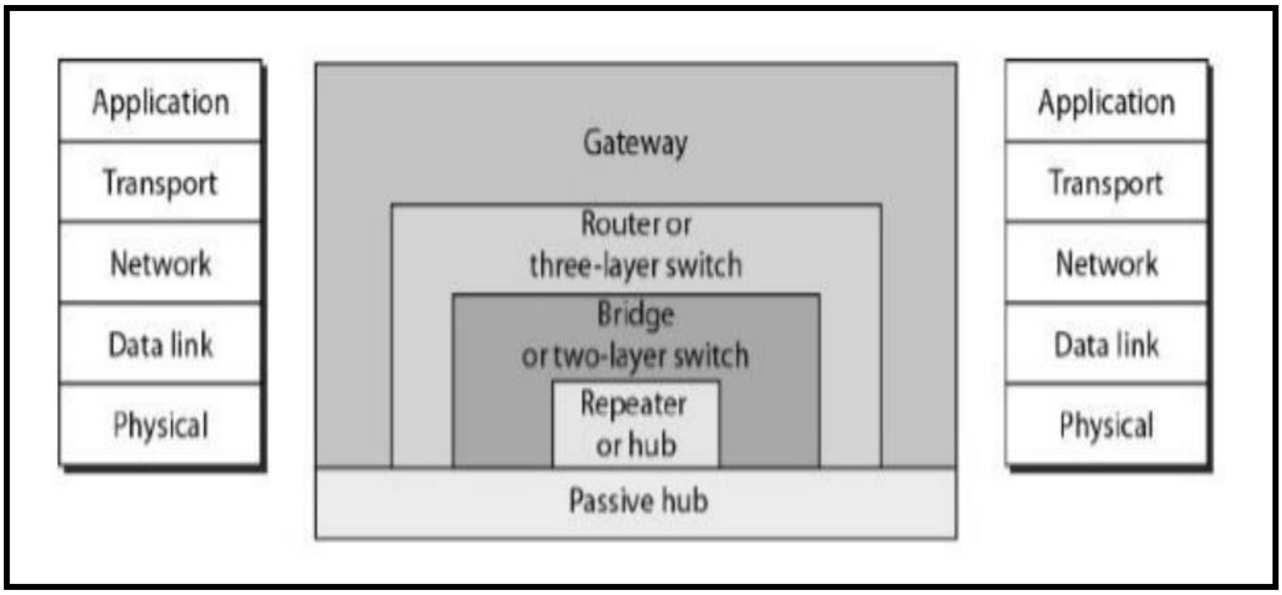
PART A

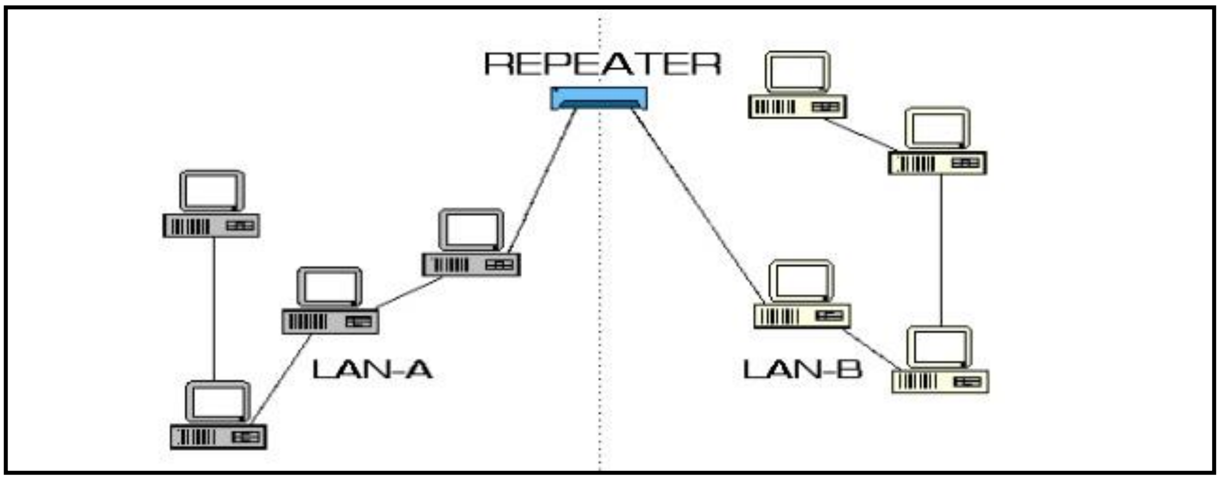
EXPERIMENT NO. 1:

Study of network devices in detail

AIM:To study the network devices in detail

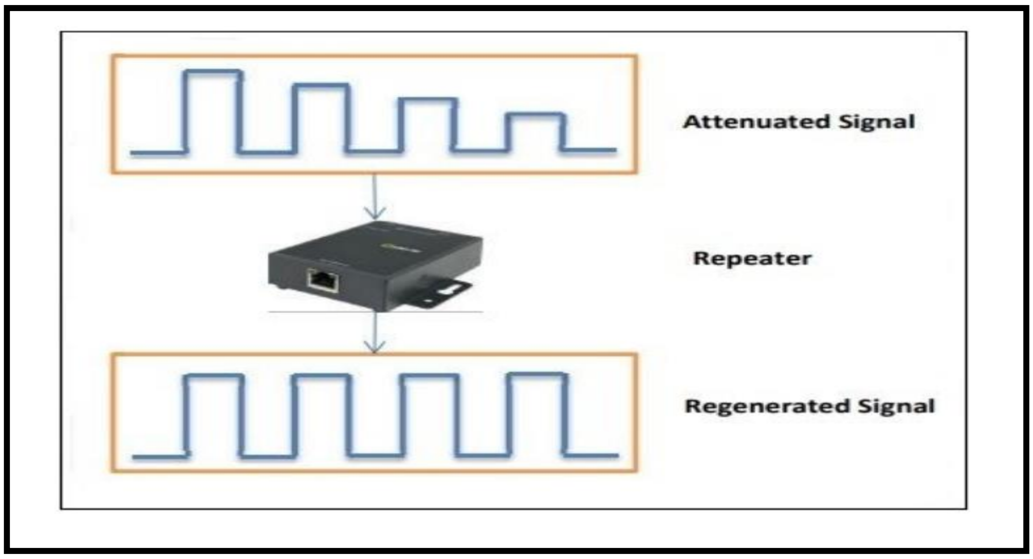


1. **Repeaters:**



- Repeaters are network devices operating at the physical layer of the OSI model that **amplify** or **regenerate** an **incoming signal** before retransmitting it.
- They are incorporated in networks to expand its coverage area. They are also known as **signal boosters**.

- Signals that carry information within a network can travel a fixed distance before attenuation endangers the integrity of the data.
- A repeater receives a signal and, before it becomes too weak or corrupted, **regenerates** the **original bit pattern**.
- The repeater then sends the refreshed signal.
- A repeater can extend the **physical length of a LAN**.
- The location of a repeater on a link is vital. A repeater must be placed so that a signal reaches it before any noise changes the meaning of any of its bits.
- If the corrupted bit travels much farther, however, accumulated noise can change its meaning completely.
- At that point, the original voltage is not recoverable, and the error needs to be corrected.
- A repeater placed on the line before the legibility of the signal becomes lost can still read the signal well enough to determine the intended voltages and replicate them in



Types of Repeaters:

According to the **types of signals that they regenerate**, repeaters can be classified in to two categories –

A. **Analog Repeaters** − They can only amplify the analog signal.
B. **Digital Repeaters** − They can reconstruct a distorted signal.

According to the **types of networks that they connect**, repeaters can be categorized in to two types −

A. **Wired Repeaters** − They are used in wired LANs.
B. **Wireless Repeaters** − They are used in wireless LANs and cellular networks

According to the **domain of LANs they connect**, repeaters can be divided into two categories –

A. **Local Repeaters** − They connect LAN segments separated by small distance.

B. **Remote Repeaters** − They connect LANs that are far from each other.

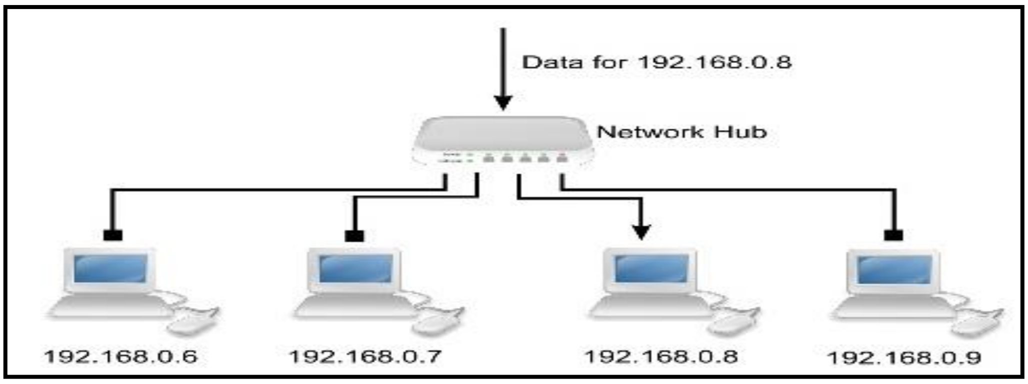
Advantages of Repeaters:

- Repeaters are simple to install and can easily extend the length or the coverage area of networks.
- They are cost effective.
- Repeaters don't require any processing overhead. The only time they need to be investigated is in case of degradation of performance.
- They can connect signals using different types of cables.

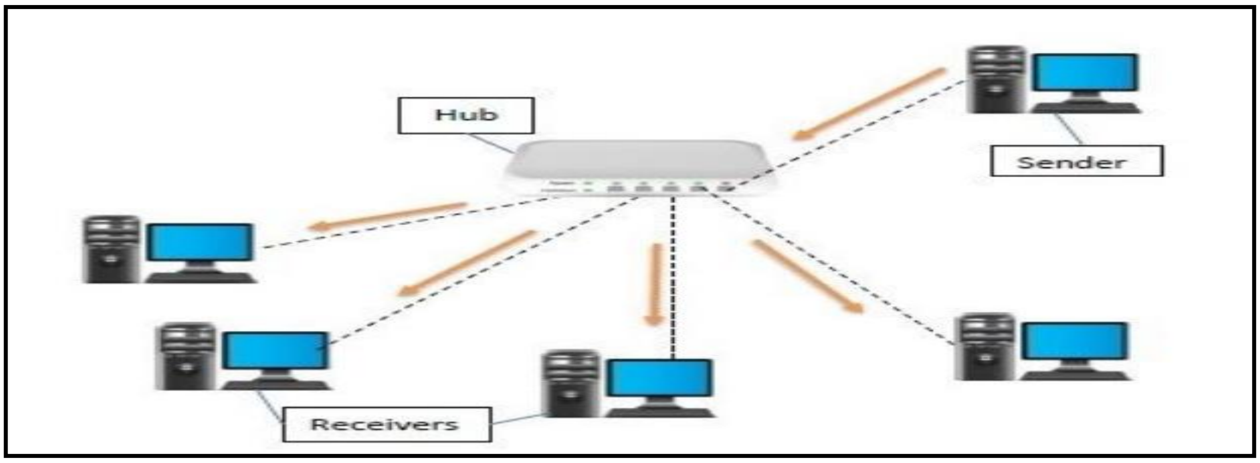
Disadvantages of Repeaters:

- Repeaters cannot connect dissimilar networks.
- They cannot differentiate between actual signal and noise.
- They cannot reduce network traffic or congestion. Most networks have limitations upon the number of repeaters that can be deployed.

2. Hub



- A hub is a physical layer networking device which is used to connect multiple devices in a network. They are generally used to connect computers in a LAN.
- A hub has many ports in it. A computer which intends to be connected to the network is plugged in to one of these ports.
- When a data frame arrives at a port, it is broadcast to every other port, without considering whether it is destined for a particular destination or not.



Types of Hubs:

A. Passive Hubs:

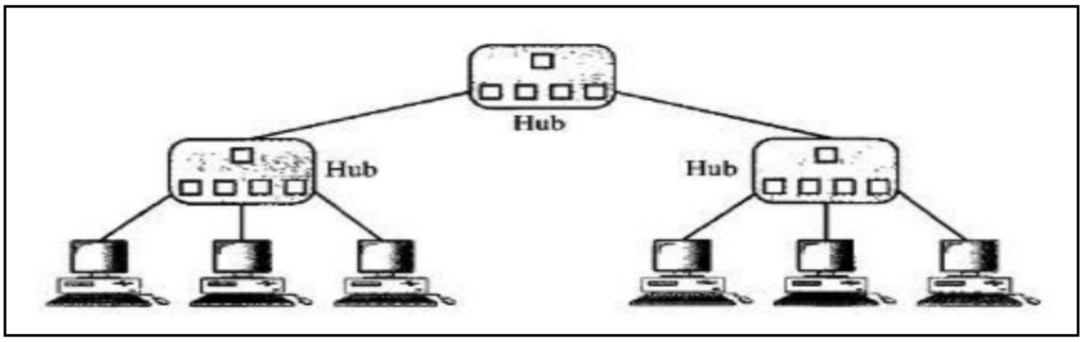
- A passive hub is just a connector.
- It connects the wires coming from different branches.
- In a star – topology Ethernet LAN, a passive hub is just a point where the signals coming from different stations collide; the hub is the collision point.
- This type of a hub is part of the media; its location in the Internet model is below the physical layer.

B. Active Hubs:

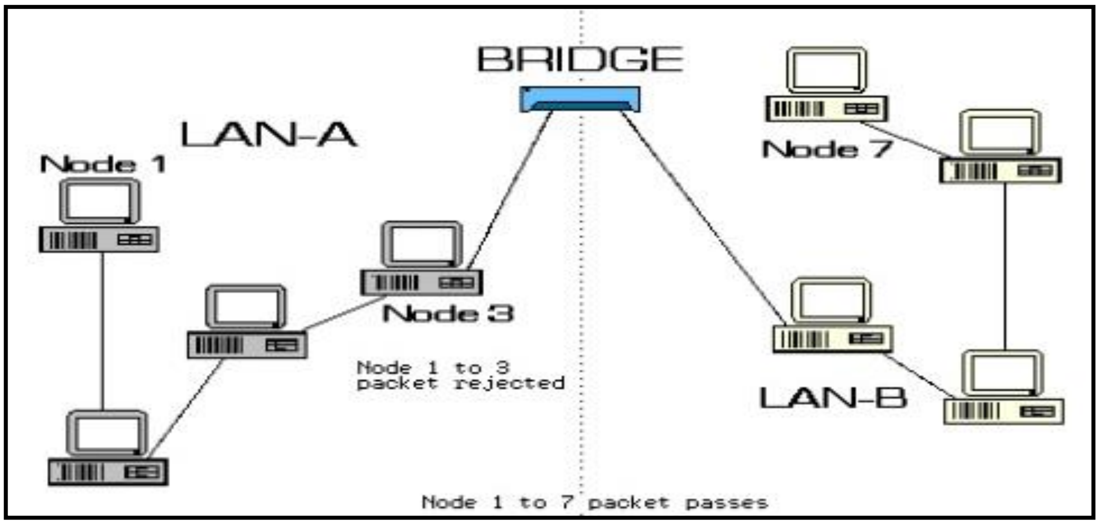
- An active hub is actually a multipart repeater. It is normally used to create connections between stations in a physical star topology.
- However, hubs can also be used to create multiple levels of hierarchy, as shown in Figure. The hierarchical use of hubs removes the length limitation of 10Base-T (100 m).

C. Intelligent Hubs:

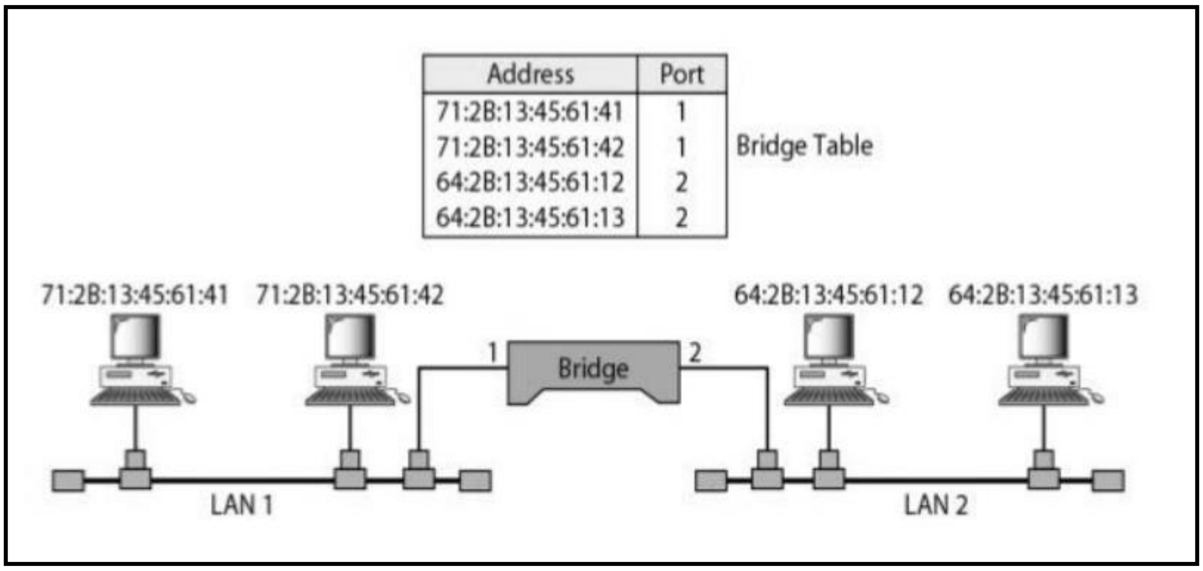
- Intelligent hubs are active hubs that provide additional network management facilities.
- They can perform a variety of functions of more intelligent network devices like network management, switching, providing flexible data rates etc.



3. Bridges:



- A bridge operates in the physical layer as well as in the data link layer. It can regenerate the signal that it receives and as a data link layer device, it can check the physical addresses of source and destination contained in the frame.
- The major difference between the bridge and the repeater is that the bridge and the repeater is that the bridge has a filtering capability.
- That means it can check the destination address of a frame and decide if the frame should be forwarded or dropped.
- If the frame is forwarded, then the bridge should specify the port over which it should be forwarded.



Address	Port
71:2B:13:45:61:41	1
71:2B:13:45:61:42	1
64:2B:13:45:61:12	2
64:2B:13:45:61:13	2

Types of Bridges :

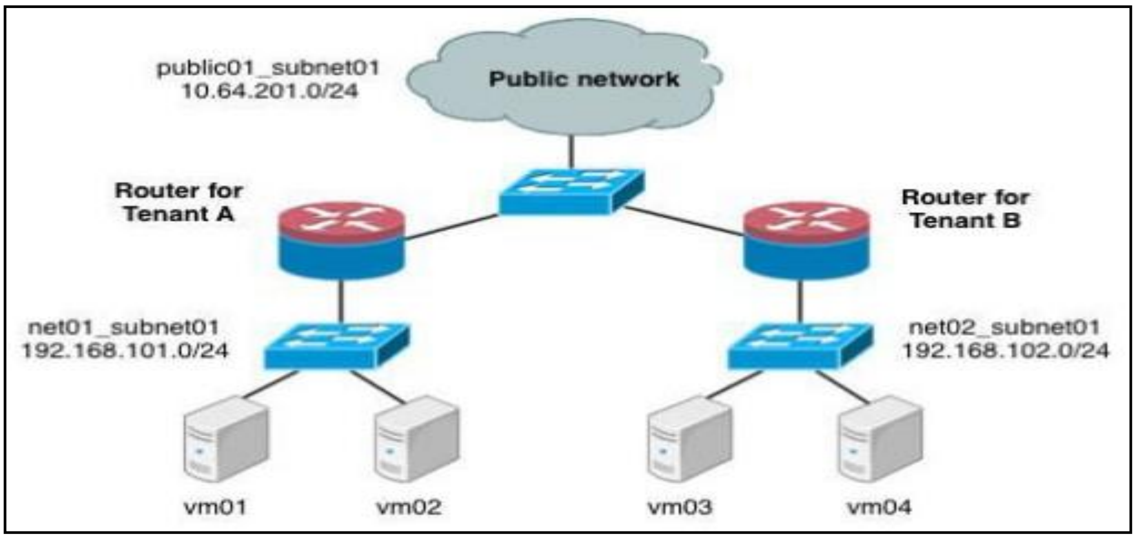
A. Transparent Bridges

- These are the bridges 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 are unnecessary.
- These bridges make use of two processes i.e. bridge forwarding and bridge learning.

B. Source Routing Bridges

- In these bridges, routing operation is performed by source station and the frame specifies which route to follow.
- The host can discover frames by sending a special frame called discovery frame, which spreads through the entire network using all possible paths to destination.

4. Router:



Routers are networking devices operating at layer 3 or a network layer of the OSI model.

They are responsible for receiving, analysing, and forwarding data packets among the connected computer networks.

When a data packet arrives, the router inspects the destination address, consults its routing tables to decide the optimal route and then transfers the packet along this route.

A router is a three-layer device that routes packets based on their logical addresses (host-to-host addressing).

A router normally connects LANs and WANs in the Internet and has a routing table that is used for making decisions about the route. The routing tables are normally dynamic and are updated using routing protocols.

Data is grouped into packets, or blocks of data.

Each packet has a physical device address as well as logical network address. The network address allows routers to calculate the optimal path to a workstation or computer.

The functioning of a router depends largely upon the routing table stored in it. The routing table stores the available routes for all destinations. The router consults the routing table to determine the optimal route through which the data packets can be sent

A routing table typically contains the following entities –

IP addresses and subnet mask of the nodes in the network

IP addresses of the routers in the network

Interface information among the network devices and channels

Routing tables are of two types–

Static Routing Table: Here, the routes are fed manually and are not refreshed automatically. It is suitable for small networks containing 2-3 routers.

Dynamic Routing Table :Here, the router communicates with other routers using routing protocols to determine the available routes. It is suited for larger networks having large numbers of routers

Types of Routers:

A variety of routers are available depending upon their usages. The main types of routers are as follows:-

A] Wireless Router

They provide WiFi connection WiFi devices like laptops, smartphones etc.

They can also provide standard Ethernet routing.

For indoor connections, the range is 150 feet while it's 300 feet for outdoor connections.

B] Broadband Routers

They are used to connect to the Internet through telephone and to use voice over Internet Protocol (VoIP) technology for providing high-speed Internet access.

They are configured and provided by the Internet Service Provider (ISP)

C] Core Routers

They can route data packets within a given network, but cannot route the packets between the networks. They help to link all devices within a network thus forming the backbone of the network. It is used by ISP and communication interfaces.

D] Edge Routers

They are low-capacity routers placed at the periphery of the networks.

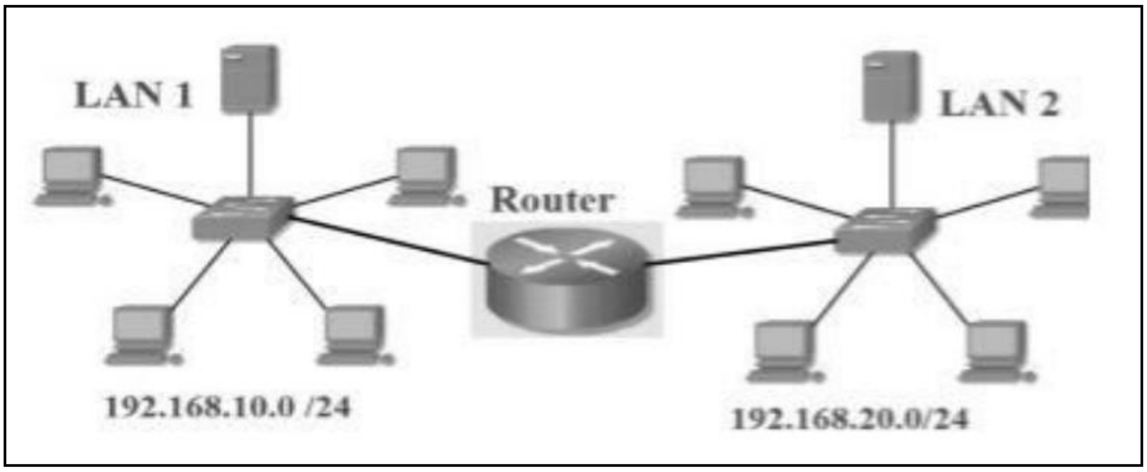
They connect the internal network to the external networks, and are suitable for transferring data packets across networks.

They use the Border Gateway Protocol (BGP) for connectivity. There are two types of edge routers, subscriber edge routers and label edge routers.

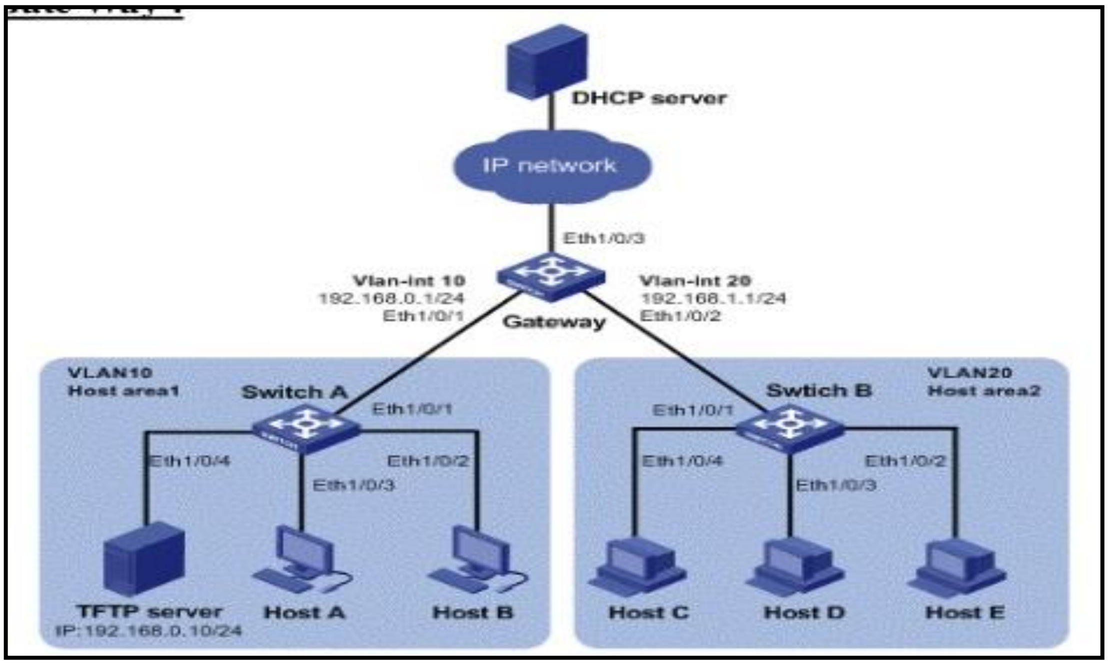
E] Brouters

Brouters are specialised routers that can provide the functionalities of bridges as well.

Like a bridge, brouters help to transfer data between networks. And like a router, they route the data within the devices of a network.



5. Gateway:



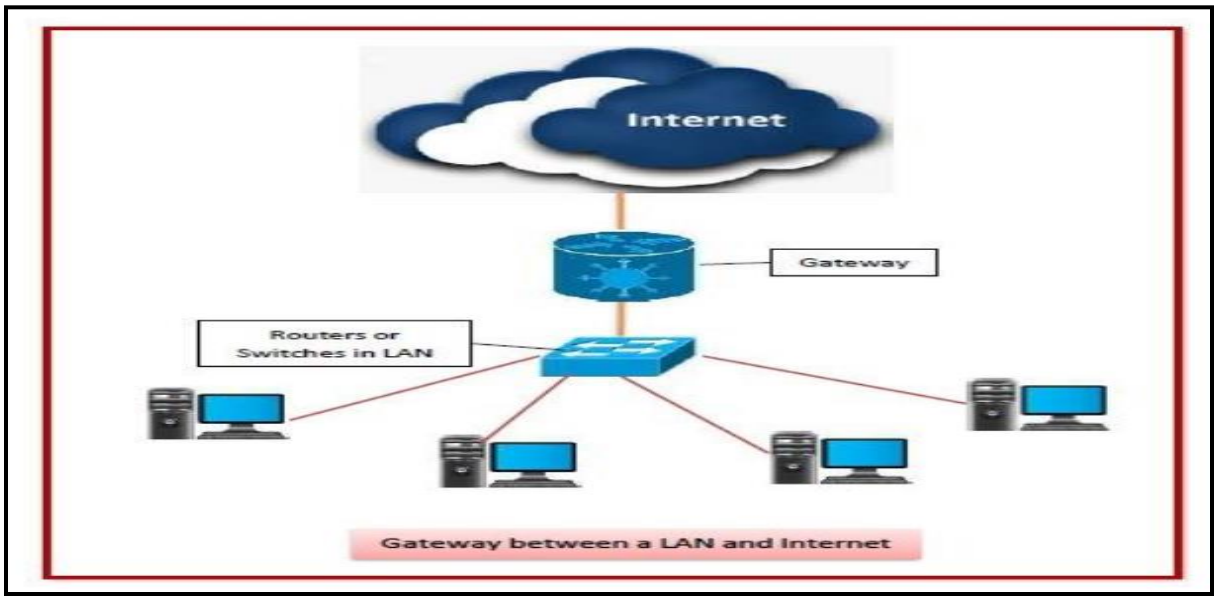
A gateway is a network node that forms a passage between two networks operating with different transmission protocols.

The most common type of gateways, the network gateway operates at layer 3, i.e. network layer of the OSI (open systems interconnection) model.

However, depending upon the functionality, a gateway can operate at any of the seven layers of OSI model.

It acts as the entry – exit point for a network since all traffic that flows across the networks should pass through the gateway.

Only the internal traffic between the nodes of a LAN does not pass through the gateway


Internet
Gateway
Routers or Switches in LAN
Gateway between a LAN and Internet

Gateway is located at the boundary of a network and manages all data that inflows or outflows from that network.

It forms a passage between two different networks operating with different transmission protocols.

A gateway operates as a protocol converter, providing compatibility between the different protocols used in the two different networks.

The feature that differentiates a gateway from other network devices is that it can operate at any layer of the OSI model.

It also stores information about the routing paths of the communicating networks.

When used in enterprise scenarios, a gateway node may be supplemented as a proxy server or firewall. A gateway is generally implemented as a node with multiple NICs (network interface cards) connected to different networks. However, it can also be configured using software.It uses a packet switching technique to transmit data across the networks.

It uses a packet switching technique to transmit data across the networks.

Types of Gateways

On basis of direction of data flow, gateways are broadly divided into two categories

A] Unidirectional Gateways

They allow data to flow in only one direction.

Changes made in the source node are replicated in the destination node, but not vice versa. They can be used as archiving tools.

B] Bidirectional Gateways

They allow data to flow in both directions.

They can be used as syn

On basis of functionalities, there can be a variety of gateways, the prominent among them are as follows –

A] Network Gateway

This is the most common type of gateway that provides an interface between two dissimilar networks operating with different protocols.

Whenever the term gateway is mentioned without specifying the type, it indicates a network gateway.

B] Cloud Storage Gateway

It is a network node or server that translates storage requests with different cloud storage service API calls, such as SOAP (Simple Object Access Protocol) or REST (REpresentational State Transfer).

It facilitates integration of private cloud storage into applications without necessitating transfer of the applications into any public cloud, thus simplifying data communicationsychronization tools

C] Internet-To-Orbit Gateway (I2O)

It connects devices on the Internet to satellites and spacecraft orbiting the earth.

Two prominent I2O gateways are Project HERMES and Global Educational Network for Satellite Operations (GENSO).

D] IoT Gateway

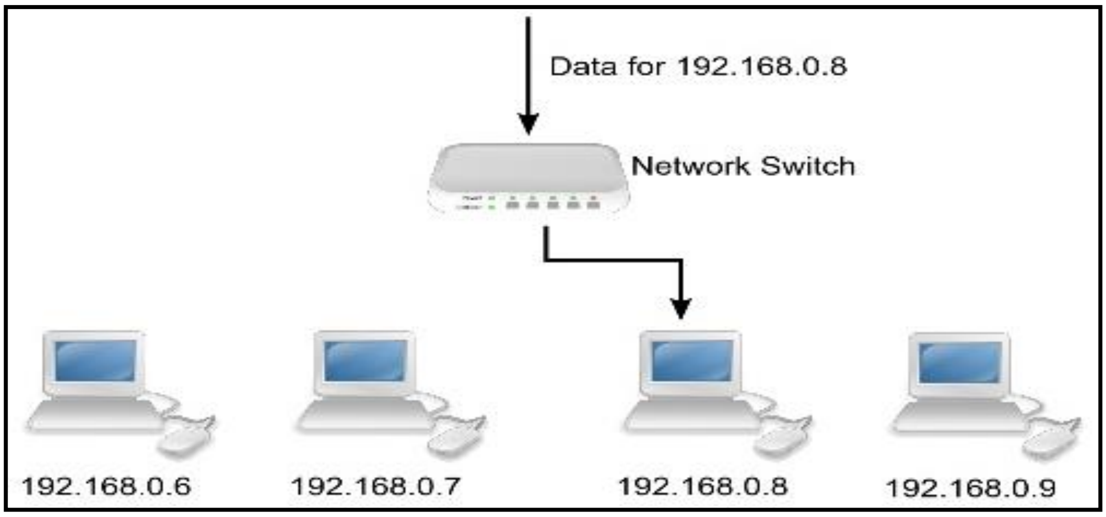
IoT gateways assimilates sensor data from IoT (Internet of Things) devices in the field and translates between sensor protocols before sending it to the cloud network.

They connect IoT devices, cloud networks and user applications.

E] VoiP Trunk Gateway

It facilitates data transmission between plain old telephone service (POTS) devices like landline phones and fax machines, with VoIP (voice over Internet Protocol) network.

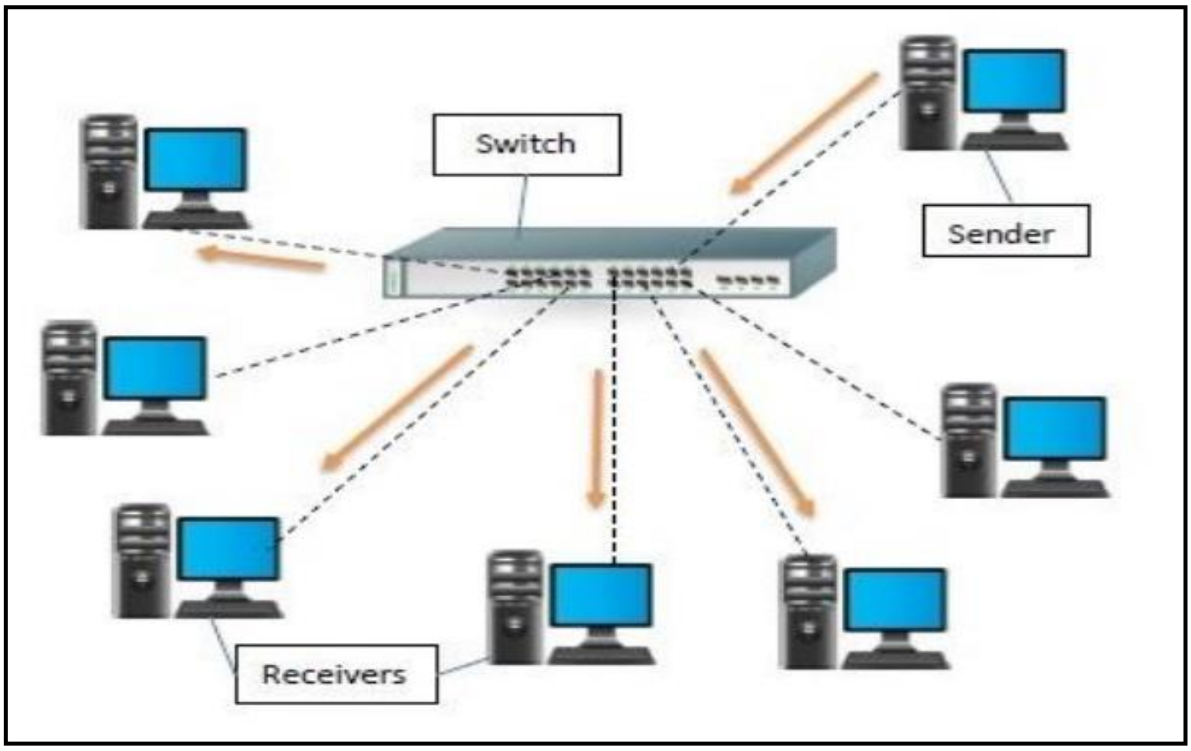
6. Switch



A switch is a data link layer networking device which connects devices in a network and uses packet switching to send and receive data over the network.

Like a hub, a switch also has many ports, to which computers are plugged in. However, when a data frame arrives at any port of a network switch, it examines the destination address and sends the frame to the corresponding device(s).

Thus, it supports both unicast and multicast communications.



We can have a two-layer switch or a three-layer switch.

A three-layer switch is used at the network layer; it is a kind of router.

The two-layer switch performs at the physical and data link layers.

A two-layer switch is a bridge, a bridge with many ports and a design that allows better (faster) performance.

A bridge with a few ports can connect a few LANs together. A bridge with many ports may be able to allocate a unique port to each station, with each station on its own independent entity.

This means no competing traffic (no collision, as we saw in Ethernet).

A two-layer switch, as a bridge does, makes a filtering decision based on the MAC Address of the frame it received.

However, a two-layer switch can be more sophisticated. It can have a buffer to hold the frames for processing.

It can have a switching factor that forwards the frames faster. Some new two-layer switches, called cut-through switches, have been designed to forward the frame as soon as they check the MAC addresses in the header of the frame.

Types of Switches

There are variety of switches that can be broadly categorised into 4 types

1. **Unmanaged Switch**

- These are inexpensive switches commonly used in home networks and small businesses. They can be set up by simply plugging in to the network, after which they instantly start operating.
- When more devices need to be added, more switches are simply added by this plug and play method.
- They are referred to as u managed since they do not require to be configured or monitored

2. **Managed Switch**

- These are costly switches that are used in organisations with large and complex networks, since they can be customized to augment the functionalities of a standard switch.
- The augmented features may be QoS (Quality of Service) like higher security levels, better precision control and complete network management.
- Despite their cost, they are preferred in growing organizations due to their scalability and flexibility.

