SARASWATI Education Society's

SARASWATI College of Engineering

Learn Live Achieve and Contribute

Kharghar, Navi Mumbai - 410 210.

NAAC Accredited

Department of Computer Engineering

Name: Shaikh Adnan Shaukat Ali

Roll No: 68

Subject: Computer Network Lab(CNL)

Class/Sem: T.E. / SEM-V

SARASWATI Education Society's

SARASWATI College of Engineering

Learn Live Achieve and Contribute

Kharghar, Navi Mumbai - 410 210.

NAAC Accredited

COMPUTER ENGINEERING DEPARTMENT

Subject: Computer Network Lab (CNL) **Class/Sem: TE/V**

Name of the Laboratory: System Lab **Year: 2021-2022**

LIST OF EXPERIMENTS

H/W Requirement	P I and above, RAM 128MB, Printer, Cartridges
S/W Requirement	C, C++, Java, Cisco Packet Tracer, Wireshark.

Prof. Arti K. Gore
Subject In-charge

Prof. Sujata Bhairnallykar
HOD

Experiment No. 1

Aim: To study RJ45 and CAT6 Cabling and connection using crimping tool.

Requirements: RJ45 connector, CAT6 Cable and Crimping tool.

Theory:
RJ45 Interface/Connector: RJ45 interface is considered the most common twisted-pair connector for Ethernet cables and networks.

- "RJ" means "registered jack" — a standardized telecommunication network interface for connecting voice and data equipment to a service provided by a local exchange carrier or long-distance carrier.
- "45" is the number of the interface standard. Physically speaking, the connectors that registered jacks use are mainly the modular connector and 50-pin miniature ribbon connector types. RJ45 connector is an **8-position, 8-contact (8P8C) modular plug, and jack**, applied for Ethernet-based local area networks (LAN). RJ45 cable plug is usually made of a plastic piece with eight pins on the port. Four of the pins are used for sending and receiving data, and the other four are used for other technologies or power networking devices.
- RJ45 connectors can support 10Gbps over Ethernet

How to identify RJ45 interface: RJ45 is identified using color code scheme. T568A vs T568B are the two common wiring schemes, which are used to terminate the twisted-pair cable onto the connector interface. The two standards define how the RJ45 pinouts arrange the individual eight wires when linking the RJ45 connector to a cable. These wiring layouts have their own color convention to follow for electrical compatibility. The T-568B wiring scheme is considered to be the more commonly used one.

The differences between T568A vs T568B in color conventions are shown in the figure below.

With regard to the two standards, there are two different connectivity forms. If both ends of the patch cords are wired on the basis of one standard, it is a straight-through connection. If not, it is a crossover connection. Some networking applications require a crossover Ethernet cable, which has a T-568A connector on one end and a T-568B connector on the other. This type of

cable is typically used for direct computer-to-computer connections when there is no router, hub, or switch available.

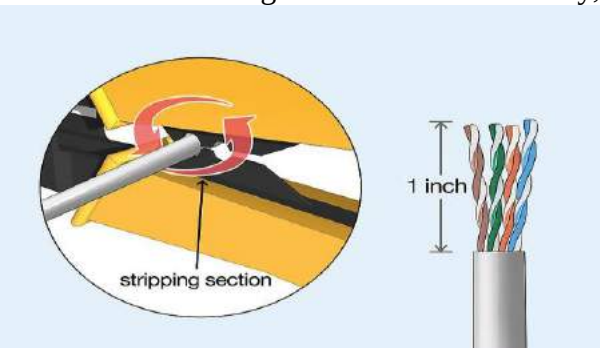
Cat6 Cable: Short for Category 6, Cat6 is an Ethernet cable standard defined by the Electronic Industries Association (EIA) and Telecommunications Industry Association (TIA). What is Cat6 cable used for? As the sixth generation of twisted pair Ethernet cabling, Cat6 cable consists of four twisted pairs and is either terminated by an RJ45 or terminated on a patch or a keystone jack. Theoretically, the maximum speed of the Cat6 network cable is 10Gbps.
A single run of Ethernet cable is designed to work at a maximum distance of 100 meters (328 ft). A length longer than that will result in issues such as dropped packets, reduced performance, and loss of signal when deploying Cat6 cable. The max length of a Cat6 cable usually consists of 90 meters (295 ft) of solid "horizontal" cabling between the patch panel and the wall jack, plus 5 meters (16 ft) of stranded patch cable between each jack and the attached device. For 10GBASE-T, an unshielded Cat6 cable should not exceed 55 meters.

Crimping tool: It is a tool that is used to connect RJ45 connector to Cat6 cable

Steps to crimp RJ45 with Cat6 using Crimping tool:

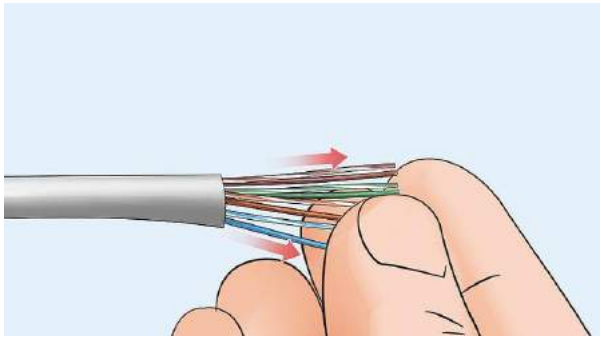
Step 1: **Strip the cable back 1 inch (25 mm) from the end**. Insert the cable into the stripper section of the tool and squeeze it tight. Then, rotate the crimping tool around the cable in a smooth and even motion to create a clean cut. Keep the tool clamped and pull away towards the end of the wire to remove the sheathing.

- The stripping section is a round hole near the handle of the tool.
- The sheathing should come off cleanly, leaving the wires exposed.



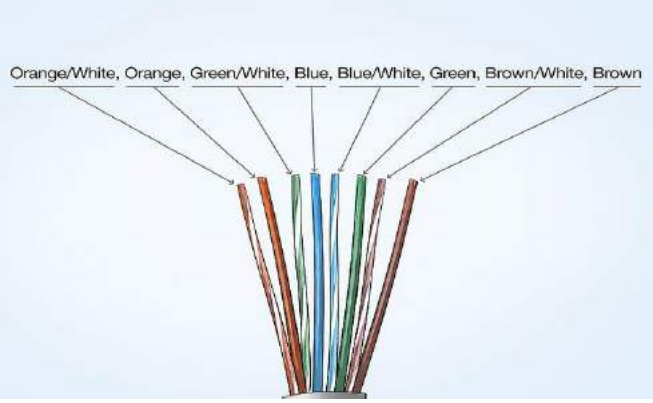
Step 2: **Untwist and straighten the wires inside of the cable.** Inside of the cable you'll see a bunch of smaller wires twisted together. Separate the twisted wires and straighten them out so they're easier to sort into the right order.

- Cut off the small plastic wire separator or core so it's out of the way.
- Don't cut off or remove any of the wires or you won't be able to crimp them into the connector.



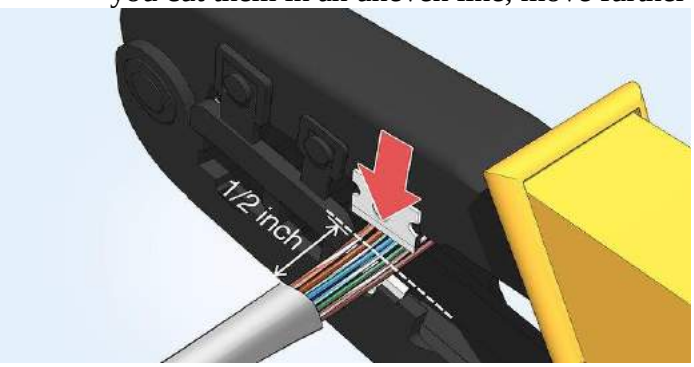
Step 3: **Arrange the wires into the right order.** Use your fingers to put the wires in the correct order so they can be properly crimped. The proper sequence is as follows from left to right: Orange/White, Orange, Green/White, Blue, Blue/White, Green, Brown/White, Brown.

- There are 8 wires in total that need to be arranged in the right sequence.
- Note that the wires labeled Orange/White or Brown/White indicate the small wires that have 2 colors.



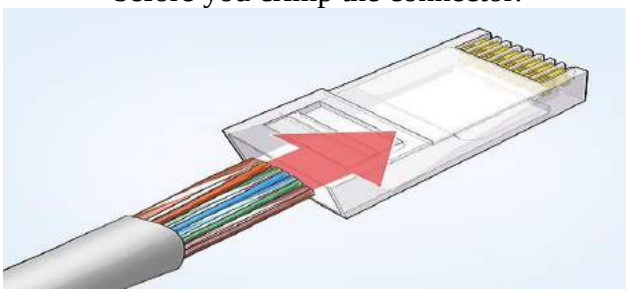
Step 4: **Cut the wires into an even line ½ inch (13 mm) from sheathing.** Hold the wires with your thumb and index finger to keep them in order. Then, use the cutting section of the crimping tool to cut them into an even line.

- The cutting section of the tool will resemble wire cutters.
- The wires must be in an even line to be crimped into the RJ-45 connector properly. If you cut them in an uneven line, move further down the wires and cut them again.



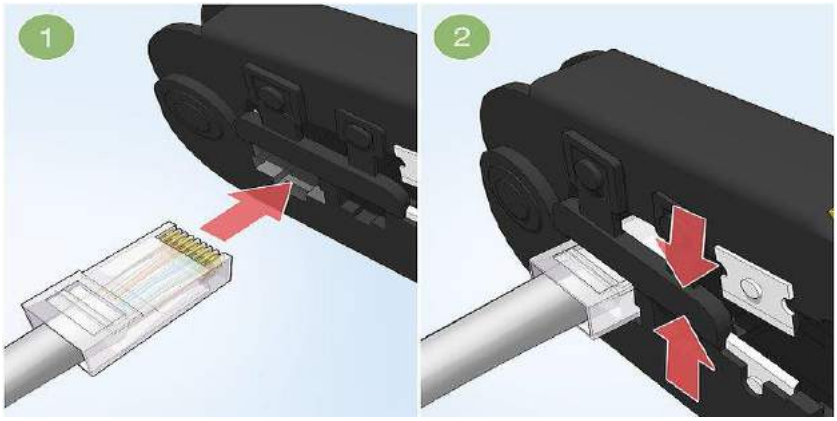
Step 5: **Insert the wires into the RJ-45 connector.** Hold the RJ-45 connector so the clip is on the underside and the small metal pins are facing up. Insert the cable into the connector so that each of the small wires fits into the small grooves in the connector.[5]

- The sheathing of the cable should fit just inside of the connector so it's past the base.
- If any of the small wires bend or don't fit into a groove correctly, take the cable out and straighten the wires with your fingers before trying again.
- The wires must be inserted in the correct order and each wire must fit into a groove before you crimp the connector.



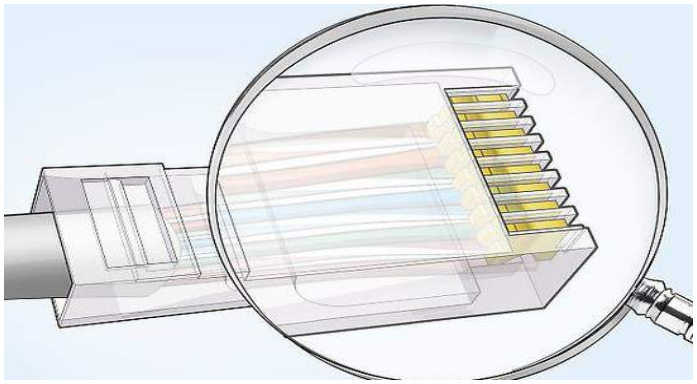
Step 6: **Stick the connector into the crimping part of the tool and squeeze twice.** Insert the connector in the crimping section of the tool until it can't fit any further. Squeeze the handles to crimp the connector and secure the wires. Release the handles, then squeeze the tool again to make sure all of the pins are pushed down.

- The crimping tool pushes small pins in the grooves down onto the wires to hold and connect them to the RJ-45 connector.



Step 7: **Remove the cable from the tool and check that all of the pins are down.** Take the connector out of the tool and look at the pins to see that they're all pushed down in an even line. Lightly tug at the connector to make sure it's attached to the cable.

- If any of the pins aren't pushed down, put the wire back into the crimping tool and crimp it again.



Conclusion: We have successfully learnt crimping/cabling of RJ45 connector to Cat6 Cable using crimping tool by carefully following the above given steps.

Experiment No. 2

Aim: To use basic networking commands in Linux (ping, tracert, nslookup, netstat, ARP, RARP, ip, ifconfig, dig, route)

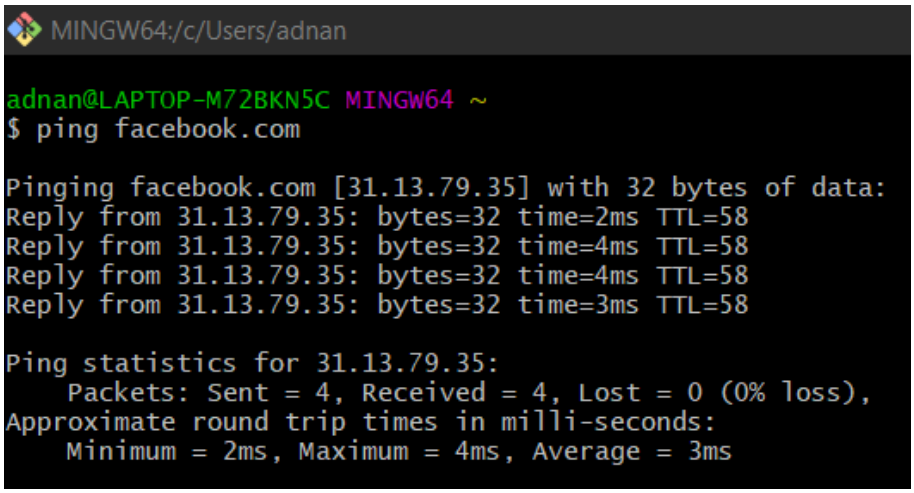
Requirements: Windows/Linux/MAC OS in PC/Laptop, compatible version of terminal in OS.

Theory:

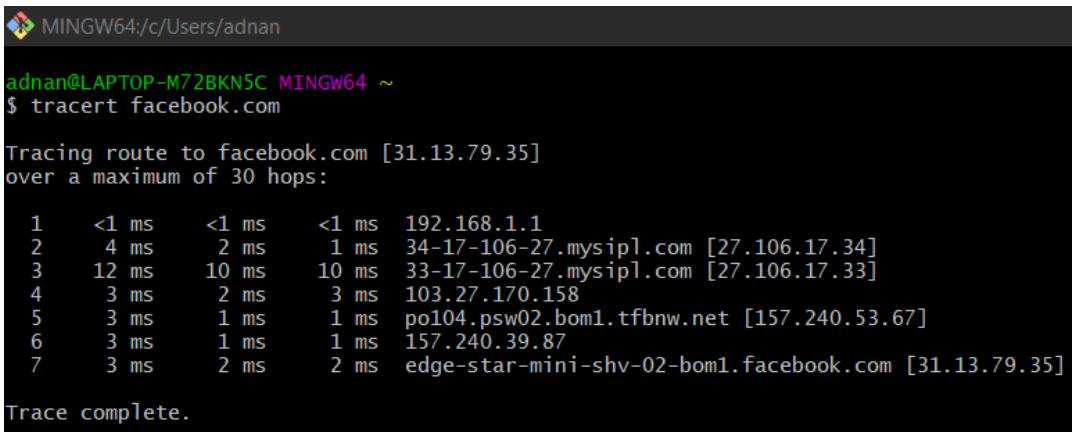
Networking commands	Operations
ping(packet internet groper)	This command is used to check the network connectivity between host and server/host.
tracert(trace route)	It's used to show the path from the source computer to the destination computer
nslookup**(**name server lookup**)**	It translates a domain name to an IP address and vice versa
netstat	Displays active TCP connections, ports on which the computer is listening, Ethernet statistics, the IP routing table, IPv4 statistics (for the IP, ICMP, TCP, and UDP protocols), and IPv6 statistics (for the IPv6, ICMPv6, TCP over IPv6, and UDP over IPv6 protocols).
arp(address resolution protocol)	Displays and modifies entries in the Address Resolution Protocol (ARP) cache.
rarp	RARP provides the opposite service to ARP in that it is used when only the ethernet address is known and the IP address is needed.
ip	This is used to assign an address to a network interface and/or configure network interface parameters on Linux operating systems
ipconfig/ifconfig	Displays all current TCP/IP network configuration values and refreshes Dynamic Host Configuration Protocol (DHCP) and Domain Name System (DNS) settings
dig(domain information groper)	The dig command, allows you to query information about various DNS records, including host addresses, mail exchanges, and name servers.
route	Displays and modifies the entries in the local IP routing table.

Commands and Output:

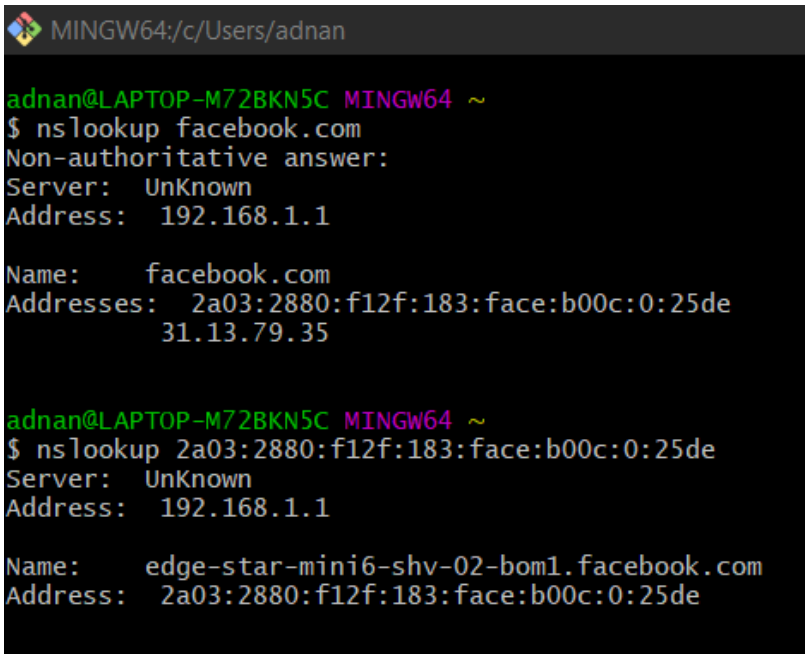
ping:



tracert:



nslookup:



netstat:

```
MINGW64:/c/Users/adnan

adnan@LAPTOP-M72BKN5C MINGW64 ~
$ netstat -ao

Active Connections

  Proto  Local Address          Foreign Address        State           PID
  TCP    0.0.0.0:135            LAPTOP-M72BKN5C:0      LISTENING       1320
  TCP    0.0.0.0:445            LAPTOP-M72BKN5C:0      LISTENING       4
  TCP    0.0.0.0:3306           LAPTOP-M72BKN5C:0      LISTENING       6344
  TCP    0.0.0.0:5040           LAPTOP-M72BKN5C:0      LISTENING       10180
  TCP    0.0.0.0:5357           LAPTOP-M72BKN5C:0      LISTENING       4
  TCP    0.0.0.0:5432           LAPTOP-M72BKN5C:0      LISTENING       6540
  TCP    0.0.0.0:33060          LAPTOP-M72BKN5C:0      LISTENING       6344
  TCP    0.0.0.0:49664          LAPTOP-M72BKN5C:0      LISTENING       900
  TCP    0.0.0.0:49665          LAPTOP-M72BKN5C:0      LISTENING       1004
  TCP    0.0.0.0:49666          LAPTOP-M72BKN5C:0      LISTENING       1972
  TCP    0.0.0.0:49667          LAPTOP-M72BKN5C:0      LISTENING       2216
  TCP    0.0.0.0:49668          LAPTOP-M72BKN5C:0      LISTENING       4648
  TCP    0.0.0.0:49676          LAPTOP-M72BKN5C:0      LISTENING       932
  TCP    0.0.0.0:50128          LAPTOP-M72BKN5C:0      LISTENING       4
  TCP    127.0.0.1:3213         LAPTOP-M72BKN5C:0      LISTENING       5924
  TCP    127.0.0.1:5354         LAPTOP-M72BKN5C:0      LISTENING       5028
  TCP    127.0.0.1:5354         LAPTOP-M72BKN5C:49670  ESTABLISHED     5028
  TCP    127.0.0.1:5354         LAPTOP-M72BKN5C:49671  ESTABLISHED     5028
  TCP    127.0.0.1:5939         LAPTOP-M72BKN5C:0      LISTENING       5584
  TCP    127.0.0.1:27015        LAPTOP-M72BKN5C:0      LISTENING       5020
  TCP    127.0.0.1:49670        LAPTOP-M72BKN5C:5354   ESTABLISHED     5020
  TCP    127.0.0.1:49671        LAPTOP-M72BKN5C:5354   ESTABLISHED     5020
  TCP    127.0.0.1:49672        LAPTOP-M72BKN5C:49673  ESTABLISHED     6344
  TCP    127.0.0.1:49673        LAPTOP-M72BKN5C:49672  ESTABLISHED     6344
```

arp:

```
MINGW64:/c/Users/adnan

adnan@LAPTOP-M72BKN5C MINGW64 ~
$ arp -a

Interface: 192.168.56.1 --- 0x7
  Internet Address      Physical Address      Type
  192.168.56.255        ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static

Interface: 192.168.1.108 --- 0xe
  Internet Address      Physical Address      Type
  192.168.1.1           38-6b-1c-be-27-71     dynamic
  192.168.1.105         34-ce-00-23-01-1d     dynamic
  192.168.1.255         ff-ff-ff-ff-ff-ff     static
  224.0.0.2             01-00-5e-00-00-02     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static
```

ip:

```
adnan@adnan-VirtualBox: $ ip a
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
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:17:b3:0f brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
       valid_lft 85211sec preferred_lft 85211sec
    inet6 fe80::a614:9179:6a9f:bf95/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
adnan@adnan-VirtualBox: $ ip link ls up
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000
    link/ether 08:00:27:17:b3:0f brd ff:ff:ff:ff:ff:ff
adnan@adnan-VirtualBox: $
```

