**DHRUV SINGHAL**

**12012055**

**CSB5**

**CN FILE**

**SUBMITTED TO - VIJAY VERMA SIR**

**Expriment-1**

**Q. Study the History of the internet.**

**THE ORIGINS OF THE INTERNET**

The origins of the internet are rooted in the USA of the 1950s. The Cold War was at its height and huge tensions existed between North America and the Soviet Union. Both superpowers were in possession of deadly nuclear weapons, and people lived in fear of long-range surprise attacks. The US realized it needed a communications system that could not be affected by a Soviet nuclear attack.

These machines were powerful but limited in numbers, and researchers grew increasingly frustrated: they required access to the technology, but had to travel great distances to use it.

To solve this problem, researchers started ‘time-sharing’. This meant that users could simultaneously access a mainframe computer through a series of terminals, although individually they had only a fraction of the computer’s actual power at their command.

The difficulty of using such systems led various scientists, engineers and organizations to research the possibility of a large-scale computer network.

## THE FIRST USE OF A COMPUTER NETWORK

In 1965, Lawrence Roberts made two separate computers in different places ‘talk’ to each other for the first time. This experimental link used a telephone line with an acoustically coupled modem, and transferred digital data using packets.

When the first packet-switching network was developed, Leonard Kleinrock was the first person to use it to send a message. He used a computer at UCLA to send a message to a computer at Stanford. Kleinrock tried to type ‘login’ but the system crashed after the letters ‘L’ and ‘O’ had appeared on the Stanford monitor.

A second attempt proved successful and more messages were exchanged between the two sites. The ARPANET was born.

## WHAT IS PACKET SWITCHING?

‘Packet switching’ is a method of splitting and sending data. A computer file is effectively broken up into thousands of small segments called ‘packets’—each typically around 1500 bytes—distributed across a network, and then reordered back into a single file at their destination. The packet switching method is very reliable and allows data to be sent securely, even over damaged networks; it also uses bandwidth very efficiently and doesn’t need a single dedicated link, like a telephone call does.

The world’s first packet-switching computer network was produced in 1969. Computers at four American universities were connected using separate minicomputers known as ‘Interface Message Processors’ or ‘IMPs’. The IMPs acted as gateways for the packets and have since evolved into what we now call ‘routers’.

Packet switching is the basis on which the internet still works today.

## WHAT IS TCP/IP?

TCP/IP stands for Transmission Control Protocol/Internet Protocol. The term is used to describe a set of protocols that govern how data moves through a network.

After the creation of ARPANET, more networks of computers began to join the network, and the need arose for an agreed set of rules for handling data. In 1974 two American computer scientists, Bob Kahn and Vint Cerf, proposed a new method that involved sending data packets in a digital envelope or ‘datagram’. The address on the datagram can be read by any computer, but only the final host machine can open the envelope and read the message inside.

Kahn and Cerf called this method transmission-control protocol (TCP). TCP allowed computers to speak the same language, and it helped the ARPANET to grow into a global interconnected network of networks, an example of ‘internetworking’—internet for short.

IP stands for Internet Protocol and, when combined with TCP, helps internet traffic find its destination. Every device connected to the internet is given a unique IP number. Known as an IP address, the number can be used to find the location of any internet-connected device in the world.

## WHAT IS DNS?

DNS stands for Domain Name System. It is the internet’s equivalent of a phone book, and converts hard-to-remember IP addresses into simple names.

In the early 1980s, cheaper technology and the appearance of desktop computers allowed the rapid development of local area networks (LANs). An increase in the amount of computers on the network made it difficult to keep track of all the different IP addresses.

This problem was solved by the introduction of the Domain Name System (DNS) in 1983. DNS was invented by Paul Mockapetris and Jon Postel at the University of Southern California. It was one of the innovations that paved the way for the World Wide Web.

Q. Study the different types of network cables.

To connect two or more computers or networking devices in a network, network cables are used. There are three types of network cables; coaxial, twisted-pair, and fiber-optic.

## Coaxial cable

This cable contains a conductor, insulator, braiding, and sheath. The sheath covers the braiding, the braiding covers the insulation, and the insulation covers the conductor.

The following image shows these components.

At the beginning of computer networking, when there were no dedicated media cables available for computer networks, network administrators began using coaxial cables to build computer networks.

Because of its low cost and long durability, coaxial cables were used in computer networking for nearly two decades (the 80s and 90s). Coaxial cables are no longer used to build any type of computer network.

### **Specifications of coaxial cables**

Coaxial cables have been in use for the last four decades. During these years, based on several factors such as the thickness of the sheath, the metal of the conductor, and the material used in insulation, hundreds of specifications have been created to specify the characteristics of coaxial cables.

From these specifications, only a few were used in computer networks. The following table lists them.

* Coaxial cable uses RG rating to measure the materials used in shielding and conducting cores.
* RG stands for the Radio Guide. Coaxial cable mainly uses radio frequencies in transmission.
* Impedance is the resistance that controls the signals. It is expressed in the ohms.
* AWG stands for American Wire Gauge. It is used to measure the size of the core. The larger the AWG size, the smaller the diameter of the core wire.