3. **Simple Network Management Protocol (S LAN Switch)**

- Local Area Network (LAN) switches connect devices in the internal LAN of an organization. They are also referred to as Ethernet switches or data switches.
- These switches are particularly helpful in reducing network congestion or bottlenecks.
- They allocate bandwidth in a manner so that there is no overlapping of data packets in a network.

4. **PoE Switch**

- Power over Ethernet (PoE) switches are used in PoE Gigabit Ethernets.
- PoE technology combines data and power transmission over the same cable so that devices connected to it can receive both electricity as well as data over the same line.
- PoE switches offer greater flexibility and simplifies the cabling connectionsNMP) is used for configuring managed switches

7. Modem

A modem is the most important network device and it is used daily in our life. If we notice the internet connection to homes was given with the help of a wire. then wire carries internet data from one place to another.

But, every computer gives digital or binary data in the form of zeros & ones.

The full form of the modem is a modulator and a demodulator. So it modulates as well as demodulates the signal among the computer and a telephone line because the computer generates digital data whereas the telephone line generates an analog signal.

Types of Modem

Modem can be categorized in several ways like direction in which it can transmit data, type of connection to the transmission line, transmission mode, etc.

Depending on direction of data transmission, modem can be of these types −

A] Simplex − A simplex modem can transfer data in only one direction, from digital device to network (modulator) or network to digital device (demodulator).

B] Half duplex − A half-duplex modem has the capacity to transfer data in both the directions but only one at a time.

C] Full duplex − A full duplex modem can transmit data in both the directions simultaneously.

RESULT : Thus the study of network devices is done

PART B

Design and implement the following experiments using C compiler or and packet tracer software

EXPERIMENT NO. 2

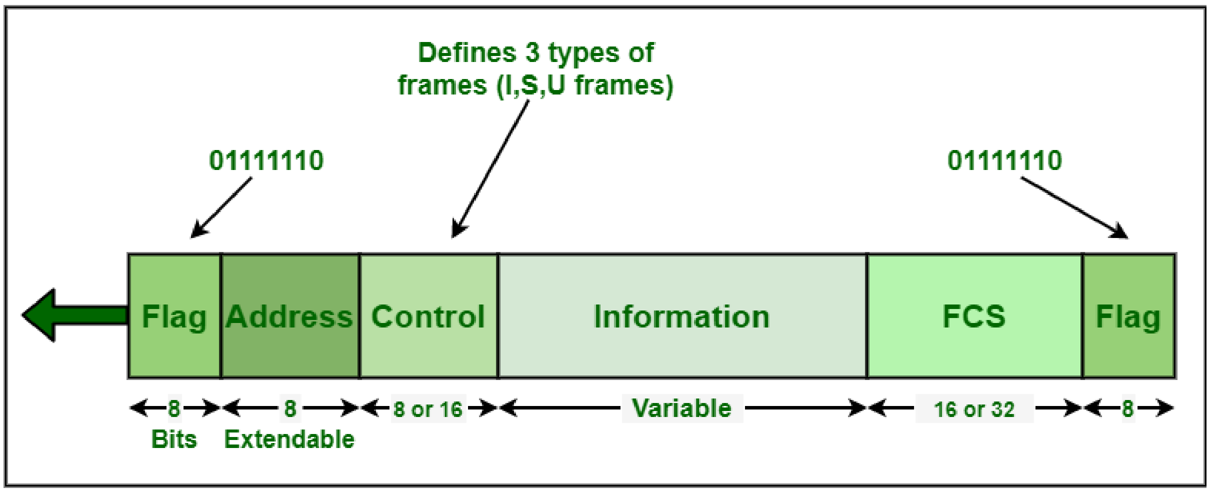
A HLDC frame to perform the following. i) Bit stuffing ii) Character stuffing.

AIM 2A: Write a program in C to design a HDLC frame with Bit Stuffing method

THEORY:

HDLC: High-Level Data Link Control (HDLC) generally uses term “frame” to indicate and represent an entity of data or a protocol of data unit often transmitted or transferred from one station to another station. Each and every frame on link should begin and end with Flag Sequence Field (F). Each of frames in HDLC includes mainly six fields. It begins with a flag field, an address field, a control field, an information field, an frame check sequence (FCS) field, and an ending flag field. The ending flag field of one frame can serve as beginning flag field of the next frame in multiple-frame transmissions.

The basic frame structure of HDLC protocol is shown below:



Basic Frame Structure

The fields of a HDLC frame are –

Flag – It is an 8-bit sequence that marks the beginning and the end of the frame. The bit pattern of the flag is 01111110.

Address – It contains the address of the receiver. If the frame is sent by the primary station, it contains the address(es) of the secondary station(s). If it is sent by the secondary station, it contains the address of the primary station. The address field may be from 1 byte to several bytes.

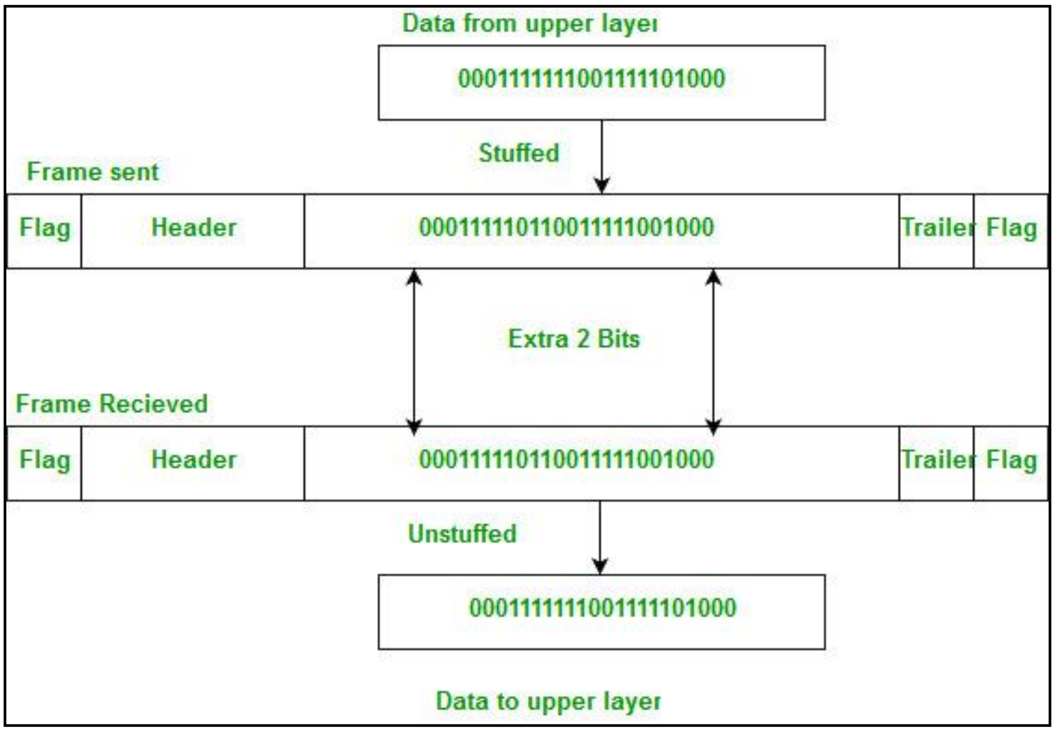
Control − It is 1 or 2 bytes containing flow and error control information.

Payload − This carries the data from the network layer. Its length may vary from one network to another.

FCS − It is a 2 byte or 4 bytes frame check sequence for error detection. The standard code used is CRC (cyclic redundancy code)

BIT STUFFING :

Bit stuffing is the insertion of non information bits into data. Note that stuffed bits should not be confused with overhead bits. Overhead bits are non-data bits that are necessary for transmission (usually as part of headers, checksums etc.).



Applications of Bit Stuffing –

1. synchronize several channels before multiplexing
2. rate-match two single channels to each other
3. run length limited coding

Run length limited coding – To limit the number of consecutive bits of the same value(i.e., binary value) in the data to be transmitted. A bit of the opposite value is inserted after the maximum allowed number of consecutive bits.

Bit stuffing technique does not ensure that the sent data is intact at the receiver side (i.e., not corrupted by transmission errors). It is merely a way to ensure that the transmission starts and ends at the correct places.

The new technique allows data frames to contain an arbitrary number if bits and allows character codes with an arbitrary no of bits per character. Each frame begins and ends with special bit pattern, 01111110, called a flag byte. Whenever the sender's data link layer encounters five consecutive ones in the data , it automatically stuffs a 0 bit into the outgoing bit stream. This bit stuffing is analogous to character stuffing, in which a D LE is stuffed into the outgoing character stream before DLE in the data.

Disadvantages of Bit Stuffing –

The code rate is unpredictable; it depends on the data being transmitted.

Example of bit stuffing –

Bit sequence: 110101111101011111101011111110 (without bit stuffing)

Bit sequence: 110101111100101111101010111110110 (with bit stuffing)

After 5 consecutive 1-bits, a 0-bit is stuffed. Stuffed bits are marked bold.

PLATFORM: DEV C++

PROGRAM:

```
#include<stdio.h>
#include<string.h>>
int main()
{
 int i=0,count=0;
 char databits[80];
printf("Enter Data Bits: ");
 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;
}
```

STEPS OF EXECUTION:

1. Open File tab → New Project →Console Application → C Project
2. Name the Project
3. Add a file with .c extension
4. Write the code and save the file
5. Compile and execute

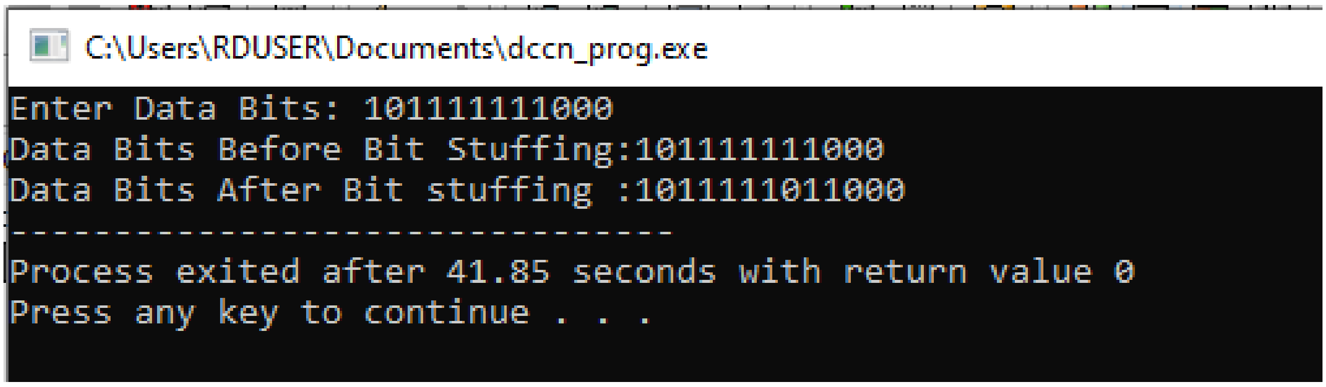
INPUT:

Enter Data Bits: 101111111000

OUTPUT:

Data Bits Before Bit Stuffing:101111111000

Data Bits After Bit stuffing :1011111011000



C:\Users\RDUSER\Documents\dccn_prog.exe

```
Enter Data Bits: 101111111000
Data Bits Before Bit Stuffing:101111111000
Data Bits After Bit stuffing :1011111011000
--------------------------------
Process exited after 41.85 seconds with return value 0
Press any key to continue . . .
```

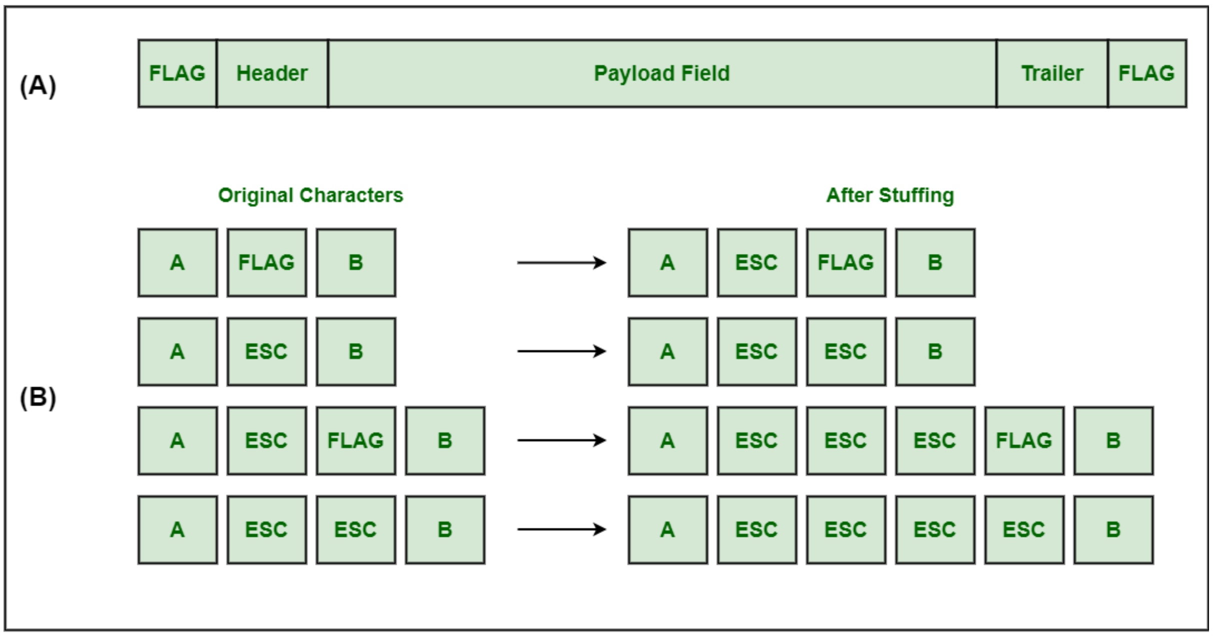
AIM 2B: Write a program in C to design a HDLC frame with Character Stuffing method

THEORY:

Character Stuffing method:

Character stuffing is also known as byte stuffing or character-oriented framing and is same as that of bit stuffing but byte stuffing actually operates on bytes whereas bit stuffing operates on bits. In byte stuffing, special byte that is basically known as ESC (Escape Character) that has predefined pattern is generally added to data section of the data stream or frame when there is message or character that has same pattern as that of flag byte.

But receiver removes this ESC and keeps data part that causes some problems or issues. In simple words, we can say that character stuffing is addition of 1 additional byte if there is presence of ESC or flag in text.



A Character Stuffing
(A) A frame delimited by flag bytes
(B) Four examples of byte sequences before and after byte stuffing

The framing method gets around the problem of resynchronization after an error by having each frame start with the ASCII character sequence D L E ST X and the sequence DLE ETX. If the destination ever losses the track of the frame boundaries all it has to do is look for DLE STX or DLE ETX characters to figure out. The data link layer on the receiving end removes the D LE before the data are given to the network layer. This technique is called character stuffing.

PLATFORM: DEV C++

PROGRAM:

//PROGRAM FOR CHARACTER STUFFING

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<process.h>
int main()
{
    int i=0,j=0,n,pos;
    char a[20],b[50],ch;
    printf("Enter string\n");
```

```
scanf("%s",&a);
n=strlen(a);
printf("\nEnter position : ");
scanf("%d",&pos);
if(pos>n)
{
    printf("Invalid position, Enter again : ");
    scanf("%d",&pos);
}
printf("Enter the character: ");
ch=getche();
b[0]='d';
b[1]='l';
b[2]='e';
b[3]='s';
b[4]='t';
b[5]='x';
j=6;
while(i<n)
{
    if(i==pos-1)
    {
        b[j]='d';
        b[j+1]='l';
        b[j+2]='e';
        b[j+3]=ch;
        b[j+4]='d';
        b[j+5]='l';
        b[j+6]='e';
        j=j+7;
    }
    if(a[i]=='d' && a[i+1]=='l' && a[i+2]=='e')
    {
        b[j]='d';
```

```
            b[j+1]='l';
            b[j+2]='e';
            j=j+3;
        }
        b[j]=a[i];
        i++;
        j++;
    }
    b[j]='d';
    b[j+1]='l';
    b[j+2]='e';
    b[j+3]='e';
    b[j+4]='t';
    b[j+5]='x';
    b[j+6]='\0';
    printf("\n\nFrame after stuffing: ");
    printf("%s",b);
    getch();
    return 0;
}
```

STEPS OF EXECUTION:

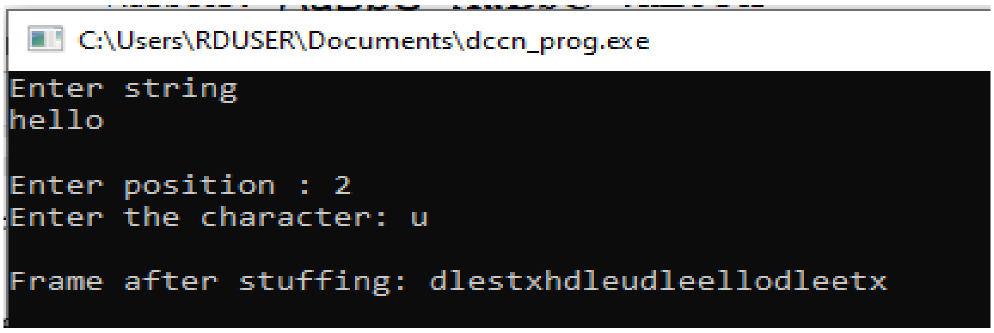
1. Open File tab → New Project →Console Application → C Project
2. Name the Project
3. Add a file with .c extension
4. Write the code and save the file
5. Compile and execute

INPUT:

```
Enter string
hello
Enter position : 2
Enter the character: u
```

OUTPUT:

Frame after stuffing: dlestxhdleudleellodleetx



```
Enter string
hello

Enter position : 2
Enter the character: u

Frame after stuffing: dlestxhdleudleellodleetx
```

RESULT : Thus the implementation of Bit stuffing and Character stuffing in a HLDC frame is successfully done.

EXPERIMENT NO. 3

Distance vector algorithm and find path for transmission.

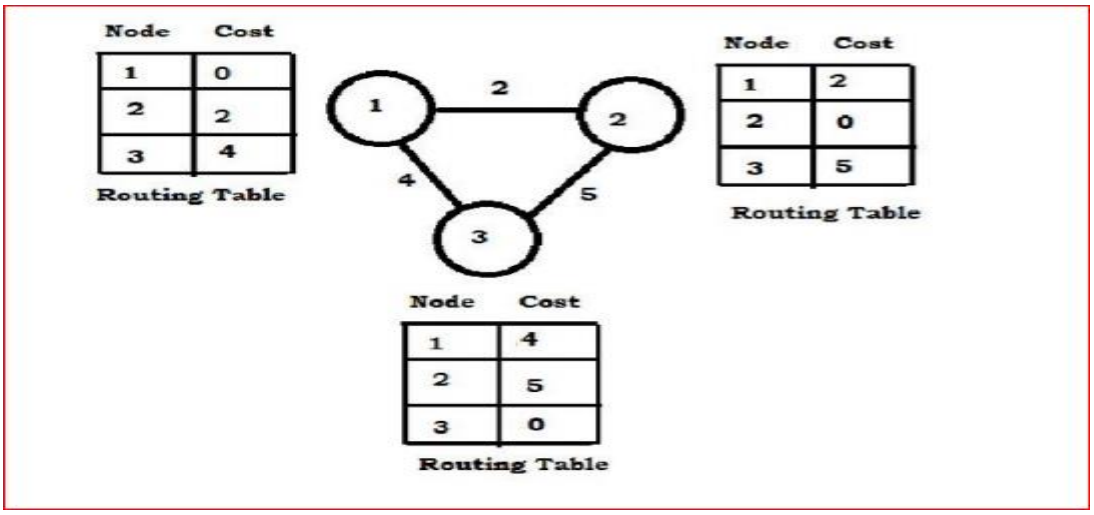
AIM: Implement Distance vector algorithm and find path for transmission.

THEORY:

Routing algorithm is a part of network layer software which is responsible for deciding which output line an incoming packet should be transmitted on. If the subnet uses datagram internally, this decision must be made anew for every arriving data packet since the best route may have changed since last time. If the subnet uses virtual circuits internally, routing decisions are made only when a new established route is being set up. The latter case is sometimes called session routing, because a rout remains in force for an entire user session (e.g., login session at a terminal or a file).

Distance vector routing algorithms operate by having each router maintain a table (i.e., vector) giving the best-known distance to each destination and which line to get there. These tables are updated by exchanging information with the neighbours. The distance vector routing algorithm is sometimes called by other names, including the distributed Bellman-Ford routing algorithm and the Ford-Fulkerson algorithm, after the researchers who developed it (Bellman, 1957; and Ford and Fulkerson, 1962). In distance vector routing, each router maintains a routing table indexed by, and containing one entry for, each router in subnet. This entry contains two parts: the preferred outgoing line to use for that destination, and an estimate of the time or distance to that destination. The metric used might be number of hops, time delay in milliseconds, total number of packets queued along the path, or something similar.
The router is assumed to know the "distance" to each of its neighbour. If the metric is hops, the distance is just one hop. If the metric is queue length, the router simply examines each queue. If the metric is delay, the router can measure it directly with special ECHO packets that the receiver just time stamps and sends back as fast as possible.

The Distance vector algorithm is iterative, asynchronous and distributed.
Distributed: It is distributed in that each node receives information from one or more of its directly attached neighbours, performs calculation and then distributes the result back to its neighbours.
Iterative: It is iterative in that its process continues until no more information is available to be exchanged between neighbors.
Asynchronous: It does not require that all of its nodes operate in the lock step with each other.



ALGORITHM:

```
At each node x,
Initialization
for all destinations y in N:
Dx(y) = c(x,y)     // If y is not a neighbor then c(x,y) = ∞
for each neighbor w
Dw(y) = ?     for all destination y in N.
for each neighbor w
send distance vector Dx = [ Dx(y)  : y in N ] to w
loop
wait(until I receive any distance vector from some neighbour w)
  for each y in N:
  Dx(y) = minv{c(x,v)+Dv(y)}
If Dx(y) is changed for any destination y
Send distance vector Dx = [ Dx(y) : y in N ] to all neighbours
forever
```

PLATFORM: GCC COMPILER (VI EDITOR[UBUNTU])

PROGRAM:

```
#include<stdio.h>
int dist[50][50],temp[50][50],n,i,j,k,x;
void dvr();
```

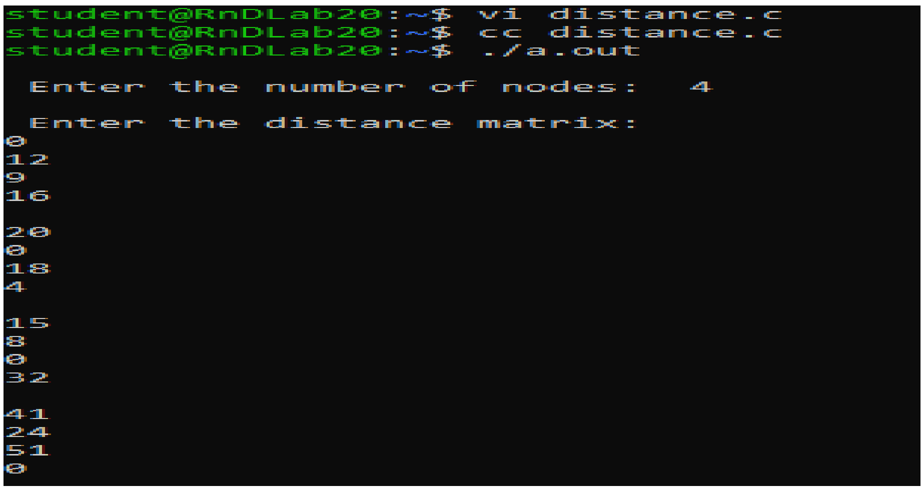
```
int main()
{
printf("\nEnter the number of nodes : ");
    scanf("%d",&n);
printf("\nEnter the distance matrix :\n");
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
        {
            scanf("%d",&dist[i][j]);
            temp[i][j]=j;
        }
        printf("\n");
            }
dvr();
printf("enter value of i &j:");
     scanf("%d",&i);
             scanf("%d",&j);
            printf("enter the new cost");
             scanf("%d",&x);
             dist[i][j]=x;
            printf("After update\n\n");
dvr();
             return 0;
}
void dvr()
{
            for (i = 0; i < n; i++)
            {
for (j = 0; j < n; j++)
                  {
            for (k = 0; k < n; k++)
                        {
            if (dist[i][k] + dist[k][j] < dist[i][j])
```

```
          {
                              dist[i][j] = dist[i][k] + dist[k][j];
                              temp[i][j] = k;
                    }
                         }
                    }
                }
          }

          for(i=0;i<n;i++)
       {
printf("\n\nState value for router %d is \n",i+1);
          for(j=0;j<n;j++)
printf("\t\nnode %d via %d Distance%d",j+1,temp[i][j]+1,dist[i][j]);
       }
   printf("\n\n");

}
```

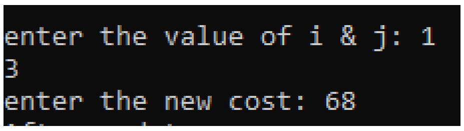
STEPS OF EXECUTION:

1. Open the vi editor (vi <filename.c>)
2. Write the code and save
3. Run cc <filename.c>
4. ./a.out

INPUT:

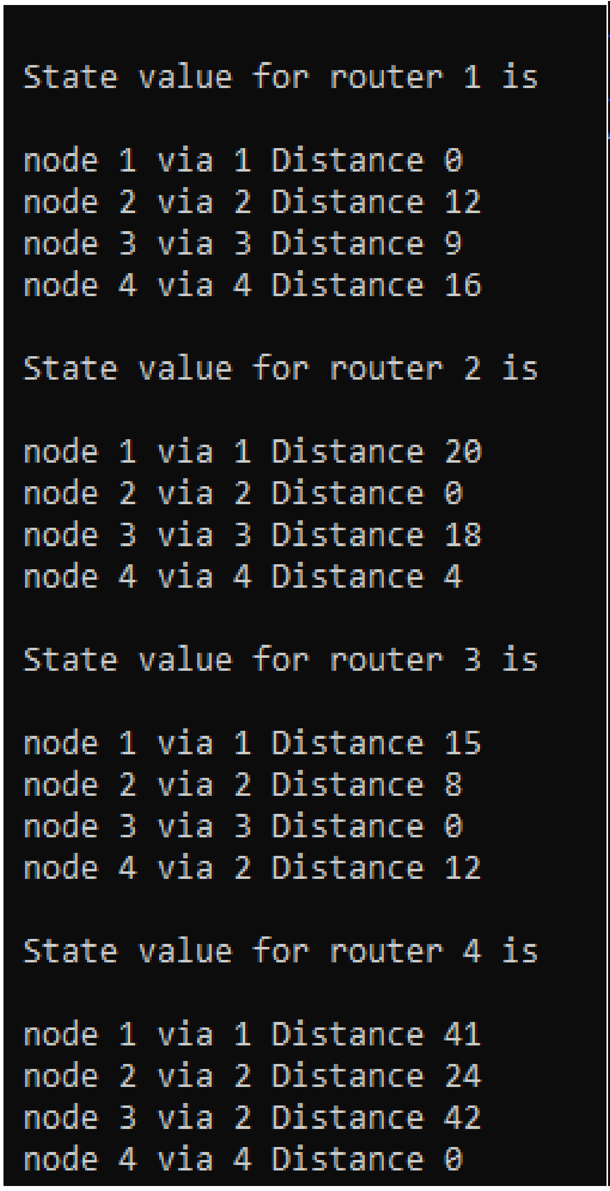


```
student@RnDLab20:~$ vi distance.c
student@RnDLab20:~$ cc distance.c
student@RnDLab20:~$ ./a.out

 Enter the number of nodes:  4

 Enter the distance matrix:
0
12
9
16

20
0
18
4

15
8
0
32

41
24
51
0
```

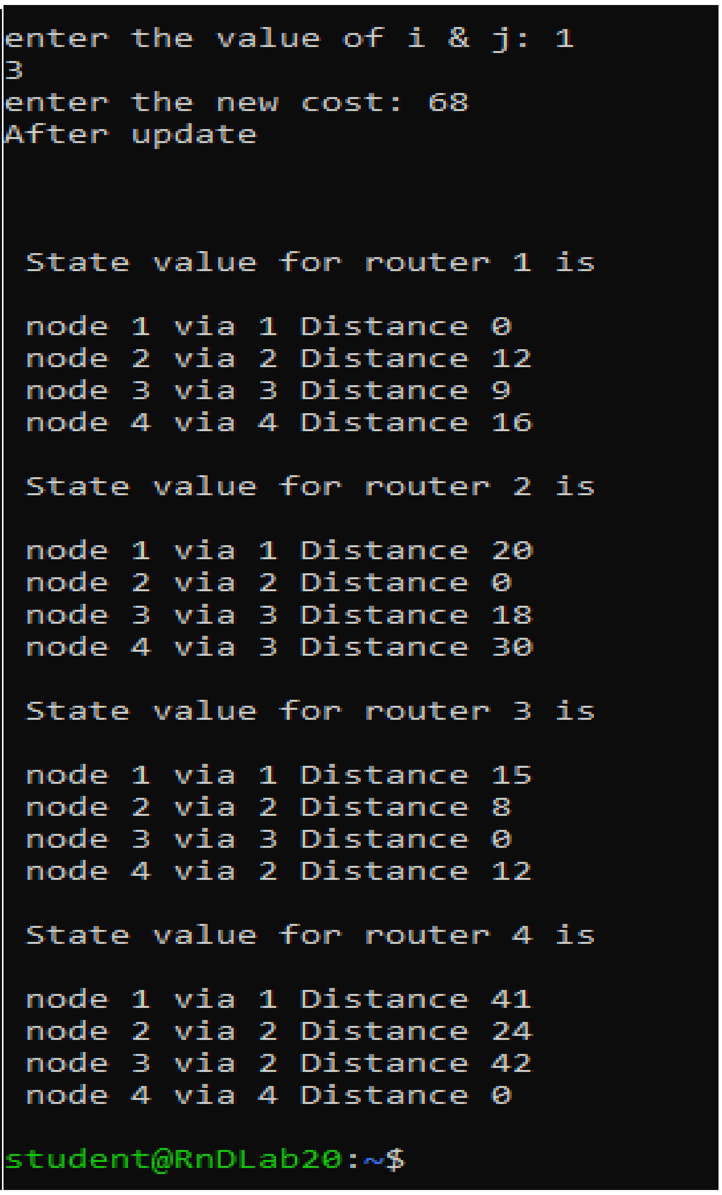


```
enter the value of i & j: 1
3
enter the new cost: 68
```

OUTPUT:



```

 State value for router 1 is

 node 1 via 1 Distance 0
 node 2 via 2 Distance 12
 node 3 via 3 Distance 9
 node 4 via 4 Distance 16

 State value for router 2 is

 node 1 via 1 Distance 20
 node 2 via 2 Distance 0
 node 3 via 3 Distance 18
 node 4 via 4 Distance 4

 State value for router 3 is

 node 1 via 1 Distance 15
 node 2 via 2 Distance 8
 node 3 via 3 Distance 0
 node 4 via 2 Distance 12

 State value for router 4 is

 node 1 via 1 Distance 41
 node 2 via 2 Distance 24
 node 3 via 2 Distance 42
 node 4 via 4 Distance 0
```



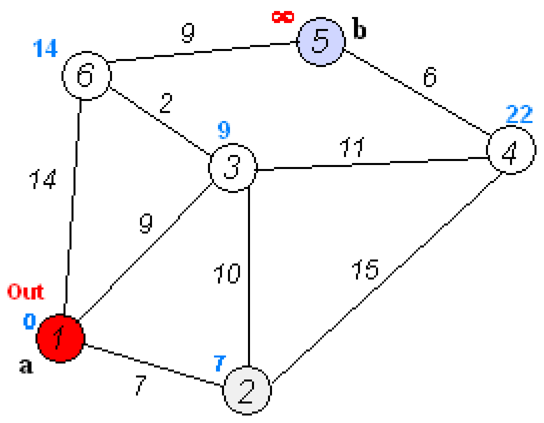
```
enter the value of i & j: 1
3
enter the new cost: 68
After update


 State value for router 1 is

 node 1 via 1 Distance 0
 node 2 via 2 Distance 12
 node 3 via 3 Distance 9
 node 4 via 4 Distance 16

 State value for router 2 is

 node 1 via 1 Distance 20
 node 2 via 2 Distance 0
 node 3 via 3 Distance 18
 node 4 via 3 Distance 30

 State value for router 3 is

 node 1 via 1 Distance 15
 node 2 via 2 Distance 8
 node 3 via 3 Distance 0
 node 4 via 2 Distance 12

 State value for router 4 is

 node 1 via 1 Distance 41
 node 2 via 2 Distance 24
 node 3 via 2 Distance 42
 node 4 via 4 Distance 0

student@RnDLab20:~$
```

RESULT: Thusimplementation of Distance vector algorithm is done successfully

EXPERIMENT NO. 4:

Dijkstra's algorithm to compute the shortest routing path.

AIM:Implement Dijkstra's algorithm to compute the Shortest path thru a given graph.



THEORY:

The Dijkstra's algorithm finds the shortest path from a particular node, called the source node to every other node in a connected graph. It produces a shortest path tree with the source node as the root. It is profoundly used in computer networks to generate optimal routes with the aim of minimizing routing costs.

ALGORITHM:

Input − A graph representing the network; and a source node, s

Output − A shortest path tree, spt[], with s as the root node.

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.

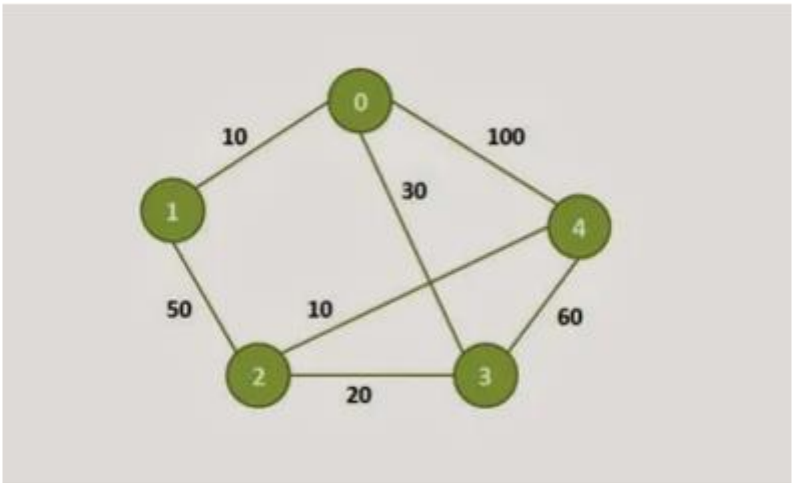
– Recalculate the shortest distance of remaining vertices from the source.

– 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])
```

Time Complexity

The program contains two nested loops each of which has a complexity of O(n). n is number of vertices. So the complexity of algorithm is O(n2).



