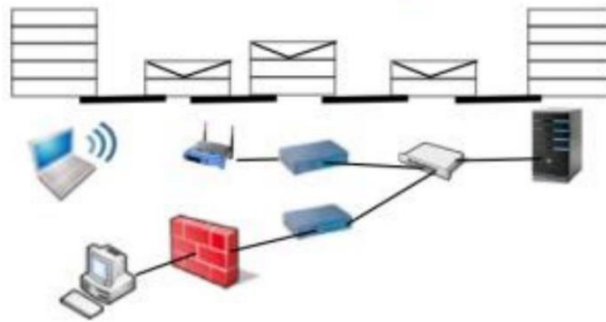


## Experiment No. 1

### Study of Networking Commands

#### Introduction to Networking commands -

The operating system consists of various built-in, command-line networking utilities that are used for network troubleshooting. We will see various networking commands which are most essentials for every network administrator.



## 1. Ping

Ping is used to testing a network host capacity to interact with another host. Just enter the command Ping, followed by the target host's name or IP address. The ping utilities seem to be the most common network tool. This is performed by using the Internet Control Message Protocol, which allows the echo packet to be sent to the destination host and a listening mechanism. If the destination host reply to the requesting host, that means the host is reachable. This utility usually gives a basic image of where there may be a specific networking issue,

**For Example:** If an Internet connection is not in the office, for instance, the ping utility is used to determine if the problem exists in the office or the Internet provider's network. The following shows an image of how ping tools to obtain the locally connected router's connectivity status.

## 2. NetStat

Netstat is a Common TCP – IP networking command-line method present in most Windows, Linux, UNIX, and other operating systems. The netstat provides the statistics and information in the use of the current TCP-IP Connection network about the protocol.

There are various options a user can use with the Netstat command.

Options are as follows-

- **-a:** This will display all connection and ports
- **-b:** Shows the executable involved in each connection or hearing port
- **-e:** This protocol will combine with the -s and display the ethernet statistics

- **-n:** This will display the address and the port number in the form of numerical
- **-o:** It will display the ID of each connection for the ownership process.
- **-r:** It will display the routing table
- **-v:** When used in combination with **-b**, the link or hearing port sequence for every executable is shown.

### 3. Ip Config

The command IP config will display basic details about the device's IP address configuration. Just type IP config in the Windows prompt and the IP, subnet mask and default gateway that the current device will be presented. If you have to see full information, then type on command prompt config-all and then you will see full information. There are also choices to assist you in resolving DNS and DHCP issues.

### 4. Hostname

To communicate with each and other, the computer needs a unique address. A hostname can be alphabetic or alphanumeric and contain specific symbols used specifically to define a specific node or device in the network. For example, a hostname should have a domain name (TLD) of the top-level and a distance between one and 63 characters when used in a domain name system (DNS) or on the Internet.

### 5. Tracert

The tracert command is a Command Prompt command which is used to get the network packet being sent and received and the number of hops required for that packet to reach to target. This command can also be referred to as a traceroute. It provides several details about the path that a packet takes from the source to the specified destination.

The tracert command is available for the Command Prompt in all Windows operating systems.

The syntax for Tracert Command

```
tracert [-d] [-h MaxHops] [-w TimeOut] target
```

There are various options the user can use with tracert command.

Options for tracert Command are as follows-

- **target:** This is the destination, either an IP address or hostname.
- **-d:** This option prevents Tracert from resolving IP addresses to hostnames to get faster results.
- **-h MaxHops:** This Tracert option specifies the maximum number of hops in the search for the target. If the MaxHops option is not specified the target has not been found by 30 hops, then the tracert command will stop looking.
- **-w timeout:** A timeout value must be specified while executing this ping command. It adjusts the amount of time in milliseconds.

## 6. Nslookup

The Nslookup, which stands for name server lookup command, is a network utility command used to obtain information about internet servers. It provides name server information for the DNS (Domain Name System), i.e. the default DNS server's name and IP Address.

The syntax for Nslookup is as follows.

```
Nslookup
```

```
or
```

```
Nslookup [domain name]
```

## 7. Route

In IP networks, routing tables are used to direct packets from one subnet to another. The Route command provides the device's routing tables. To get this result, just type route print. The Route command returns the routing table, and the user can make changes by Commands such as Route Add, Route Delete, and Route Change, which allows modifying the routing table as a requirement.

## 8. ARP

ARP stands for Address Resolution Protocol. Although network communications can readily be thought of as an IP address, the packet delivery depends ultimately on the media access control (MAC). This is where the protocol for address resolution comes into effect. You can add the remote host IP address, which is an arp -a command, in case you have issues to communicate with a given host. The ARP command provides information like Address, Flags, Mask, IFace, Hardware Type, Hardware Address, etc.

## 9. Path Ping

We discussed the Ping command and the Tracert command. There are similarities between these commands. The pathping command which provides a combination of the best aspects of Tracert and Ping.

This command takes 300 seconds to gather statistics and then returns reports on latency and packet loss statistics at intermediate hops between the source and the target in more detail than those reports provided by Ping or Tracert commands.

The syntax for path ping is as follows:

```
path ping [-n] [-h] [-g <Hostlist>] [-p <Period>] [-q <NumQueries> [-w <timeout>]] [-i <IPaddress>] [-4 <IPv4>] [-6 <IPv6>][<TargetName>]
```

- **N:** Prevents path ping functioning from attempting to resolve routers' IP addresses to their names.
- **-h MaxHops:** This tracert option specifies the maximum number of hops in the search for the target. If the MaxHops option is not specified the target has not been found by 30 hops then the tracert command will stop looking.
- **-w timeout:** A timeout value must be specified while executing this ping command. It adjusts the amount of time in milliseconds.
- **-ip <IPaddress>:** Indicates the source address.
- **target:** This is the destination IP address or a hostname user want to ping.

**We could also focus on the networking architectures in which these types of networks can be implemented:**

1. PEER-TO-PEER NETWORKING – which might be implemented in a workgroup consisting of computers running Microsoft Windows 98 or Windows 2000 Professional;
2. SERVER-BASED NETWORKING – which might be based on the domain model of Microsoft Windows NT, the domain trees and forests of Active Directory in Windows 2000, or another architecture such as Novell Directory Services (NDS) for Novell NetWare;
3. TERMINAL-BASED NETWORKING – which might be the traditional host-based mainframe environment; the UNIX X Windows environment; the terminal services of Windows NT 4, Server Enterprise Edition; Windows 2000 Advanced Server; or Citrix MetaFrame.

**Conclusion:** so, in this way we studied the basics of networking commands.

## Experiment No. 2

### Construction of CAT 5 Ethernet cable (straight/ cross-over)

#### Apparatus:








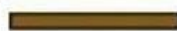




1. RJ-45 connector
2. IO Connector
3. Crimping Tool
4. Twisted pair Cable
5. Cable Tester

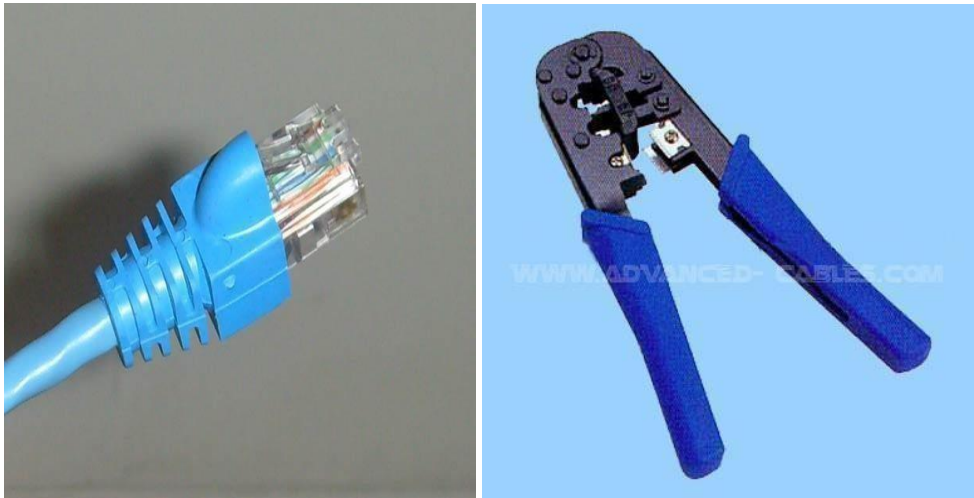
#### Procedure:

To do these practical following steps should be done:

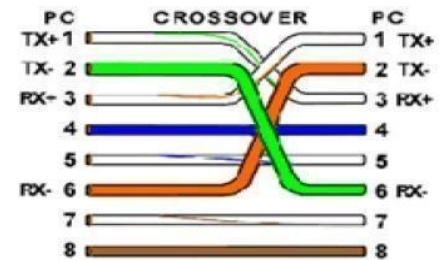
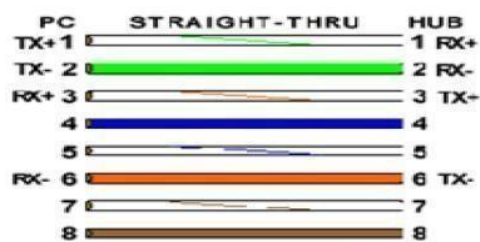
1. Start by stripping off about 2 inches of the plastic jacket off the end of the cable. Be very careful at this point, as to not nick or cut into the wires, which are inside. Doing so could alter the characteristics of your cable, or even worse render it useless. Check the wires, one more time for nicks or cuts. If there are any, just whack the whole end off, and start over.
2. Spread the wires apart, but be sure to hold onto the base of the jacket with your other hand. You do not want the wires to become untwisted down inside the jacket. Category 5 cable must only have 1/2 of an inch of 'untwisted' wire at the end; otherwise, it will be 'out of spec'. At this point, you obviously have ALOT more than 1/2 of an inch of un-twisted wire.
3. You have 2 end jacks, which must be installed on your cable. If you are using a pre-made cable, with one of the ends whacked off, you only have one end to install - the crossed over end. Below are two diagrams, which show how you need to arrange the cables for each type of cable end. Decide at this point which end you are making and examine the associated picture below.

**Diagram shows you how to prepare straight through wired connection**

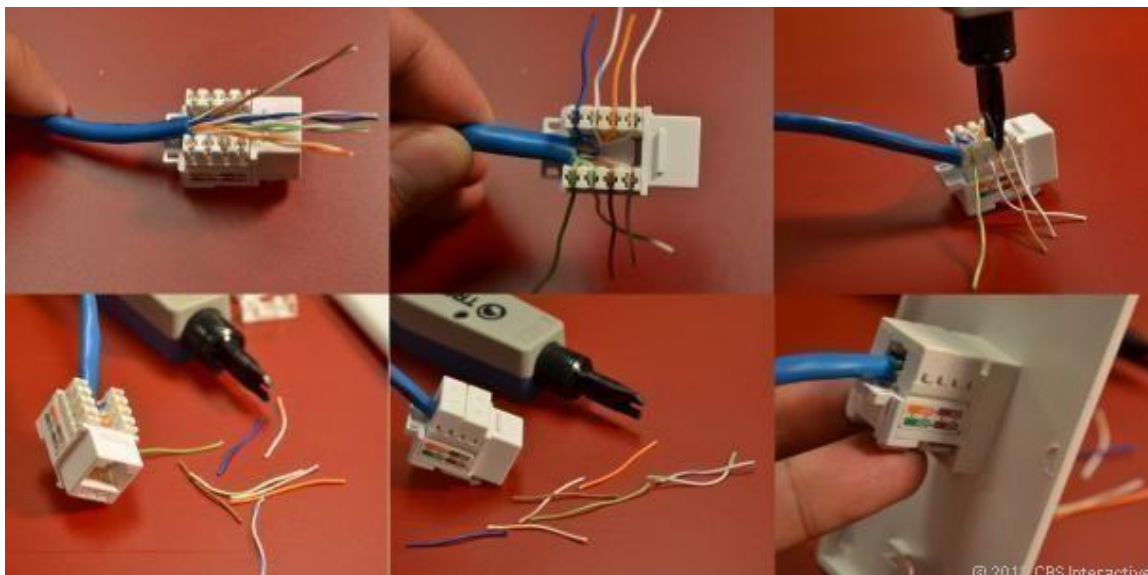
| RJ45 Pin # (END 1) | Wire Color   | Diagram End #1  | RJ45 Pin # (END 2) | Wire Color   | Diagram End #2  |
|--------------------|--------------|---|--------------------|--------------|---|
| 1                  | White/Orange |  | 1                  | White/Green  |  |
| 2                  | Orange       |  | 2                  | Green        |  |
| 3                  | White/Green  |  | 3                  | White/Orange |  |
| 4                  | Blue         |  | 4                  | White/Brown  |  |
| 5                  | White/Blue   |  | 5                  | Brown        |  |
| 6                  | Green        |  | 6                  | Orange       |  |
| 7                  | White/Brown  |  | 7                  | Blue         |  |
| 8                  | Brown        |  | 8                  | White/Blue   |  |



4. After you made it, you don't need to take care of the direction of the cable.



5. At one end, cut the wire to length leaving enough length to work, but not too much excess. Strip off about 2 inches of the Ethernet cable sheath. Align each of the coloured wires according to the layout of the jack. Use the punch down tool to insert each wire into the jack. Repeat the above steps for the second RJ45 jack.



