Geetanjali Institute of Technical Studies

(Approved by AICTE, New Delhi and Affiliated to Rajasthan Technical University Kota (Raj.))

DABOK, UDAIPUR, RAJASTHAN 313022

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

B. Tech - IV SEMESTER

तमसो मा ज्योतिर्गमय्

Darkness to Light

ACADEMIC YEAR – 2023-24

NETWORK PROGRAMMING LAB (4CCS4-23)

Submitted To:
Ms. Meenal Joshi

Submitted By:
Student Name:
Roll No:
Section:

VISION & MISSION OF INSTITUTE

Vision of Geetanjali institute of Technical studies

To achieve excellence in technical and management education through quality teaching and innovation.

Mission of Geetanjali institute of Technical studies

M1: To provide an excellent learning environment to produce socially responsible and productive technical professionals.

M2: To set up the state-of-the-art facilities for quality education and innovation.

M3: To impart knowledge & Skills leading to shaping a budding manager as a quality executive.

M4: To encourage for life-long learning and team-based problem solving through learning environment.

Department Vision

To nurture the students to become employable graduates who can provide solutions to the societal issues through ICT.

Department Mission

- To focus on practical approach towards learning and exposing the students on the latest ICT technologies.
- To foster logical thinking among the students to solve real-time problems using innovative approaches.
- To provide state-of-the-art resources that contributes to inculcate ethical and life-long learning environment.

Programme Educational Objectives (PEOs)

The Programme Educational Objectives of the programme offered by the department are listed below:

PEO1: To enable the students to think out-of-the-box solutions for addressing societal issues through ICT.

PEO2: To impart skills in students to analyse, design and implement Software/Hardware solutions to solve interdisciplinary and complex problems.

PEO3: To expose the students towards effective dissemination of research findings in order to become successful entrepreneurs or to pursue higher education.

PROGRAM SPECIFIC OUTCOMES (PSO's)

PSO1: **Professional Skills**: The ability to understand, analyze and develop computer programs in the areas related to algorithms, system software, multimedia, web design, big data analytics, and networking for efficient design of computer-based systems of varying complexity.

PSO2: **Problem-Solving Skills**: The ability to apply standard practices and strategies in software project development using open-ended programming environments to deliver a quality product for business success.

PSO3: **Successful Career and Entrepreneurship**: The ability to employ modern computer languages, environments, and platforms in creating innovative career paths to be an entrepreneur, and a zest for higher studies.

PROGRAMME OUTCOMES (POs)

A student will develop:

PO1 - ENGINEERING KNOWLEDGE: An ability to apply knowledge of Mathematics, Science and Engineering Fundamentals in Electronics and Communication Engineering.

PO2 - PROBLEM ANALYSIS: An ability to analyze and interpret data by designing and conducting experiments. Develop the knowledge of developing algorithms, designing, implementation and testing applications in electronics and communication related areas.

PO3 - DESIGN/ DEVELOPMENT OF SOLUTION: An ability to Design a system Component or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability, and sustainability.

PO4 - CONDUCTION OF INVESTIGATION OF COMPLEX PROBLEMS: An ability to Identify, formulate and solve engineering problems.

PO5 - MODERN TOOL USAGE: An ability to use the techniques, skills and modern engineering tools necessary for engineering practice.

PO6 - THE ENGINEERING AND SOCIETY: Broad education necessary to understand the impact of engineering solutions in a global, economic, environmental and societal context.

PO7 - ENVIRONMENT & SUSTAINABILITY: Understand the impact of professional engineering solution in societal and environmental contexts, and demonstrate the knowledge of, and need of sustainable development.

PO8 - ETHICS: An ability to understand the professional, social and ethical responsibility.

PO9 - INDIVIDUAL AND TEAM WORK: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10 - COMMUNICATION: An ability to Communicate effectively in order to succeed in their profession such as, being able to write effective reports and design documentation, make effective presentations.

PO11 - PROJECT MANAGEMENT & FINANCE: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in team, to manage projects and in multidisciplinary environment.

PO12 - LIFE-LONG LEARNING: Recognize the need and an ability to engage in life-long learning.

COURSE OUTCOMES (COs)

CO1	Students will be able to understand Different Type of LAN& Network Equipments.
CO2	Students will be able to understand and verify standard Network topologies i.e. Star, Bus, Ring etc.
CO3	Students will be able to understand LAN installations and Configurations.
CO4	Students will be able to implement various types of error correcting techniques.
CO5	Students will be able to implement various types of framing methods.
CO6	Students will be able to understand and implement client-server communication and estimate the RTT from client to server using TCP.
CO7	Students will be able to implement client-server communication using UDP (Single client connections/multiple client's connections).
CO8	Students will be able to understand and implement client-server communication using TCP with multiplexed I/O operations.
CO9	Students will be able to implement Routing algorithm.

INDEX

Experiment No.1
Networks and Network Equipment

Practical No. 1

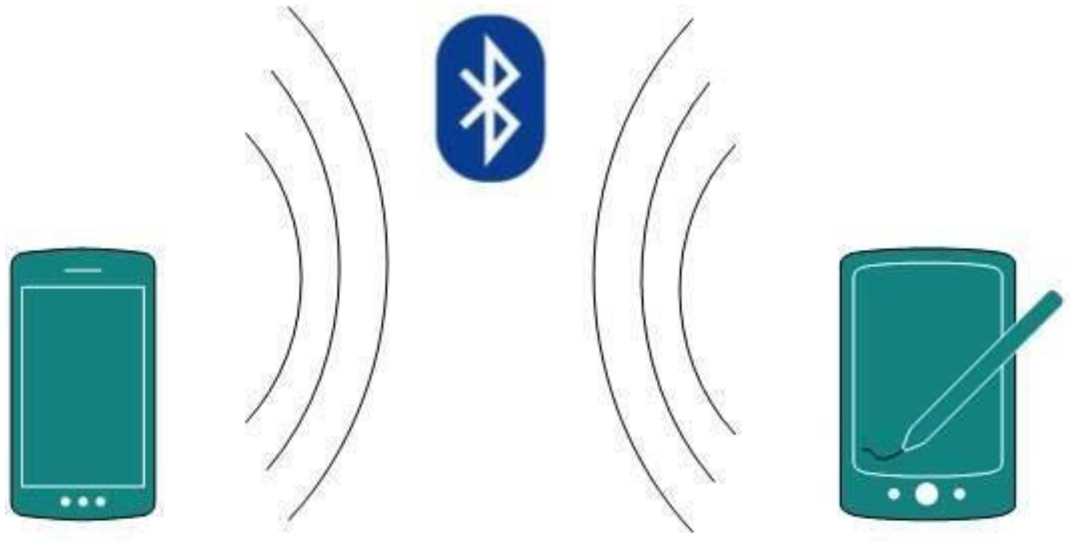
Aim: Study of different type of Networks and Network Equipment.

Description:

<u>Types of Networks</u>

a) Personal Area Network

A Personal Area Network (PAN) is smallest network which is very personal to a user. This may include Bluetooth enabled devices or infra-red enabled devices. PAN has connectivity range up to 10 meters. PAN may include wireless computer keyboard and mouse, Bluetooth enabled headphones, wireless printers and TV remotes.

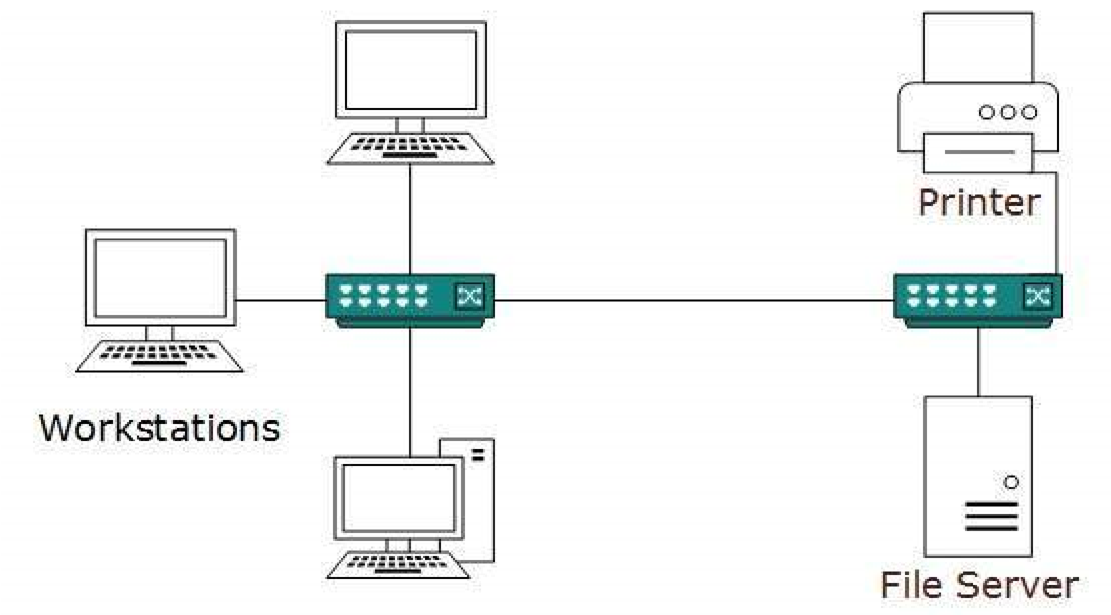


For example, Piconet is Bluetooth-enabled Personal Area Network which may contain up to 8 devices connected together in a master-slave fashion.

b) Local Area Network

A computer network spanned inside a building and operated under single administrative system is generally termed as Local Area Network (LAN). Usually,LAN covers an organization' offices, schools, colleges or universities. Number of systems connected in LAN may vary from as least as two to as much as 16 million.

LAN provides a useful way of sharing the resources between end users.The resources such as printers, file servers, scanners, and internet are easily sharable among computers.



LANs are composed of inexpensive networking and routing equipment. It may contains local servers serving file storage and other locally shared applications. It mostly operates on private IP addresses and does not involve heavy routing. LAN works under its own local domain and controlled centrally.

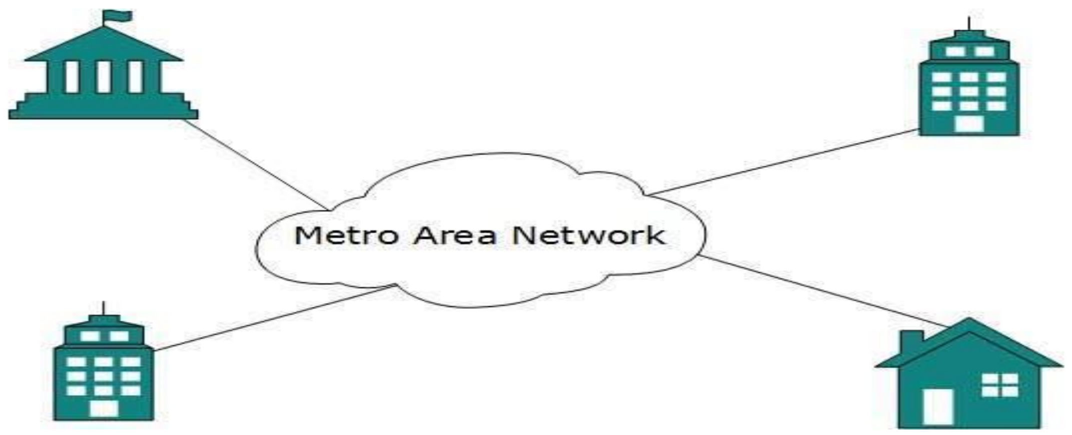
LAN uses either Ethernet or Token-ring technology. Ethernet is most widely employed LAN technology and uses Star topology, while Token-ring is rarely seen.

LAN can be wired,wireless, or in both forms at once.

c) Metropolitan Area Network

The Metropolitan Area Network (MAN) generally expands throughout a city such as cable TV network. It can be in the form of Ethernet,Token-ring, ATM, or Fiber Distributed Data Interface (FDDI).

Metro Ethernet is a service which is provided by ISPs. This service enables its users to expand their Local Area Networks. For example, MAN can help an organization to connect all of its offices in a city.

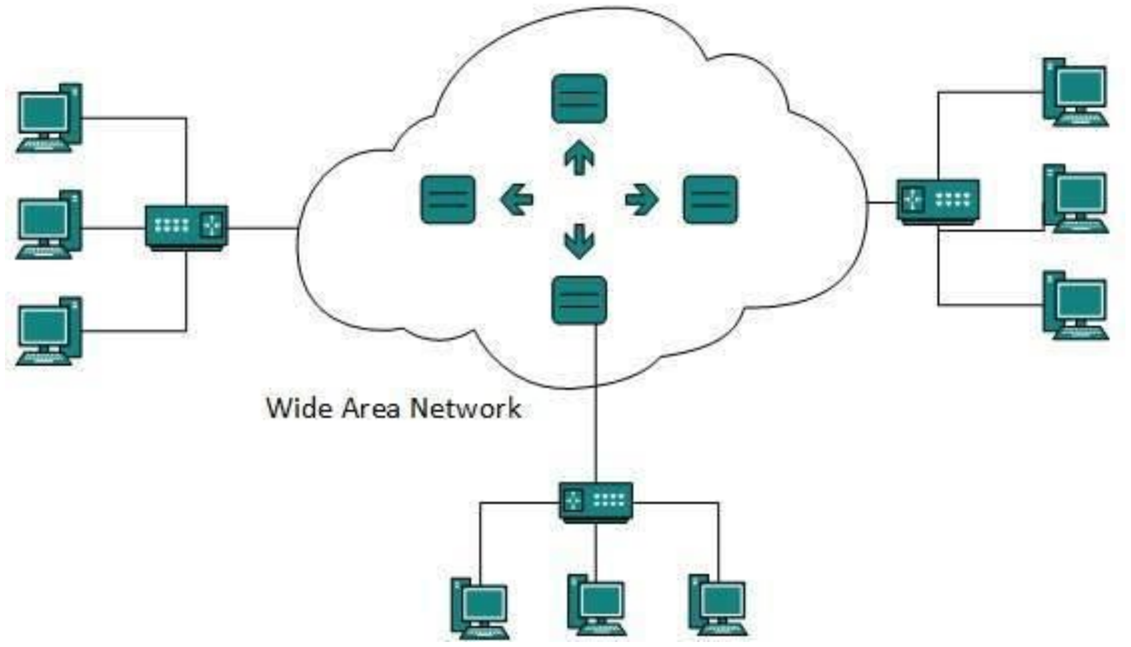


Backbone of MAN is high-capacity and high-speed fiber optics. MAN works in between Local Area Network and Wide Area Network. MAN provides uplink for LANs to WANs or internet.

d) Wide Area Network

As the name suggests,the Wide Area Network (WAN) covers a wide area which may span across provinces and even a whole country. Generally, telecommunication networks are Wide Area

Network. These networks provide connectivity to MANs and LANs. Since they are equipped with very high speed backbone, WANs use very expensive network equipment.



WAN may use advanced technologies such as Asynchronous Transfer Mode (ATM), Frame Relay, and Synchronous Optical Network (SONET). WAN may be managed by multiple administration.

e) Internetwork

A network of networks is called an internetwork, or simply the internet. It is the largest network in existence on this planet.The internet hugely connects all WANs and it can have connection to LANs and Home networks. Internet uses TCP/IP protocol suite and uses IP as its addressing protocol. Present day, Internet is widely implemented using IPv4. Because of shortage of address spaces, it is gradually migrating from IPv4 to IPv6.

Internet enables its users to share and access enormous amount of information worldwide. It uses WWW, FTP, email services, audio and video streaming etc. At huge level, internet works on Client-Server model.

Internet uses very high speed backbone of fiber optics. To inter-connect various continents, fibers are laid under sea known to us as submarine communication cable.

Internet is widely deployed on World Wide Web services using HTML linked pages and is accessible by client software known as Web Browsers. When a user requests a page using some web browser located on some Web Server anywhere in the world, the Web Server responds with the proper HTML page. The communication delay is very low.

Internet is serving many proposes and is involved in many aspects of life. Some of them are:

- Web sites
- E-mail
- Instant Messaging
- Blogging
- Social Media
- Marketing
- Networking
- Resource Sharing
- Audio and Video Streaming

1) **Inter-connectivity**- Components of a network can be connected to each other differently in some fashion. By connectedness we mean either logically , physically , or both ways.

- Every single device can be connected to every other device on network, making the network mesh.
- All devices can be connected to a single medium but geographically disconnected, created bus like structure.
- Each device is connected to its left and right peers only, creating linear structure.
- All devices connected together with a single device, creating star like structure.
- All devices connected arbitrarily using all previous ways to connect each other, resulting in a hybrid structure.

2) **Administration**- From an administrator's point of view, a network can be private network which belongs a single autonomous system and cannot be accessed outside its physical or logical domain.A network can be public which is accessed by all.

3) **Architecture-** Computer networks can be discriminated into various types such as Client-Server,peer-to-peer or hybrid, depending upon its architecture.

- There can be one or more systems acting as Server. Other being Client, requests the Server to serve requests.Server takes and processes request on behalf of Clients.
- Two systems can be connected Point-to-Point, or in back-to-back fashion. They both reside at the same level and called peers.
- There can be hybrid network which involves network architecture of both the above types.

Network Applications

Computer systems and peripherals are connected to form a network.They provide numerous advantages:

- Resource sharing such as printers and storage devices
- Exchange of information by means of e-Mails and FTP
- Information sharing by using Web or Internet
- Interaction with other users using dynamic web pages
- IP phones
- Video conferences
- Parallel computing
- Instant messaging

Computer Network Components

Computer networks components comprise both physical parts(hardware) as well as the software required for installing computer networks, both at organizations and at home. The hardware components are the server, client, peer, transmission medium, and connecting devices. The software components are operating system and protocols.

a) **Computer Network Hardware/Equipments**

Networking hardware, also known as network equipment or computer networking devices, are electronic devices which are required for communication and interaction between devices on a computer network. Specifically, they mediate data transmission in a computer network.