PLATFORM: GCC COMPILER (VI EDITOR[UBUNTU])

PROGRAM:

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

void dijkstra(int G[MAX][MAX],int n,int startnode);

int main()
{
int G[MAX][MAX],i,j,n,u;
printf("Enter no. of vertices:");
scanf("%d",&n);
```

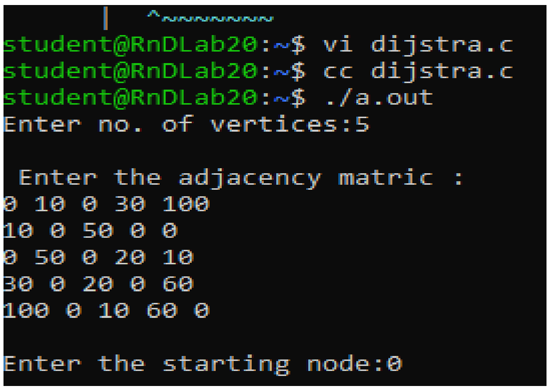
```
printf("\nEnter the adjacency matrix:\n");
for(i=0;i<n;i++)
for(j=0;j<n;j++)
scanf("%d",&G[i][j]);
printf("\nEnter the starting node:");
scanf("%d",&u);
dijkstra(G,n,u);
return 0;
}
 void dijkstra(int G[MAX][MAX],int n,int startnode)
{
 int cost[MAX][MAX],distance[MAX],pred[MAX];
int visited[MAX],count,mindistance,nextnode,i,j;
//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];
//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);
}
}
```

STEPS OF EXECUTION:

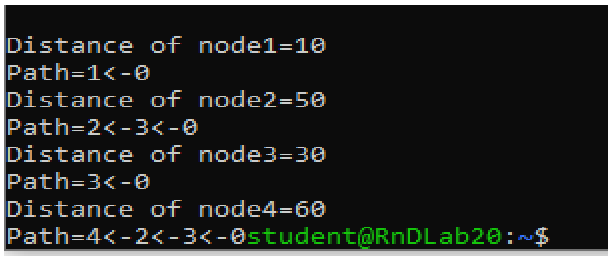
1. Open the vi editor (vi <filename.c>)
1. Write the code and save
2. Run cc <filename.c>
3. ./a.out

INPUT:



```
     |   ^~~~~~~~
student@RnDLab20:~$ vi dijstra.c
student@RnDLab20:~$ cc dijstra.c
student@RnDLab20:~$ ./a.out
Enter no. of vertices:5

 Enter the adjacency matric :
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
```

OUTPUT:



```
Distance of node1=10
Path=1<-0
Distance of node2=50
Path=2<-3<-0
Distance of node3=30
Path=3<-0
Distance of node4=60
Path=4<-2<-3<-0student@RnDLab20:~$
```

RESULT: Thus implementation of Dijkstra‘s algorithm to compute the Shortest path for a given graph is successfully done.

EXPERIMENT NO.5

Simulation of network topologies

AIM: Simulate the different network topologies using packet tracer

THEORY:

INTRODUCTION TO PACKET TRACER

This is a tool built by Cisco. This tool provides a network simulation to practice simple and complex networks.

To download: https://www.netacad.com/courses/packet-tracer/introduction-packet-tracer.

Workspace:

1. Logical –

Logical workspace shows the logical network topology of the network the user has built. It represents the placing, connecting and clustering virtual network devices.

2. Physical –

Physical workspace shows the graphical physical dimension of the logical network. It depicts the scale and placement in how network devices such as routers, switches and hosts would look in a real environment. It also provides geographical representation of networks, including multiple buildings, cities and wiring closets.

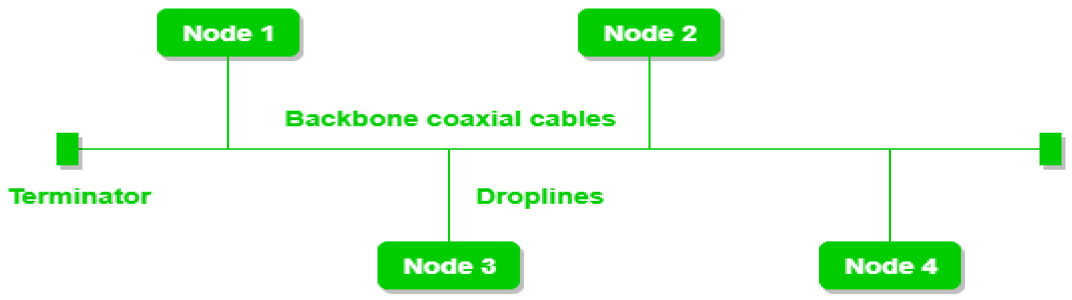
TOPOLOGY: The arrangement of a network that comprises nodes and connecting lines via sender and receiver is referred to as network topology.

5.1 BUS TOPOLOGY

AIM: Implement Bus topology using Packet tracer

THEORY:

Bus topology is a network type in which every computer and network device is connected to a single cable. It transmits the data from one end to another in a single direction. No bi-directional feature is in bus topology. It is a multi-point connection and a non-robust topology because if the backbone fails the topology crashes.



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 a 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.

The cost of the cable is less as compared to other topologies, but it is used to build 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 the MAC layer known as Pure Aloha, Slotted Aloha, CSMA/CD, etc.

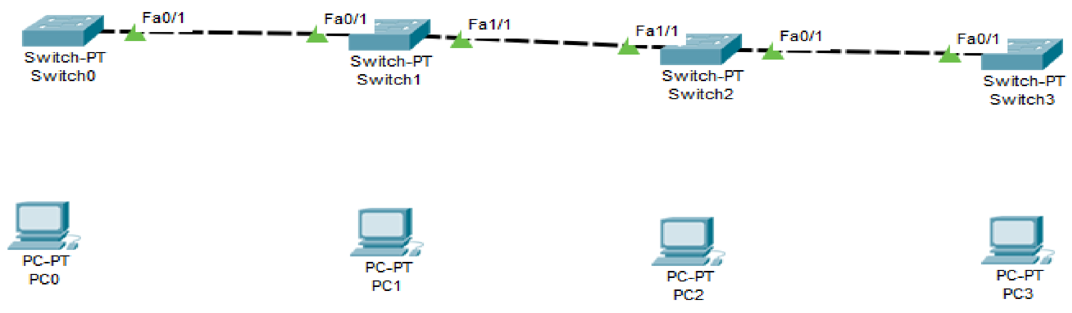
Security is very low.

PROGRAM

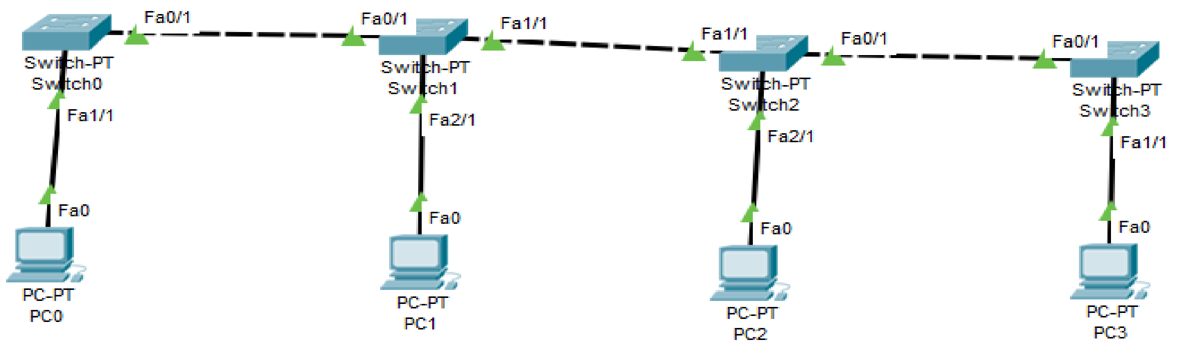
STEP 1: Take four devices (PC's)

STEP 2: Take four switches (one for each device)

STEP 3: Click on Connections. Connect between two switches with fast Ethernet. Likewise connect all the switches (dotted black line – Copper Cross Over)



STEP 4: Connect the switch with the respective PC's (solid black line- Copper Straight through)



STEP 5: Assign IP address

Double click on the device. A pop up window will get open. Click on Desktop and IP Configuration

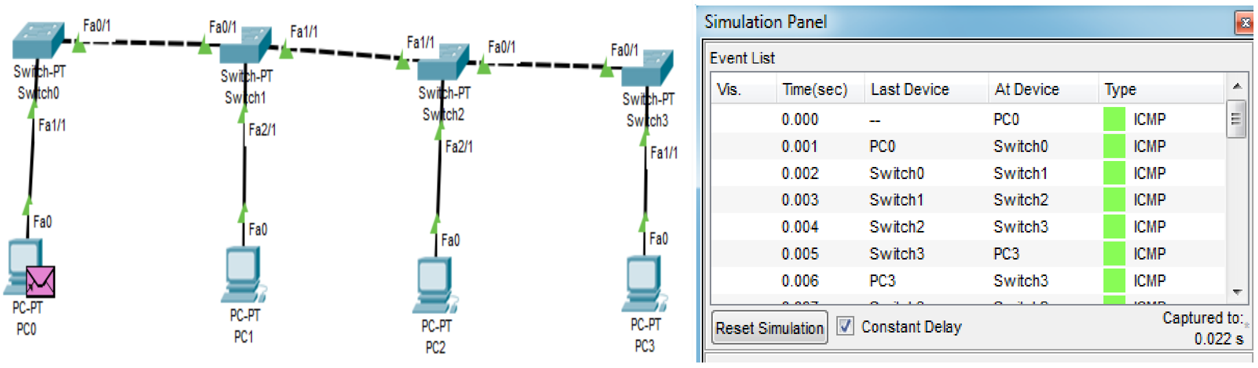
PC0: IP Address – 192.168.0.1   Subnet mask –255.255.255.0

PC1: IP Address – 192.168.0.2   Subnet mask –255.255.255.0

PC2: IP Address – 192.168.0.3   Subnet mask –255.255.255.0

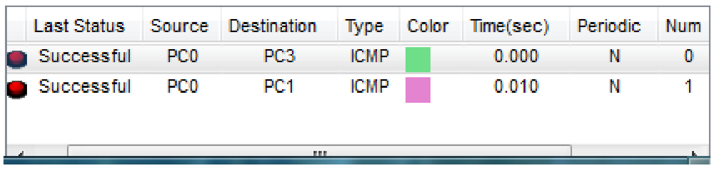
PC3: IP Address – 192.168.0.4   Subnet mask –255.255.255.0

STEP 6: Select a packet and check whether it is delivered from PC0 to PC3



Vis.	Time(sec)	Last Device	At Device	Type
	0.000	--	PC0	🟩 ICMP
	0.001	PC0	Switch0	🟩 ICMP
	0.002	Switch0	Switch1	🟩 ICMP
	0.003	Switch1	Switch2	🟩 ICMP
	0.004	Switch2	Switch3	🟩 ICMP
	0.005	Switch3	PC3	🟩 ICMP
	0.006	PC3	Switch3	🟩 ICMP

Captured to: 0.022 s



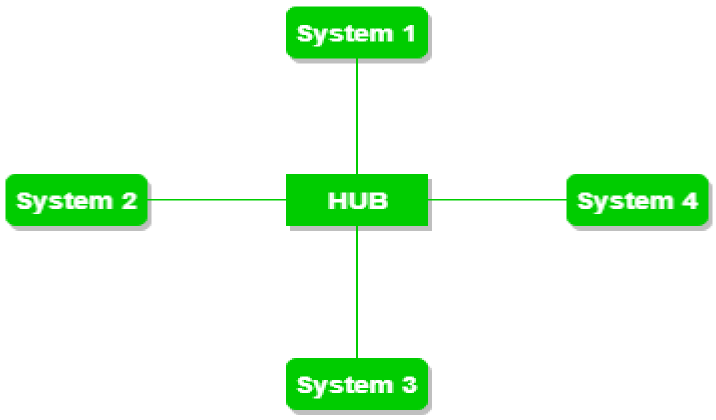
Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num
Successful	PC0	PC3	ICMP	🟩	0.000	N	0
Successful	PC0	PC1	ICMP	🟪	0.010	N	1

5.2 STAR TOPOLOGY

AIM: Implement star topology using packet tracer

THEORY:

In star topology, all the devices are connected to a single hub through a cable. This hub is the central node and all other 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 a star topology, then the number of cables required to connect them is N. So, it is easy to set up.

Each device requires only 1 port i.e. to connect to the hub, therefore total number of ports required is N.

Problems with this topology:

If the concentrator (hub) on which the whole topology relies fails, the whole system will crash down.

The cost of installation is high.

Performance is based on the single concentrator i.e. hub.

PROGRAM

STEP 1: Take four devices (PC's)

STEP 2: Take one switch

STEP 3: Click on Connections. Connect all the PC's to the switch

STEP 4: Assign IP address

Double click on the device. A pop up window will get open. Click on Desktop and IP Configuration

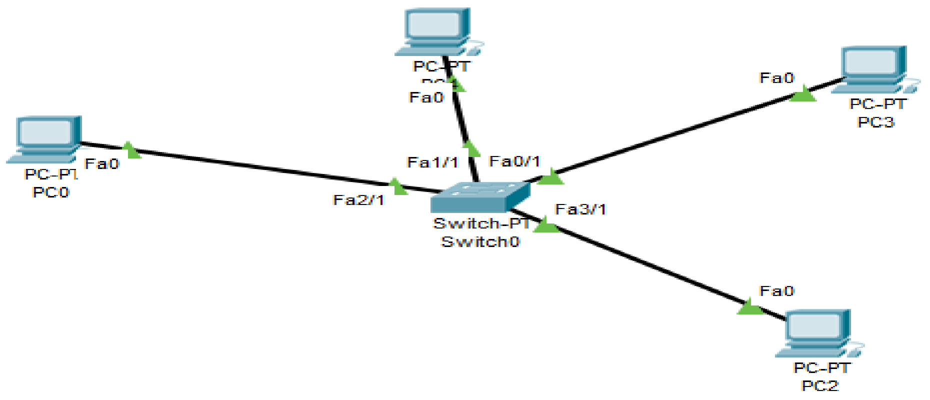
PC0: IP Address – 192.168.0.1   Subnet mask –255.255.255.0

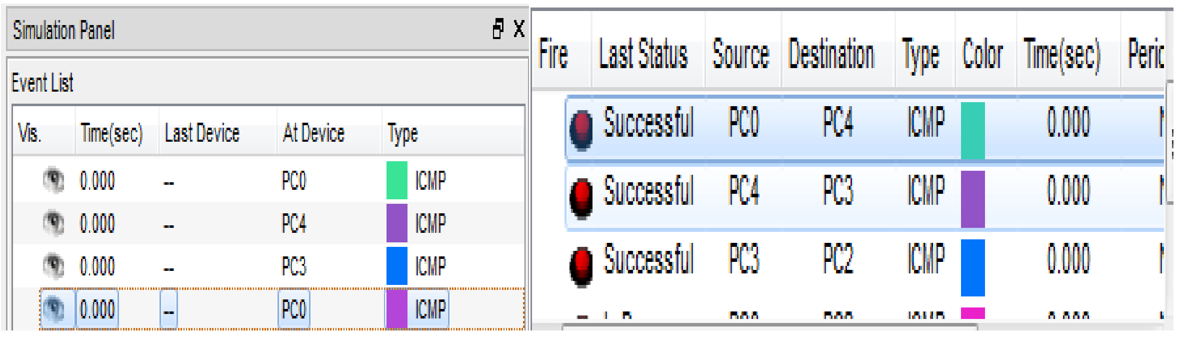
PC1: IP Address – 192.168.0.2   Subnet mask –255.255.255.0

PC2: IP Address – 192.168.0.3   Subnet mask –255.255.255.0

PC3: IP Address – 192.168.0.4   Subnet mask –255.255.255.0

STEP 5: Select a packet ping all PCs

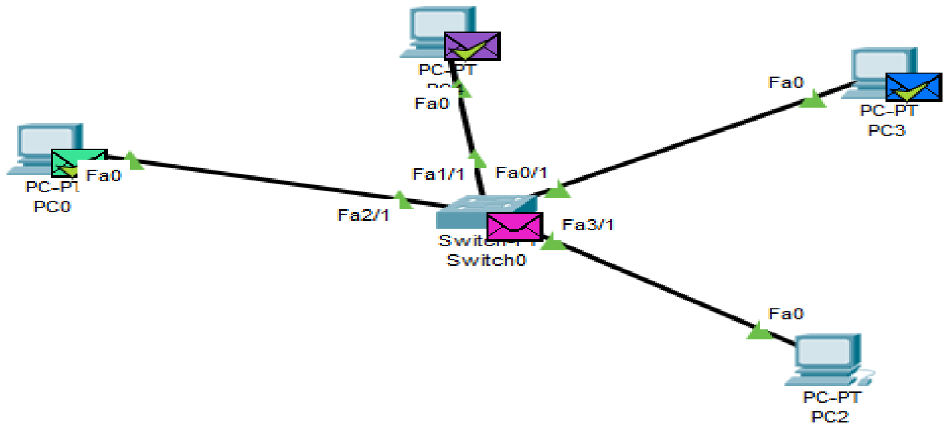



Simulation Panel

Event List

Vis.	Time(sec)	Last Device	At Device	Type
	0.000	--	PC0	ICMP
	0.000	--	PC4	ICMP
	0.000	--	PC3	ICMP
	0.000	--	PC0	ICMP

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Peric
●	Successful	PC0	PC4	ICMP	🟩	0.000	
●	Successful	PC4	PC3	ICMP	🟪	0.000	
●	Successful	PC3	PC2	ICMP	🟦	0.000	

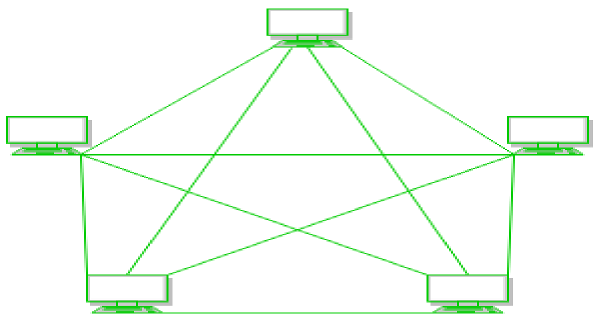


5.3 MESH TOPOLOGY

AIM: Implement mesh topology using packet tracer

THEORY:

In a mesh topology, every device is connected to another device via the 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 a mesh topology, the total number of ports that are required by each device is N-1. In Figure 1, there are 5 devices connected to each other, hence the total number of ports required by each device is 4. Total number of ports required=N*(N-1).
If suppose, N number of devices are connected with each other in a mesh topology, then a total number of dedicated links required to connect them is NC2 i.e. N(N-1)/2. In Figure 1, there are 5 devices connected to each other, hence the total number of links required is 5*4/2 = 10.

Advantages of this topology :

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

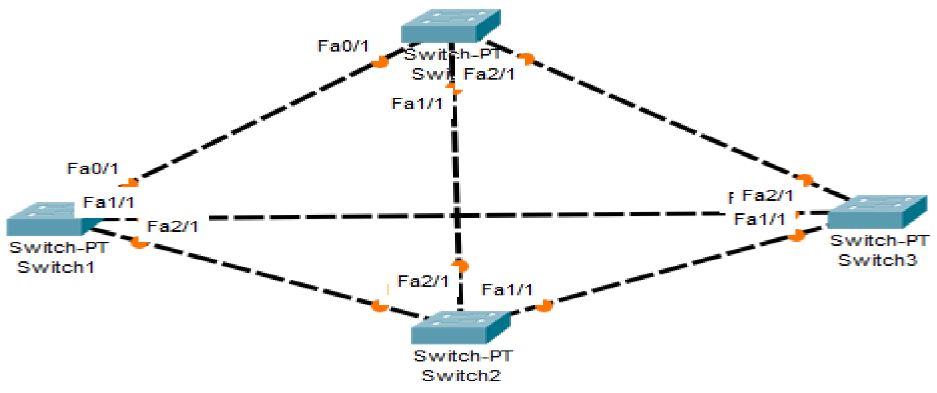
Problems with this topology :

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

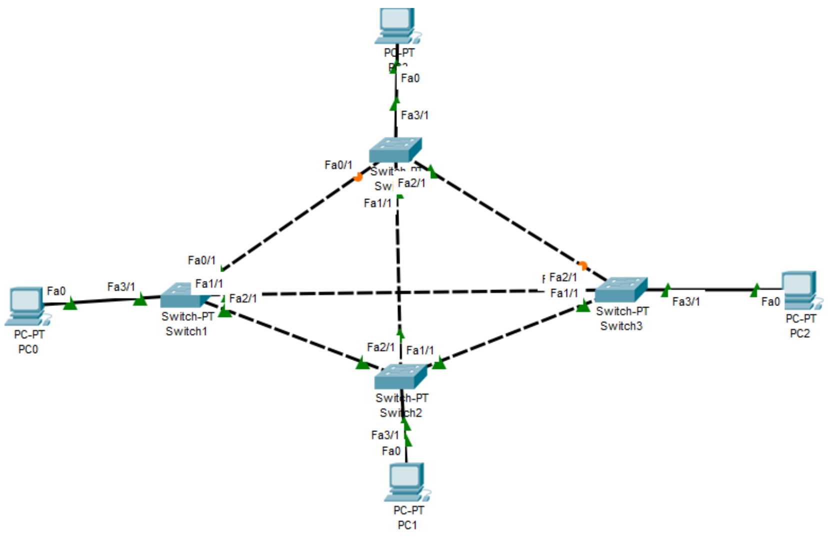
PROGRAM

STEP 1: Take four switch

STEP 2: Connect all the switches to one another



STEP 3: Take some PC's and attach to switch



STEP 4: Assign IP address

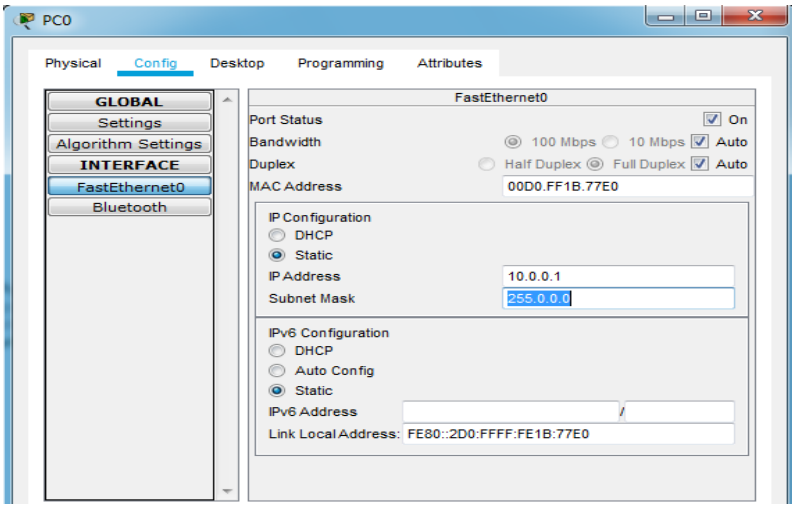
Double click on the device. A pop up window will get open. Click on Config then fast Ethernet 0 and assign IP addresses

PC0: IP Address – 10.0.0.1   Subnet mask –255.0.0.0

PC1: IP Address – 10.0.0.2   Subnet mask –255.0.0.0

PC2: IP Address – 10.0.0.3   Subnet mask –255.0.0.0

PC3: IP Address – 10.0.0.4   Subnet mask –255.0.0.0



PC0

Physical | Config | Desktop | Programming | Attributes

GLOBAL
Settings
Algorithm Settings
INTERFACE
FastEthernet0
Bluetooth

FastEthernet0

Port Status	[✓ On]
Bandwidth	100 Mbps, 10 Mbps, [✓ Auto]
Duplex	Half Duplex, Full Duplex, [✓ Auto]
MAC Address	00D0.FF1B.77E0

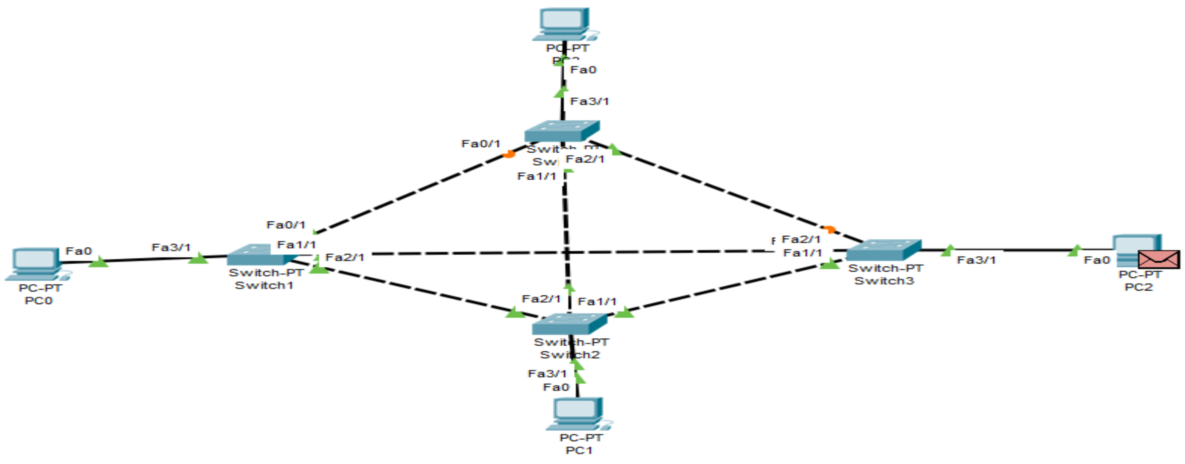
IP Configuration

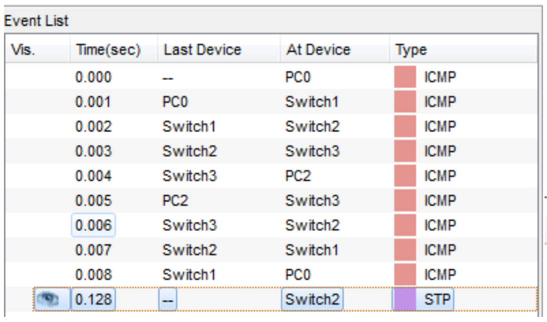
	DHCP [✓ Static]
IP Address	10.0.0.1
Subnet Mask	255.0.0.0

IPv6 Configuration

	DHCP, Auto Config, [✓ Static]
IPv6 Address	/
Link Local Address:	FE80::2D0:FFFF:FE1B:77E0

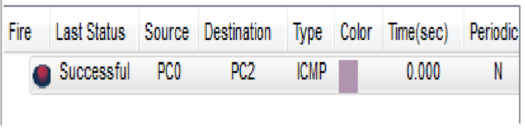
STEP 5: Select a packet(PDU) ping from PC0 to PC2(for ex)





Event List

Vis.	Time(sec)	Last Device	At Device	Type
	0.000	--	PC0	ICMP
	0.001	PC0	Switch1	ICMP
	0.002	Switch1	Switch2	ICMP
	0.003	Switch2	Switch3	ICMP
	0.004	Switch3	PC2	ICMP
	0.005	PC2	Switch3	ICMP
	0.006	Switch3	Switch2	ICMP
	0.007	Switch2	Switch1	ICMP
	0.008	Switch1	PC0	ICMP
	0.128	--	Switch2	STP



Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic
●	Successful	PC0	PC2	ICMP	🟪	0.000	N

5.4 RING TOPOLOGY

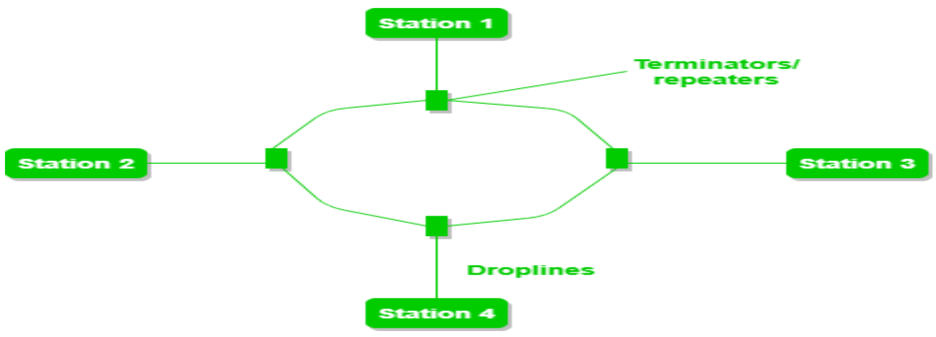
AIM: Implement ring topology using packet tracer

THEORY:

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

A number of repeaters are used for Ring topology with a 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.

The transmission is unidirectional, but it can be made bidirectional by having 2 connections between each Network Node, it is called Dual Ring Topology.



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

The following operations take place in ring topology are :

One station is known as **monitor** station which takes all the responsibility to perform the operations.

To transmit the data, the station has to hold the token. After the transmission is done, the token is to be released for other stations to use.

When no station is transmitting the data, then the token will circulate in the ring.

There are two types of token release techniques: **Early token release** releases the token just after transmitting the data and **Delay token release** releases the token after the acknowledgment 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.

The addition of stations in between or removal of stations can disturb the whole topology.

Less secure.

PROGRAM

STEP 1: Take four devices (PCs or laptop) and four switch

STEP 2: Connect all the switches to one another using cable (Copper Cross Over)

STEP 3: Connect PCs to switch using cable (Copper Straight)

STEP 4: Assign IP address

Double click on the device. A pop up window will get open. Click on Config then fast Ethernet 0 and assign IP addresses

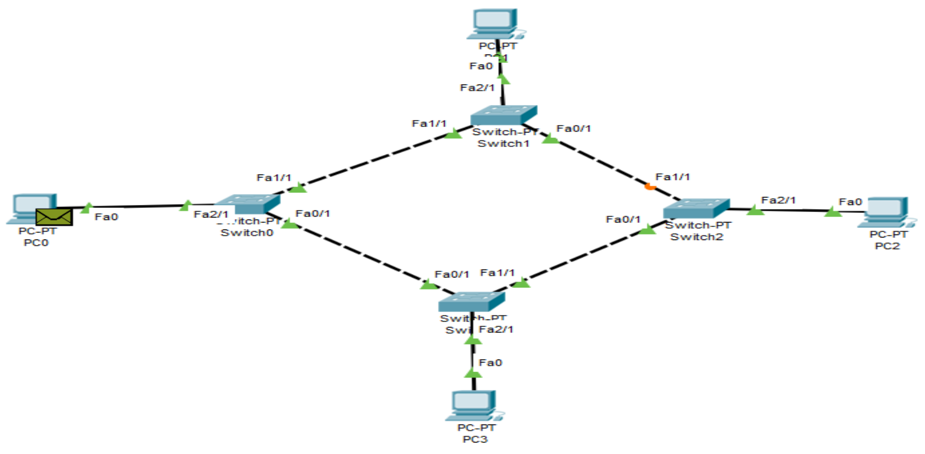
PC0: IP Address – 10.0.0.1 Subnet mask –255.0.0.0

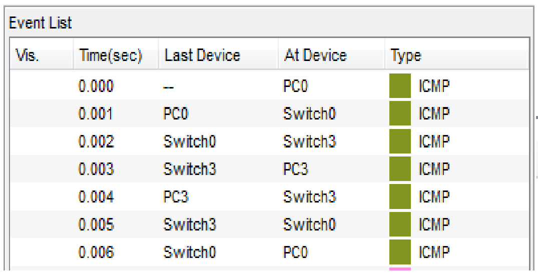
PC1: IP Address – 10.0.0.2 Subnet mask –255.0.0.0

PC2: IP Address – 10.0.0.3 Subnet mask –255.0.0.0

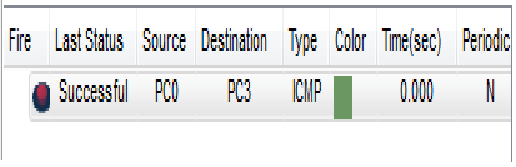
PC3: IP Address – 10.0.0.4 Subnet mask –255.0.0.0

STEP 5: Send a PDU from PC0 to PC2(for ex)




Event List

Vis.	Time(sec)	Last Device	At Device	Type
	0.000	--	PC0	🟩 ICMP
	0.001	PC0	Switch0	🟩 ICMP
	0.002	Switch0	Switch3	🟩 ICMP
	0.003	Switch3	PC3	🟩 ICMP
	0.004	PC3	Switch3	🟩 ICMP
	0.005	Switch3	Switch0	🟩 ICMP
	0.006	Switch0	PC0	🟩 ICMP



Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic
●	Successful	PC0	PC3	ICMP	🟩	0.000	N

RESULT: Thus simulation of different network topologies like star, bus, ring, mesh is done successfully.

EXPERIMENT NO.6

Configuration of a network using different routing protocols.

AIM: Configuration of a network using different routing protocols.

THEORY:

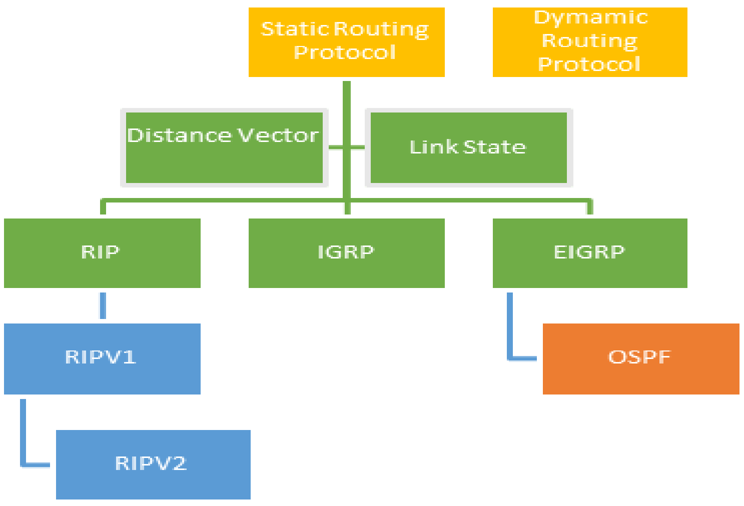
Routing Protocols are the set of defined rules used by the routers to communicate between source & destination. They do not move the information to the source to a destination, but only update the routing table that contains the information.

Network Router protocols helps you to specify way routers communicate with each other. It allows the network to select routes between any two nodes on a computer network.

Types of Routing Protocols

There are mainly two types of Network Routing Protocols

1. Static
2. Dynamic



Static Routing Protocols

Static routing protocols are used when an administrator manually assigns the path from source to the destination network. It offers more security to the network.

Advantages

- No overhead on router CPU.
- No unused bandwidth between links.
- Only the administrator is able to add routes

Disadvantages

- The administrator must know how each router is connected.
- Not an ideal option for large networks as it is time intensive.
- Whenever link fails all the network goes down which is not feasible in small networks.

Dynamic Routing Protocols

Dynamic routing protocols are another important type of routing protocol. It helps routers to add information to their routing tables from connected routers automatically. These types of protocols also send out topology updates whenever the network changes' topological structure.

Advantage:

- Easier to configure even on larger networks.
- It will be dynamically able to choose a different route in case if a link goes down.
- It helps you to do load balancing between multiple links.

Disadvantage:

- Updates are shared between routers, so it consumes bandwidth.
- Routing protocols put an additional load on router CPU or RAM.

6.1 OPEN SHORTEST PATH FIRST (OSPF) ALGORITHM

AIM: Implement Link state routing algorithm using packet tracer

THEORY:

Open Shortest Path First (OSPF) is a link-state routing protocol that is used to find the best path between the source and the destination router using its own Shortest Path First). OSPF is developed by Internet Engineering Task Force (IETF) as one of the Interior Gateway Protocol (IGP), i.e, the protocol which aims at moving the packet within a large autonomous system or routing domain. It is a network layer protocol which works on the protocol number 89 and uses AD value 110. OSPF uses multicast address 224.0.0.5 for normal communication and 224.0.0.6 for update to designated router(DR)/Backup Designated Router (BDR).

OSPF terms –

1. Router I'd – It is the highest active IP address present on the router. First, highest loopback address is considered. If no loopback is configured then the highest active IP address on the interface of the router is considered.Router priority – It is a 8 bit value assigned to a router operating OSPF, used to elect DR and BDR in a broadcast network.

2. Designated Router (DR) – It is elected to minimize the number of adjacency formed. DR distributes the LSAs to all the other routers. DR is elected in a broadcast network to which all the other routers shares their DBD. In a broadcast network, router requests for an update to DR and DR will respond to that request with an update.
3. Backup Designated Router (BDR) – BDR is backup to DR in a broadcast network. When DR goes down, BDR becomes DR and performs its functions.

PROGRAM

STEP1 **:** Take three generic routers(Router PT(generic router))

STEP 2: Take two generic computers

STEP 3: Establish the connections

STEP 4: Assign the IP Address

Double click on the device. A pop up window will get open. Click on Config then fast Ethernet 0 and assign IP addresses

PC0: IP Address – 192.168.1.2 Subnet mask –255.255.255.0 Default gateway:192.168.1.1

PC1: IP Address – 192.168.2.2 Subnet mask –255.255.255.0 Default gateway:192.168.2.1

STEP 5:Configure the routers and serial ports

Double click on the router. Go to config then to FastEthernet 0, and then IP Congiguration

Set it ON

- Router1: 192.168.1.1   255.255.255.0

Select Serial2/0 (connection between the routers)

Click on ON

IP address: 10.0.0.2   255.0.0.0

Set the clock to 64000

Select Serial3/0 (connection between the routers)

Click on ON

IP address: 12.12.0.2   255.0.0.0

Set the clock to 64000

- Router2: No fast Ethernet connection

Select Serial2/0 (connection between the routers)

Click on ON(Port Status)

IP address: 10.10.0.3   255.0.0.0

Set the clock to 64000

Select Serial3/0 (connection between the routers)

Click on ON(Port Status)

IP address: 11.11.0.2   255.0.0.0

Set the clock to 64000

- Router3: 192.168.2.1   255.255.255.0

Select Serial2/0 (connection between the routers)

Click on ON(Port Status)

IP address: 11.11.0.3   255.0.0.0

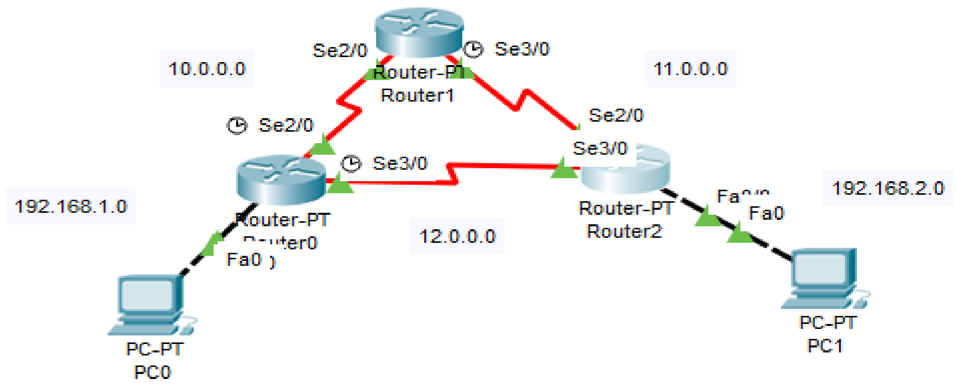
Set the clock to 64000

Select Serial3/0 (connection between the routers)

Click on ON(Port Status)

IP address: 12.12.0.3   255.0.0.0

Set the clock to 64000



STEP 6: Configure the network with OSPF

a. Select Router0 , select CLI(command line interface)

```
 exit
 Router(config)#router ospf 1  // goes into ospf configuration with process id 1
 // For connection with PC and other routers (Adding the network)
//Command is network <ip address><wild card mask →which parts of ip address are
// available (complement of subnet mask)
 Router(config-router)#network 192.168.1.0 0.0.0.255 area 0  //PC to router 0
 Router(config-router)#network 10.0.0.0 0.255.255.255 area 0 //Router 0 to router 1
 Router(config-router)#network 12.0.0.0 0.255.255.255 area 0 // Router 0 to router 2
 Router(config-router)# exit
```

Go to Config → Settings → Save→Close

b. Select Router1 , select CLI(command line interface)

```
 Router(config-if)#exit
 Router(config)#router ospf 1
Router(config-router)#network 10.0.0.0 0.255.255.255 area 0
    Router(config-router)#network 11.0.0.0 0.255.255.255 area 0
    Router(config-router)#exit
```

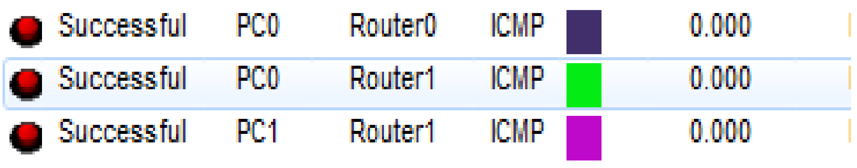
Go to Config → Settings → Save→Close

c. Select Router2 , select CLI(command line interface)