**Result:** so, in this way we learn how to construct the CAT5 ethernet cable (straight / crossover).

## Experiment No. 3

### Study of LAN

#### LAN

A local area network (LAN) is a collection of devices connected together in one physical location, such as a building, office, or home. A LAN can be small or large, ranging from a home network with one user to an enterprise network with thousands of users and devices in an office or school.

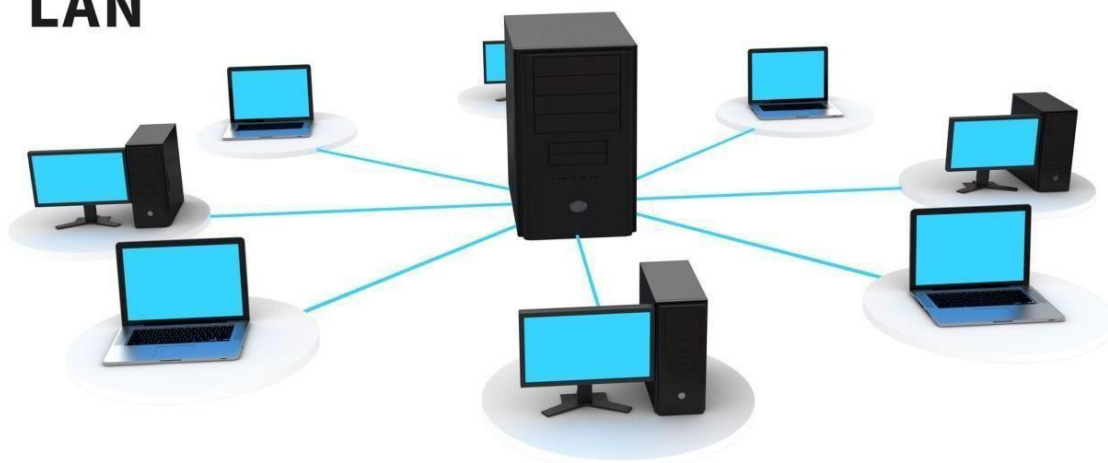
Regardless of size, a LAN's single defining characteristic is that it connects devices that are in a single, limited area. In contrast, a wide area network (WAN) or metropolitan area network (MAN) covers larger geographic areas. Some WANs and MANs connect many LANs together.

#### TYPES OF NETWORK

**Common examples of area network types are:**

1. LAN - Local Area Network
2. WLAN - Wireless Local Area Network
3. WAN - Wide Area Network
4. MAN - Metropolitan Area Network
5. SAN - Storage Area Network, System Area Network, Server Area Network, or sometimes Small Area Network.
6. CAN - Campus Area Network, Controller Area Network, or sometimes Cluster Area Network
7. PAN - Personal Area Network
8. DAN - Desk Area Network

#### LAN

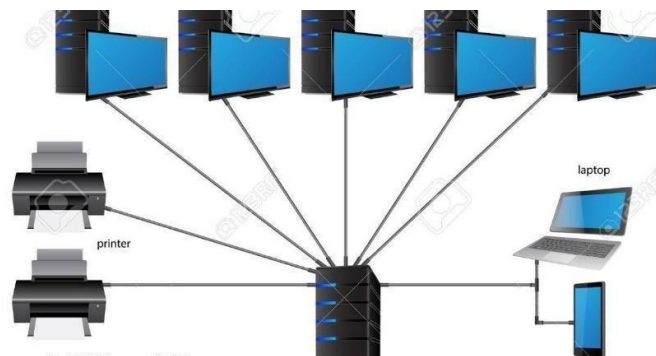




## Procedure to connect the LAN -

### Procedure On the host computer

- On the host computer, follow these steps to share the Internet connection:
- Log on to the host computer as Administrator or as Owner.
- Click Start, and then click Control Panel.
- Click Network and Internet Connections.
- Click Network Connections.
- Right-click the connection that you use to connect to the Internet. For example, if you connect to the Internet by using a modem, right-click the connection that you want under Dial-up / other network available.
- Click Properties.
- Click the Advanced tab.
- Under Internet Connection Sharing, select the Allow other network users to connect through this computer's Internet connection check box.
- If you are sharing a dial-up Internet connection, select the Establish a dial-up connection whenever a computer on my network attempts to access the Internet check box if you
- want to permit your computer to automatically connect to the Internet.
- Click OK. You receive the following message:
- When Internet Connection Sharing is enabled, your LAN adapter will be set to use IP address
- 192.168.0. 1. Your computer may lose connectivity with other computers on your network. If these other computers have static IP addresses, it is a good idea to set them to obtain their IP addresses automatically. Are you sure you want to enable Internet Connection Sharing?
- Click Yes.
- The connection to the Internet is shared to other computers on the local area network (LAN).
- The network adapter that is connected to the LAN is configured with a static IP address of
- 192.168.0. 1 and a subnet mask of 255.255.255.0





### On the client computer

- To connect to the Internet by using the shared connection, you must confirm the LAN adapter IP configuration, and then configure the client computer. To confirm the LAN adapter IP configuration, follow these steps:
- Log on to the client computer as Administrator or as Owner.
- Click Start, and then click Control Panel.
- Click Network and Internet Connections.
- Click Network Connections.
- Right-click Local Area Connection and then click Properties.
- Click the General tab, click Internet Protocol (TCP/IP) in the connection uses the following items list, and then click Properties.
- In the Internet Protocol (TCP/IP) Properties dialog box, click Obtain an IP address automatically (if it is not already selected), and then click OK.
- Note: You can also assign a unique static IP address in the range of 192.168.0.2 to 254.
- For example, you can assign the following static IP address, subnet mask, and default gateway:
  - IP Address 192.168.31.202
  - Subnet mask 255.255.255.0
  - Default gateway 192.168.31.1
- In the Local Area Connection Properties dialog box, click OK.
- Quit Control Panel.

**Conclusion:** So, in this way we studied about the LAN (local area network).

## Experiment No. 4

### Simulation of Bit Stuffing

Q. Create a program for Simulation and implementation of bit stuffing which will have output as

Enter Data Bits:

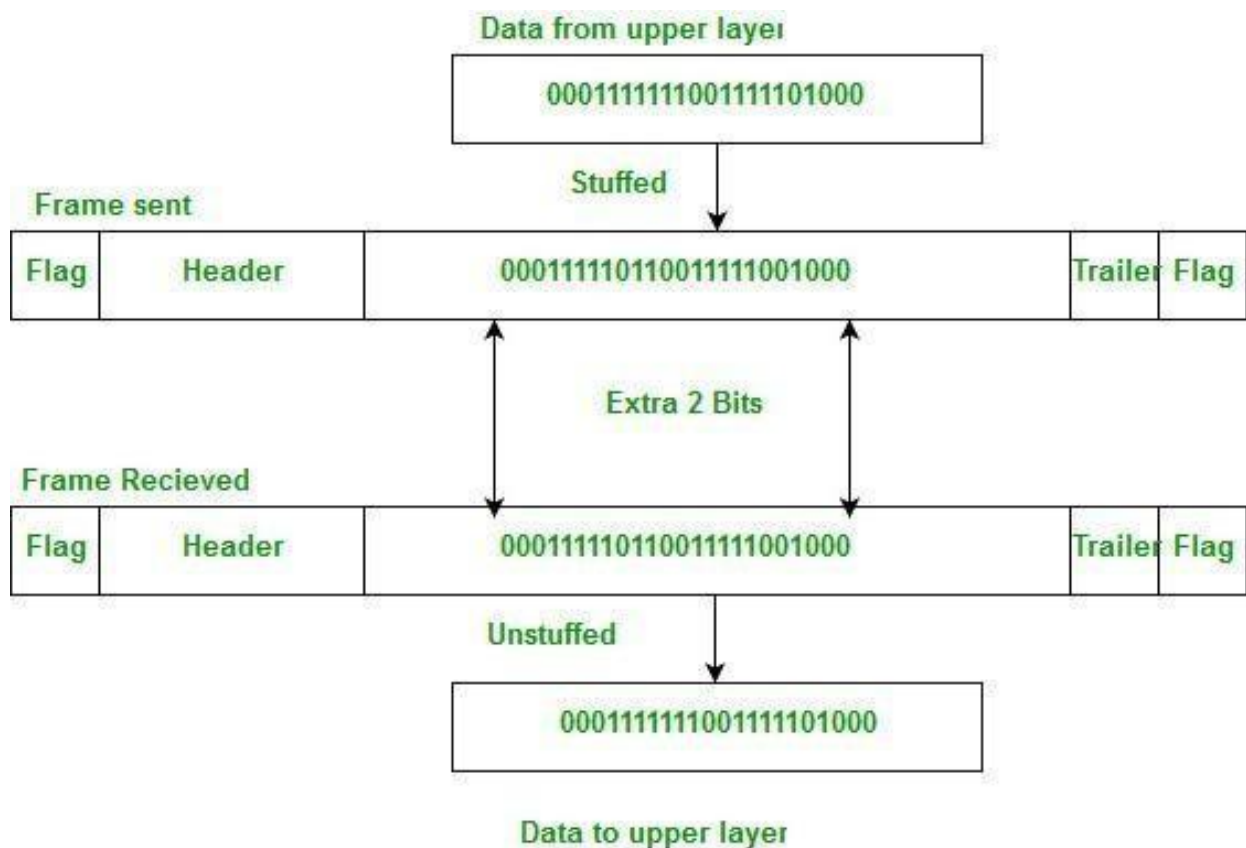
Data Bits Before Bit stuffing:

Data Bits After Bit stuffing:

Theory: -

Data link layer is responsible for something called Framing, which is the division of stream of bits from network layer into manageable units (called frames). Frames could be of fixed size or variable size. In variable-size framing, we need a way to define the end of the frame and the beginning of the next frame.

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.).



