<u>**BHARTIYA INSTITUTE OF ENGINEERING & TECHNOLOGY, SIKAR**</u>

Network Programming Lab Manual (4CS4-23)
Credit: 1.5 Max. Marks: 75(IA: 45, ETE: 30)

Prepared by: Dileep Kumar Agarwal

List of Experiments:

1. Study of Different Type of LAN& Network Equipments.

2. Study and Verification of standard Network topologies i.e. Star, Bus, Ring etc.

3. LAN installations and Configurations.

4. Write a program to implement various types of error correcting techniques.

5. Write a program to implement various types of farming methods.

6. Write two programs in C: hello_client and hello_server

a. 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

b. The client connects to the server, sends the string “Hello, world!”, then

closes the connection

7. Write an Echo_Client and Echo_server using TCP to estimate the round trip

time from client to the server. The server should be such that it can accept

multiple connections at any given time.

8. Repeat Exercises 6 & 7 for UDP.

9. Repeat Exercise 7 with multiplexed I/O operations.

10. Simulate Bellman-Ford Routing algorithm in NS2.

1. Study of Different Type of LAN& Network Equipments.

What is LAN?

A **L**ocal **A**rea **N**etwork (LAN) is a group of computer and peripheral devices which are connected in a limited area such as school, laboratory, home, and office building. It is a widely useful network for sharing resources like files, printers, games, and other application. The simplest type of LAN network is to connect computers and a printer in someone's home or office. In general, LAN will be used as one type of transmission medium.

It is a network which consists of less than 5000 interconnected devices across several buildings.

Characteristics of LAN

Here are important characteristics of a LAN network:

- It is a private network, so an outside regulatory body never controls it.
- LAN operates at a relatively higher speed compared to other WAN systems.
- There are various kinds of media access control methods like token ring and ethernet.

Advantages of LAN

Here are pros/benefits of using LAN:

- Computer resources like hard-disks, DVD-ROM, and printers can share local area networks. This significantly reduces the cost of hardware purchases.
- You can use the same software over the network instead of purchasing the licensed software for each client in the network.
- Data of all network users can be stored on a single hard disk of the server computer.
- You can easily transfer data and messages over networked computers.
- It will be easy to manage data at only one place, which makes data more secure.
- Local Area Network offers the facility to share a single internet connection among all the LAN users.

Disadvantages of LAN

Here are the important cons/ drawbacks of LAN:

- LAN will indeed save cost because of shared computer resources, but the initial cost of installing Local Area Networks is quite high.

- The LAN admin can check personal data files of every LAN user, so it does not offer good privacy.
- Unauthorized users can access critical data of an organization in case LAN admin is not able to secure centralized data repository.
- Local Area Network requires a constant LAN administration as there are issues related to software setup and hardware failures

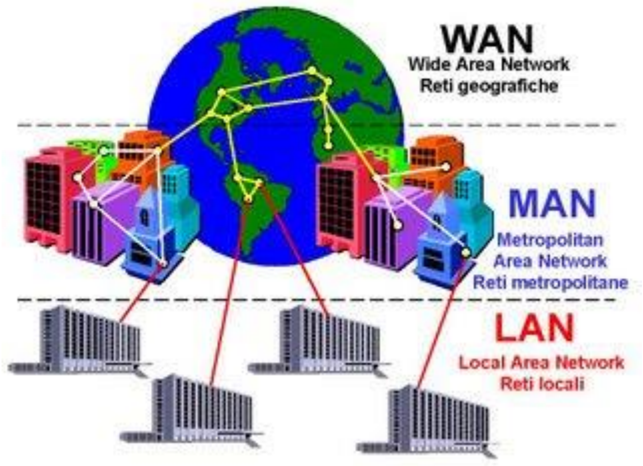
The following types of network equipment:

- **Hubs** provide a central location for attaching wires to workstations. There are two types: passive and active.
- **Switches** connect devices to host computers and allow large numbers of these devices to share a limited number of ports.
- **Routers** are protocol-dependent devices that connect sub-networks or divide a very large network into smaller sub-networks.
- **Repeaters** use regeneration and retiming to ensure that signals are transmitted clearly through all network segments.
- **Bridges** are used to interconnect local or remote networks. They centralize network administration.
- **Gateways** can interconnect networks with different, incompatible communications protocols.
- **Multiplexers** combine multiple signal inputs into one output.
- **Transceivers** connect nodes and send and receive signals. They are sometimes called medium access units (MAU).
- **Firewalls** safeguard a network against unauthorized access.

Other network devices such as wireless access points (WAP) and modular platforms are also available.