Network Hardware Components

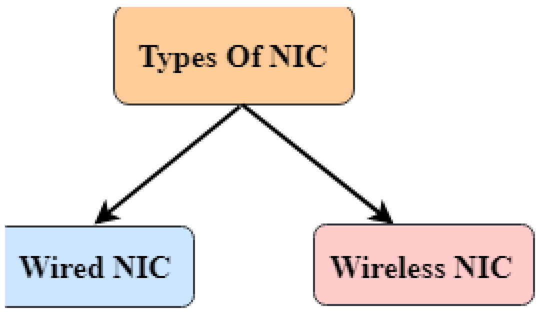
1. **Servers:** Servers are high-configuration computers that manage the resources of the network. The network operating system is typically installed in the server and so they give user accesses to the network resources. Servers can be of various kinds: file servers, database servers, print servers etc.
2. **Clients:** Clients are computers that request and receive service from the servers to access and use the network resources.
3. **Peers:** Peers are computers that provide as well as receive services from other peers in a workgroup network.
4. **Transmission Media:** Transmission media are the channels through which data is transferred from one device to another in a network. Transmission media may be guided media like coaxial cable, fibre optic cables etc; or maybe unguided media like microwaves, infra-red waves etc.
5. **Network Interface Unit(NIC):** Each computer in a network has a special expansion card called a network interface card (NIC). The NIC prepares(formats) and sends data, receives data, and controls data flow between the computer and the network. On the transmit side, the NIC passes frames of data on to the physical layer, which transmits the data to the physical link. On the receiver's side, the NIC processes bits received from the physical layer and processes the message based on its contents.

   It can support a transfer rate of 10,100 to 1000 Mb/s.The MAC address or physical address is encoded on the network card chip which is assigned by the IEEE to identify a network card uniquely. The MAC address is stored in the PROM (Programmable read-only memory).

   There are two types of NIC:

   a)Wired NIC: The Wired NIC is present inside the motherboard. Cables and connectors are used with wired NIC to transfer data.

   b)Wireless NIC: The wireless NIC contains the antenna to obtain the connection over the wireless network. For example, laptop computer contains the wireless NIC.



6. **Connecting Devices:** Connecting devices act as middleware between networks or computers, by binding the network media together. Some of the common connecting devices are:

   i. Repeater – A repeater operates at the physical layer. Its job is to regenerate the signal over the same network before the signal becomes too weak or corrupted so as to extend the length to which the signal can be transmitted over the same network. An important point to be noted about repeaters is that they do not amplify the signal. When the signal becomes weak, they copy the signal bit by bit and regenerate it at the original strength. It is a 2 port device.

ii. Hub – A hub is basically a multiport repeater. A hub connects multiple wires coming from different branches, for example, the connector in star topology which connects different stations. Hubs cannot filter data, so data packets are sent to all connected devices. In other words, collision domain of all hosts connected through Hub remains one. Also, they do not have intelligence to find out best path for data packets which leads to inefficiencies and wastage.

Types of Hub

- **Active Hub:-** These are the hubs which have their own power supply and can clean, boost and relay the signal along with the network. It serves both as a repeater as well as wiring centre. These are used to extend the maximum distance between nodes.
- **Passive Hub :-** These are the hubs which collect wiring from nodes and power supply from active hub. These hubs relay signals onto the network without cleaning and boosting them and can't be used to extend the distance between nodes.

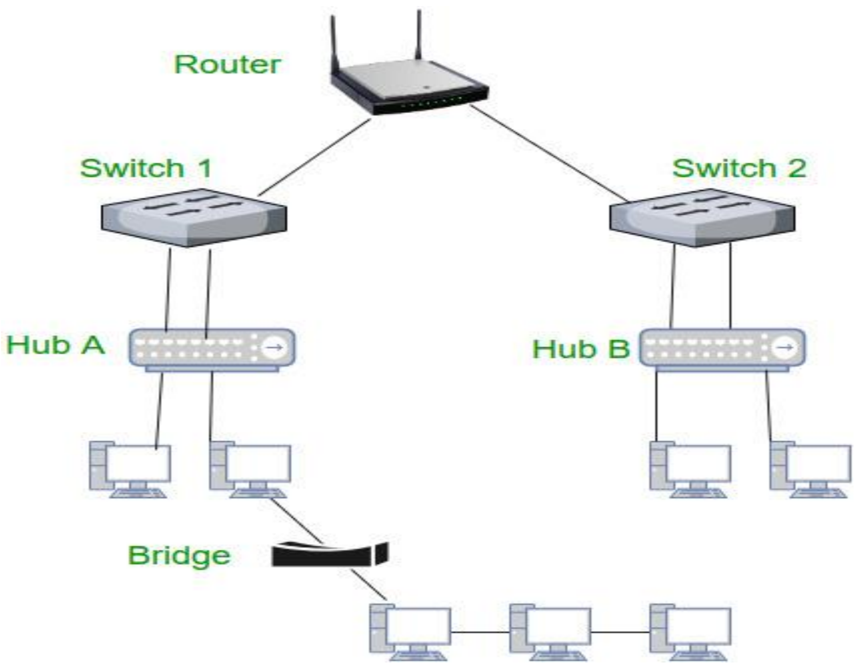
iii. Bridge – A bridge operates at data link layer. A bridge is a repeater, with add on the functionality of filtering content by reading the MAC addresses of source and destination. It is also used for interconnecting two LANs working on the same protocol. It has a single input and single output port, thus making it a 2 port device.

Types of Bridges

- **Transparent Bridges:-** These are the bridge in which the stations are completely unaware of the bridge's existence i.e. whether or not a bridge is added or deleted from the network, reconfiguration of the stations is unnecessary. These bridges make use of two processes i.e. bridge forwarding and bridge learning.
- **Source Routing Bridges:-** In these bridges, routing operation is performed by source station and the frame specifies which route to follow. The hot can discover frame by sending a special frame called discovery frame, which spreads through the entire network using all possible paths to destination.

iv. Switch – A switch is a multiport bridge with a buffer and a design that can boost its efficiency(a large number of ports imply less traffic) and performance. A switch is a data link layer device. The switch can perform error checking before forwarding data, that makes it very efficient as it does not forward packets that have errors and forward good packets selectively to correct port only. In other words, switch divides collision domain of hosts, but broadcast domain remains same.

v. Routers – A router is a device like a switch that routes data packets based on their IP addresses. Router is mainly a Network Layer device. Routers normally connect LANs and WANs together and have a dynamically updating routing table based on which they make decisions on routing the data packets. Router divide broadcast domains of hosts connected through it.



vi. Gateway – A gateway, as the name suggests, is a passage to connect two networks together that may work upon different networking models. They basically work as the messenger agents that take data from one system, interpret it, and transfer it to another system. Gateways are also called protocol converters and can operate at any network layer. Gateways are generally more complex than switch or router.

b) **Computer Network Software**

Network software encompasses a broad range of software used for design, implementation, and operation and monitoring of computer networks.

Network Software Components

1. **Networking Operating System:** Network Operating Systems is typically installed in the server and facilitate workstations in a network to share files, database, applications, printers etc.
2. **Protocol Suite:** A protocol is a rule or guideline followed by each computer for data communication. Protocol suite is a set of related protocols that are laid down for computer networks. The two popular protocol suites are:
   a. OSI Model (Open System Interconnections)
   b. TCP / IP Model

Experiment No.2
Network Topologies
Practical No. 2

Aim: Study and verification of standard Network Topologies i.e. Star, Bus, Ring, Mesh etc.

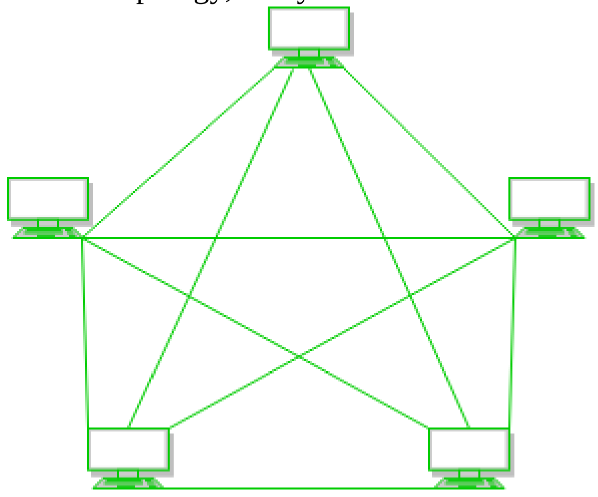
Description:

Network Topologies

The arrangement of a network which comprises of nodes and connecting lines via sender and receiver is referred as network topology. The various network topologies are :

a) Mesh Topology :

In mesh topology, every device is connected to another device via particular channel.



Every device is connected with another via dedicated channels. These channels are known as links.

- If suppose, N number of devices are connected with each other in mesh topology, then total number of ports that is required by each device is N-1. In the Figure 1, there are 5 devices connected to each other, hence total number of ports required is 4.
- If suppose, N number of devices are connected with each other in mesh topology, then total number of dedicated links required to connect them is $^{N}C_{2}$ i.e. N(N-1)/2. In the Figure 1, there are 5 devices connected to each other, hence total number of links required is 5*4/2 = 10.

Advantages of this topology :

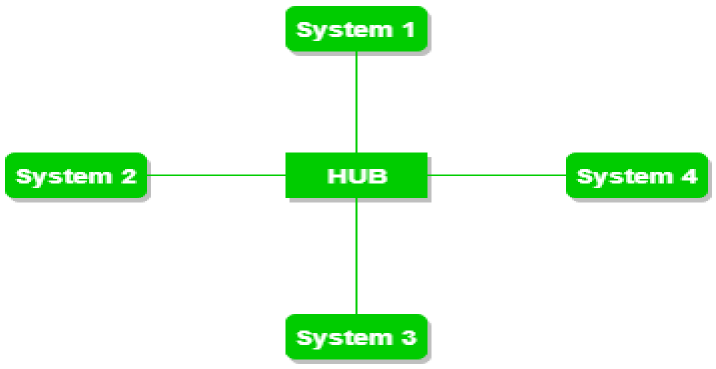
- It is robust.
- Fault is diagnosed easily. Data is reliable because data is transferred among the devices through dedicated channels or links.
- Provides security and privacy.

Problems with this topology :

- Installation and configuration is difficult.
- Cost of cables are high as bulk wiring is required, hence suitable for less number of devices.
- Cost of maintenance is high.

b) Star Topology :

In star topology, all the devices are connected to a single hub through a cable. This hub is the central node and all others nodes are connected to the central node. The hub can be passive in nature i.e. not intelligent hub such as broadcasting devices, at the same time the hub can be intelligent known as active hubs. Active hubs have repeaters in them.



A star topology having four systems connected to single point of connection i.e. hub.

Advantages of this topology :

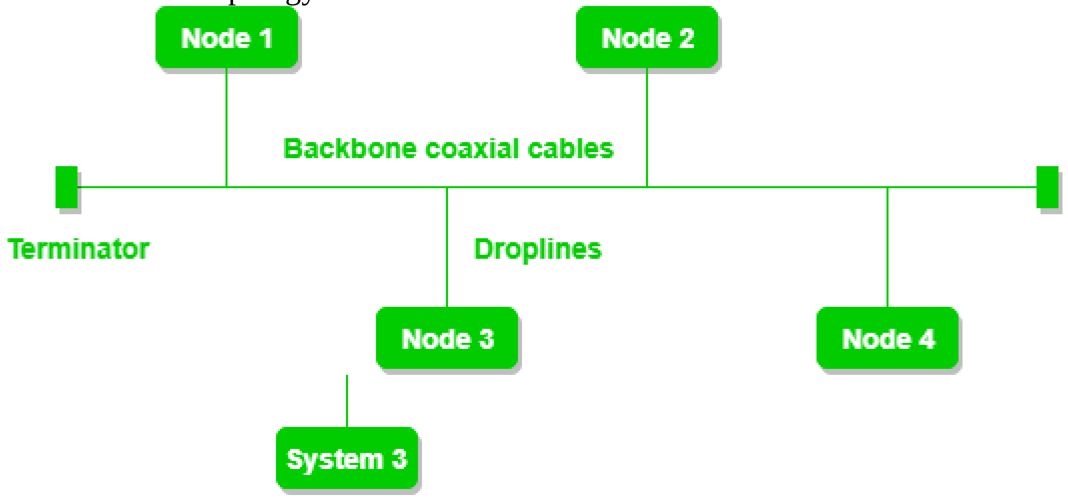
- If N devices are connected to each other in star topology, then the number of cables required to connect them is N. So, it is easy to set up.
- Each device require only 1 port i.e. to connect to the hub.

Problems with this topology :

- If the concentrator (hub) on which the whole topology relies fails, the whole system will crash down.
- Cost of installation is high.
- Performance is based on the single concentrator i.e. hub.

c) Bus Topology :

Bus topology is a network type in which every computer and network device is connected to single cable. It transmits the data from one end to another in single direction. No bi-directional feature is in bus topology.



A bus topology with shared backbone cable. The nodes are connected to the channel via drop lines.

Advantages of this topology :

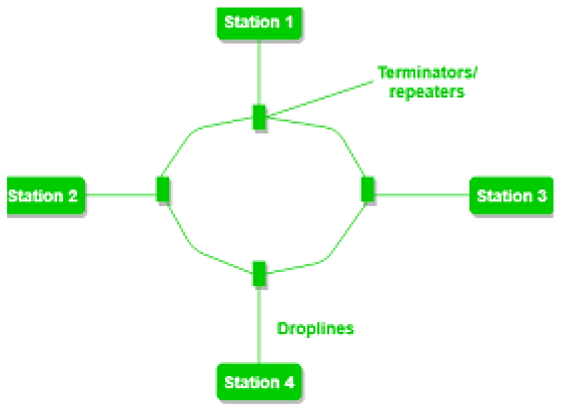
- If N devices are connected to each other in bus topology, then the number of cables required to connect them is 1 which is known as backbone cable and N drop lines are required.
- Cost of the cable is less as compared to other topology, but it is used to built small networks.

Problems with this topology :

- If the common cable fails, then the whole system will crash down.
- If the network traffic is heavy, it increases collisions in the network. To avoid this, various protocols are used in MAC layer known as Pure Aloha, Slotted Aloha, CSMA/CD etc.

d) Ring Topology :

In this topology, it forms a ring connecting a devices with its exactly two neighbouring devices.



A ring topology comprises of 4 stations connected with each forming a ring..

The following operations takes place in ring topology are :

1. One station is known as **monitor** station which takes all the responsibility to perform the operations.
2. To transmit the data, station has to hold the token. After the transmission is done, the token is to be released for other stations to use.
3. When no station is transmitting the data, then the token will circulate in the ring.
4. There are two types of token release techniques : **Early token release** releases the token just after the transmitting the data and **Delay token release** releases the token after the acknowledgement is received from the receiver.

Advantages of this topology :

- The possibility of collision is minimum in this type of topology.
- Cheap to install and expand.

Problems with this topology :

- Troubleshooting is difficult in this topology.
- Addition of stations in between or removal of stations can disturb the whole topology.

e) Hybrid Topology :

This topology is a collection of two or more topologies which are described above. This is a scalable topology which can be expanded easily. It is reliable one but at the same it is a costly topology.

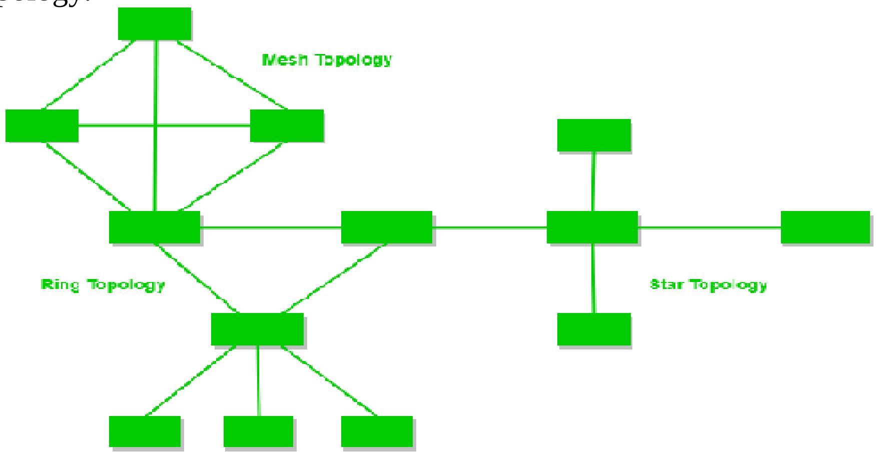


Figure - A Hybrid Topology

Experiment No.3
LAN Installation and Configurations
Practical No. 3

Aim: Study of LAN installation and configuration using Packet Tracer.

Description:

Introduction of Packet Tracer

Packet Tracer is a cross-platform visual simulation tool designed by Cisco Systems that allows users to create network typologies and imitate modern computer networks. The software allows users to simulate the configuration of Cisco routers and switches using a simulated command line interface.

Packet Tracer can also be run on Linux and Microsoft Windows and also macOS. Similar Android and iOS apps are also available. Packet Tracer allows users to create simulated network topologies by dragging and dropping routers, switches and various other types of network devices. A physical connection between devices is represented by a 'cable' item. Packet Tracer supports an array of simulated Application Layer protocols, as well as basic routing with RIP, OSPF, EIGRP, BGP, to the extents required by the current CCNA curriculum. As of version 5.3, Packet Tracer also supports the Border Gateway Protocol.