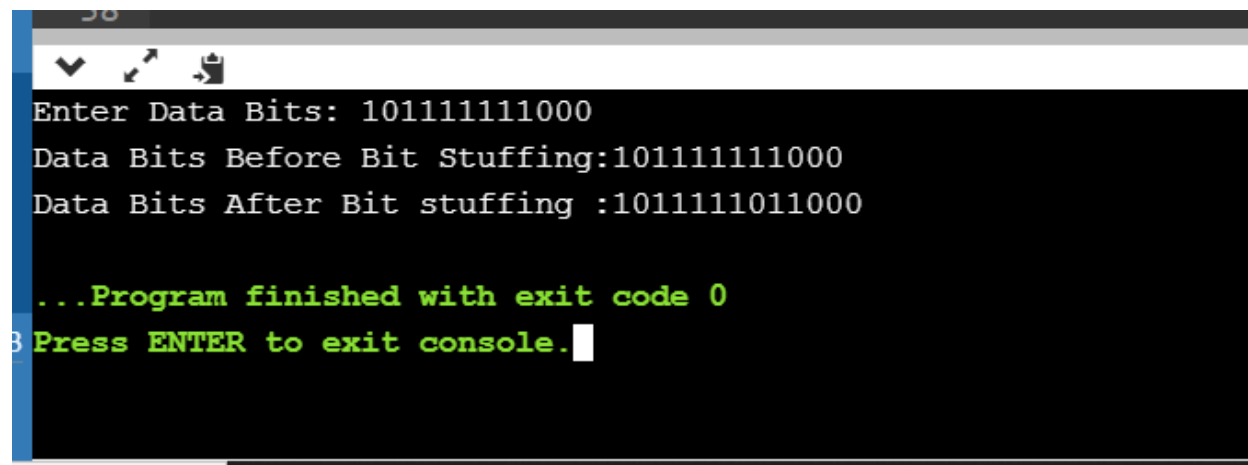
Applications of Bit Stuffing –

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

Code -

```
#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;}
```

Output: -

A screenshot of a terminal window with a black background and white and green text. The text shows the execution of a bit stuffing program. It starts with a prompt to enter data bits, followed by the input '10111111000'. Then it displays the data bits before and after stuffing. The stuffing process is shown as inserting a '0' at the 8th position. The program ends with a green message indicating it finished with exit code 0 and a prompt to press ENTER to exit the console.

```
Enter Data Bits: 10111111000
Data Bits Before Bit Stuffing:10111111000
Data Bits After Bit stuffing :1011111011000

...Program finished with exit code 0
Press ENTER to exit console.
```

**Conclusion :**

So,in this way we studied bit stuffing

## Experiment No. 5

### Simulation of Byte Stuffing

Byte stuffing refers stuffing with character oriented hardware

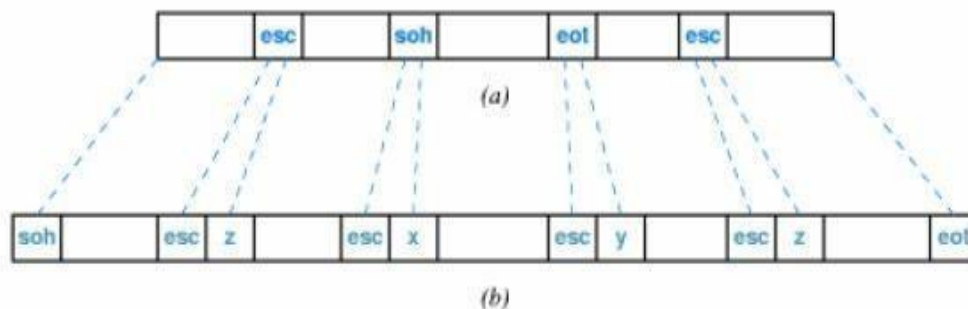
**Byte stuffing** translates each reserved byte into two unreserved bytes. For example: it can use esc as prefix followed by x for soh, y for eot and z for eco.

The receiver then replaces each occurrence of esc x, esc y and esc z by the corresponding single character. This is shown in figure below:

| Character<br>In Data | Characters<br>Sent |
|----------------------|--------------------|
| soh                  | esc x              |
| eot                  | esc y              |
| esc                  | esc z              |

Figure 5.1

Byte stuffing is illustrated in another figure below we can see the replacement of characters.



**Figure 7.5** Illustration of byte stuffing, where (a) is an example of data that includes characters such as *soh*, and (b) is the frame after byte stuffing. The dashed lines show the locations in the original data where characters have been replaced or new characters added.

#### BYTE STUFFING PROGRAM USING C:

```
#include <stdio.h>
#include <string.h>
#define FLAG 0x7E
#define ESC 0x7D
void byteStuffing(unsigned char *data, int *size) {
    unsigned char stuffedData[100];
    int j = 0;
```

## Computer Network Lab

```
for (int i = 0; i < *size; i++) {
    if (data[i] == FLAG || data[i] == ESC) {
        stuffedData[j++] = ESC;
        stuffedData[j++] = data[i] ^ 0x20;
    } else {
        stuffedData[j++] = data[i];
    }
}
*size = j;
memcpy(data, stuffedData, *size);
}

void byteDestuffing(unsigned char *data, int *size) {
    unsigned char destuffedData[100];
    int j = 0;
    for (int i = 0; i < *size; i++) {
        if (data[i] == ESC) {
            i++;
            destuffedData[j++] = data[i] ^ 0x20;
        } else {
            destuffedData[j++] = data[i];
        }
    }
    *size = j;
    memcpy(data, destuffedData, *size);
}

void printData(unsigned char *data, int size) {
    for (int i = 0; i < size; i++) {
        printf("%02X ", data[i]);
    }
    printf("\n");
}

int main() {
    unsigned char data[] = { 0x7E, 0x45, 0x7D, 0x7E, 0x4A, 0x5C };
    int size = sizeof(data) / sizeof(data[0]);

    printf("Original Data: ");
```

## Computer Network Lab

```
printData(data, size);  
byteStuffing(data, &size);  
printf("\nStuffed Data: ");  
printData(data, size);  
byteDestuffing(data, &size);  
printf("\nDestuffed Data: ");  
printData(data, size);  
return 0;  
}
```

Output: -

Original Data: 7E 45 7D 7E 4A 5C

Stuffed Data: 7D 5E 45 7D 5D 7D 5E 4A 5C

Destuffed Data: 7E 45 7D 7E 4A 5C

### **Conclusion :**

So,in this way we studied byte stuffing



## Experiment No. 6 Simulation of

## Cyclic Redundancy Code

Q. Simulation of Cyclic Redundancy Code: Concept of Binary division in CRC generator and checker: Use C/C++/ JAVA/ Python/PHP/MATLAB or any suitable programming language Output will be

Transmission End

1. Input Polynomial
2. Data to be Transmitted
3. Adding Padding Bits if Required
4. Reminder
5. Data Transmitted with Reminder

Receiver End

1. Data Received
2. Divide with Polynomial
3. Show reminder 0
4. Change received Data
5. Perform polynomial division
6. Show Error

Answer -

- If we consider the data unit 1001 and divisor or polynomial generator 1011 their polynomial representation is:

|                        |      |               |
|------------------------|------|---------------|
| Data                   | 1001 | $x^3 + 1$     |
| Division               | 1011 | $x^3 + x + 1$ |
| (polynomial generator) |      |               |

Divisor

$x^3 + x + 1 \overline{) \begin{array}{r} x^6 + \phantom{x^5} + x^3 \\ x^6 + x^4 + x^3 \\ \hline x^4 \\ x^4 + x^2 + x \\ \hline x^2 + x \end{array}}$

$x^3 + x$   
 $x^6 + \phantom{x^5} + x^3$  ← Dividend

$x^4$   
 $x^4 + x^2 + x$   


---

$x^2 + x$

→ Remainder (degree is less than that of divisor)

|                             |             |           |
|-----------------------------|-------------|-----------|
| Data unit to be transmitted | $x^6 + x^3$ | $x^2 + x$ |
|                             | Data        | Remainder |

CRC division using polynomial

- Now string of n 0s (one less than that of divisor) is appended to data. Now data is 1001000 and its corresponding polynomial representation is  $x^6 + x^3$ .
- The division of  $x^6 + x^3$  by  $x^3 + x + 1$  is shown in fig.

Code for Transmitter End [C++] –

```
#include<iostream>
using namespace std;
int main(){
    string
    msg,crc,encoded="";
    cout<<"Enter the Data
    = "; getline(cin,msg);
    cout<<"Enter the Generator Polynomial
    = "; getline(cin,crc);
    int m =
    msg.length(),n=crc.length();
    encoded += msg;
    for (int i = 1; i <= n-1 ; i++)//append n 0s to encoded msg
    {
        encoded += "0";
    }
    for (int i = 0; i <= encoded.length()-n; )
    {
        for (int j = 0; j < n ; j++)
        {
```

## Computer Network Lab

```
encoded[i+j]=encoded[i+j]==crc[j]? '0':'1';

    }

    for (; i < encoded.length() && encoded[i]!='1'; i++);

}

    cout<<"##### Transmitter End
        #####"<<endl;

cout<<"The Input Data = "<<msg<<endl;

cout<<"The Generator Polynomial =

"<<crc<<endl;

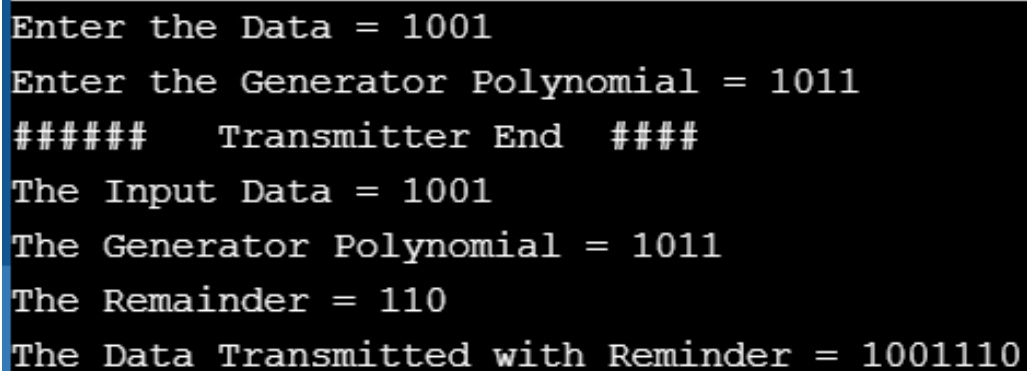
    cout<<"The Remainder = "<<encoded.substr(encoded.length()-n+1)<<endl;

    cout<<"The Data Transmitted with Reminder =

"<<msg+encoded.substr(encoded.length()-n+1)<<endl; return 0;

}
```

Output for Transmitter End [C++] –



```
Enter the Data = 1001
Enter the Generator Polynomial = 1011
##### Transmitter End ####
The Input Data = 1001
The Generator Polynomial = 1011
The Remainder = 110
The Data Transmitted with Reminder = 1001110
```

### Code for Receiver End [C++] –

```
#include<iostream>

using namespace std;

int main(){

    string crc,encoded;

    cout<<"Enter the Data Recived = ";

    getline(cin,encoded);

    cout<<"Enter the Generator Polynomial

    = "; getline(cin,crc);

    cout<<"##### Receiver End

    #####"<<endl; cout<<"Data Received

    = "<<encoded<<endl; int

    n=crc.length();

    for (int i = 0; i <= encoded.length()-n; )//changed the Received data

    {

        for (int j = 0; j < n ; j++)//performed the division

        {

            encoded[i+j]=encoded[i+j]==crc[j]? '0':'1';

        }

        for (; i < encoded.length() && encoded[i]!='1'; i++);

    }
```

## Computer Network Lab

```
cout<<"Divide with Polynomial = "<<crc<<endl;

cout<<"Remainder = "<<encoded.substr(encoded.length()-n+1)<<endl;

for (char i : encoded.substr(encoded.length()-n) ) // checking the Error
{
    if(i!='0'){
        cout<<"Error in communication  "<<endl;
        return 0;
    }
}

cout<<"No Error."<<endl;

return 0;
}
```

Output for Receiver End Without Error –

```
Enter the Data Recived = 1001110
Enter the Generator Polynomial = 1011
##### Receiver End #####
Data Received = 1001110
Divide with Polynomial = 1011
Remainder = 000
No Error.....
```