## Twisted-pair cables

The twisted-pair cable was primarily developed for computer networks. This cable is also known as **Ethernet cable**. Almost all modern LAN computer networks use this cable.

This cable consists of color-coded pairs of insulated copper wires. Every two wires are twisted around each other to form pair. Usually, there are four pairs. Each pair has one solid color and one stripped color wire. Solid colors are blue, brown, green, and orange. In stripped color, the solid color is mixed with the white color.

Based on how pairs are stripped in the plastic sheath, there are two types of twisted-pair cable; UTP and STP.

In the **UTP (*Unshielded twisted-pair*) cable**, all pairs are wrapped in a single plastic sheath.

In the **STP (*Shielded twisted-pair*) cable**, each pair is wrapped with an additional metal shield, then all pairs are wrapped in a single outer plastic sheath.

### **Similarities and differences between STP and UTP cables**

* Both STP and UTP can transmit data at 10Mbps, 100Mbps, 1Gbps, and 10Gbps.
* Since the STP cable contains more materials, it is more expensive than the UTP cable.
* Both cables use the same RJ-45 (registered jack) modular connectors.
* The STP provides more noise and EMI resistance than the UTP cable.
* The maximum segment length for both cables is 100 meters or 328 feet.
* Both cables can accommodate a maximum of 1024 nodes in each segment.

Fiber optic cable

This cable consists of a core, cladding, buffer, and jacket. The core is made from thin strands of glass or plastic that can carry data over a long distance. The core is wrapped in the cladding; the cladding is wrapped in the buffer, and the buffer is wrapped in the jacket.

* Core carries the data signals in the form of light.
* Cladding reflects light back to the core.
* Buffer protects the light from leaking.
* The jacket protects the cable from physical damage.

Fiber optic cable is completely immune to EMI and RFI. This cable can transmit data over a long distance at the highest speed. It can transmit data up to 40 kilometers at the speed of 100Gbps.

Fiber optic uses light to send data. It reflects light from one endpoint to another. Based on how many beams of light are transmitted at a given time, there are two types of fiber optical cable; SMF and MMF.

### **SMF (Single-mode fiber) optical cable**

This cable carries only a single beam of light. This is more reliable and supports much higher bandwidth and longer distances than the MMF cable. This cable uses a laser as the light source and transmits 1300 or 1550 nano-meter wavelengths of light.

### **MMF (multi-mode fiber) optical cable**

This cable carries multiple beams of light. Because of multiple beams, this cable carries much more data than the SMF cable. This cable is used for shorter distances. This cable uses an LED as the light source and transmits 850 or 1300 nano-meter wavelengths of light.

That’s all for this tutorial. If you like this tutorial, please share it with friends via your favorite social networking sites and subscribe to our YouTube channel.

**Expriment-2**

**1. Study various network devices Switch, Hub, gateway, Bridge, Repeaters, Routers.**

Network devices or networking hardware are the physical devices that are used for establishing connections and facilitating interaction between different devices in a computer network.

**Switch**

Switches may operate at one or more layers of the OSI model. They may operate in the data link layer and network layer; a device that operates simultaneously at more than one of these layers is known as a *multilayer switch*.

A Switch can check the errors before forwarding the data, which makes it more efficient and improves its performance. A switch is the better version of a hub. It is a multi-port bridge device.

**Hub**

Hubs work in the physical layer of the OSI model. A hub is a device for connecting multiple Ethernet devices and making them act as a single network segment. It has multiple inputs and output ports in which a signal introduced at the input of any port appears at the output of every port except the original incoming port.

A hub can be used with both digital and analog data. Hubs do not perform packet filtering or addressing function, they send the data packets to all the connected devices.

**Types of Hub −**

* Active Hub
* Passive Hub
* Intelligent Hub

**Repeater**

A repeater operates at the physical layer of the OSI model.

* A Repeater connects two segments of a network cable.
* Sometimes it regenerates the signals to proper amplitudes and sends them to the other segment.
* If the signal becomes weak, it can copy the signal bit by bit and regenerate it at the original strength.
* It is a 2-port device.

## Bridge

A bridge operates at the data link layer of the OSI model. It can read only the outmost hardware address of the packet but cannot read the IP address. It reads the outmost section of the data packet to tell where the message is going. It reduces the traffic on other network segments. It does not send all the packets. So, a bridge can be programmed to reject packets from a particular network.

**Router**

Routers are small physical devices that operate at the network layer to join multiple networks together.

* A router is a device like a switch that routes data packets based on their IP addresses.
* Routers normally connect LANs and WANs and have a dynamically updating routing table based on which they make decisions on routing the data packets.
* A Router divides the broadcast domains of hosts connected through it.
* Routers perform the traffic directing functions on the Internet. A data packet is typically forwarded from one router to another through the networks that constitute the internetwork until it reaches its destination code.
* Routers may also be used to connect two or more logical groups of computer devices known as subnets, each with a different subnetwork address. The subnet addresses recorded in a router do not necessarily map directly to the physical interface connections.

**Two types of routers −**

* **Static routers** – Static routers are configured manually and route data packets based on the information in a router table.
* **Dynamic routers** – Dynamic routers use adaptive routing which is a process where a router can forward data by a different route.