Product and Performance Specifications

Network equipment may be designed for local area networks (LAN), metropolitan area networks (MAN), or wide area networks (WAN).



Processor type, speed, and computer memory are other important product parameters. Form factors include chips, boards or cards, and stand-alone or enclosed modules. Performance specifications include data rate and operating temperature, the number of users and concurrent connections that devices can support, and the total number of media access control (MAC) addressed that can be stored.

Features and Applications

When selecting network equipment, buyers may need to consider whether a device is power over Ethernet (PoE) ready, or if it supports voice-over-Internet protocol (VoIP). Devices with full duplex capabilities can transmit data simultaneously in both directions, and may be stackable or rack-mountable. Product features such as alarms and LED indicators provide audible and visual notifications to network administrators.

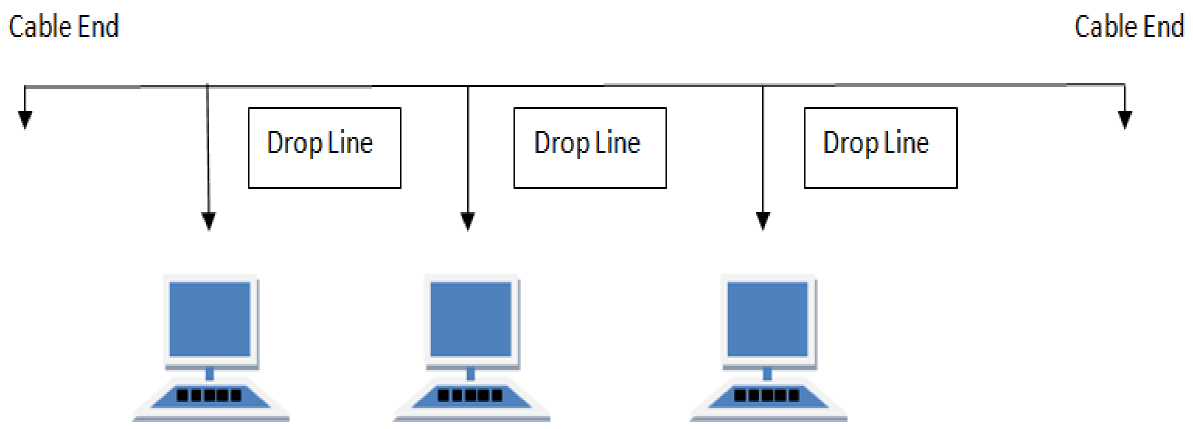
Network equipment may be designed or suitable for particular applications. For example, hardened products are often used in telecommunications applications. Their casings provide protection from weather-related conditions and can act as a heat sink, directing high temperatures away from sensitive components.

2. Study and Verification of standard Network topologies: Star, Bus, Ring etc.

Network Topology is the schematic description of a network arrangement, connecting various nodes(sender and receiver) through lines of connection.

BUS Topology

Bus topology is a network type in which every computer and network device is connected to single cable. When it has exactly two endpoints, then it is called **Linear Bus topology**.