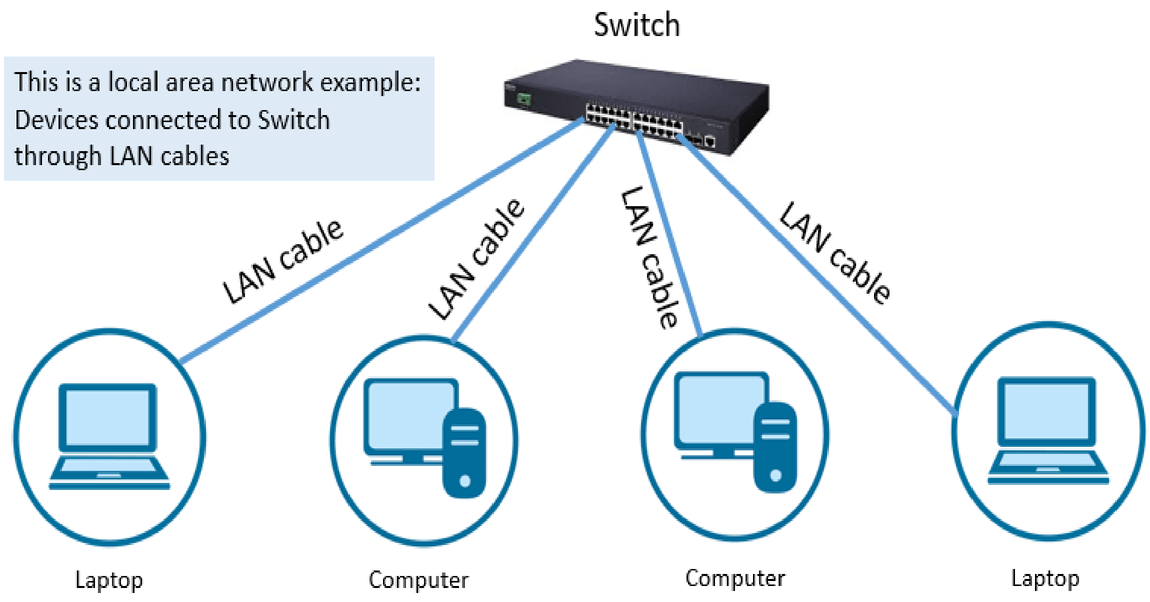
In addition to simulating certain aspects of computer networks, Packet Tracer can also be used for collaboration. As of Packet Tracer 5.0, Packet Tracer supports a multi-user system that enables multiple users to connect multiple topologies together over a computer network. Packet Tracer also allows instructors to create activities that students have to complete. Packet Tracer is often used in educational settings as a learning aid. Cisco Systems claims that Packet Tracer is useful for network experimentation.

Overview of LAN

When two or more than two devices or computers or laptops are connected together they form a network. Local area network is a private network that operates within a building or Home or Office etc. Local area networks are widely used for sharing resources (Printer) or information among devices or computers which are connected in the network.

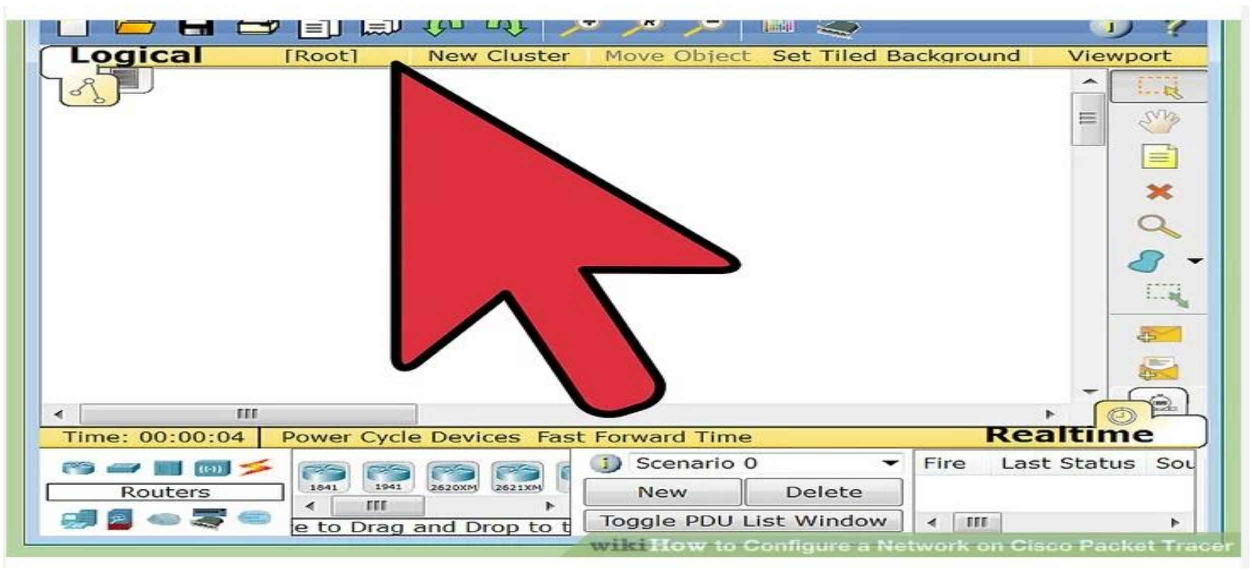
Local area networks (LANs) have become common place in the information age. The introduction of low-cost computers, along with the need to share information and hardware resources, in an office environment led the way to the development of inexpensive LAN technologies.

LAN technologies are designed to interconnect computers across short distances (e.g., within a single building or room). Today’s LANs are inexpensive, highly reliable, and easy to install and maintain. To connect a computer to a LAN, a computer must have an interface card. The computer views this interface card as any other I/O device. Data sent to the interface card are transmitted onto the communication medium (copper wire, coax cable, optical fiber, or through the air at a given radio frequency), where it is then received by another computer attached to the same medium

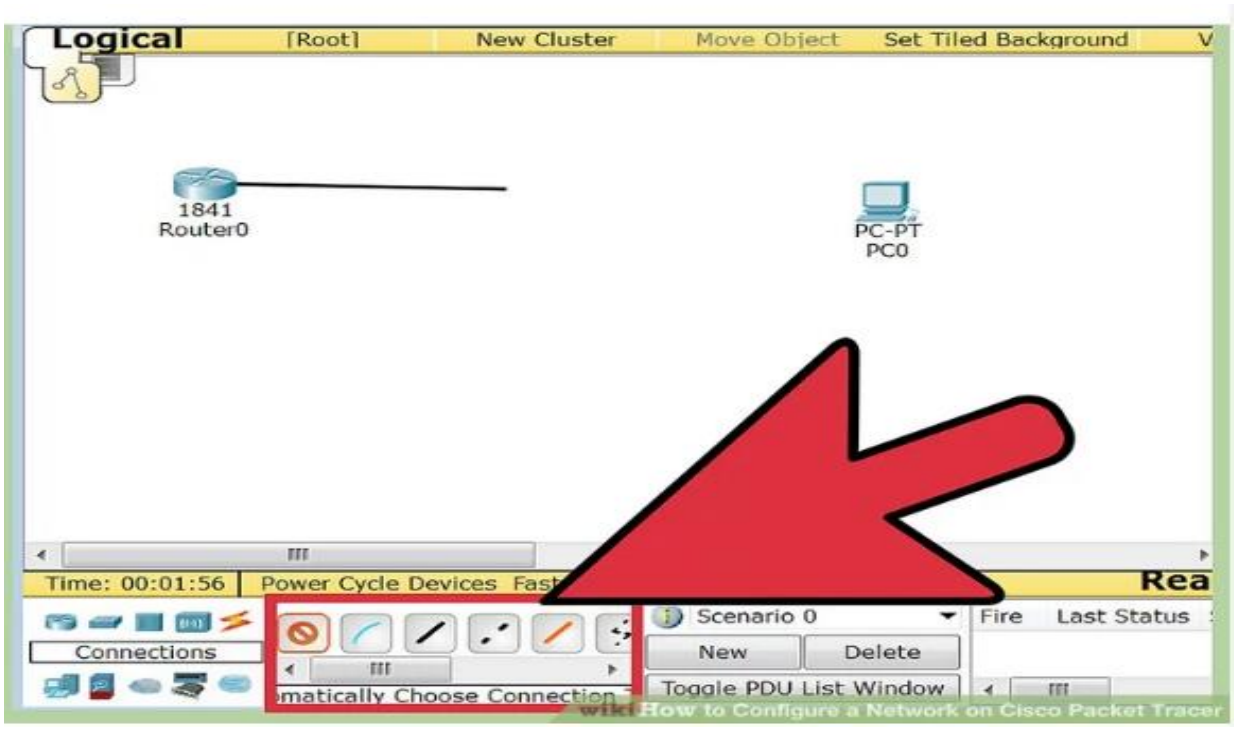

Switch
This is a local area network example: Devices connected to Switch through LAN cables
LAN cable
LAN cable
LAN cable
LAN cable
Laptop
Computer
Computer
Laptop

General Steps of Configurations of LAN using Packet Tracer

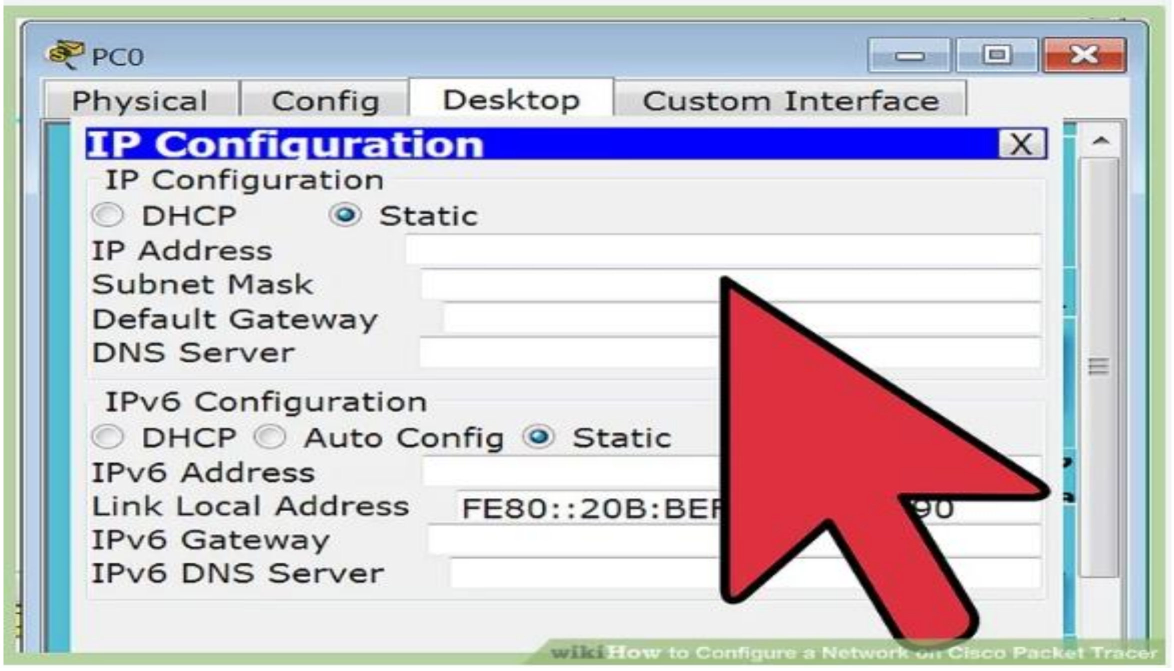
1. **Open Network Topology**: Once we have opened Network Topology on Cisco Packet Tracer, access network and identify the components of network, for example; Servers, Routers, End Devices, etc.



2. **Complete the cabling:** Access the cables section and connect completely and correctly the cables between the networks in order to ensure connectivity between the devices in the network using the connections table given.



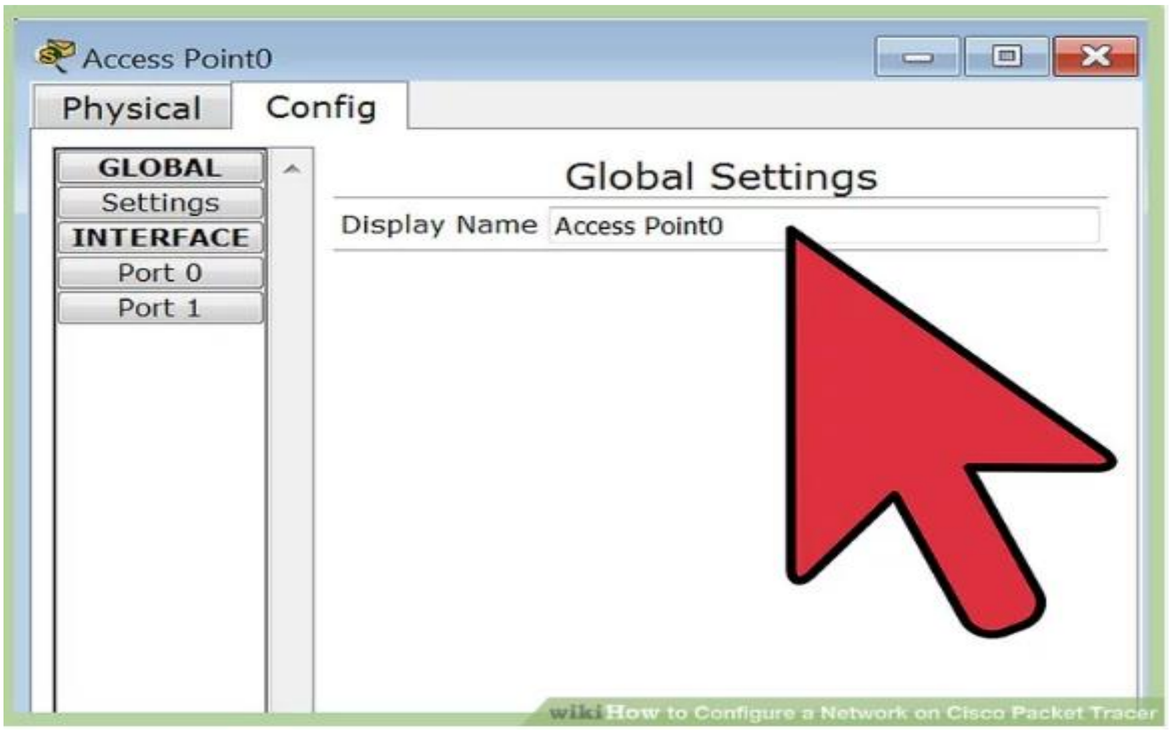
3. **Configure the IP addresses on the end devices:**Using the address table still, correctly and completely configure the IP addresses on all end devices. This can be done by accessing the desktop platform on each device and locating the IP configuration section. The reason for doing this is to enable the devices be on the right network.



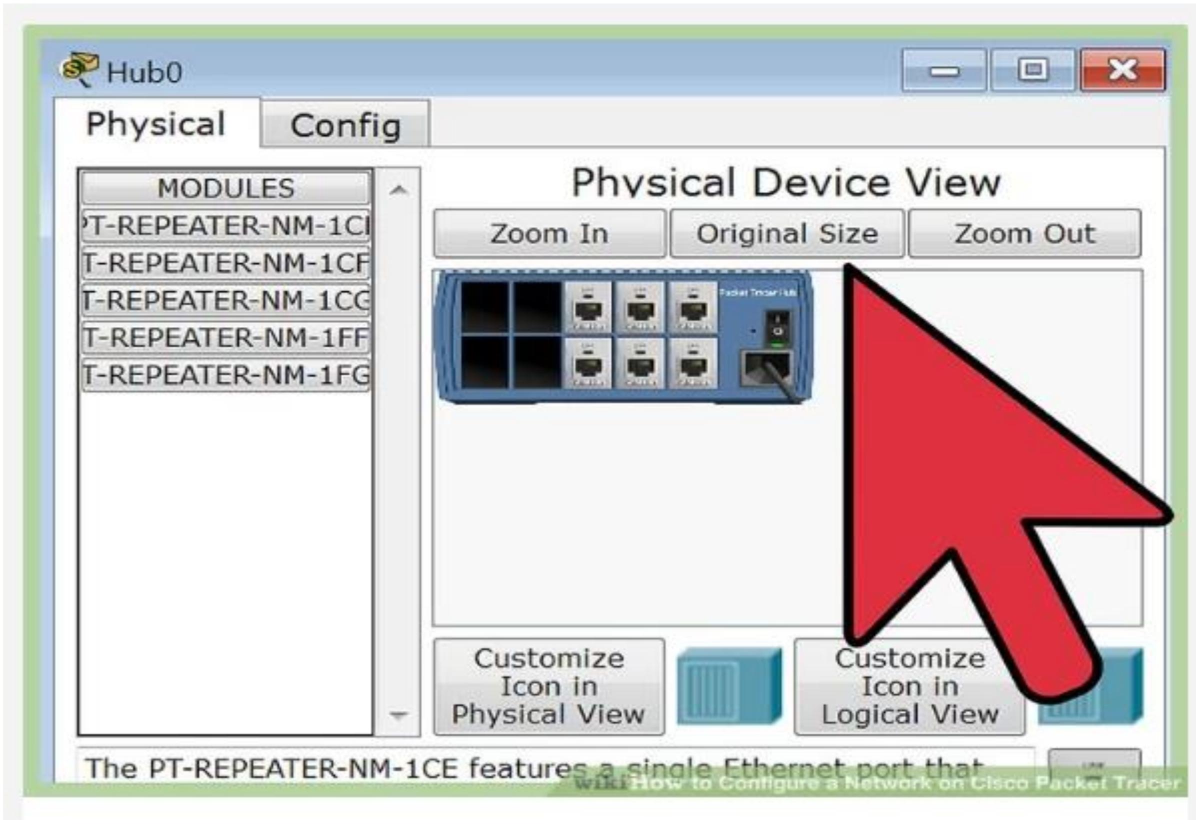
4. **Configure the IP addresses on your routers and switches:** After configuring the right IP addresses on the end devices, you will have to do the same on the routers and switches also, using the address table. But this time in a different way because there's no desktop platform on the routers and switches. You will have to access the configuration panel on both devices and this can be done in two ways:

- Click on the device and open the Command Line Interface (CLI) and then type in the right commands to configure the right addresses for the router using the addressing table.