**Gateway**

A gateway is an internetworking capable of joining together two networks that use different base protocols.

A network gateway can be implemented completely in software, hardware, or a combination of both, depending on the types of protocols they support.

A network gateway can operate at any level of the OSI model. A broadband router typically serves as the network gateway, although ordinary computers can also be configured to perform equivalent functions.

* A gateway is a router or proxy server that routes between networks.
* A gateway belongs to the same subnet to which the PC belongs.

**2. Study different types of addresses- IP-address, port-address, and physical.**

Four levels of addresses are used in the TCP/IP protocol: **physical address, logical address, port address, and application-specific address** as shown in Figure.

**Physical Addresses**

* The physical address, also known as the link address, is the address of a node as defined by its LAN or WAN.
* The size and format of these addresses vary depending on the network. For example, Ethernet uses a 6-byte (48-bit) physical address.
* Physical addresses can be either unicast (one single recipient), multicast (a group of recipients), or broadcast (to be received by all systems in the network.
* Example: Most local area networks use a 48-bit (6-byte) physical address written as 12 hexadecimal digits; every byte (2 hexadecimal digits) is separated by a colon, as shown below: A 6-byte (12 hexadecimal digits) physical address **07:01:02:01:2C:4B**

**Logical Addresses**

* Logical addresses are used by networking software to allow packets to be independent of the physical connection of the network, that is, to work with different network topologies and types of media.
* A logical address in the Internet is currently a 32-bit address that can uniquely define a host connected to the Internet. An internet address in IPv4 in decimal numbers **132.24.75.9**
* No two publicly addressed and visible hosts on the Internet can have the same IP address.
* The physical addresses will change from hop to hop, but the logical addresses remain the same.
* The logical addresses can be either unicast (one single recipient), multicast (a group of recipients), or broadcast (all systems in the network). There are limitations on broadcast addresses.

**Port Addresses**

* There are many application running on the computer. Each application run with a port no.(logically) on the computer.
* A port number is part of the addressing information used to identify the senders and receivers of messages.
* Port numbers are most commonly used with TCP/IP connections.
* These port numbers allow different applications on the same computer to share network resources simultaneously.
* The physical addresses change from hop to hop, but the logical and port addresses usually remain the same.
* Example: a port address is a 16-bit address represented by one decimal number **753**

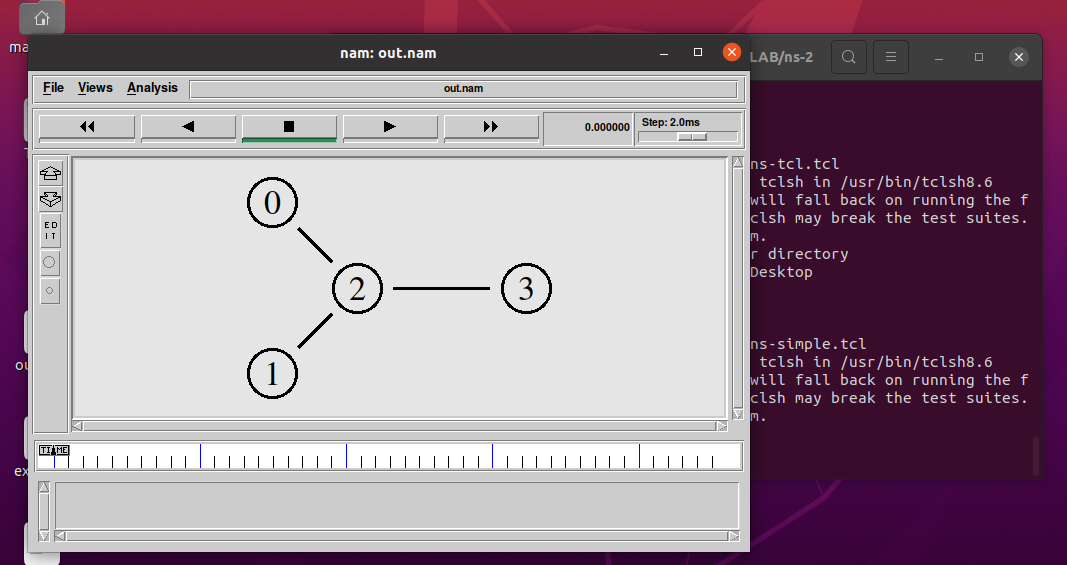
**Application-Specific Addresses**

* Some applications have user-friendly addresses that are designed for that specific application.
* Examples include the e-mail address (for example, forouzan@fhda.edu) and the Universal Resource Locator (URL) (for example, www.mhhe.com). The first defines the recipient of an e-mail; the second is used to find a document on the World Wide Web.