Features of Bus Topology

1. It transmits data only in one direction.
2. Every device is connected to a single cable

Advantages of Bus Topology

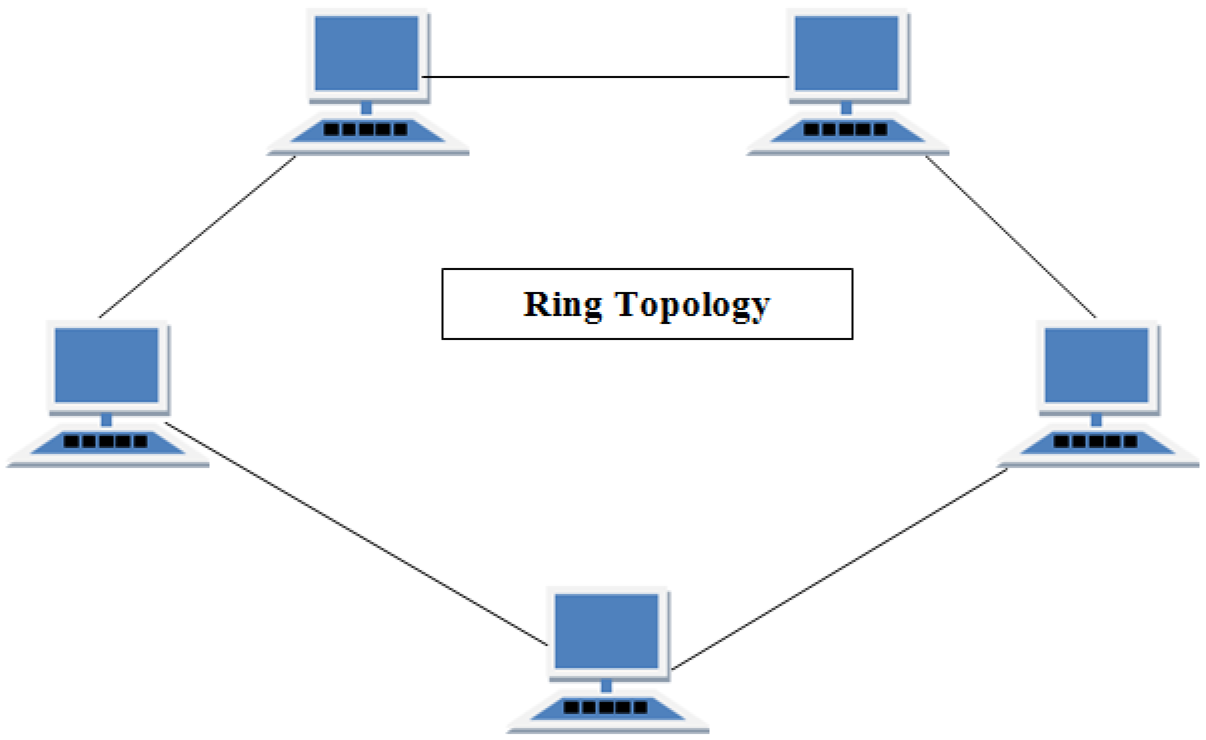
1. It is cost effective.
2. Cable required is least compared to other network topology.
3. Used in small networks.
4. It is easy to understand.
5. Easy to expand joining two cables together.

Disadvantages of Bus Topology

1. Cables fails then whole network fails.
2. If network traffic is heavy or nodes are more the performance of the network decreases.
3. Cable has a limited length.
4. It is slower than the ring topology.

RING Topology

It is called ring topology because it forms a ring as each computer is connected to another computer, with the last one connected to the first. Exactly two neighbours for each device.



Features of Ring Topology

1. A number of repeaters are used for Ring topology with large number of nodes, because if someone wants to send some data to the last node in the ring topology with 100 nodes, then the data will have to pass through 99 nodes to reach the 100th node. Hence to prevent data loss repeaters are used in the network.
2. The transmission is unidirectional, but it can be made bidirectional by having 2 connections between each Network Node, it is called **Dual Ring Topology**.
3. In Dual Ring Topology, two ring networks are formed, and data flow is in opposite direction in them. Also, if one ring fails, the second ring can act as a backup, to keep the network up.
4. Data is transferred in a sequential manner that is bit by bit. Data transmitted, has to pass through each node of the network, till the destination node.

Advantages of Ring Topology

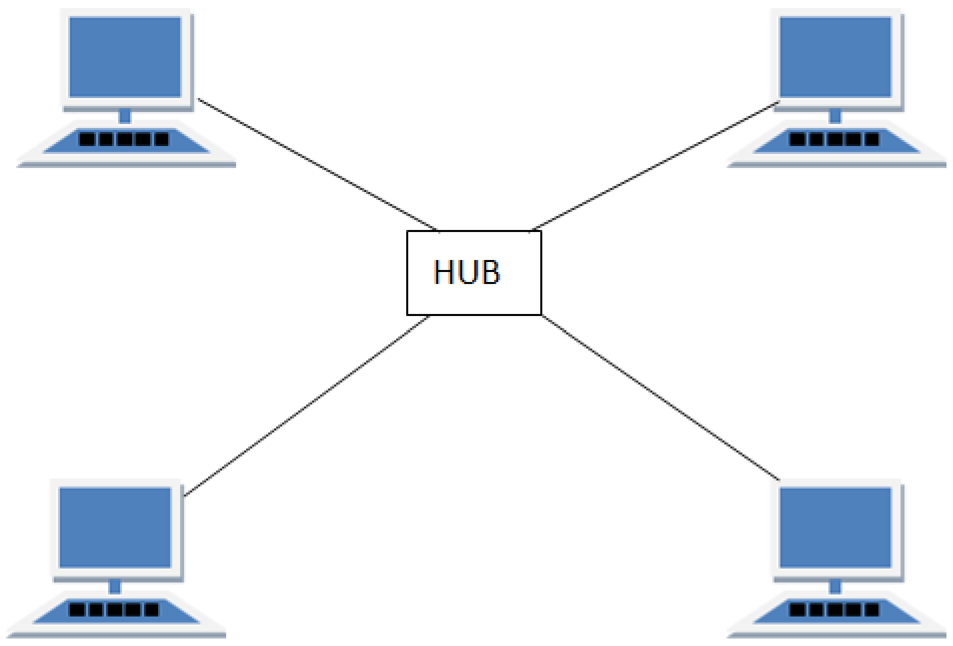
1. Transmitting network is not affected by high traffic or by adding more nodes, as only the nodes having tokens can transmit data.
2. Cheap to install and expand

Disadvantages of Ring Topology

1. Troubleshooting is difficult in ring topology.
2. Adding or deleting the computers disturbs the network activity.
3. Failure of one computer disturbs the whole network.