Ipconfig/ifconfig:

```
adnan@LAPTOP-M72BKN5C MINGW64 ~
$ ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Ethernet adapter Ethernet 2:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::bde0:e9ac:a1e2:4330%7
   IPv4 Address. . . . . . . . . . . : 192.168.56.1
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Wireless LAN adapter Local Area Connection* 1:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 2:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::f9b5:dc45:7c55:698a%14
   IPv4 Address. . . . . . . . . . . : 192.168.1.108
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1

Ethernet adapter Ethernet 3:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
```

dig:

```
adnan@adnan-VirtualBox:~$ dig linux.org

; <<>> DiG 9.16.6-Ubuntu <<>> linux.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 52074
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;linux.org.                     IN      A

;; ANSWER SECTION:
linux.org.              300     IN      A       104.21.50.111
linux.org.              300     IN      A       172.67.161.161

;; Query time: 8 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Sun Aug 01 18:46:18 IST 2021
;; MSG SIZE  rcvd: 70
```

route:

```
adnan@LAPTOP-M72BKN5C MINGW64 ~
$ route print
===========================================================================
Interface List
  6...54 05 db 10 4a 64 ......Realtek PCIe GbE Family Controller
  7...0a 00 27 00 00 07 ......VirtualBox Host-Only Ethernet Adapter
 21...a4 b1 c1 17 71 a5 ......Microsoft Wi-Fi Direct Virtual Adapter
 11...a6 b1 c1 17 71 a4 ......Microsoft Wi-Fi Direct Virtual Adapter #2
 14...a4 b1 c1 17 71 a4 ......Intel(R) Wi-Fi 6 AX201 160MHz
  9...00 ff 6c d4 a8 15 ......TeamViewer VPN Adapter
 20...a4 b1 c1 17 71 a8 ......Bluetooth Device (Personal Area Network)
  1...........................Software Loopback Interface 1
===========================================================================

IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0      192.168.1.1    192.168.1.108     35
        127.0.0.0        255.0.0.0         On-link         127.0.0.1    331
        127.0.0.1  255.255.255.255         On-link         127.0.0.1    331
  127.255.255.255  255.255.255.255         On-link         127.0.0.1    331
      192.168.1.0    255.255.255.0         On-link     192.168.1.108    291
    192.168.1.108  255.255.255.255         On-link     192.168.1.108    291
    192.168.1.255  255.255.255.255         On-link     192.168.1.108    291
     192.168.56.0    255.255.255.0         On-link      192.168.56.1    281
     192.168.56.1  255.255.255.255         On-link      192.168.56.1    281
   192.168.56.255  255.255.255.255         On-link      192.168.56.1    281
        224.0.0.0        240.0.0.0         On-link         127.0.0.1    331
        224.0.0.0        240.0.0.0         On-link      192.168.56.1    281
        224.0.0.0        240.0.0.0         On-link     192.168.1.108    291
  255.255.255.255  255.255.255.255         On-link         127.0.0.1    331
  255.255.255.255  255.255.255.255         On-link      192.168.56.1    281
  255.255.255.255  255.255.255.255         On-link     192.168.1.108    291
===========================================================================
Persistent Routes:
  None

IPv6 Route Table
===========================================================================
Active Routes:
 If Metric Network Destination      Gateway
  1    331 ::1/128                  On-link
  7    281 fe80::/64                On-link
 14    291 fe80::/64                On-link
  7    281 fe80::bde0:e9ac:a1e2:4330/128
                                    On-link
 14    291 fe80::f9b5:dc45:7c55:698a/128
                                    On-link
  1    331 ff00::/8                 On-link
  7    281 ff00::/8                 On-link
 14    291 ff00::/8                 On-link
===========================================================================
Persistent Routes:
  None
```

Conclusion: We have successfully executed and got the output of basic networking commands (ping, tracert, nslookup, netstat, ARP,RARP, ip, ifconfig, dig, route) in Linux Shell.

Experiment No. 3

Aim: To build a simple network topology and configure it for static routing protocol using packet tracer.

Requirements: Windows OS in P.C and Stable version of CISCO packet tracer.

Theory:

The arrangement of wires, work stations (P.C.) and other peripherals in a network is known as network topology.

Some of the topologies widely known and in used are Mesh Topology, Star Topology, Bus Topology, Ring Topology and Hybrid Topology (combination of two or more topology).

We're using simple network topology analogous to Star topology for static routing protocol using CISCO packet tracer.

Network Configuration:

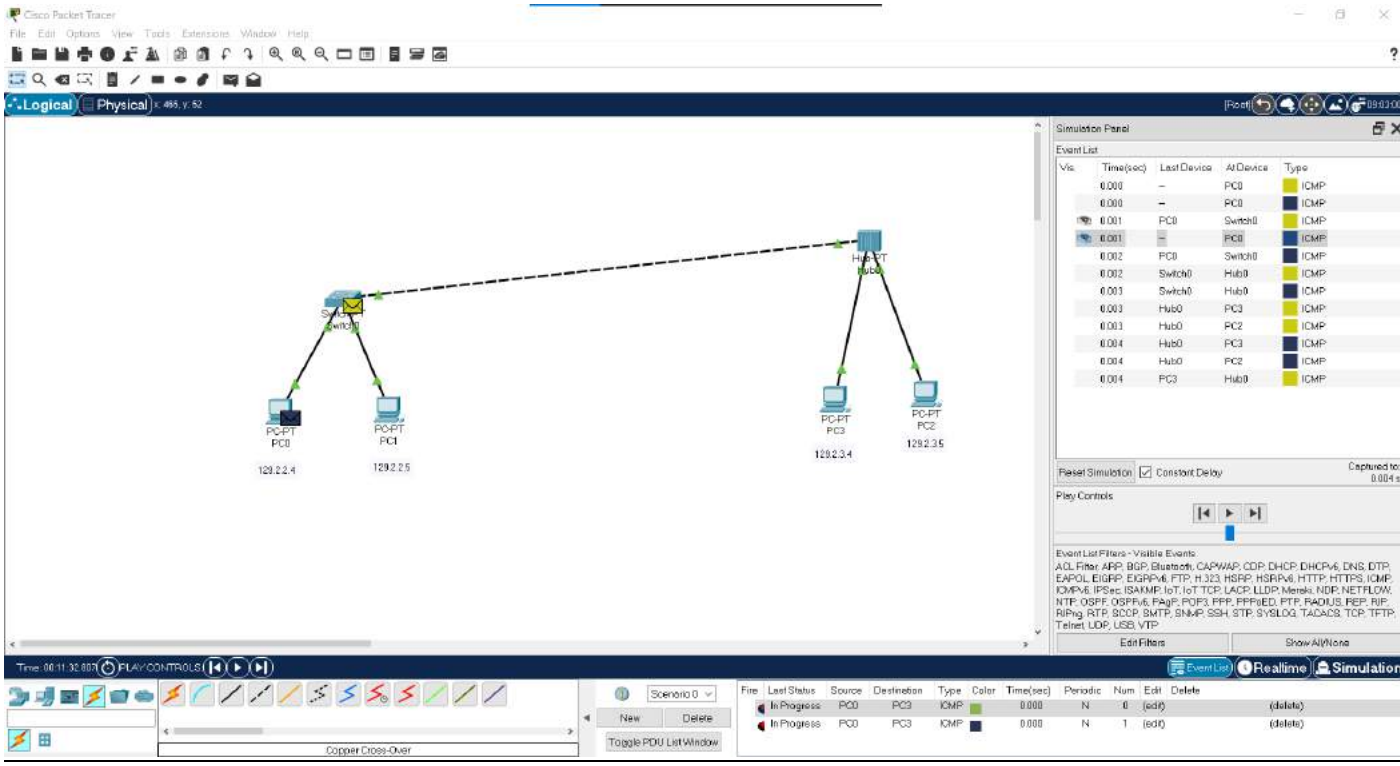
1) There are 2 different Gateways (having different address) connected through switch and hub using copper cross-over wire.

2) Switch in 1st gateway is connected to two P.C. in Star topology using copper straight-through wire each having same Gateway address but different I.P address.

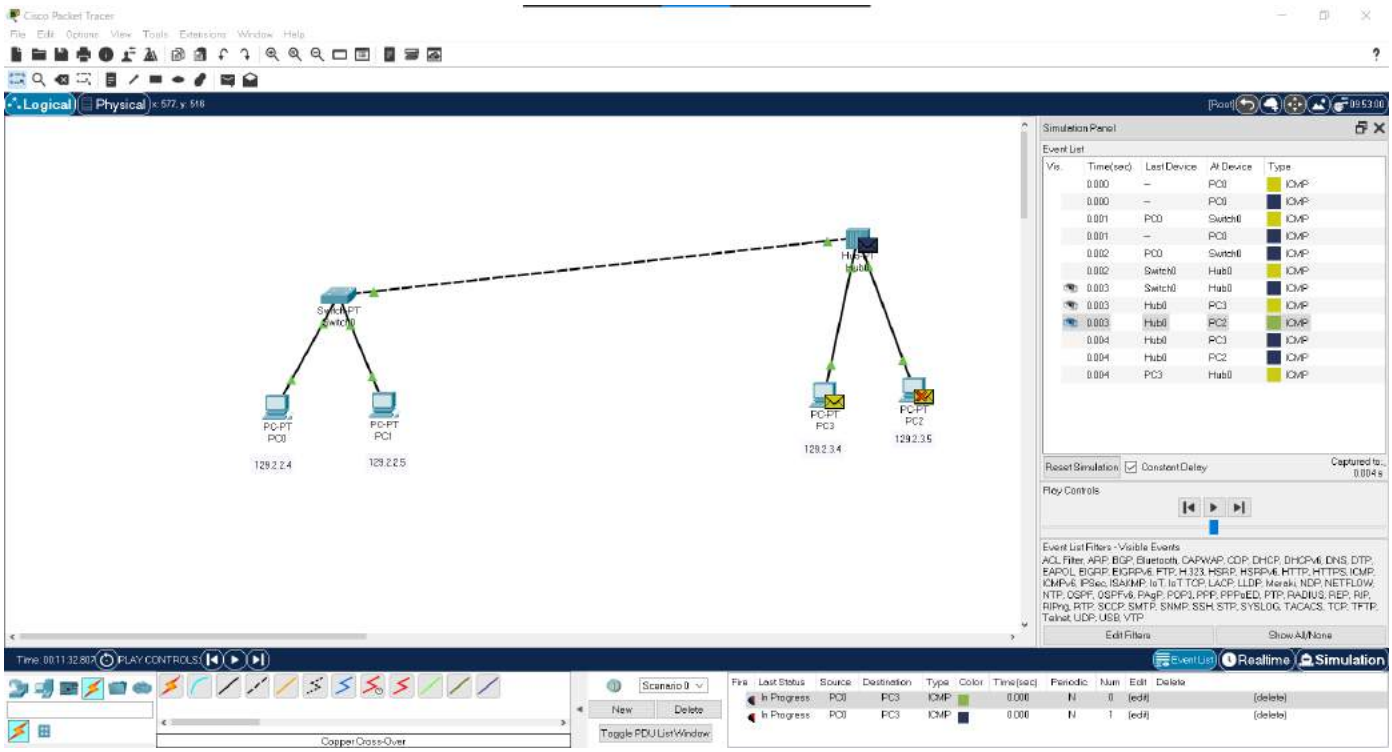
3) Hub in 2nd gateway is connected to two P.C. in Star topology using copper straight-through wire each having same Gateway address but different I.P address.

Demonstration of sending packet from PC0 (in 1st Gateway) to PC3 (in 2nd gateway):

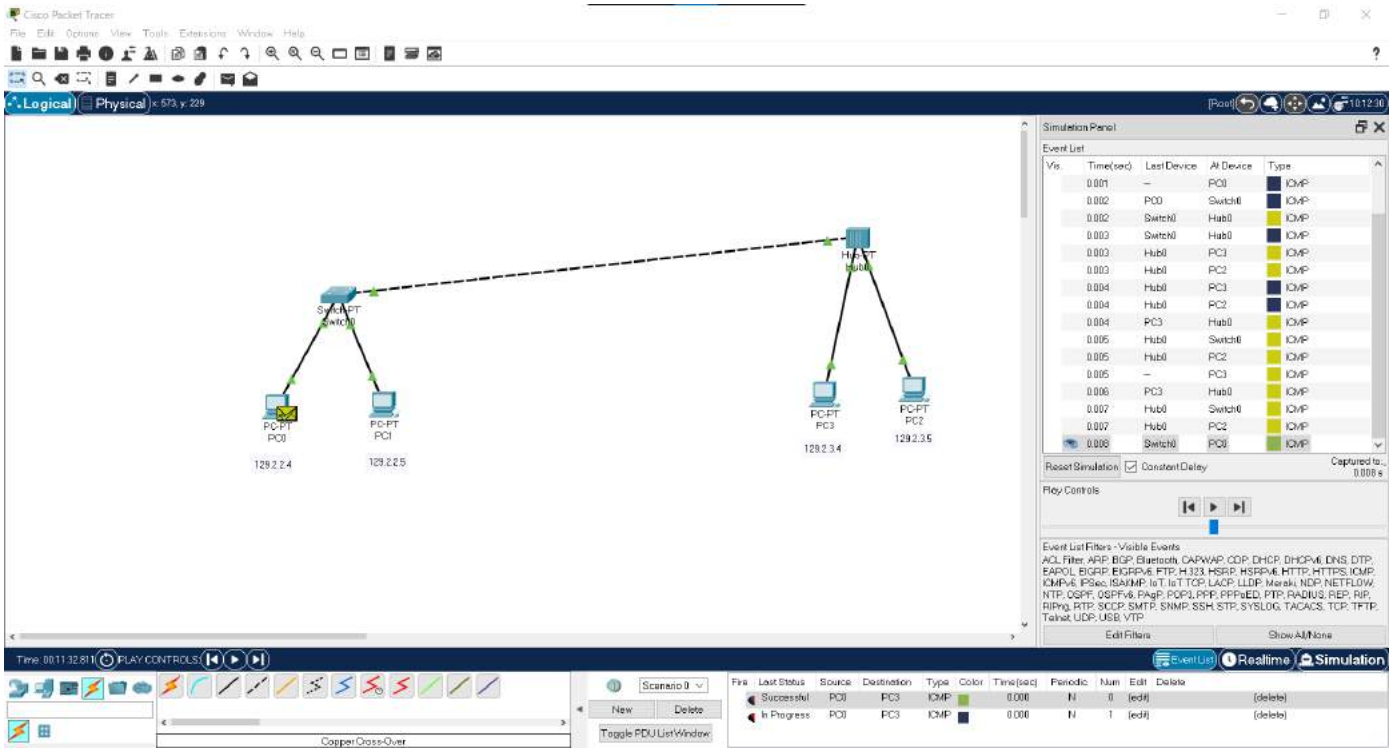
1) Packet send from PC0 to Switch:



2) Packet route to HUB from switch which in turn route to all PC in HUB network and PC3 accepted the package and other PC rejected it:



3) PC0 received acknowledgment packet has been delivered successfully:



Conclusion: We have successfully understand the concept of network topology and built a simple network topology and configure it for static routing protocol using CISCO packet tracer.

EXPERIEMENT NO. 4

Aim: To setup a network and configure IP addressing, subnetting, masking using CISCO Packet Tracer.

Requirements: Windows O.S and CISCO Packet Tracer.

Theory:

What is a subnet?

A subnet, or subnetwork, is a network inside a network. Subnets make networks more efficient. Through subnetting, network traffic can travel a shorter distance without passing through unnecessary routers to reach its destination.

Like the postal service, networks are more efficient when messages travel as directly as possible. When a network receives data packets from another network, it will sort and route those packets by subnet so that the packets do not take an inefficient route to their destination.

What is an IP address?

In order to understand subnets, we must quickly define IP addresses. Every device that connects to the Internet is assigned a unique IP (Internet Protocol) address, enabling data sent over the Internet to reach the right device out of the billions of devices connected to the Internet. While computers read IP addresses as binary code (a series of 1s and 0s), IP addresses are usually written as a series of alphanumeric characters.

What do the different parts of an IP address mean?

This section focuses on IPv4 addresses, which are presented in the form of four decimal numbers separated by periods, like 203.0.113.112. (IPv6 addresses are longer and use letters as well as numbers.)

Every IP address has two parts. The first part indicates which network the address belongs to. The second part specifies the device within that network. However, the length of the "first part" changes depending on the network's class.

Networks are categorized into different classes, labeled A through E. Class A networks can connect millions of devices. Class B networks and Class C networks are progressively smaller in size. (Class D and Class E networks are not commonly used.)

Let's break down how these classes affect IP address construction:

Class A network: Everything before the first period indicates the network, and everything after it specifies the device within that network. Using 203.0.113.112 as an example, the network is indicated by "203" and the device by "0.113.112."

Class B network: Everything before the second period indicates the network. Again using 203.0.113.112 as an example, "203.0" indicates the network and "113.112" indicates the device within that network.

Class C network: For Class C networks, everything before the third period indicates the network. Using the same example, "203.0.113" indicates the Class C network, and "112" indicates the device.

Why is subnetting necessary?

In a Block A network (for instance), there could be millions of connected devices, and it could take some time for the data to find the right device. This is why subnetting comes in handy: subnetting narrows down the IP address to usage within a range of devices.

Because an IP address is limited to indicating the network and the device address, IP addresses cannot be used to indicate which subnet an IP packet should go to. Routers within a network use something called a subnet mask to sort data into subnetworks.

What is a subnet mask?

A subnet mask is like an IP address, but for only internal usage within a network. Routers use subnet masks to route data packets to the right place. Subnet masks are not indicated within data packets traversing the Internet — those packets only indicate the destination IP address, which a router will match with a subnet.