```
        Router(config-if)#exit
        Router(config)#router ospf 1
        Router(config-router)#network 192.168.2.0 0.0.0.255 area 0
 Router(config-router)#network 11.0.0.0 0.255.255.255 area 0
 Router(config-router)#network 12.0.0.0 0.255.255.255 area 0
Router(config-router)#exit
```

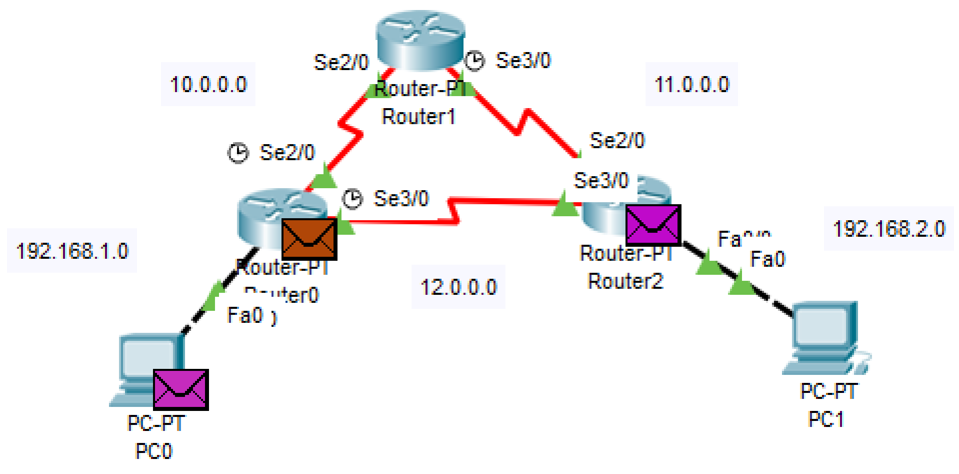
Go to Config → Settings → Save→Close

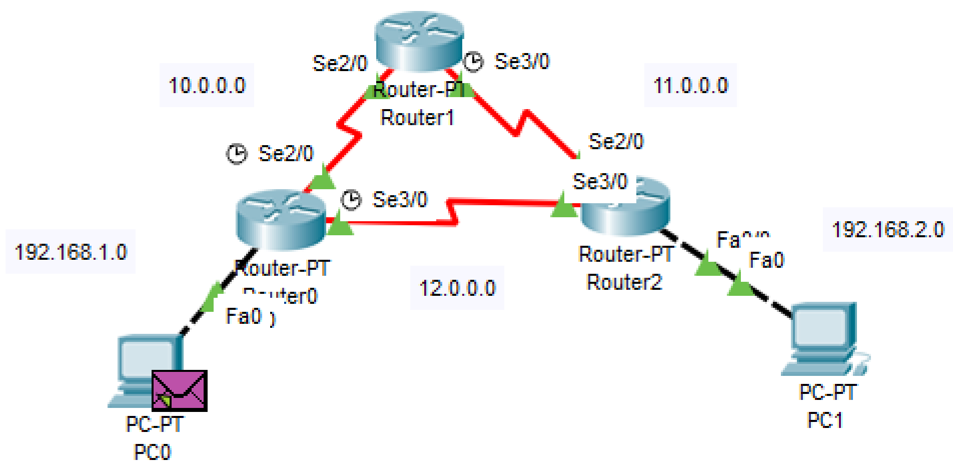
STEP 7: Drop the packets from PC to router and check whether they are successful or not

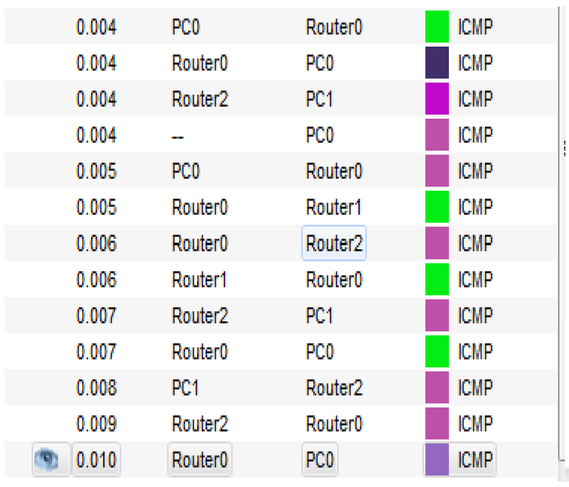


●	Successful	PC0	Router0	ICMP	■	0.000
●	Successful	PC0	Router1	ICMP	🟩	0.000
●	Successful	PC1	Router1	ICMP	■	0.000

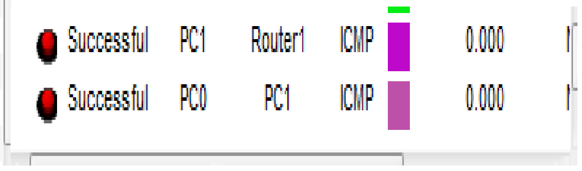
STEP 8: Check whether the packet is sent from PC0 to PC1







0.004	PC0	Router0	🟩	ICMP
0.004	Router0	PC0	🟪	ICMP
0.004	Router2	PC1	🟪	ICMP
0.004	--	PC0	🟪	ICMP
0.005	PC0	Router0	🟪	ICMP
0.005	Router0	Router1	🟩	ICMP
0.006	Router0	Router2	🟪	ICMP
0.006	Router1	Router0	🟩	ICMP
0.007	Router2	PC1	🟪	ICMP
0.007	Router0	PC0	🟩	ICMP
0.008	PC1	Router2	🟪	ICMP
0.009	Router2	Router0	🟪	ICMP
0.010	Router0	PC0	🟪	ICMP



●	Successful	PC1	Router1	ICMP	🟪	0.000
●	Successful	PC0	PC1	ICMP	🟪	0.000

6.2 RIP (ROUTING INFORMATION PROTOCOL)

AIM: Implement RIP Routing Protocol using packet tracer

THEORY:

Routing Information Protocol (RIP) is **a distance-vector routing protocol**. Routers running the distance-vector protocol send all or a portion of their routing tables in routing-update messages to their neighbors. You can use RIP to configure the hosts as part of a RIP network.

Hop Count:

Hop count is the number of routers occurring in between the source and destination network. The path with the lowest hop count is considered as the best route to reach a network and therefore placed in the routing table. RIP prevents routing loops by limiting the number of hopes allowed in a path from source and destination. The maximum hop count allowed for RIP is 15 and hop count of 16 is considered as network unreachable.

Features of RIP:

1**.** Updates of the network are exchanged periodically.

2. Updates (routing information) are always broadcast.

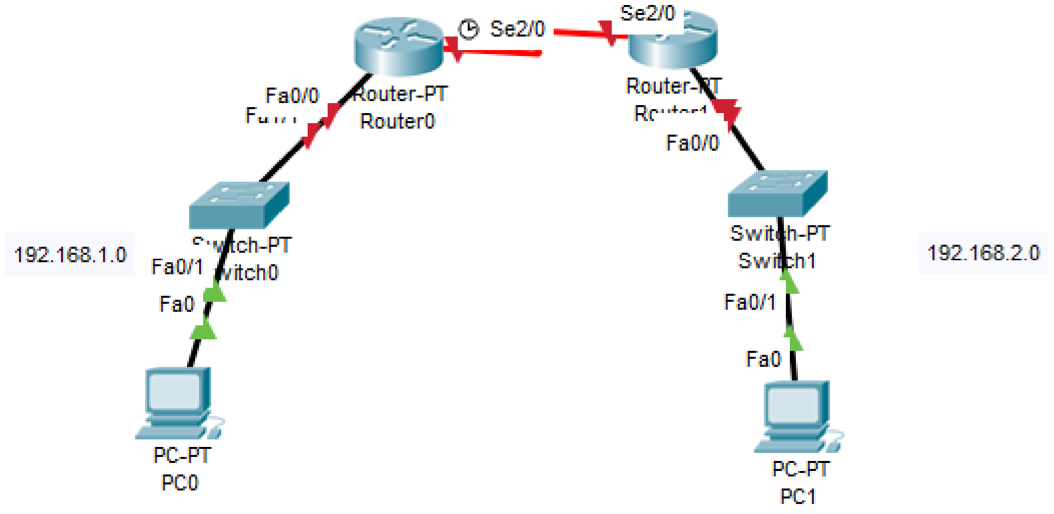
3. Full routing tables are sent in updates.

4. Routers always trust on routing information received from neighbor routers. This is also known as Routing on rumours.

PROGRAM

STEP 1: Take two PC's , two routers and two switches

STEP 2: Connect PC to switch and router. Connect the router



STEP 3: Configure the network

a. Assign the IP Address

Double click on the device. A pop up window will get open. Click on Config then fast Ethernet 0 and assign IP addresses

PC0: IP Address – 192.168.1.2 Subnet mask –255.255.255.0 Default gateway:192.168.1.1

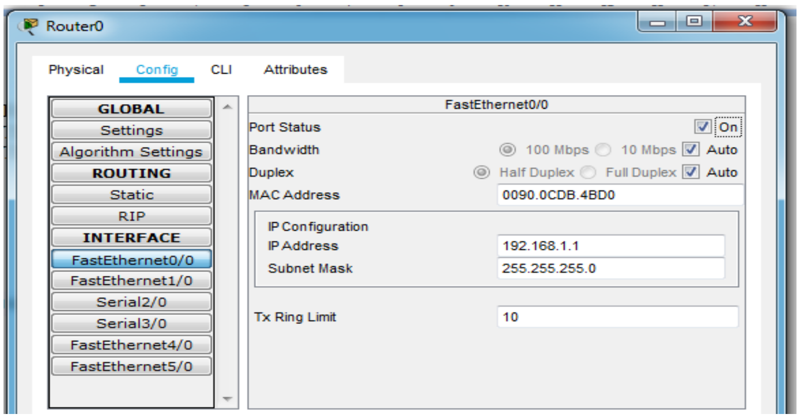
PC1: IP Address – 192.168.2.2 Subnet mask –255.255.255.0 Default gateway:192.168.2.1

b. Configure router

Router 0

Go to Config, Fast Ethernet 0 , Assign the IP Address

192.168.1.1 255.255.255.0



FastEthernet0/0

Port Status ✓ On

Bandwidth ● 100 Mbps ○ 10 Mbps ✓ Auto

Duplex ● Half Duplex ○ Full Duplex ✓ Auto

MAC Address 0090.0CDB.4BD0

IP Configuration

IP Address 192.168.1.1

Subnet Mask 255.255.255.0

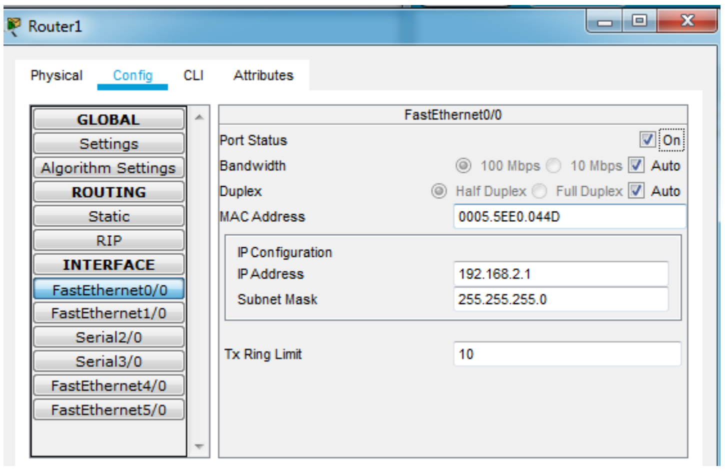
Tx Ring Limit 10

Click on ON

Router 1

Go to Config, Fast Ethernet 0 , Assign the IP Address

192.168.2.1 255.255.255.0



FastEthernet0/0

Port Status ✓ On

Bandwidth ● 100 Mbps ○ 10 Mbps ✓ Auto

Duplex ● Half Duplex ○ Full Duplex ✓ Auto

MAC Address 0005.5EE0.044D

IP Configuration

IP Address 192.168.2.1

Subnet Mask 255.255.255.0

Tx Ring Limit 10

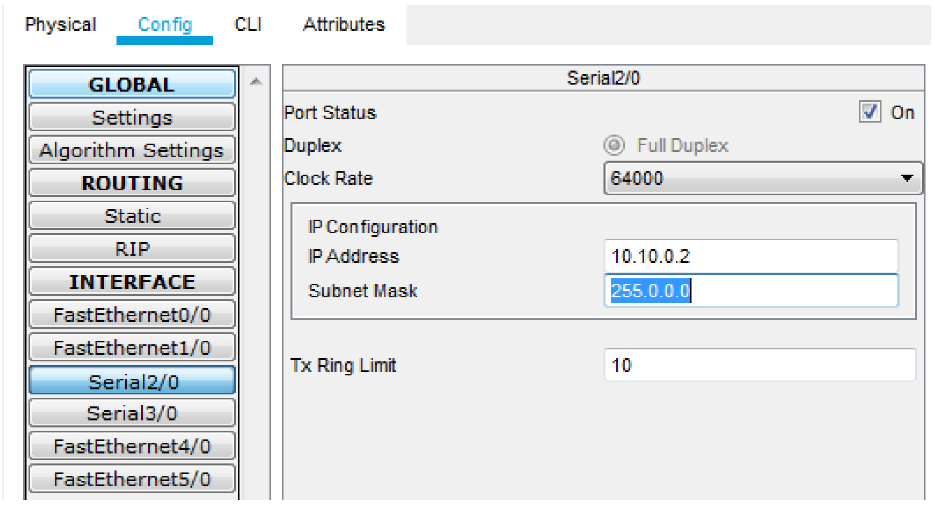
Click on ON

Configure the network between two Routers(10.0.0.0)

Click on Router 0

Select Config, Serial 2/0 and assign the IP Address 10.10.0.2 255.0.0.0

Clock rate 64000 Port Status → ON

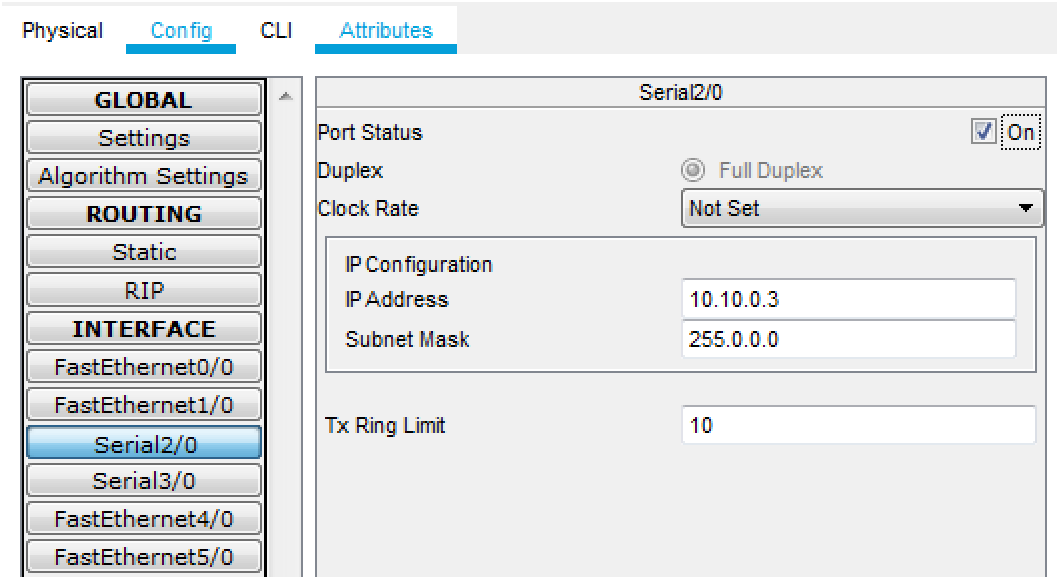


Serial2/0
Port Status ☑ On
Duplex ● Full Duplex
Clock Rate 64000
IP Configuration
IP Address 10.10.0.2
Subnet Mask 255.0.0.0
Tx Ring Limit 10

Click on Router 1

Select Config, Serial 2/0 and assign the IP Address 10.10.0.3 255.0.0.0

Clock rate → Not Set Port Status → ON

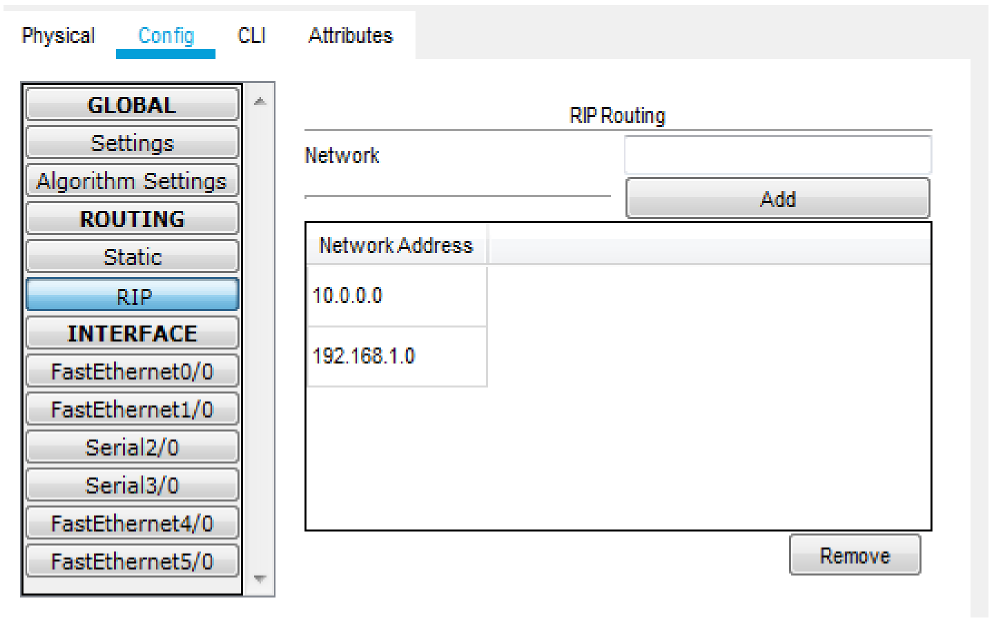


Serial2/0
Port Status ☑ On
Duplex ● Full Duplex
Clock Rate Not Set
IP Configuration
IP Address 10.10.0.3
Subnet Mask 255.0.0.0
Tx Ring Limit 10

STEP 4: Configure the routers to use RIP and to know other network

Click on Router 0 → Config → RIP

In network tab add the networks id which the router knows



RIP Routing

Network

Network Address
10.0.0.0
192.168.1.0

Click on Setting and Save option

Click on Router 1 → CLI

exit

Router(config)#router rip

Router(config-router)#network 192.168.2.0

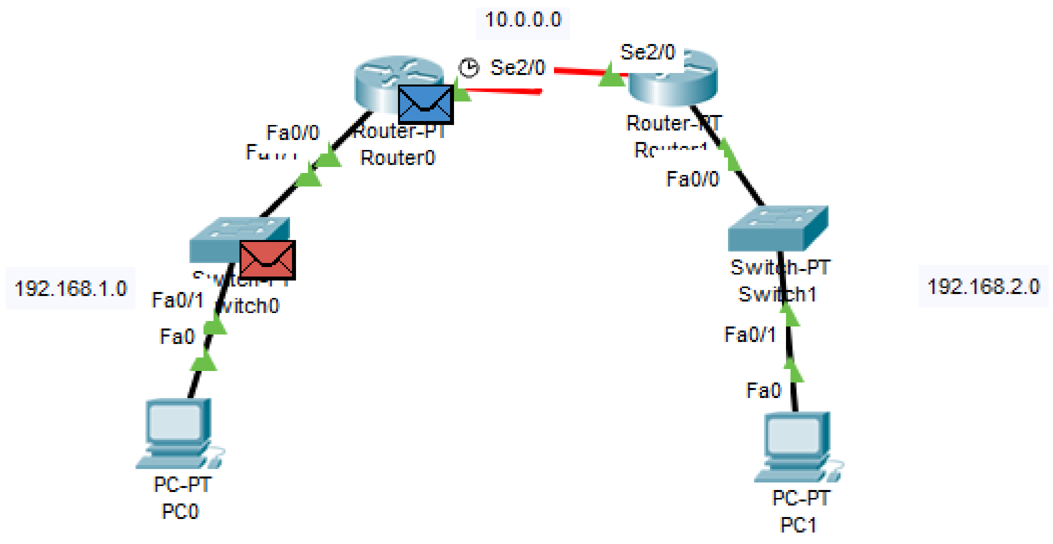
Router(config-router)#network 10.0.0.0

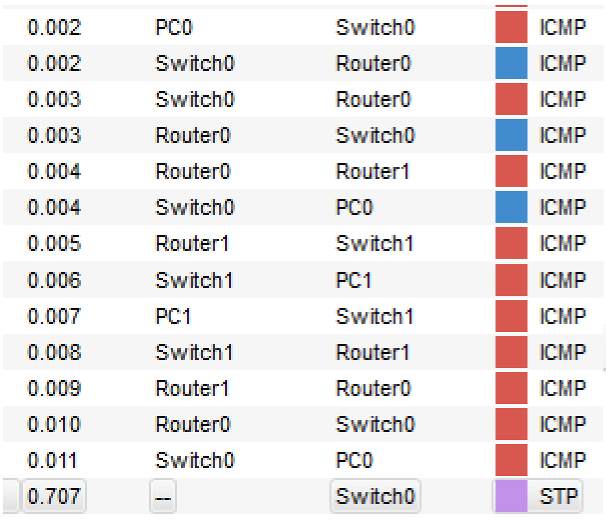
Router(config-router)#exit

Router(config)#

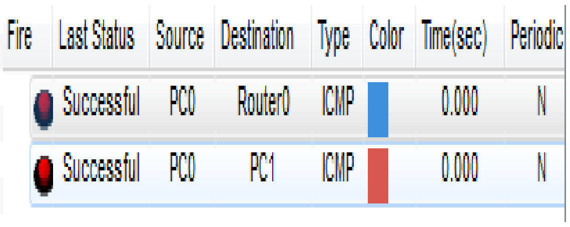
Click on Setting and Save option

STEP 5: Check whether the packet is routed successfully from one machine to another





0.002	PC0	Switch0	🟥	ICMP
0.002	Switch0	Router0	🟦	ICMP
0.003	Switch0	Router0	🟥	ICMP
0.003	Router0	Switch0	🟦	ICMP
0.004	Router0	Router1	🟥	ICMP
0.004	Switch0	PC0	🟦	ICMP
0.005	Router1	Switch1	🟥	ICMP
0.006	Switch1	PC1	🟥	ICMP
0.007	PC1	Switch1	🟥	ICMP
0.008	Switch1	Router1	🟥	ICMP
0.009	Router1	Router0	🟥	ICMP
0.010	Router0	Switch0	🟥	ICMP
0.011	Switch0	PC0	🟥	ICMP
0.707	--	Switch0	🟪	STP



Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic
●	Successful	PC0	Router0	ICMP	🟦	0.000	N
●	Successful	PC0	PC1	ICMP	🟥	0.000	N

RESULT:Thus the configuration of a network using different routing protocols is done successfully.

PART C

Simulate experiments using NS2/ NS3/ NCTUNS / NetSim/ or any other equivalent tool

Introduction to NS2

THEORY:

Network Simulator (Version 2), widely known as NS2, an event driven simulation tool that has proved useful in studying the dynamic nature of communication networks. Simulation of wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be done using NS2. NS2 provides users with a way of specifying such network protocols and simulating their corresponding behaviours. Due to its flexibility and modular nature, NS2 has gained constant popularity in the networking research community since its birth in 1989. University of California and Cornell University developed the REAL network simulator,1 the foundation which NS is based on. Since 1995 the Defense Advanced Research Projects Agency (DARPA) supported development of NS through the Virtual Inter Network Testbed (VINT) project. Currently the National Science Foundation (NSF) has joined the ride in development.

BASIC ARCHITECTURE:

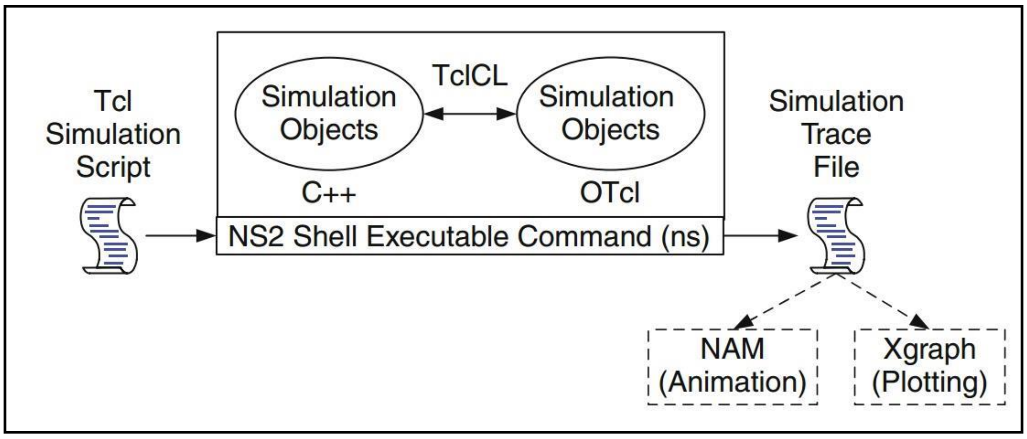


Figure shows the basic architecture of NS2. NS2 provides users with an executable command ns which takes on input argument, the name of a Tcl simulation scripting file. Users are feeding the name of a Tcl simulation script (which sets up a simulation) as an input argument of an NS2 executable command ns. In most cases, a simulation trace file is created, and is used to plot graph and/or to create animation. NS2 consists of two key languages: C++ and Object-oriented Tool Command Language (OTcl). While the C++ defines the internal mechanism (i.e., a backend) of the simulation objects, the OTcl sets up simulation by assembling and configuring the objects as well as scheduling discrete events (i.e., a frontend).

The C++ and the OTcl are linked together using TclCL. Mapped to a C++ object, variables in the OTcl domains are sometimes referred to as handles. Conceptually, a handle (e.g., n as a Node handle) is just a string (e.g.,_o10) in the OTcl domain, and does not contain any functionality. Instead, the functionality (e.g., receiving a packet) is defined in the mapped C++ object (e.g., of class Connector). In the OTcl domain, a handle acts as a frontend which interacts with users and other OTcl objects. It may defines its own procedures and variables to facilitate the interaction. Note that the member procedures and variables in the OTcl domain are called instance procedures (instprocs) and instance variables (instvars), respectively.

NS2 provides a large number of built-in C++ objects. It is advisable to use these C++ objects to set up a simulation using a Tcl simulation script. However, advance users may find these objects insufficient.

They need to develop their own C++ objects, and use a OTcl configuration interface to put together these objects. After simulation, NS2 outputs either text-based or animation – based simulation results. To interpret these results graphically and interactively, tools such as NAM (Network AniMator) and XGraph are used. To analyze a particular behaviour of the network, users can extract a relevant subset of text-based data and transform it to a more conceivable presentation.

CONCEPT OVERVIEW:

NS uses two languages because simulator has two different kinds of things it needs to do. On one hand, detailed simulations of protocols requires a systems programming language which can efficiently manipulate bytes, packet headers, and implement algorithms that run over large data sets. For these tasks run-time speed is important and turn-around time (run simulation, find bug, fix bug, recompile, re-run) is less important. On the other hand, a large part of network research involves slightly varying parameters or configurations, or quickly exploring a number of scenarios.

In these cases, iteration time (change the model and re-run) is more important. Since configuration runs once (at the beginning of the simulation), run-time of this part of the task is less important. ns meets both of these needs with two languages, C++ and OTcl.

Tcl scripting

Tcl is a general purpose scripting language. [Interpreter].

- Tcl runs on most of the platforms such as Unix, Windows, and Mac.
- The strength of Tcl is its simplicity.
- It is not necessary to declare a data type for variable prior to the usage.

Basics of TCL

Syntax: command arg1 arg2 arg3

Hello World!

puts stdout{Hello, World!} Hello, World!

Variables Command Substitution

set a 5 set len [string length foobar]

set b $a set len [expr [string length foobar] + 9]

Wired TCL Script Components

Create the event scheduler.

Open new files & turn on the tracing.

Create the nodes.

Setup the links.

Configure the traffic type (e.g., TCP, UDP, etc).

Set the time of traffic generation (e.g., CBR, FTP).

Terminate the simulation.

NS Simulator Preliminaries

1. Initialization and termination aspects of the ns simulator.
2. Definition of network nodes, links, queues and topology.
3. Definition of agents and of applications.
4. The nam visualization tool.
5. Tracing and random variables.
6. Initialization and Termination of TCL Script in NS – 2.

An ns simulation starts with the command

set ns [new Simulator]

Which is thus the first line in the tcl script. This line declares a new variable as using the set command , you can call this variable as you wish, In general people declares it as ns because it is an instance of the Simulator class, so an object the code[new Simulator] is indeed the installation of the class Simulator using the reserved word new.

In order to have output files with data on the simulation (trace files) or files used for visualization (nam files), we need to create the files using – open command:

#Open the Trace file

set tracefile1 [open out.tr w]

$ns trace – all $tracefile1

#Open the NAM trace file

set namfile [open out.nam w]

$ns namtrace – all $namfile

The above creates a dta trace file called out.tr and a nam visualization trace file called out.nam. Within the tcl script, these files are not called explicitly by their names, but instead by pointers that are declared above and called – tracefile1 and – namfile respectively. Remark that they begins with a # symbol.

The second line open the file – out.tr to be used for writing, declared with the letter – w. The third line uses a simulator method called trace – all that have as parameter the name of the file where the traces will go.

Define a "finish" procedure

```
Proc finish { } {
global ns tracefile1 namfile
$ns flush-trace
Close $tracefile1
Close $namfile
Exec nam out.nam &
Exit 0
}
```

Definition of a network of links and nodes

The way to define a node is

set n0 [$ns node]

Once we define several nodes, we can define the links that connect them. An example of a definition of a link is:

$ns duplex-link $n0 $n2 10Mb 10ms DropTail

Which means that $n0 and $n2 are connected using a bi-directional link that has 10ms of propagation delay and a capacity of 10Mb per sec for each direction.

To define a directional link instead of a bi-directional one, we should replace – duplex-link by – simplex-link.

In ns, an output queue of a node is implemented as a part of each link whose input is that node. We should also define the buffer capacity of the queue related to each link. An example would be:

#set Queue Size of link (n0-n2) to 20

$ns queue-limit $n0 $n2 20

FTP over TCP

TCP is a dynamic reliable congestion control protocol. It uses Acknowledgements created by the destination to know whether packets are well received.

There are number variants of the TCP protocol, such as Tahoe, Reno, NewReno, Vegas. The type of agent appears in the first line:

set tcp [new Agent/TCP]

The command **$ns attach-agent $n0 $tcp** defines the source node of the tcp connection.

The command **set sink [new Agent /TCPSink]** Defines the behavior of the destination node of TCP and assigns to it a pointer called sink.

#Setup a UDP connection

set udp [new Agent/UDP]

$ns attach – agent $n1 $udp

set null [new Agent/Null]

$ns attach – agent $n5 $null

$ns connect $udp $null

$udp set fid_2

#setup a CBR over UDP connection

The below shows the definition of a CBR application using a UDP agent .The command **$ns attach – agent $n4 $sink** defines the destination node. The command $**ns connect $tcp $sink** finally makes the TCP connection between the source and destination nodes.

set cbr [new Application/Traffic/CBR]

$cbr attach – agent $udp

$cbr set packetsize_100

$cbr set rate_ 0.01Mb

$cbr set random_ false

TCP has many parameters with initial fixed defaults values that can be changed if mentioned explicitly. For example, the default TCP packet size has a size of 1000 bytes. This can be changed to another value, say 552bytes, using the command $tcp set packetSize 552.

When we have several flows, we may wish to distinguish them so that we can identify them with different colors in the visualization part. This is done by the command $tcp set fid_ 1 that assigns to the TCP connection a flow identification of – 1.We shall later give the flow identification of - 2‖ to the UDP connection.

Wired TCL Script Components

Create the event scheduler.

Open new files & turn on the tracing.

Create the nodes.

Setup the links.

Configure the traffic type (e.g., TCP, UDP, etc).

Set the time of traffic generation (e.g., CBR, FTP).

Terminate the simulation.

NS Simulator Preliminaries.

Initialization and termination aspects of the ns simulator.

Definition of network nodes, links, queues and topology.

Definition of agents and of applications.

The nam visualization tool.

Tracing and random variables.

Features of NS2

NS2 can be employed in most unix systems and windows. Most of the NS2 code is in C++. It uses TCL as its scripting language, Otcl adds object orientation to TCL. NS (version 2) is an object oriented, discrete event driven network simulator that is freely distributed and open source.

- Traffic Models: CBR, VBR, Web etc.
- Protocols: TCP, UDP, HTTP, Routing algorithms,MAC etc.
- Error Models: Uniform, bursty etc.
- Misc: Radio propagation, Mobility models , Energy Models.
- Topology Generation tools.
- Visualization tools (NAM), Tracing.

Structure of NS

- NS is an object oriented discrete event simulator.
- Simulator maintains list of events and executes one event after another.
- Single thread of control: no locking or race conditions.
- Back end is C++ event scheduler.
- Protocols mostly.
- Fast to run, more control.
- Front end is OTCL.
- Creating scenarios, extensions to C++ protocols.
- fast to write and change.

Platforms

- It can be employed in most unix systems (FreeBSD, Linux, Solaris) and Windows.

Source code

- Most of NS2 code is in C++

Scripting language

It uses TCL as its scripting language OTcl adds object orientation to TCL.

Protocols implemented in NS2.

Transport layer (Traffic Agent) – TCP, UDP.

Network layer (Routing agent).

Interface queue – FIFO queue, Drop Tail queue, Priority queue.

Logic link contol layer – IEEE 802.2, AR.

How to use NS2

Design Simulation – Determine simulation scenario.

Build ns-2 script using tcl.

Run simulation.

Simulation with NS2

Define objects of simulation.

Connect the objects to each other.

Start the source applications. Packets are then created and are transmitted through network.

Exit the simulator after a certain fixed time.

NS programming Structure

- Create the event scheduler
- Turn on tracing
- Create network topology
- Create transport connections
- Generate traffic
- Insert errors

Sample Wired Simulation using NS – 2

Creating Event Scheduler

- Create event scheduler: set ns [new simulator]
- Schedule an event:

  $ns at <time><event>

  – event is any legitimate ns/tcl function

  ```
  $ns at 5.0 “finish”
  proc finish {} {
  global ns nf
  close $nf
  exec nam out.nam &
  exit 0
  }
  ```
- Start Scheduler

  $ns run

Tracing

- All packet trace

  $ns traceall[open out.tr w]

  <event><time><from><to><pkt><size>

  ...