STAR Topology

In this type of topology all the computers 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.



Features of Star Topology

1. Every node has its own dedicated connection to the hub.
2. Hub acts as a repeater for data flow.

3. Can be used with twisted pair, Optical Fibre or coaxial cable.

Advantages of Star Topology

1. Fast performance with few nodes and low network traffic.
2. Hub can be upgraded easily.
3. Easy to troubleshoot.
4. Easy to setup and modify.
5. Only that node is affected which has failed, rest of the nodes can work smoothly.

Disadvantages of Star Topology

1. Cost of installation is high.
2. Expensive to use.
3. If the hub fails then the whole network is stopped because all the nodes depend on the hub.
4. Performance is based on the hub that is it depends on its capacity

MESH Topology

It is a point-to-point connection to other nodes or devices. All the network nodes are connected to each other. Mesh has $n(n-1)/2$ physical channels to link n devices.

There are two techniques to transmit data over the Mesh topology, they are :

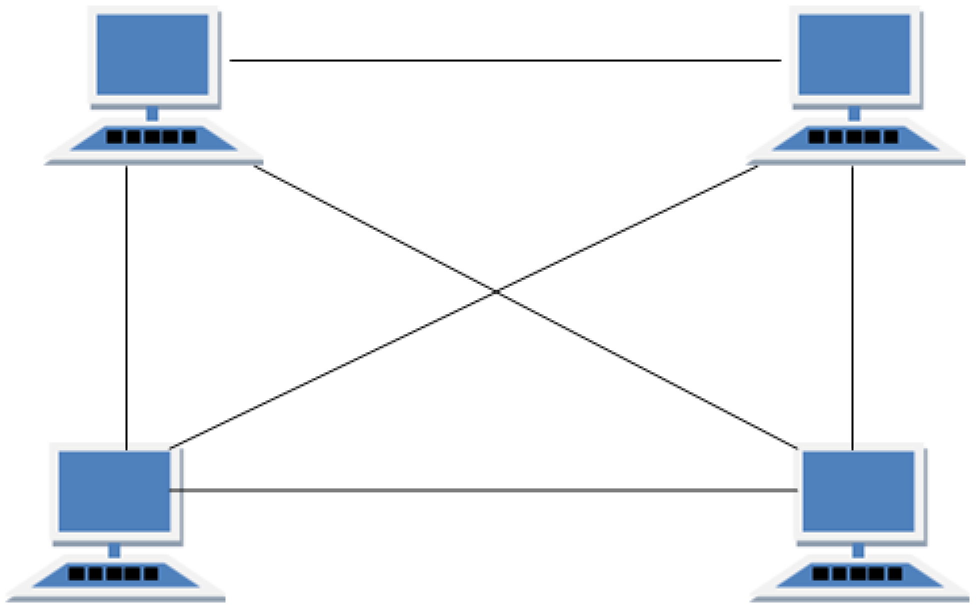
1. Routing
2. Flooding

MESH Topology: Routing

In routing, the nodes have a routing logic, as per the network requirements. Like routing logic to direct the data to reach the destination using the shortest distance. Or, routing logic which has information about the broken links, and it avoids those node etc. We can even have routing logic, to re-configure the failed nodes.

MESH Topology: Flooding

In flooding, the same data is transmitted to all the network nodes, hence no routing logic is required. The network is robust, and the its very unlikely to lose the data. But it leads to unwanted load over the network.



Types of Mesh Topology

1. **Partial Mesh Topology :** In this topology some of the systems are connected in the same fashion as mesh topology but some devices are only connected to two or three devices.
2. **Full Mesh Topology :** Each and every nodes or devices are connected to each other.