Output for Receiver End With Error –

```
Enter the Data Recived = 1001111
Enter the Generator Polynomial = 1011
##### Receiver End #####
Data Received = 1001111
Divide with Polynomial = 1011
Remainder = 001
Error in communication.....
```

**Conclusion :**

So,in this way we studied Simulation of Cyclic Redundancy

Code

## Experiment No. 7

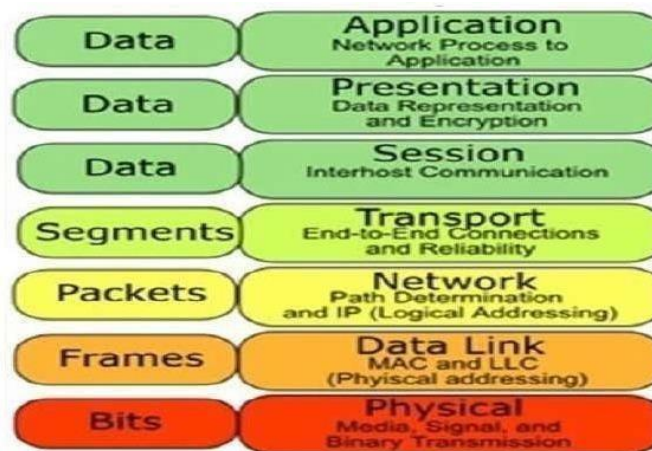
### Study of stop and wait, Go back N protocol

#### What is Protocol?

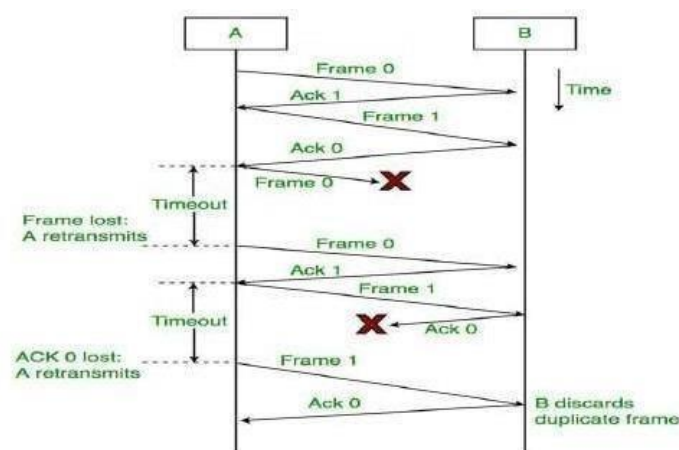
A network protocol defines rules and conventions for communication between network devices. Network protocols include mechanisms for devices to identify and make connections with each other, as well as formatting rules that specify how data is packaged into messages sent and received.

Some protocols also support message acknowledgment and data compression designed for reliable and/or high-performance network communication. Protocols exist at several levels in a telecommunication connection.

For example, there are protocols for the data interchange at the hardware device level and protocols for data interchange at the application program level. In the standard model known as Open Systems Interconnection(OSI), there are one or more protocols at each layer in the telecommunication exchange that both ends of the exchange must recognize and observe.



#### Concept of Stop & Wait Protocol -





It is a fundamental technique to provide reliable transfer of frames in computer network.

1. In this method of flow control, the sender sends a single frame to receiver & waits for an
2. acknowledgment.
3. The next frame is sent by sender only when acknowledgment of previous frame is received.
4. This process of sending a frame & waiting for an acknowledgment continues as long as the
5. sender has data to send.
6. To end up the transmission sender transmits end of transmission (EOT) frame.

Sender:

Rule 1) Send one data at a time

Rule 2) Send next packet only after receiving acknowledgement for previous.

Receiver:

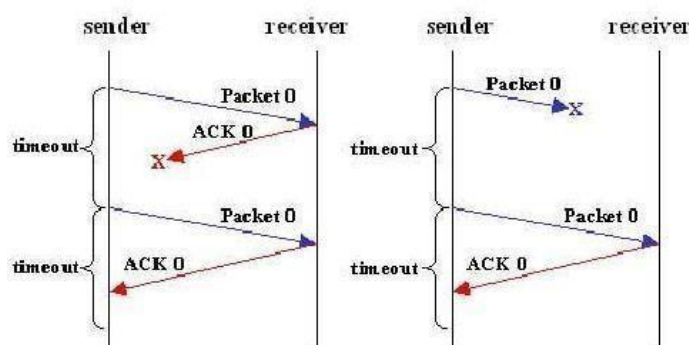
Rule 1) Send Acknowledgement after receiving and consuming of data packet.

Rule 2) After consuming packet acknowledgement need to be sent (Flow Control).

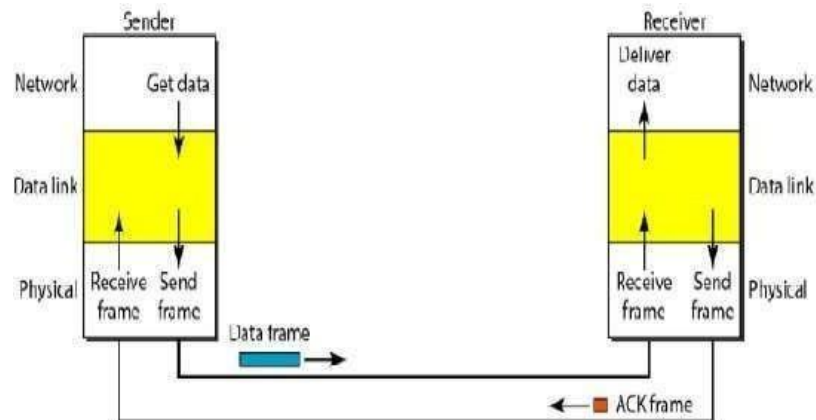
### Working Architecture of Stop & Wait Protocol -

Normal Operation

1. After transmitting one frame, the sender waits for an acknowledgment (ACK) from the receiver before transmitting the next one.
2. In this way, the sender can recognize that the previous frame is transmitted successfully and we could say "stop-n-wait" guarantees reliable transfer between nodes.
3. To support this feature, the sender keeps a record of each frame it sends.
4. To avoid confusion caused by delayed or duplicate ACKs, "stop-n-wait" sends each frame with unique sequence numbers and receives that numbers in each ACKs.
5. Stop-and-wait ARQ is inefficient compared to other ARQs, because the time between packets, if the ACK and the data are received successfully, is twice the transit time (assuming the turnaround time can be zero). The throughput on the channel is a fraction of what it could be. To solve this problem, one can send more than one packet at a time with a larger sequence number and use one ACK for a set Timeout.
6. If the sender doesn't receive ACK of previous frame after a certain period of time (with in fixed time) the sender times out and retransmits that frame again.
7. There are two cases when the sender doesn't receive ACK:1
  1. when the ACK is lost.2.
  2. when the frame itself is not transmitted



## Architecture of Stop and Wait Protocol -

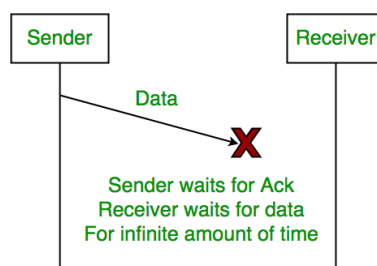


## Advantages of Stop & Wait Protocol -

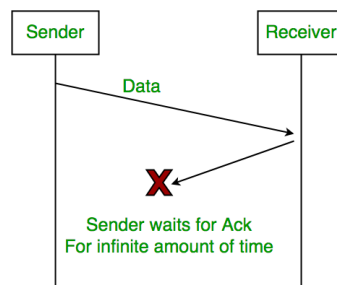
1. The main advantage of stop & wait protocols is its accuracy. Next frame is transmitted only when the first frame is acknowledged. So, there is no chance of frame being lost.
2. It can be used for noisy channels.
3. It has both error and flow control mechanism.
4. It has timer implementation

## Disadvantages of Stop and Wait Protocol -

- Efficiency is very less.
- Only one frame is sent at a time.
- Timer should be set for each individual frame.
- No pipelining.
- Sender window size is 1 (disadvantage over Go Back N ARQ)
- Receiver window size is 1 (disadvantage over Selective Repeat ARQ)
- Lost Data



- Lost Acknowledgement



## C++ program to Implement Stop and Wait Flow Control Protocol:

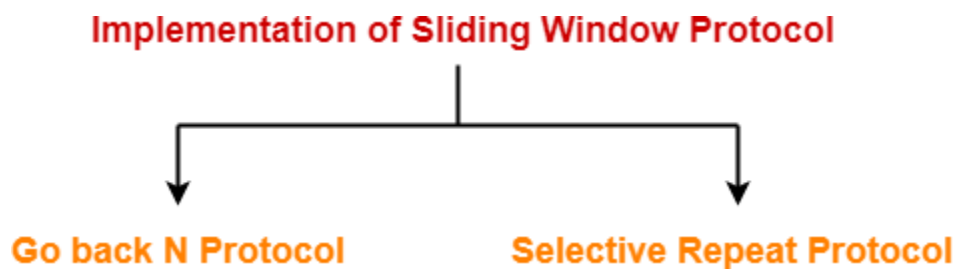
```
#include<iostream>
#include <time.h>
#include <cstdlib>
#include<ctime>
#include <unistd.h>
using namespace std;
class timer {
private:
    unsigned long begTime;
public:
    void start() {
        begTime = clock();
    }
    unsigned long elapsedTime() {
        return ((unsigned long) clock() - begTime) / CLOCKS_PER_SEC;
    }
    bool isTimeout(unsigned long seconds) {
        return seconds >= elapsedTime();
    }
};
int main()
{
    int frames[] = {1,2,3,4,5,6,7,8,9,10};
    unsigned long seconds = 5;
    srand(time(NULL));
    timer t;
    cout<<"Sender has to send frames : ";
    for(int i=0;i<10;i++)
        cout<<frames[i]<<" ";
    cout<<endl;
    int count = 0;
    bool delay = false;
    cout<<endl<<"Sender\t\t\t\t\tReceiver"<<endl;
    do
    {
        bool timeout = false;
        cout<<"Sending Frame : "<<frames[count];
        cout.flush();
        cout<<"\t\t";
        t.start();
        if(rand()%2)
        {
            int to = 24600 + rand()%(64000 - 24600) + 1;
            for(int i=0;i<64000;i++)
                for(int j=0;j<to;j++) {}
        }
        if(t.elapsedTime() <= seconds)
        {
            cout<<"Received Frame : "<<frames[count]<<" ";
            if(delay)
            {
                cout<<"Duplicate";
                delay = false;
            }
            cout<<endl;
            count++;
        }
        else
        {
            cout<<"---"<<endl;
            cout<<"Timeout"<<endl;
            timeout = true;
        }
        t.start();
        if(rand()%2 || !timeout)
        {
```

```
int to = 24600 + rand()%(64000 - 24600) + 1;
for(int i=0;i<64000;i++)
    for(int j=0;j<to;j++) {}
if(t.elapsedTime() > seconds )
{
    cout<<"Delayed Ack"<<endl;
    count--;
    delay = true;
}
else if(!timeout)
    cout<<"Acknowledgement : "<<frames[count]-1<<endl;
}
}while(count!=10);
return 0;
}
```

Output :

Sender has to send frames : 1 2 3 4 5 6 7 8 9 10

Go back N protocol –



The three main characteristic features of GBN are:

**1. Sender Window Size (WS)**

It is N itself. If we say the protocol is GB10, then  $W_s = 10$ . N should be always greater than 1 in order to implement pipelining. For  $N = 1$ , it reduces to Stop and Wait protocol.

$$\text{Efficiency Of GBN} = N/(1+2a)$$

where  $a = T_p/T_t$

If B is the bandwidth of the channel, then

Effective Bandwidth or Throughput

$$= \text{Efficiency} * \text{Bandwidth}$$

$$= (N/(1+2a)) * B$$

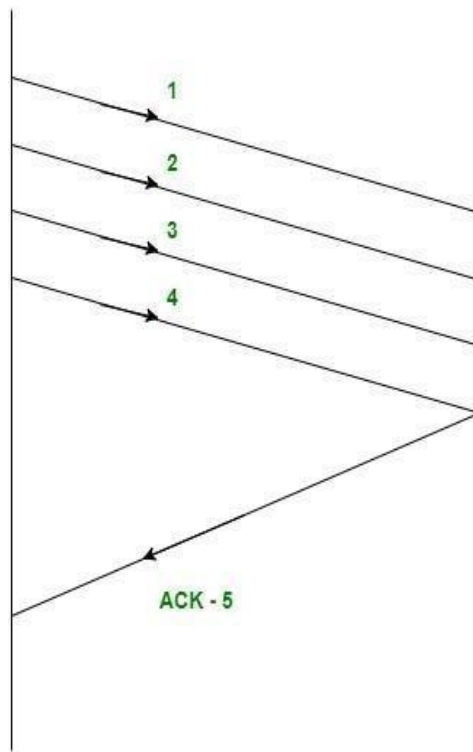
**2. Receiver Window Size (WR)**

WR is always 1 in GBN.

