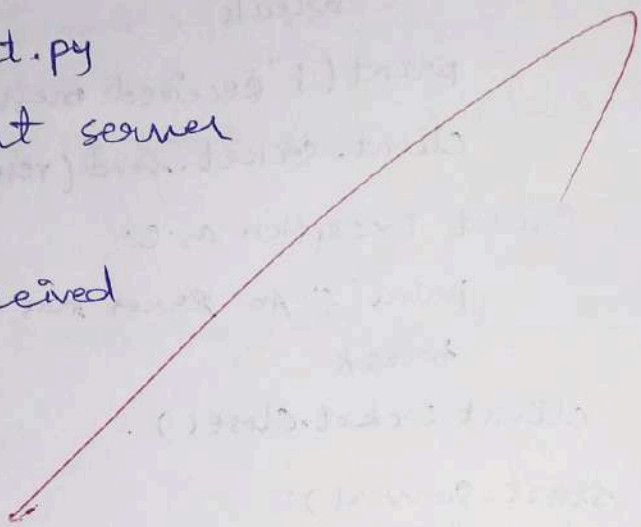
Type from message to client : ~~Hello~~ Received

>python chat-client.py

Connected to chat server

You: chithra

You : Server : Received



Result:

Thus, the implementation of chat client server using TCP/UDP socket has been successfully executed & verified.

2024/11/20 14:43

Practical-13.

Aim:

Implement your own ping program.

Algorithm:

ping-client.py

1. socket creation
2. Then set a timeout of 2 second to ensure that if no response is received it will stop waiting and print "request time out".
3. Sends a 'ping' msg to specified host & port
4. It listens for a response & calculator the time difference b/w sending & receiving the packet.

ping-server.py

1. Initialise UDP socket
2. Bind to IP address & port.
3. Listen for incoming messages.
4. Receive data
5. send Response

CODE:

ping-server.py:

```
import socket
def start-server(hos='127.0.0.1', port=12345)
    with socket.socket(AF-INET, socket.sock-STREAM)
                S.bind(host, port))
        print(f"UDP server running on{host}'{port}")
while True
    data, addr = S.recvfrom(1024)
    print(f"Received message from{addr}:
        {data.decode()}")
```

2024/11/20 14:44

```
        s.sendto(b'Pong', addr)

if __name__ == '__main__':
    start_server()
```

ping_client.py:

```
import socket
import time
def ping_server(host='127.0.0.1', port=12345):
    with socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
as s:
    try:
        s.settimeout(2)
        start = time.time()
        s.sendto(b'Ping', (host, port))
        data, addr = s.recvfrom(1024)
        end = time.time()
        print(f"Received {data.decode()} from {addr}
            in {end - start:.2f} seconds")
    except socket.timeout:
        print("Request timed out")

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

2024/11/20 14:44

Output:

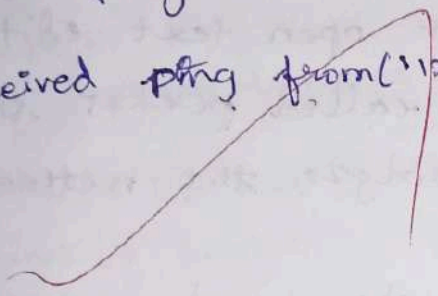
>python ping-server.py

UDP server running on 127.0.0.1:12345

Received message from ('127.0.0.1', 5734):

>python ping-client.py

Received ping from ('127.0.0.1', 12345) in 0.00 seconds



Result: 28/11

The program to create and implement ping message has been successfully executed.

2024/11/20 14:44

Practical-14

Aim:

Write a code using RAW sockets to implement packet sniffing.

Algorithm:

1. Install python and scapy
2. create a program open text editor and create a file in notepad called packet sniffer.py & code to capture & analyze the network packets.
3. set up packet tracer by check of the packet has IP layer, identify the packets overload protocol such as TCP, UDP, ICMP
4. Capture Network Packets
5. Run the packet sniffer by using command.
6. Generate Network traffic by running the program.

Program:

```
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
        dst_ip = ip_layer.dst
```

2024/11/20 14:45

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

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:

```
>python packet_sniffer.py
Protocol : TCP
Source IP : 232.15.215.227
Destination IP : 192.168.109.73
```

Result:

The program to implement packet sniffing is done successfully

2024/11/20 14:45

Practical-15

Aim:

To analyse the different types of weblogs using webalizer tool.

Algorithm:

Step 1: Run webalizer windows version

Step 2: Input web log file / download from web.

Step 3: Press run webalizer.

Result:

Thus the experiment for using webalizer for web log analysis is executed and verified.

2024/11/20 14:46

completed

2024/11/20 14:46