Output:

Original network (192.168.16.0) is divided into 4-subnet:

Block A subnet network address: 192.168.16.0

Block A subnet First address: 192.168.16.1

Block A subnet network address: 192.168.16.254

Block A subnet Broadcast address: 192.168.16.255

Block B subnet network address: 192.168.17.0

Block B subnet First address: 192.168.17.1

Block B subnet network address: 192.168.17.126

Block B subnet Broadcast address: 192.168.17.127

Block C subnet network address: 192.168.17.128

Block C subnet First address: 192.168.17.129

Block C subnet network address: 192.168.17.190

Block C subnet Broadcast address: 192.168.17.191

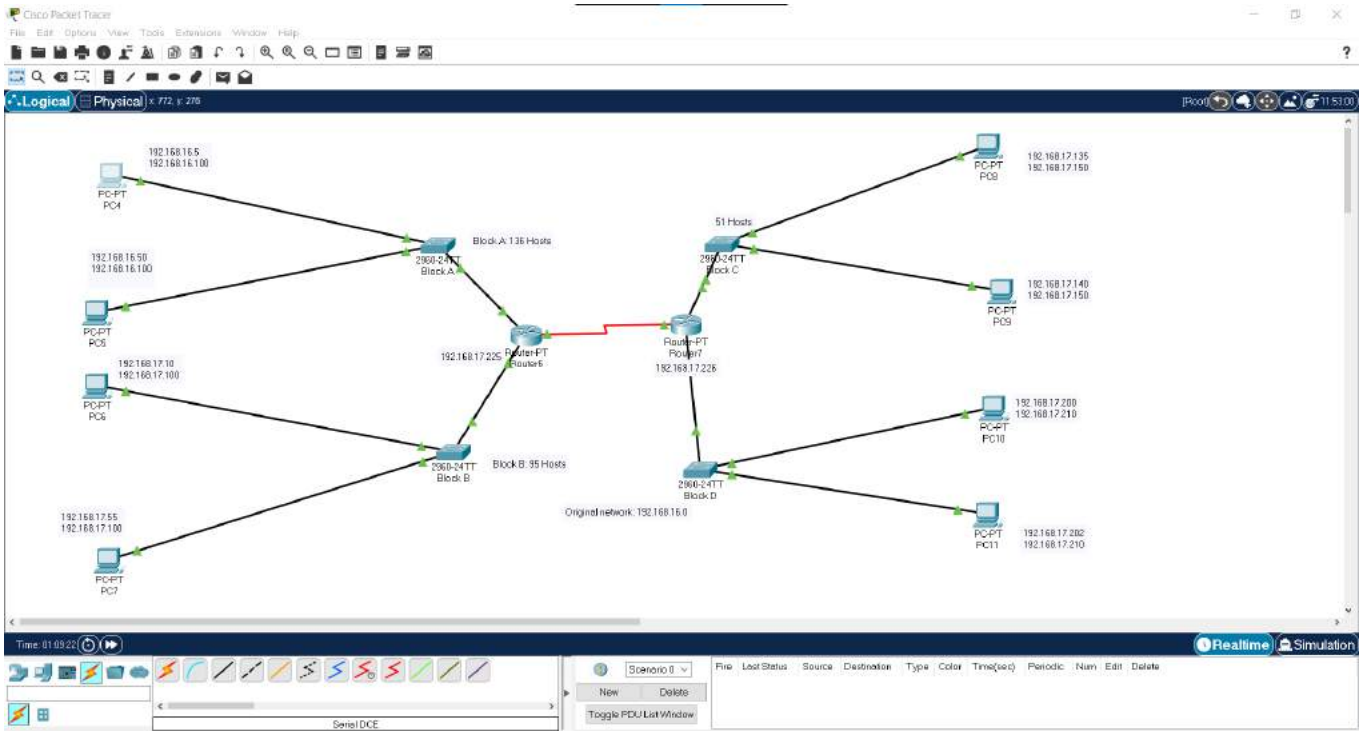
Block D subnet network address: 192.168.17.192

Block D subnet First address: 192.168.17.193

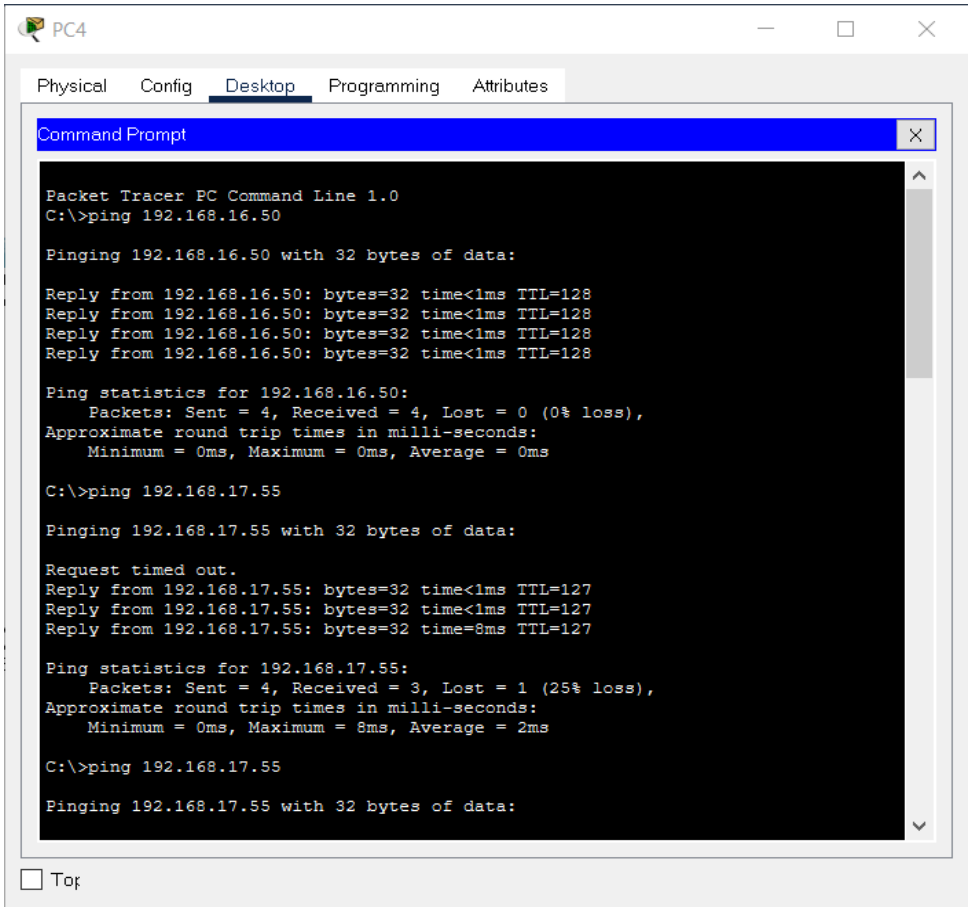
Block D subnet network address: 192.168.17.222

Block D subnet Broadcast address: 192.168.17.223

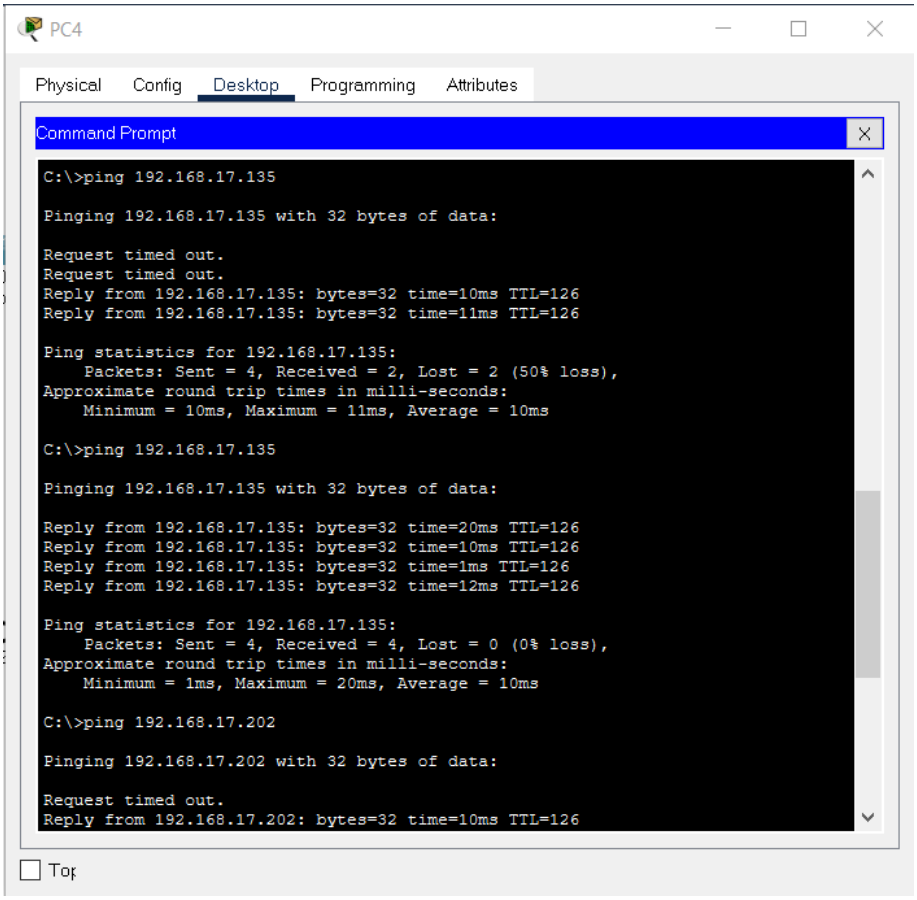
Network:



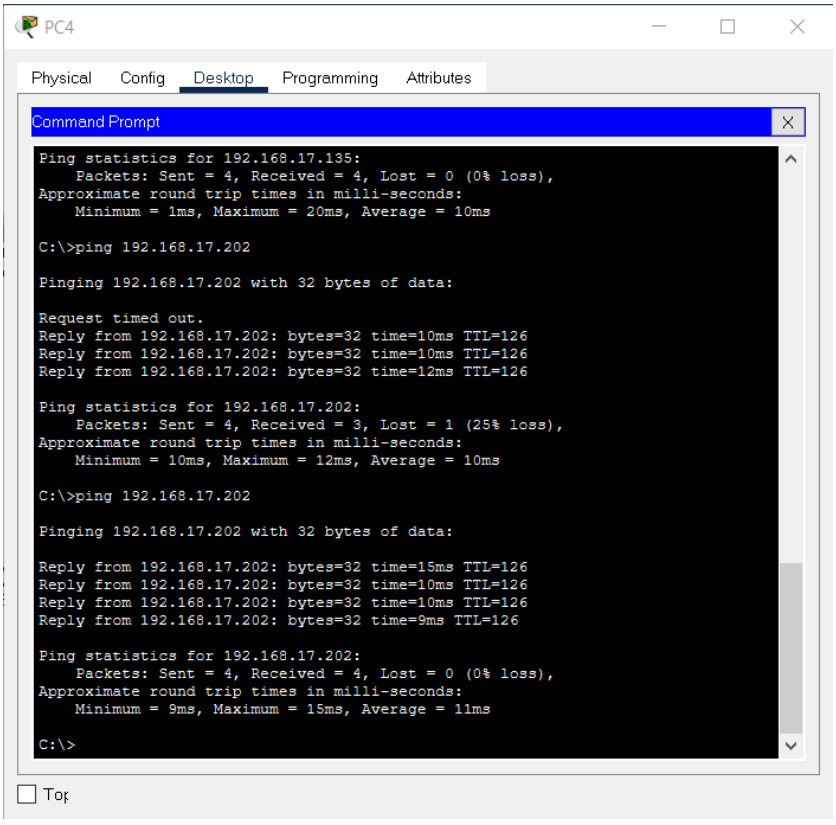
Pinging from Block A P.C to Block A and B P.C:



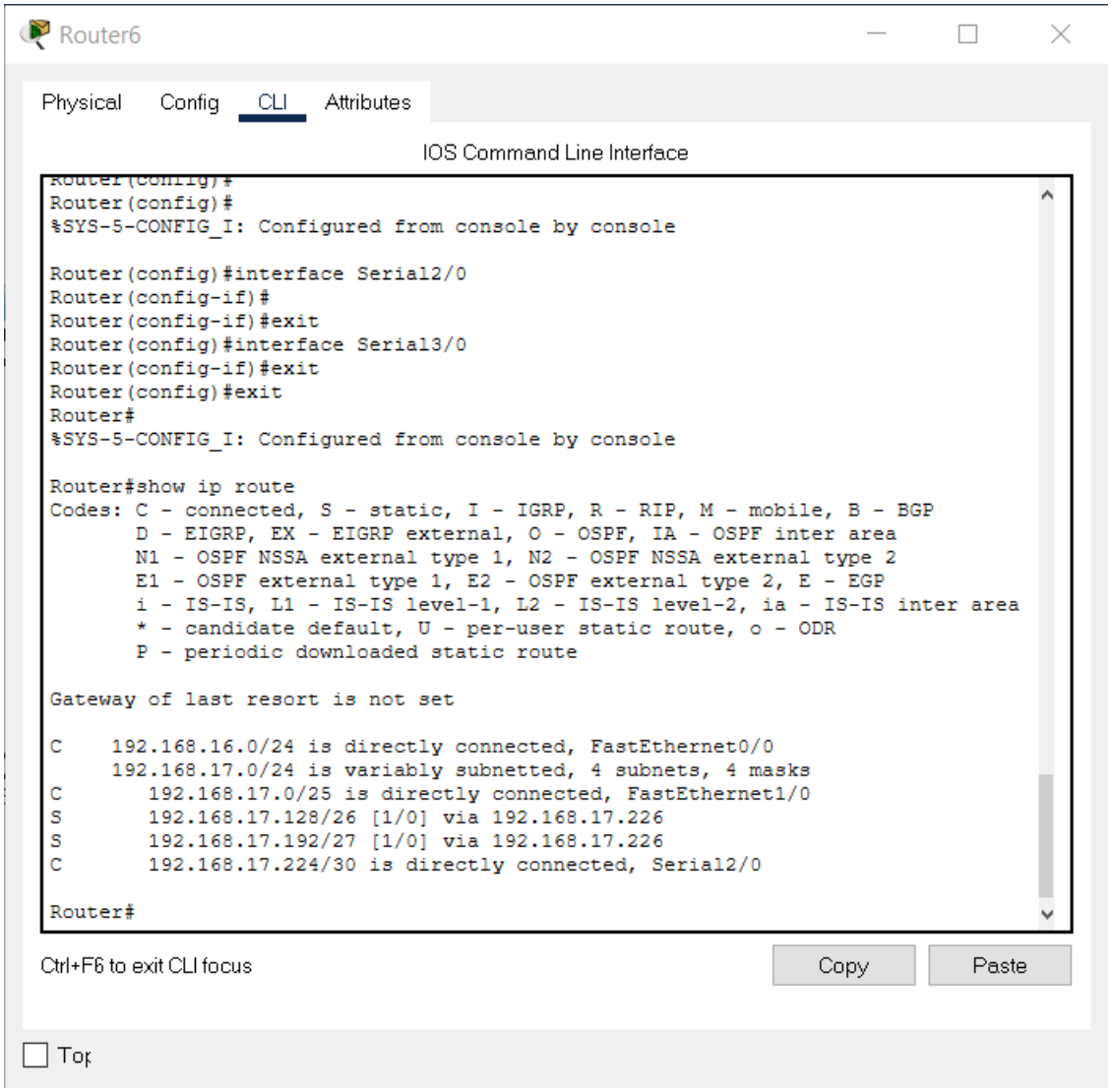
Pinging from Block A P.C to Block C P.C:



Pinging from Block A P.C to Block D P.C:



IP routing table of router after configuration:



Conclusion: We have successfully setup a network and configure IP address to each P.Cs and routers and configured 4 subnets and routing table of routers to send packet packet from one subnet block to another.

Experiment No. 5

Aim: To set up multiple IP addresses on a single LAN and using netstat and route commands viewing current routing table.

Requirement: Windows/Linux/MAC OS in PC/Laptop, compatible version of terminal in OS.

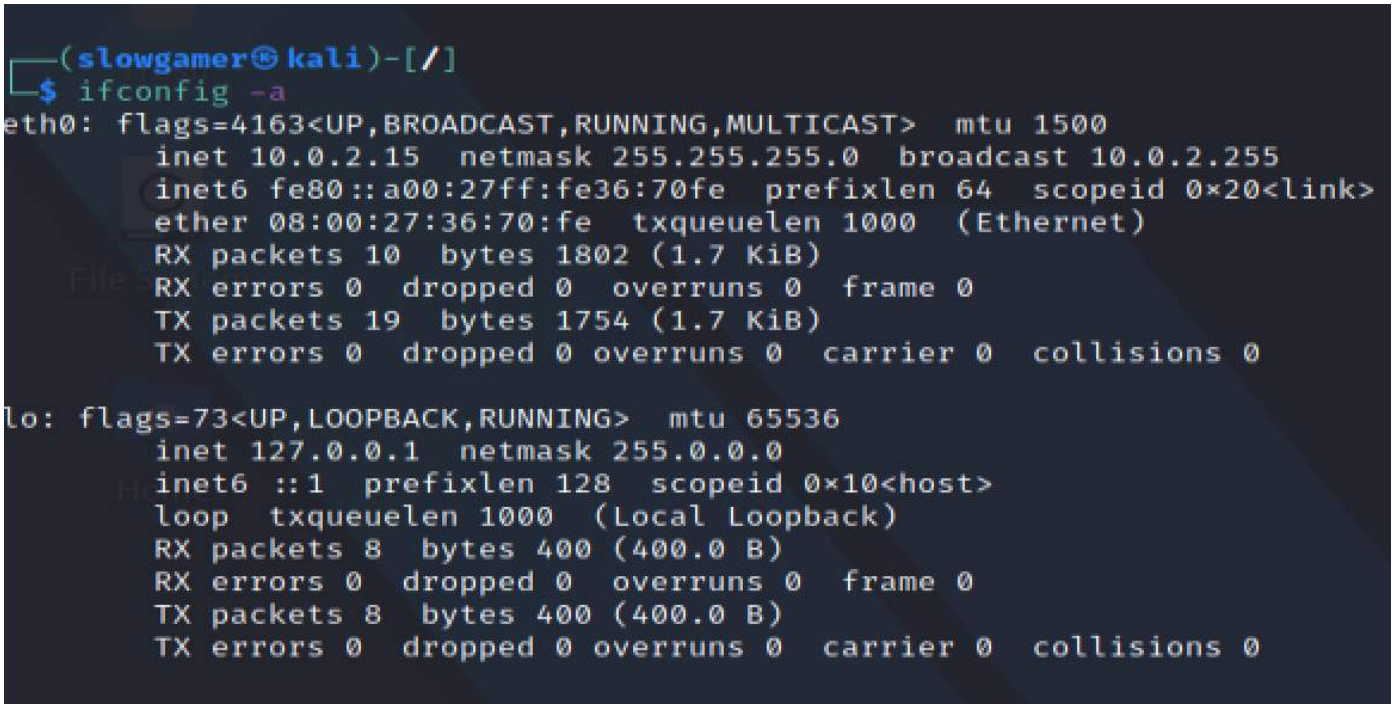
Theory:

The concept of creating or configuring multiple IP addresses on a single network interface is called IP aliasing. IP aliasing is very useful for setting up multiple virtual sites on Apache using one single network interface with different IP addresses on a single subnet network.

The main advantage of using this IP aliasing is, you don't need to have a physical adapter attached to each IP, but instead you can create multiple or many virtual interfaces (aliases) to a single physical card.