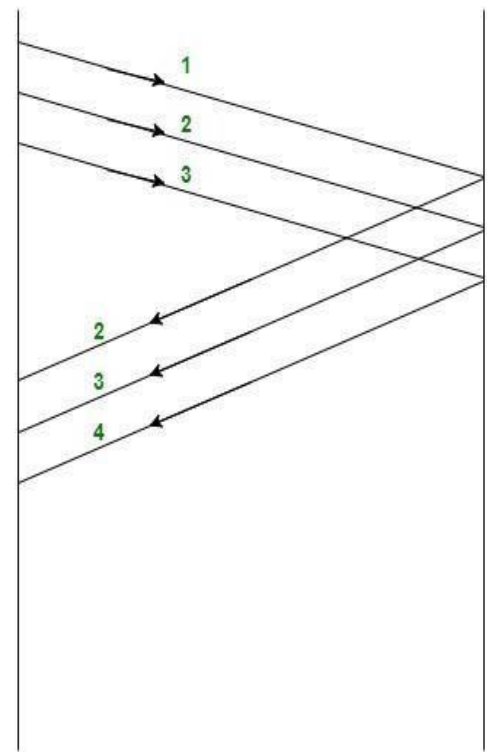
Now what exactly happens in GBN, we will explain with a help of example. Consider the diagram given below. We have sender window size of 4. Assume that we have lots of sequence numbers just for the sake of explanation. Now the sender has sent the packets 0, 1, 2 and 3. After acknowledging the packets 0 and 1, receiver is now expecting packet 2 and sender window has also slid to further transmit the packets 4 and 5. Now suppose the packet 2 is lost in the network, Receiver will discard all the packets which sender has transmitted after packet 2 as it is expecting sequence number of 2.

## Computer Network Lab

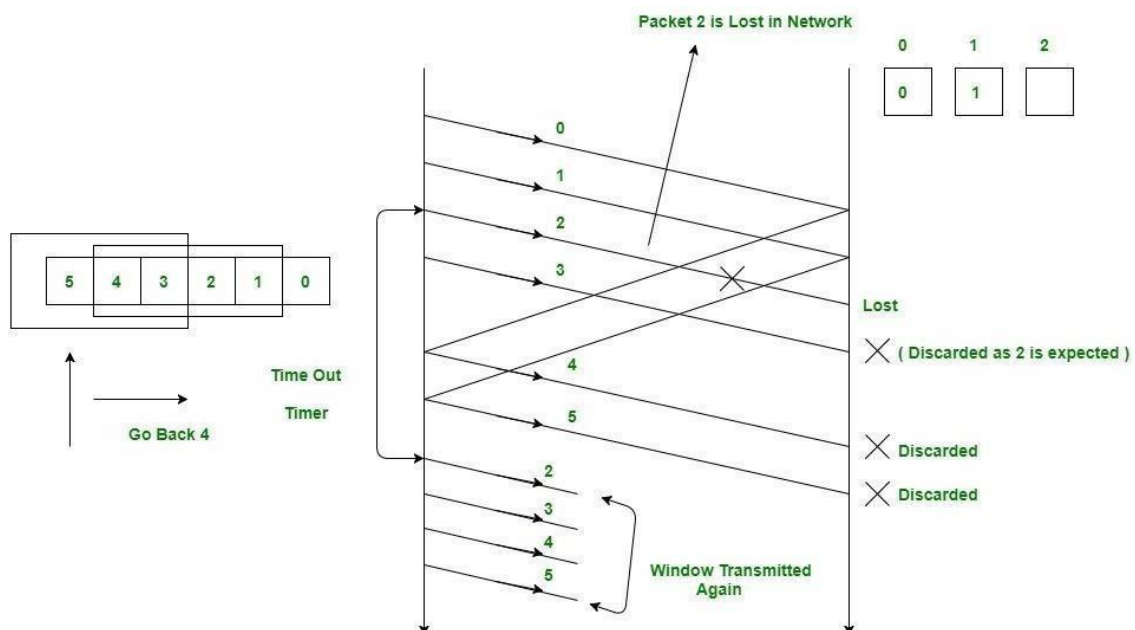
On the sender side for every packet send there is a time out timer which will expire for packet number 2. Now from the last transmitted packet 5 sender will go back to the packet number 2 in the current window and transmit all the packets till packet number 5. That's why it is called Go Back N. Go back means sender has to go back N places from the last transmitted packet in the unacknowledged window and not from the point where the packet is lost.



Cummulative



INDEPENDENT



### 3. Acknowledgements

There are 2 kinds of acknowledgements namely:

1. Cumulative Ack: One acknowledgement is used for many packets. The main advantage is traffic is less. A disadvantage is less reliability as if one ack is the loss that would mean that all the packets sent are lost.
2. Independent Ack: If every packet is going to get acknowledgement independently. Reliability is high here but a disadvantage is that traffic is also high since for every packet we are receiving independent ack.
3. GBN uses Cumulative Acknowledgement. At the receiver side, it starts a acknowledgement timer whenever receiver receives any packet which is fixed and when it expires, it is going to send a cumulative Ack for the number of packets received in that interval of timer. If receiver has received N packets, then the Acknowledgement number will be N+1. Important point is Acknowledgement timer will not start after the expiry of first timer but after receiver has received a packet.
4. Time out timer at the sender side should be greater than Acknowledgement timer.

### C++ Program To Implement Go Back N :

```
#include<iostream>
#include<ctime>
#include<cstdlib>
using namespace std;
int main()
{
    int nf,N;
    int no_tr=0;
    srand(time(NULL));
    cout<<"Enter the number of frames : ";
    cin>>nf;
    cout<<"Enter the Window Size : ";
    cin>>N;
    int i=1;
    while(i<=nf)
    {
        int x=0;
        for(int j=i;j<i+N && j<=nf;j++)
        {

            cout<<"Sent Frame "<<j<<endl;
            no_tr++;
        }
        for(int j=i;j<i+N && j<=nf;j++)
        {
            int flag = rand()%2;
            if(!flag)
            {
                cout<<"Acknowledgment for Frame "<<j<<endl;
                x++;
            }
            else
            {
                cout<<"Frame "<<j<<" Not Received"<<endl;
                cout<<"Retransmitting Window"<<endl;
                break;
            }
        }
    }
}
```

## Computer Network Lab

```
}  
    cout<<endl;  
    i+=x;  
  
}  
cout<<"Total number of transmissions : "<<no_tr<<endl;  
return 0;  
}
```

### Output :

Enter the number of frames: 10  
Enter the Window Size : 3  
Sent Frame 1  
Sent Frame 2  
Sent Frame 3 Acknowledgment for  
Frame 1 Acknowledgment for  
Frame 2 Acknowledgment for  
Frame 3

Sent Frame 4  
Sent Frame 5  
Sent Frame 6  
Frame 4 Not Received  
Retransmitting Window

Sent Frame 4  
Sent Frame 5  
Sent Frame 6 Acknowledgment for  
Frame 4 Acknowledgment for  
Frame 5 Acknowledgment for  
Frame 6

Sent Frame 7  
Sent Frame 8  
Sent Frame 9 Acknowledgment for  
Frame 7 Acknowledgment for  
Frame 8 Acknowledgment for  
Frame 9

Sent Frame 10 Acknowledgment for  
Frame 10

Total number of transmissions: 13

**Conclusion:** so, in this way we studied the stop and wait, Go back N protocol



## Experiment No. 8

### Study of Token Bucket algorithm

A token bucket provides a mechanism that allows a desired level of burstiness within a flow by limiting its average rate as well as its [maximum burst size](#). A token bucket can be viewed as an abstraction of the rate of transfer expressed as a relationship between a committed burst size CBS, a [committed information rate](#) CIR, and a time interval T.

$$CIR = CBS/T$$

Where,

- Committed information rate (CIR) specifies the amount of data that can be sent per unit time interval and represents the long-term average rate at which a flow can send its data into the network. Sometimes, this parameter is referred to as the mean rate. The mean rate is specified in bits per sec or bytes per sec as IP packets are of variable lengths.
- Committed burst size (CBS) specifies the maximum number of bytes that can be transmitted into the network in an extremely short interval of time. In theory, as the time interval tends to zero, the committed burst size represents the number of bytes that can be instantaneously transmitted into the network. However, in practice, it is not possible to instantaneously send multiple bytes into the network since there is an upper bound on the physical transmission rate that cannot be exceeded.
- Time interval (T) specifies the time per burst.

According to the definition, the rate of the traffic sent over the network, and over any integral multiple of the time interval, will not exceed the mean rate. However, within the interval, the rate can be arbitrarily fast. For instance, consider a flow that is constrained to send data at an average rate of 12,000 bits per sec. This flow can send 3000 bits within a time interval of 100 millisecond. When considering this 100-millisecond interval, it might appear that its average rate is 30,000 bits per sec. However, as long as the flow does not send more than 9000 bits in the second interval containing the 100 milliseconds in which it sent 3000 bits, it will not exceed the average of 12,000 bits per sec. Before examining the token bucket algorithm, let us consider an example of how a token bucket is specified.

### Example

Specifying a token bucket.

Assume that the traffic needs to be sent into the network at a mean rate CIR of 2.4 Mbps. If a burst for a duration of 10 millisecc (= 0.01 sec) needs to be sustained, CBS can be calculated using the token bucket definition as

$$CBS = \frac{2400000 \text{ bits/sec} \times 0.01 \text{ sec}}{8 \text{ bytes/byte}}$$

which yields 3000 bytes. Thus, the token rate is 300,000 ( $= 2,400,000/8$ ) bytes per sec, the committed burst size is 3000 bytes, and the token interval (T) is 10 millisec. Therefore, the token generator credits the token bucket with 3000 bytes worth of tokens every 10 millisec. This indicates that the conforming traffic will, in the worst case, come in 100 bursts per sec of 3000 bytes each and at a committed information rate not exceeding 2.4 Mbps.