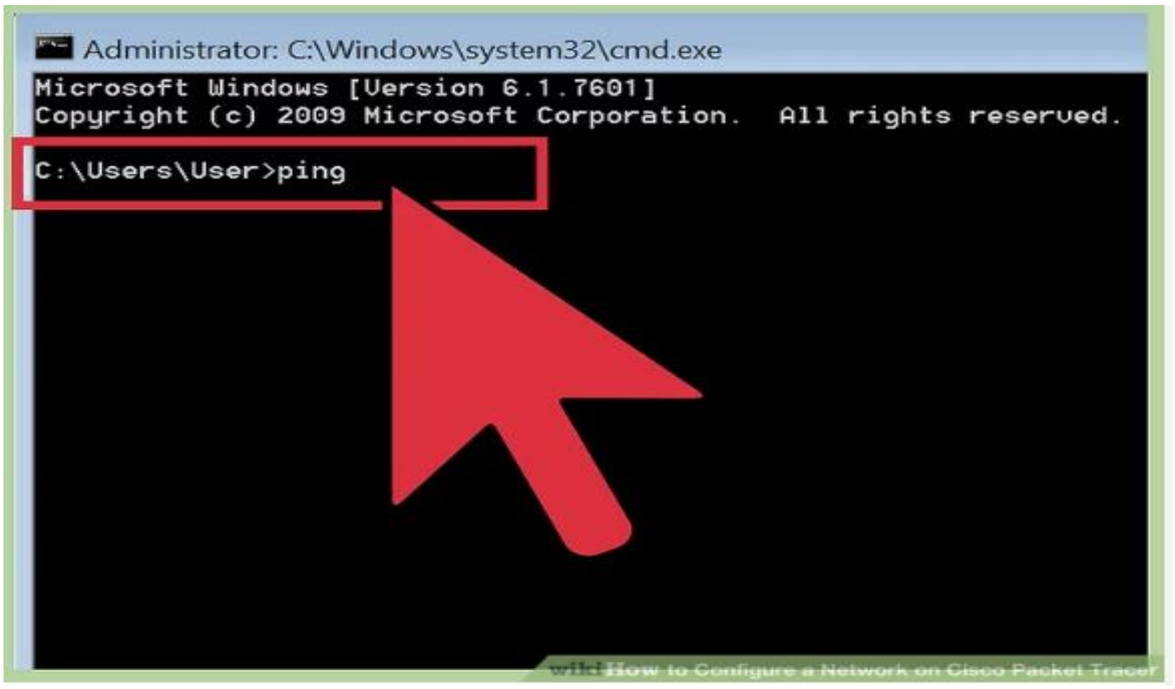
- Use a console cable from an end device and connect it to the device you wish to configure and access the terminal platform on the end device and it will take you to the device's Command Line Interface and then you type in the commands in other to configure the right addresses.



5. **Configure your default gateway.** After configuring the IP addresses, you will need to configure the default gateway also. The reason for this is so the end devices would know what network they are operating on. You can find the default gateway either in the addressing table (if given) or in the network topology.


Hub0
Physical
Config
MODULES
PT-REPEATER-NM-1CE
PT-REPEATER-NM-1CF
PT-REPEATER-NM-1CG
PT-REPEATER-NM-1FF
PT-REPEATER-NM-1FG
Physical Device View
Zoom In
Original Size
Zoom Out
Customize Icon in Physical View
Customize Icon in Logical View
The PT-REPEATER-NM-1CE features a single Ethernet port that
wiki How to Configure a Network on Cisco Packet Tracer

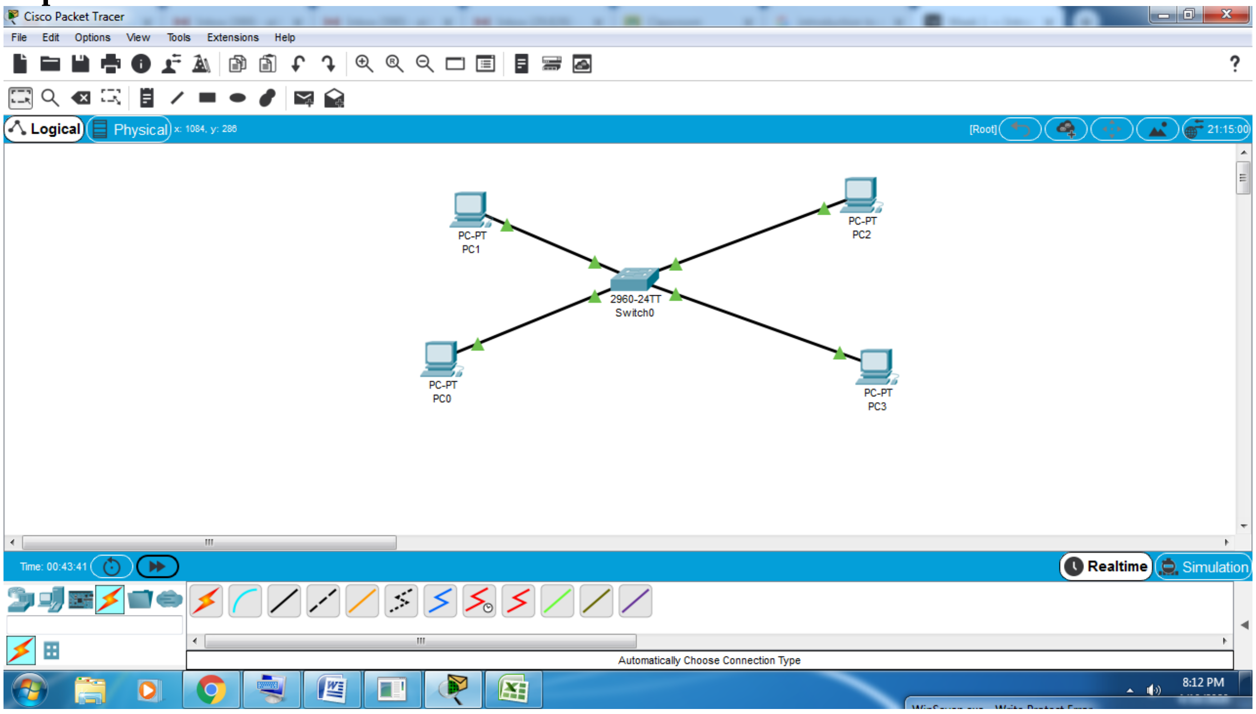
6. **Test connectivity:** After configuring the addresses, we will have to test connectivity by opening a command prompt window on the end devices and try pinging the address which the network operates on. If it gives us a reply, it means our network was configured correctly.

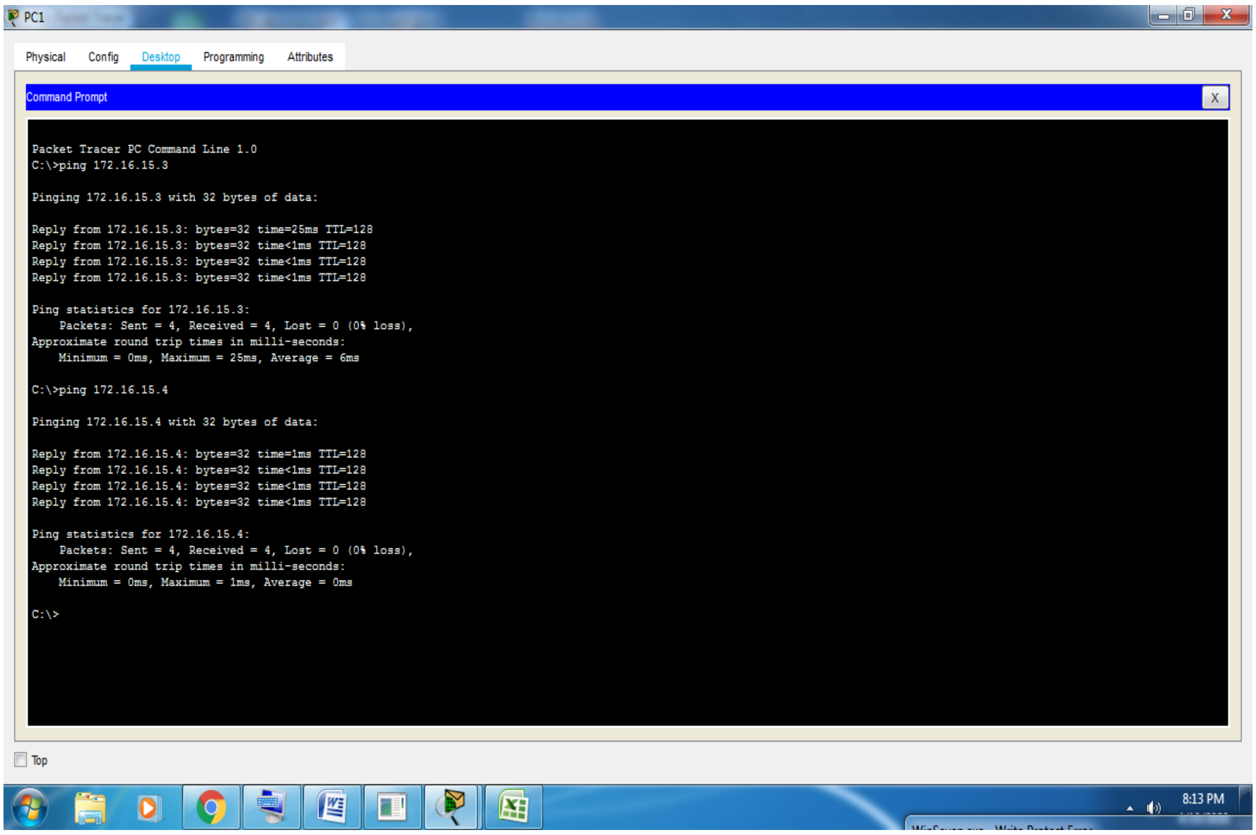


Administrator: C:\Windows\system32\cmd.exe

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\User>ping
```

Snapshot





```
Packet Tracer PC Command Line 1.0
C:\>ping 172.16.15.3

Pinging 172.16.15.3 with 32 bytes of data:

Reply from 172.16.15.3: bytes=32 time=25ms TTL=128
Reply from 172.16.15.3: bytes=32 time<1ms TTL=128
Reply from 172.16.15.3: bytes=32 time<1ms TTL=128
Reply from 172.16.15.3: bytes=32 time<1ms TTL=128

Ping statistics for 172.16.15.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 25ms, Average = 6ms

C:\>ping 172.16.15.4

Pinging 172.16.15.4 with 32 bytes of data:

Reply from 172.16.15.4: bytes=32 time=1ms TTL=128
Reply from 172.16.15.4: bytes=32 time<1ms TTL=128
Reply from 172.16.15.4: bytes=32 time<1ms TTL=128
Reply from 172.16.15.4: bytes=32 time<1ms TTL=128

Ping statistics for 172.16.15.4:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\>
```

Experiment No. 4
Error Detection and Correcting Techniques
Practical No. 4

Aim: WAP in c to implement Error Correcting Technique using Hamming Code method.

Description:

Hamming code is a popular error detection and error correction method in data communication. Hamming code can only detect 2 bit error and correct a single bit error which means it is unable to correct burst errors if may occur while transmission of data.

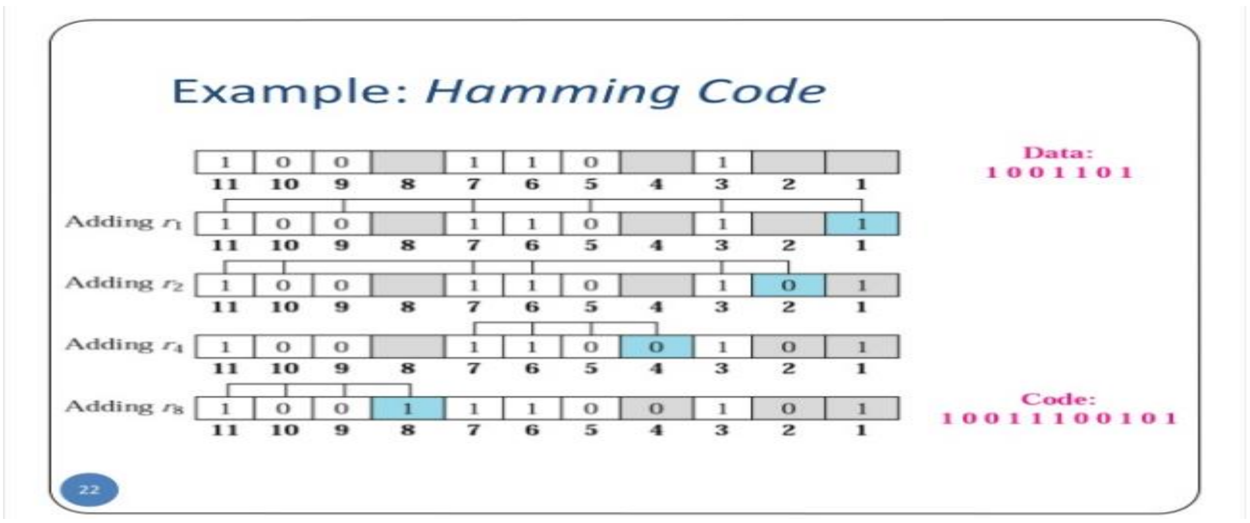
Hamming code uses redundant bits (extra bits) which are calculated according to the below formula:-

2r ≥ m+r+1
Where **r** is the number of redundant bits required and **m** is the number of data bits.
R is calculated by putting **r = 1, 2, 3 …** until the above equation becomes true.
R1 bit is appended at position **20**
R2 bit is appended at position **21**
R3 bit is appended at position **22** and so on.

These redundant bits are then added to the original data for the calculation of error at receiver's end.

At receiver's end with the help of even parity (generally) the erroneous bit position is identified and since data is in binary we take complement of the erroneous bit position to correct received data.

Respective index parity is calculated for **r1, r2, r3, r4** and so on.



Advantages of Hamming Code

1. Easy to encode and decode data at both sender and receiver end.
2. Easy to implement.

Disadvantages of Hamming Code

1. Cannot correct burst errors.
2. Redundant bits are also sent with the data therefore it requires more bandwidth to send the data.

Program:

```
#include<stdio.h>
 void main() {
   int data[10];
   intdataatrec[10],c,c1,c2,c3,i;
```

```
printf("Enter 4 bits of data one by one\n");
scanf("%d",&data[0]);
scanf("%d",&data[1]);
scanf("%d",&data[2]);
scanf("%d",&data[4]);

//Calculation of even parity
data[6]=data[0]^data[2]^data[4];
        data[5]=data[0]^data[1]^data[4];
        data[3]=data[0]^data[1]^data[2];

        printf("\nEncoded data is\n");
        for(i=0;i<7;i++)
    printf("%d",data[i]);

printf("\n\nEnter received data bits one by one\n");
for(i=0;i<7;i++)
    scanf("%d",&dataatrec[i]);

c1=dataatrec[6]^dataatrec[4]^dataatrec[2]^dataatrec[0];
        c2=dataatrec[5]^dataatrec[4]^dataatrec[1]^dataatrec[0];
        c3=dataatrec[3]^dataatrec[2]^dataatrec[1]^dataatrec[0];
        c=c3*4+c2*2+c1 ;

if(c==0) {
                    printf("\nNo error while transmission of data\n");
}
        else {
                    printf("\nError on position %d",c);

                    printf("\nData sent : ");
    for(i=0;i<7;i++)
        printf("%d",data[i]);

                    printf("\nData received : ");
    for(i=0;i<7;i++)
        printf("%d",dataatrec[i]);

                    printf("\nCorrect message is\n");

                    //if errorneous bit is 0 we complement it else vice versa
                    if(dataatrec[7-c]==0)
                                dataatrec[7-c]=1;
    else
                                dataatrec[7-c]=0;

                    for (i=0;i<7;i++) {
                                printf("%d",dataatrec[i]);
                    }
        }
```

Sample Input-Output :