Below we create virtual interface and assign multiple IP Address in Kali Linux:

1) Before ipaliasing:



2) After ipaliasing:

```
┌──(slowgamer㉿kali)-[/]
└─$ sudo ifconfig eth0:0 10.0.1.15 up

┌──(slowgamer㉿kali)-[/]
└─$ sudo ifconfig eth0:1 10.0.1.16 up

┌──(slowgamer㉿kali)-[/]
└─$ sudo ifconfig eth0:2 10.0.1.17 up

┌──(slowgamer㉿kali)-[/]
└─$ ifconfig -a
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::a00:27ff:fe36:70fe  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:36:70:fe  txqueuelen 1000  (Ethernet)
        RX packets 10  bytes 1802 (1.7 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 19  bytes 1754 (1.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth0:0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.1.15  netmask 255.0.0.0  broadcast 10.255.255.255
        ether 08:00:27:36:70:fe  txqueuelen 1000  (Ethernet)

eth0:1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.1.16  netmask 255.0.0.0  broadcast 10.255.255.255
        ether 08:00:27:36:70:fe  txqueuelen 1000  (Ethernet)

eth0:2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.1.17  netmask 255.0.0.0  broadcast 10.255.255.255
        ether 08:00:27:36:70:fe  txqueuelen 1000  (Ethernet)

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)
        RX packets 8  bytes 400 (400.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 400 (400.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

3) Netstat and route command:

```
┌──(slowgamer㉿kali)-[~]
└─$ netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         10.0.2.2        0.0.0.0         UG        0 0          0 eth0
10.0.0.0        0.0.0.0         255.0.0.0       U         0 0          0 eth0
10.0.1.15       0.0.0.0         255.255.255.255 UH        0 0          0 eth0
10.0.1.16       0.0.0.0         255.255.255.255 UH        0 0          0 eth0
10.0.1.17       0.0.0.0         255.255.255.255 UH        0 0          0 eth0
10.0.1.18       0.0.0.0         255.255.255.255 UH        0 0          0 eth0
10.0.2.0        0.0.0.0         255.255.255.0   U         0 0          0 eth0

┌──(slowgamer㉿kali)-[~]
└─$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.0.2.2        0.0.0.0         UG    100    0        0 eth0
10.0.0.0        0.0.0.0         255.0.0.0       U     0      0        0 eth0
10.0.1.15       0.0.0.0         255.255.255.255 UH    0      0        0 eth0
10.0.1.16       0.0.0.0         255.255.255.255 UH    0      0        0 eth0
10.0.1.17       0.0.0.0         255.255.255.255 UH    0      0        0 eth0
10.0.1.18       0.0.0.0         255.255.255.255 UH    0      0        0 eth0
10.0.2.0        0.0.0.0         255.255.255.0   U     100    0        0 eth0
```

Conclusion: We have successfully added Multiple IP addresses to single NIC in Kali Linux OS and using netstat and route command we have displayed routing table.

Experiment No. 6

Aim: To perform remote login using Telnet server.

Requirement: Windows/Linux OS in P.C., CISCO Packet Tracer.

Theory:

Telnet:

Telnet is a network protocol used to virtually access a computer and to provide a two-way, collaborative and text-based communication channel between two machines. It follows a user command Transmission Control Protocol/Internet Protocol (TCP/IP) networking protocol for creating remote sessions. On the web, Hypertext Transfer Protocol (HTTP) and File Transfer Protocol (FTP) simply enable users to request specific files from remote computers, while, through Telnet, users can log on as a regular user with the privileges they are granted to the specific applications and data on that computer.

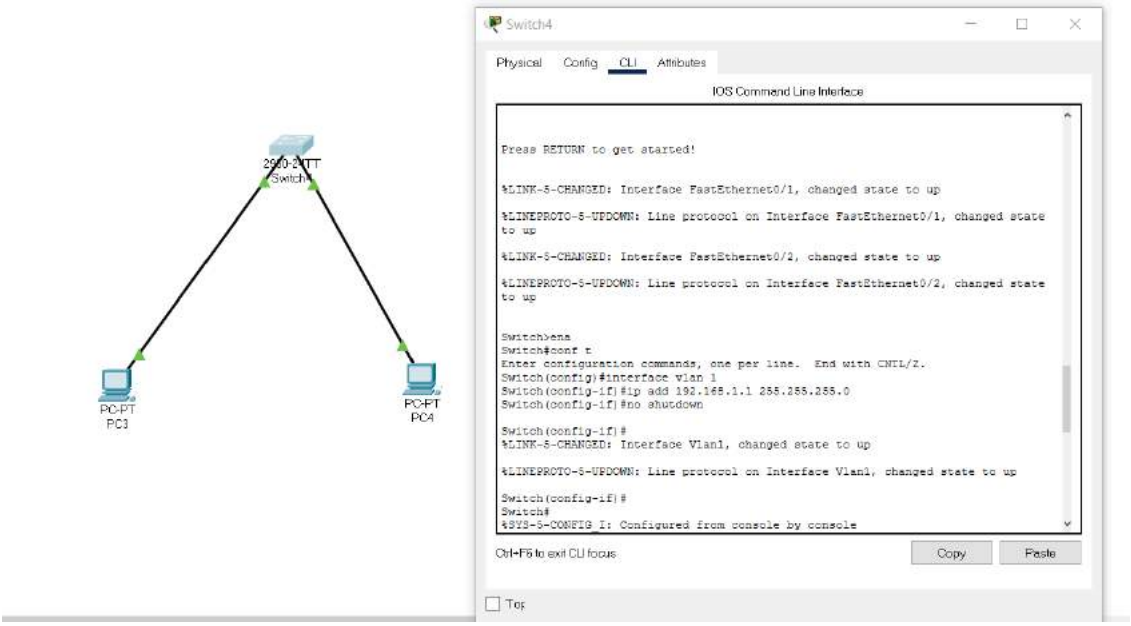
Telnet Working:

Telnet is a type of client-server protocol that can be used to open a command line on a remote computer, typically a server. Users can utilize this tool to ping a port and find out whether it is open. Telnet works with what is called a virtual terminal connection emulator, or an abstract instance of a connection to a computer, using standard protocols to act like a physical terminal connected to a machine.

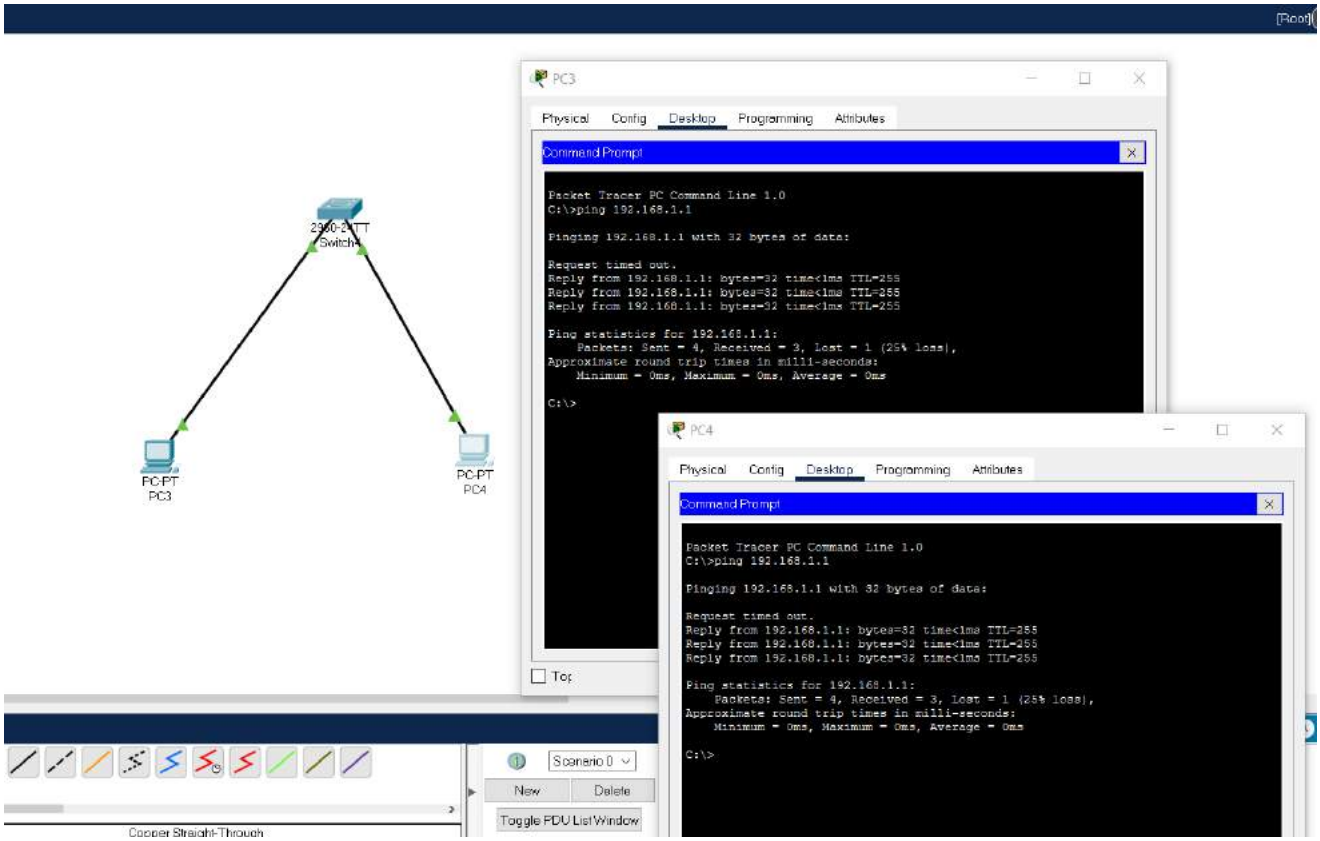
Users connect remotely to a machine using Telnet, sometimes referred to as Telnetting into the system. They are prompted to enter their username and password combination to access the remote computer, which enables the running of command lines as if logged in to the computer in person. Despite the physical location of users, their IP address will match the computer logged in to rather than the one physically used to connect.

Telnet using CISCO Packet Tracer:

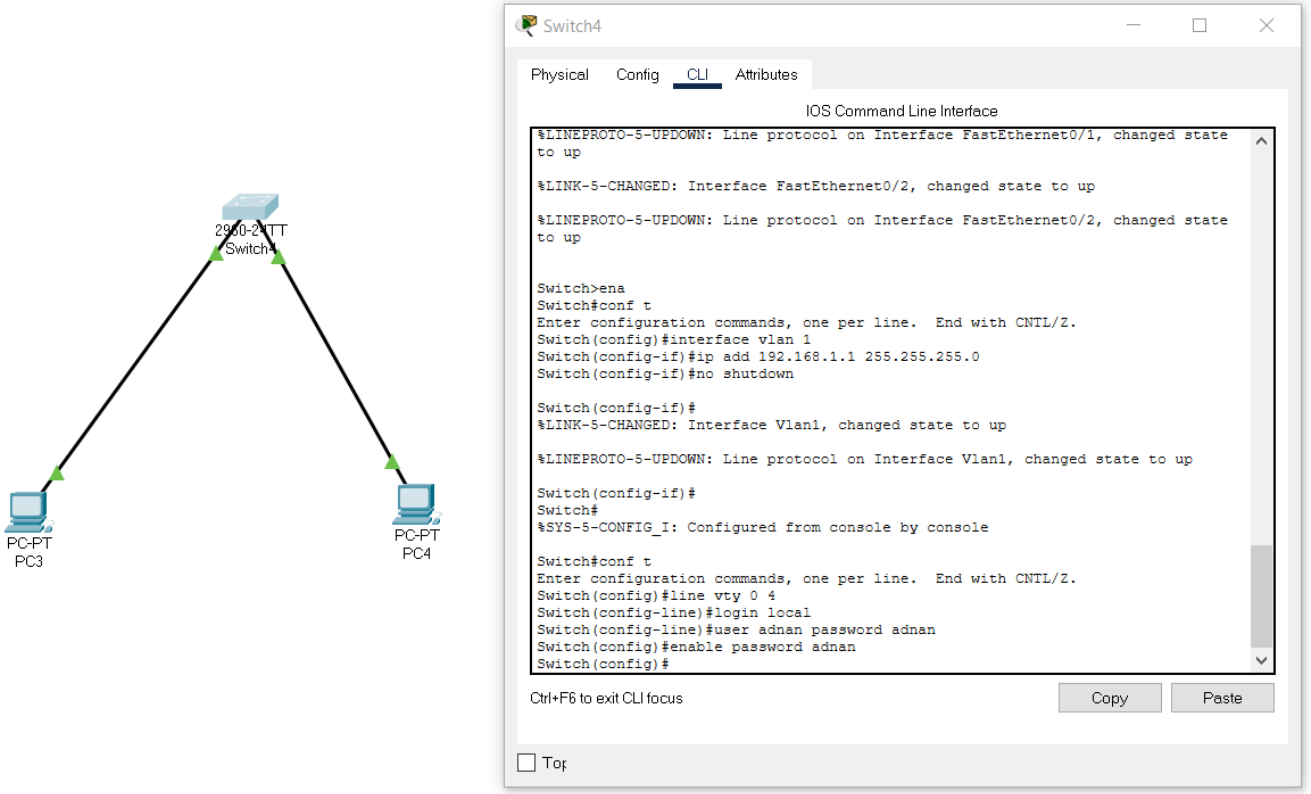
1) Configuring Switch:



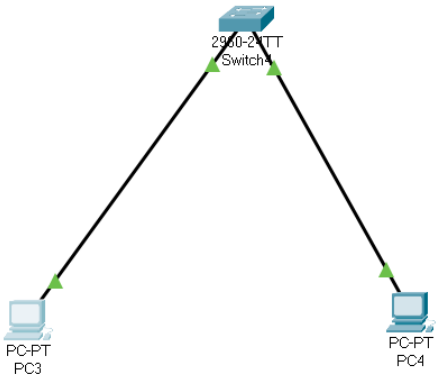
2) Pinging switch from PC3 and PC4:

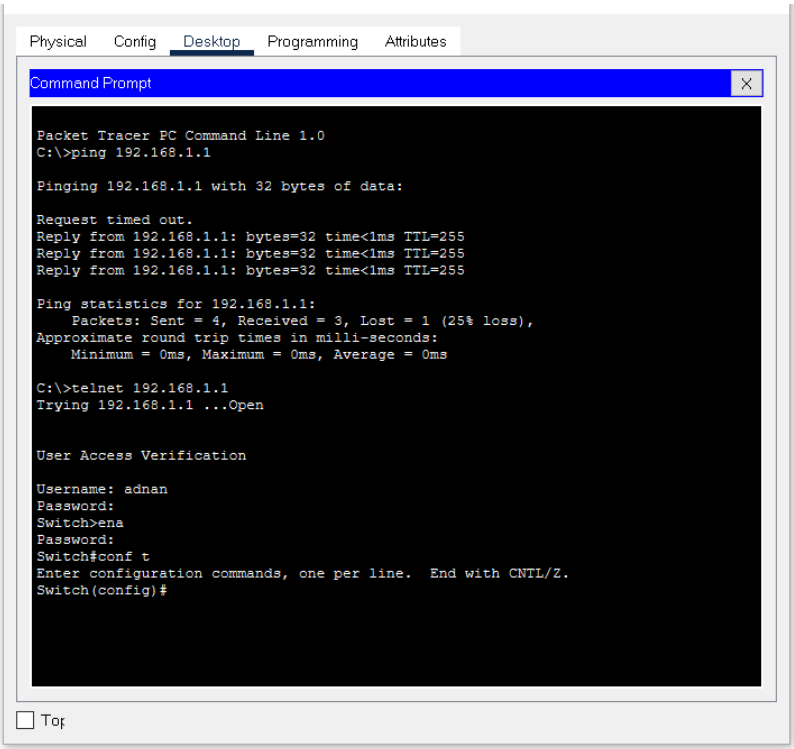


3) Configuring switch(Telnet Server) for remote login using Telnet:



4) Remote login from PC3 to Telnet Server using telnet command:





Conclusion: We have successfully remote login from P.C. to Telnet Server by performing proper configuration in CISCO Packet Tracer.

EXPERIMENT NO. 7

Aim: Use Wireshark to understand the operation of TCP/IP layers.

Requirements: Linux/Windows O.S, Compatible version of Wireshark.

Theory:

What is Wireshark?

Wireshark is a network packet analyzer. A network packet analyzer presents captured packet data in as much detail as possible.

You could think of a network packet analyzer as a measuring device for examining what's happening inside a network cable, just like an electrician uses a voltmeter for examining what's happening inside an electric cable (but at a higher level, of course).

In the past, such tools were either very expensive, proprietary, or both. However, with the advent of Wireshark, that has changed. Wireshark is available for free, is open source, and is one of the best packet analyzers available today.

Some intended purposes

Here are some reasons people use Wireshark:

- Network administrators use it to *troubleshoot network problems*
- Network security engineers use it to *examine security problems*
- QA engineers use it to *verify network applications*
- Developers use it to *debug protocol implementations*
- People use it to *learn network protocol* internals

Wireshark can also be helpful in many other situations.

Features:

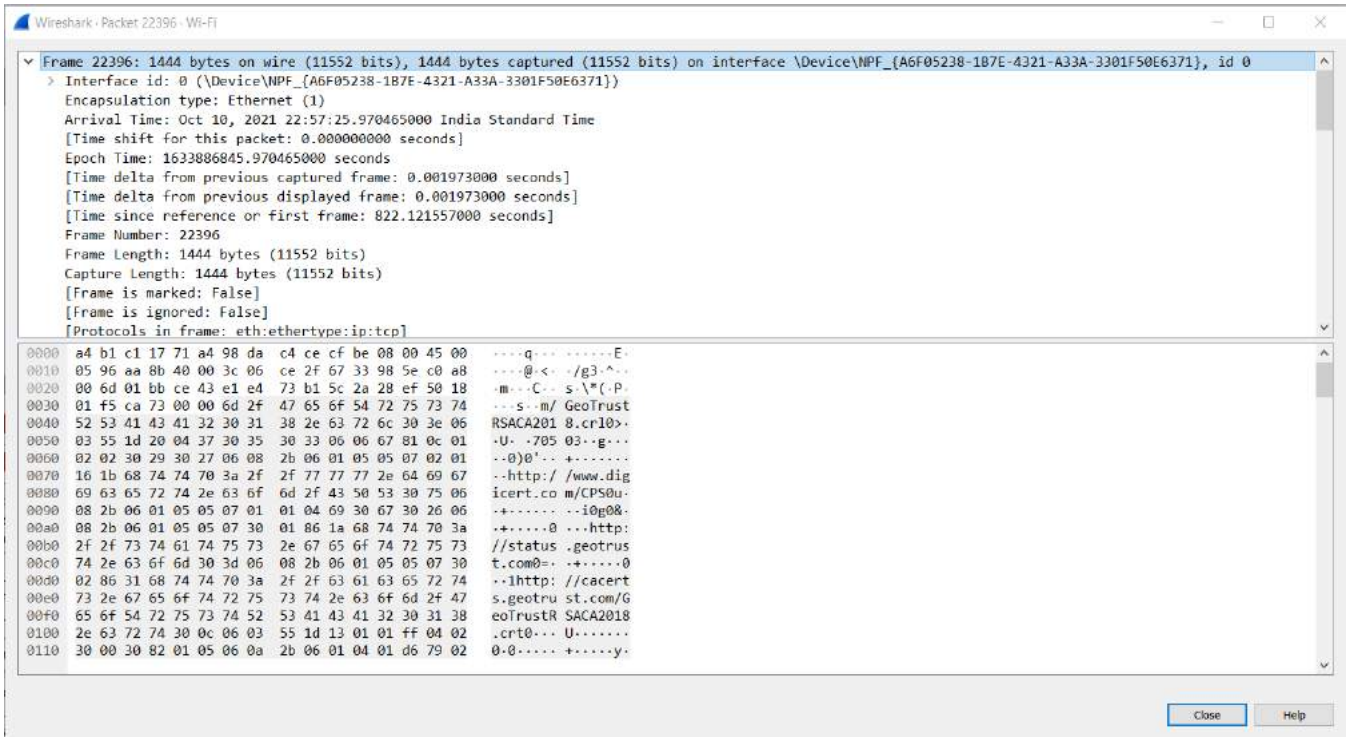
The following are some of the many features Wireshark provides:

- Available for *UNIX* and *Windows*.
- *Capture* live packet data from a network interface.
- *Open* files containing packet data captured with tcpdump/WinDump, Wireshark, and many other packet capture programs.
- *Import* packets from text files containing hex dumps of packet data.
- Display packets with *very detailed protocol information*.
- *Save* packet data captured.
- *Export* some or all packets in a number of capture file formats.
- *Filter packets* on many criteria.
- *Search* for packets on many criteria.
- *Colorize* packet display based on filters.

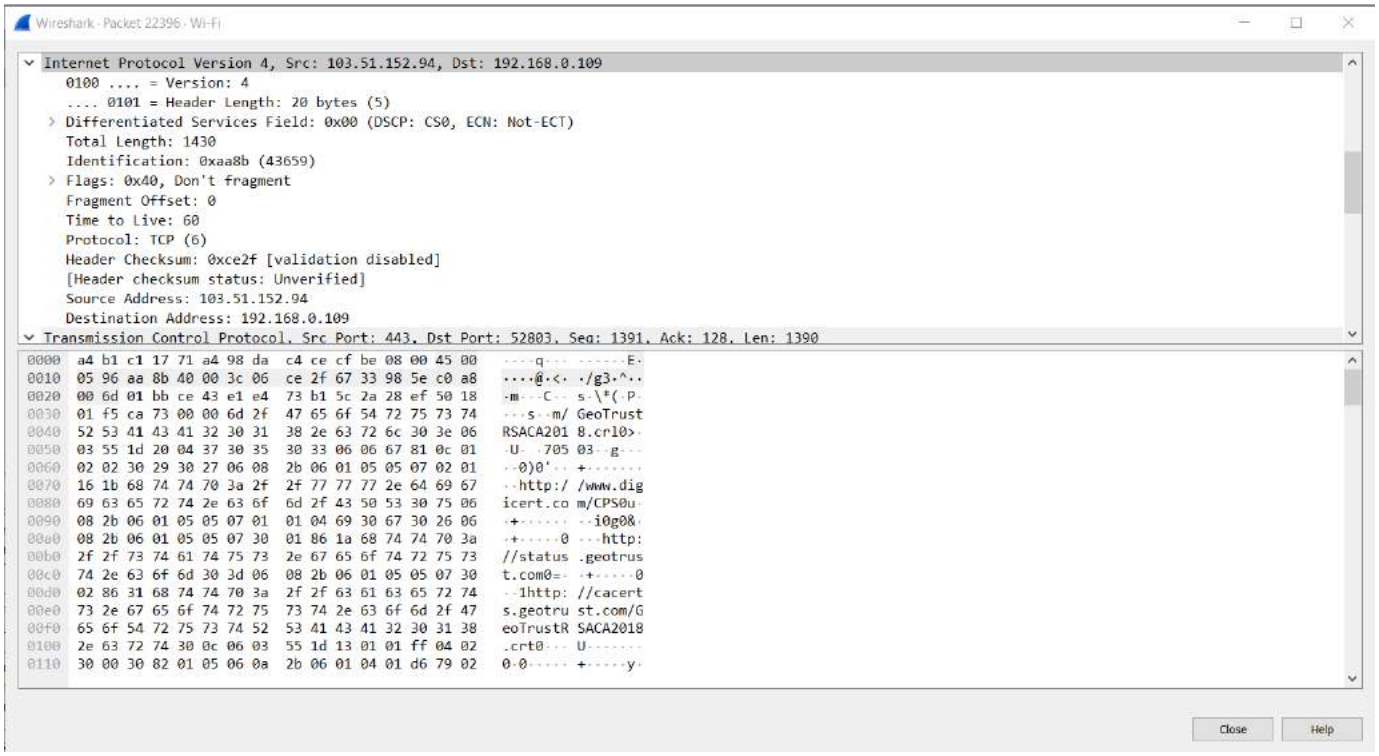
- Create various *statistics*.
- …And *a lot more!*

Wireshark Output:

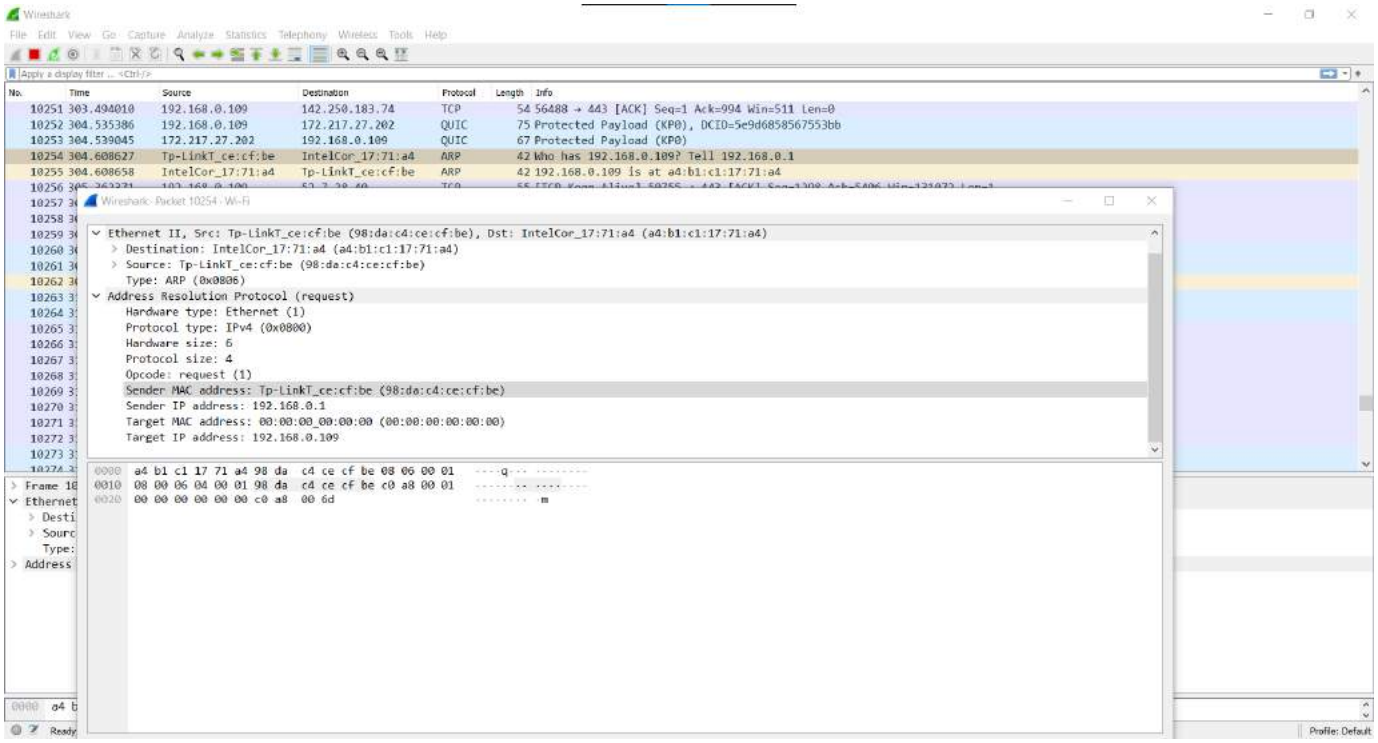
Frame Header and Frame Size:



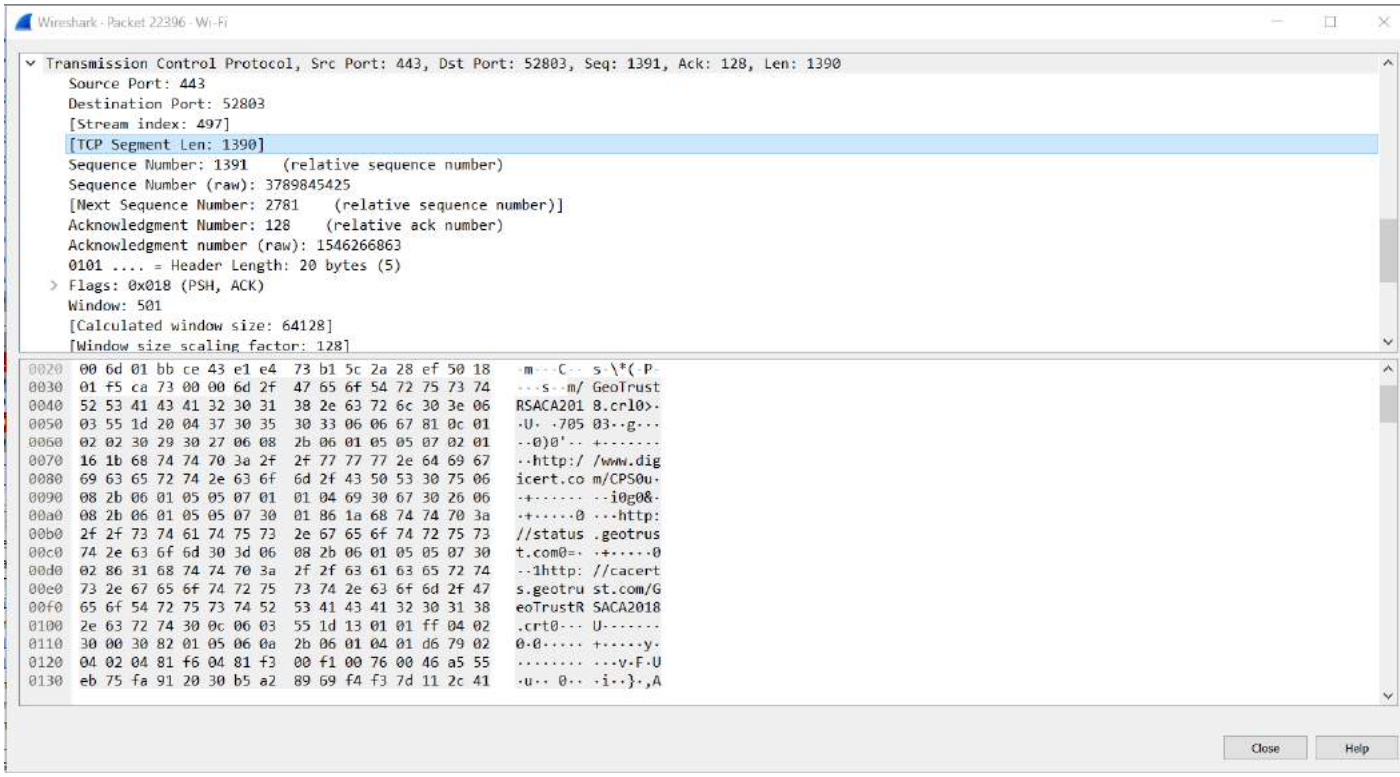
IP header:



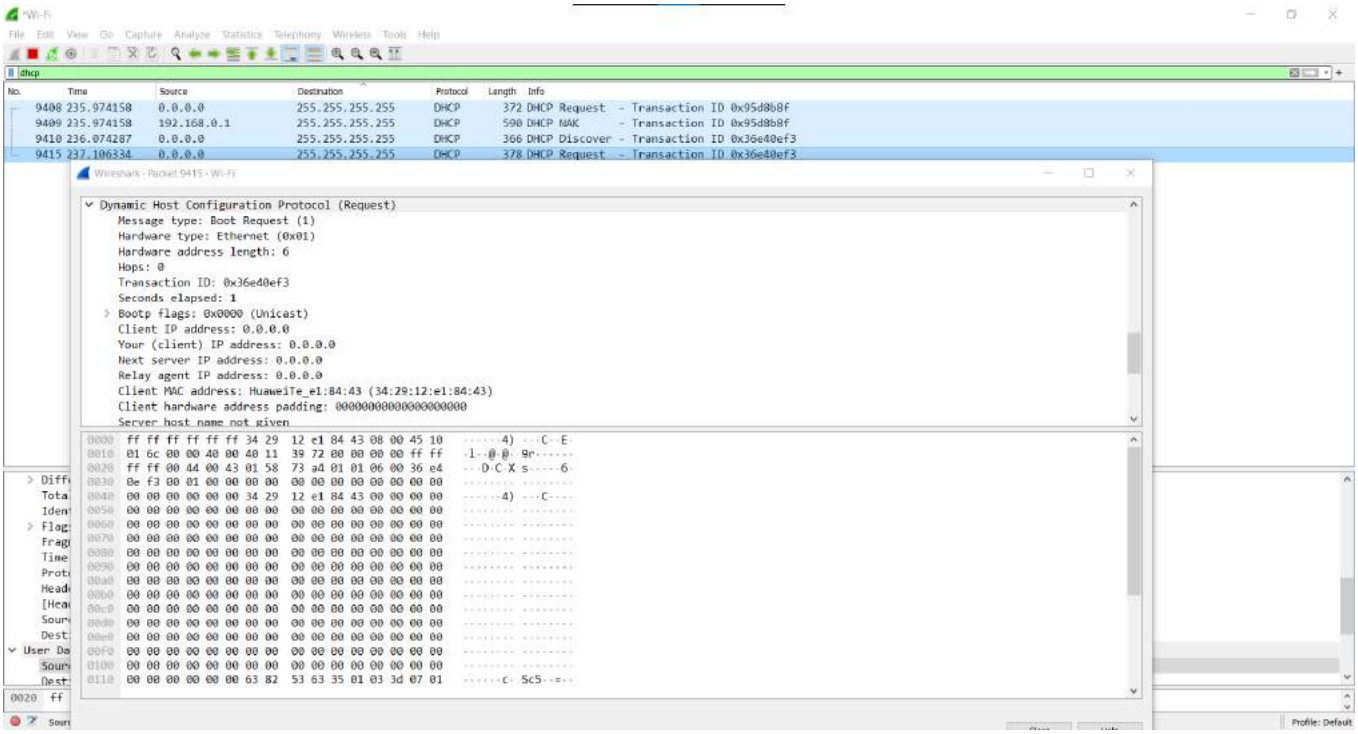
MAC Address and ARP:



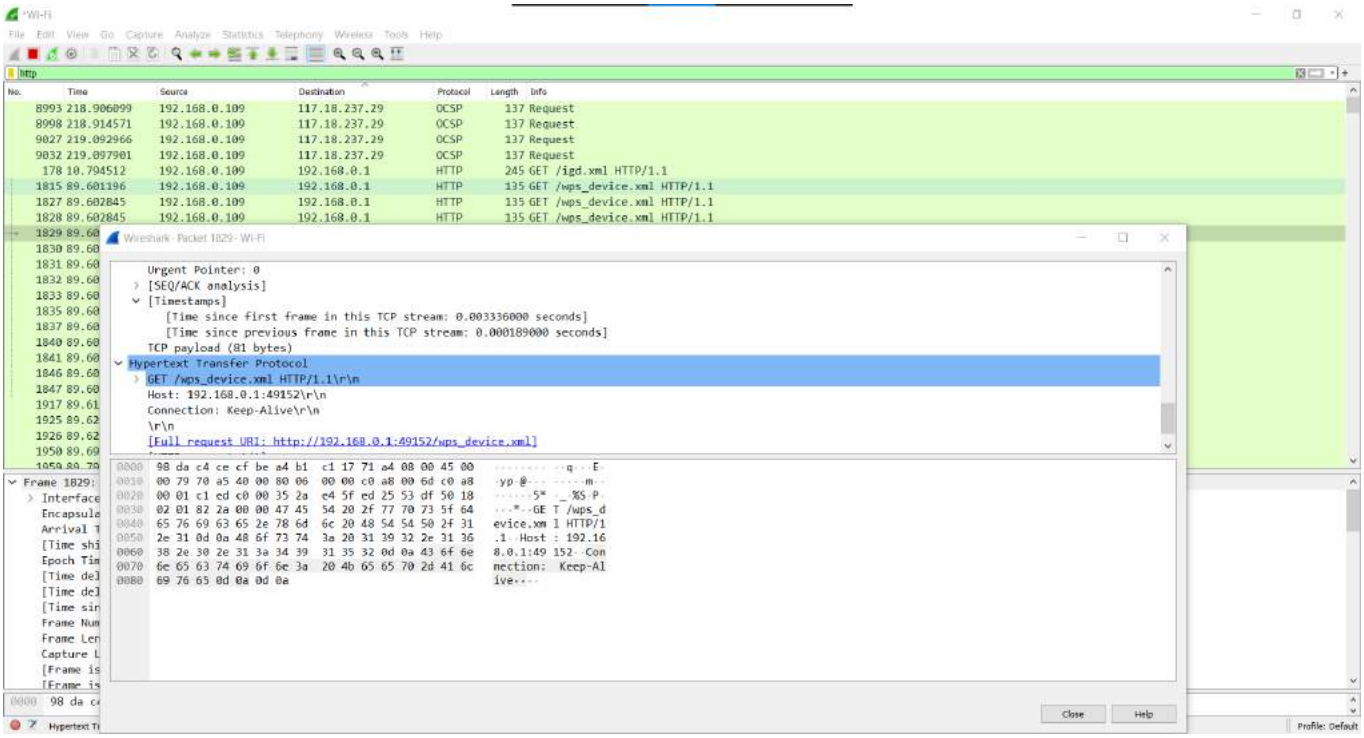
TCP Header:



DHCP:



HTTP:



Conclusion: We have successfully use Wireshark to understand the operation of TCP/IP layers and use it to get different header formats of packets.

Experiment No. 8

Aim: To implement chat application using Socket programming.

Requirement: Windows/MAC/Linux OS in P.C and JAVA/Python programming Language in OS.

Theory:

A socket is one endpoint of a two way communication link between two programs running on the network. The socket mechanism provides a means of inter-process communication (IPC) by establishing named contact points between which the communication take place.

Like 'Pipe' is used to create pipes and sockets is created using 'socket' system call. The socket provides bidirectional FIFO Communication facility over the network. A socket connecting to the network is created at each end of the communication. Each socket has a specific address. This address is composed of an IP address and a port number.

Socket are generally employed in client server applications. The server creates a socket, attaches it to a network port addresses then waits for the client to contact it. The client creates a socket and then attempts to connect to the server socket. When the connection is established, transfer of data takes place.

Types of Sockets:

There are two types of Sockets: the datagram socket and the stream socket.

Datagram Socket:

This is a type of network which has connection less point for sending and receiving packets. It is similar to mailbox. The letters (data) posted into the box are collected and delivered (transmitted) to a letterbox (receiving socket).

Stream Socket:

In Computer operating system, a stream socket is type of interprocess communications socket or network socket which provides a connection-oriented, sequenced, and unique flow of data without record boundaries with well-defined mechanisms for creating and destroying connections and for detecting errors. It is similar to phone. A connection is established between the phones (two ends) and a conversation (transfer of data) takes place.

Function Call	Description
create()	To create a socket
bind()	It's a socket identification like a telephone number to contact
listen()	Ready to receive a connection
connect()	Ready to act as a sender
accept()	Confirmation, it is like accepting to receive a call from a sender
write()	To send data
read()	To receive data
close()	To close a connection

Chat Application Code:

Server Side:

```
package chatapp;
import java.util.*;
import java.io.*;
import java.net.*;
public class Server{
   static HashMap<Integer,ClientHandler> acc = new HashMap<Integer,ClientHandler>();
   static int count = 0;
   public static void main(String args[]) throws IOException {
      ServerSocket ss = new ServerSocket(200);
      while(true){
         Socket s = ss.accept();
         ClientHandler ct = new ClientHandler("client "+count, new
DataInputStream(s.getInputStream()), new DataOutputStream(s.getOutputStream()),s);
         Thread t = new Thread(ct);
         acc.put(count, ct);
         count++;
         t.start();
      }
   }
}
class ClientHandler implements Runnable{
   private final String name;
   final DataInputStream dis;
   final DataOutputStream dos;
   public final Socket s;
   boolean loggedin;
   public ClientHandler(String name,DataInputStream dis,DataOutputStream dos,Socket s){
      this.name = name;
      this.dis = dis;
      this.dos = dos;
      this.s = s;
      this.loggedin = true;
   }
   @Override
   public void run(){
      while(true){
         String receive;
         try{
            receive = this.dis.readUTF();
            if(receive.equals("logout")){
               this.loggedin = false;
               this.s.close();
               break;
            }
            if(receive.contains("#")){
               StringTokenizer st = new StringTokenizer(receive,"#");
               String msg = st.nextToken();
```

```
                String name = st.nextToken();
                ClientHandler cc =
Server.acc.get(Character.getNumericValue(name.charAt(name.length()-1)));
                if(cc == null || !cc.loggedin){
                    this.dos.writeUTF("Client not found");
                }
                else{
                    cc.dos.writeUTF(this.name+": "+msg);
                }
            }
            else{
                this.dos.writeUTF("Wrong Input");
            }
        }catch (IOException e){
            e.printStackTrace();
        }
    }
  }
}
```

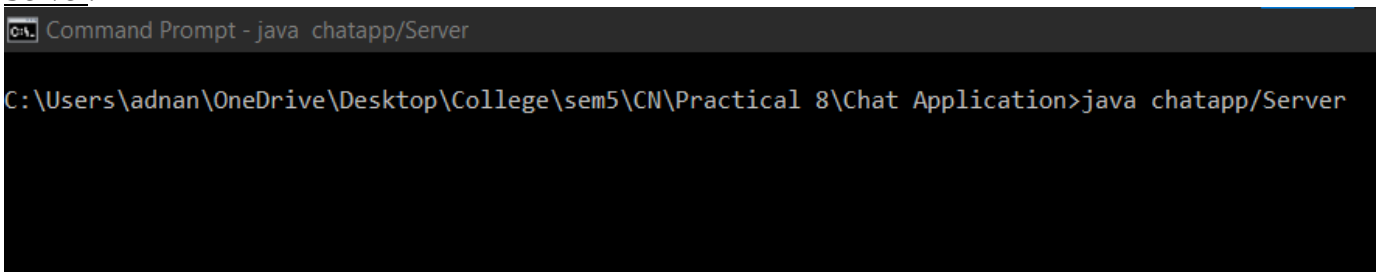
Client Side:

```
package chatapp;
import java.util.*;
import java.io.*;
import java.net.*;
public class Client {
   public static void main(String[] args) throws IOException{
      InetAddress ip = InetAddress.getByName("localhost");
      Socket s= new Socket(ip,200);
      Thread sm = new Thread(new SendMessage(s, new
DataOutputStream(s.getOutputStream())));
      Thread rm = new Thread(new ReadMessage(s, new
DataInputStream(s.getInputStream())));
      sm.start();
      rm.start();
   }
}
class SendMessage implements Runnable{
   Scanner sc = new Scanner(System.in);
   final private Socket s;
   DataOutputStream dos;
   public SendMessage(Socket s, DataOutputStream dos){
      this.s = s;
      this.dos = dos;
   }
   @Override
   public void run(){
      while(true){
         try{
            this.dos.writeUTF(sc.nextLine());
```

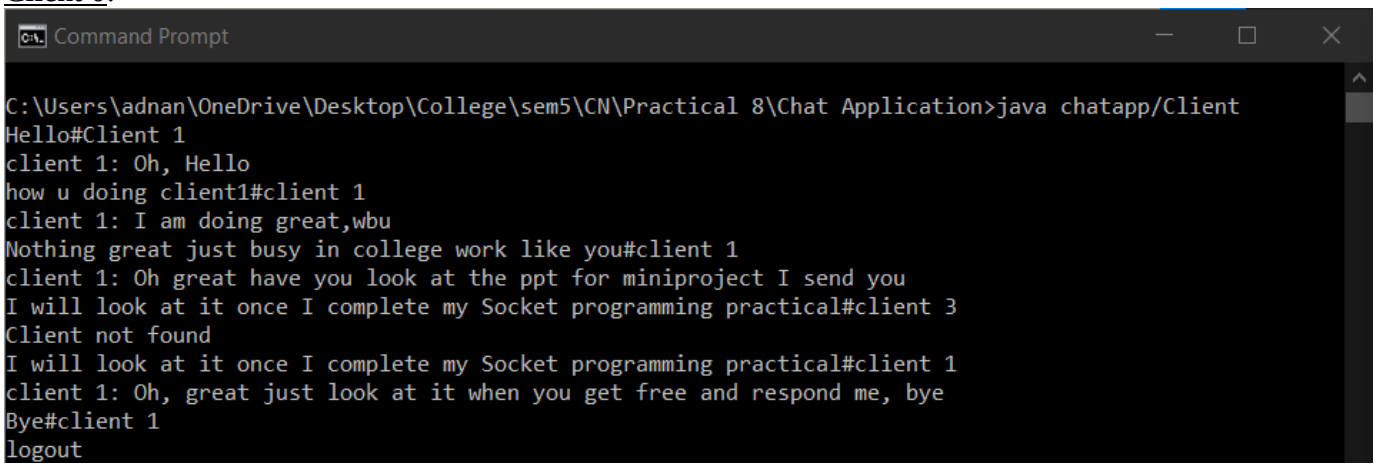
```
            }catch(IOException e){
                e.printStackTrace();
            }
        }
    }
}
class ReadMessage implements Runnable{
    final private Socket s;
    DataInputStream dis;
    public ReadMessage(Socket s, DataInputStream dis){
        this.s = s;
        this.dis = dis;
    }
    @Override
    public void run(){
        while(true){
            try{
                System.out.println(this.dis.readUTF());
            }catch(IOException e){
                e.printStackTrace();
            }
        }
    }
}
```

Output:

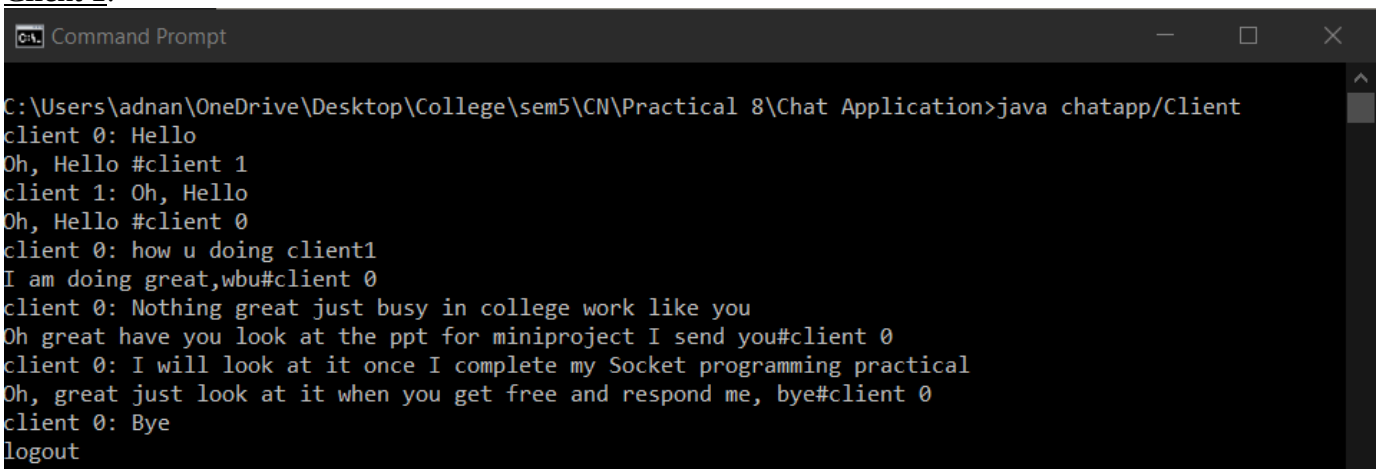
Server:



Client 0:



Client 1:



Conclusion: We have successfully implemented Chat Application with the help of Socket Programming using TCP or UDP in JAVA.

EXPERIMENT NO. 9

Aim: Study and Installation of Network Simulator (NS3).

Requirements:

Software	**Package/version**
Linux O.S	Any new variant distro (Arch, Debian, Fedro, etc.)
C++ compiler	clang++ or g++ (g++ version 4.9 or greater)
Python Git	python2 version >= 2.7.10, or python3 version >=3.4 any recent version
tar	any recent version
bunzip2	any recent version

Theory:

Introduction to NS3:

Network simulator is a tool used for simulating the real world network on one computer by writing scripts in C++ or Python. Normally if we want to perform experiments, to see how our network works using various parameters. We don't have required number of computers and routers for making different topologies. Even if we have these resources it is very expensive to build such a network for experiment purposes.

So to overcome these drawbacks we used NS3, which is a discrete event network simulator for Internet. NS3 helps to create various virtual nodes (i.e., computers in real life) and with the help of various Helper classes it allows us to install devices, internet stacks, application, etc. to our nodes.

Using NS3 we can create Point-to-point, Wireless, CSMA, etc. connections between nodes. Point-to-point connection is same as a LAN connected between two computers. Wireless connection is same as Wi-Fi connection between various computers and routers. CSMA connection is same as bus topology between computers. After building connections we try to install NIC to every node to enable network connectivity.

When network cards are enabled in the devices, we add different parameters in the channels (i.e., real world path used to send data) which are data-rate, packet size, etc. Now we use Application to generate traffic and send the packets using these applications.

NS3 gives us special features which can be used for real life integrations. Some of these features are:

Tracing of the nodes:

NS3 allows us to trace the routes of the nodes which helps us to know how much data is send or received. Trace files are generated to monitor these activities.

NetAnim:

It stands for Network Animator. It is an animated version of how network will look in real and how data will be transferred from one node to other.

Pcap file:

NS3 helps to generate pcap file which can be used to get all information of the packets (e.g., Sequence number, Source IP, destination IP, etc.). These pcaps can be seen using a software tool known as Wireshark.

Gnu Plot:

Gnu Plot is used to plot graphs from the data which we get from trace file of NS3. Gnu plot gives more accurate graph compare to other graph making tools and also it is less complex than other tools.

Advantages:

- The system has been modularized
- To allow for modular libraries
- Individual modules contains with directory structure
- To allow the node to use external routing

Disadvantages:

- Ns3 suffers from lack of credibility
- Modules, component based on ns2
- Ns3 needs lot of maintainers
- Active maintainers are required

Introduction to NS2:

NS-2 can be used to implement network protocols such as TCP and UPD, traffic source behaviour such as FTP, Telnet, Web, CBR and VBR, router queue management mechanism such as Drop Tail, RED and CBQ, routing algorithms and many more. In ns2, C++ is used for detailed protocol implementation and Otcl is used for the setup. The compiled C++ objects are made available to the Otcl interpreter and in this way, the ready-made C++ objects can be controlled from the Otcl level.

Downloading NS3:

This option is for the new user who wishes to download and experiment with the most recently released and packaged version of NS3. NS3 publishes its releases as compressed source archives, sometimes referred to as a tarball. A tarball is a particular format of software archive where multiple files are bundled together and the archive is usually compressed. The process for downloading NS3 via tarball is simple; you just have to pick a release, download it and uncompress it.

Let’s assume that you, as a user, wish to build NS3 in a local directory called workspace. If you adopt the workspace directory approach, you can get a copy of a release by typing the following into your Linux shell (substitute the appropriate version numbers, of course)

```
$ cd
$ mkdir workspace
$ cd workspace
$ wget https://www.nsnam.org/release/ns-allinone-3.29.tar.bz2
$ tar xjf ns-allinone-3.29.tar.bz2
```

Notice the use above of the wget utility, which is a command-line tool to fetch objects from the web; if you do not have this installed, you can use a browser for this step.

Following these steps, if you change into the directory ns-allinone-3.29, you should see a number of files and directories

```
$ cd ns-allinone-3.29
$ ls
bake      constants.py   NS3.29                          README
build.py  netanim-3.108  pybindgen-0.17.0.post58+ngcf00cc0  util.py
```

You are now ready to build the base NS3 distribution.

Building NS3:

When working from a released tarball, a convenience script available as part of NS3-allinone can orchestrate a simple build of components. This program is called build.py. This program will get the project configured for you in the most commonly useful way. However, please note that more advanced configuration and work with NS3 will typically involve using the native NS3 build system, Waf, to be introduced later in this tutorial.

If you downloaded using a tarball you should have a directory called something like ns-allinone-3.29 under your ~/workspace directory. Type the following:

```
$ ./build.py --enable-examples --enable-tests
```

Because we are working with examples and tests in this tutorial, and because they are not built by default in NS3, the arguments for build.py tells it to build them for us. The program also defaults to building all available modules. Later, you can build NS3 without examples and tests, or eliminate the modules that are not necessary for your work, if you wish.

You will see lots of compiler output messages displayed as the build script builds the various pieces you downloaded. First, the script will attempt to build the netanim animator, then the pybindgen bindings generator, and finally NS3. Eventually you should see the following:

```
Waf: Leaving directory '/path/to/workspace/ns-allinone-3.29/NS3.29/build'
'build' finished successfully (6m25.032s)


Modules built:
antenna                aodv                   applications
```

```
bridge                buildings              config-store
core                  csma                   csma-layout
dsdv                  dsr                    energy
fd-net-device         flow-monitor           internet
internet-apps         lr-wpan                lte
mesh                  mobility               mpi
netanim (no Python)   network                nix-vector-routing
olsr                  point-to-point         point-to-point-layout
propagation           sixlowpan              spectrum
stats                 tap-bridge             test (no Python)
topology-read         traffic-control        uan
virtual-net-device    visualizer             wave
wifi                  wimax

Modules not built (see NS3 tutorial for explanation):
brite                 click                  openflow

Leaving directory ./NS3.29
```

Conclusion: We have successfully learnt the concept of NS3 and installed it in our Linux O.S using source archive release of NS3.

Experiment No. 10

Aim: To Perform File Transfer and Access using FTP using Packet tracer.

Requirements: Windows O.S and Cisco Packet Tracer.

Theory:

What is FTP (File Transfer Protocol)?

FTP (File Transfer Protocol) is a network protocol for transmitting files between computers over Transmission Control Protocol/Internet Protocol (TCP/IP) connections. Within the TCP/IP suite, FTP is considered an application layer protocol. In an FTP transaction, the end user's computer is typically called the local host. The second computer involved in FTP is a remote host, which is usually a server. Both computers need to be connected via a network and configured properly to transfer files via FTP. Servers must be set up to run FTP services, and the client must have FTP software installed to access these services.

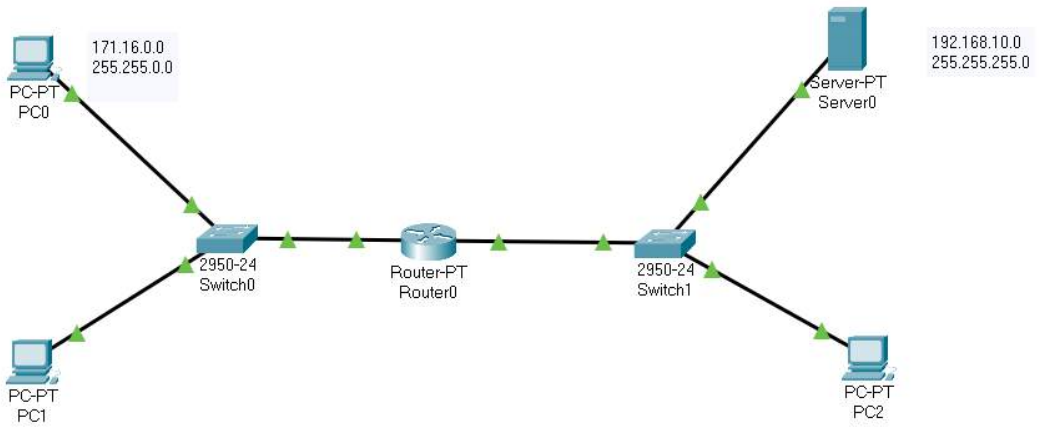
How does FTP work?

FTP is a client-server protocol that relies on two communications channels between the client and server: a command channel for controlling the conversation and a data channel for transmitting file content.

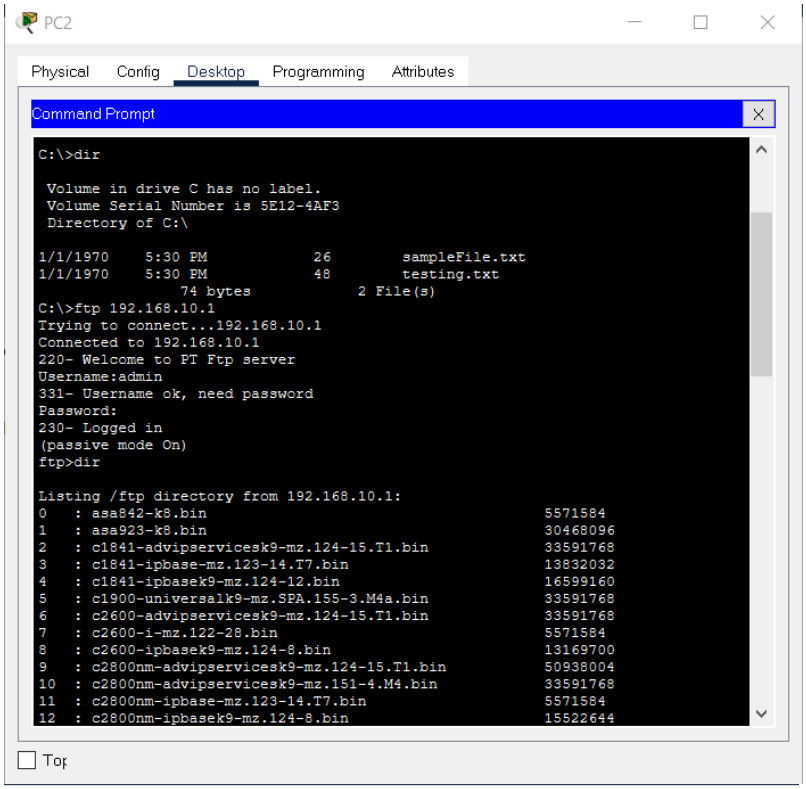
Here is how a typical FTP transfer works:

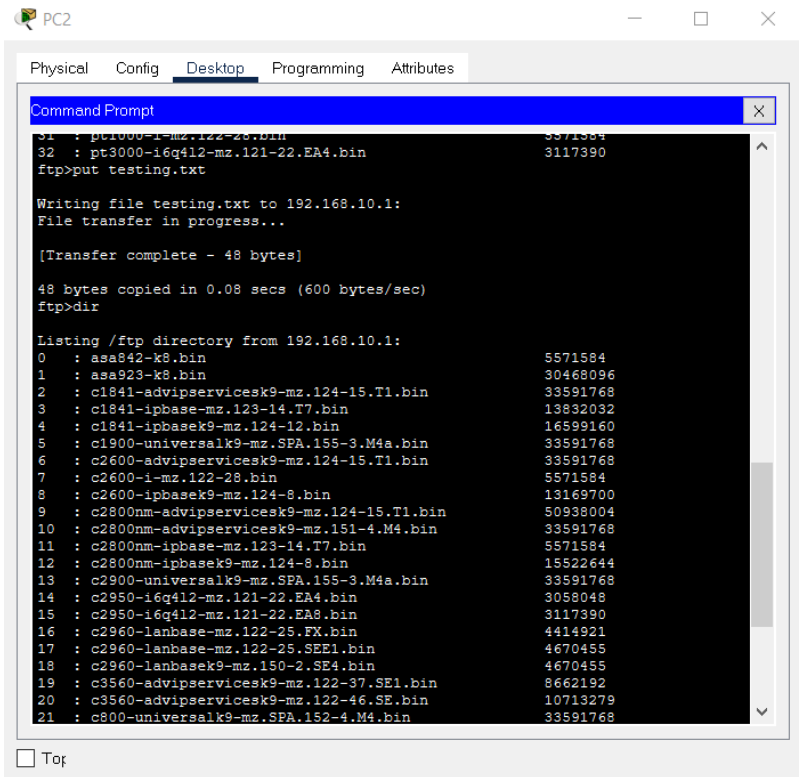
1. A user typically needs to log on to the FTP server, although some servers make some or all of their content available without a login, a model known as anonymous FTP.
2. The client initiates a conversation with the server when the user requests to download a file.
3. Using FTP, a client can upload, download, delete and rename, move and copy files on a server.

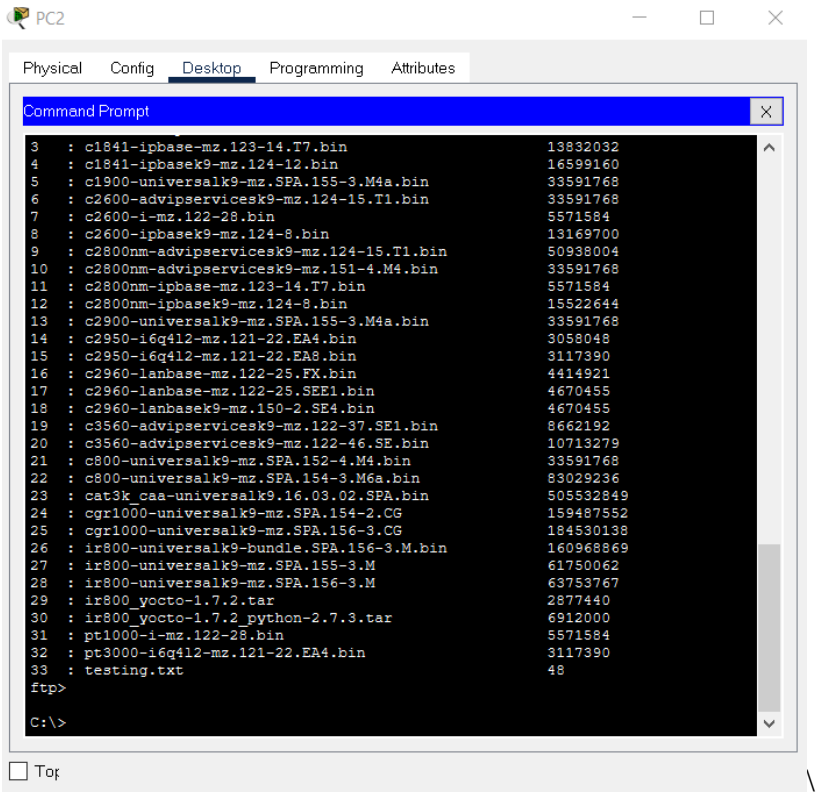
Output:



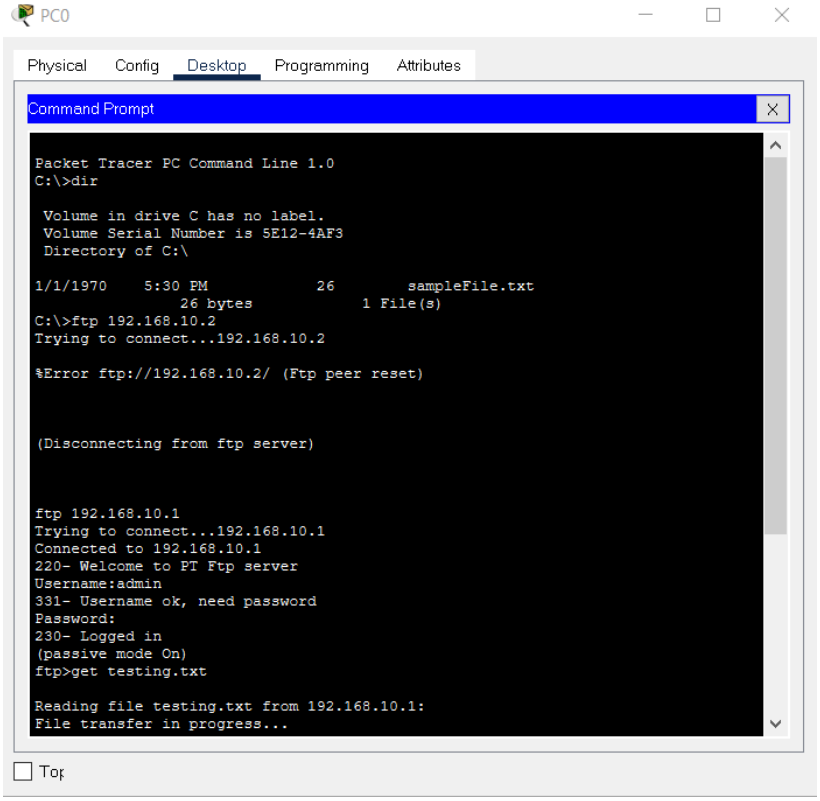
Uploading File (testing.txt) to the server from PC2 using FTP:

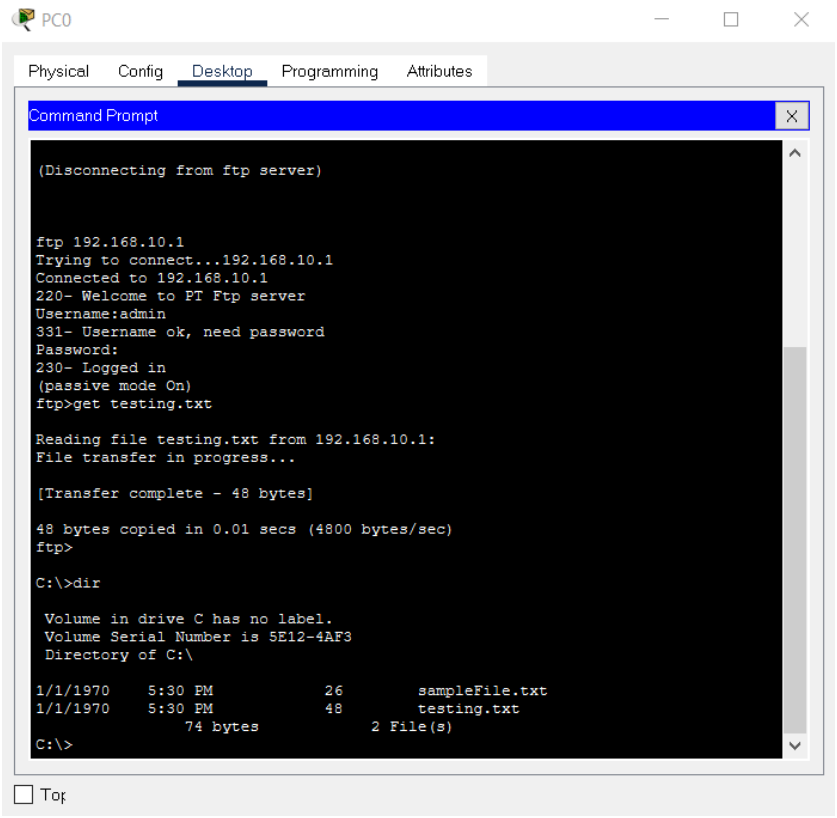




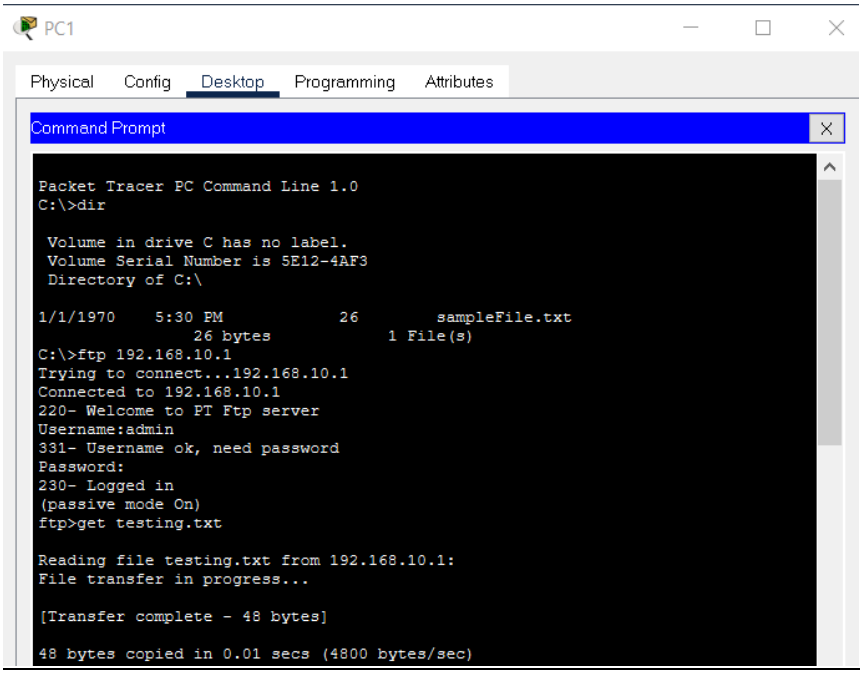


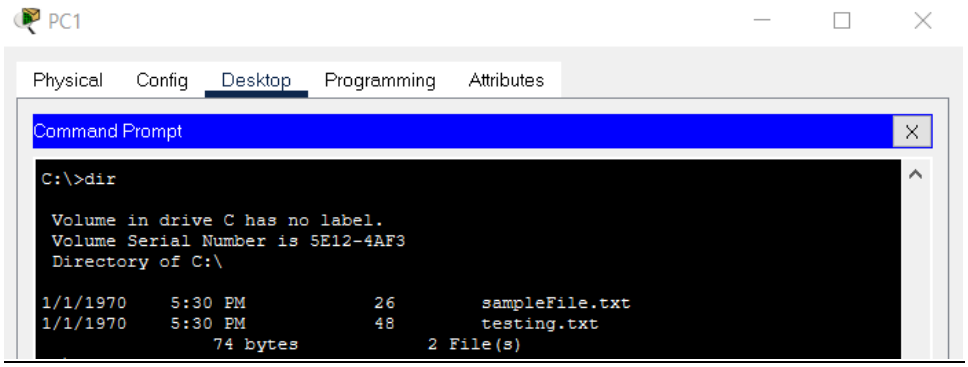
Downloading testing.txt from server to PC0 using FTP:





Downloading testing.txt from server to PC1 using FTP:





Conclusion: We have successfully access and transfer file from and to the server using FTP command in CISCO Packet tracer.

Experiment No. 11

Aim: To implement leaky bucket algorithm using programming language.

Requirement: Windows/MAC/Linux OS in P.C and JAVA/Python programming Language in OS.

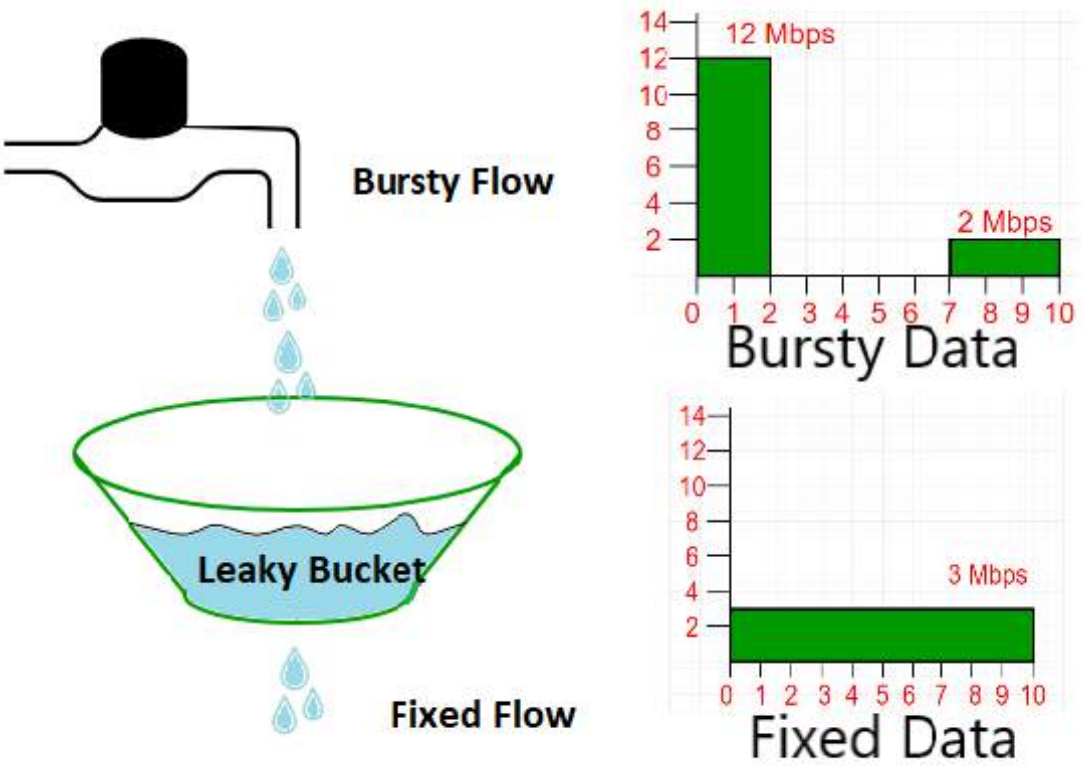
Theory:

The leaky bucket algorithm is a method of temporarily storing a variable number of requests and organizing them into a set-rate output of packets in an asynchronous transfer mode (ATM) network.

The leaky bucket is used to implement traffic policing and traffic shaping in Ethernet and cellular data networks. The algorithm can also be used to control metered-bandwidth Internet connections to prevent going over the allotted bandwidth for a month, thereby avoiding extra charges.

The algorithm works similarly to the way an actual leaky bucket holds water: The leaky bucket takes data and collects it up to a maximum capacity. Data in the bucket is only released from the bucket at a set rate and size of packet. When the bucket runs out of data, the leaking stops. If incoming data would overfill the bucket, then the packet is considered to be non-conformant and is not added to the bucket. Data is added to the bucket as space becomes available for conforming packets.

The leaky bucket algorithm can also detect both gradually increasing and dramatic memory error increases by comparing how the average and peak data rates exceed set acceptable background amounts.



A simple leaky bucket algorithm can be implemented using FIFO queue. A FIFO queue holds the packets. If the traffic consists of fixed-size packets (e.g., cells in ATM networks), the process removes a fixed number of packets from the queue at each tick of the clock. If the traffic consists of variable-length packets, the fixed output rate must be based on the number of bytes or bits.

The following is an algorithm for variable-length packets:

1. Initialize a counter to n at the tick of the clock.
2. If n is greater than the size of the packet, send the packet and decrement the counter by the packet size. Repeat this step until n is smaller than the packet size.
3. Reset the counter and go to step 1.

Code:

```
import java.io.*;
import java.util.*;
class LB{
   public static void main (String[] args) {
      int no_of_queries,storage,output_pkt_size;
      int input_pkt_size,bucket_size,size_left;
      storage=0;
      no_of_queries=4;
      bucket_size=10;
      input_pkt_size=4;
      output_pkt_size=1;
      for(int i=0;i<no_of_queries;i++)
      {
         size_left=bucket_size-storage;
         if(input_pkt_size<=(size_left))
         {
            storage+=input_pkt_size;
            System.out.println("Buffer size= "+storage+
               " out of bucket size= "+bucket_size);
         }
         else
         {
```

```
            System.out.println("Packet loss = " +(input_pkt_size-(size_left)));
            storage=bucket_size;
            System.out.println("Buffer size= "+storage+" out of bucket size= "+bucket_size);
        }
        storage-=output_pkt_size;
    }
  }
}
```

Output:

```
C:\Users\adnan\OneDrive\Desktop\College\sem5\CN\Practical 11>java LB
Buffer size= 4 out of bucket size= 10
Buffer size= 7 out of bucket size= 10
Buffer size= 10 out of bucket size= 10
Packet loss = 3
Buffer size= 10 out of bucket size= 10
```

Conclusion: We have understand the concept of Leaky Bucket algorithm and successfully implemented it in Java Programming Language.

Subject: Computer Network Sem: V

Assignment 1

Q.1) What are the design issues of layers in OSI reference model of networking?

Ans) The following are the design issues for the layers in OSI reference model of networking:-

1. Reliability: It is a design issue of making a network that operates correctly even when it is made up of unreliable components. e.g:- Routing, error correction and detection.

2. Addressing:- There are multiple processes running on one machine. Every layer needs a mechanism to identify senders and receivers.

3. Error Control:- It is an important issue because physical communication circuits are not perfect. Many error detecting and correcting codes are available. Both send and receiving ends must agree to use any one code.

4. Flow control: If a rate at which data is produced by the sender is higher than rate at which data is received by the receiver, there are chances of swamping the receiver. So, a proper flow control mechanism needs to be implemented.

5.) Scalability:- Network sizes are continually increasing leading to congestion. Also, when new technologies are applied to the added components, it may lead to incompatibility issues. Hence, the design should be done so that the networks are scalable and can accomodate such additions and alterations.

6.) Resource Allocation:- Computer networks provide services in the form of network resources to the end users. The main design issue is to allocate and de-allocate resources to processes, optimal usage of the resources.

7.) Statistical Multiplexing:- It is not feasible to allocate a dedicated path for each message while it is being transferred from the source to the destination. So, the data channel needs to be multiplexed, so as to allocate a fraction of bandwidth or time to each host.

8.) Routing:- There may be multiple paths from the source to destination. Routing involves choosing an optimal path among all possible paths, in terms of cost and time. There are several routing algorithms that are used in network systems.

Q.1) Security (Confidentiality & Integrity) :- A major factor of data communication is to defend it against threats like eavesdropping and surreptitious alteration of messages. So, there should be adequate mechanisms to prevent unauthorized access to data through authentication and Cryptography. Mechanisms that provide confidentiality defend against threats like eaves dropping. Mechanisms for integrity prevent faulty changes to messages.

Q.2) Write a short note on Twisted Pair, Coaxial Cable.

Ans (I) Twisted-Pair Cable:-

(i) One of the earliest guided transmission media is twisted pair cables. A twisted pair Cable comprises of two seperate insulated copper wires, which are twisted together and run in parallel.

(ii) The copper wires are typically 1mm in diameter

(iii) One of the wires is used to transmit data & the other is the ground reference.

(iv) Reason for Twisting:- (i) All transmissions are prone to noise, interferences, and crosstalks.

(ii) When the wires are twisted, some part of the noise signals is in the direction of data signals while the other parts are in the opposite directions. Thus the external waves cancel out due to different twists.

(iii) The receiver calculates the difference in the

Voltages of two wire for retrieving data. Thus, a much better immunity against noise is obtained.

(5) Application:- (i) In LANs (ii) In OSL lines (iii) In telephone lines.

(6) Types:- (i) Unshielded Twisted Pair (UTP):- These generally comprise of wires & insulators. (ii) Shielded Twisted Pair (STP):- They have a braided wired mesh that encases each pair of insulated wires.

(7) Categories:- (i) CAT-1 :- UTP, B.W < 0.1 Mbps
(ii) CAT-2 :- UTP, B.W ≤ 2 Mbps
(iii) CAT-3 :- UTP, B.W ≤ 10 Mbps
(iv) CAT-4 :- UTP, B.W ≤ 20 Mbps
(v) CAT-5 :- UTP, B.W ≤ 10^2 Mbps
(vi) CAT-6 :- UTP, B.W ≤ 2×10^2 Mbps.

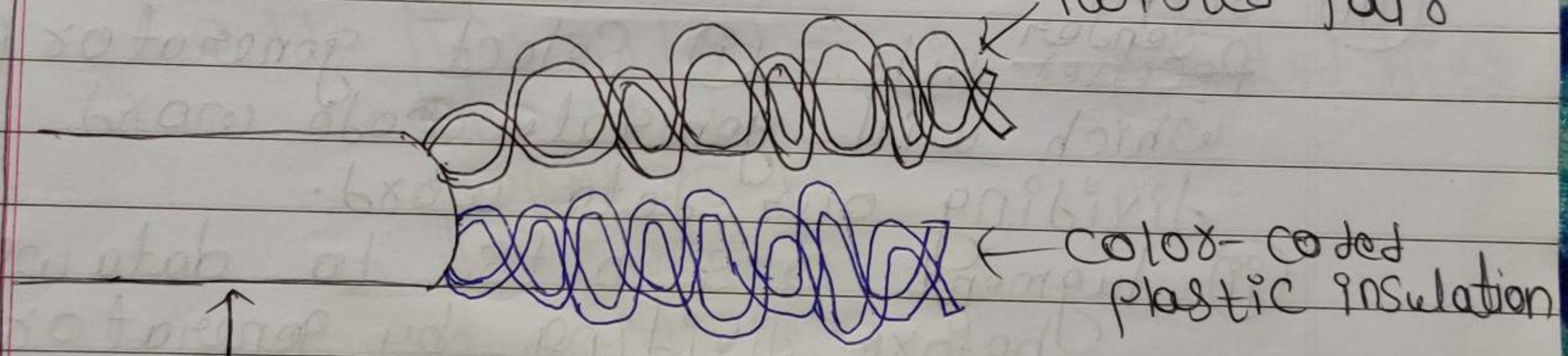
(II) Coaxial Cable:-
(1) Coaxial cables, commonly called Coax, are copper cables with metal shielding designed to provide immunity against noise and greatet bandwidth.
(2) Coax transmit signals over larger distances at a higher speed as compared to twisted pair cables.
(3) Structure:- (i) Coax has a central core of stiff copper conductor for transmitting signals. This is covered by an insulating material.
(ii) The insulator is encased by a closely woven

braided metal outer conductor that acts as a shield against noise.

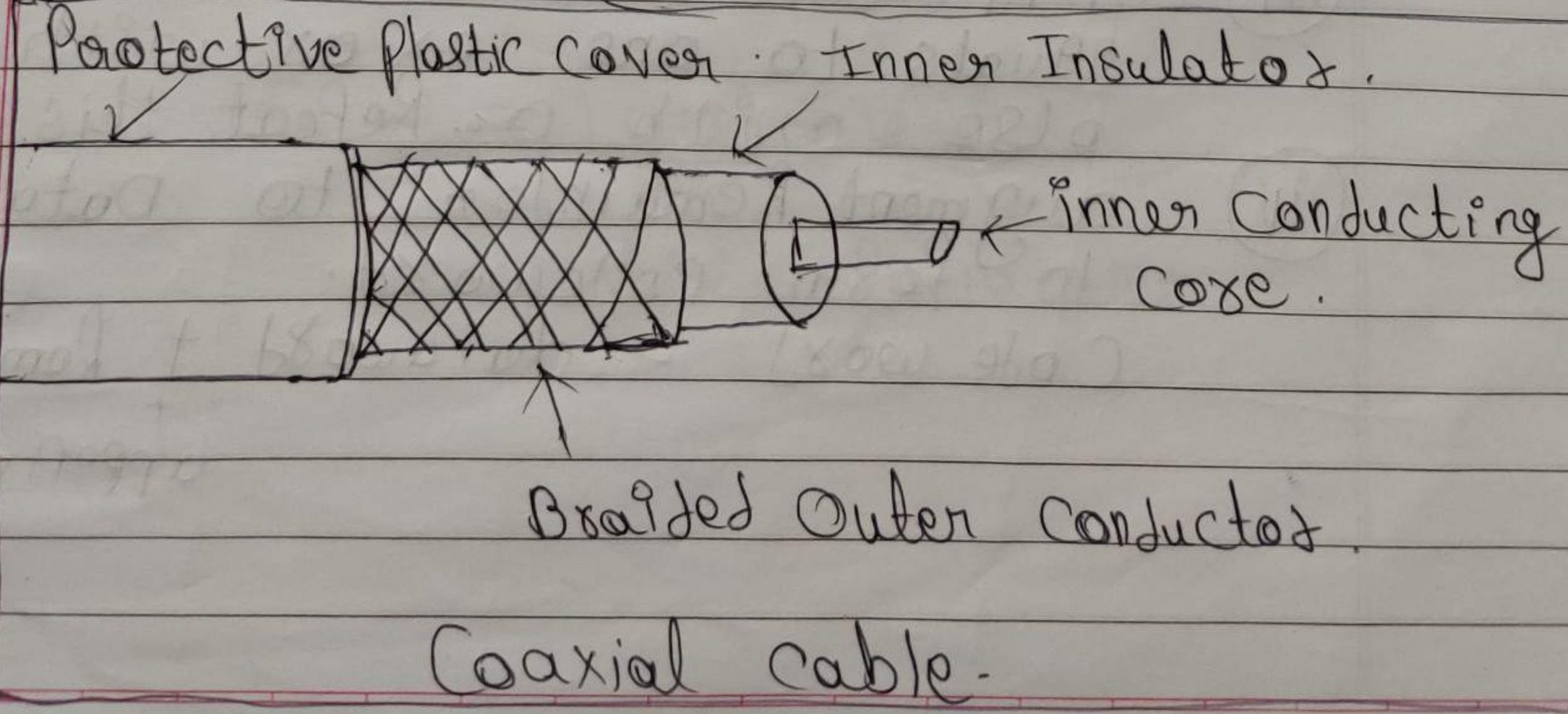
(iii) The outer conductor is again enclosed by a plastic insulating cover.

(4) Categories :- Coaxial cables are categorized into three types as per radio government (RG) ratings:- (i) RG-59: Z = 75 W, Used = Cable TV
(ii) R.G-58: Z = 50W, Used = Ethernet.
(iii) R.G-11: Z = 50W, Used = thick Ethernet.

(5) Applications:- (i) In analog telephone networks.
(ii) In digital telephone networks (< 600 Mbps).
(iii) Cable TV networks.
(iv) Ethernet LANs.
(v) In MANs.



Unshielded Twisted-pair.



Coaxial cable.

Q3) Explain different error detection techniques with example.

Ans) (I) Cyclic Redundancy Check (CRC):- 7

(i) Cyclic codes are special linear block codes with one extra property. In a cyclic code, if a code word is cyclically shifted (rotated), the result is another code word.

(ii) A cyclic redundancy check (CRC) is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data.

(iii) Steps:-
~~Receiver~~ Sender side:- (i) Select generator polynomial which will generate code word by dividing our data word.

(ii) Augment 0's bits to data word before dividing by generator
no. of zero's = no. of bits in generator - 1

(iii) A. If Left-most of dividend equals to one ex-or with divisor else with 0. Repeat this procedure.

(iv) augment Remainder to Data word to form codeword.
Code word = dataword + Remainder.
↑ append.

Receiver Side:- (i) Divide the code-word received from generator using Step-3 from Sender Side.

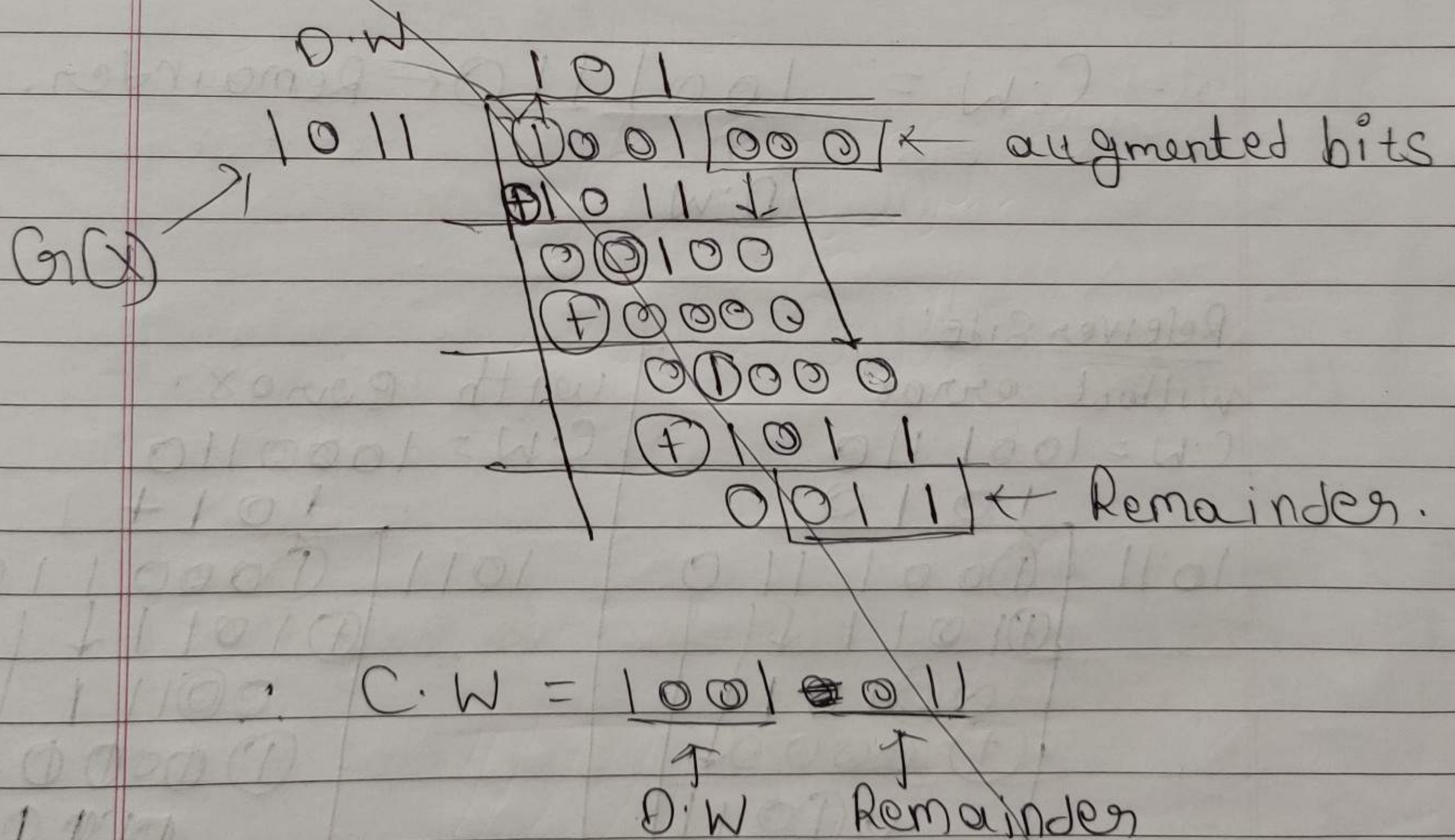
(ii) If Remainder or Syndrome is 0 then Code-word is received in correct remove no. of bits in generator - 1 from C.W to form data word.

(iii) If Remainder is not zero received C.W is incorrect ask for retransmission.

e.g:-

D.W = 1001 ; Generator = 1011

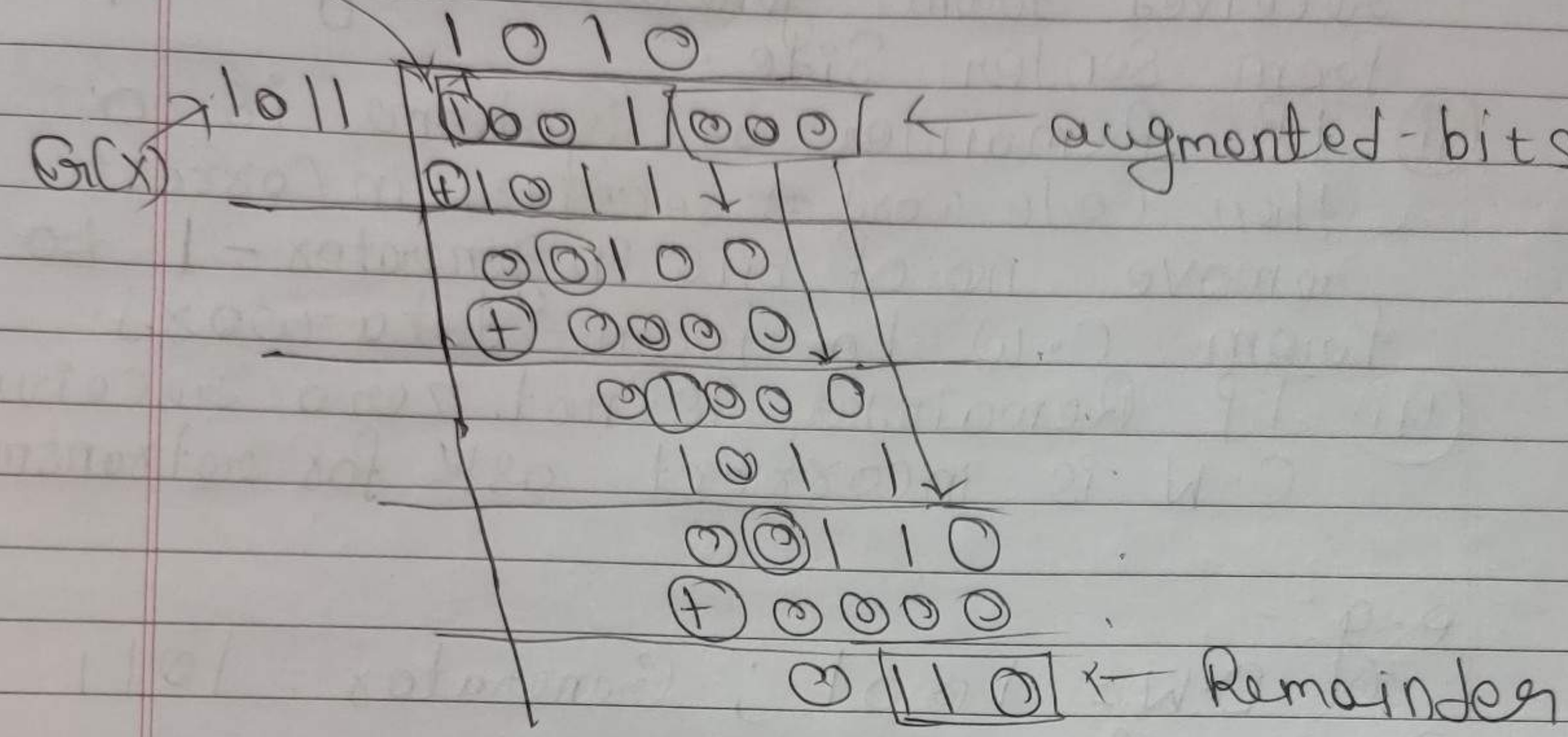
C.W :- ?



∴ C.W = 1001011

D.W Remainder

e.g.:- D.W = 1001 ; Generator = 1011



∴ C.W = 1001 | 110 ← Remainder.
↑
D.W

Receiver Site:-

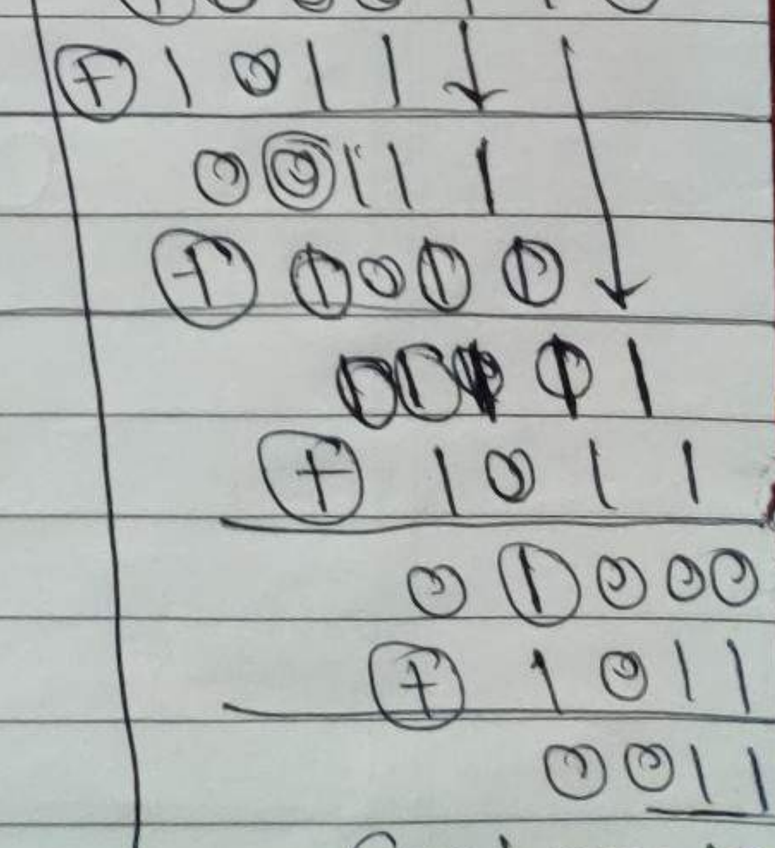
without error

C.W = 1001 | 110

∴ D.W = 1001

with error.

C.W = 1000 | 110



ASK for retransmission.

(II) **Checksum:-**

Steps:-

(1) At the source, the message is first divided into m-bits units.
(2) The generator then creates the checksum i.e an extra m-bits unit.
(3) At destination, the checker creates a new checksum from the combination of the message & sent checksum.
(4) If check sum is 0 then message is accepted else message is discarded.

e.g:-

Sender:-

7
11
12
0
6
+ [0]
36

← extra bit will be replaced with Check-sum bits.

max-bits = 4

1 0 [0 1 0 0]
+ → 1 0
6 = 0 1 1 0 —1's→ 1 0 0 1 = 9

∴ Packet = 7, 11, 12, 0, 6, [9]

Receiver:-

7
11
12 max-bits = 4
0
6
9
+
10 1 1 0 1 ← 45
↳ 1 0
15 = 1 1 1 1 —1's comp→ 0 0 0 0 = 0

∴ Check-sum = 0
Received packet is correct.

∴ Required-info = 7, 11, 12, 0, 6

Q.3) ① Hamming code is used for error detection as well as error correction in for data frame.

② It does so my adding 'r' redudant bit in actual frame.

③ Let, us see how it encode by example:-

④ Consider, we have dataframe (dw) as 1001011

⑤ Total ~~ten the~~ length $l(dw) = m = 7$

⑥ we add 'r' redundant bit by following formula :- $m <= 2^r - r - 1$

we can observe that $r = 4$ is suited for above formula

$m <= 2^4 - 4 - 1 <= 11$

⑦ We insert r bits at the position of power of i.e $2^0, 2^1, 2^2, 2^3$

$P_1 \quad P_2 \quad P_4 \quad P_8$

⑧ Total bit of encoded word is $n = m + r$

i.e $n = 7 + 4 = 11$

⑨ encoding :-

D_{11}	D_{10}	D_9	P_8	D_7	D_6	D_5	P_4	D_3	P_2	P_1
1	0	0	1	1	0	1	1	1	1	0

Table-a

we insert Dw bits at D position sequenti-ally & encode value of P using even parity by selecting 2^n alterate bits.

$P_1 = P_1\ D_3\ D_5\ D_7\ D_9\ D_{11} = 0$

$P_1\ 1\ 1\ 1\ 0\ 1\ 1$

∴ even parity $P_1 = 0$

$P_2 = P_2\ D_3\ D_6\ D_7\ D_{10}\ D_{11} = 1$

$P_2\ 1\ 0\ 1\ 0\ 1\ 0$

∴ odd parity ∴ $P_2 = 1$

$P_4 = P_4\ D_5\ D_6\ D_7 = 0$

$P_4\ 1\ 0\ 1$

∴ Even parity ∴ $P_4 = 0$

$P_8 = P_8\ D_9\ D_{10}\ D_{11} = 1$

$P_8\ 0\ 0\ 1$

∴ odd parity ∴ $P_8 = 1$

∴ C.W = 0110

∴ C.W = 1p0110101110

This will be send to receiver

(10) Receiver side Decoding: Receiver used reversed engineering to form dw from the code and check if any error is detected & if detected it will correct it.

(11) Receiver will decode C.W in following way (referencing table a):-

$P_1 = P_1\ D_3\ D_5\ D_7\ D_9\ D_{11} = 0$ ✓

$1\ 1\ 1\ 0\ 1$

$P_2 = P_2\ D_3\ D_6\ D_7\ D_{10}\ D_{11} = 1$ ✓

$P_2\ 1\ 0\ 1\ 0\ 1\ 1$

$P_4 = P_4\ D_5\ D_6\ D_7 = 0$ ✓

$P_4\ 1\ 0\ 1\ 1$

$P_8 = P_8\ D_9\ D_{10}\ D_{11} = 1$ ✓

$P_8\ 0\ 0\ 1$

we see all the bits are correct now we remove parity bits & form original d·w = 1001101

(12) If any bit was inverted we have gotten wrong parity for it.

(13) For detecting wrong parity we select common bits from parity encoding i.e; for P_1 o P_2 o $P_4 = D_7$ & so, on.

(14) So, we will invert the common bit, & we will have our right code.

(15) we can also add parity bits to check where the error is i.e $1+2+4 = 7^{th}$ bit.

C.N Assignment no. 2 68 Adnan

Q.1) What is routing? Explain different routing protocols - Distance vector and Linked State.

Ans) Network routing is the process of selecting a path across one or more network. Routers refer to internal routing table to make decision about how to route packets along network paths. A routing table records the path that packets should take to reach every destination that router is responsible for.

a) Static Routing: It is a technique in which the administrator manually adds the route in routing table. Useful in long term Virtual circuit.

b) Dynamic Routing: It is a technique in which a router adds a new route in the routing table for each packet in response to changes in condition or topology of network.

c) Default Routing: It is technique in which a router is configured to send all packets to same hop devices.

- Distance Vector:

A distance vector routing algorithm operates by having each router maintain a table giving best known distance to each destination and which link to use to get there. These tables are updated by exchanging information with the neighbours.

Eventually every router knows the best link to reach destination.

In distance vector, each router maintains a routing table indexed by and containing one entry for each router in network. The entry has two parts: the preferred outgoing line to use for that destination and an estimate of the distance to the destination. The distance might be measured as the numbers of hops, delay in propagation Bandwidth or using any suitable metric.

The router is assumed to know the distance to each of its neighbours. If the metric is hops, the distance is just one hop. If the metric is delay in propagation, the router can measure it directly with special ECHO packets that the reciever just timestamps and send back as fast as it can.

- Link-State Routing:

It is one of the main classes of routing protocols used in packet switching network for computer communication.

The Basic concept of link-state routing is that every node constructs a map of the connectivity ~~of the connectivity~~ to the network, in form of graph showing which nodes are connected to which other nodes.

Each node then independently calculates the next best logical path from it to every possible destination in the network. Idea behind link state routing is fairly simple and can be stated as five parts

a) Discover its neighbour and learn their network address.
b) Set the distance or cost metric to each of its neighbours.
c) Construct the packet telling all it has just learned.
d) Send and recieve packet from all other routers
e) Compute the shortest path to every other router.

- Learning about neighbours: When the router is boot its first task is to learn who its neighbours are, It accomplish this goal by sending a special 'Hello' packet on each point-to-point. The router on other end is expected to send back a reply giving it name. These name must be globally unique

- Setting Link Costs: The Link state routing algorithm requires each link to have distance or cost metric for finding shortest path. The cost to reach neighbours can be set automatically or configured by the network operator. If the network is geographically spread out the delay of the links maybe

factored into cost so that paths over shorter link are better choices.

- Building link State Packet: Once the information needed for the exchange has been collected ne the next step is for each router to build a pa packet containing all the data. The packet starts with the identity of the sender t followed by a sequence number and age and list of neighbours. Cost of each neighbour is also given.

- Distributing link State Packet: All of the routers must get all the link state quickly and reliably. If different routers are using different version of topology the routes they compute can have inconsistencies such as loops, unreachable machine etc. Flooding algorithm can be use to distribute the link State packet to routers. Each packet contains a sequence number that is incremented for each new packet sent. Second, if a router ever crashes it will lose track of its sequence number. If it starts again at 0, the next packet it sends will be rejected as duplicate. Therefore, to tackle this problem age info is used the information router has expires if the age ~~is~~ time is exceeded & in this way it will accept the new packet even if its sequence number is 0 or has some defects.

- Computing new routes: Once a router has accumulated a full set of link state packets, it can construct entire network graph because every link is represented twice for each direction. Compared to distance vector routing link state routing requires more memory and computation. For a network with n routers each of which has k neighbours the memory required to store input data is proportional to kn which is atleast as large as routing table listing all the destinations.

Q.2) Explain TCP Header

Ans)

← 16 bits →									← 16 bits →
Source Port									Destination Port
Sequence Number									
Acknowledgment Number									
TCP Header Length (4-bits)	Reserved bits (6-bits)	U R G	A C K	P S H	R S T	S Y N	F I N	Window Size (Advertisement window)	
Options (0-32 bit words)									
Checksum									Urgent Pointer
Options (0-40 bytes)									
Data (Optional)									

TCP Header

1. Source Port: 16-bit field identifies the port of the sending application.

2. Destination Port: 16-bit field identifies the port of the receiving application.

3. Sequence Number: 32 bit field contains the sequence number of the first data byte.

4. Acknowledgment Number: 32 bit field always contain sequence number of the last received data byte incremented by 1

5. Header length: 4-bit field helps in knowing from where the actual data begins because of varying size of option field always consider in multiple of 4 and start from 5. Range [20 bytes, 60 bytes]

6. Reserved Bit: 6-bit field not used.

7. Urgent Bit: 1-bit field set to 1 if data is to be treated on urgent basis.

8. ~~Acknowledg~~
8. ACK bit: 1-bit field set to 1 if acknowledgement number is valid. For all Tcp segments except ~~se~~request segment, ACK bit is set to 1

9. PSH Bit: 1-bit field set to 1 if the entire buffer is ~~w~~ to pushed immediately to the receiving application.

10. RST Bit: 1-bit field used to reset TCP connection when set to 1. It causes both the sides to release the connection and all its resources abnormally.

11. SYN Bit: It is used to Synchronize the Sequence numbers.

12. FIN Bit: FIN bit is used to terminate the T-C-P connection.

13. Window Size: 16-bit ~~con~~field contains the size of the receiving window of the Sender. When congestion detected, the window size is reduced dynamically.

14. Checksum: 16-bit field used for error control. It verifies the integrity of data in TCP payload.

15. Urgent Pointer: 16 bitfield indicates how much data in the current segment from the first data byte is urgent. It is consider valid only if URG bit is set to 1.

16. Options: options field vary from 0 bytes to 40 bytes.
Use for following purposes:
1. Time Stamp.
2. window size extension.
3. Parameter negotiation.
4. Padding.

Q.3) Write short note on HTTP and SMTP.

Ans) HTTP: (i) Stands for Hyper text transfer protocol.

(ii) It can be used to carry data in form of MIME format.

(iii) It is a connectionless protocol. HTTP client ~~initia~~ initiates a request and waits a response from the server.

(iv) HTTP protocol is media independent as data can be sent as long as both the client and server know how to handle the data content.

(v) It is a Stateless protocol as both the client & server know each other during request.

SMTP: (i) It stands for simple mail transfer protocol.

(ii) It is a program used for sending messages to other computer based on e-mail address.

(iii) SMTP server is always on listening mode.

(iv) Client initiates a T.C.P connection with SMTP server.

(v) SMTP server listens for a connection and initiates a connection on that port.

(vi) Connection is estabilished and client informs SMTP that it would like to send an email.

(vii) Assuming server is ok, client sends the mail to its mail server and client map server use DNS to get IP address.

(viii) SMTP then transfer mail from sender to received mail server.