### Algorithm

Based on the token bucket definition, an algorithm can be devised that controls the rate of the traffic in such a way that over a [long time period](#) the average allowed rate approaches the desired mean rate CIR asymptotically and over a short time interval the burst size of the traffic is upper bounded by bucket size CBS. The algorithm assigns a token bucket for each flow that consists of a bucket that can hold up to CBS tokens. Each token can be considered as permission for the source to transmit a certain number of bits into the network. These CBS tokens in the bucket represent the allowed burst size. New tokens are added to the bucket at the rate of CIR tokens per token interval. If the bucket contains less than CBS tokens when a new token is generated, it is added to the bucket; otherwise, the newly generated token is dropped and the token bucket remains filled with CBS tokens. A conceptual illustration of the token bucket algorithm is shown below

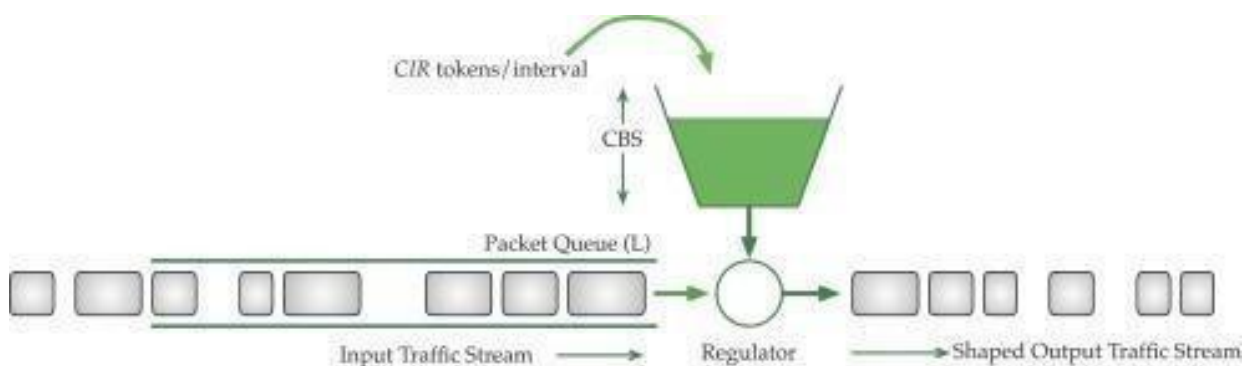


Figure. Token bucket algorithm.

Now, continuing with the algorithm, the [packets arriving](#) in a flow are placed into a packet queue that has a maximum length of L. If the flow delivers more packets than the queue can store, the excess packets are discarded. When a packet of size P bytes arrives in a flow, one of the following cases will apply.

If the token bucket is full, P tokens are removed before the packet is transmitted into the network. If the token bucket is empty, the packet is queued until P tokens are accumulated in the bucket. Eventually, when the bucket contains P tokens, that many tokens are removed from the bucket and the packets are transmitted into the network.

Finally, consider the case when the token bucket is partially filled with say X tokens. If  $P \leq X$ , then P tokens are removed from the bucket and the packet is sent into the network. If  $P > X$ , the packet is queued and it waits until the remaining  $P - X$  tokens are accumulated. Once the bucket accumulates the required P tokens, they are removed from the bucket and the packet is forwarded into the network.

If packets arrive in short bursts, up to CBS tokens would be available in the bucket, and thus, up to CBS bytes would still get through. As a consequence, the maximum burst size is limited to at most CBS bytes for the flow. Furthermore, as the token replenishment rate is CIR tokens every token interval,

the maximum number of bytes that can enter the network over any time period of  $t$  is  $CBS + t \times CIR/T$ . Thus, the token replenishment rate, CIR, serves to limit the long-term average rate of bytes entering the network. The length of the packet queue bounds the delay incurred by a packet.

#### Implementation of Token Bucket Using Python code :

```
#include <iostream>

#include <chrono>

#include <thread>

#include <algorithm>

class TokenBucket {

private:

double capacity;

double tokens;

double fill_rate; // tokens per second

std::chrono::steady_clock::time_point last_refill;

void refill() {

    auto now = std::chrono::steady_clock::now();

    std::chrono::duration<double> elapsed = now - last_refill;

    double new_tokens = fill_rate * elapsed.count();

    tokens = std::min(capacity, tokens + new_tokens);

    last_refill = now;

}

public:

TokenBucket(double capacity, double fill_rate)

: capacity(capacity), tokens(capacity), fill_rate(fill_rate),

last_refill(std::chrono::steady_clock::now()) {}

bool consume(double amount) {

    refill();

    if (tokens >= amount) {

        tokens -= amount;

        return true;
```

## Computer Network Lab

```
    }

    return false;
}

double get_tokens() {
    refill();
    return tokens;
}

};

int main() {
    TokenBucket bucket(80, 1); // 80 tokens, 1 token/sec

    std::cout << "tokens = " << bucket.get_tokens() << std::endl;
    std::cout << "consume(10) = " << bucket.consume(10) << std::endl;
    std::cout << "consume(10) = " << bucket.consume(10) << std::endl;

    std::this_thread::sleep_for(std::chrono::seconds(1));
    std::cout << "tokens = " << bucket.get_tokens() << std::endl;

    std::this_thread::sleep_for(std::chrono::seconds(1));
    std::cout << "tokens = " << bucket.get_tokens() << std::endl;

    std::cout << "consume(90) = " << bucket.consume(90) << std::endl;
    std::cout << "tokens = " << bucket.get_tokens() << std::endl;

    return 0;
}
```

**Output:**

```
tokens = 80  
consume(10) = 1  
consume(10) = 1  
tokens = 61.0001  
tokens = 62.0002  
consume(90) = 0  
tokens = 62.0003
```

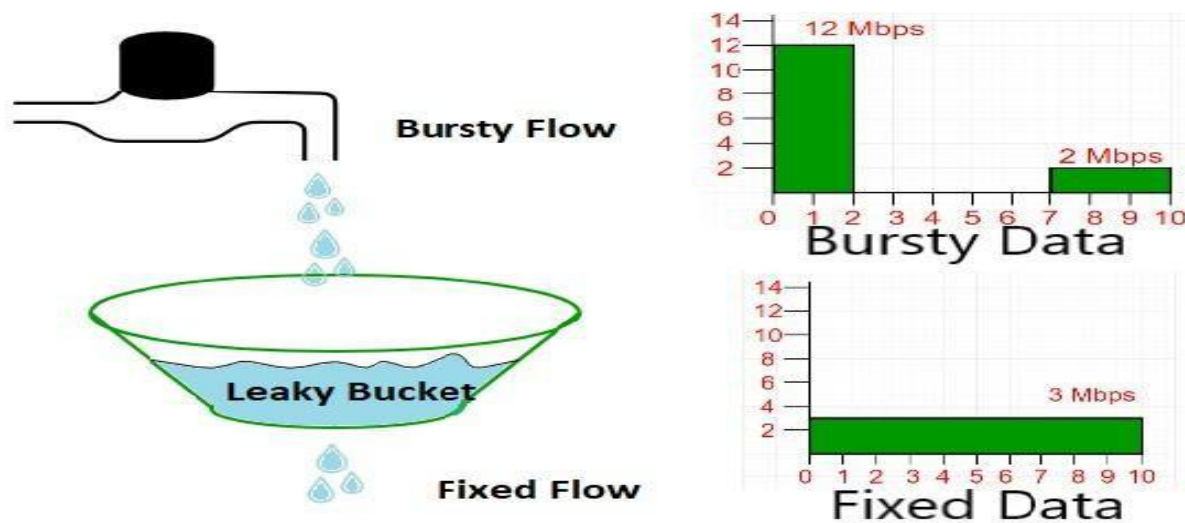
**Conclusion:** so, in this way we studied the token bucket algorithm.

## Experiment No. 9

### Study of Leaky Bucket algorithm

Suppose we have a bucket in which we are pouring water in a random order but we have to get water in a fixed rate, for this we will make a hole at the bottom of the bucket. It will ensure that water coming out is in a some fixed rate, and also if bucket will full we will stop pouring in it.

The input rate can vary, but the output rate remains constant. Similarly, in networking, a technique called leaky bucket can smooth out bursty traffic. Bursty chunks are stored in the bucket and sent out at an average rate.



In the figure, we assume that the network has committed a bandwidth of 3 Mbps for a host. The use of the leaky bucket shapes the input traffic to make it conform to this commitment. In Figure the host sends a burst of data at a rate of 12 Mbps for 2 s, for a total of 24 Mbits of data. The host is silent for 5 s and then sends data at a rate of 2 Mbps for 3 s, for a total of 6 Mbits of data. In all, the host has sent 30 Mbits of data in 10 s. The leaky bucket smooths the traffic by sending out data at a rate of 3 Mbps during the same 10 s.

Without the leaky bucket, the beginning burst may have hurt the network by consuming more bandwidth than is set aside for this host. We can also see that the leaky bucket may prevent congestion.

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

The following is an algorithm for variable-length packets:

1. Initialize a counter to  $n$  at the tick of the clock.
2. If  $n$  is greater than the size of the packet, send the packet and decrement the counter by the packet

## Computer Network

size. Repeat this step until  $n$  is smaller than the packet size.

3. Reset the counter and go to step

**Example** – Let  $n=1000$  Packet=

|     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|
| 200 | 700 | 500 | 450 | 400 | 200 |
|-----|-----|-----|-----|-----|-----|

Since  $n > \text{front of Queue}$  i.e.  $n > 200$  Therefore,  $n = 1000 - 200 = 800$

Packet size of 200 is sent to the network.

|     |     |     |     |     |
|-----|-----|-----|-----|-----|
| 200 | 700 | 500 | 450 | 400 |
|-----|-----|-----|-----|-----|

Now Again  $n > \text{front of the queue}$  i.e.  $n > 400$  Therefore,  $n = 800 - 400 = 400$

Packet size of 400 is sent to the network.

|     |     |     |     |
|-----|-----|-----|-----|
| 200 | 700 | 500 | 450 |
|-----|-----|-----|-----|

Since  $n < \text{front of queue}$  Therefore, the procedure is stop.

Initialize  $n=1000$  on another tick of clock.

This procedure is repeated until all the packets are sent to the network. Below is the implementation of above approach:

**cpp program to implement leakybucket :**

```
#include<bits/stdc++.h> using namespace std; int main()
```

```
{
```

```
int no_of_queries, storage, output_pkt_size; int input_pkt_size, bucket_size, size_left;
```



## Computer Network

```
// initial packets in the bucket storage = 0;
// total no. of times bucket content is checked no_of_queries = 4;
// total no. of packets that can
// be accommodated in the bucket bucket_size = 10;
// no. of packets that enters the bucket at a time input_pkt_size = 4;
// no. of packets that exits the bucket at a time output_pkt_size = 1;
for(int i = 0; i < no_of_queries; i++) //space left
{
    size_left = bucket_size - storage; if(input_pkt_size <= size_left)
    {
        // update storage storage += input_pkt_size;
        printf("Buffer size= %d out of bucket size= %d\n", storage, bucket_size);
    }
    else
    {
        printf("Packet loss = %d\n", (input_pkt_size-(size_left)));
        // full size storage=bucket_size;
        printf("Buffer size= %d out of bucket size= %d\n", storage, bucket_size);
    }
    storage -= output_pkt_size;
}
return 0;
```

### Output :

Buffer size= 4 out of bucket size= 10 Buffer size= 7 out of bucket size= 10 Buffer size= 10 out of bucket size= 10 Packet loss = 3

Buffer size= 10 out of bucket size= 10

**Conclusion :** so, in this way we studied the leaky bucket algorithm.

## Experiment No.10

### Implementation of distance vector routing algorithm using C++.

#### Theory:

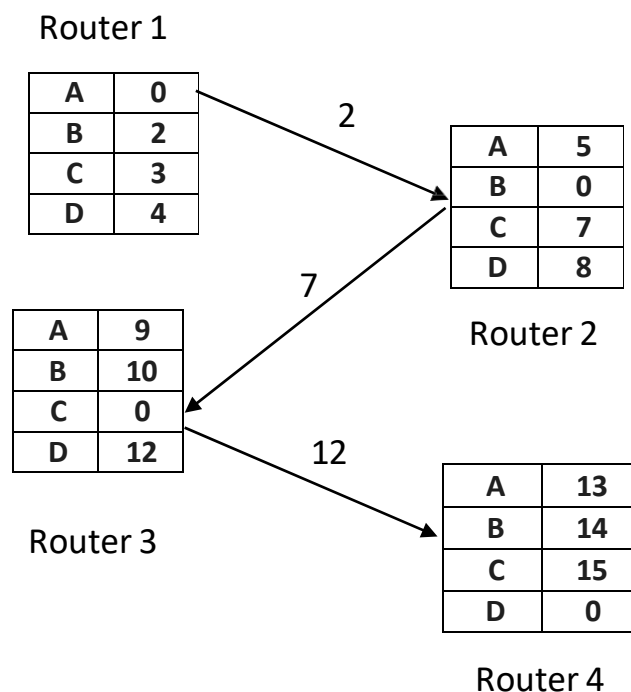
A distance-vector routing protocol requires that a router informs its neighbours of topology changes periodically. Compared to link-state protocols, which require a router to inform all the nodes in a network of topology changes, distance-vector routing protocols have less computational complexity and message overhead. The term distance vector refers to the fact that the protocol manipulates vectors (arrays) of distances to other nodes in the network. The vector distance algorithm was the original ARPANET routing algorithm and was also used in the internet under the name of RIP.