Features of Mesh Topology

1. Fully connected.
2. Robust.
3. Not flexible.

Advantages of Mesh Topology

1. Each connection can carry its own data load.
2. It is robust.

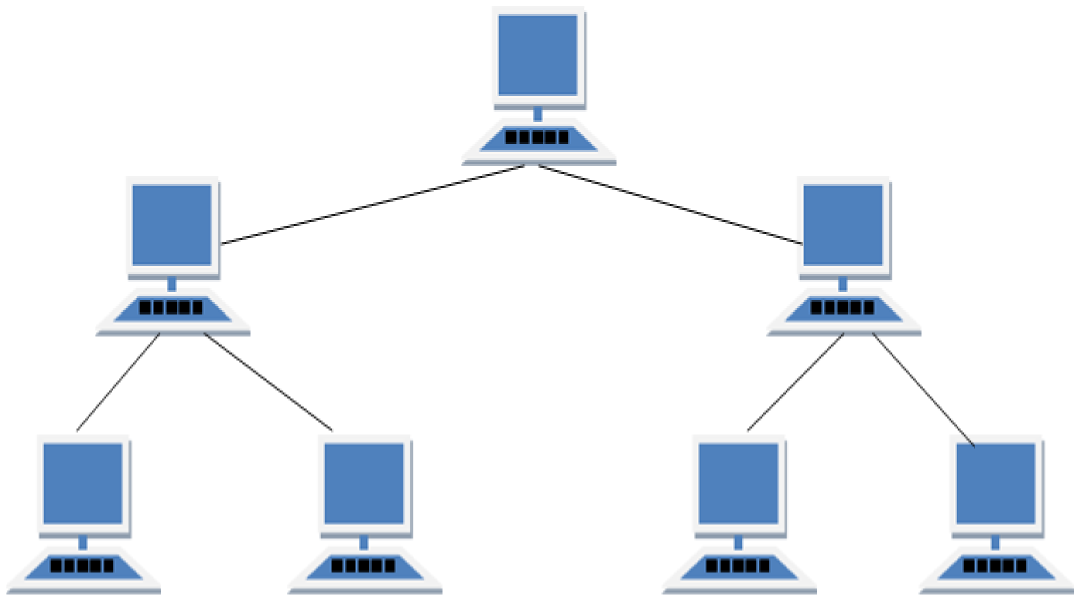
3. Fault is diagnosed easily.
4. Provides security and privacy.

Disadvantages of Mesh Topology

1. Installation and configuration is difficult.
2. Cabling cost is more.
3. Bulk wiring is required.

TREE Topology

It has a root node and all other nodes are connected to it forming a hierarchy. It is also called hierarchical topology. It should at least have three levels to the hierarchy.



Features of Tree Topology

1. Ideal if workstations are located in groups.
2. Used in Wide Area Network.

Advantages of Tree Topology

1. Extension of bus and star topologies.

2. Expansion of nodes is possible and easy.
3. Easily managed and maintained.
4. Error detection is easily done.

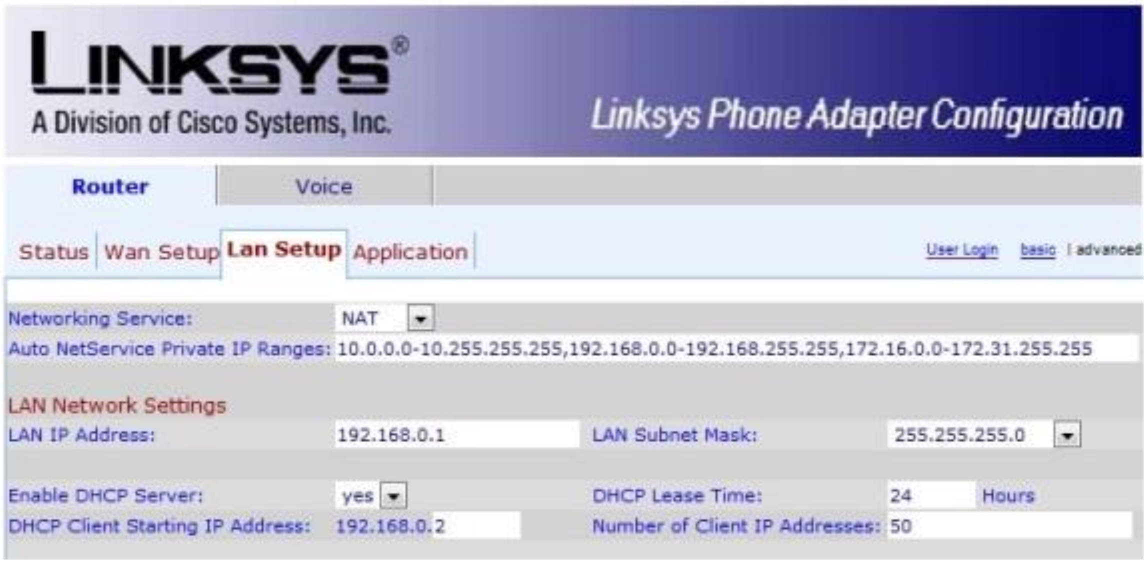
Disadvantages of Tree Topology

1. Heavily cabled.
2. Costly.
3. If more nodes are added maintenance is difficult.
4. Central hub fails, network fails.

3. LAN installations and Configurations.

Configuration of LAN Network Settings

Step 1. Log in to the web configuration utility and choose **Router > Advanced > Lan Setup**.