```
Enter 4 bits of data one by one
1
```

```
0
1
0
Encoded data is
1010010
Enter received data bits one by one
1
0
1
0
0
1
0
No error while transmission of data
```

Practical 5

Aim: WAP in c to implement Error Detection Technique using checksum calculation.

Description:

A checksum is a error detection method in Data Communication. It is used for errors which may have been introduced during transmission or storage. It is usually applied to an installation file after it is received from the download server.

The actual procedure which yields the checksum, given a data input is called a checksum function or checksum algorithm.

Checksum method can only detect errors but is unable to correct the error.

In this method a checksum is calculated based on the given binary strings which is sent with the data as redundant bits. This data + checksum is received at receiver end and checksum is calculated again, if checksum is 0 it means no error in data received, else there exists some error in the received data.

For the purpose of this program we are finding checksum for 2 binary strings.

Checksum Algorithm

1. Take 2 binary input strings.
2. Do their binary sum to find out the checksum which will be sent to the destination or to the receiver.
3. In binary sum there are 6 cases:-
   a. If both bits are 0 and carry is 0, sum=0 and carry=0
   b. If both bits are 0 and carry is 1,sum=1 and carry=0
   c. If both bits are 1 and carry is 0,sum=0 and carry=1
   d. If both bits are 1 and carry is 1,sum=1 and carry=1
   e. If either bit is 1 and carry is 0,sum=1 and carry=0
   f. If either bit is 1 and carry is 1,sum=0 and carry=1
4. While doing the addition we have to add the binary strings from rightmost end i.e LSB to MSB.
5. When binary sum is done 1’s complement of it is taken by reversing 1’s to 0’s and vice versa.
6. The resulting 1’s complement is the Checksum.
7. Stop.

Program:

```
#include<iostream>
#include<string.h>

using namespace std;

int main()
{
   char a[20],b[20];
   char sum[20],complement[20];
   inti;

         cout<<"Enter first binary string\n";
   cin>>a;
   cout<<"Enter second binary string\n";
   cin>>b;
```

```
        if(strlen(a)==strlen(b))
    {
      char carry='0';
      int length=strlen(a);

                for(i=length-1;i>=0;i--)
      {
        if(a[i]=='0' && b[i]=='0' && carry=='0')
        {
          sum[i]='0';
          carry='0';
        }
        else if(a[i]=='0' && b[i]=='0' && carry=='1')
        {
          sum[i]='1';
          carry='0';

        }
        else if(a[i]=='0' && b[i]=='1' && carry=='0')
        {
          sum[i]='1';
          carry='0';

        }
        else if(a[i]=='0' && b[i]=='1' && carry=='1')
        {
          sum[i]='0';
          carry='1';

        }
        else if(a[i]=='1' && b[i]=='0' && carry=='0')
        {
          sum[i]='1';
          carry='0';

        }
        else if(a[i]=='1' && b[i]=='0' && carry=='1')
        {
          sum[i]='0';
          carry='1';

        }
        else if(a[i]=='1' && b[i]=='1' && carry=='0')
        {
          sum[i]='0';
          carry='1';

        }
        else if(a[i]=='1' && b[i]=='1' && carry=='1')
        {
          sum[i]='1';
          carry='1';

        }
```

```
            else
                break;
        }
        cout<<"\nSum="<<carry<<sum;

        for(i=0;i<length;i++)
        {
            if(sum[i]=='0')
                complement[i]='1';
            else
                complement[i]='0';
        }

                    if(carry=='1')
            carry='0';
        else
            carry='1';

                cout<<"\nChecksum="<<carry<<complement;
    }
    else
        cout<<"\nWrong input strings";

    return 0;
}
```

Sample Input/Output

```
Enter first binary string
101101
Enter Second binary string
110010
Sum=1011111
Checksum=0100000
```

Experiment No. 5
FRAMING METHODS
Practical 6

Aim: WAP in c to implement Bit Stuffing.

Description:

In Data Link layer, the stream of bits from physical layer are divided into data frames. The data frames can be of fixed length or variable length. In variable – length framing, the size of each frame to be transmitted may be different. So, a pattern of bits known as flag is used as a delimiter to mark the end of one frame and the beginning of the next frame. However, if the pattern occurs in the message, then mechanisms needs to be incorporated so that this situation is avoided.

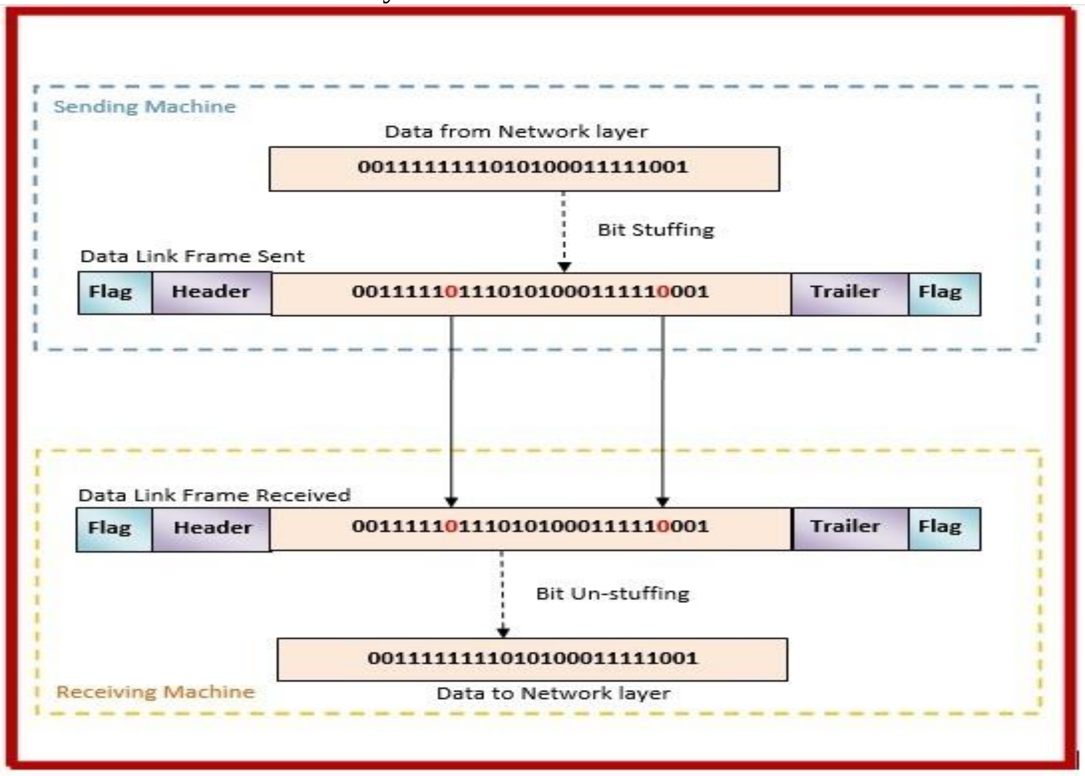
The two common approaches are −

- **Byte – Stuffing**
- **Bit – Stuffing**

Bit stuffing

A pattern of bits of arbitrary length is stuffed in the message to differentiate from the delimiter. This is also called bit – oriented framing.

Mostly flag is a special 8-bit pattern "01111110" used to define the beginning and the end of the frame. But it may be the case that this flag appears in the message. So, in this protocol what we do is, if we encounter 0 and five consecutive 1 bits, an extra 0 is added after these bits. This extra stuffed bit is removed from the data by the receiver.



Program:

```
#include<stdio.h>
int main()
{
inti=0,count=0;
 char databits[80];
printf("Enter Data Bits in the form of zero’s and one’s: ");
scanf("%s",databits);

printf("Data Bits Before Bit Stuffing:%s",databits);
printf("\nData Bits After Bit stuffing :");

 for(i=0; i<strlen(databits); i++)
 {
    if(databits[i]=='1')
        count++;
    else
        count=0;
printf("%c",databits[i]);
    if(count==5)
    {
printf("0");
        count=0;
    }
 }
 return 0;
}
```

Sample Input Output:

```
Enter frame size (Example: 8):12

Enter the frame in the form of 0 and 1 :0 1 0 1 1 1 1 1 1 0 0 1

After Bit Stuffing :0101111101001
```

Practical 7

Aim: WAP in c to implement Byte Stuffing.

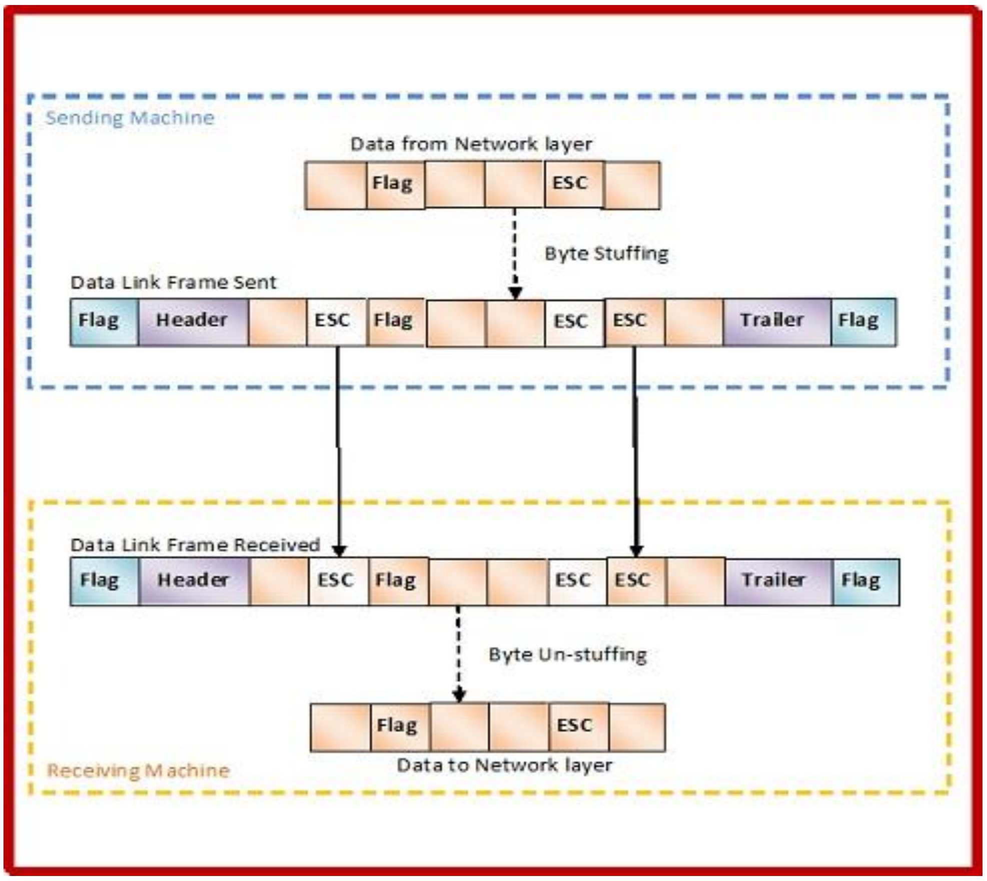
Description:

Byte-Stuffing

In character-oriented protocol, we add special characters (called flag) to distinguish beginning and end of a frame. Usually flag has 8-bit length. The character-oriented protocols are popular only with text data. While using character–oriented protocol another problem is arises, pattern used for the flag may also part of the data to send. If this happens, the destination node, when it encounters this pattern in the middle of the data, assumes it has reached the end of the frame. To deal with this problem, a **byte stuffing (also known as character stuffing)** approach was included to character-oriented protocol.

In byte stuffing a byte (usually escape character(ESC)), which has a predefined bit pattern is added to the data section of the frame when there is a character with the same pattern as the flag. Whenever the receiver encounters the ESC character, it removes from the data section and treats the next character as data, not a flag.

But the problem arises when the text contains one or more escape characters followed by a flag. To solve this problem, the escape characters that are part of the text are marked by another escape character i.e., if the escape character is part of the text, an extra one is added to show that the second one is part of the text.



Program:

```
#include<stdio.h>
#include<fcntl.h>
#include<string.h>
#define flag '*'
#define esc '#'
```

```
void main()
{
int j=1,i,pid;
 char str[50],frame[50];
printf("Enter the string:");
scanf("%s",str);
 frame[0]=flag;
  for(i=0;i<strlen(str);i++)
 {
     if (str[i]==flag || str[i]==esc)
  {
      frame[j]=esc;
      j++;
  }
      frame[j]=str[i];
  j++;
  }
 frame[j++]=flag;
 frame[j]='\0';
printf("\nFrame to send  : %s",frame);
 return 0;
}
```

Sample Input Output:

```
Enter the string:*abc##*

Frame to send  : *#*abc#####**
```

Experiment No.6
TCP DATA TRANSFER
Practical 8 and 9

Aim: Practical 8- WAP hello_client in Java where the server listens for, and accepts, a single TCP connection; it reads all the data it can from that connection, and prints it to the screen; then it closes the connection.

Practical 9- WAP hello_server in Java where the client connects to the server, sends the string “Hello, world!”, then closes the connection.

Description:

Java Socket Programming

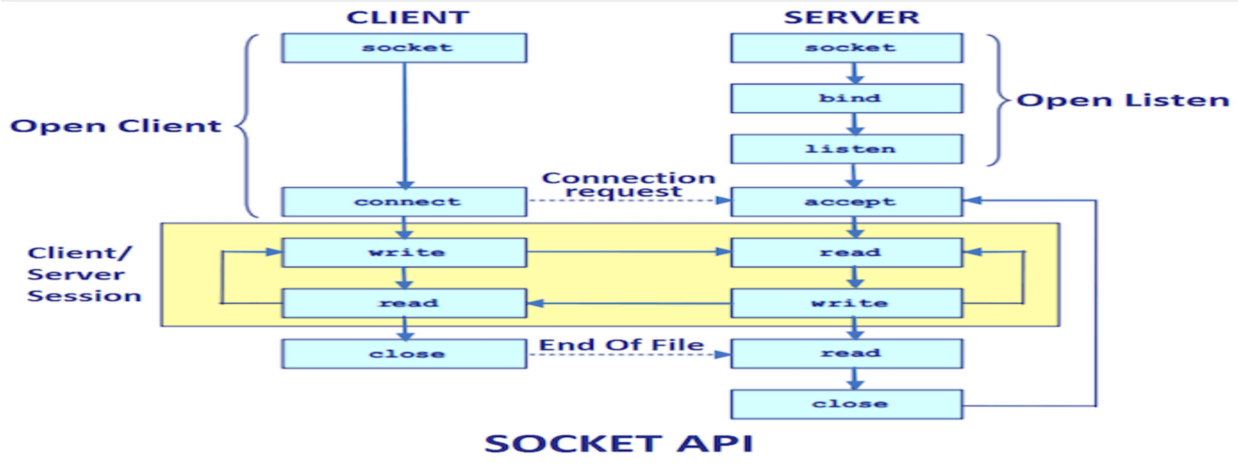
Java Socket programming is used for communication between the applications running on different JRE. Java Socket programming can be connection-oriented or connection-less.

Socket and ServerSocket classes are used for connection-oriented socket programming.

The client in socket programming must know two information:

- IP Address of Server, and
- Port number.

Here, we are going to make one-way client and server communication. In this application, client sends a message to the server, server reads the message and prints it. Here, two classes are being used: Socket and ServerSocket. The Socket class is used to communicate client and server. Through this class, we can read and write message. The ServerSocket class is used at server-side. The accept() method of ServerSocket class blocks the console until the client is connected. After the successful connection of client, it returns the instance of Socket at server-side.



Socket class

A socket is simply an endpoint for communications between the machines. The Socket class can be used to create a socket.

Important methods

Method	**Description**
1) public InputStreamgetInputStream()	returns the InputStream attached with this socket.
2) public OutputStreamgetOutputStream()	returns the OutputStream attached with this socket.
3) public synchronized void close()	closes this socket

ServerSocket class

The ServerSocket class can be used to create a server socket. This object is used to establish communication with the clients.

Important methods

Method	**Description**
1) public Socket accept()	returns the socket and establish a connection between server and client.
2) public synchronized void close()	closes the server socket.

Example of Java Socket Programming

Creating Server:

To create the server application, we need to create the instance of ServerSocket class. Here, we are using 6666 port number for the communication between the client and server. You may also choose any other port number. The accept() method waits for the client. If clients connects with the given port number, it returns an instance of Socket.

ServerSocketss=new ServerSocket(6666);

Socket s=ss.accept();//establishes connection and waits for the client

Creating Client:

To create the client application, we need to create the instance of Socket class. Here, we need to pass the IP address or hostname of the Server and a port number. Here, we are using "localhost" because our server is running on same system.

Socket s=new Socket("localhost",6666);

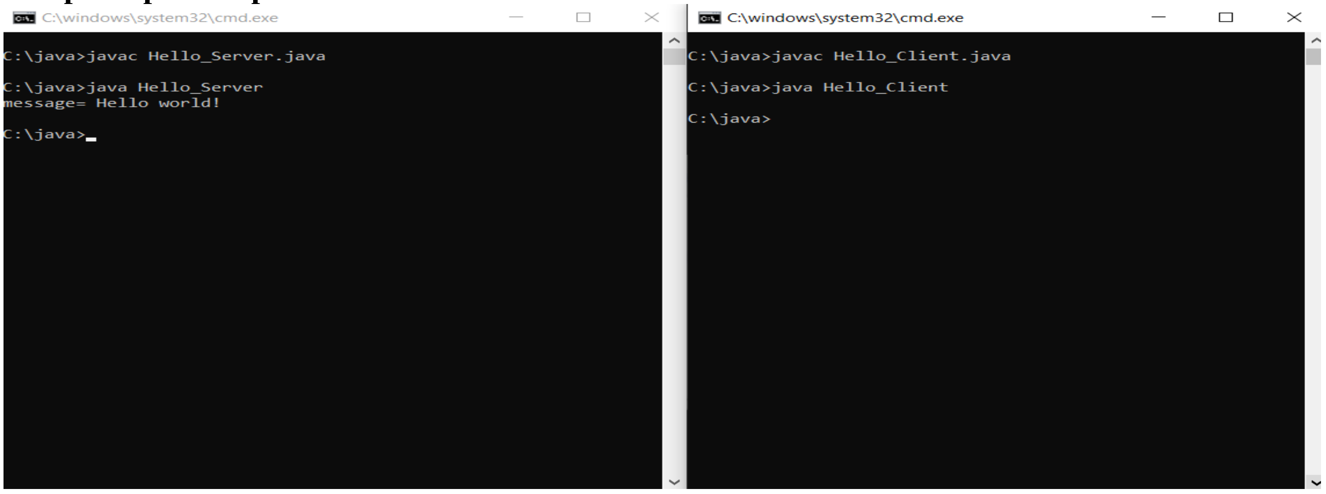
Program:
File: hello_server.java

```
import java.io.*;
import java.net.*;
public class Hello_Server
{
public static void main(String[] args)
{
try
{
ServerSocketss=new ServerSocket(6666);
Socket s=ss.accept();//establishes connection
DataInputStream dis=new DataInputStream(s.getInputStream()) ;
String  str=(String)dis.readUTF();
System.out.println("message= "+str);
ss.close();
}
catch(Exception e)
{
System.out.println(e);
}
}}
```

File: hello_client.java

```
import java.io.*;
import java.net.*;
public class Hello_Client
{
public static void main(String[] args)
{
try
{
Socket s=new Socket("localhost",6666);
DataOutputStreamdout=new DataOutputStream(s.getOutputStream());
dout.writeUTF("Hello world!");
dout.flush();
dout.close();
s.close();
}
catch(Exception e)
{
System.out.println(e);
}
}
}
```

Sample Input Output:



```
C:\java>javac Hello_Server.java

C:\java>java Hello_Server
message= Hello world!