**Lab-6 (ns2 Installation)**

**Q. Execute a simple program of tcl script to show that ns2 is working.**

Execution of ns-simple.tcl file:



**Lab-7 ( ns2 Expriment2)**

**1. Create the following scenario with two nodes n0 and n1 and link in between.**

**– Sender agent: Agent/UDP**

**– Receiver agent: Agent/Null**

**– Connect agents**

**– Data source: Application/Traffic/CBR**

**– Run from 0.5 to 4.5 sec, finish at 5.0 sec**

**Code:**

#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]

#Create links between the nodes

$ns duplex-link $n0 $n1 2Mb 10ms DropTail

#Set Queue Size of link (n0-n1) to 10

$ns queue-limit $n0 $n1 10

#Give node position (for NAM)

$ns duplex-link-op $n0 $n1 orient right

#Monitor the queue for link (n0-n1). (for NAM)

$ns duplex-link-op $n0 $n1 queuePos 0.5

#Setup a UDP connection

set udp [new Agent/UDP]

$ns attach-agent $n0 $udp

set null [new Agent/Null]

$ns attach-agent $n1 $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\_ 1.5mb

$cbr set random\_ false

#Schedule events for the CBR and FTP agents

$ns at 0.5 "$cbr start"

$ns at 4.5 "$cbr stop"

#Call the finish procedure after 5 seconds of simulation time

$ns at 5.0 "finish"

#Print CBR packet size and interval

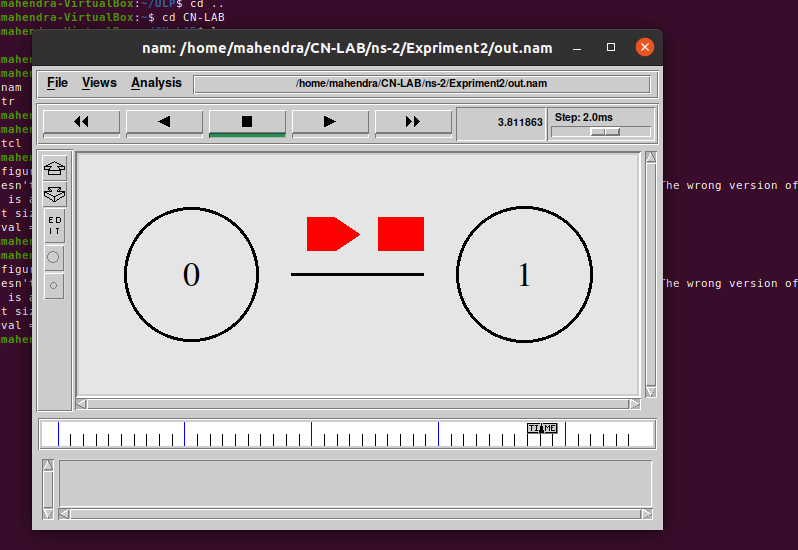
puts "CBR packet size = [$cbr set packet\_size\_]"

puts "CBR interval = [$cbr set interval\_]"

#Run the simulation

$ns run

**Output:**



**2. Create the following scenario and connect the appropriate agents**

**• Start the FTP application at t = 0.5s**

**• Start the CBR data source at t = 1s**

**• Terminate both at t = 4.5 s**

**• Visualize the bottle neck queue**

**Code:**

# 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

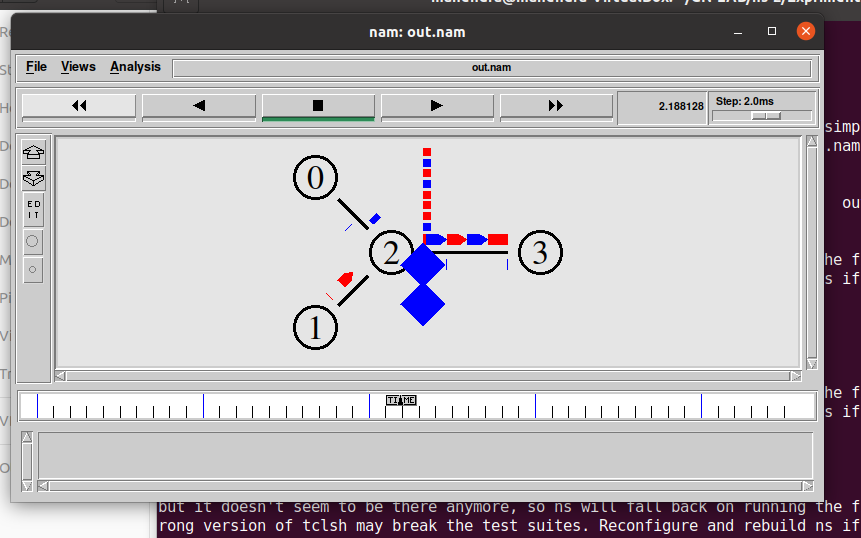
$ns at 5.0 "finish"

# Print CBR packet size and interval

puts "CBR packet size = [$cbr set packet\_size\_]"

puts "CBR interval = [$cbr set interval\_]"

**Output:**



**Implement following concepts using C++/Java.**

**1. Bit Stuffing**

**Code:**

#include<stdio.h>

#include<string.h>

int main()

{

    int a[20],b[30],i,j,k,count,n;

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

    scanf("%d",&n);

    printf("Enter the frame in the form of 0 and 1 :");

    for(i=0; i<n; i++)

        scanf("%d",&a[i]);

    i=0;

    count=1;

    j=0;

    while(i<n)

    {

        if(a[i]==1)