Networking Service: NAT
Auto NetService Private IP Ranges: 10.0.0.0-10.255.255.255,192.168.0.0-192.168.255.255,172.16.0.0-172.31.255.255

LAN Network Settings
LAN IP Address: 192.168.0.1
LAN Subnet Mask: 255.255.255.0
Enable DHCP Server: yes
DHCP Lease Time: 24 Hours
DHCP Client Starting IP Address: 192.168.0.2
Number of Client IP Addresses: 50

Step 2. In the LAN IP Address field, enter the IP address of the SPA3102 on the local network.

Step 3. From the LAN Subnet Mask drop-down list, choose the subnet that correlates to the LAN IP Address in Step 2.

Step 4. From the Enable DHCP Server drop-down list, choose **Yes** to enable the dynamic host configuration protocol which will dynamically assign IP address to devices on the LAN, or choose **No** to have devices given static IP addresses.
Note: If you choose **No**, skip to Step 8.

Step 5. In the DHCP Lease Time field, enter the number, in hours, that you wish an IP address to be assigned to a device before being renewed.

Step 6. In the DHCP Client Starting IP Address field, enter the beginning IP address of the client devices that will be assigned a dynamic IP address.

Step 7. In the Number of Client IP Addresses field, enter the number of client devices that you wish to be able to connect to the local network.

Step 8. Click **Save Settings** to save the settings or click **Cancel Settings** to discard the settings.

4. Write a program to implement various types of error correcting techniques.

```
#include<stdio.h>

void main() {
   int data[10];
   int dataatrec[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]);
                        }
            }
}
```

5. Write a program to implement various types of farming methods.

```
#include<stdio.h>

#include<conio.h>

#include<string.h>

void main()

{

    int a[20],b[30],i,j,k,count,n;

    printf("Enter frame size (Example: 8):");

    scanf("%d",&n);
```

```
printf("Enter the frame in the form of 0 and 1 :");
for(i=0; i<n; i++)
    scanf("%d",&a[i]);
i=0;
count=1;
j=0;
while(i<n)
{
    if(a[i]==1)
    {
        b[j]=a[i];
        for(k=i+1; a[k]==1 && k<n && count<5; k++)
        {
            j++;
            b[j]=a[k];
            count++;
            if(count==5)
            {
                j++;
                b[j]=0;
            }
            i=k;
        }
    }
    else
```

```
        {
            b[j]=a[i];
        }
        i++;
        j++;
    }
    printf("After Bit Stuffing :");
    for(i=0; i<j; i++)
        printf("%d",b[i]);
    getch();
}
```

```
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
```

6. Write two programs in C: hello_client and hello_server

Client Code

```
/****************** CLIENT CODE ****************/


#include <stdio.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>


int main(){

  int clientSocket;

  char buffer[1024];
struct sockaddr_in serverAddr;

socklen_t addr_size;


/*---- Create the socket. The three arguments are: ----*/

/* 1) Internet domain 2) Stream socket 3) Default protocol (TCP in this case) */

clientSocket = socket(PF_INET, SOCK_STREAM, 0);


/*---- Configure settings of the server address struct ----*/

/* Address family = Internet */

serverAddr.sin_family = AF_INET;

/* Set port number, using htons function to use proper byte order */

serverAddr.sin_port = htons(7891);

/* Set IP address to localhost */

serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");

/* Set all bits of the padding field to 0 */

memset(serverAddr.sin_zero, '\0', sizeof serverAddr.sin_zero);


/*---- Connect the socket to the server using the address struct ----*/

addr_size = sizeof serverAddr;

connect(clientSocket, (struct sockaddr *) &serverAddr, addr_size);


/*---- Read the message from the server into the buffer ----*/

recv(clientSocket, buffer, 1024, 0);


/*---- Print the received message ----*/

printf("Data received: %s",buffer);


return 0;
```

```
}
```