C:\java>
```

```
C:\java>javac Hello_Client.java

C:\java>java Hello_Client

C:\java>
```

Practical 10

Aim: Write an Echo_Client and Echo_Server in java using TCP to estimate the round trip time from client to server. The server should be such that it can accept multiple connections at any given time.

Description:

Round trip time(RTT) is the length of time it takes for a signal to be sent plus the length of time it takes for an acknowledgement of that signal to be received. This time therefore consists of the propagation times between the two point of signal.

On the Internet, an end user can determine the RTT to and from an IP(Internet Protocol) address by pinging that address. The result depends on various factors :-

- The data rate transfer of the source’s internet connection.
- The nature of transmission medium.
- The physical distance between source and destination.
- The number of nodes between source and destination.
- The amount of traffic on the LAN(Local Area Network) to which end user is connected.
- The number of other requests being handled by intermediate nodes and the remote server.
- The speed with which intermediate node and the remote server function.
- The presence of Interference in the circuit.

Program:

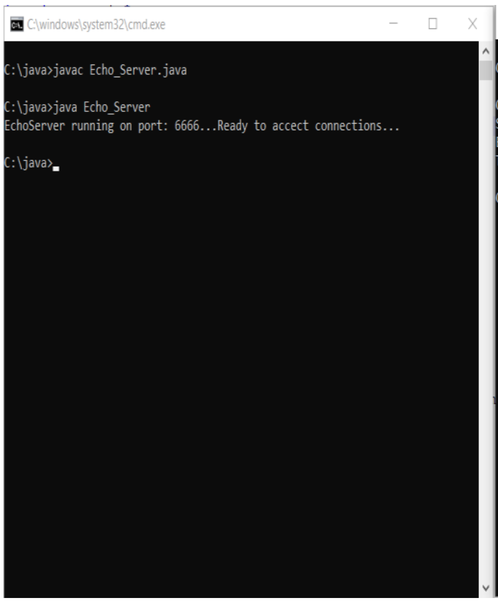
File 1- Echo_Server.java:

```
import java.io.*;
import java.net.*;
public class Echo_Server
{
public static void main(String[] args)
{
try
{
ServerSocketss=new ServerSocket(6666);
Socket s=ss.accept();//establishes connection
System.out.println("EchoServer running on port: 6666...Ready to accect connections...");
ss.close();
}
catch(Exception e)
{
System.out.println(e);
}
}
}
```

File 2- Echo_Client.java:

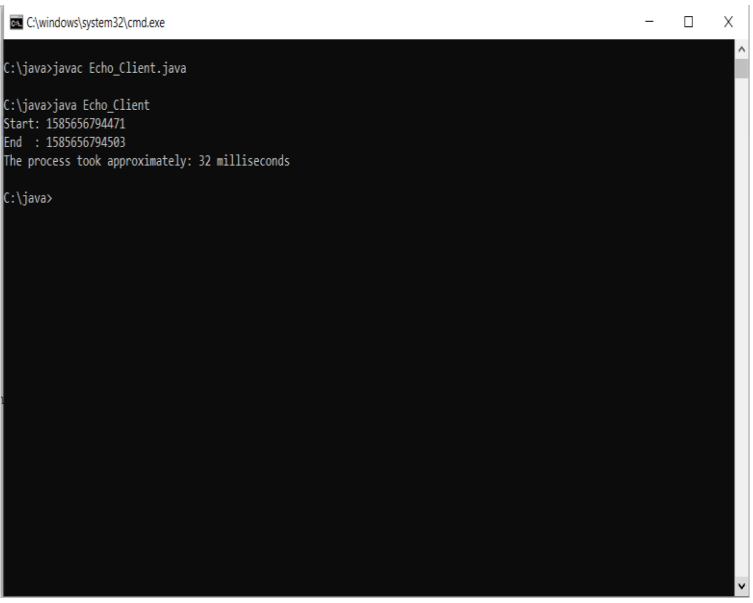
```
import java.io.*;
import java.net.*;
public class Echo_Client
{
public static void main(String[] args)
{
long start = System.currentTimeMillis();
System.out.println("Start: " + start);
try
{
Socket s=new Socket("localhost",6666);
s.close();
}
catch(Exception e)
{
System.out.println(e);
}
long end = System.currentTimeMillis();
System.out.println("End  : " + end);
long elapsedTime = end - start;
// Show how long it took to finish the process
System.out.println("The process took approximately: " + elapsedTime + " milliseconds");
}
}
```

Sample Input Output:



C:\windows\system32\cmd.exe

```
C:\java>javac Echo_Server.java

C:\java>java Echo_Server
EchoServer running on port: 6666...Ready to accect connections...

C:\java>
```



C:\windows\system32\cmd.exe

```
C:\java>javac Echo_Client.java

C:\java>java Echo_Client
Start: 1585656794471
End  : 1585656794503
The process took approximately: 32 milliseconds

C:\java>
```

Experiment No.7
UDP DATA TRANSFER
Practical 11 and 12

Aim: Practical 11 -WAP hello_client in Java where the server listens for, and accepts, a single UDP connection; it reads all the data it can from that connection, and prints it to the screen; then it closes the connection.

Practical 12 - WAP hello_server in Java where the client connects to the server, sends the string "Hello, world!", then closes the connection.

Description:

Java DatagramSocket and DatagramPacket classes are used for connection-less socket programming.

Java DatagramSocket class

Java DatagramSocket class represents a connection-less socket for sending and receiving datagram packets.

A datagram is basically an information but there is no guarantee of its content, arrival or arrival time.

Commonly used Constructors of DatagramSocket class

DatagramSocket() throws SocketEeption: it creates a datagram socket and binds it with the available Port Number on the localhost machine.

DatagramSocket(int port) throws SocketEeption: it creates a datagram socket and binds it with the given Port Number.

DatagramSocket(int port, InetAddress address) throws SocketEeption: it creates a datagram socket and binds it with the specified port number and host address.

Java DatagramPacket class

Java DatagramPacket is a message that can be sent or received. If you send multiple packet, it may arrive in any order. Additionally, packet delivery is not guaranteed.

Commonly used Constructors of DatagramPacket class

DatagramPacket(byte[] barr, int length): it creates a datagram packet. This constructor is used to receive the packets.

DatagramPacket(byte[] barr, int length, InetAddress address, int port): it creates a datagram packet. This constructor is used to send the packets.

Program:

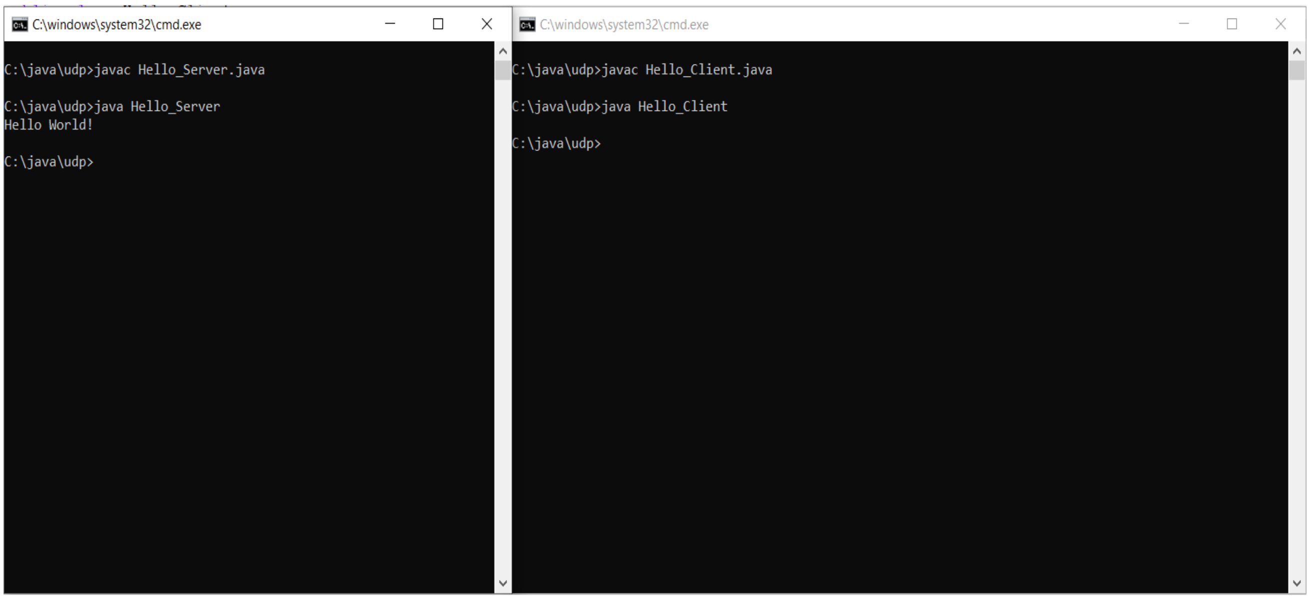
File: hello_server.java

```
import java.net.*;
public class hello_server
{
public static void main(String[] args) throws Exception
{
DatagramSocket ds = new DatagramSocket(3000);
byte[] buf = new byte[1024];
DatagramPacketdp = new DatagramPacket(buf, 1024);
ds.receive(dp);
String str = new String(dp.getData(), 0, dp.getLength());
System.out.println(str);
ds.close();
}
}
```

File: hello_client.java

```
import java.net.*;
public class hello_client
{
public static void main(String[] args) throws Exception
{
DatagramSocket ds = new DatagramSocket();
String str = "Hello World!";
InetAddressip = InetAddress.getByName("127.0.0.1");
DatagramPacketdp = new DatagramPacket(str.getBytes(), str.length(), ip, 3000);
ds.send(dp);
ds.close();
}
}
```

Sample Input Output:



```
C:\java\udp>javac Hello_Server.java

C:\java\udp>java Hello_Server
Hello World!

C:\java\udp>
```

```
C:\java\udp>javac Hello_Client.java

C:\java\udp>java Hello_Client

C:\java\udp>
```

Practical 13

Aim: WAP Echo Client and Echo Server in Java using UDP to estimate the round trip time from client to server. The server should be such that it can accept multiple connections at any given time.

Description:

Program:

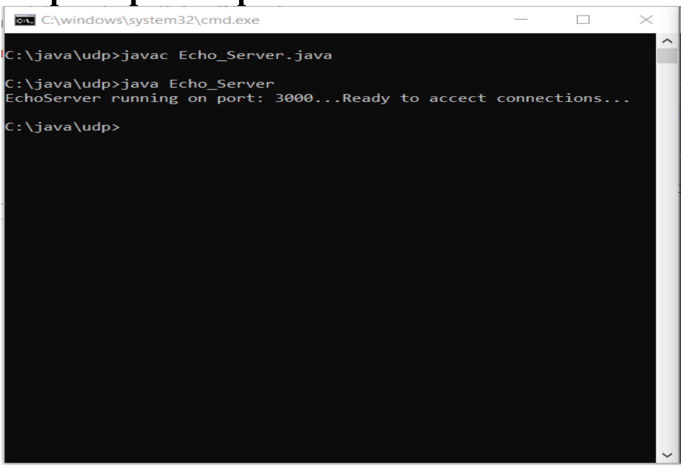
File: Echo_Server.java

```
import java.net.*;
public class Echo Server
{
public static void main(String[] args) throws Exception
{
DatagramSocket ds = new Datagram Socket(3000);
System.out.println("Echo Server running on port: 3000...Ready to accept connections...");
ds.close();
}
}
```

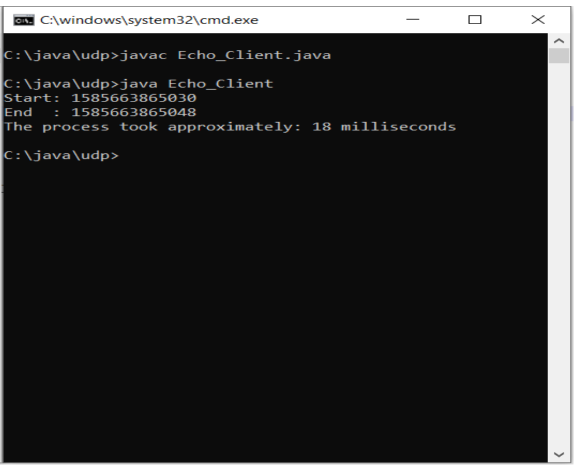
File: Echo_Client.java

```
import java.net.*;
public class Echo_Client
{
public static void main(String[] args) throws Exception
{
long start = System.currentTimeMillis();
System.out.println("Start: " + start);
DatagramSocket ds = new DatagramSocket();
InetAddressip = InetAddress.getByName("127.0.0.1");
long end = System.currentTimeMillis();
System.out.println("End  : " + end);
long elapsedTime = end - start;
// Show how long it took to finish the process
System.out.println("The process took approximately: " + elapsedTime + " milliseconds");
}
}
```

Sample Input Output:



```
C:\java\udp>javac Echo_Server.java

C:\java\udp>java Echo_Server
EchoServer running on port: 3000...Ready to accect connections...

C:\java\udp>
```



```
C:\java\udp>javac Echo_Client.java

C:\java\udp>java Echo_Client
Start: 1585663865030
End  : 1585663865048
The process took approximately: 18 milliseconds