#### Instead, they use two methods:

1. Direction in which router or exit interface a packet should be forwarded.
2. Distance from its destination.

Distance-vector protocols are based on calculating the direction and distance to any link in a network. "Direction" usually means the next hop address and the exit interface. "Distance" is a measure of the cost to reach a certain node.

Updates are performed periodically in a distance-vector protocol where all or part of a router's routing table is sent to all its neighbours that are configured to use the same distance- vector routing protocol. RIP supports cross-platform distance vector routing whereas IGRP is a Cisco Systems proprietary distance vector routing protocol. Once a router has this information it is able to amend its own routing table to reflect the changes and then inform its neighbours of the changes.



**Program:**

```
#include<stdio.h>
#include<iostream>
using namespace std;
struct node
{
    unsigned dist[6];
    unsigned from[6];
}DVR[10];
int main()
{
    cout<<"\n\n----- Distance Vector Routing Algorithm -----";
    int costmat[6][6];
    int nodes, i, j, k;
    cout<<"\n\n Enter the number of nodes : ";
    cin>>nodes; //Enter the nodes
    cout<<"\n Enter the cost matrix: \n";
    for (i = 0; i < nodes; i++)
    {
        for (j = 0; j < nodes; j++)
        {
            cin>>costmat[i][j];
            costmat[i][i] = 0;
            DVR[i].dist[j] = costmat[i][j]; //initialise the distance equal to cost matrix
            DVR[i].from[j] = j;
        }
    }

    for(i = 0; i < nodes; i++) //We choose arbitrary vertex k and we calculate the
    //direct distance from the node i to k using the cost matrix and add the distance from k to
    node j
        for(j = i+1; j < nodes; j++)
            for(k = 0; k < nodes; k++)
                if(DVR[i].dist[j] > costmat[i][k] + DVR[k].dist[j])
                { //We calculate the minimum distance
                    DVR[i].dist[j] = DVR[i].dist[k] + DVR[k].dist[j];
                    DVR[j].dist[i] = DVR[i].dist[j];
                    DVR[i].from[j] = k;
                    DVR[j].from[i] = k;
                }
    for(i = 0; i < nodes; i++)
    {
        cout<<"\n\n For router: "<<i+1;
        for (j = 0; j < nodes; j++)
            cout<<"\t\n node "<<j+1<<" via "<<DVR[i].from[j]+1<<" Distance "<<DVR[i].dist[j];
    }
    cout<<"\n\n ";
    return 0;
```

## Computer Network

}

### Output:

----- Distance Vector Routing Algorithm-----

Enter the number of nodes: 4

Enter the cost matrix:

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16

For router: 1

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

For router: 2

node 1 via 1 Distance 5  
node 2 via 2 Distance 0  
node 3 via 3 Distance 7  
node 4 via 4 Distance 8

For router: 3

node 1 via 1 Distance 9  
node 2 via 2 Distance 10  
node 3 via 3 Distance 0  
node 4 via 4 Distance 12

For router: 4

node 1 via 1 Distance 13  
node 2 via 2 Distance 14  
node 3 via 3 Distance 15  
node 4 via 4 Distance 0

## Experiment No. 11

Configure DHCP Server on Cisco Packet Tracer

Apparatus:

1) Cisco Packet Tracer

2) Devices:

- 1 Router
- 1 Switch
- 1 Server
- 2 or more PCs

Procedure:

Step 1: Create the Network Topology

1. Drag and place the following devices from the device list:
  - 1 Router (e.g., 2911)
  - 1 Switch
  - 1 Server
  - 2 PCs
2. Connect them using Copper Straight-through cables.

Step 2: Assign Static IP to the DHCP Server

1. Click on the Server → Desktop → IP Configuration.
2. Enter:

o IP Address: 192.168.1.2

o Subnet Mask: 255.255.255.0

- Default Gateway: 192.168.1.1

Step 3: Configure the DHCP Service on the Server

1. Click on the Server → Services tab → DHCP.
2. Turn the DHCP service ON.
3. Fill the following:
  - Pool Name: e.g., Pool1
  - Default Gateway: 192.168.1.1
  - DNS Server: 8.8.8.8 or leave default
  - Start IP Address: 192.168.1.10
  - Subnet Mask: 255.255.255.0
  - Maximum Users: 50
4. Click on Add.

Step 4: Set PCs to Obtain IP Automatically

1. Click on each PC → Desktop → IP Configuration.
2. Select DHCP.
3. The IP should be automatically assigned from the server.

Step 5: Verification:-

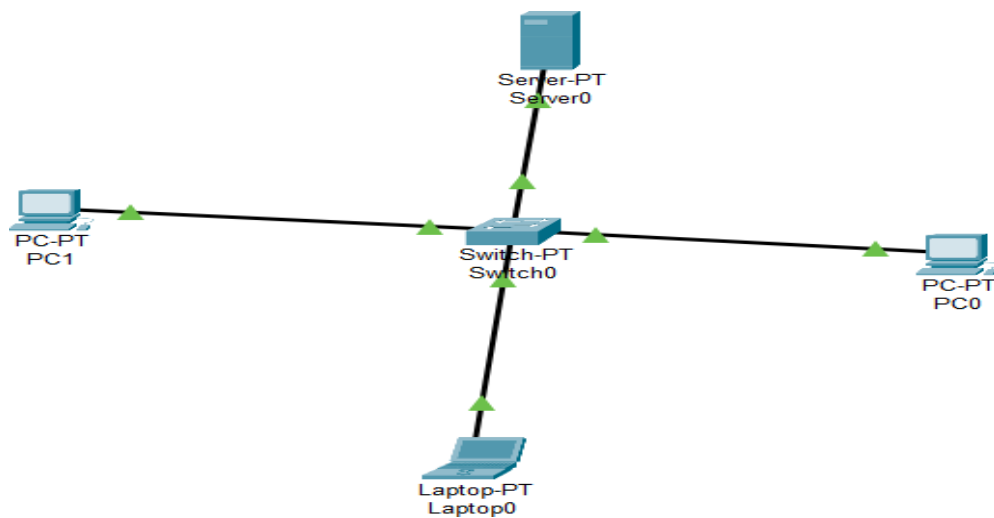
Now verify the configuration using the ping command:

- Click on PC0 → Desktop → Command Prompt
- Type the following and press Enter: ping 192.168.1.1

If everything is configured correctly, you should see replies from the router IP, which confirms network connectivity. If everything is configured correctly, you should see replies from the router IP, which confirms network connectivity.

## Computer Network

### Topology Diagram:-



### Result:

```
PC1
Physical Config Desktop Programming Attributes
Command Prompt
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.1.4

Pinging 192.168.1.4 with 32 bytes of data:

Reply from 192.168.1.4: bytes=32 time<1ms TTL=128
Reply from 192.168.1.4: bytes=32 time=1ms TTL=128
Reply from 192.168.1.4: bytes=32 time<1ms TTL=128
Reply from 192.168.1.4: bytes=32 time=17ms TTL=128

Ping statistics for 192.168.1.4:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 17ms, Average = 4ms

C:\>
```

The screenshot shows the 'Desktop' tab of the PC1 configuration window in Cisco Packet Tracer. The 'Command Prompt' application is open, displaying the output of a 'ping 192.168.1.4' command. The output shows four successful replies with varying round-trip times (all less than 17ms) and a 0% loss rate. The ping statistics summary at the bottom confirms that all four packets were sent and received successfully.

**Conclusion:-** Successfully configured and verified a DHCP server in Cisco Packet Tracer to automatically assign IP addresses to client devices.

## Experiment No:- 12

**Aim:** Basic network configuration of switch and router in cisco packet tracer.

**Software:** cisco packet tracer.

**Theory:**

**Switch:**



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

**Routers:**



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

Procedure:

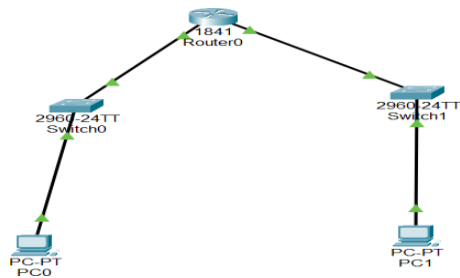
**Step 1:**

## Computer Network

Open Cisco Packet Tracer Cisco Packet Tracer on your system. Drag and drop a Router, Switch, and PCs from the device panel.

### Step 2:

Connect Devices Using Cables Use Copper Straight-Through Cable to connect: PC to Switch (FastEthernet0 to FastEthernet port) Switch to Router (FastEthernet0/1 to GigabitEthernet0/0) Use Copper Cross-Over Cable to connect two routers (if required).



### Step 3:

Assign IP Address to PC

1. Click on the PC, go to the Desktop tab, and open IP Configuration.
2. Assign the following IP:

IP Address: 192.168.1.2

Subnet Mask: 255.255.255.0

Default Gateway: 192.168.1.1 (Router's IP)

### Step 4:

Configure the Router

1. Click on the Router, go to the CLI.
2. Enter the following commands to assign IP and enable the interface:



```
Would you like to enter the initial configuration dialog? [yes/no]: no

Press RETURN to get started!

Router>enable
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface g0/0
Router(config-if)#ip address 192.168.1.1 255.255.255.0
Router(config-if)#no shutdown

Router(config-if)#
%LINK-3-CHANGED: Interface GigabitEthernet0/0, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/0, changed state to up

Router(config-if)#exit
Router(config)#interface g0/1
Router(config-if)#ip address 192.168.2.1 255.255.255.0
Router(config-if)#no shutdown
```



### Step 5:

Verify the Configuration Use the ping command to check connectivity:

Open the PC Command Prompt and type: ping 192.168.1.1

If successful, the connection is working properly.

```
Command Prompt

Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.2.11

Pinging 192.168.2.11 with 32 bytes of data:

Reply from 192.168.2.11: bytes=32 time<1ms TTL=127
Reply from 192.168.2.11: bytes=32 time=8ms TTL=127
Reply from 192.168.2.11: bytes=32 time<1ms TTL=127
Reply from 192.168.2.11: bytes=32 time=1ms TTL=127

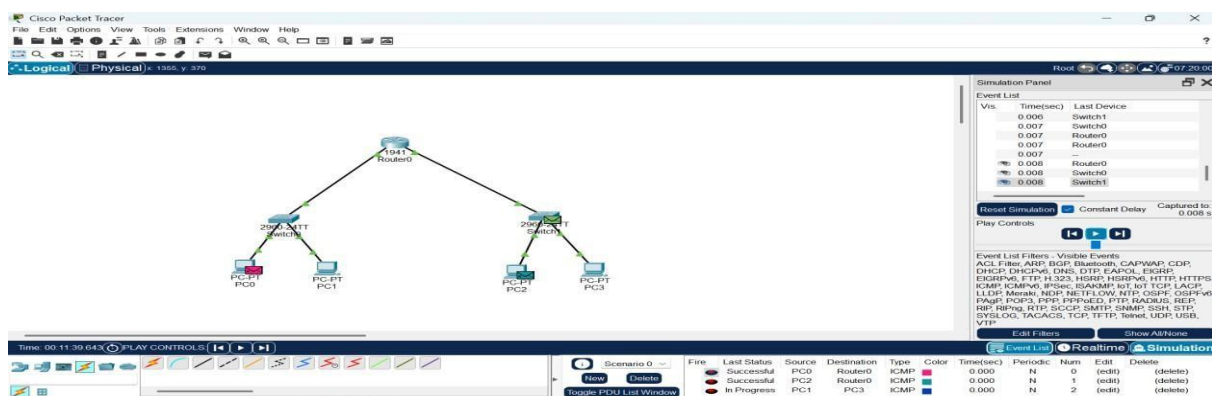
Ping statistics for 192.168.2.11:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 8ms, Average = 2ms

C:\>ping 192.168.2.11

Pinging 192.168.2.11 with 32 bytes of data:

Reply from 192.168.2.11: bytes=32 time=1ms TTL=127
Reply from 192.168.2.11: bytes=32 time=1ms TTL=127
Reply from 192.168.2.11: bytes=32 time<1ms TTL=127
Reply from 192.168.2.11: bytes=32 time<1ms TTL=127

Ping statistics for 192.168.2.11:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms
```



### Conclusion

In this experiment we can design the network using switch and router in cisco packet tracer.