Server Code

```
/****************** SERVER CODE ****************/


#include <stdio.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>


int main(){

  int welcomeSocket, newSocket;

  char buffer[1024];

  struct sockaddr_in serverAddr;

  struct sockaddr_storage serverStorage;

  socklen_t addr_size;


  /*---- Create the socket. The three arguments are: ----*/

  /* 1) Internet domain 2) Stream socket 3) Default protocol (TCP in this case) */

  welcomeSocket = socket(PF_INET, SOCK_STREAM, 0);


  /*---- Configure settings of the server address struct ----*/

  /* Address family = Internet */

  serverAddr.sin_family = AF_INET;

  /* Set port number, using htons function to use proper byte order */

  serverAddr.sin_port = htons(7891);

  /* Set IP address to localhost */

  serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
  /* Set all bits of the padding field to 0 */
  memset(serverAddr.sin_zero, '\0', sizeof serverAddr.sin_zero);

  /*---- Bind the address struct to the socket ----*/
  bind(welcomeSocket, (struct sockaddr *) &serverAddr, sizeof(serverAddr));

  /*---- Listen on the socket, with 5 max connection requests queued ----*/
  if(listen(welcomeSocket,5)==0)
    printf("Listening\n");
  else
    printf("Error\n");

  /*---- Accept call creates a new socket for the incoming connection ----*/
  addr_size = sizeof serverStorage;
  newSocket = accept(welcomeSocket, (struct sockaddr *) &serverStorage, &addr_size);

  /*---- Send message to the socket of the incoming connection ----*/
  strcpy(buffer,"Hello World\n");
  send(newSocket,buffer,13,0);

  return 0;
}
```

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

Echo Client

```
#include <sys/types.h>
```

```
#include <sys/socket.h>
#include <netdb.h>
#include <stdio.h>
#include<string.h>

int main(int argc,char **argv)
{
    int sockfd,n;
    char sendline[100];
    char recvline[100];
    struct sockaddr_in servaddr;

    sockfd=socket(AF_INET,SOCK_STREAM,0);
    bzero(&servaddr,sizeof servaddr);

    servaddr.sin_family=AF_INET;
    servaddr.sin_port=htons(22000);

    inet_pton(AF_INET,"127.0.0.1",&(servaddr.sin_addr));

    connect(sockfd,(struct sockaddr *)&servaddr,sizeof(servaddr));

    while(1)
    {
        bzero( sendline, 100);
        bzero( recvline, 100);
        fgets(sendline,100,stdin); /*stdin = 0 , for standard input */

        write(sockfd,sendline,strlen(sendline)+1);
        read(sockfd,recvline,100);
        printf("%s",recvline);
    }

}
```

Echo Server

Echo Server is a device which sends back whatever data it has received

```
/*Required Headers*/

#include <sys/types.h>

#include <sys/socket.h>
```

```
#include <netdb.h>

#include <stdio.h>

#include<string.h>

int main()

{

       char str[100];

       int listen_fd, comm_fd;

       struct sockaddr_in servaddr;

       listen_fd = socket(AF_INET, SOCK_STREAM, 0);

       bzero( &servaddr, sizeof(servaddr));

       servaddr.sin_family = AF_INET;

       servaddr.sin_addr.s_addr = htons(INADDR_ANY);

       servaddr.sin_port = htons(22000);

       bind(listen_fd, (struct sockaddr *) &servaddr,
sizeof(servaddr));

       listen(listen_fd, 10);

       comm_fd = accept(listen_fd, (struct sockaddr*) NULL,
NULL);

       while(1)

       {

     bzero( str, 100);

             read(comm_fd,str,100);
```

```
            printf("Echoing back - %s",str);

            write(comm_fd, str, strlen(str)+1);

      }

}
```

8. Repeat Exercises 6 & 7 for UDP.

UDP_client.c

```
// UDP client program
#include <arpa/inet.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include <sys/socket.h>
#include <sys/types.h>
#define PORT 5000
#define MAXLINE 1024
int main()
{
    int sockfd;
    char buffer[MAXLINE];
    char* message = "Hello Server";
    struct sockaddr_in servaddr;

    int n, len;
    // Creating socket file descriptor
    if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
        printf("socket creation failed");
        exit(0);
    }

    memset(&servaddr, 0, sizeof(servaddr));

    // Filling server information
    servaddr.sin_family = AF_INET;
    servaddr.sin_port = htons(PORT);
    servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    // send hello message to server
    sendto(sockfd, (const char*)message, strlen(message),
           0, (const struct sockaddr*)&servaddr,
           sizeof(servaddr));
    // receive server's response
    printf("Message from server: ");
    n = recvfrom(sockfd, (char*)buffer, MAXLINE,
                 0, (struct sockaddr*)&servaddr,
                 &len);
    puts(buffer);
    close(sockfd);
    return 0;
}
```