```
<flowid><src><dst><seqno><aseqno>
+0.51    0   1   cbr  500—— 0   0.0  1.0  0   2
_ 0.51    0   1   cbr   500—— 0  0.0  1.0  0  2
R 0.514  0  1  cbr    500 ——  0  0.0  1.0  0  0
```

- Variable trace

```
set par [open output/param.tr w]
$tcp attach $par
$tcp trace cwnd_
$tcp trace maxseq_
$tcp trace rtt_
```

Tracing and Animation

- Network Animator

```
set nf [open out.nam w]
$ns namtraceall
$nf
proc finish {} {
global ns nf
close $nf
exec nam out.nam &
exit 0
}
```

Creating topology

- Two nodes connected by a link
- Creating nodes

```
set n0 [$ns node]
set n1 [$ns node]
```

- Creating link between nodes

```
$ns <link_type> $n0 $n1 <bandwidth><delay><queue-type>
$ns duplex-link$n0 $n1 1Mb 10ms DropTail
```

Data Sending

- Create UDP agentset udp0 [new Agent/UDP]

```
$ns attach-agent $n0 $udp0
```

- Create CBR traffic source for feeding into UDP agent

```
set cbr0 [new Application/Traffic/CBR]
```

```
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent$udp0
```

- Create traffic sink

```
set null0 [new Agent/Null]
$ns attach-agent$n1 $null0
```

- Connect two agents

```
$ns connect $udp0 $null0
```

- Start and stop of data

```
$ns at 0.5 “$cbr0 start”
$ns at 4.5 “$cbr0 stop”
```

Traffic on top of TCP

- **FTP**

```
set ftp [new Application/FTP]
$ftp attach -agent$tcp0
```

- **Telnet**

```
set telnet [new Application/Telnet]
$telnet attach-agent$tcp0
```

PROCEDURE:

STEP 1: Start.

STEP 2: Create the simulator object ns for designing the given simulation.

STEP 3: Open the trace file and nam file in the write mode.

STEP 4: Create the nodes of the simulation using the ‘set’ command.

STEP 5: Create links to the appropriate nodes using $ns duplex link command.

STEP 6: Set the orientation for the nodes in the simulation using ‘orient’ command.

STEP 7: Create TCP agent for the nodes and attach these agents to the nodes.

STEP 8: The traffic generator used is FTP for both node0 and node1.

STEP 9: Configure node1 as the sink and attach it.

STEP10: Connect node0 and node1 using ‘connect’ command.

STEP 11: Setting color for the nodes.

STEP 12: Schedule the events for FTP agent 10 sec.

STEP 13: Schedule the simulation for 5 minutes.

Structure of Trace Files

When tracing into an output ASCII file, the trace is organized in 12 fields as follows in fig shown below,

The meaning of the fields are:

Event	Time	From Node	To Node	PKT Type	PKT Size	Flags	Fid	Src Addr	Dest Addr	Seq Num	Pkt ID

1. The first field is the event type. It is given by one of four possible symbols r, +, -, d which correspond respectively to receive (at the output of the link), enqueued, dequeued and dropped.
2. The second field givesthe time at which the event occurs.
3. Gives the input node of the link at which the event occurs.
4. Gives the output node of the link at which the event occurs.
5. Gives the packet type (eg CBR or TCP).
6. Gives the packet size.
7. Some flags.
8. This is the flow id (fid) of IPv6 that a user can set for each flow at the input OTcl script one can further use this field for analysis purposes; it is also used when specifying stream color for the NAM display.
9. This is the source address given in the form of — node.port.
10. This is the destination address, given in the same form.
11. This is the network layer protocol’s packet sequence number. Even though UDP implementations in a real network do not use sequence number, ns keeps track of UDP packet sequence number for analysis purposes.
12. The last field shows the Unique id of the packet

XGRAPH

The xgraph program draws a graph on an x -display given data read from either data file or from standard input if no files are specified. It can display upto 64 independent data sets using different colors and line styles for each set. It annotates the graph with a title, axis labels, grid lines or tick marks, grid labels and a legend.

Syntax:

Xgraph[options]file – name

Options are listed here

/-bd <color> (Border)

This specifies the border color of the xgraph window.

/-bg <color> (Background)

This specifies the background color of the xgraph window.

/-fg<color> (Foreground)

This specifies the foreground color of the xgraph window.

/-lf <fontname> (LabelFont)

All axis labels and grid labels are drawn using this font.

/-t<string> (Title Text)

This string is centered at the top of the graph.

/-x <unit name> (XunitText)

This is the unit name for the x-axis. Its default is "X".

/-y <unit name> (YunitText)

This is the unit name for the y-axis. Its default is "Y".

Awk – An Advanced

awk is a programmable, pattern-matching, and processing tool available in UNIX. It works equally well with text and numbers.

awk is not just a command, but a programming language too. In other words, awk utility is a pattern scanning and processing language. It searches one or more files to see if they contain lines that match specified patterns and then perform associated actions, such as writing the line to the standard output or incrementing a counter each time it finds a match.

Syntax:awk option 'selection_criteria {action}' file(s)

Here, selection_criteria filters input and select lines for the action component to act upon. The selection_criteria is enclosed within single quotes and the action within the curly braces. Both the selection_criteria and action forms an awk program.

Example: $ awk „/manager/ {print}‟ emp.lst

Variables

Awk allows the user to use variables of there choice. You can now print a serial number, using the variable kount, and apply it those directors drawing a salary exceeding 6700:

```
$ awk –F"|" „$3 == "director" && $6 > 6700 {
kount =kount+1
printf " %3f %20s % - 12s %d\ n", kount,$2,$3,$6 }‟ empn.lst
```

THE -f OPTION: STORING awk PROGRAMS IN A FILE

You should holds large awk programs in separate file and provide them with the awk extension for easier identification.

Let‟s first store the previous program in the file empawk.awk:

```
$ cat empawk.awk
```

Observe that this time we haven‟t used quotes to enclose the awk program. You can now use awk with the –f filename option to obtain the same output:

```
Awk –F"|"–f empawk.awk empn.lst
```

The BEGIN and END Sections

Awk statements are usually applied to all lines selected by the address, and if there are no addresses, then they are applied to every line of input. But, if you have to print something before processing the first line, for example, a heading, then the BEGIN section can be used gainfully. Similarly, the end section useful in printing some totals after processing is over.

The BEGIN and END sections are optional and take the form

```
BEGIN {action}
END {action}
```

These two sections, when present, are delimited by the body of the awk program. You can use them to print a suitable heading at the beginning and the average salary at the end.

Built – In Variables

Awk has several built –in variables. They are all assigned automatically, though it is also possible for a user to reassign some of them. You have already used NR, which signifies the record number of the current line. We‟ll now have a brief look at some of the other variable.

The FS Variable: as stated elsewhere, awk uses acontiguous string of spaces as the default field delimiter. FS redefines this field separator, which in the sample database happens to be the |. When used at all, it must occur in the BEGIN section so that the body of the program knows its value before it starts processing:

```
BEGIN {FS="|"}
```

This is an alternative to the -F option which does the same thing.

The OFS Variable: when you used the print statement with comma-separated arguments, each argument was separated from the other by a space. This is awk"s default output field separator, and can reassigned using the variable OFS in the BEGIN section:

```
BEGIN { OFS=”~” }
```

When you reassign this variable with a ~ (tilde), awk will use this character for delimiting the print arguments. This is a useful variable for creating lines with delimited fields.

The NF variable: NF comes in quite handy for cleaning up a database of lines that don"t contain the right number of fields. By using it on a file, say emp.lst, you can locate those lines not having 6 fields, and which have crept in due to faulty data entry:

```
$awk „BEGIN {FS = “|”}
NF! =6 {
Print “Record No “, NR, “has”, “fields”}‟ empx.lst
```

PROGRAM NO: 7.

Implement a point to point network with four nodes and duplex links between them. Analyse the network performance by setting the queue size and varying bandwidth

AIM:Implement a point to point network with four nodes and duplex links between them. Analyse the network performance by setting the queue size and varying bandwidth

THEORY:

POINT TO POINT

Point to Point networks contains exactly two hosts such as computer, switches, routers, or servers connected back to back using a single piece of cable. Often, the receiving end of one host is connected to sending end of the other and vice versa. If the hosts are connected point-to-point logically, then may have multiple intermediate devices. But the end hosts are unaware of underlying network and see each other as if they are connected directly.

DUPLEX LINK

A duplex communication system is a point to point system composed of two connected parties or devices that can communicate with one another in both directions. "Duplex" comes from "duo" that means "double", and "plex" that means "structure" or "parts of"; thus, a duplex system has two clearly defined data transmissions, with each path carrying information in only one direction: A to B over one path, and B to A over the other. There are two types of duplex communication systems: full-Duplex and half – Duplex.

In a full duplex system, both parties can communicate with each other simultaneously. An example of a full – duplex device is a telephone the parties at both ends of a call can speak and be heard by the other party simultaneously. The earphone reproduces the speech of the remote party as the microphone transmits the speech of the local party, because there is a two-way communication channel between them, or more strictly speaking, because there are two communication paths/channels between them.

In a half – duplex system, there are still two clearly defined paths / channels, and each party can communicate with the other but not simultaneously; the communication is one direction at a time. An example of a half – duplex device is a walkie-talkie two – way radio that has a “push to talk” button; when the local user wants to speak to the remote person they push this button, which turns on the transmitter but turns off the receiver, so they cannot hear the remote person. To listen to the other person they release the button, which turns on the receiver but turns off the transmitter.

Duplex systems are employed in many communications networks, either to allow for a communication "two-way street" between two connected parties or to provide a "reverse path" for the monitoring and remote adjustment of equipment in the field.

PROGRAM:

FIRST1.TCL

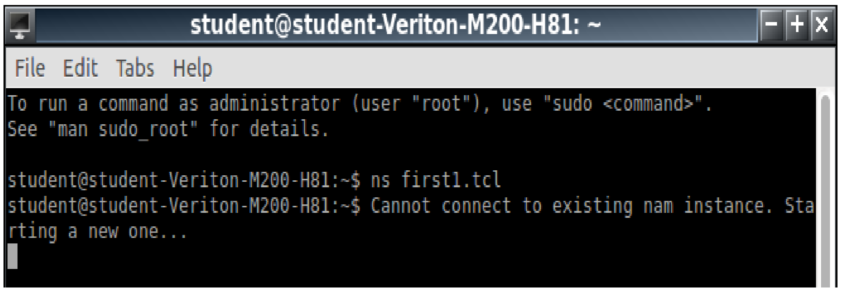
```
#Create a simulator object
        set ns [ new Simulator ]
#Open the nam trace file
        set tf [ open first1.tr w ]
        $ns trace-all $tf
#Open the nam trace file
        set nf [ open first1.nam w ]
        $ns namtrace-all $nf
#Define a 'finish' procedure
        proc finish { } {
        global ns nf tf
        $ns flush-trace
        exec nam first1.nam &
        close $tf
        close $nf
        exit 0
        }
#Creating nodes
        set n0 [$ns node]
        set n1 [$ns node]
        set n2 [$ns node]
        set n3 [$ns node]
#Define different colors and labels for data flows
        $ns color 1 "red"
        $ns color 2 "blue"
        $n0 label "Source/udp0"
        $n1 label "Source/udp1"
        $n2 label "Router"
        $n3 label "Destination/Null"
#Create link between nodes
        $ns duplex-link $n0 $n2 100Mb 300ms DropTail
        $ns duplex-link $n1 $n2 100Mb 300ms DropTail
        $ns duplex-link $n2 $n3 1Mb 300ms DropTail
#Set queue size of links
        $ns set queue-limit $n0 $n2 50
        $ns set queue-limit $n1 $n2 50
        $ns set queue-limit $n2 $n3 5
#Setup a UDP connection
        set udp0 [new Agent/UDP]
        $ns attach-agent $n0 $udp0
# Create a CBR traffic source and attach it to udp0
        set cbr0 [new Application/Traffic/CBR]
        $cbr0 set packetSize_ 500
        $cbr0 set interval_ 0.005
        $cbr0 attach-agent $udp0
#Create a UDP agent and attach it to node n1
        set udp1 [new Agent/UDP]
        $udp1 set class_ 2
        $ns attach-agent $n1 $udp1
# Create a CBR traffic source and attach it to udp1
        set cbr1 [new Application/Traffic/CBR]
        $cbr1 set packetSize_ 500
        $cbr1 set interval_ 0.005
        $cbr1 attach-agent $udp1
#Create a Null agent (a traffic sink) and attach it to node n3
        set null0 [new Agent/Null]
        $ns attach-agent $n3 $null0
#Connect the traffic sources with the traffic sink
        $ns connect $udp0 $null0
        $ns connect $udp1 $null0
#Schedule events for the CBR agents
        $ns at 0.5 "$cbr0 start"
        $ns at 1.0 "$cbr1 start"
        $ns at 4.0 "$cbr1 stop"
        $ns at 4.5 "$cbr0 stop"
#Call the finish procedure after 5 seconds of simulation time
```

```
        $ns at 5.0 "finish"
#Run the simulation
        $ns run
```

FIRST1.AWK

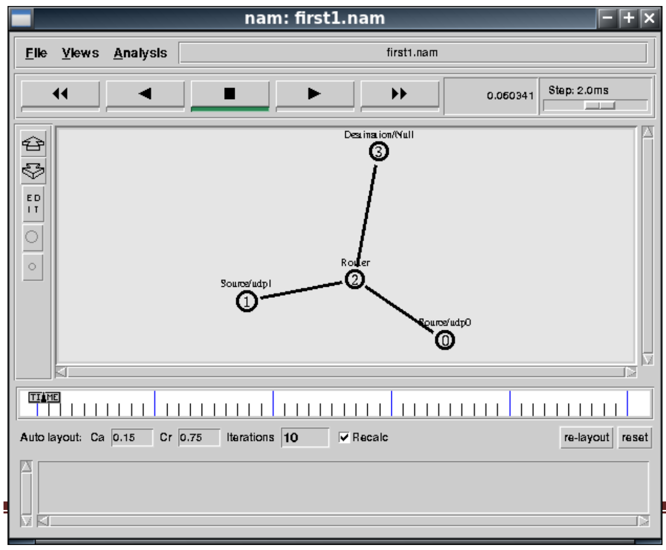
```
BEGIN
{
        count=0;
}
{
        if($1=="d")
                count++
}
END
{
        printf("The Total no of Packets Drop is :%d \n\n", count)
}
```

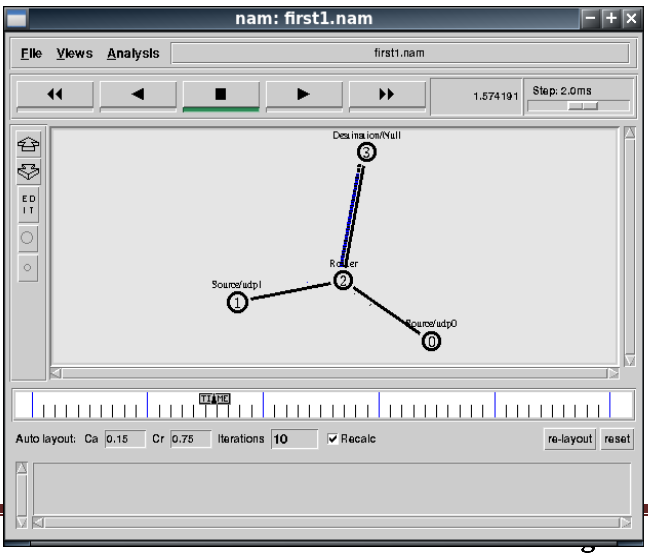
STEPS OF EXECUTION

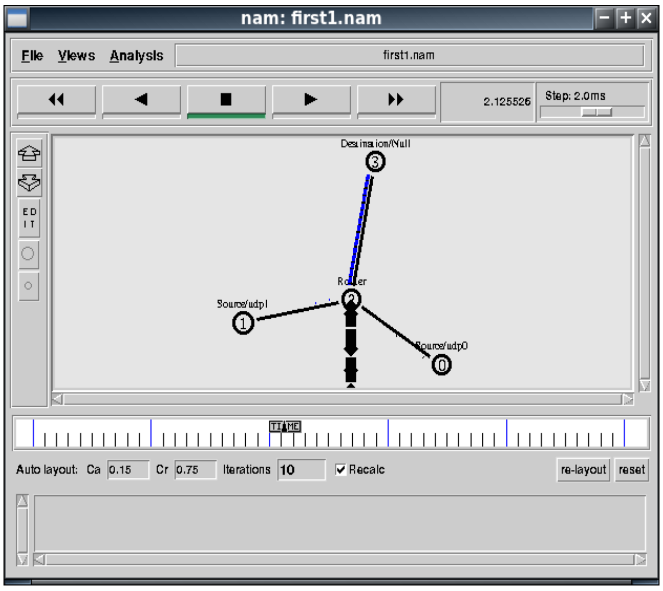


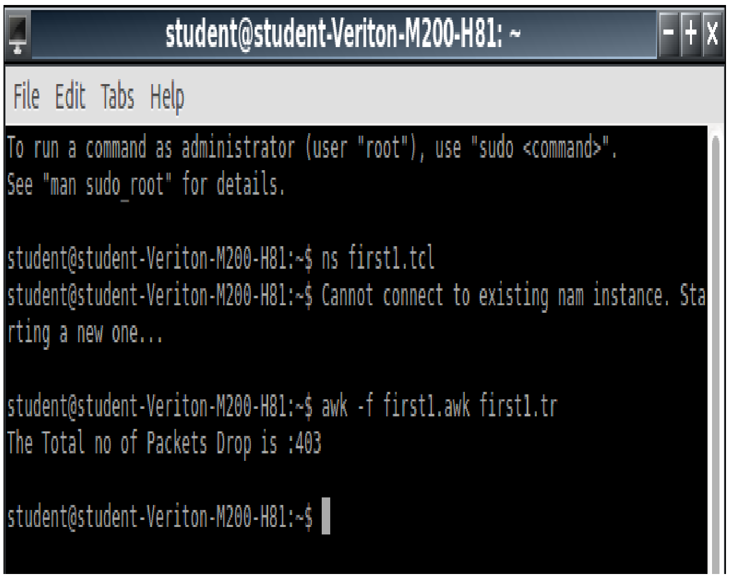
```
student@student-Veriton-M200-H81: ~
File  Edit  Tabs  Help
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

student@student-Veriton-M200-H81:~$ ns first1.tcl
student@student-Veriton-M200-H81:~$ Cannot connect to existing nam instance. Sta
rting a new one...
```

OUTPUT:









```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

student@student-Veriton-M200-H81:~$ ns first1.tcl
student@student-Veriton-M200-H81:~$ Cannot connect to existing nam instance. Sta
rting a new one...

student@student-Veriton-M200-H81:~$ awk -f first1.awk first1.tr
The Total no of Packets Drop is :403

student@student-Veriton-M200-H81:~$ 
```

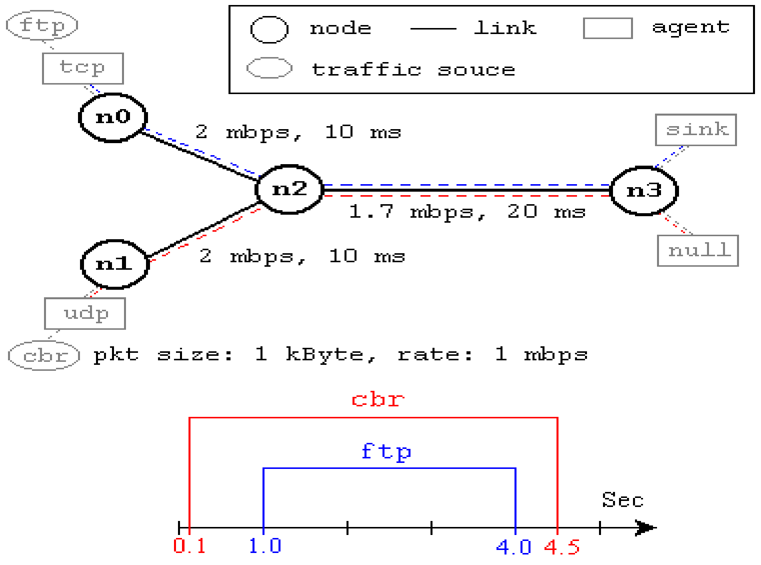
RESULT: Implementation of a point to point network with four nodes and duplex links between them is analysed by setting the queue size and varying bandwidth.

PROGRAM NO: 8

Implement a four node point to point network with links n0 – n2, n1-n2 and n2-n3. Apply TCP agent between n0 – n3 and UDP between n1 – n3. Apply relevant applications over TCP and UDP agents changing the parameter and determine the number of packets sent by TCP / UDP.

AIM:Implement a four node point to point network with links n0 – n2, n1-n2 and n2-n3. Apply TCP agent between n0 – n3 and UDP between n1 – n3. Apply relevant applications over TCP and UDP agents changing the parameter and determine the number of packets sent by TCP / UDP.

THEORY:



This network consists of 4 nodes (n0, n1, n2, n3) as shown in above figure. The duplex links between n0 and n2, and n1 and n2 have 2 Mbps of bandwidth and 10 ms of delay. The duplex link between n2 and n3 has 1.7 Mbps of bandwidth and 20 ms of delay. Each node uses a DropTail queue, of which the maximum size is 10. A "tcp" agent is attached to n0, and a connection is established to a tcp "sink" agent attached to n3. As default, the maximum size of a packet that a "tcp" agent can generate is 1KByte. A tcp "sink" agent generates and sends ACK packets to the sender (tcp agent) and frees the received packets. A "udp" agent that is attached to n1 is connected to a "null" agent attached to n3. A "null" agent just frees the packets received. A "ftp" and a "cbr" traffic generator are attached to "tcp" and "udp" agents respectively, and the "cbr" is configured to generate 1 KByte packets at the rate of 1 Mbps. The "cbr" is set to start at 0.1 sec and stop at 4.5 sec, and "ftp" is set to start at 1.0 sec and stop at 4.0 sec.

EXPLANATION OF PROGRAM

- **set ns [new Simulator]:** generates an NS simulator object instance, and assigns it to variable ns (italics is used for variables and values in this section). What this line does is the following:

Initialize the packet format (ignore this for now).

Create a scheduler (default is calendar scheduler).

Select the default address format (ignore this for now).

The "Simulator" object has member functions that do the following:

- Create compound objects such as nodes and links (described later)
- Connect network component objects created (ex. attach-agent)
- Set network component parameters (mostly for compound objects)
- Create connections between agents (ex. make connection between a "tcp" and "sink")
- Specify NAM display options
- Etc.

Most of member functions are for simulation setup (referred to as plumbing functions in the Overview section) and scheduling, however some of them are for the NAM display. The "Simulator" object member function implementations are located in the **"ns–2 /tcl/lib/ns–lib.tcl"** file.

- **$ns color fid color:** is to set color of the packets for a flow specified by the flow id (fid). This member function of "Simulator" object is for the NAM display, and has no effect on the actual simulation.
- **$ns namtrace–all file–descriptor:** This member function tells the simulator to record simulation traces in NAM input format. It also gives the file name that the trace will be written to later by the command **$ns flush–trace.** Similarly, the member function **trace–all** is for recording the simulation trace in a general format.
- **proc finish {}:** is called after this simulation is over by the command **$ns at 5.0 "finish".** In this function, post-simulation processes are specified.
- **set n0 [$ns node]:** The member function **node** creates a node. A node in NS is compound object made of address and port classifiers (described in a later section). Users can create a node by separately creating an address and a port classifier objects and connecting them together. However, this member function of Simulator object makes the job easier. To see how a node is created, look at the files: **"ns–2/tcl/libs/ns–lib.tcl"** and **"ns–2/tcl/libs/ns–node.tcl"**.
- **$ns duplex–link node1 node2 bandwidth delay queue–type:** creates two simplex links of specified bandwidth and delay, and connects the two specified nodes. In NS, the output queue of a node is implemented as a part of a link, therefore users should specify the queue-type when creating links. In the above simulation script, DropTail queue is used. If the reader wants to use a RED queue, simply replace the word DropTail with RED. The NS implementation of a link is shown in a later section. Like a node, a link is a compound object, and users can create its sub-objects and connect them and the nodes. Link source codes can be found in **"ns–2/tcl/libs/ns-lib.tcl"** and **"ns–2/tcl/libs/ns–**

link.tcl" files. One thing to note is that you can insert error modules in a link component to simulate a lossy link (actually users can make and insert any network objects). Refer to the NS documentation to find out how to do this.

- **$ns queue–limit node1 node2 number:** This line sets the queue limit of the two simplex links that connect node1 and node2 to the number specified. At this point, the authors do not know how many of these kinds of member functions of Simulator objects are available and what they are. Please take a look at **"ns–2/tcl/libs/ns–lib.tcl"** and **"ns–2/tcl/libs/ns–link.tcl"**, or NS documentation for more information.
- **$ns duplex–link-op node1 node2 ...:** The next couple of lines are used for the NAM display. To see the effects of these lines, users can comment these lines out and try the simulation.

Now that the basic network setup is done, the next thing to do is to setup traffic agents such as TCP and UDP, traffic sources such as FTP and CBR, and attach them to nodes and agents respectively.

- **set tcp [new Agent/TCP]:** This line shows how to create a TCP agent. But in general, users can create any agent or traffic sources in this way. Agents and traffic sources are in fact basic objects (not compound objects), mostly implemented in C++ and linked to OTcl. Therefore, there are no specific Simulator object member functions that create these object instances. To create agents or traffic sources, a user should know the class names these objects (Agent/TCP, Agnet/TCPSink, Application/FTP and so on). This information can be found in the NS documentation or partly in this documentation. But one shortcut is to look at the **"ns–2/tcl/libs/ns–default.tcl"** file. This file contains the default configurable parameter value settings for available network objects. Therefore, it works as a good indicator of what kind of network objects are available in NS and what are the configurable parameters.
- **$ns attach – agent node agent:** The **attach – agent** member function attaches an agent object created to a node object. Actually, what this function does is call the **attach** member function of specified node, which attaches the given agent to itself. Therefore, a user can do the same thing by, for example, **$n0 attach $tcp**. Similarly, each agent object has a member function **attach – agent** that attaches a traffic source object to itself.
- **$ns connect agent1 agent2:** After two agents that will communicate with each other are created, the next thing is to establish a logical network connection between them. This line establishes a network connection by setting the destination address to each others' network and port address pair.

Assuming that all the network configuration is done, the next thing to do is write a simulation scenario (i.e. simulation scheduling). The Simulator object has many scheduling member functions. However, the one that is mostly used is the following:

- **$ns at time "string":** This member function of a Simulator object makes the scheduler (scheduler_ is the variable that points the scheduler object created by [new Scheduler] command at the beginning of the script) to schedule the execution of the specified string at given simulation time. For example, **$ns at 0.1 "$cbr start"** will make the scheduler call a **start** member function of the CBR traffic source object, which starts the CBR to transmit data. In NS, usually a traffic source does not transmit actual data, but it notifies the underlying agent that it has some amount of data to transmit, and the agent, just knowing how much of the data to transfer, creates packets and sends them.

After all network configuration, scheduling and post-simulation procedure specifications are done, the only thing left is to run the simulation. This is done by **$ns run**.

PROGRAM:

```
#Create a simulator object
        set ns [new Simulator]
#Define different colors for data flows (for NAM)
        $ns color 1 Blue
        $ns color 2 Red
#Open the NAM trace file
        set nf [open out.nam w]
        $ns namtrace-all $nf
#Define a 'finish' procedure
        proc finish {} {
        global ns nf
        $ns flush-trace
        #Close the NAM trace file
        close $nf
        #Execute NAM on the trace file
        exec nam out.nam &
        exit 0
        }
#Create four nodes
        set n0 [$ns node]
        set n1 [$ns node]
        set n2 [$ns node]
        set n3 [$ns node]
```

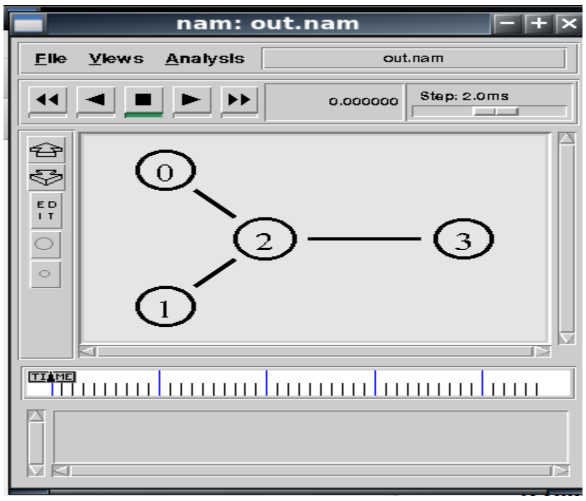
```
#Create links between the nodes
        $ns duplex-link $n0 $n2 2Mb 10ms DropTail
        $ns duplex-link $n1 $n2 2Mb 10ms DropTail
        $ns duplex-link $n2 $n3 1.7Mb 20ms DropTail
#Set Queue Size of link (n2-n3) to 10
        $ns queue-limit $n2 $n3 10
#Give node position (for NAM)
        $ns duplex-link-op $n0 $n2 orient right-down
        $ns duplex-link-op $n1 $n2 orient right-up
        $ns duplex-link-op $n2 $n3 orient right
#Monitor the queue for link (n2-n3). (for NAM)
        $ns duplex-link-op $n2 $n3 queuePos 0.5
#Setup a TCP connection
        set tcp [new Agent/TCP]
        $tcp set class_ 2
        $ns attach-agent $n0 $tcp
        set sink [new Agent/TCPSink]
        $ns attach-agent $n3 $sink
        $ns connect $tcp $sink
        $tcp set fid_ 1
#Setup a FTP over TCP connection
        set ftp [new Application/FTP]
        $ftp attach-agent $tcp
        $ftp set type_ FTP
#Setup a UDP connection
        set udp [new Agent/UDP]
        $ns attach-agent $n1 $udp
        set null [new Agent/Null]
        $ns attach-agent $n3 $null
        $ns connect $udp $null
        $udp set fid_ 2
#Setup a CBR over UDP connection
        set cbr [new Application/Traffic/CBR]
        $cbr attach-agent $udp
        $cbr set type_ CBR
        $cbr set packet_size_ 1000
        $cbr set rate_ 1mb
        $cbr set random_ false
#Schedule events for the CBR and FTP agents
        $ns at 0.1 "$cbr start"
        $ns at 1.0 "$ftp start"
        $ns at 4.0 "$ftp stop"
        $ns at 4.5 "$cbr stop"
#Detach tcp and sink agents (not really necessary)
        $ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink"
#Call the finish procedure after 5 seconds of simulation time
        $ns at 5.0 "finish"
#Print CBR packet size and interval
        puts "CBR packet size = [$cbr set packet_size_]"
        puts "CBR interval = [$cbr set interval_]"
#Run the simulation
        $ns run
```

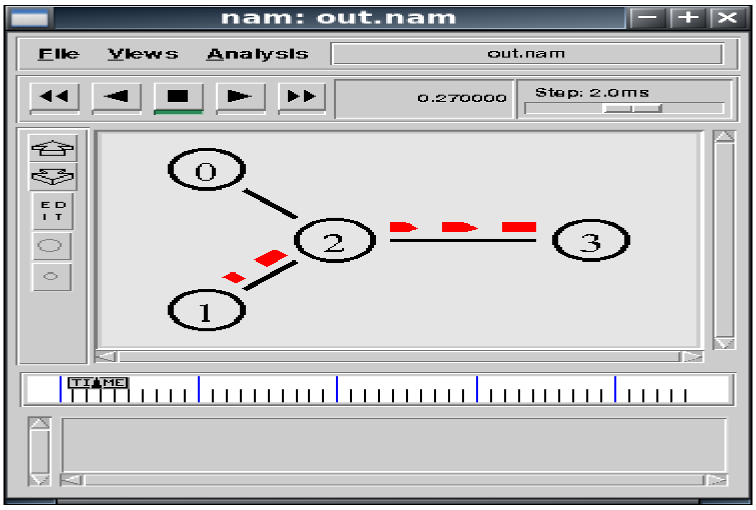
STEPS OF EXECUTION:

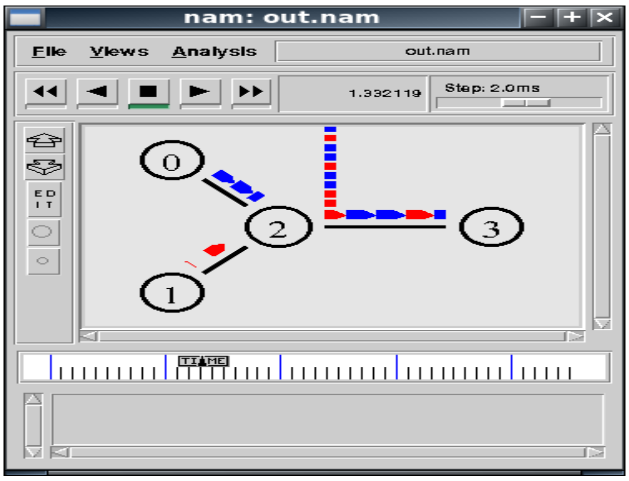
1. Open vi editor and type the program. Save the files with <filename>.tcl and <filename>.awk extensions.
2. Open the command prompt and type ns <filename>.tcl.

OUTPUT:

```
student@student-Veriton-M200-H81:~$ ns ns-simple.tcl
CBR packet size = 1000
CBR interval = 0.0080000000000000002
student@student-Veriton-M200-H81:~$
```







RESULT: Thus implementation of a four node point to point network with links n0 – n2, n1 – n2 and n2 – n3 and determining the number of packets sent by TCP/UDP is done successfully.

PROGRAM NO: 9

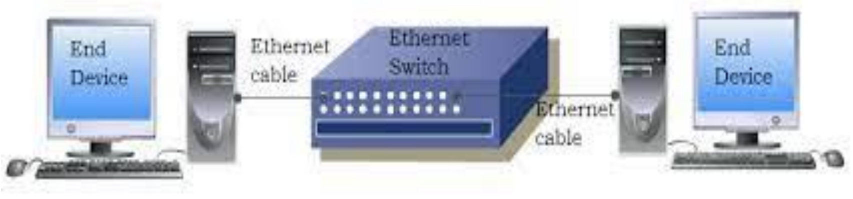
Implement Ethernet LAN using n(6 – 10) nodes. Compare the throughput by changing the error rate and data rate.

AIM: Implement Ethernet LAN using n(6 – 10) nodes. Comapre the throughput by changing the error rate and data rate.

THEORY:

Ethernet is most widely used LAN Technology, which is defined under IEEE standards 802.3. The reason behind its wide usability is Ethernet is easy to understand, implement, maintain and allows low – cost network implementation. Also, Ethernet offers flexibility in terms of topologies which are allowed. Ethernet generally uses Bus Topology. Ethernet operates in two layers of the OSI model, Physical Layer, and Data Link Layer. For Ethernet, the protocol data unit is Frame since we mainly deal with DLL. In order to handle collision, the Access control mechanism used in Ethernet is CSMA/CD.

Manchester Encoding Technique is used in Ethernet.



How Ethernet Works

When a machine on the network wants to send data to another, it senses the carrier, which is the main wire connecting the devices. If it is free, meaning no one is sending anything, it sends the data packet on the network, and the other devices check the packet to see whether they are the recipient. The recipient consumes the packet. If there is a packet on the highway, the device that wants to send holds back for some thousandths of a second to try again until it can send.

PROGRAM:

Ether.tcl

```
#ETHERNET LAN
        set ns [new Simulator]
#Open a new file for NAMTRACE
        set nf [open out.nam w]
        $ns namtrace-all $nf
#Open a new file to log TRACE
```

```
        set tf [open out.tr w]
        $ns trace-all $tf
#Body of the finish procedure
        proc finish {} {
        global ns nf tf
        $ns flush-trace
        close $nf
        close $tf
        exec nam out.nam &
        exec awk -f exp5.awk out.tr &
        exit 0
        }
#Create Nodes
        set n0 [$ns node]
        set n1 [$ns node]
        set n2 [$ns node]
        set n3 [$ns node]
        set n4 [$ns node]
        set n5 [$ns node]
        set n6 [$ns node]
        set n7 [$ns node]
        set n8 [$ns node]
        set n9 [$ns node]
        set n10 [$ns node]
#Create a Local Area Network (LAN) of 10 Nodes
        $ns make-lan "$n0 $n1 $n2 $n3 $n4 $n5 $n6 $n7 $n8 $n9 $n10" 100Mb 20ms LL Queue/DropTail
        Mac/802_3
#Create TCP Agent between node 0 and node 3
        set tcp0 [new Agent/TCP]
        $ns attach-agent $n0 $tcp0
        set sink0 [new Agent/TCPSink]
        $ns attach-agent $n3 $sink0
        $ns connect $tcp0 $sink0
#Create FTP Application for TCP Agent
```

```
        set ftp0 [new Application/FTP]
        $ftp0 attach-agent $tcp0
#Specify TCP packet size
        Agent/TCP set packetSize_ 1000
#Start and Stop FTP Traffic
        $ns at 0.75 "$ftp0 start"
        $ns at 4.75 "$ftp0 stop"
#Stop the simulation
        $ns at 5.0 "finish"
#Run the simulation
        $ns run
```

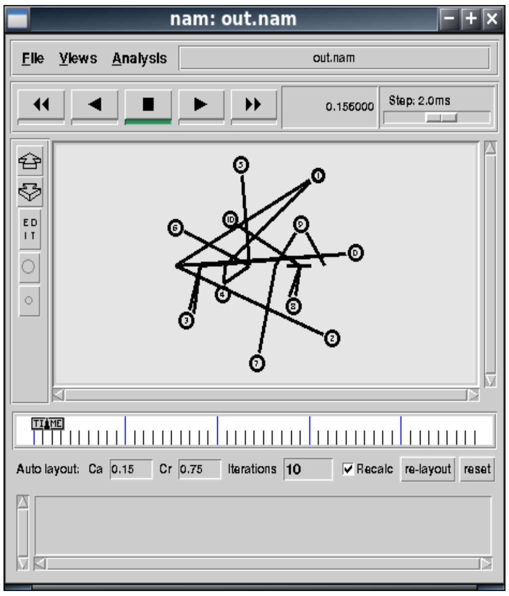
Ether.awk

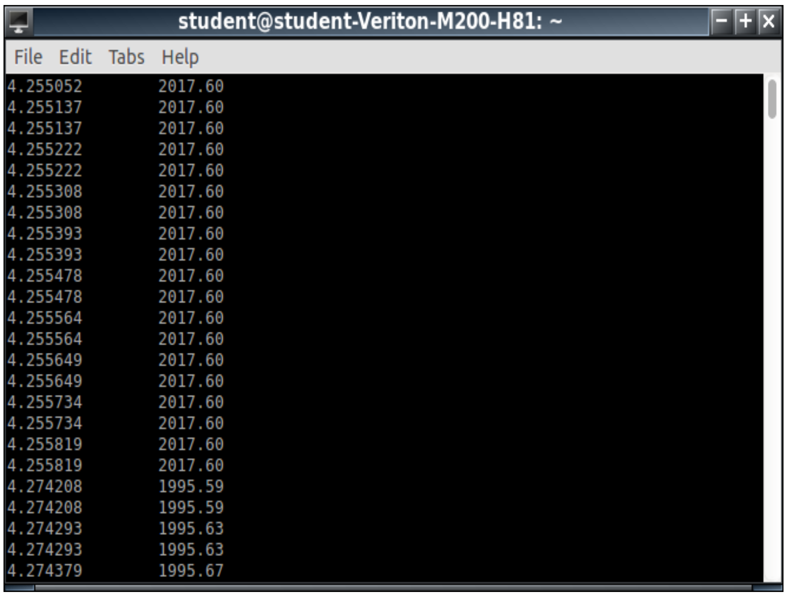
```
BEGIN
{
        sSize = 0;
        startTime = 5.0;
        stopTime = 0.1;
        Tput = 0;
}
{
        event = $1;
        time = $2;
        size = $6;
        if(event == "+")
        {
                if(time < startTime)
                {
                        startTime = time;
                }
        }
        if(event == "r")
        {
                if(time > stopTime)
                {
                        stopTime = time;
                } sSize += size;
        }
        Tput = (sSize / (stopTime-startTime))*(8/1000);
        printf("%f\t%.2f\n", time, Tput);
}
END
{
}
```

STEPS OF EXECUTION:

1. Save the file as <filename.tcl>.
2. Open the terminal (cmd prompt).
3. Go to the specific directory where the program is saved.
4. ns <filename.tcl>.
5. awk -f <filename.awk><filename.tr>.

OUTPUT:





```
4.255052        2017.60
4.255137        2017.60
4.255137        2017.60
4.255222        2017.60
4.255222        2017.60
4.255308        2017.60
4.255308        2017.60
4.255393        2017.60
4.255393        2017.60
4.255478        2017.60
4.255478        2017.60
4.255564        2017.60
4.255564        2017.60
4.255649        2017.60
4.255649        2017.60
4.255734        2017.60
4.255734        2017.60
4.255819        2017.60
4.255819        2017.60
4.274208        1995.59
4.274208        1995.59
4.274293        1995.63
4.274293        1995.63
4.274379        1995.67
```

RESULT: Thus Ethernet LAN was implemented using n(6 – 10) nodes.

PROGRAM NO: 10

Implement Ethernet LAN using n nodes and assign multiple traffic to the nodes and obtain congestion window from different sources/ destinations.

AIM:Implement Ethernet LAN using n nodes and assign multiple traffic to the nodes and obtain congestion window from different sources/ destinations

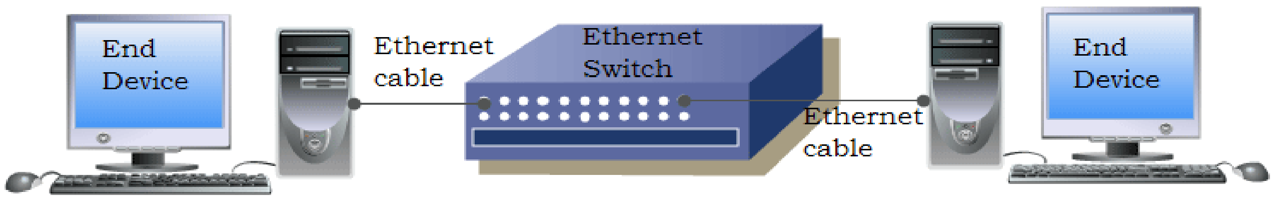
THEORY:

An Ethernet LAN is the combination of components that allows users to access applications and data, share resources and connect with other networks.

Common components of an Ethernet LAN are; User devices (such as Computers, PCs, Servers and Network printers), Network devices (LAN switches, hubs, firewalls, so on) and different types of media (such as Coaxial, UTP, and STP). Usually, these components are owned by the same company or organization which builds the Ethernet LAN.

Based on scalability, an Ethernet LAN can be categorized in two types; SOHO LAN and the Enterprise LAN

SOHO stands for Small Office/ Home Office. This is the smallest form of an Ethernet LAN. To build this LAN a device known as Ethernet LAN Switch is used. An Ethernet LAN Switch has many ports. To connect an end device or user device on one of these ports, a cable known as Ethernet cable is used.



Enterprise Ethernet LAN also uses the same technologies and protocols to build the network, but on a much larger scale. An Enterprise Ethernet LAN can span in entire building, campus or even a large geographical area.

Unlike the SOHO network which usually connects a small number of computers, an Enterprise LAN network can connect a large number of computers. Basically, an Enterprise Ethernet LAN is the extended version of SOHO LAN. For, example there are five SOHO LANs. If we connect all of these LANs together to create a large LAN, we create an Enterprise Ethernet LAN.

Network congestion in data networking and queueing theory is the reduced quality of service that occurs when a network node or link is carrying more data than it can handle. Typical effects include queueing delay, packet loss or the blocking of new connections.

Common causes of network congestion including:

- Over – subscription
- Poor network design / mis – configuration
- Over – utilized devices
- Faulty devices
- Security attack

The effects of a congested network include:

- **Delay:**

Also known as Latency, Delay is the time it takes for a destination to receive the packet sent by the sender. For example, the time it takes for a webpage to load is a result of how long it takes for the packets from the web server to get to the client. Another evidence of delay is the buffering you experience when watching a video, say on YouTube.

- **Packet Loss:**

While packets may take a while to get to their destination (delay), packet loss is an even more negative effect of network congestion. This is especially troubling for applications like Voice over IP (VoIP) that do not deal well with delay and packet loss, resulting in dropped calls and Call Detail Records, lag, robotic voices, and so on.

- **Timeouts:**

Network congestion can also result in timeouts in various applications. Since most connections will not stay up indefinitely waiting for packets to arrive, this can result in lost connections.

PROGRAM:

p4.tcl

```
#Set ns simulation
        set ns [new Simulator]
#Open trace-file
        set tf [open lab4.tr w]
        $ns trace-all $tf
#Open nam file
        set nf [open lab4.nam w]
        $ns namtrace-all $nf
#Create four nodes
        set n0 [$ns node]
        set n1 [$ns node]
        set n2 [$ns node]
        set n3 [$ns node]
$ns color 1 "blue"
$ns color 2 "red"
$ns at 0.0 "$n3 color blue"
$ns at 0.0 "$n1 color blue"
        $ns at 0.0 "$n0 label Source/Tcp0"
        $ns at 0.0 "$n2 label Source/Tcp2"
        $ns at 0.0 "$n1 label Destination/Sink1"
        $ns at 0.0 "$n3 label Destination/Sink3"
$ns make-lan "$n0 $n1 $n2 $n3" 10mb 10ms LL Queue/DropTail Mac/802_3
#Setup TCP connection
        set tcp0 [new Agent/TCP/Reno]
        $ns attach-agent $n0 $tcp0
#Set FTP over tcp connection
        set ftp0 [new Application/FTP]
        $ftp0 attach-agent $tcp0
set sink3 [new Agent/TCPSink]
        $ns attach-agent $n3 $sink3
#Setup TCP connection
        set tcp2 [new Agent/TCP]
        $ns attach-agent $n2 $tcp2
#Set FTP over tcp connection
        set ftp2 [new Application/FTP]
        $ftp2 attach-agent $tcp2
set sink1 [new Agent/TCPSink]
        $ns attach-agent $n1 $sink1
$ns connect $tcp0 $sink3
        $ns connect $tcp2 $sink1
$tcp0 set class_ 1
$tcp2 set class_ 2
### To trace the congestion window###
        set file1 [open file1.tr w]
        $tcp0 attach $file1
        $tcp0 trace cwnd_
        #$tcp0 set maxcwnd_ 10
        set file2 [open file2.tr w]
        $tcp2 attach $file2
        $tcp2 trace cwnd_
#Define the finish procedure
        proc finish {} {
        global ns tf nf
        $ns flush-trace
        exec nam lab4.nam &
        close $nf
        close $tf
        exit 0
        }
#Scheduling the events
        $ns at 0.1 "$ftp0 start"
        $ns at 1.5 "$ftp0 stop"
        $ns at 2 "$ftp0 start"
        $ns at 3 "$ftp0 stop"
        $ns at 0.2 "$ftp2 start"
        $ns at 2 "$ftp2 stop"
        $ns at 2.5 "$ftp2 start"
        $ns at 4 "$ftp2 stop"
        $ns at 5.0 "finish"
$ns run
```

p4.awk

```
BEGIN
{
}
{
        if($6==cwnd_")
                printf("%f\t%f\t\n",$1,$7);
}
END
```

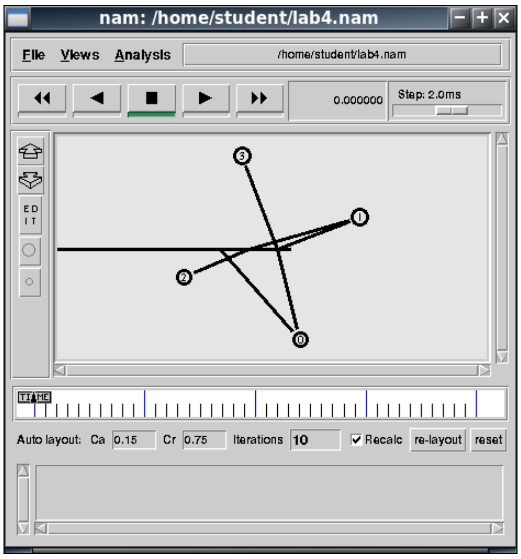
{

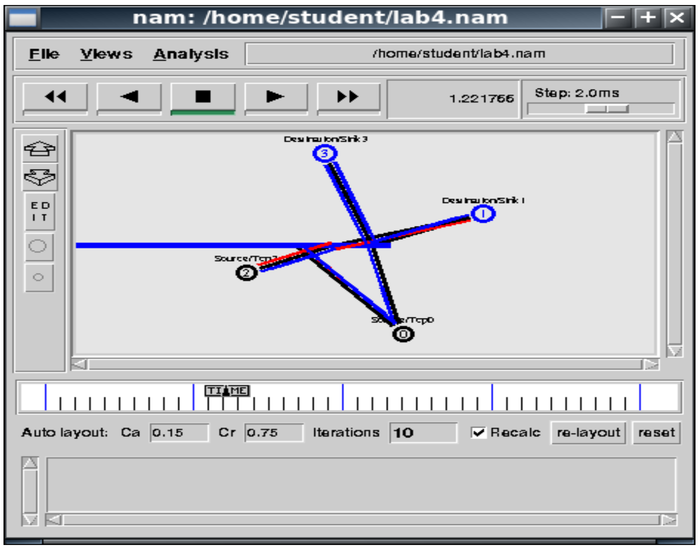
}

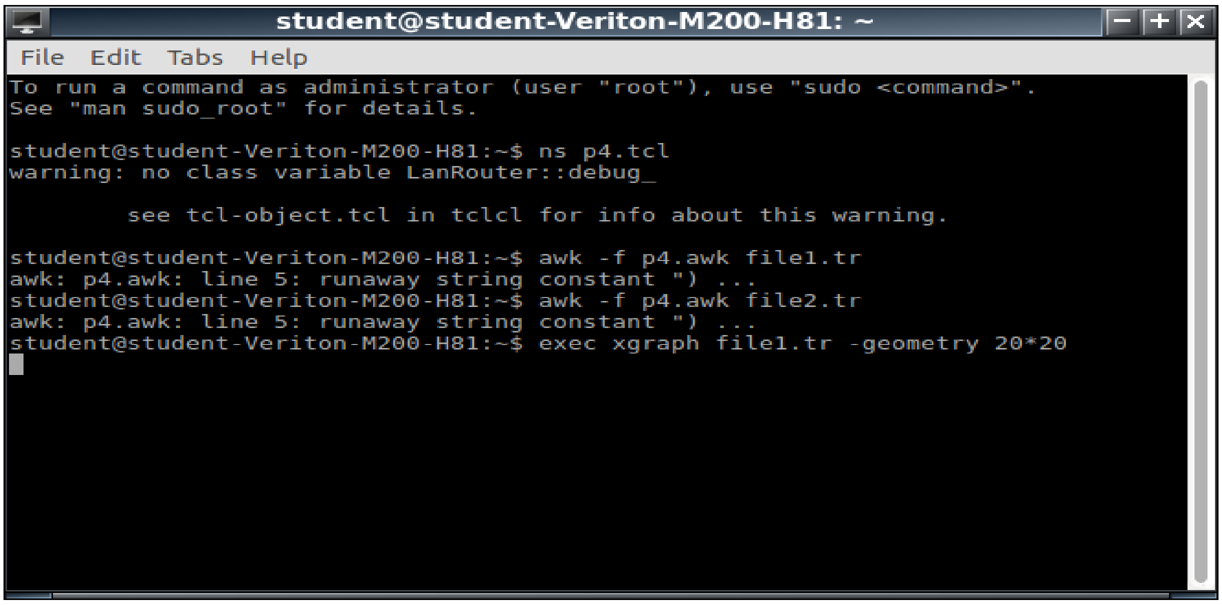
STEPS OF EXECUTION

1. Save the file as <filename.tcl>and <filename.awk>.
2. Open the terminal (cmd prompt).
3. Go to the specific directory where the program is saved.
4. ns <filename.tcl>.

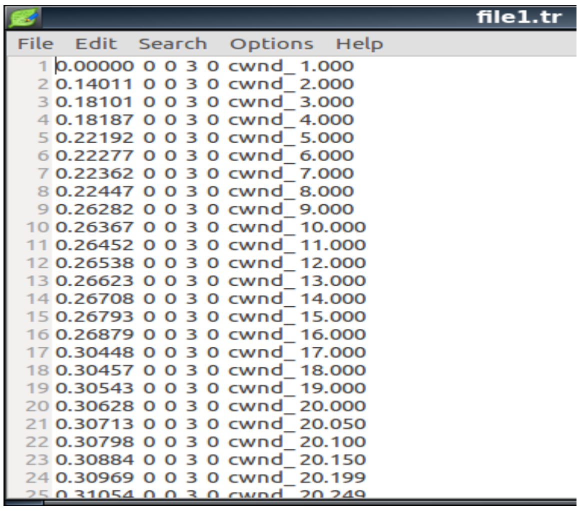
OUTPUT


nam: /home/student/lab4.nam
File Views Analysis
/home/student/lab4.nam
0.000000
Step: 2.0ms
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc re-layout reset

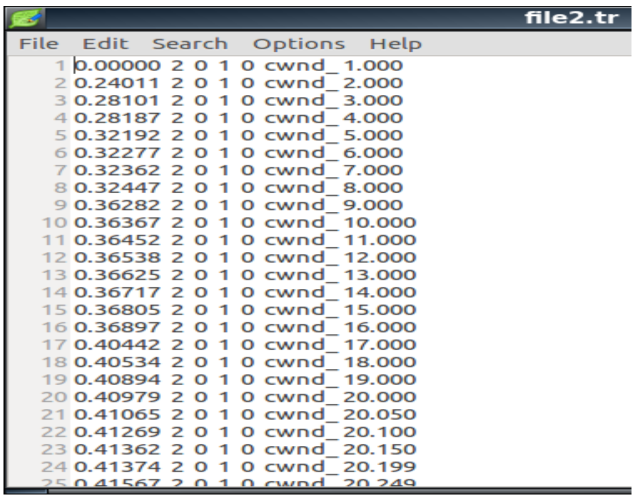




```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

student@student-Veriton-M200-H81:~$ ns p4.tcl
warning: no class variable LanRouter::debug_

        see tcl-object.tcl in tclcl for info about this warning.

student@student-Veriton-M200-H81:~$ awk -f p4.awk file1.tr
awk: p4.awk: line 5: runaway string constant ") ...
student@student-Veriton-M200-H81:~$ awk -f p4.awk file2.tr
awk: p4.awk: line 5: runaway string constant ") ...
student@student-Veriton-M200-H81:~$ exec xgraph file1.tr -geometry 20*20
```



```
0.00000 0 0 3 0 cwnd_ 1.000
0.14011 0 0 3 0 cwnd_ 2.000
0.18101 0 0 3 0 cwnd_ 3.000
0.18187 0 0 3 0 cwnd_ 4.000
0.22192 0 0 3 0 cwnd_ 5.000
0.22277 0 0 3 0 cwnd_ 6.000
0.22362 0 0 3 0 cwnd_ 7.000
0.22447 0 0 3 0 cwnd_ 8.000
0.26282 0 0 3 0 cwnd_ 9.000
0.26367 0 0 3 0 cwnd_ 10.000
0.26452 0 0 3 0 cwnd_ 11.000
0.26538 0 0 3 0 cwnd_ 12.000
0.26623 0 0 3 0 cwnd_ 13.000
0.26708 0 0 3 0 cwnd_ 14.000
0.26793 0 0 3 0 cwnd_ 15.000
0.26879 0 0 3 0 cwnd_ 16.000
0.30448 0 0 3 0 cwnd_ 17.000
0.30457 0 0 3 0 cwnd_ 18.000
0.30543 0 0 3 0 cwnd_ 19.000
0.30628 0 0 3 0 cwnd_ 20.000
0.30713 0 0 3 0 cwnd_ 20.050
0.30798 0 0 3 0 cwnd_ 20.100
0.30884 0 0 3 0 cwnd_ 20.150
0.30969 0 0 3 0 cwnd_ 20.199
0.31054 0 0 3 0 cwnd_ 20.249
```



```
0.00000 2 0 1 0 cwnd_ 1.000
0.24011 2 0 1 0 cwnd_ 2.000
0.28101 2 0 1 0 cwnd_ 3.000
0.28187 2 0 1 0 cwnd_ 4.000
0.32192 2 0 1 0 cwnd_ 5.000
0.32277 2 0 1 0 cwnd_ 6.000
0.32362 2 0 1 0 cwnd_ 7.000
0.32447 2 0 1 0 cwnd_ 8.000
0.36282 2 0 1 0 cwnd_ 9.000
0.36367 2 0 1 0 cwnd_ 10.000
0.36452 2 0 1 0 cwnd_ 11.000
0.36538 2 0 1 0 cwnd_ 12.000
0.36625 2 0 1 0 cwnd_ 13.000
0.36717 2 0 1 0 cwnd_ 14.000
0.36805 2 0 1 0 cwnd_ 15.000
0.36897 2 0 1 0 cwnd_ 16.000
0.40442 2 0 1 0 cwnd_ 17.000
0.40534 2 0 1 0 cwnd_ 18.000
0.40894 2 0 1 0 cwnd_ 19.000
0.40979 2 0 1 0 cwnd_ 20.000
0.41065 2 0 1 0 cwnd_ 20.050
0.41269 2 0 1 0 cwnd_ 20.100
0.41362 2 0 1 0 cwnd_ 20.150
0.41374 2 0 1 0 cwnd_ 20.199
0.41567 2 0 1 0 cwnd_ 20.249
```

RESULT:Implementation of Ethernet LAN using n nodes and assigning multiple traffic to the nodes and obtaining congestion window from different sources/ destinations is done successfully.

PROGRAM NO: 11

Implement ESS with transmission nodes in Wireless LAN and obtain performance parameters.

AIM: Implement ESS with transmission nodes in Wireless LAN and obtain performance parameters.

THEORY:

An extended service set (ESS) is one or more interconnected basic service sets (BSSs) and their associated LANs. Each BSS consists of a single access point (AP) together with all wireless client devices (stations, also called STAs) creating a local or enterprise 802.11 wireless LAN (WLAN). To the logical link control layer (part of layer 2 of the 7 – layer OSI Reference Model) the ESS appears as a solitary BSS at any one of the STAs.

The most basic BSS consists of one AP and one STA.

An extended service set, consisting of a set of BSSs, must have a common service set identifier (SSID). The BSSs can all work on the same or different channels. This helps to boost the signal throughout the wireless network.

A single service set consists of all STAs receiving signals from a given AP and creates an 802.11 wireless LAN (WLAN). Each STA may receive a signal from several APs within their range. Depending on its configuration each STA can, manually or automatically, select the network with which to associate. And multiple APs may share the same SSID as part of an extended service set.

Although not part of the 802.11 standard, some wireless APs may broadcast multiple SSIDs, allowing virtual access points to be created – each with their own security and network settings.

PROGRAM:

ess_prg1.tcl

```
#Create a NS simulator object
        set ns [new Simulator]
#Setup topography object
        set topo [new Topography]
        $topo load_flatgrid 1500 1500
#Open the NS trace file
        set tracefile [open p4.tr w]
        $ns trace-all $tracefile
#Open the NAM trace file
        set namfile [open p4.nam w]
        $ns namtrace-all $namfile
        $ns namtrace-all-wireless $namfile 1500 1500
#Mobile node parameter setup
```

```
        $ns node-config -adhocRouting DSDV \
                -llType LL \
                -macType Mac/802_11 \
                -ifqType Queue/DropTail \
                -ifqLen 50 \
                -phyType Phy/WirelessPhy \
                -channelType Channel/WirelessChannel \
                -propType Propagation/TwoRayGround \
                -antType Antenna/OmniAntenna \
                -topoInstance $topo \
                -agentTrace ON \
                -routerTrace ON
#Nodes Definition
#Create god object [Creates 6 nodes]
create-god 6   #god- general operation director
        set n0 [$ns node]
        $n0 set X_ 630
        $n0 set Y_ 501
        $n0 set Z_ 0.0
        $ns intial_node_pos $n0 20
        set n1 [$ns node]
        $n1 set X_ 454
        $n1 set Y_ 340
        $n1 set Z_ 0.0
        $ns intial_node_pos $n1 20
        set n2 [$ns node]
        $n2 set X_ 785
        $n2 set Y_ 326
        $n2 set Z_ 0.0
        $ns intial_node_pos $n2 20
        set n3 [$ns node]
        $n3 set X_ 270
        $n3 set Y_ 190
        $n3 set Z_ 0.0
        $ns intial_node_pos $n3 20
        set n4 [$ns node]
        $n4 set X_ 539
        $n4 set Y_ 131
        $n4 set Z_ 0.0
        $ns intial_node_pos $n4 20
        set n5 [$ns node]
        $n5 set X_ 964
        $n5 set Y_ 177
        $n5 set Z_ 0.0
        $ns intial_node_pos $n5 20
#Agent Definition
#Setup UDP connection
        set udp0 [new Agent/UDP]
        $ns attach-agent $n0 $udp0
        set null1 [new Agent/Null]
        $ns attach-agent $n4 $null1
        $ns connect $udp0 $null1
        $udp0 set packetSize_ 1500
#Setup TCP connection
        set tcp0 [new Agent/TCP]
        $ns attach-agent $n0 $tcp0
        set sink1 [new Agent/TCPSink]
        $ns attach-agent $n5 $sink1
        $ns connect $tcp0 $sink1
#Application Definition
#Setup CBR Application over UDP Connection
        set cbr0 [new Application/Traffic/CBR]
        $cbr0 attach-agent $udp0
        $cbr0 set packetSize_ 1000
        $cbr0 set rate_ 1.0Mb
        $cbr0 set random_null
#Setup FTP Application over TCP connection
        set ftp0 [new Apllication/FTP]
        $ftp0 attach-agent $tcp0
#Termination
#Define finish procedure
        proc finish { } {
        global ns tracefile namfile
        $ns flush-trace
        close $tracefile
        close $namfile

        exec nam p4.nam &
        exec echo "Number of packets dropped is : "&
        exec grep -c "^/home/student/dccn_lab"  p4.tr &
        exit 0
}
        $ns at 1.0 "$cbr0 start"
        $ns at 2.0 "$ftp0 start"
        $ns at 180.0 "$ftp0 stop"
        $ns at 200.0 "$cbr0 stop"
        $ns at 200.0 "finish"
        $ns at 70 "$n4 set dest 100 60 20"
        $ns at 100 "$n4 set dest 700 300 20"
        $ns at 150 "$n4 set dest 900 200 20"
        $ns run
```

ess_prg1.awk

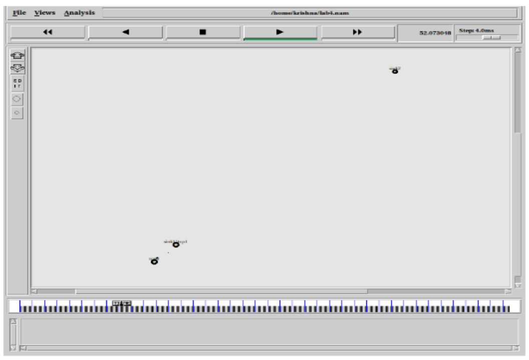
```
BEGIN
{
        count1=0
        count2=0
        pack1=0
        pack2=0
        time1=0
        time2=0
}
```

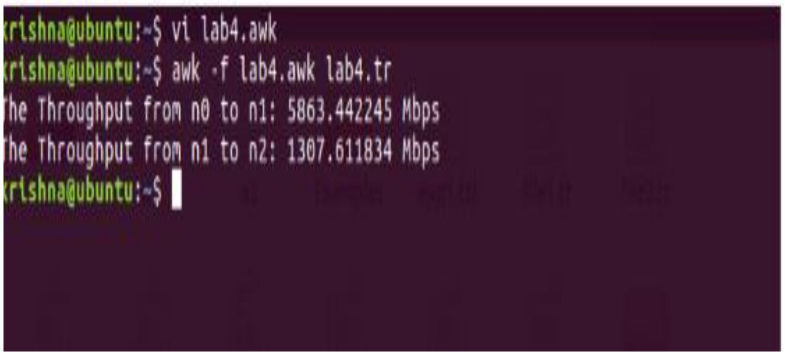
```
if($1=="r"&&$3=="_1_"&&$4=="RTR")
{
        count1++
        pack1=pack1+$8
        time1=$2
}
if($1=="r"&&$3=="_2_"&&$4=="RTR")
{
        count2++
        pack2=pack2+$8
        time2=$2
}
END
{
        printf("The throughput from n0 to n1: %fMbps\n",((count1*pack1*8)/(time1*1000000));
        printf("The throughput from n1 to n2: %fMbps\n",((count2*pack2*8)/(time2*1000000));
}
```

STEPS OF EXECUTION

1. Save the file as <filename.tcl>and <filename.awk>**.**
2. Open the terminal (cmd prompt)**.**
3. Go to the specific directory where the program is saved**.**
4. ns <filename.tcl>

OUTPUT





```
krishna@ubuntu:~$ vi lab4.awk
krishna@ubuntu:~$ awk -f lab4.awk lab4.tr
The Throughput from n0 to n1: 5863.442245 Mbps
The Throughput from n1 to n2: 1307.611834 Mbps
krishna@ubuntu:~$
```

RESULT: Thus the Implementation of ESS with transmission nodes in Wireless LAN is done successfully.

PROGRAM No: 12

Implementation of Link state routing algorithm.

AIM: Implementation of Link state routing algorithm.

THEORY:

In link state routing, each router shares its knowledge of its neighbourhood with every other router in the internet work.

i. Knowledge about Neighbourhood: Instead of sending its entire routing table a router sends info about its neighbourhood only.

ii. To all Routers: each router sends this information to every other router on the internet work not just to its neighbour. It does so by a process called flooding.

iii. Information sharing when there is a change: Each router sends out information about the neighbours when there is change.

PROCEDURE:

The Dijkstra algorithm follows four steps to discover what is called the shortest path tree(routing table) for each router .The algorithm begins to build the tree by identifying its roots. The root router's trees the router itself. The algorithm then attaches all nodes that can be reached from the root. The algorithm compares the tree's temporary arcs and identifies the arc with the lowest cumulative cost. This arc and the node to which it connects are now a permanent part of the shortest path tree. The algorithm examines the database and identifies every node that can be reached from its chosen node. These nodes and their arcs are added temporarily to the tree.

The last two steps are repeated until every node in the network has become a permanent part of the tree.

ALGORITHM:

1. Create a simulator object.
2. Define different colors for different data flows.
3. Open a nam trace file and define finish procedure then close the trace file, and execute nam on trace file.
4. Create n number of nodes using for loop.
5. Create duplex links between the nodes.
6. Setup UDP Connection between n(0) and n(5).
7. Setup another UDP connection between n(1) and n(5).
8. Apply CBR Traffic over both UDP connections.
9. Choose Link state routing protocol to transmit data from sender to receiver.
10. Schedule events and run the program.

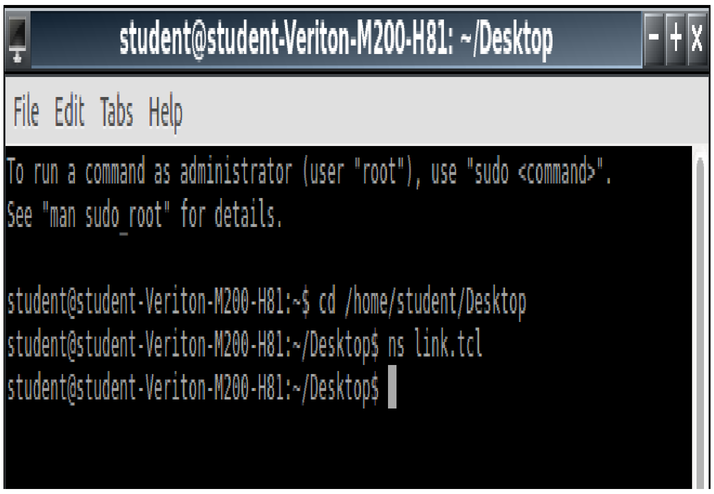
PROGRAM:

```
set udp0 [new Agent/UDP]
$ns attach-agent $n(0) $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
set null0 [new Agent/Null]
$ns attach-agent $n(5) $null0
$ns connect $udp0 $null0
set udp1 [new Agent/UDP]
$ns attach-agent $n(1) $udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1
set null0 [new Agent/Null]
$ns attach-agent $n(5) $null0
$ns connect $udp1 $null0
$ns rtproto LS
$ns rtmodel-at 10.0 down $n(11) $n(5)
$ns rtmodel-at 15.0 down $n(7) $n(6)
$ns rtmodel-at 30.0 up $n(11) $n(5)
$ns rtmodel-at 20.0 up $n(7) $n(6)
$udp0 set fid_ 1
$udp1 set fid_ 2
$ns color 1 Red
$ns color 2 Green
$ns at 1.0 "$cbr0 start"
$ns at 2.0 "$cbr1 start"
$ns at 45 "finish"
$ns run
```

Steps of Execution:

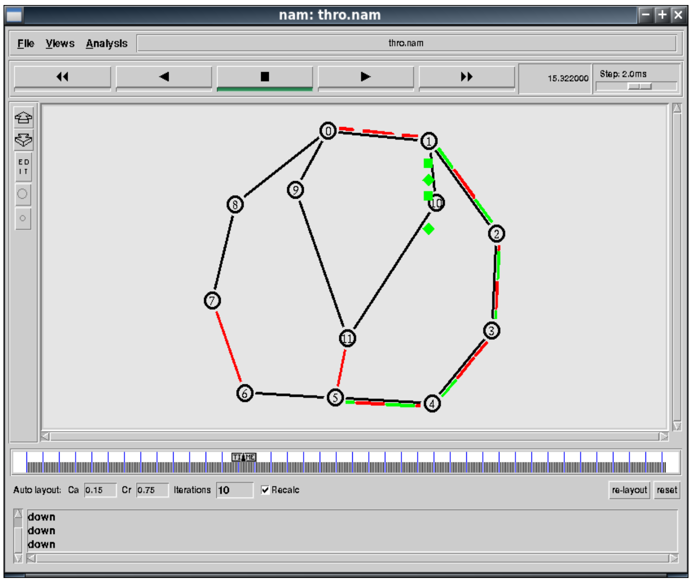
1. Save the file as <filename.tcl>**.**

2. Open the terminal(cmd prompt).
3. Go to the specific directory where the program is saved.
4. ns <filename.tcl>.

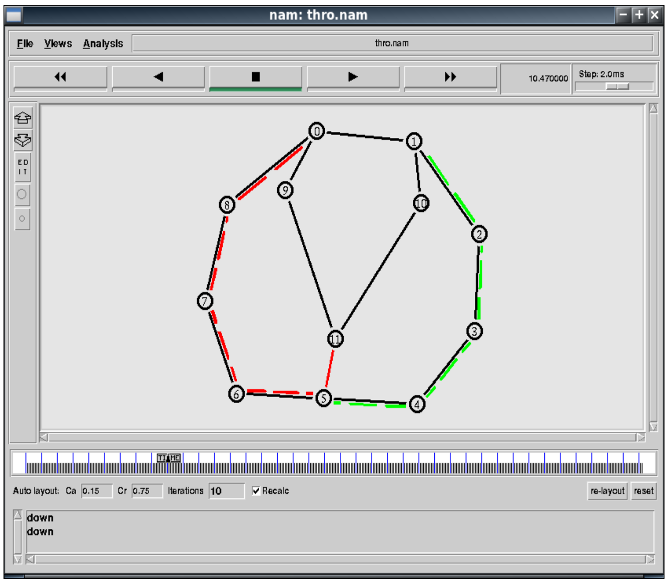


```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

student@student-Veriton-M200-H81:~$ cd /home/student/Desktop
student@student-Veriton-M200-H81:~/Desktop$ ns link.tcl
student@student-Veriton-M200-H81:~/Desktop$ 
```

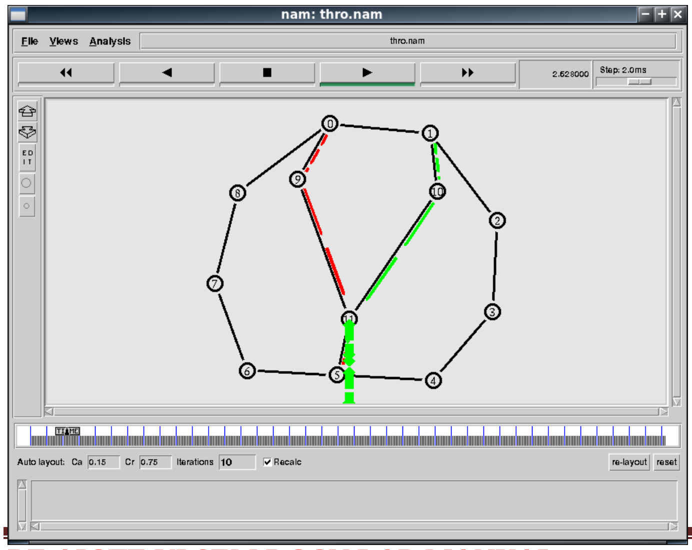
Output Screen:

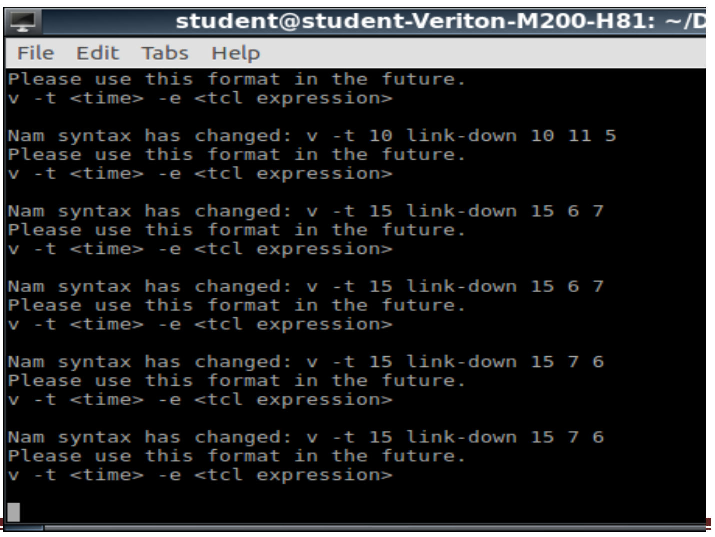


```
down
down
down
```

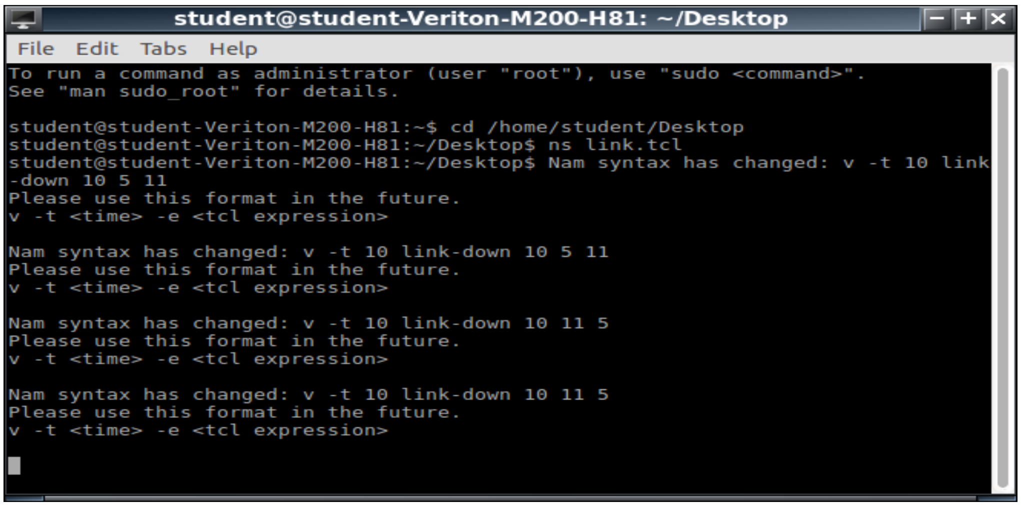


```
down
down
```





```
Please use this format in the future.
v -t <time> -e <tcl expression>

Nam syntax has changed: v -t 10 link-down 10 11 5
Please use this format in the future.
v -t <time> -e <tcl expression>

Nam syntax has changed: v -t 15 link-down 15 6 7
Please use this format in the future.
v -t <time> -e <tcl expression>

Nam syntax has changed: v -t 15 link-down 15 6 7
Please use this format in the future.
v -t <time> -e <tcl expression>

Nam syntax has changed: v -t 15 link-down 15 7 6
Please use this format in the future.
v -t <time> -e <tcl expression>

Nam syntax has changed: v -t 15 link-down 15 7 6
Please use this format in the future.
v -t <time> -e <tcl expression>

```



```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

student@student-Veriton-M200-H81:~$ cd /home/student/Desktop
student@student-Veriton-M200-H81:~/Desktop$ ns link.tcl
student@student-Veriton-M200-H81:~/Desktop$ Nam syntax has changed: v -t 10 link
-down 10 5 11
Please use this format in the future.
v -t <time> -e <tcl expression>

Nam syntax has changed: v -t 10 link-down 10 5 11
Please use this format in the future.
v -t <time> -e <tcl expression>

Nam syntax has changed: v -t 10 link-down 10 11 5
Please use this format in the future.
v -t <time> -e <tcl expression>

Nam syntax has changed: v -t 10 link-down 10 11 5
Please use this format in the future.
v -t <time> -e <tcl expression>
```

RESULT:Thus Implementation of Link state routing algorithm is done successfully.

ADDITIONAL EXPERIMENTS

EXPERIMENT NO. 1

AIM: Running and using services/commands like tcpdump, netstat, ifconfig, nslookup, FTP, TELNET and traceroute. Capture ping and trace route PDUs using a network protocol.

1.TCPDUMP

tcpdump is a most powerful and widely used command-line packets sniffer or package analyzer tool which is used to capture or filter **TCP/IP** packets that received or transferred over a network on a specific interface. It is available under most of the **Linux/Unix** based operating systems. tcpdump also gives us a option to save captured packets in a file for future analysis. It saves the file in a **pcap** format, that can be viewed by tcpdump command or a open source GUI based tool called Wireshark (Network Protocol Analyzier) that reads tcpdump **pcap** format files.

How to Install tcpdump in Linux

Many of Linux distributions already shipped with **tcpdump** tool, if in case you don't have it on systems, you can install it using following Yum command.

#yum install tcpdump

Once **tcpdump** tool is installed on systems, you can continue to browse following commands with their examples.

1. Capture Packets from Specific Interface

The command screen will scroll up until you interrupt and when we execute **tcpdump** command it will captures from all the interfaces, however with **-i** switch only capture from desire interface.

```
# tcpdump -i eth0

tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
11:33:31.976358 IP 172.16.25.126.ssh > 172.16.25.125.apwi-rxspooler: Flags [P.], seq
3500440357:3500440553, ack 3652628334, win 18760, length 196
11:33:31.976603 IP 172.16.25.125.apwi-rxspooler > 172.16.25.126.ssh: Flags [.], ack 196, win 64487,
length 0
11:33:31.977243 ARP, Request who-has tecmint.com tell 172.16.25.126, length 28
11:33:31.977359 ARP, Reply tecmint.com is-at 00:14:5e:67:26:1d (oui Unknown), length 46
11:33:31.977367 IP 172.16.25.126.54807 > tecmint.com: 4240+ PTR? 125.25.16.172.in-addr.arpa. (44)
11:33:31.977599 IP tecmint.com > 172.16.25.126.54807: 4240 NXDomain 0/1/0 (121)
```

```
11:33:31.977742 IP 172.16.25.126.44519 > tecmint.com: 40988+ PTR? 126.25.16.172.in-addr.arpa. (44)
11:33:32.028747 IP 172.16.20.33.netbios-ns > 172.16.31.255.netbios-ns: NBT UDP PACKET(137):
QUERY; REQUEST; BROADCAST
11:33:32.112045 IP 172.16.21.153.netbios-ns > 172.16.31.255.netbios-ns: NBT UDP PACKET(137):
QUERY; REQUEST; BROADCAST
11:33:32.115606 IP 172.16.21.144.netbios-ns > 172.16.31.255.netbios-ns: NBT UDP PACKET(137):
QUERY; REQUEST; BROADCAST
11:33:32.156576 ARP, Request who-has 172.16.16.37 tell old-oraclehp1.midcorp.mid-day.com, length 46
11:33:32.348738 IP tecmint.com > 172.16.25.126.44519: 40988 NXDomain 0/1/0 (121)
```

2. Capture Only N Number of Packets

When you run **tcpdump** command it will capture all the packets for specified interface, until you **Hit** cancel button. But using **-c** option, you can capture specified number of packets. The below example will only capture **6** packets.