        {

            b[j]=a[i];

            for(k=i+1; a[k]==1 && k<n && count<5; k++)

            {

                j++;

                b[j]=a[k];

                count++;

                if(count==5)

                {

                    j++;

                    b[j]=0;

                }

                i=k;

            }

        }

        else

        {

            b[j]=a[i];

        }

        i++;

        j++;

    }

    printf("After Bit Stuffing :");

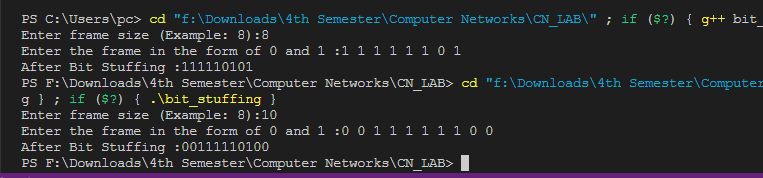
    for(i=0; i<j; i++)

        printf("%d",b[i]);

    return 0;

}

**Output:**



**2. CRC Error Correction:**

**Code:**

// A C++ program for cyclic redundency check

#include<iostream>

using namespace std;

class CRC

{

public:

int nf, ng, frame[20], gen[10], temp[20], b;

char a;

int\* divide(int n, int g, int temp[10], int gen[10]) {

for(int i=0; i<n; i++)

{

if(gen[0]==temp[i])

{

for(int j=0, k=i; j<g+1; j++, k++)

{

if(temp[k]^gen[j]==1)

temp[k]=1;

else

temp[k]=0;

}

}

}

return temp;

}

void input()

{

cout<<"Enter length of your frame:";

cin>>nf;

cout<<"Enter your frame:";

for(int i=0; i<nf; i++)

{

cin>>frame[i];

temp[i]=frame[i];

}

cout<<"Enter length of your generator:"; cin>>ng;

cout<<"Enter your generator:";

for(int i=0; i<ng; i++)

{

cin>>gen[i];

}

ng--;

for(int i=0; i<ng; i++)

{

temp[nf+i]=0;

}

}

void sender\_side()

{

int\* sender;

sender = divide(nf,ng,temp,gen);

cout<<endl<<"-----Senders Side-----\n"<<"CRC:";

for(int i=0; i<ng; i++)

{

frame[nf+i]=sender[nf+i];

cout<<sender[nf+i]<<' ';

}

cout<<endl<<"Transmitted frame:";

for(int i=0; i<nf+ng; i++)

cout<<frame[i]<<' ';

cout<<endl;

}

int receiver\_side()

{

int\* receiver;

cout<<"\n-----Receivers Side-----\n"<<"Received message:";

for(int i=0; i<nf+ng; i++)

cout<<frame[i]<<' ';

cout<<endl;

cout<<"Enter which bit you want to change(from 0-"<<nf+ng<<")-";

cin>>b;

if(frame[b]==1)

frame[b]=0;

else

frame[b]=1;

receiver = divide(nf,ng,frame,gen);

cout<<"Error:";

for(int i=0; i<nf+ng; i++)

{

if(receiver[i]!=0)

{

cout<<"Error Detected!!"<<endl;

return 0;

}

}

cout<<"No error detected!"<<endl;

}

};

int main()

{

CRC o;

o.input();

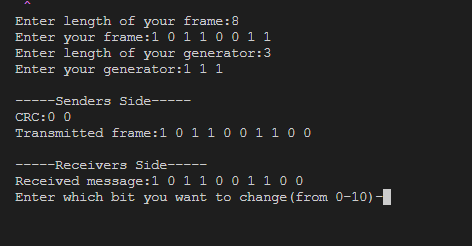
o.sender\_side();

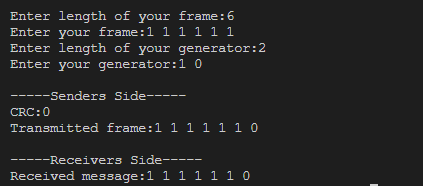
o.receiver\_side();

return 0;

}

**Output:**





**Lab-8**

**Expriment-2(ns-2)**

**3. Write a TCL script to simulate a file transfer with using ns 2:**

Consider a client and a server. The server is running a FTP application (over TCP). The client sends a request to download a file of size 10 MB from the server. Write a script to simulate this scenario. Let node #0 be the server and node #1 be the client. TCP packet size is 1500 B. Assume typical values for other parameters.