9. Repeat Exercise 7 with multiplexed I/O operations

```
#include    "unp.h"


void
str_cli(FILE *fp, int sockfd)
{
    int         maxfdp1;
    fd_set      rset;
    char        sendline[MAXLINE], recvline[MAXLINE];


    FD_ZERO(&rset);
    for ( ; ; ) {
        FD_SET(fileno(fp), &rset);
        FD_SET(sockfd, &rset);
        maxfdp1 = max(fileno(fp), sockfd) + 1;
        Select(maxfdp1, &rset, NULL, NULL, NULL);


        if (FD_ISSET(sockfd, &rset)) {  /* socket is readable */
            if (Readline(sockfd, recvline, MAXLINE) == 0)
                err_quit("str_cli: server terminated prematurely");
            Fputs(recvline, stdout);
        }
        if (FD_ISSET(fileno(fp), &rset)) {  /* input is readable */
            if (Fgets(sendline, MAXLINE, fp) == NULL)
                return;     /* all done */
            Writen(sockfd, sendline, strlen(sendline));
        }
    }
}
```

10. Simulate Bellman-Ford Routing algorithm in NS2.

```
#include <stdio.h>
#include <stdlib.h>

#define INFINITY 99999

//struct for the edges of the graph
struct Edge {
        int u;      //start vertex of the edge
        int v;      //end vertex of the edge
        int w;      //weight of the edge (u,v)
};

//Graph - it consists of edges
struct Graph {
        int V;      //total number of vertices in the graph
        int E;      //total number of edges in the graph
        struct Edge *edge; //array of edges
};

void bellmanford(struct Graph *g, int source);
void display(int arr[], int size);

int main(void) {
        //create graph
        struct Graph *g = (struct Graph*)malloc(sizeof(struct Graph));
        g->V = 4;           //total vertices
        g->E = 5;           //total edges
        //array of edges for graph
        g->edge = (struct Edge*)malloc(g->E * sizeof(struct Edge));

        //------- adding the edges of the graph
        /*
                edge(u, v)
                where   u = start vertex of the edge (u,v)
                                v = end vertex of the edge (u,v)

                w is the weight of the edge (u,v)
        */

        //edge 0 --> 1
        g->edge[0].u = 0;
        g->edge[0].v = 1;
        g->edge[0].w = 5;

        //edge 0 --> 2
        g->edge[1].u = 0;
        g->edge[1].v = 2;
        g->edge[1].w = 4;

        //edge 1 --> 3
        g->edge[2].u = 1;
        g->edge[2].v = 3;
        g->edge[2].w = 3;

        //edge 2 --> 1
        g->edge[3].u = 2;
        g->edge[3].v = 1;
        g->edge[3].w = -6;

        //edge 3 --> 2
        g->edge[4].u = 3;
        g->edge[4].v = 2;
        g->edge[4].w = 2;

        bellmanford(g, 0);              //0 is the source vertex

        return 0;
}

void bellmanford(struct Graph *g, int source) {
```

```
//variables
int i, j, u, v, w;

//total vertex in the graph g
int tV = g->V;

//total edge in the graph g
int tE = g->E;

//distance array
//size equal to the number of vertices of the graph g
int d[tV];

//predecessor array
//size equal to the number of vertices of the graph g
int p[tV];

//step 1: fill the distance array and predecessor array
for (i = 0; i < tV; i++) {
		d[i] = INFINITY;
		p[i] = 0;
}

//mark the source vertex
d[source] = 0;

//step 2: relax edges |V| - 1 times
for(i = 1; i <= tV-1; i++) {
		for(j = 0; j < tE; j++) {
				//get the edge data
				u = g->edge[j].u;
				v = g->edge[j].v;
				w = g->edge[j].w;

				if(d[u] != INFINITY && d[v] > d[u] + w) {
						d[v] = d[u] + w;
						p[v] = u;
				}
		}
}

//step 3: detect negative cycle
//if value changes then we have a negative cycle in the graph
```

```
        //and we cannot find the shortest distances
        for(i = 0; i < tE; i++) {
                u = g->edge[i].u;
                v = g->edge[i].v;
                w = g->edge[i].w;
                if(d[u] != INFINITY && d[v] > d[u] + w) {
                        printf("Negative weight cycle detected!\n");
                        return;
                }
        }

        //No negative weight cycle found!
        //print the distance and predecessor array
        printf("Distance array: ");
        display(d, tV);
        printf("Predecessor array: ");
        display(p, tV);
}

void display(int arr[], int size) {
        int i;
        for(i = 0; i < size; i ++) {
                printf("%d ", arr[i]);
        }
        printf("\n");
}
```