C:\java\udp>
```

Experiment No.8
MULTIPLEXED I/O OPERATIONS
Practical 14

Aim: WAP in Java to implement client server communication using Multiplexed I/O Operations.

Description:

I/O Models

There are 5 I/O models -

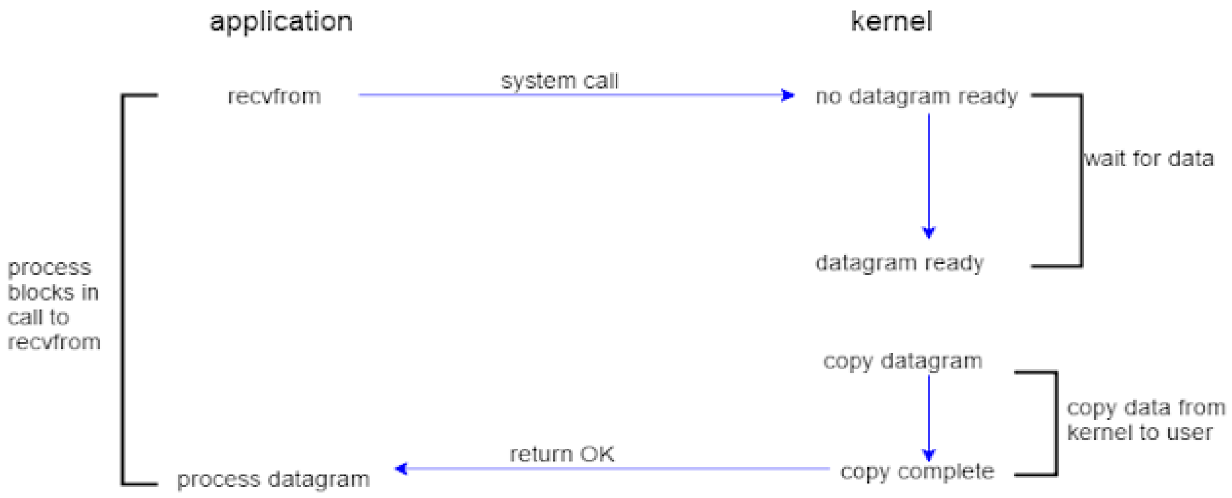
1. Blocking I/O
2. Non-blocking I/O
3. I/O multiplexing
4. Signal driven I/O
5. Asynchronous I/O

There are two phases for any input operation -

- Waiting for data to be ready
- Copying data from kernel to process

1. Blocking I/O Model

By default, all sockets are blocking.



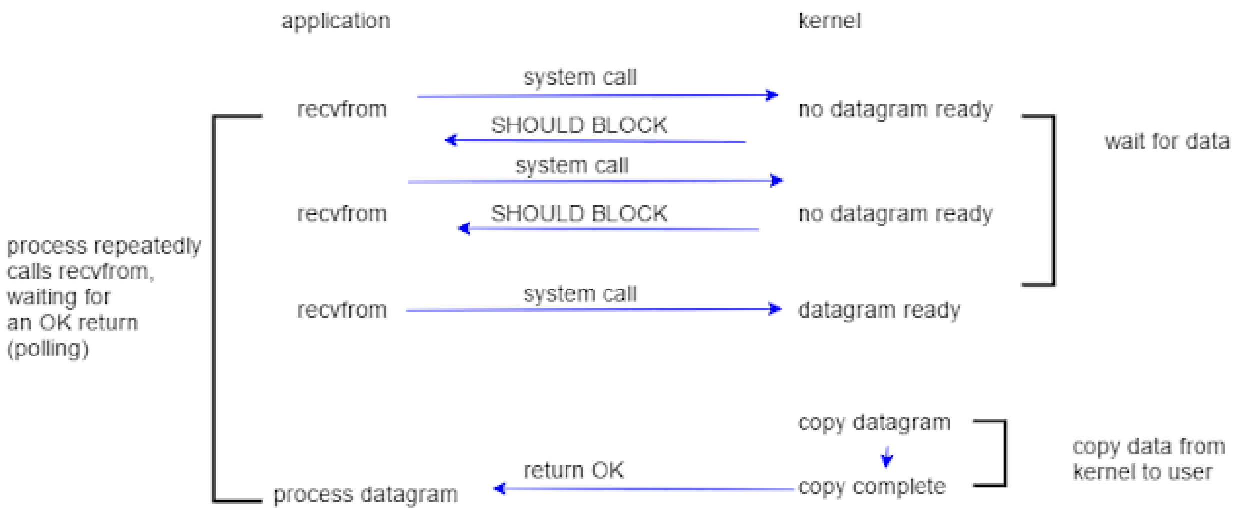
We use UDP in this diagram because it's easier, as we only have to deal with sending and receiving datagrams.

There is a switch from running in the application to running in the kernel, and returning back to application.

Most common error occurring is when system call is interrupted by a signal.

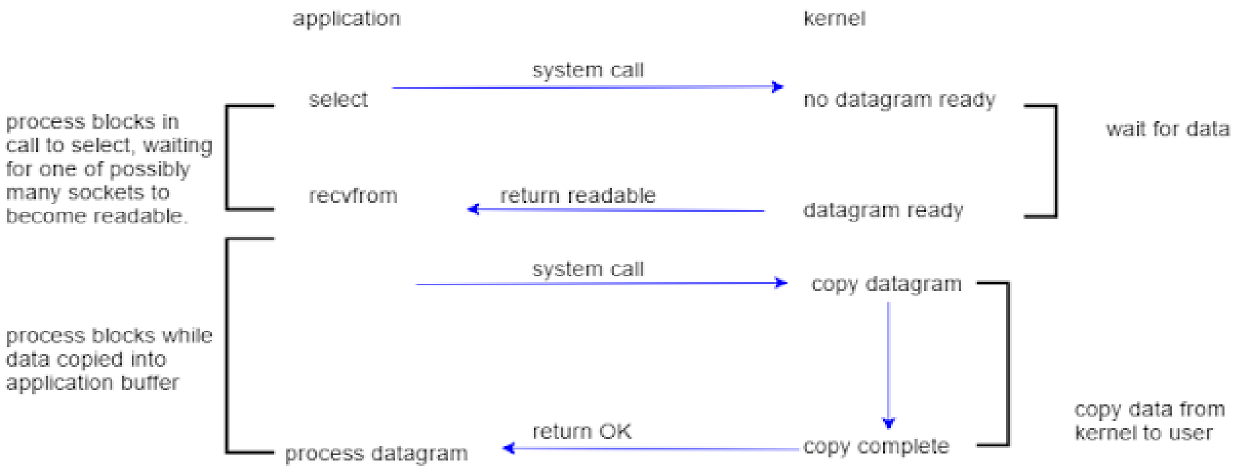
2. Non-blocking I/O Model

When we set a socket to non blocking, we tell the kernel that if an I/O request from me will put the process to sleep, return an error instead of blocking the process (putting the process to sleep). This method is called polling. It is often a waste of CPU time, but sometimes used on system dedicated to one function.



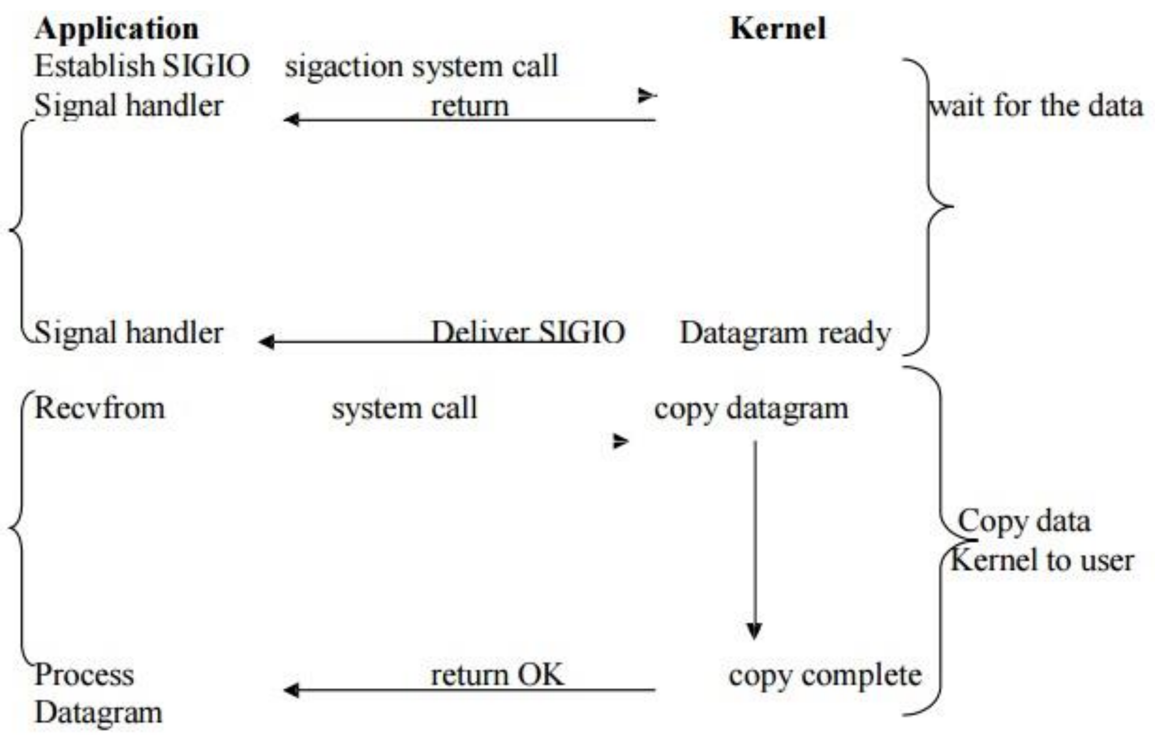
3. I/O Multiplexing
I/O multiplexing is the capability to tell the kernel that we want to be notified if one or more I/O conditions are ready, like input is ready to be read, or descriptor is capable of taking more output.
Scenarios is which I/O multiplexing is used -

- When client is handling multiple descriptors (like standard input and network socket).
- When client handles multiple sockets at the same time, example - Web client.
- When TCP server handles both listening and its connected sockets.
- When server handles both TCP and UDP.
- When server handles multiple services and perhaps multiple protocols.


application
kernel
system call
select
no datagram ready
process blocks in call to select, waiting for one of possibly many sockets to become readable.
wait for data
recvfrom
return readable
datagram ready
system call
copy datagram
process blocks while data copied into application buffer
copy data from kernel to user
return OK
process datagram
copy complete

Disadvantage - select requires two system calls instead of one.
Advantage - We can wait for more than one descriptor to be ready. Another alternative is using multithreading. Instead of using select to block on multiple file descriptors, the program uses multiple threads (one per file descriptor), and each thread is then free to call blocking system calls like recvfrom.

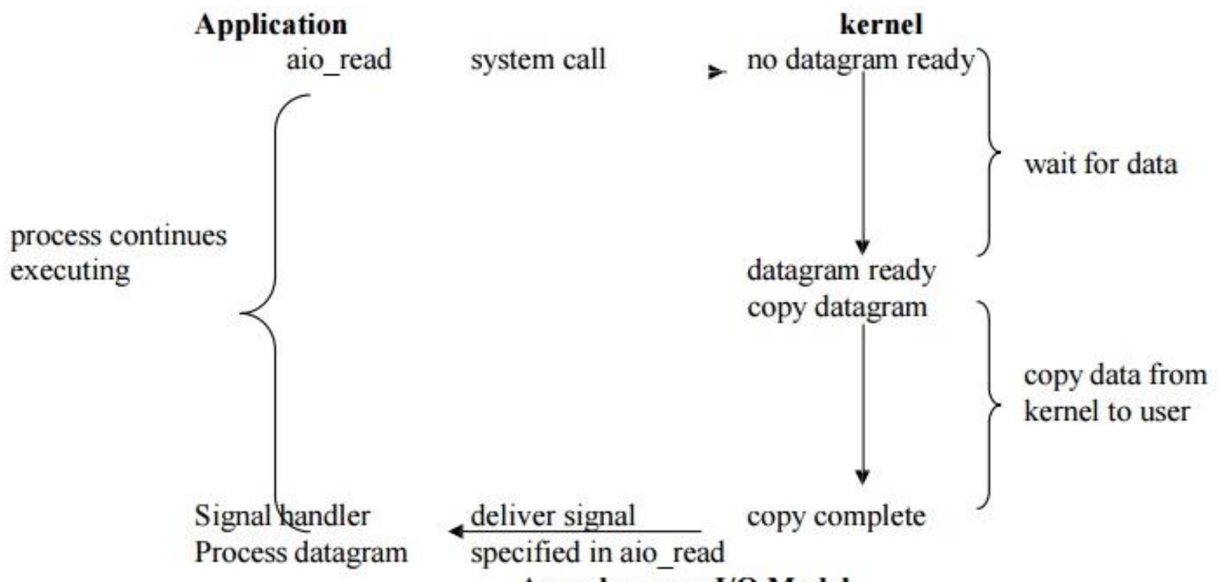
4. Signal-Driven I/O Model
The signal-driven I/O model uses signals, telling the kernel to notify us with the SIGIO signal when the descriptor is ready. The figure is below:



Signal Driven I/O Model

First enable the socket for the signal driven I/O and install a signal handler using the sigaction system call. The return from this system call is immediate and our process continuous, it is not block. When the datagram is ready to be ready, the SIGIO signal is generated for our process. We can either read the datagram from the signal handler by calling recvfrom and then notify the main loop that the data is ready to be process, or we can notify the main lop and let it read the datagram.
The advantage I this model is that we are not blocked while waiting for the datagram top arrive. The main loop can continue executing and just wait to be notified by the signal handler that either the data is read to process or that the datagram is ready to be read.

5. Asynchronous I/O Model
In this the kernel is told to start operation and to notify when the entire operation (including the copy of the data from the kernel to our buffer) is over.
The main difference between this and the signal driven I/O model in the previous section is that with the signal driven I/O the kernel tells when the I/O operation can be initiated. But with asynchronous I/O, the kernel tells us when an I/O operation is complete. It is summarized as given below:



Asynchronous I/O Model

The function **aio_read** is called and the descriptor, buffer pointer, buffer size, file offset and how to notify when the entire operation is complete are passed to the kernel. The system calls returns immediately and our process is not blocked waiting for the I./O to complete. It is expected that the kernel will generate some signal when the operation is complete. The signal is generated only when the data is copied completely in the application buffer.

Program:

File-1: m_server.java:

```
import java.io.IOException;
import java.net.InetSocketAddress;
import java.nio.ByteBuffer;
import java.nio.channels.SelectionKey;
import java.nio.channels.Selector;
import java.nio.channels.ServerSocketChannel;
import java.nio.channels.SocketChannel;
import java.util.Iterator;
import java.util.Set;
 public class m_server {
public static void main(String[] args) throws IOException {
// Selector: multiplexor of SelectableChannel objects
Selector selector = Selector.open(); // selector is open here
// ServerSocketChannel: selectable channel for stream-oriented listening sockets
ServerSocketChannelcrunchifySocket = ServerSocketChannel.open();
InetSocketAddresscrunchifyAddr = new InetSocketAddress("localhost", 1111);
// Binds the channel's socket to a local address and configures the socket to listen for connections
crunchifySocket.bind(crunchifyAddr);
// Adjusts this channel's blocking mode.
crunchifySocket.configureBlocking(false);
```

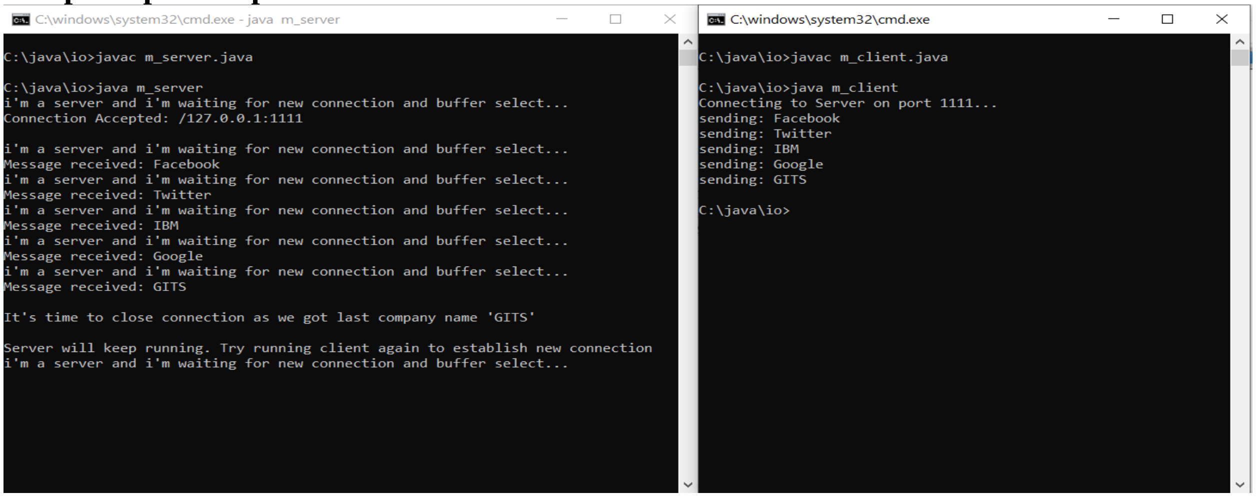
```
int ops = crunchifySocket.validOps();
SelectionKeyselectKy = crunchifySocket.register(selector, ops, null);
// Infinite loop..
// Keep server running
while (true) {
log("i'm a server and i'm waiting for new connection and buffer select...");
// Selects a set of keys whose corresponding channels are ready for I/O operations
selector.select();
// token representing the registration of a SelectableChannel with a Selector
Set<SelectionKey>crunchifyKeys = selector.selectedKeys();
Iterator<SelectionKey>crunchifyIterator = crunchifyKeys.iterator();
while (crunchifyIterator.hasNext()) {
SelectionKeymyKey = crunchifyIterator.next();
// Tests whether this key's channel is ready to accept a new socket connection
if (myKey.isAcceptable()) {
SocketChannelcrunchifyClient = crunchifySocket.accept();
// Adjusts this channel's blocking mode to false
crunchifyClient.configureBlocking(false);
// Operation-set bit for read operations
crunchifyClient.register(selector, SelectionKey.OP_READ);
log("Connection Accepted: " + crunchifyClient.getLocalAddress() + "\n");
// Tests whether this key's channel is ready for reading
} else if (myKey.isReadable()) {
SocketChannelcrunchifyClient = (SocketChannel) myKey.channel();
ByteBuffercrunchifyBuffer = ByteBuffer.allocate(256);
crunchifyClient.read(crunchifyBuffer);
String result = new String(crunchifyBuffer.array()).trim();
log("Message received: " + result);
if (result.equals("GITS")) {
crunchifyClient.close();
log("\nIt's time to close connection as we got last company name 'GITS'");
log("\nServer will keep running. Try running client again to establish new connection");
}
}
crunchifyIterator.remove();
}
}
}
private static void log(String str) {
System.out.println(str);
}
}
```

File-2: m_client.java:

```
import java.io.IOException;
import java.net.InetSocketAddress;
import java.nio.ByteBuffer;
import java.nio.channels.SocketChannel;
import java.util.ArrayList;
public class m_client {
public static void main(String[] args) throws IOException, InterruptedException {
InetSocketAddresscrunchifyAddr = new InetSocketAddress("localhost", 1111);
SocketChannelcrunchifyClient = SocketChannel.open(crunchifyAddr);
log("Connecting to Server on port 1111...");
ArrayList<String>companyDetails = new ArrayList<String>();
// create a ArrayList with companyName list
companyDetails.add("Facebook");
companyDetails.add("Twitter");
companyDetails.add("IBM");
companyDetails.add("Google");
companyDetails.add("GITS");
for (String companyName : companyDetails) {
byte[] message = new String(companyName).getBytes();
ByteBuffer buffer = ByteBuffer.wrap(message);
crunchifyClient.write(buffer);
log("sending: " + companyName);
buffer.clear();
// wait for 2 seconds before sending next message
Thread.sleep(2000);
}
crunchifyClient.close();
}
private static void log(String str) {
```

```
System.out.println(str);
}
}
```

Sample Input Output:


C:\windows\system32\cmd.exe - java m_server
C:\windows\system32\cmd.exe

```
C:\java\io>javac m_server.java

C:\java\io>java m_server
i'm a server and i'm waiting for new connection and buffer select...
Connection Accepted: /127.0.0.1:1111

i'm a server and i'm waiting for new connection and buffer select...
Message received: Facebook
i'm a server and i'm waiting for new connection and buffer select...
Message received: Twitter
i'm a server and i'm waiting for new connection and buffer select...
Message received: IBM
i'm a server and i'm waiting for new connection and buffer select...
Message received: Google
i'm a server and i'm waiting for new connection and buffer select...
Message received: GITS