**Code:**

#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 3.nam w]

$ns namtrace-all $nf

#Define a 'finish' procedure

set outfile [open "bytesReceived.xg" w]

# procedure to plot the bytesReceived window

proc plotWindow {tcpSource outfile} {

global ns

set now [$ns now]

set cwnd [$tcpSource set cwnd\_]

# the data is recorded in a file called bytesReceived.xg (this can be plotted # using xgraph or gnuplot. this example uses xgraph to plot the cwnd\_

puts $outfile "$now $cwnd"

$ns at [expr $now+0.1] "plotWindow $tcpSource $outfile"

}

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 &

exec xgraph bytesReceived.xg -geometry 500x500 &

exit 0

}

#Create two nodes

set n0 [$ns node]

set n1 [$ns node]

#Create links between the nodes

$ns duplex-link $n0 $n1 10Mb 10ms DropTail

#Setup a TCP connection $n0->$n1

set tcp [new Agent/TCP]

$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink]

$ns attach-agent $n1 $sink

$ns connect $tcp $sink

$tcp set fid\_ 1

set ftp [new Application/FTP]

$ftp attach-agent $tcp

$ftp set type\_ FTP

#schedule

$ns at 0.0 "plotWindow $tcp $outfile"

$ns at 1.0 "$ftp start"

$ns at 4.0 "$ftp stop"

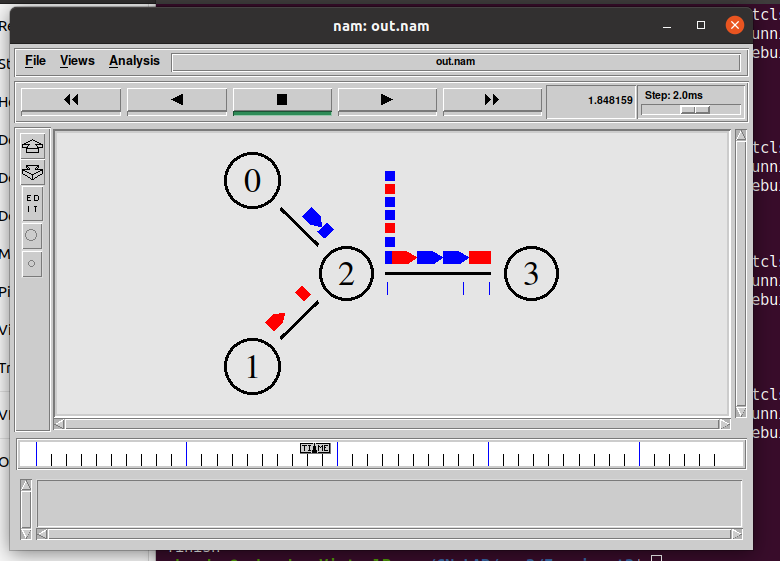
#Call the finish procedure after 5 seconds of simulation time

$ns at 6.0 "finish"

#Run the simulation

$ns run

**Output:**



**4. Write a TCL script to simulate the network described below using ns2:**

Consider a small network with five nodes n0, n1, n2, n3, n4, forming a star topology. The node n4 is at the center. Node n0 is a TCP source, which transmits packets to node n3 (a TCP sink) through the node n4. Node n1 is another traffic source, and sends UDP packets to node n2 through n4. The duration of the simulation time is 10 seconds.

**Code:**

set ns [new Simulator]

set namfile [open 4.nam w]

$ns namtrace-all $namfile

set tracefile [open 4.tr w]

$ns trace-all $tracefile

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

$ns duplex-link $n0 $n4 1Mb 10ms DropTail

$ns duplex-link $n1 $n4 1Mb 10ms DropTail

$ns duplex-link $n4 $n3 1Mb 10ms DropTail

$ns duplex-link $n4 $n2 1Mb 10ms DropTail

set tcp [new Agent/TCP]

$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink]

$ns attach-agent $n3 $sink

$ns connect $tcp $sink

set ftp [new Application/FTP]

$ftp attach-agent $tcp

set udp [new Agent/UDP]

$ns attach-agent $n1 $udp

set null [new Agent/Null]

$ns attach-agent $n2 $null

$ns connect $udp $null

$udp set class\_ 1

$ns color 1 Blue

$tcp set class\_ 2

$ns color 2 Red

set cbr [new Application/Traffic/CBR]

$cbr set packetsize\_ 500

$cbr set interval\_ 0.005

$cbr attach-agent $udp

$ns at 0.0 "$cbr start"

$ns at 0.0 "$ftp start"

$ns at 9.0 "$cbr stop"

$ns at 9.0 "$ftp stop"

proc finish {} {

global ns namfile tracefile

$ns flush-trace

close $namfile

close $tracefile

exec nam 4.nam &

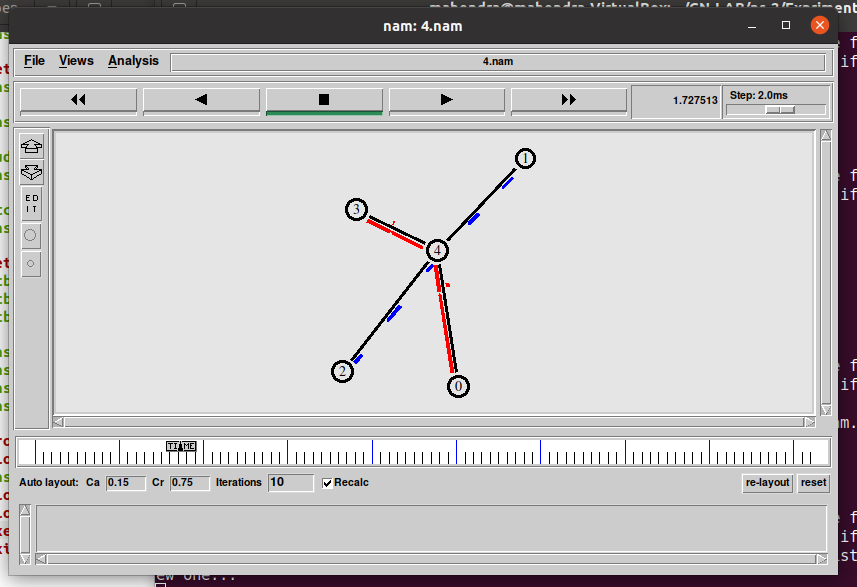
exit 0

}

$ns at 10.0 "finish"