```
# tcpdump -c 5 -i eth0

tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
11:40:20.281355 IP 172.16.25.126.ssh > 172.16.25.125.apwi-rxspooler: Flags [P.], seq
3500447285:3500447481, ack 3652629474, win 18760, length 196
11:40:20.281586 IP 172.16.25.125.apwi-rxspooler > 172.16.25.126.ssh: Flags [.], ack 196, win 65235,
length 0
11:40:20.282244 ARP, Request who-has tecmint.com tell 172.16.25.126, length 28
11:40:20.282360 ARP, Reply tecmint.com is-at 00:14:5e:67:26:1d (oui Unknown), length 46
11:40:20.282369 IP 172.16.25.126.53216 >tecmint.com.domain: 49504+ PTR? 125.25.16.172.in-addr.arpa.
(44)
11:40:20.332494 IP tecmint.com.netbios-ssn > 172.16.26.17.nimaux: Flags [P.], seq
3058424861:3058424914, ack 693912021, win 64190, length 53 NBT Session Packet: Session Message
6 packets captured
23 packets received by filter
0 packets dropped by kernel
```

3. Print Captured Packets in ASCII

The below **tcpdump** command with option **-A** displays the package in **ASCII** format. It is a character-encoding scheme format.

```
# tcpdump -A -i eth0
```

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
09:31:31.347508 IP 192.168.0.2.ssh > 192.168.0.1.nokia-ann-ch1: Flags [P.], seq 3329372346:3329372542,
ack 4193416789, win 17688, length 196
M.r0...vUP.E.X.......~.%..>N..oFk.........KQ..)Eq.d.,....r^l......m\.oyE....-
....g~m..Xy.6..1.....c.O.@...o_..J....i.*.....2f.mQH...Q.c...6....9.v.gb........;..4.).UiCY]..9..x.)..Z.XF....'|..E......M.
.u.5.......ul
09:31:31.347760 IP 192.168.0.1.nokia-ann-ch1 > 192.168.0.2.ssh: Flags [.], ack 196, win 64351, length 0
M....vU.r1~P.._..........
^C09:31:31.349560 IP 192.168.0.2.46393 > b.resolvers.Level3.net.domain: 11148+ PTR? 1.0.168.192.in-
addr.arpa. (42)
E..F..@.@............9.5.2.f+............1.0.168.192.in-addr.arpa.....

3 packets captured
11 packets received by filter
0 packets dropped by kernel
```

4. Display Available Interfaces

To list number of available interfaces on the system, run the following command with **-D** option.

```
# tcpdump -D

 1.eth0
2.eth1
3.usbmon1 (USB bus number 1)
4.usbmon2 (USB bus number 2)
5.usbmon3 (USB bus number 3)
6.usbmon4 (USB bus number 4)
7.usbmon5 (USB bus number 5)
8.any (Pseudo-device that captures on all interfaces)
9.lo
```

5. Display Captured Packets in HEX and ASCII

The following command with option **-XX** capture the data of each packet, including its link level header in **HEX** and **ASCII** format.

```
# tcpdump -XX -i eth0

11:51:18.974360 IP 172.16.25.126.ssh > 172.16.25.125.apwi-rxspooler: Flags [P.], seq
3509235537:3509235733, ack 3652638190, win 18760, length 196
        0x0000:  b8ac 6f2e 57b3 0001 6c99 1468 0800 4510  ..o.W...l..h..E.
        0x0010:  00ec 8783 4000 4006 275d ac10 197e ac10  ....@.@.']...~..
        0x0020:  197d 0016 1129 d12a af51 d9b6 d5ee 5018  .}...).*.Q....P.
        0x0030:  4948 8bfa 0000 0e12 ea4d 22d1 67c0 f123  IH.......M".g..#
        0x0040:  9013 8f68 aa70 29f3 2efc c512 5660 4fe8  ...h.p).....V`O.
        0x0050:  590a d631 f939 dd06 e36a 69ed cac2 95b6  Y..1.9...ji.....
        0x0060:  f8ba b42a 344b 8e56 a5c4 b3a2 ed82 c3a1  ...*4K.V........
        0x0070:  80c8 7980 11ac 9bd7 5b01 18d5 8180 4536  ..y.....[.....E6
        0x0080:  30fd 4f6d 4190 f66f 2e24 e877 ed23 8eb0  0.OmA..o.$.w.#..
        0x0090:  5a1d f3ec 4be4 e0fb 8553 7c85 17d9 866f  Z...K....S|....o
        0x00a0:  c279 0d9c 8f9d 445b 7b01 81eb 1b63 7f12  .y....D[{....c..
        0x00b0:  71b3 1357 52c7 cf00 95c6 c9f6 63b1 ca51  q..WR.......c..Q
        0x00c0:  0ac6 456e 0620 38e6 10cb 6139 fb2a a756  ..En..8...a9.*.V
        0x00d0:  37d6 c5f3 f5f3 d8e8 3316 d14f d7ab fd93  7.......3..O....
        0x00e0:  1137 61c1 6a5c b4d1 ddda 380a f782 d983  .7a.j\....8.....
        0x00f0:  62ff a5a9 bb39 4f80 668a                 b....9O.f.
11:51:18.974759 IP 172.16.25.126.60952 > mddc-01.midcorp.mid-day.com.domain: 14620+ PTR?
125.25.16.172.in-addr.arpa. (44)
        0x0000:  0014 5e67 261d 0001 6c99 1468 0800 4500  ..^g&...l..h..E.
        0x0010:  0048 5a83 4000 4011 5e25 ac10 197e ac10  .HZ.@.@.^%...~..
        0x0020:  105e ee18 0035 0034 8242 391c 0100 0001  .^...5.4.B9.....
        0x0030:  0000 0000 0000 0331 3235 0232 3502 3136  .......125.25.16
        0x0040:  0331 3732 0769 6e2d 6164 6472 0461 7270  .172.in-addr.arp
        0x0050:  6100 000c 0001                           a.....
```

6. Capture and Save Packets in a File

As we said, that **tcpdump** has a feature to capture and save the file in a **.pcap** format, to do this just execute command with **-w** option.

```
# tcpdump -w 0001.pcap -i eth0

tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
```

```
4 packets captured
4 packets received by filter
0 packets dropped by kernel
```

7. Read Captured Packets File

To read and analyze captured packet **0001.pcap** file use the command with **-r** option, as shown below.

```
# tcpdump -r 0001.pcap

reading from file 0001.pcap, link-type EN10MB (Ethernet)
09:59:34.839117 IP 192.168.0.2.ssh > 192.168.0.1.nokia-ann-ch1: Flags [P.], seq 3353041614:3353041746,
ack 4193563273, win 18760, length 132
09:59:34.963022 IP 192.168.0.1.nokia-ann-ch1 > 192.168.0.2.ssh: Flags [.], ack 132, win 65351, length 0
09:59:36.935309 IP 192.168.0.1.netbios-dgm > 192.168.0.255.netbios-dgm: NBT UDP PACKET(138)
09:59:37.528731 IP 192.168.0.1.nokia-ann-ch1 > 192.168.0.2.ssh: Flags [P.], seq 1:53, ack 132, win 65351,
length 5
```

8. Capture IP address Packets

To capture packets for a specific interface, run the following command with option **-n**.

```
# tcpdump -n -i eth0

tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
12:07:03.952358 IP 172.16.25.126.ssh > 172.16.25.125.apwi-rxspooler: Flags [P.], seq
3509512873:3509513069, ack 3652639034, win 18760, length 196
12:07:03.952602 IP 172.16.25.125.apwi-rxspooler > 172.16.25.126.ssh: Flags [.], ack 196, win 64171,
length 0
12:07:03.953311 IP 172.16.25.126.ssh > 172.16.25.125.apwi-rxspooler: Flags [P.], seq 196:504, ack 1, win
18760, length 308
12:07:03.954288 IP 172.16.25.126.ssh > 172.16.25.125.apwi-rxspooler: Flags [P.], seq 504:668, ack 1, win
18760, length 164
12:07:03.954502 IP 172.16.25.125.apwi-rxspooler > 172.16.25.126.ssh: Flags [.], ack 668, win 65535,
length 0
12:07:03.955298 IP 172.16.25.126.ssh > 172.16.25.125.apwi-rxspooler: Flags [P.], seq 668:944, ack 1, win
18760, length 276
```

```
12:07:03.955425 IP 172.16.23.16.netbios-ns > 172.16.31.255.netbios-ns: NBT UDP PACKET(137):
REGISTRATION; REQUEST; BROADCAST
12:07:03.956299 IP 172.16.25.126.ssh > 172.16.25.125.apwi-rxspooler: Flags [P.], seq 944:1236, ack 1, win
18760, length 292
12:07:03.956535 IP 172.16.25.125.apwi-rxspooler > 172.16.25.126.ssh: Flags [.], ack 1236, win 64967,
length 0
```

9. Capture only TCP Packets.

To capture packets based on **TCP** port, run the following command with option **tcp**.

```
# tcpdump -i eth0 tcp

tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
12:10:36.216358 IP 172.16.25.126.ssh > 172.16.25.125.apwi-rxspooler: Flags [P.], seq
3509646029:3509646225, ack 3652640142, win 18760, length 196
12:10:36.216592 IP 172.16.25.125.apwi-rxspooler > 172.16.25.126.ssh: Flags [.], ack 196, win 64687,
length 0
12:10:36.219069 IP 172.16.25.126.ssh > 172.16.25.125.apwi-rxspooler: Flags [P.], seq 196:504, ack 1, win
18760, length 308
12:10:36.220039 IP 172.16.25.126.ssh > 172.16.25.125.apwi-rxspooler: Flags [P.], seq 504:668, ack 1, win
18760, length 164
12:10:36.220260 IP 172.16.25.125.apwi-rxspooler > 172.16.25.126.ssh: Flags [.], ack 668, win 64215,
length 0
12:10:36.222045 IP 172.16.25.126.ssh > 172.16.25.125.apwi-rxspooler: Flags [P.], seq 668:944, ack 1, win
18760, length 276
12:10:36.223036 IP 172.16.25.126.ssh > 172.16.25.125.apwi-rxspooler: Flags [P.], seq 944:1108, ack 1, win
18760, length 164
12:10:36.223252 IP 172.16.25.125.apwi-rxspooler > 172.16.25.126.ssh: Flags [.], ack 1108, win 65535,
length 0
^C12:10:36.223461 IP mid-pay.midcorp.mid-day.com.netbios-ssn > 172.16.22.183.recipe: Flags [.], seq
283256512:283256513, ack 550465221, win 65531, length 1[|SMB]
```

10. Capture Packet from Specific Port

Let’s say you want to capture packets for specific port 22, execute the below command by specifying port number **22** as shown below.

# tcpdump -i eth0 port 22

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
10:37:49.056927 IP 192.168.0.2.ssh > 192.168.0.1.nokia-ann-ch1: Flags [P.], seq 3364204694:3364204890,
ack 4193655445, win 20904, length 196
10:37:49.196436 IP 192.168.0.2.ssh > 192.168.0.1.nokia-ann-ch1: Flags [P.], seq 4294967244:196, ack 1,
win 20904, length 248
10:37:49.196615 IP 192.168.0.1.nokia-ann-ch1 > 192.168.0.2.ssh: Flags [.], ack 196, win 64491, length 0
10:37:49.379298 IP 192.168.0.2.ssh > 192.168.0.1.nokia-ann-ch1: Flags [P.], seq 196:616, ack 1, win
20904, length 420
10:37:49.381080 IP 192.168.0.2.ssh > 192.168.0.1.nokia-ann-ch1: Flags [P.], seq 616:780, ack 1, win
20904, length 164
10:37:49.381322 IP 192.168.0.1.nokia-ann-ch1 > 192.168.0.2.ssh: Flags [.], ack 780, win 65535, length 0
```

11. Capture Packets from source IP

To capture packets from source **IP**, say you want to capture packets for **192.168.0.2**, use the command as follows.

```
# tcpdump -i eth0 src 192.168.0.2

tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
10:49:15.746474 IP 192.168.0.2.ssh > 192.168.0.1.nokia-ann-ch1: Flags [P.], seq 3364578842:3364579038,
ack 4193668445, win 20904, length 196
10:49:15.748554 IP 192.168.0.2.56200 > b.resolvers.Level3.net.domain: 11289+ PTR? 1.0.168.192.in-
addr.arpa. (42)
10:49:15.912165 IP 192.168.0.2.56234 > b.resolvers.Level3.net.domain: 53106+ PTR? 2.0.168.192.in-
addr.arpa. (42)
10:49:16.074720 IP 192.168.0.2.33961 > b.resolvers.Level3.net.domain: 38447+ PTR? 2.2.2.4.in-addr.arpa.
(38)
```

12. Capture Packets from destination IP

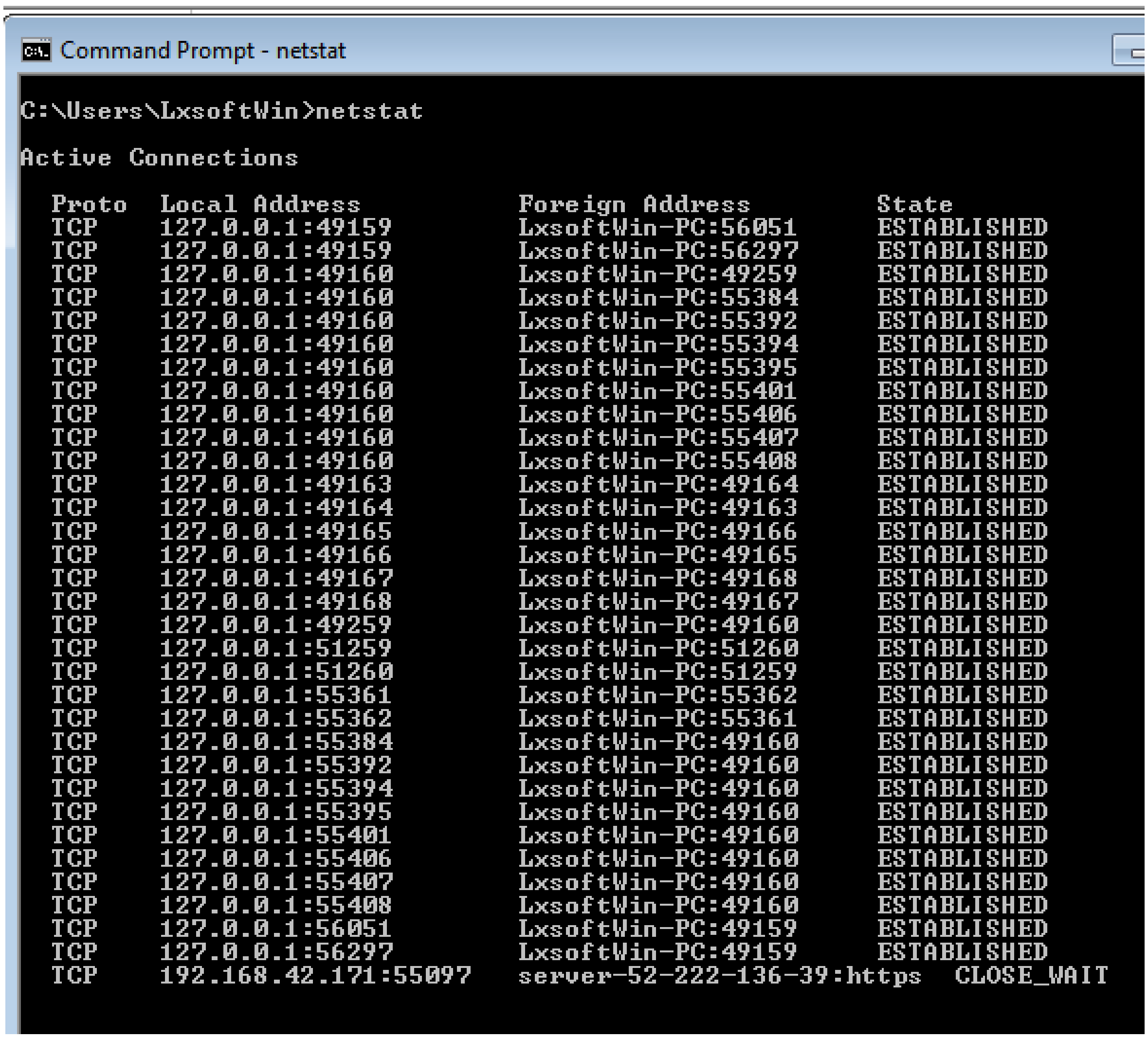
To capture packets from destination **IP**, say you want to capture packets for **50.116.66.139**, use the command as follows.

```
# tcpdump -i eth0 dst 50.116.66.139

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

```
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
10:55:01.798591 IP 192.168.0.2.59896 > 50.116.66.139.http: Flags [.], ack 2480401451, win 318, options
[nop,nop,TS val 7955710 ecr 804759402], length 0
10:55:05.527476 IP 192.168.0.2.59894 > 50.116.66.139.http: Flags [F.], seq 2521556029, ack 2164168606,
win 245, options [nop,nop,TS val 7959439 ecr 804759284], length 0
10:55:05.626027 IP 192.168.0.2.59894 > 50.116.66.139.http: Flags [.], ack 2, win 245, o
```

2.NETSTAT

Displays active TCP connections, ports on which the computer is listening, Ethernet statistics, the IP routing table, IPv4 statistics (for the IP, ICMP, TCP, and UDP protocols), and IPv6 statistics (for the IPv6, ICMPv6, TCP over IPv6, and UDP over IPv6 protocols).



```
C:\Users\LxsoftWin>netstat

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    127.0.0.1:49159        LxsoftWin-PC:56051     ESTABLISHED
  TCP    127.0.0.1:49159        LxsoftWin-PC:56297     ESTABLISHED
  TCP    127.0.0.1:49160        LxsoftWin-PC:49259     ESTABLISHED
  TCP    127.0.0.1:49160        LxsoftWin-PC:55384     ESTABLISHED
  TCP    127.0.0.1:49160        LxsoftWin-PC:55392     ESTABLISHED
  TCP    127.0.0.1:49160        LxsoftWin-PC:55394     ESTABLISHED
  TCP    127.0.0.1:49160        LxsoftWin-PC:55395     ESTABLISHED
  TCP    127.0.0.1:49160        LxsoftWin-PC:55401     ESTABLISHED
  TCP    127.0.0.1:49160        LxsoftWin-PC:55406     ESTABLISHED
  TCP    127.0.0.1:49160        LxsoftWin-PC:55407     ESTABLISHED
  TCP    127.0.0.1:49160        LxsoftWin-PC:55408     ESTABLISHED
  TCP    127.0.0.1:49163        LxsoftWin-PC:49164     ESTABLISHED
  TCP    127.0.0.1:49164        LxsoftWin-PC:49163     ESTABLISHED
  TCP    127.0.0.1:49165        LxsoftWin-PC:49166     ESTABLISHED
  TCP    127.0.0.1:49166        LxsoftWin-PC:49165     ESTABLISHED
  TCP    127.0.0.1:49167        LxsoftWin-PC:49168     ESTABLISHED
  TCP    127.0.0.1:49168        LxsoftWin-PC:49167     ESTABLISHED
  TCP    127.0.0.1:49259        LxsoftWin-PC:49160     ESTABLISHED
  TCP    127.0.0.1:51259        LxsoftWin-PC:51260     ESTABLISHED
  TCP    127.0.0.1:51260        LxsoftWin-PC:51259     ESTABLISHED
  TCP    127.0.0.1:55361        LxsoftWin-PC:55362     ESTABLISHED
  TCP    127.0.0.1:55362        LxsoftWin-PC:55361     ESTABLISHED
  TCP    127.0.0.1:55384        LxsoftWin-PC:49160     ESTABLISHED
  TCP    127.0.0.1:55392        LxsoftWin-PC:49160     ESTABLISHED
  TCP    127.0.0.1:55394        LxsoftWin-PC:49160     ESTABLISHED
  TCP    127.0.0.1:55395        LxsoftWin-PC:49160     ESTABLISHED
  TCP    127.0.0.1:55401        LxsoftWin-PC:49160     ESTABLISHED
  TCP    127.0.0.1:55406        LxsoftWin-PC:49160     ESTABLISHED
  TCP    127.0.0.1:55407        LxsoftWin-PC:49160     ESTABLISHED
  TCP    127.0.0.1:55408        LxsoftWin-PC:49160     ESTABLISHED
  TCP    127.0.0.1:56051        LxsoftWin-PC:49159     ESTABLISHED
  TCP    127.0.0.1:56297        LxsoftWin-PC:49159     ESTABLISHED
  TCP    192.168.42.171:55097   server-52-222-136-39:https   CLOSE_WAIT
```

Netstat provides statistics for the following:

- Proto - The name of the protocol (TCP or UDP).

- Local Address - The IP address of the local computer and the port number being used. The name of the local computer that corresponds to the IP address and the name of the port is shown unless the -n parameter is specified. If the port is not yet established, the port number is shown as an asterisk (*).
- Foreign Address - The IP address and port number of the remote computer to which the socket is connected. The names that corresponds to the IP address and the port are shown unless the -n parameter is specified. If the port is not yet established, the port number is shown as an asterisk (*).

(state) Indicates the state of a TCP connection. The possible states are as follows:

- CLOSE_WAIT
- CLOSED
- ESTABLISHED
- FIN_WAIT_1
- FIN_WAIT_2
- LAST_ACK
- LISTEN
- SYN_RECEIVED
- SYN_SEND
- TIMED_WAIT

Syntax

netstat [-a] [-e] [-n] [-o] [-p Protocol] [-r] [-s] [Interval]

Parameters

USED WITHOUT PARAMETERS	**DISPLAYS ACTIVE TCP CONNECTIONS.**
-a	Displays all active TCP connections and the TCP and UDP ports on which the computer is listening.
-e	Displays Ethernet statistics, such as the number of bytes and packets sent and received. This parameter can be combined with -s.
-n	Displays active TCP connections, however, addresses and port numbers are expressed numerically and no attempt is made to determine names.
-o	Displays active TCP connections and includes the process ID (PID) for each connection. You can find the application based on the PID on the Processes tab in Windows Task Manager. This parameter can be combined with -a, -n, and -p.
-p	Shows connections for the protocol specified by Protocol. In this case, the Protocol can be tcp, udp, tcpv6, or udpv6. If this parameter is used with -s to display statistics by

	protocol, Protocol can be tcp, udp, icmp, ip, tcpv6, udpv6, icmpv6, or ipv6.
-s	Displays statistics by protocol. By default, statistics are shown for the TCP, UDP, ICMP, and IP protocols. If the IPv6 protocol for Windows XP is installed, statistics are shown for the TCP over IPv6, UDP over IPv6, ICMPv6, and IPv6 protocols. The -p parameter can be used to specify a set of protocols.
-r	Displays the contents of the IP routing table. This is equivalent to the route print command.
Interval	Redisplays the selected information every Interval seconds. Press CTRL+C to stop the redisplay. If this parameter is omitted, netstat prints the selected information only once.
/?	- Displays help at the command prompt.

3. IPCONFIG

Displays all current TCP/IP network configuration values and refreshes Dynamic Host Configuration Protocol (DHCP) and Domain Name System (DNS) settings. This command is most useful on computers that are configured to obtain an IP address automatically. This enables users to determine which TCP/IP configuration values have been configured by DHCP, Automatic Private IP Addressing (APIPA), or an alternate configuration.

- If the Adapter name contains any spaces, use quotation marks around the adapter name (that is, "Adapter Name").
- For adapter names, ipconfig supports the use of the asterisk (*) wildcard character to specify either adapters with names that begin with a specified string or adapters with names that contain a specified string.
- For example, **Local*** matches all adapters that start with the string Local and ***Con*** matches all adapters that contain the string Con.

Syntax

ipconfig [**/all**] [**/renew** [Adapter]] [**/release** [Adapter]] [**/flushdns**] [**/displaydns**] [**/registerdns**] [**/showclassid** Adapter] [**/setclassid** Adapter [ClassID]]

Parameters

Used without parameters	**displays the IP address, subnet mask, and default gateway for all adapters.**
/all	Displays the full TCP/IP configuration for all adapters. Without this parameter, ipconfig displays only the IP address, subnet mask, and default gateway values for each adapter. Adapters can represent physical interfaces, such as installed network adapters, or logical interfaces, such as dial-up connections.
/renew [Adapter]	Renews DHCP configuration for all adapters (if an adapter is not specified) or for a specific adapter if the Adapter parameter is included. This parameter is available only on computers with adapters that are configured to obtain an IP address automatically. To specify an adapter name, type the adapter name that appears when you use ipconfig without parameters.
/release [Adapter]	Sends a DHCPRELEASE message to the DHCP server to release the current DHCP configuration and discard the IP address configuration for either all adapters (if an adapter is not specified) or for a specific adapter if the Adapter parameter is included. This parameter disables TCP/IP for adapters configured to obtain an IP address automatically. To specify an adapter name, type the adapter name that appears when you use ipconfig without parameters.
/flushdns	Flushes and resets the contents of the DNS client resolver cache. During DNS troubleshooting, you can use this procedure to discard negative cache entries from the cache, as well as any other entries that have been added dynamically.
/displaydns	Displays the contents of the DNS client resolver cache, which includes both entries preloaded from the local Hosts file and any recently obtained resource records for name queries resolved by the computer. The DNS Client service uses this information to resolve frequently queried names quickly, before querying its configured DNS servers.
/registerdns	Initiates manual dynamic registration for the DNS names and IP addresses that are configured at a computer. You can use this parameter to troubleshoot a failed DNS name registration or resolve a dynamic update problem between a client and the DNS server without rebooting the client computer. The DNS settings in the advanced properties of the TCP/IP protocol determine which names are registered in DNS.
/showclassid	Adapter Displays the DHCP class ID for a specified adapter. To see the DHCP class ID for all adapters, use the asterisk (*) wildcard character in place of Adapter. This parameter is available only on computers with adapters that are configured to obtain an IP

	address automatically.
/setclassid	Adapter [ClassID] Configures the DHCP class ID for a specified adapter. To set the DHCP class ID for all adapters, use the asterisk (*) wildcard character in place of Adapter. This parameter is available only on computers with adapters that are configured to obtain an IP address automatically. If a DHCP class ID is not specified, the current class ID is removed.

Examples:

ipconfig	To display the basic TCP/IP configuration for all adapters
ipconfig /all	To display the full TCP/IP configuration for all adapters
ipconfig /renew "Local Area Connection"	To renew a DHCP-assigned IP address configuration for only the Local Area Connection adapter
ipconfig /flushdns	To flush the DNS resolver cache when troubleshooting DNS name resolution problems
ipconfig /showclassid Local	To display the DHCP class ID for all adapters with names that start with Local
ipconfig /setclassid "Local Area Connection" TEST	To set the DHCP class ID for the Local Area Connection adapter to TEST

4. NSLOOKUP

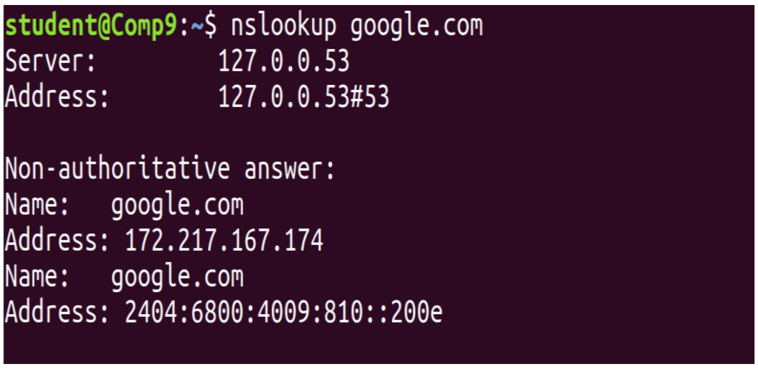
Nslookup (stands for “Name Server Lookup”) is a useful command for getting information from DNS server. It is a network administration tool for querying the Domain Name System (DNS) to obtain domain name or IP address mapping or any other specific DNS record. It is also used to troubleshoot DNS related problems.

Syntax:

nslookup [option]

Options of nslookup command:

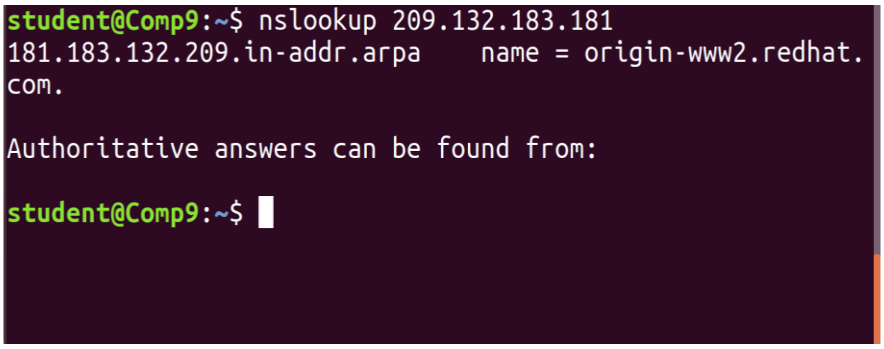
- **nslookup google.com :**
  nslookup followed by the domain name will display the “A Record” (IP Address) of the domain. Use this command to find the address record for a domain. It queries to domain name servers and get the details.



```
student@Comp9:~$ nslookup google.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   google.com
Address: 172.217.167.174
Name:   google.com
Address: 2404:6800:4009:810::200e
```

- **nslookup 192.168.0.10 :** Reverse DNS lookup

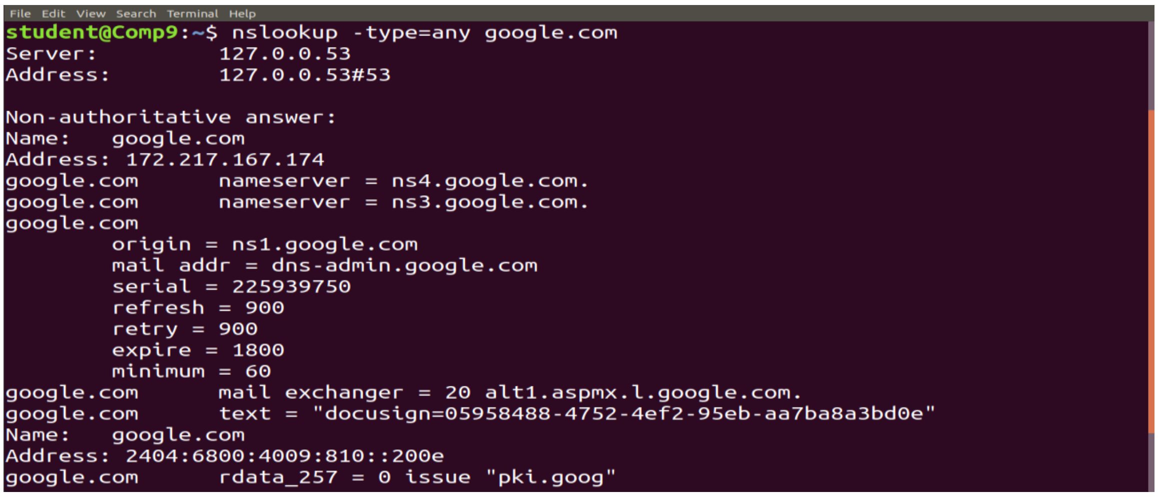
  You can also do the reverse DNS look-up by providing the IP Address as argument to nslookup.



```
student@Comp9:~$ nslookup 209.132.183.181
181.183.132.209.in-addr.arpa    name = origin-www2.redhat.
com.

Authoritative answers can be found from:

student@Comp9:~$ 
```

- **nslookup -type=any google.com :** Lookup for any record

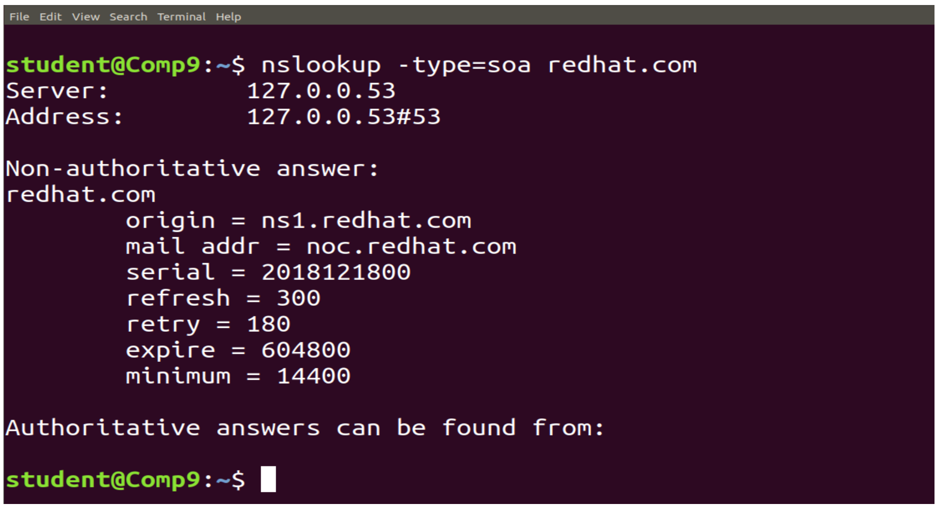
  We can also view all the available DNS records using -type=any option.



```
student@Comp9:~$ nslookup -type=any google.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   google.com
Address: 172.217.167.174
google.com      nameserver = ns4.google.com.
google.com      nameserver = ns3.google.com.
google.com
        origin = ns1.google.com
        mail addr = dns-admin.google.com
        serial = 225939750
        refresh = 900
        retry = 900
        expire = 1800
        minimum = 60
google.com      mail exchanger = 20 alt1.aspmx.l.google.com.
google.com      text = "docusign=05958488-4752-4ef2-95eb-aa7ba8a3bd0e"
Name:   google.com
Address: 2404:6800:4009:810::200e
google.com      rdata_257 = 0 issue "pki.goog"
```

- **nslookup -type=soa redhat.com :** Lookup for an soa record

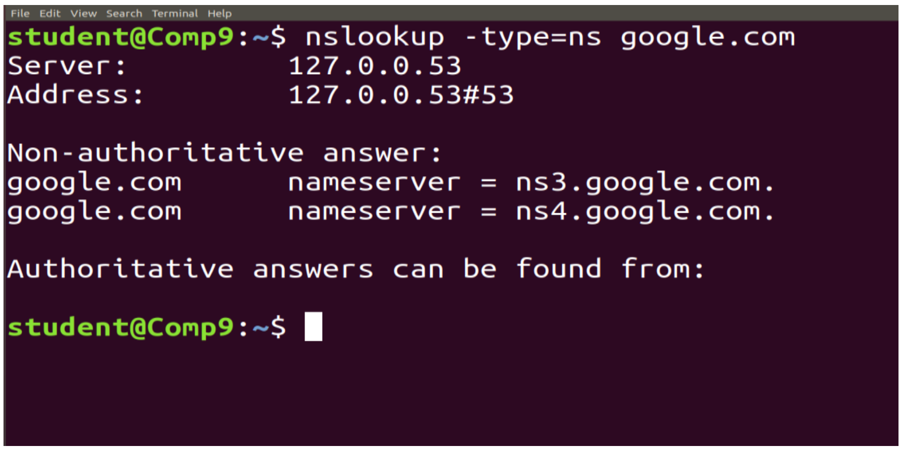
  SOA record (start of authority), provides the authoritative information about the domain, the e-mail address of the domain admin, the domain serial number, etc…



```
student@Comp9:~$ nslookup -type=soa redhat.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
redhat.com
        origin = ns1.redhat.com
        mail addr = noc.redhat.com
        serial = 2018121800
        refresh = 300
        retry = 180
        expire = 604800
        minimum = 14400

Authoritative answers can be found from:

student@Comp9:~$ 
```

- **nslookup -type=ns google.com :** Lookup for an ns record

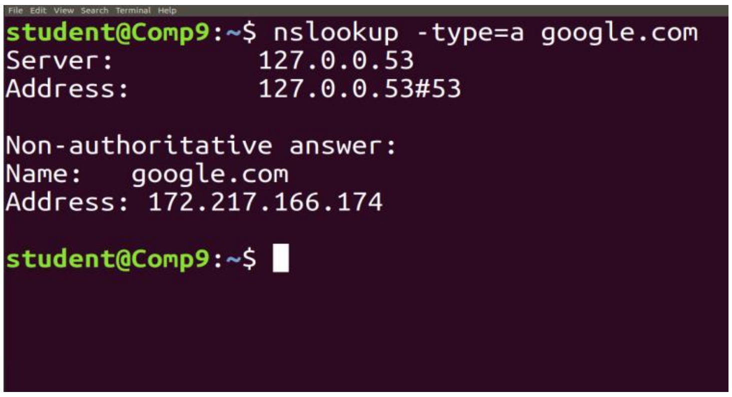
  NS (Name Server) record maps a domain name to a list of DNS servers authoritative for that domain. It will output the name serves which are associated with the given domain.



```
student@Comp9:~$ nslookup -type=ns google.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
google.com      nameserver = ns3.google.com.
google.com      nameserver = ns4.google.com.

Authoritative answers can be found from:

student@Comp9:~$ 
```

- **nslookup -type=a google.com :** Lookup for an a record

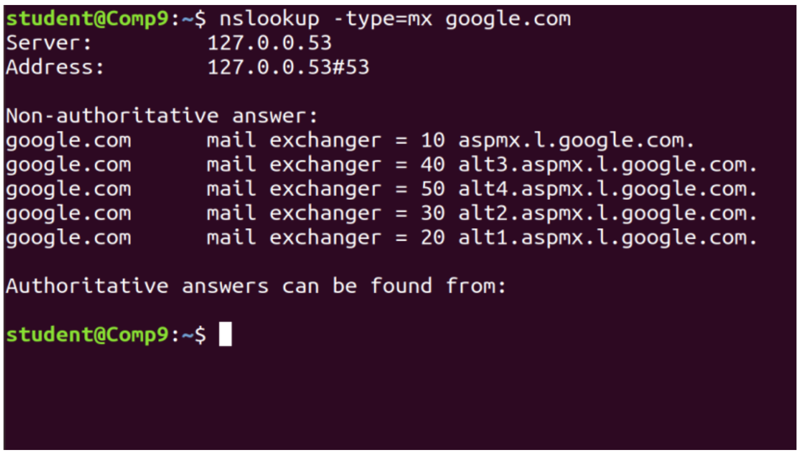
  We can also view all the available DNS records for a particular record using -type=a option.



```
student@Comp9:~$ nslookup -type=a google.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   google.com
Address: 172.217.166.174

student@Comp9:~$ 
```

- **nslookup -type=mx google.com :** Lookup for an mx record

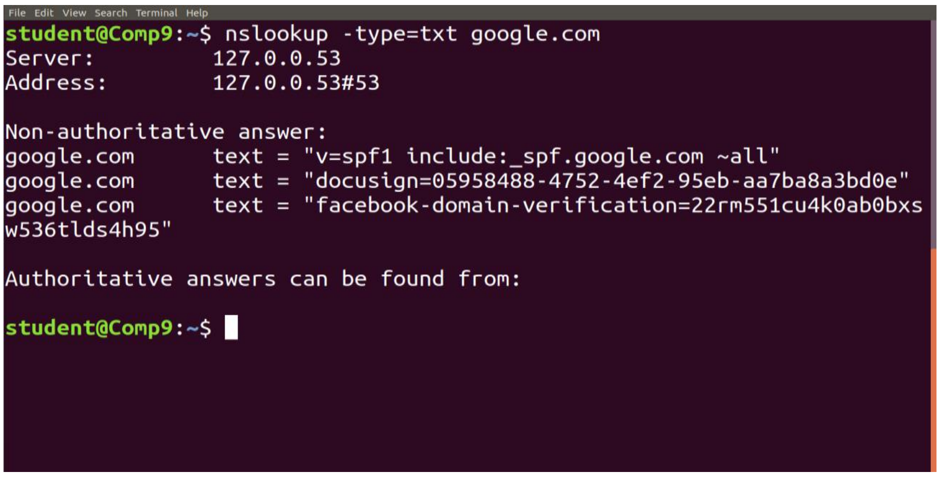
  MX (Mail Exchange) record maps a domain name to a list of mail exchange servers for that domain. The MX record tells that all the mails sent to “google.com” should be routed to the Mail server in that domain.



```
student@Comp9:~$ nslookup -type=mx google.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
google.com      mail exchanger = 10 aspmx.l.google.com.
google.com      mail exchanger = 40 alt3.aspmx.l.google.com.
google.com      mail exchanger = 50 alt4.aspmx.l.google.com.
google.com      mail exchanger = 30 alt2.aspmx.l.google.com.
google.com      mail exchanger = 20 alt1.aspmx.l.google.com.

Authoritative answers can be found from:

student@Comp9:~$
```

- **nslookup -type=txt google.com :** Lookup for an txt record

  TXT records are useful for multiple types of records like DKIM, SPF, etc. You can find all TXT records configured for any domain using below command.



```
student@Comp9:~$ nslookup -type=txt google.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
google.com      text = "v=spf1 include:_spf.google.com ~all"
google.com      text = "docusign=05958488-4752-4ef2-95eb-aa7ba8a3bd0e"
google.com      text = "facebook-domain-verification=22rm551cu4k0ab0bxs
w536tlds4h95"

Authoritative answers can be found from:

student@Comp9:~$
```

5.FTP

FTP (File Transfer Protocol) is a standard network protocol used to exchange files between computers on a private network or through the internet.