## Experiment No. 13

### Study of LAN

#### Introduction to LAN:

A local area network (LAN) is a collection of devices connected together in one physical location, such as a building, office, or home. A LAN can be small or large, ranging from a home network with one user to an enterprise network with thousands of users and devices in an office or school.

Regardless of size, a LAN's single defining characteristic is that it connects devices that are in a single, limited area. In contrast, a wide area network (WAN) or metropolitan area network (MAN) covers larger geographic areas. Some WANs and MANs connect many LANs together.

#### TYPES OF NETWORK

**Common examples of area network types are:**

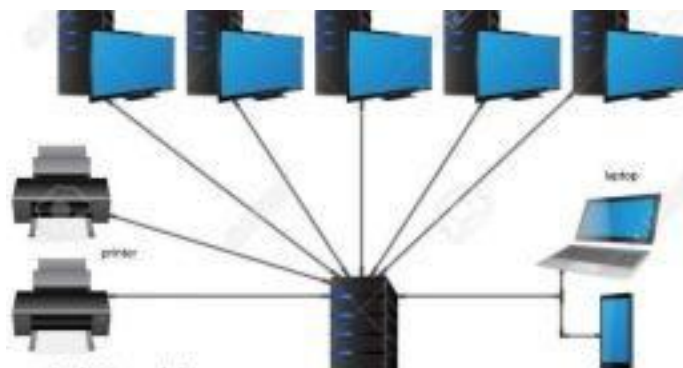
1. LAN - Local Area Network
2. WLAN - Wireless Local Area Network
3. WAN - Wide Area Network
4. MAN - Metropolitan Area Network
5. SAN - Storage Area Network, System Area Network, Server Area Network, or sometimes Small Area Network.
6. CAN - Campus Area Network, Controller Area Network, or sometimes Cluster Area Network
7. PAN - Personal Area Network
8. DAN - Desk Area Network



## Procedure to connect the LAN -

### Procedure On the host computer

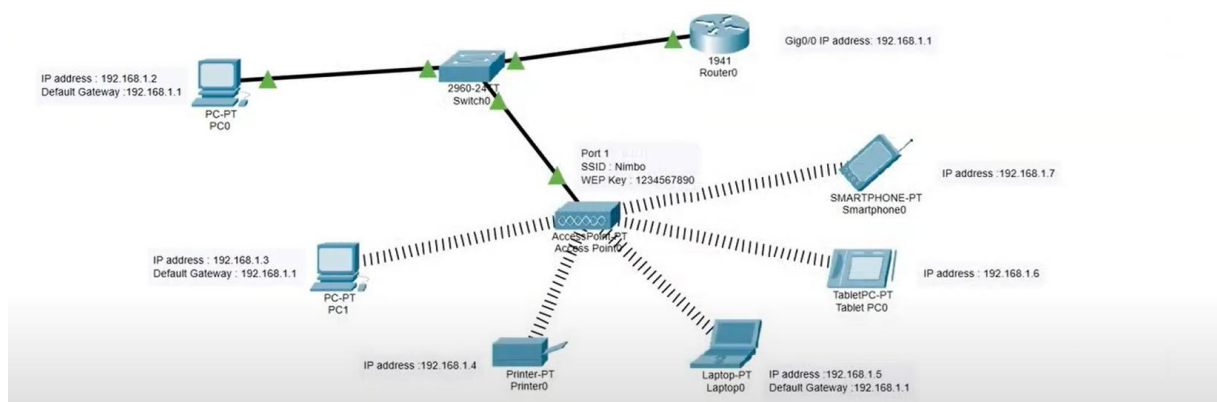
- On the host computer, follow these steps to share the Internet connection:
  - Log on to the host computer as Administrator or as Owner.
  - Click Start, and then click Control Panel.
  - Click Network and Internet Connections.
  - Click Network Connections.
  - Right-click the connection that you use to connect to the Internet. For example, if you connect to the Internet by using a modem, right-click the connection that you want under Dial-up /other network available.
  - Click Properties.
  - Click the Advanced tab.
  - Under Internet Connection Sharing, select the Allow other network users to connect through this computer's Internet connection check box.
  - If you are sharing a dial-up Internet connection, select the Establish a dial up connection whenever a computer on my network attempts to access the Internet check box if you
    - want to permit your computer to automatically connect to the Internet.
  - Click OK. You receive the following message:
    - When Internet Connection Sharing is enabled, your LAN adapter will be set to use IP address
  - 192.168.0. 1. Your computer may lose connectivity with other computers on your network. If these other computers have static IP addresses, it is a good idea to set them to obtain their IP addresses automatically. Are you sure you want to enable Internet Connection Sharing?
    - Click Yes.
  - The connection to the Internet is shared to other computers on the local area network (LAN).
  - The network adapter that is connected to the LAN is configured with a static IP address of
  - 192.168.0. 1 and a subnet mask of 255.255.255.0



### On the client computer

- To connect to the Internet by using the shared connection, you must confirm the LAN adapter IP configuration, and then configure the client computer. To confirm the LAN adapter IP configuration, follow these steps:
- Log on to the client computer as Administrator or as Owner.
- Click Start, and then click Control Panel.
- Click Network and Internet Connections.
- Click Network Connections.
- Right-click Local Area Connection and then click Properties.
- Click the General tab, click Internet Protocol (TCP/IP) in the connection uses the following items list, and then click Properties.
- In the Internet Protocol (TCP/IP) Properties dialog box, click Obtain an IP address automatically (if it is not already selected), and then click OK. · Note: You can also assign a unique static IP address in the range of 192.168.0.2 to 254.
- For example, you can assign the following static IP address, subnet mask, and default gateway:
- IP Address 192.168.31.202
- Subnet mask 255.255.255.0
- Default gateway 192.168.31.1
- In the Local Area Connection Properties dialog box, click OK.
- Quit Control Panel.

### Implementation:



**Conclusion:** So, in this way we studied about the LAN (local area network).

## EXPERIMENT NO. 14

Implementation of Topology in Cisco Packet Tracer.

### Introduction to Topology :

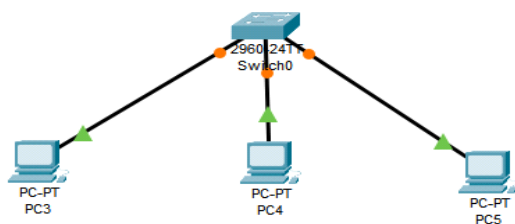
Network topology is the way devices are connected in a network. It defines how these components are connected and how data transfer between the network. Understanding the different types of network topologies can help in choosing the right design for a specific network. There are two major categories of Network Topology i.e. Physical Network topology and Logical Network Topology. Physical Network Topology refers to the actual structure of the physical medium for the transmission of data. Logical network Topology refers to the transmission of data between devices present in the network irrespective of the way devices are connected. The structure of the network is important for the proper functioning of the network. one must choose the most suitable topology as per their requirement.

### Types of Network Topology :

1. Mesh Topology
2. Star Topology
3. Bus Topology
4. Ring Topology
5. Tree Topology
6. Hybrid Topology

### Star Topology :

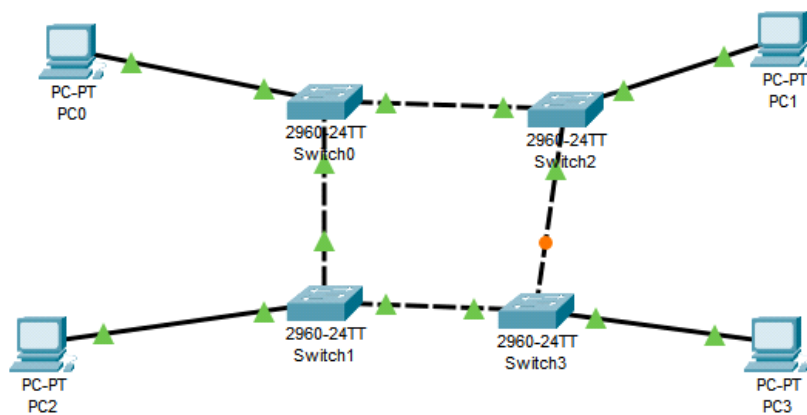
In Star Topology, all the devices are connected to a single hub through a cable. This hub is the central node and all other nodes are connected to the central node. The hub can be passive in nature i.e., not an intelligent hub such as broadcasting devices, at the same time the hub can be intelligent known as an active hub. Active hubs have repeaters in them. Coaxial cables or RJ-45 cables are used to connect the computers. In Star Topology, many popular Ethernet LAN protocols are used as CD(Collision Detection), CSMA (Carrier Sense Multiple Access), etc.



### Ring Topology :

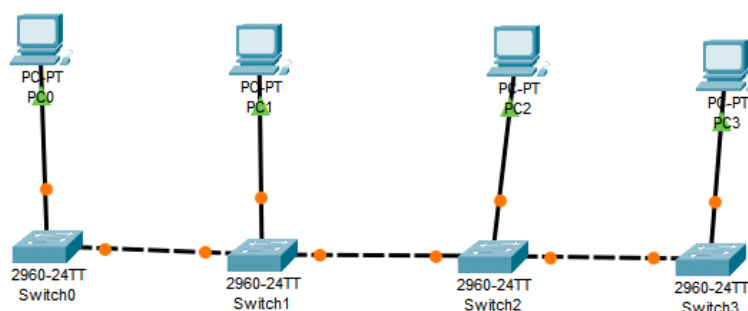
## Computer Network

In a Ring Topology, it forms a ring connecting devices with 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 data flows in one direction, i.e. it is unidirectional, but it can be made bidirectional by having 2 connections between each Network Node, it is called Dual Ring Topology. In-Ring Topology, the Token Ring Passing protocol is used by the workstations to transmit the data.



### Bus Topology :

Bus Topology is a network type in which every computer and network device is connected to a single cable. It is bi-directional. It is a multi-point connection and a non-robust topology because if the backbone fails the topology crashes. In Bus Topology, various MAC (Media Access Control) protocols are followed by LAN ethernet connections like TDMA, Pure Aloha, CDMA, Slotted Aloha, etc.



### Conclusion :

We implement star, ring and bus topology using cisco packet tracer.