$ns run

**Output:**



**1. Go Back N Protocol Implementation:**

**Code:**

#include<iostream>

#include<ctime>

#include<cstdlib>

using namespace std;

int main()

{

 int nf,N;//nf is number of frames and N is window size

 int no\_tr=0;

 srand(time(NULL));

 cout<<"Enter the number of frames : ";

 cin>>nf;// no. of frames to transfer

 cout<<"Enter the Window Size : ";

 cin>>N;//window size

 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;

             }

     }

     cout<<endl;

     i+=x;

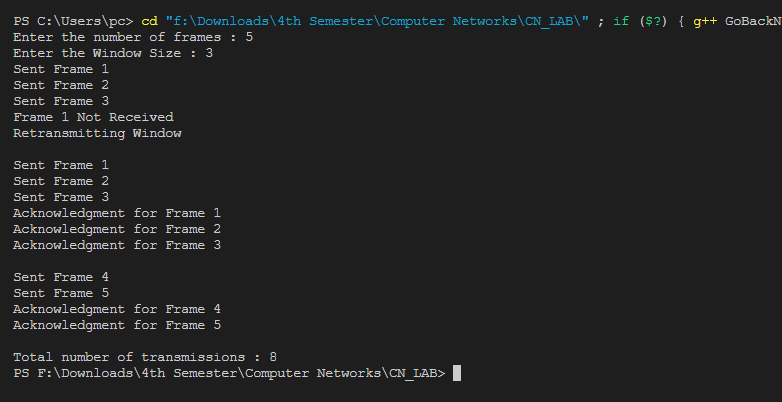
 }

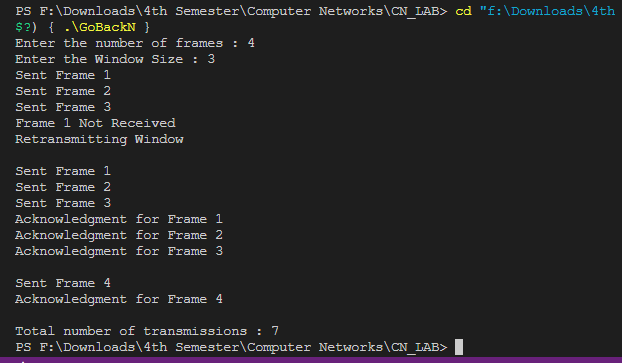
 cout<<"Total number of transmissions : "<<no\_tr<<endl;

 return 0;

}

**Output:**





**2. Selective Repeat ARQ Protocol Implementation:**

**Code:**

// A C++ program to implement selective repeat ARQ

#include<iostream>

using namespace std;

int main()

{

    //variables are: w is for window size

    //f is the number of frames to send

    int w,i,f,frames[50];

    cout<<"Enter window size: ";

    cin>>w;

    cout<<"Enter number of frames to transmit: ";

    cin>>f;

    cout<<"\nEnter data of "<<f<<" frames: ";

    for(i=1;i<=f;i++)

        cin>>frames[i];

    cout<<"\nWith sliding window protocol the frames will be sent in the following manner (assuming no corruption of frames)\n\n";

    cout<<"After sending "<<w<<" frames at each stage sender waits for acknowledgement sent by the receiver\n\n";

    for(i=1;i<=f;i++)

    {

        if(i%w==0)

        {

            cout<<frames[i]<<"\n";

            cout<<"Acknowledgement of above frames sent is received by sender\n\n";

        }

        else

            cout<<frames[i]<<" ";

    }

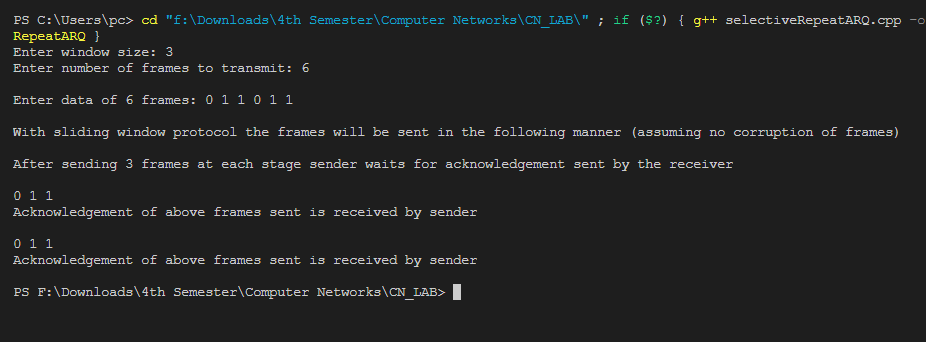
    if(f%w!=0)