There are three ways in which FTP is commonly accessed:

1. Command-line FTP client
2. Web browser
3. Graphical FTP clients

The first two are straightforward methods that allow you to directly use a Web browser (such as Google Chrome, Firefox, or Internet Explorer) or an FTP client application (such as FTP Voyager) to connect to the FTP server to exchange files. Using the command-line interface, you need to enter a set of commands to send or receive files from other computers.

Windows, Mac OS X, and Linux operating systems have built-in command-line clients you can use to establish an FTP connection. To initiate an FTP connection from Windows, type ftp at the command prompt, and press enter.

- **FTP commands for Windows command prompt**

FTP Command	**Description of Command**
!	This command toggles back and forth between the operating system and ftp. Once back in the operating system, typing exit takes you back to the FTP command line.
?	Accesses the Help screen.
append	Append text to a local file.
ascii	Switch to ASCII transfer mode.
bell	Turns bell mode on or off.
binary	Switches to binary transfer mode.
bye	Exits from FTP.
cd	Changes directory.
close	Exits from FTP.
delete	Deletes a file.
debug	Sets debugging on or off.
dir	Lists files, if connected. dir -C = lists the files in wide format. dir -1 = Lists the files in bare format in alphabetic order. dir -r = Lists directory in reverse alphabetic order. dir -R = Lists all files in current directory and sub directories.
disconnect	Exits from FTP.
get	Get file from the remote computer.
glob	Sets globbing on or off. When turned off, the file name in the **put** and **get** commands is taken literally, and wildcards will not be looked at.

FTP Command	Description of Command
hash	Sets hash mark printing on or off. When turned on, for each 1024 bytes of data received, a hash-mark (#) is displayed.
help	Accesses the Help screen and displays information about the command if the command is typed after help.
lcd	Displays local directory if typed alone or if path typed after lcd will change the local directory.
literal	Sends a literal command to the connected computer with an expected one-line response.
ls	Lists files of the remotely connected computer.
mdelete	Multiple delete.
mdir	Lists contents of multiple remote directories.
mget	Get multiple files.
mkdir	Make directory.
mls	Lists contents of multiple remote directories.
mput	Send multiple files.
open	Opens address.
prompt	Enables or disables the prompt.
put	Send one file.
pwd	Print working directory.

FTP Command	Description of Command
quit	Exits from FTP.
quote	Same as the literal command.
recv	Receive file.
remotehelp	Get help from remote server.
rename	Renames a file.
rmdir	Removes a directory on the remote computer.
send	Send single file.
status	Shows status of currently enabled and disabled options.
trace	Toggles packet tracing.
type	Set file transfer type.
user	Send new user information.
verbose	Sets verbose on or off.

- **FTP command-line options for Windows**

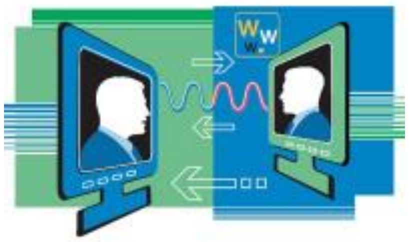
Command-Line Option	Description of Command
-v	Suppresses verbose display of remote server responses.
-n	Suppresses auto-login upon initial connection.
-i	Turns off interactive prompting during multiple file transfers.
-d	Enables debugging, displaying all ftp commands passed between the client and server.
-g	Disables filename globing, which permits the use of wildcard characters in local file and path names.
-s:filename	Specifies a text file containing ftp commands; the commands automatically run after ftp starts. No spaces are allowed in this parameter. Use this switch instead of redirection (>).

Command-Line Option	**Description of Command**
-a	Use any local interface when binding data connection.
-Windowsize	Overrides the default transfer buffer size of 4096.
computer	Specifies the computer name or IP address of the remote computer to connect to. The computer, if specified, must be the last parameter on the line.

6.TELNET

Telnet helps to -

- connect to a remote Linux computer
- run programs remotely and conduct administration



This utility is similar to the Remote Desktop feature found in Windows Machine.

The syntax for this utility is:

telnet hostname="" or=""

Example:

telnet localhost

telnet - user interface to the TELNET protocol

SYNOPSIS

telnet [-**8EFKLacdfrx**] [-**X** authtype] [-**b** hostalias] [-**e** escapechar] [-**k** realm] [-**l** user] [-**n** tracefile]
[host [port]]

DESCRIPTION

The **telnet** command is used to communicate with another host using the TELNET protocol. If **telnet** is invoked without the host argument, it enters command mode, indicated by its prompt (**telnet>**). In this mode, it accepts and executes the commands listed below. If it is invoked with arguments, it performs an **open** command with those arguments.

The options are as follows:

Tag	Description
-7	Strip 8th bit on input and output. Telnet is 8-bit clean by default but doesn't send the TELNET BINARY option unless forced.
-8	Specifies an 8-bit data path. This causes an attempt to negotiate the TELNET BINARY option on both input and output.
-E	Stops any character from being recognized as an escape character.
-F	If Kerberos V5 authentication is being used, the **-F** option allows the local credentials to be forwarded to the remote system, including any credentials that have already been forwarded into the local environment.
-K	Specifies no automatic login to the remote system.
-L	Specifies an 8-bit data path on output. This causes the BINARY option to be negotiated on output.
-X atype	
	Disables the atype type of authentication.
-a	Attempt automatic login. Currently, this sends the user name via the USER variable of the ENVIRON option if supported by the remote system. The name used is that of the current user as returned by **getlogin**(2) if it agrees with the current user ID, otherwise it is the name associated with the user ID.
-b hostalias	
	Uses **bind**(2) on the local socket to bind it to an aliased address (see **ifconfig**(8) and the ''alias'' specifier) or to the address of another interface than the one naturally chosen by **connect**(2). This can be useful when connecting to services which use IP addresses for authentication and reconfiguration of the server is undesirable (or impossible).
-c	Disables the reading of the user's .telnetrc file. (See the **toggle skiprc** command on this man page.)

-d	Sets the initial value of the **debug** toggle to TRUE.
-e escapechar	
	Sets the initial **telnet** escape character to escapechar. If escapechar is omitted, then there will be no escape character.
-f	If Kerberos V5 authentication is being used, the **-f** option allows the local credentials to be forwarded to the remote system.
-k realm	
	If Kerberos authentication is being used, the **-k** option requests that **telnet** obtain tickets for the remote host in realm realm instead of the remote host's realm, as determined by **krb_realmofhost**(3).
-l user	
	When connecting to the remote system, if the remote system understands the ENVIRON option, then user will be sent to the remote system as the value for the variable USER. This option implies the **-a** option. This option may also be used with the **open** command.
-n tracefile	
	Opens tracefile for recording trace information. See the **set tracefile** command below.
-r	Specifies a user interface similar to **rlogin**(1). In this mode, the escape character is set to the tilde (~) character, unless modified by the **-e** option.
-x	Turns on encryption of the data stream if possible.
host	Indicates the official name, an alias, or the Internet address of a remote host.
port	Indicates a port number (address of an application). If a number is not specified, the default **telnet** port is used.

7. TRACEROUTE.

Tracert: Determines the path taken to a destination by sending Internet Control Message Protocol (ICMP) Echo Request messages to the destination with incrementally increasing Time to Live (TTL) field values. The path displayed is the list of near-side router interfaces of the routers in the path between a source host and a destination. The near-side interface is the interface of the router that is closest to the sending host in the path. Used without parameters, tracert displays help.

This diagnostic tool determines the path taken to a destination by sending ICMP Echo Request messages with varying Time to Live (TTL) values to the destination. Each router along the path is required to decrement the TTL in an IP packet by at least 1 before forwarding it.Effectively, the TTL is a maximum link counter. When the TTL on a packet reaches 0, the router is expected to return an ICMP Time Exceeded message to the source computer. Tracert determines the path by sending the first Echo Request message with a TTL of 1 and incrementing the TTL by 1 on each subsequent transmission until the target responds or the maximum number of hops is reached. The maximum number of hops is 30 by default and can be specified using the -h parameter.

The path is determined by examining the ICMP Time Exceeded messages returned by intermediate routers and the Echo Reply message returned by the destination. However, some routers do not return Time Exceeded messages for packets with expired TTL values and are invisible to the tracert command. In this case, a row of asterisks (*) is displayed for that hop.

Examples:

To trace the path to the host named www.google.co.in use following command

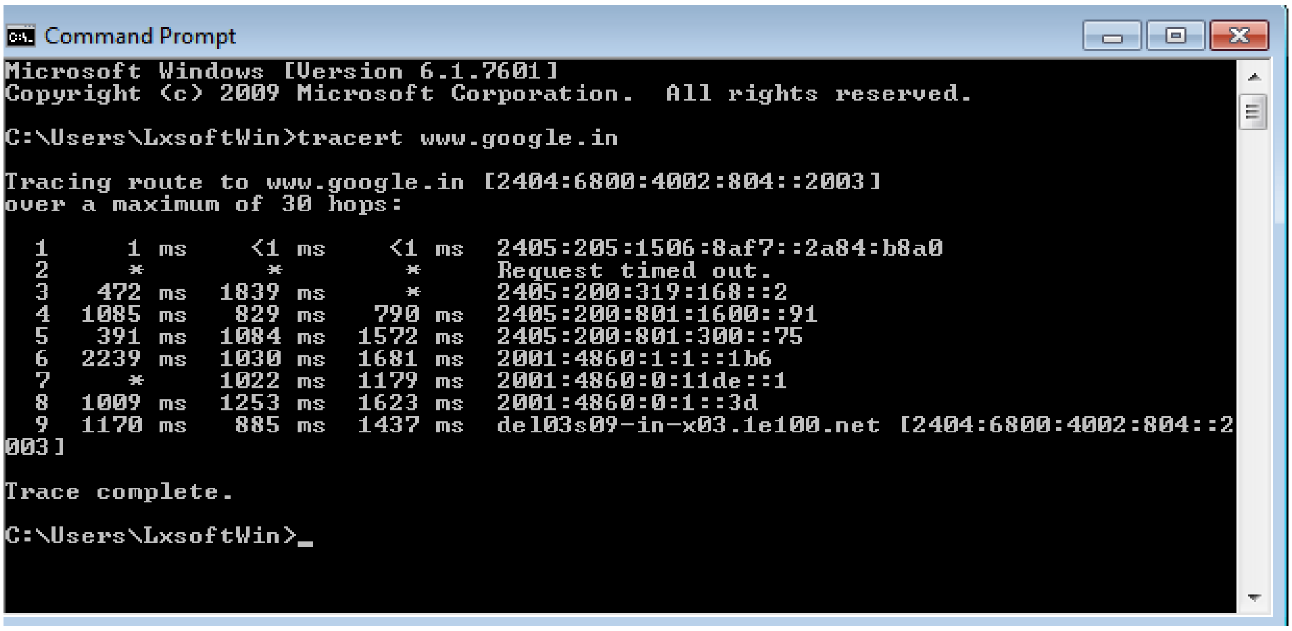
```
tracert www.google.co.in
```

To trace the path to the host named www.google.com and prevent the resolution of each IP address to its name, type:

tracert -d www.google.com

To trace the path to the host named www.google.com and use the loose source route 10.12.0.1-10.29.3.1-10.1.44.1, type:

tracert -j 10.12.0.1 10.29.3.1 10.1.44.1 www.google.com



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

C:\Users\LxsoftWin>tracert www.google.in

Tracing route to www.google.in [2404:6800:4002:804::2003]
over a maximum of 30 hops:

  1     1 ms    <1 ms    <1 ms  2405:205:1506:8af7::2a84:b8a0
  2     *        *        *     Request timed out.
  3   472 ms  1839 ms     *     2405:200:319:168::2
  4  1085 ms   829 ms   790 ms  2405:200:801:1600::91
  5   391 ms  1084 ms  1572 ms  2405:200:801:300::75
  6  2239 ms  1030 ms  1681 ms  2001:4860:1:1::1b6
  7     *     1022 ms  1179 ms  2001:4860:0:11de::1
  8  1009 ms  1253 ms  1623 ms  2001:4860:0:1::3d
  9  1170 ms   885 ms  1437 ms  del03s09-in-x03.1e100.net [2404:6800:4002:804::2
003]

Trace complete.

C:\Users\LxsoftWin>_
```

Syntax

tracert [-d] [-h MaximumHops] [-j HostList] [-w Timeout] [TargetName]

Parameters

-d	Prevents tracert from attempting to resolve the IP addresses of intermediate routers to their names. This can speed up the display of tracert results.
-h	MaximumHops Specifies the maximum number of hops in the path to search for the target (destination). The default is 30 hops.
-j	HostList Specifies that Echo Request messages use the Loose Source Route option in the IP header with the set of intermediate destinations specified in HostList. With loose source routing, successive intermediate destinations can be separated by one or multiple routers. The maximum number of addresses or names in the host list is 9. The HostList
-w	Timeout Specifies the amount of time in milliseconds to wait for the ICMP Time Exceeded or Echo Reply message corresponding to a given Echo Request message to be received. If not received within the time-out, an asterisk (*) is displayed. The default time-out is 4000 (4 seconds).

RESULT: Thus study of network commands like tcpdump, netstat, ipconfig, nslookup, FTP, TELNET and traceroute is done successfully.

EXPERIMENT NO.2

Encryption and Decryption

AIM : Write a C program to perform encryption and decryption for the given string

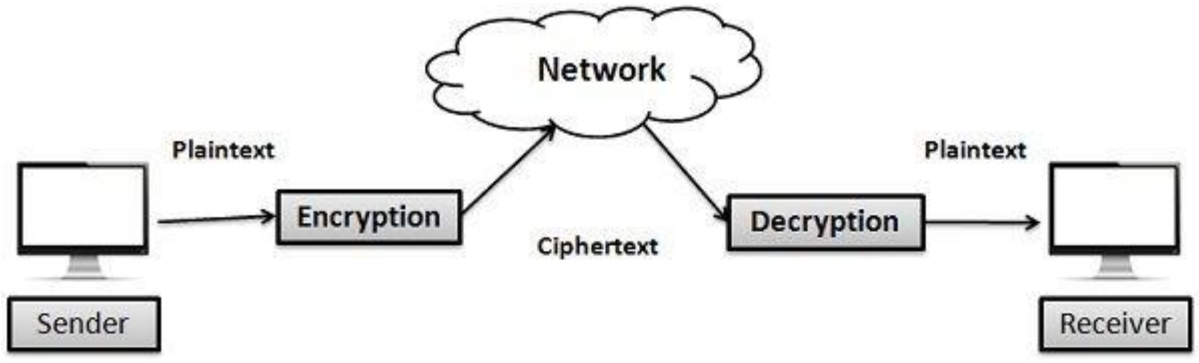
THEORY:

Encryption is the method by which information is converted into secret code that hides the information's true meaning. The science of encrypting and decrypting information is called cryptography.Unencrypted data is also known as plaintext, and encrypted data is called ciphertext. The formulas used to encode and decode messages are called encryption algorithms, or ciphers.

Encryption plays an important role in securing many different types of information technology (IT) assets. It provides the following:

- Confidentiality encodes the message's content.
- Authentication verifies the origin of a message.
- Integrity proves the contents of a message have not been changed since it was sent.
- Nonrepudiation prevents senders from denying they sent the encrypted message.

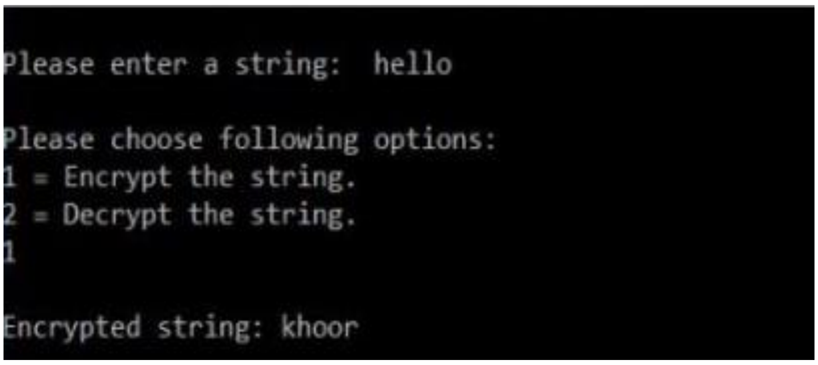
Decryption is the process of transforming data that has been rendered unreadable through encryption back to its unencrypted form. In decryption, the system extracts and converts the garbled data and transforms it to texts and images that are easily understandable not only by the reader but also by the system



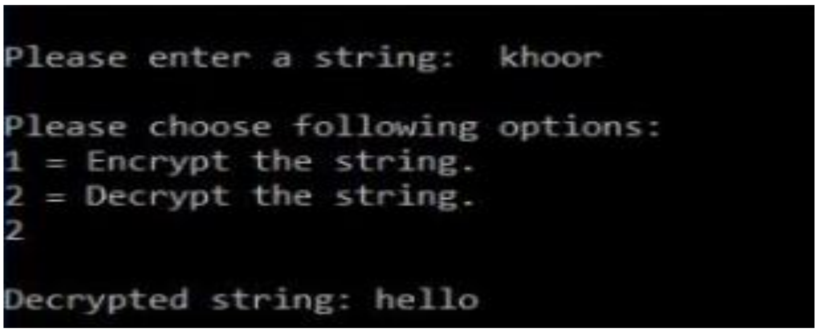
PROGRAM:

```
#include <stdio.h>

int main()
{
  int i, x;
  char str[100];

printf("\nPlease enter a string:\t");
  gets(str);
printf("\nPlease choose following options:\n");
printf("1 = Encrypt the string.\n");
printf("2 = Decrypt the string.\n");
scanf("%d", &x);

  //using switch case statements
  switch(x)
  {
  case 1:
for(i = 0; (i < 100 && str[i] != '\0'); i++)
      str[i] = str[i] + 3; //the key for encryption is 3 that is added to ASCII value

printf("\nEncrypted string: %s\n", str);
    break;

  case 2:
for(i = 0; (i < 100 && str[i] != '\0'); i++)
      str[i] = str[i] - 3; //the key for encryption is 3 that is subtracted to ASCII value

printf("\nDecrypted string: %s\n", str);
    break;

  default:
    printf("\nError\n");
  }
  return 0;
}
```

OUTPUT:



```
Please enter a string:  hello

Please choose following options:
1 = Encrypt the string.
2 = Decrypt the string.
1

Encrypted string: khoor
```



```
Please enter a string:  khoor

Please choose following options:
1 = Encrypt the string.
2 = Decrypt the string.
2

Decrypted string: hello
```

RESULT : Thus the program to perform encryption and decryption for the given string is successfully done.

VIVA QUESTIONS

1 What is Network?
A network is a set of devices connected by physical media links. A network is recursively is a connection of two or more nodes by a physical link or two or more networks connected by one or more nodes.

2. What is a Link?
At the lowest level, a network can consist of two or more computers directly connected by some physical medium such as coaxial cable or optical fiber. Such a physical medium is called as Link.

3. What is a node?
A network can consist of two or more computers directly connected by some physical medium such as coaxial cable or optical fiber. Such a physical medium is called as Links and the computer it connects is called as Nodes.

4. What is a gateway or Router?
A node that is connected to two or more networks is commonly called as router or Gateway. It generally forwards message from one network to another.

5. What is point-point link?
If the physical links are limited to a pair of nodes it is said to be point-point link.

6. What is Multiple Access?
If the physical links are shared by more than two nodes, it is said to be Multiple Access.

7. What are the advantages of Distributed Processing?
a. Security/Encapsulation
b. Distributed database
c. Faster Problem solving
d. Security through redundancy
e. Collaborative Processing

8. What are the criteria necessary for an effective and efficient network?
a. Performance
It can be measured in many ways, including transmit time and response time. b. Reliability
It is measured by frequency of failure, the time it takes a link to recover from a failure, and the networks robustness.
c. Security
Security issues includes protecting data from unauthorized access and virues.

9. Name the factors that affect the performance of the network?
a. Number of Users
b. Type of transmission medium
c. Hardware
d. Software

10. Name the factors that affect the reliability of the network?
a. Frequency of failure
b. Recovery time of a network after a failure

11. Name the factors that affect the security of the network?
a. Unauthorized Access

b. Viruses

12. What is Protocol?
A protocol is a set of rules that govern all aspects of information communication.

13. What are the key elements of protocols?
The key elements of protocols are
a. Syntax
It refers to the structure or format of the data, that is the order in which they are presented.
b. Semantics
It refers to the meaning of each section of bits.
c. Timing
Timing refers to two characteristics: When data should be sent and how fast they can be sent.

14. What are the key design issues of a computer Network?
a. Connectivity
b. Cost-effective Resource Sharing
c. Support for common Services
d. Performance

15. Define Bandwidth and Latency?
Network performance is measured in Bandwidth (throughput) and Latency (Delay). Bandwidth of a network is given by the number of bits that can be transmitted over the network in a certain period of time. Latency corresponds to how long it t5akes a message to travel from one end off a network to the other. It is strictly measured in terms of time.

16. Define Routing?
The process of determining systematically hoe to forward messages toward the destination nodes based on its address is called routing.

17. What is a peer-peer process?
The processes on each machine that communicate at a given layer are called peer-peer process.

18. When a switch is said to be congested?
It is possible that a switch receives packets faster than the shared link can accommodate and stores in its memory, for an extended period of time, then the switch will eventually run out of buffer space, and some packets will have to be dropped and in this state is said to congested state.

19. What is semantic gap?
Defining a useful channel involves both understanding the applications requirements and recognizing the limitations of the underlying technology. The gap between what applications expects and what the underlying technology can provide is called semantic gap.

20. What is Round Trip Time?
The duration of time it takes to send a message from one end of a network to the other and back, is called RTT.

21. Define the terms Unicasting, Multiccasting and Broadcasting?
If the message is sent from a source to a single destination node, it is called Unicasting.
If the message is sent to some subset of other nodes, it is called Multicasting.
If the message is sent to all the m nodes in the network it is called Broadcasting.

22. What is Multiplexing?
Multiplexing is the set of techniques that allows the simultaneous transmission of multiple signals across a single data link.

23. Name the categories of Multiplexing?
a. Frequency Division Multiplexing (FDM)
b. Time Division Multiplexing (TDM)
i. Synchronous TDM
ii. ASynchronous TDM Or Statistical TDM.
c. Wave Division Multiplexing (WDM)

24. What is FDM?
FDM is an analog technique that can be applied when the bandwidth of a link is greater than the combined bandwidths of the signals to be transmitted.

25. What is WDM?
WDM is conceptually the same as FDM, except that the multiplexing and demultiplexing involve light signals transmitted through fiber optics channel.

26. What is TDM?
TDM is a digital process that can be applied when the data rate capacity of the transmission medium is greater than the data rate required by the sending and receiving devices.

27. What is Synchronous TDM?
In STDM, the multiplexer allocates exactly the same time slot to each device at all times, whether or not a device has anything to transmit.

28. List the layers of OSI
a. Physical Layer
b. Data Link Layer
c. Network Layer
d. Transport Layer
e. Session Layer
f. Presentation Layer
g. Application Layer

29. Which layers are network support layers?
a. Physical Layer
b. Data link Layer and
c. Network Layers

30. Which layers are user support layers?
a. Session Layer
b. Presentation Layer and
c. Application Layer

31. Which layer links the network support layers and user support layers?
The Transport layer links the network support layers and user support layers.

32. What are the concerns of the Physical Layer?

Physical layer coordinates the functions required to transmit a bit stream over a physical medium.
a. Physical characteristics of interfaces and media
b. Representation of bits
c. Data rate
d. Synchronization of bits
e. Line configuration
f. Physical topology
g. Transmission mode

33. What are the responsibilities of Data Link Layer?
The Data Link Layer transforms the physical layer, a raw transmission facility, to a reliable link and is responsible for node-node delivery.
a. Framing
b. Physical Addressing
c. Flow Control
d. Error Control
e. Access Control

34. What are the responsibilities of Network Layer?
The Network Layer is responsible for the source-to-destination delivery of packet possibly across multiple networks (links).
a. Logical Addressing
b. Routing

35. What are the responsibilities of Transport Layer?
The Transport Layer is responsible for source-to-destination delivery of the entire message.
a. Service-point Addressing
b. Segmentation and reassembly
c. Connection Control
d. Flow Control
e. Error Control

36. What are the responsibilities of Session Layer?
The Session layer is the network dialog Controller. It establishes, maintains and synchronizes the interaction between the communicating systems.
a. Dialog control
b. Synchronization

37. What are the responsibilities of Presentation Layer?
The Presentation layer is concerned with the syntax and semantics of the information exchanged between two systems.
a. Translation
b. Encryption
c. Compression

38. What are the responsibilities of Application Layer?
The Application Layer enables the user, whether human or software, to access the network. It provides user interfaces and support for services such as e-mail, shared database management and other types of distributed information services.
a. Network virtual Terminal
b. File transfer, access and Management (FTAM)

c. Mail services
d. Directory Services

39. What are the two classes of hardware building blocks?
Nodes and Links.

40. What are the different link types used to build a computer network?
a. Cables
b. Leased Lines
c. Last-Mile Links
d. Wireless Links

41. What are the categories of Transmission media?
a. Guided Media
i. Twisted Pair cable
1. Shielded TP
2. Unshielded TP
ii. Coaxial Cable
iii. Fiber-optic cable
b. Unguided Media
i. Terrestrial microwave
ii. Satellite Communication

42. What are the types of errors?
a. Single-Bit error
In a single-bit error, only one bit in the data unit has changed
b. Burst Error
A Burst error means that two or more bits in the data have changed.

43. What is Error Detection? What are its methods?
Data can be corrupted during transmission. For reliable communication errors must be deducted and Corrected. Error Detection uses the concept of redundancy, which means adding extra bits for detecting errors at the destination. The common Error Detection methods are
a. Vertical Redundancy Check (VRC)
b. Longitudinal Redundancy Check (VRC)
c. Cyclic Redundancy Check (VRC)
d. Checksum

44. What is Redundancy?
The concept of including extra information in the transmission solely for the purpose of comparison. This technique is called redundancy.

45. What is VRC?
It is the most common and least expensive mechanism for Error Detection. In VRC, a parity bit is added to every data unit so that the total number of 1s becomes even for even parity. It can detect all single-bit errors. It can detect burst errors only if the total number of errors in each data unit is odd.

46. What is LRC?
In LRC, a block of bits is divided into rows and a redundant row of bits is added to the whole block. It can detect burst errors. If two bits in one data unit are damaged and bits in exactly the same positions in another

data unit are also damaged, the LRC checker will not detect an error. In LRC a redundant data unit follows n data units.

47. What is CRC?
CRC, is the most powerful of the redundancy checking techniques, is based on binary division.

48. What is Checksum?
Checksum is used by the higher layer protocols (TCP/IP) for error detection

49. List the steps involved in creating the checksum.
a. Divide the data into sections
b. Add the sections together using 1s complement arithmetic
c. Take the complement of the final sum, this is the checksum.

50. What are the Data link protocols?
Data link protocols are sets of specifications used to implement the data link layer. The categories of Data Link protocols are

Asynchronous Protocols
Synchronous Protocols
a. Character Oriented Protocols
b. Bit Oriented protocols
51. Compare Error Detection and Error Correction:
The correction of errors is more difficult than the detection. In error detection, checks only any error has occurred. In error correction, the exact number of bits that are corrupted and location in the message are known. The number of the errors and the size of the message are important factors.

52. What is Forward Error Correction?
Forward error correction is the process in which the receiver tries to guess the message by using redundant bits.

53. Define Retransmission?
Re transmission is a technique in which the receiver detects the occurrence of an error and asks the sender to resend the message. Re sending is repeated until a message arrives that the receiver believes is error-freed.

54. What are Data Words?
In block coding, we divide our message into blocks, each of k bits, called datawords. The block coding process is one-to-one. The same dataword is always encoded as the same codeword.

55. What are Code Words?
r redundant bits are added to each block to make the length n = k + r. The resulting n-bit blocks are called codewords. 2n 2k codewords that are not used. These codewords are invalid or illegal.

56. What is a Linear Block Code?
A linear block code is a code in which the exclusive OR (addition modulo-2) of two valid codewords creates another valid codeword.

57. What are Cyclic Codes?
Cyclic codes are special linear block codes with one extra property. In a cyclic code, if a codeword is cyclically shifted (rotated), the result is another codeword.

58. Define Encoder?
A device or program that uses predefined algorithms to encode, or compress audio or video data for storage or transmission use. A circuit that is used to convert between digital video and analog video.

59. Define Decoder?
A device or program that translates encoded data into its original format (e.g. it decodes the data). The term is often used in reference to MPEG-2 video and sound data, which must be decoded before it is output.

60. What is Framing?
Framing in the data link layer separates a message from one source to a destination, or from other messages to other destinations, by adding a sender address and a destination address. The destination address defines where the packet has to go and the sender address helps the recipient acknowledge the receipt.
61. What is Fixed Size Framing?
In fixed-size framing, there is no need for defining the boundaries of the frames. The size itself can be used as a delimiter.

62. Define Character Stuffing?
In byte stuffing (or character stuffing), a special byte is added to the data section of the frame when there is a character with the same pattern as the flag. The data section is stuffed with an extra byte. This byte is usually called the escape character (ESC), which has a predefined bit pattern. Whenever the receiver encounters the ESC character, it removes it from the data section and treats the next character as data, not a delimiting flag.

63. What is Bit Stuffing?
Bit stuffing is the process of adding one extra 0 whenever five consecutive Is follow a 0 in the data, so that the receiver does not mistake the pattern 0111110 for a flag.

64. What is Flow Control?
Flow control refers to a set of procedures used to restrict the amount of data that the sender can send before waiting for acknowledgment.

65. What is Error Control ?
Error control is both error detection and error correction. It allows the receiver to inform the sender of any frames lost or damaged in transmission and coordinates the retransmission of those frames by the sender. In the data link layer, the term error control refers primarily to methods of error detection and re transmission.

66. What Automatic Repeat Request (ARQ)?
Error control is both error detection and error correction. It allows the receiver to inform the sender of any frames lost or damaged in transmission and coordinates the retransmission of those frames by the sender. In the data link layer, the term error control refers primarily to methods of error detection and retransmission. Error control in the data link layer is often implemented simply: Any time an error is detected in an exchange, specified frames are retransmitted. This process is called automatic repeat request (ARQ).

67. What is Stop-and-Wait Protocol?
In Stop and wait protocol, sender sends one frame, waits until it receives confirmation from the receiver (okay to go ahead), and then sends the next frame.

68. What is Stop-and-Wait Automatic Repeat Request?
Error correction in Stop-and-Wait ARQ is done by keeping a copy of the sent frame and retransmitting of the frame when the timer expires.

69. What is usage of Sequence Number in Relaible Transmission?

The protocol specifies that frames need to be numbered. This is done by using sequence numbers. A field is added to the data frame to hold the sequence number of that frame. Since we want to minimize the frame size, the smallest range that provides unambiguous communication. The sequence numbers can wrap around.

70. What is Pipelining ?
In networking and in other areas, a task is often begun before the previous task has ended. This is known as pipelining.

71. What is Sliding Window?
The sliding window is an abstract concept that defines the range of sequence numbers that is the concern of the sender and receiver. In other words, he sender and receiver need to deal with only part of the possible sequence numbers.

72. What is Piggy Backing?
A technique called piggybacking is used to improve the efficiency of the bidirectional protocols. When a frame is carrying data from A to B, it can also carry control information about arrived (or lost) frames from B; when a frame is carrying data from B to A, it can also carry control information about the arrived (or lost) frames from A.

73. What are the two types of transmission technology available?
(i) Broadcast and (ii) point-to-point

74. What is subnet?
A generic term for section of a large networks usually separated by a bridge or router.

75. Difference between the communication and transmission.
Transmission is a physical movement of information and concern issues like bit polarity, synchronisation, clock etc.
Communication means the meaning full exchange of information between two communication media.

76. What are the possible ways of data exchange?
(i) Simplex (ii) Half-duplex (iii) Full-duplex.

77. What is SAP?
Series of interface points that allow other computers to communicate with the other layers of network protocol stack.

BASICS OF NS2

Computer simulation is the designing of a theoretical physical system on a digital computer with emphasis on model designing, execution and analysis. After creation of the mathematical model the most important step is to create a computer program for updating the state and event variables through time (by time slicing or event scheduling). If this simulation is carried out successively in parallel computers, it is called *Parallel* or *Distributed simulation*.

Network simulation (NS) is one of the types of simulation, which is used to simulate the networks such as in MANETs, VANETs etc. It provides simulation for routing and multicast protocols for both wired and wireless networks. NS is licensed for use under version 2 of the GNU (General Public License) and is popularly known as **NS2**. It is an object-oriented, discrete event-driven simulator written in C++ and

Otcl/tcl.

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

Install NS-2 using this command :
sudo apt-get install ns2

Nam is also needed to install. Nam (**Network Animator**) is an animation tool to graphically represent the network and packet traces. Use this command :
sudo apt-get install nam

Some basic Otcl script syntax :

- **Basic Command :**

```
set a 8
set b [expr $a/8]
```

- **Explanation :** In the first line, the variable **a** is assigned the value 8. In the second line, the result of the command [expr $a/8], which equals 1, is then used as an argument to another command, which in turn assigns a value to the variable **b**. The "$" sign is used to obtain a value contained in a variable and square brackets are an indication of a command substitution.

- **Define new procedures with *proc* command**

```
proc factorial fact {
   if {$fact <= 1} {
      return 1
   }
expr $fact * [factorial [expr $fact-1]]
}
```

- **To open a file for reading :**

```
set testfile [open hello.dat r]
```

- Similarly, **put** command is used to write data into the file

```
set testfile [open hello.dat w]
puts $testfile "hello1"
```

- To call subprocesses within another process, **exec** is used, which creates a subprocess and waits for it to complete.

```
exec rm $testfile
```

- To be able to run a simulation scenario, a network topology must first be created. In ns2, the topology consists of a collection of nodes and links.

```
set ns [new Simulator]
```

- The simulator object has member functions which enables to create the nodes and define the links between them. The class simulator contains all the basic functions. Since ns was defined to handle the Simulator object, the command $ns is used for using the functions belonging to the simulator class. In the network topology nodes can be added in the following manner :

```
set n0 [$ns node]
set n1 [$ns node]
```

- Traffic agents (TCP, UDP etc.) and traffic sources (FTP, CBR etc.) must be set up if the node is not a router. It enables to create CBR traffic source using UDP as transport protocol or an FTP traffic source using TCP as a transport protocol.
  CBR traffic source using UDP :

```
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr0 set packet_size_ 512
```

FTP traffic source using TCP :

```
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$tcp0 set packet_size_ 512
```

1. What are the four files on the NS2 simulator?
   The four files include the .tcl file, the .awk file, the .tr file and the nam file.

2. What is the use of a tr file?
   The tr file is mainly used to analyse our data.It is given as input to nam file.

3. What does $1,$2,..... indicate in the awk file?
   The $1, $2,.... indicate the columns that represent different events in the ex3.tr file.

4. How do we increase the throughput?
   The throughput is increased by decreasing the bandwidth.