It's time to close connection as we got last company name 'GITS'

Server will keep running. Try running client again to establish new connection
i'm a server and i'm waiting for new connection and buffer select...
```

```
C:\java\io>javac m_client.java

C:\java\io>java m_client
Connecting to Server on port 1111...
sending: Facebook
sending: Twitter
sending: IBM
sending: Google
sending: GITS

C:\java\io>
```

Experiment No.9
ROUTING ALGORITHMS
Practical 15

Aim: WAP in c to implement Distance Vector Routing Algorithm using Bellman-Ford's Algorithm.

Description:

A **distance-vector routing (DVR)** protocol requires that a router inform its neighbors of topology changes periodically. Historically known as the old ARPANET routing algorithm (or known as Bellman-Ford algorithm).

Bellman Ford Basics – Each router maintains a Distance Vector table containing the distance between itself and ALL possible destination nodes. Distances, based on a chosen metric, are computed using information from the neighbors' distance vectors.

Information kept by DV router -

Each router has an ID

Associated with each link connected to a router, there is a link cost .

Intermediate hops

Distance Vector Table Initialization -

Distance to itself = 0

Distance to ALL other routers = infinity number.

Distance Vector Algorithm –

1. A router transmits its distance vector to each of its neighbors in a routing packet.
2. Each router receives and saves the most recently received distance vector from each of its neighbors.
3. A router recalculates its distance vector when:
   - It receives a distance vector from a neighbor containing different information than before.
   - It discovers that a link to a neighbor has gone down.

The DV calculation is based on minimizing the cost to each destination

Dx(y) = Estimate of least cost from x to y

C(x,v) = Node x knows cost to each neighbor v

Dx = [Dx(y): y ∈ N] = Node x maintains distance vector

Node x also maintains its neighbors' distance vectors

– For each neighbor v, x maintains Dv = [Dv(y): y ∈ N]

Note –

- From time-to-time, each node sends its own distance vector estimate to neighbors.
- When a node x receives new DV estimate from any neighbor v, it saves v's distance vector and it updates its own DV using B-F equation:

Dx(y) = min { C(x,v) + Dv(y), Dx(y) } for each node y ∈ N

Program:

```
#include<stdlib.h>
#include<stdio.h>
#define nul 1000
#define nodes 10
int no;
struct node
{
int a[nodes][4];
}router[nodes];
void init(int r)
{
```

```
inti;
for(i=1;i<=no;i++)
{
router[r].a[i][1]=i;
router[r].a[i][2]=999;
router[r].a[i][3]=nul;
}
router[r].a[r][2]=0;
router[r].a[r][3]=r;
}
void inp(int r)
{
inti;
printf("\nEnterdist from the node %d to other nodes",r);
printf("\nPls enter 999 if there is no direct route\n",r);
for(i=1;i<=no;i++)
{
if(i!=r)
{
printf("\nEnterdist to the node %d:",i);
scanf("%d",&router[r].a[i][2]);
router[r].a[i][3]=i;
}
}
}
void display(int r)
{
inti,j;
printf("\n\nThe routing table for node %d is as follows:", r);
for(i=1;i<=no;i++)
{
if(router[r].a[i][2]>=999)
printf("\n\t\t\t %d \t no link \t no hop", router[r].a[i][1]);
else
printf("\n\t\t\t %d \t %d \t\t %d",router[r].a[i][1],router[r].a[i][2],router[r].a[i][3]);
}
}
void dv_algo(int r)
{
inti,j,z;
for(i=1;i<=no;i++)
{
        if(router[r].a[i][2]!=999 && router[r].a[i][2]<0)
        {
        printf("Negative weight cycle detected!\n");
          return;
        }

if(router[r].a[i][2]!=999 && router[r].a[i][2]!=0)
{
for(j=1;j<=no;j++)
{
z=router[r].a[i][2]+router[i].a[j][2];
if(router[r].a[j][2]>z)
{
router[r].a[j][2]=z;
router[r].a[j][3]=i;
}
}
}
}
}
int main()
{
inti,j,x,y;
char choice;
printf("Enter the no. of nodes required (less than 10 pls):");
scanf("%d",&no);
for(i=1;i<=no;i++)
{
init(i);
inp(i);
}
printf("\nThe configuration of the nodes after initialization is as follows:");
for(i=1;i<=no;i++)
display(i);
for(i=1;i<=no;i++)
dv_algo(i);
printf("\nThe configuration of the nodes after computation of paths is as follows:");
for(i=1;i<=no;i++)
{
display(i);
}
while(1)
{
printf("\n\nWant to continue (y/n):");
scanf(" %c",&choice);

if(choice=='n')
break;
printf("\nEnter the nodes between which shortest path is to be found:\n");
scanf("%d %d",&x,&y);
printf("\nThe length of the shortest path is %d",router[x].a[y][2]);
}
}
```

Sample Input Output:

```
Enter the no. of nodes required (less than 10 pls):5

Enter dist from the node 1 to other nodes
Pls enter 999 if there is no direct route

Enter dist to the node 2:10

Enter dist to the node 3:999

Enter dist to the node 4:5

Enter dist to the node 5:999

Enter dist from the node 2 to other nodes
```

```
Pls enter 999 if there is no direct route

Enter dist to the node 1:999

Enter dist to the node 3:1

Enter dist to the node 4:2

Enter dist to the node 5:999

Enter dist from the node 3 to other nodes
Pls enter 999 if there is no direct route

Enter dist to the node 1:999

Enter dist to the node 2:999

Enter dist to the node 4:999

Enter dist to the node 5:4

Enter dist from the node 4 to other nodes
Pls enter 999 if there is no direct route

Enter dist to the node 1:999

Enter dist to the node 2:3

Enter dist to the node 3:999

Enter dist to the node 5:2

Enter dist from the node 5 to other nodes
Pls enter 999 if there is no direct route

Enter dist to the node 1:7

Enter dist to the node 2:999

Enter dist to the node 3:6

Enter dist to the node 4:999

The configuration of the nodes after initialization is as follows:

The routing table for node 1 is as follows:
                    1      0            1
                    2      10            2
                    3      no link        no hop
                    4      5            4
                    5      no link        no hop

The routing table for node 2 is as follows:
                    1      no link        no hop
                    2      0            2
                    3      1            3
                    4      2            4
                    5      no link        no hop

The routing table for node 3 is as follows:
                    1      no link        no hop
                    2      no link        no hop
                    3      0            3
```

```
                         4       no link        no hop
                         5       4              5

The routing table for node 4 is as follows:
                         1       no link        no hop
                         2       3              2
                         3       no link        no hop
                         4       0              4
                         5       2              5

The routing table for node 5 is as follows:
                         1       7              1
                         2       no link        no hop
                         3       6              3
                         4       no link        no hop
                         5       0              5
The configuration of the nodes after computation of paths is as follows:

The routing table for node 1 is as follows:
                         1       0              1
                         2       8              4
                         3       11              2
                         4       5              4
                         5       7              4

The routing table for node 2 is as follows:
                         1       11              5
                         2       0              2
                         3       1              3
                         4       2              4
                         5       4              4

The routing table for node 3 is as follows:
                         1       11              5
                         2       no link        no hop
                         3       0              3
                         4       no link        no hop
                         5       4              5

The routing table for node 4 is as follows:
                         1       9              5
                         2       3              2
                         3       4              2
                         4       0              4
                         5       2              5

The routing table for node 5 is as follows:
                         1       7              1
                         2       15              1
                         3       6              3
                         4       12              1
                         5       0              5

Want to continue (y/n):y

Enter the nodes between which shortest path is to be found:
1 5
The length of the shortest path is 7

Want to continue (y/n):n
```

Experiment No.10

1-PRACTICALS BEYOND RTU SYLLABUS

Aim: WAP in check parity of data.

Description:Parity: Parity of a number refers to whether it contains an odd or even number of 1-bits. The number has “odd parity”, if it contains odd number of 1-bits and is “even parity” if it contains even number of 1-bits.

Algorithm: getParity(n)

```
1. Initialize parity = 0
2. Loop while n != 0
     a. Invert parity
          parity = !parity
     b. Unset rightmost set bit
          n = n & (n-1)
3. return parity
```

Example:

```
 Initialize: n = 13 (1101)   parity = 0

n = 13 & 12  = 12 (1100)   parity = 1
n = 12 & 11 = 8  (1000)   parity = 0
n = 8 & 7 = 0  (0000)    parity = 1
```

Program:

```
// C++ program to find parity
// of an integer
# include<bits/stdc++.h>
# define boolint
using namespace std;

// Function to get parity of number n. It returns 1
// if n has odd parity, and returns 0 if n has even
// parity
boolgetParity(unsigned int n)
{
    bool parity = 0;
    while (n)
    {
        parity = !parity;
        n     = n & (n - 1);
    }
    return parity;
}

/* Driver program to test getParity() */
int main()
{
    unsigned int n = 7;
    cout<<"Parity of no "<<n<<" = "<<(getParity(n)? "odd": "even");
    getchar();
    return 0;
}
```

Sample Input Output:

```
Parity of no 7 = odd
```

2-PRACTICALS BEYOND RTU SYLLABUS

Aim: Write a program to implement Dijkstra's Algorithm.

Description: Dijkstra algorithm is also called single source shortest path algorithm. It is based on greedy technique. The algorithm maintains a list visited [ ] of vertices, whose shortest distance from the source is already known.
If visited [1], equals 1, then the shortest distance of vertex i is already known. Initially, visited[i] is marked as, for source vertex.
At each step, we mark visited[v] as 1. Vertex v is a vertex at shortest distance from the source vertex. At each step of the algorithm, shortest distance of each vertex is stored in an array distance[ ].

Dijkstra's Algorithm

1. Create cost matrix C[ ][ ] from adjacency matrix adj[ ][ ]. C[i][j] is the cost of going from vertex i to vertex j. If there is no edge between vertices i and j then C[i][j] is infinity.

2. Array visited[ ] is initialized to zero.

```
for(i=0;i<n;i++)
        visited[i]=0;
```

3. If the vertex 0 is the source vertex then visited[0] is marked as 1.

4. Create the distance matrix, by storing the cost of vertices from vertex no. 0 to n-1 from the source vertex 0.

```
for(i=1;i<n;i++)
        distance[i]=cost[0][i];
```

Initially, distance of source vertex is taken as 0. i.e. distance[0]=0;

5. for(i=1;i<n;i++)

– Choose a vertex w, such that distance[w] is minimum and visited[w] is 0. Mark visited[w] as 1.
– Recalculate the shortest distance of remaining vertices from the source.
– Only, the vertices not marked as 1 in array visited[ ] should be considered for recalculation of distance. i.e. for each vertex v

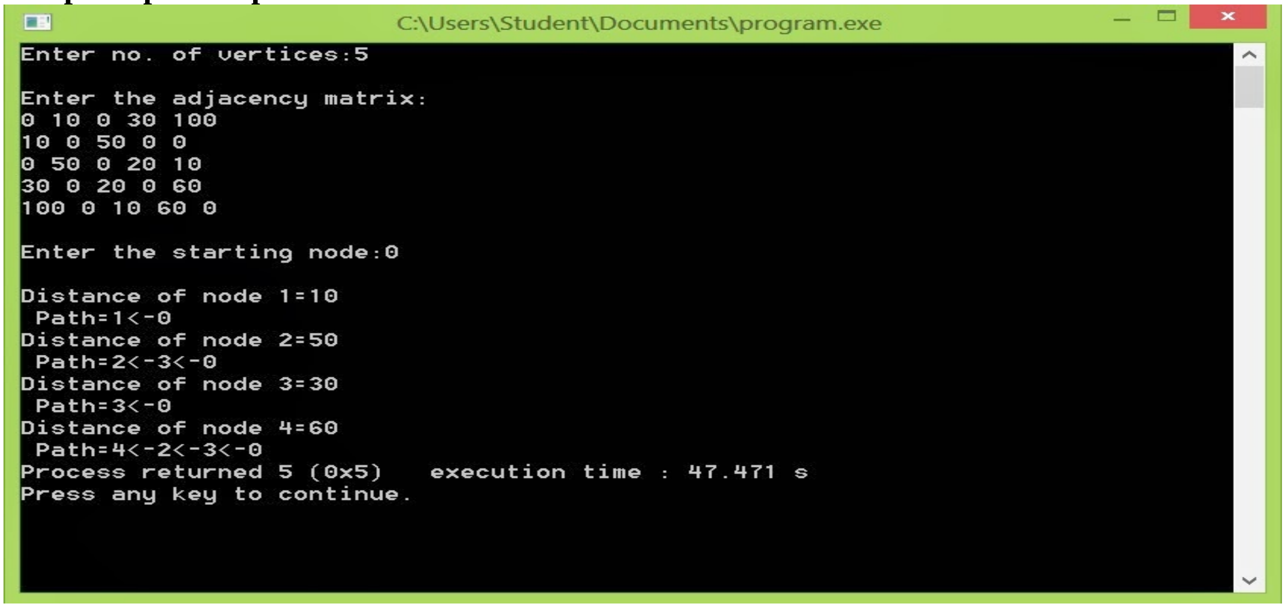
```
if(visited[v]==0)
        distance[v]=min(distance[v],
        distance[w]+cost[w][v])
```

Program:

```
#include<stdio.h>
#include<conio.h>
#define INFINITY 9999
#define MAX 10

void dijkstra(int G[MAX][MAX],intn,intstartnode);
 int main()
{
        int G[MAX][MAX],i,j,n,u;
        printf("Enter no. of vertices:");
        scanf("%d",&n);
        printf("\nEnter the adjacency matrix:\n");

        for(i=0;i<n;i++)
                for(j=0;j<n;j++)
                        scanf("%d",&G[i][j]);
	printf("\nEnter the starting node:");
	scanf("%d",&u);
	dijkstra(G,n,u);

	return 0;
}

void dijkstra(int G[MAX][MAX],intn,intstartnode)
{

	int cost[MAX][MAX],distance[MAX],pred[MAX];
	int visited[MAX],count,mindistance,nextnode,i,j;
	//pred[] stores the predecessor of each node
	//count gives the number of nodes seen so far
	//create the cost matrix
	for(i=0;i<n;i++)
		for(j=0;j<n;j++)
			if(G[i][j]==0)
				cost[i][j]=INFINITY;
			else
				cost[i][j]=G[i][j];
	s
	//initialize pred[],distance[] and visited[]
	for(i=0;i<n;i++)
	{
		distance[i]=cost[startnode][i];
		pred[i]=startnode;
		visited[i]=0;
	}

	distance[startnode]=0;
	visited[startnode]=1;
	count=1;

	while(count<n-1)
	{
		mindistance=INFINITY;

		//nextnode gives the node at minimum distance
		for(i=0;i<n;i++)
			if(distance[i]<mindistance&&!visited[i])
			{
				mindistance=distance[i];
				nextnode=i;
			}
		//check if a better path exists through nextnode
			visited[nextnode]=1;
			for(i=0;i<n;i++)
				if(!visited[i])
					if(mindistance+cost[nextnode][i]<distance[i])
					{
						distance[i]=mindistance+cost[nextnode][i];
						pred[i]=nextnode;
					}
```

```
                count++;
        }
        //print the path and distance of each node
        for(i=0;i<n;i++)
                if(i!=startnode)
                {
                        printf("\nDistance of node%d=%d",i,distance[i]);
                        printf("\nPath=%d",i);

                        j=i;
                        do
                        {
                                j=pred[j];
                                printf("<-%d",j);
                        }while(j!=startnode);
        }
}
```

Sample Input Output:



C:\Users\Student\Documents\program.exe

```
Enter no. of vertices:5

Enter the adjacency matrix:
0 10 0 30 100
10 0 50 0 0
0 50 0 20 10
30 0 20 0 60
100 0 10 60 0

Enter the starting node:0

Distance of node 1=10
 Path=1<-0
Distance of node 2=50
 Path=2<-3<-0
Distance of node 3=30
 Path=3<-0
Distance of node 4=60
 Path=4<-2<-3<-0
Process returned 5 (0x5)   execution time : 47.471 s
Press any key to continue.
```

RUBRICS EVALUATION

Performance Criteria	Scale 1 (0-25%)	Scale 2 (26-50%)	Scale 3 (51-75%)	Scale 4 (76-100%)	Score (Numerical)
Understandability Ability to analyse Problem and Identify solution	Unable to understand the problem.	Able to understand the problem partially and unable to identify the solution	Able to understand the problem completely but unable to identify the solution	Able to understand the problem completely and able to provide alternative solution too.	
Logic Ability to specify Conditions & control flow that are appropriate for the problem domain.	Program logic is incorrect	Program logic is on the right track but has several errors	Program logic is mostly correct, but may contain an occasional boundary error or redundant or contradictory condition.	Program logic is correct, with no known boundary errors, and no redundant or contradictory conditions.	
Debugging Ability to execute /debug	Unable to execute program	Unable to debug several errors.	Able to execute program with several warnings.	Able to execute program completely	
Correctness Ability to code formulae and algorithms that reliably produce correct answers or appropriate results.	Program does not produce correct answers or appropriate results for most inputs.	Program approaches correct answers or appropriate results for most inputs, but can contain miscalculations in some cases.	Program produces correct answers or appropriate results for most inputs.	Program produces correct answers or appropriate results for all inputs tested.	
Completeness Ability to demonstrate and deliver on time.	Unable to explain the code.and the code was overdue.	Unable to explain the code and the code submission was late.	Able to explain code and the program was delivered within the due date.	Able to explain code and the program was delivered on time.	
				TOTAL	