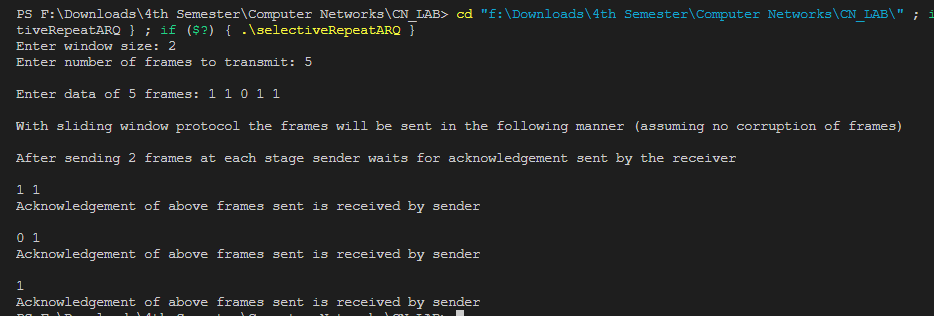
        cout<<"\nAcknowledgement of above frames sent is received by sender\n";

    return 0;

}

**Output:**





**Lab-9**

**(ns2 Expriment-3)**

**1. Setting up a local area network with ns2:**

Consider the LAN with seven nodes to be an isolated one i.e. not connected to the Internet.

Node # 0 in the LAN act as a UDP traffic source, and node # 6 is the destination node.

Assume CBR traffic to be flowing between the nodes. The simulation lasts for 25 seconds.

In Ethernet a packet is broadcasted in the shared medium, and only the destination node

accepts the packet. Other nodes simply drop it. How many hops a packet should take to

travel from node # 0 to node # 6? Verify this from the "Hop Count" plot.

**Source Code:**

# Set a simulator object

set ns [new Simulator]

#Open the NAM trace file

set nf [open out.nam w]

$ns namtrace-all $nf

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 lan [$ns newLan "$n0 $n1 $n2 $n3 $n4 $n5 $n6" 100Mb 0.5ms LL Queue/DropTail Mac/802\_3 Channel Phy/WiredPhy]

set udp [new Agent/UDP]

$ns attach-agent $n0 $udp

set null [new Agent/Null]

$ns attach-agent $n6 $null

$ns connect $udp $null

set cbr [new Application/Traffic/CBR];

$cbr set packetSize\_ 500

$cbr set interval\_ 0.005

$cbr attach-agent $udp

$ns at 0.0 "$cbr start"

$ns at 9.0 "$cbr stop"

#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 &

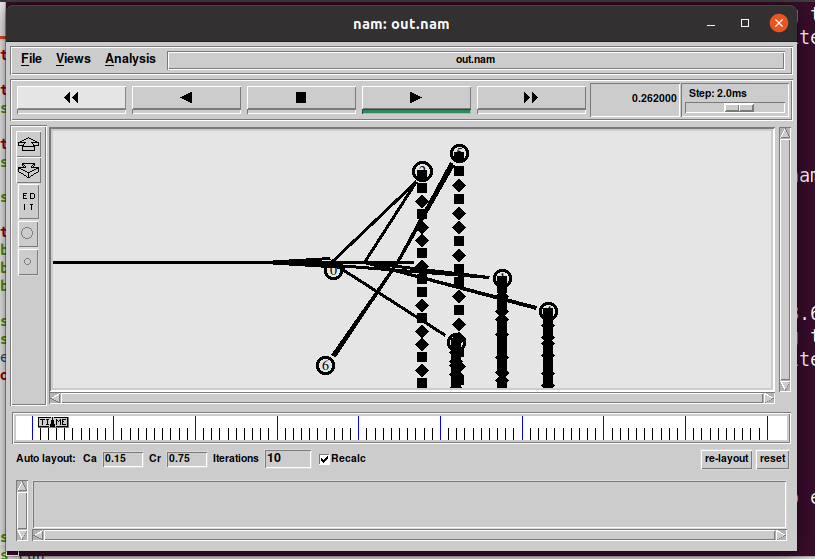
exit 0

}

$ns at 25 "finish"

$ns run

**Output:**



**2. Create the following scenario with two nodes and link in between.**

**– Sender agent: Agent/UDP**

**– Receiver agent: Agent/Null**

**– Connect agents**

**– Data source: Application/Traffic/CBR**

**– Run from 0.5 to 4.5 sec, finish at 5.0 sec**

**Source Code:**

#Create a simulator object

set ns [new Simulator]

#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 trace file

exec nam out.nam &

exit 0

}

# Create two nodes

set n0 [$ns node]

set n1 [$ns node]

#Create a duplex link between the nodes

$ns duplex-link $n0 $n1 1Mb 10ms DropTail

#Call the finish procedure after 5 seconds of simulation time

$ns at 5.0 "finish"

#Create a UDP agent and attach it to node n0

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 Null agent (a traffic sink) and

#attach it to node n1

set null0 [new Agent/Null]

$ns attach-agent $n1 $null0

#Connect the traffic source with the traffic sink

$ns connect $udp0 $null0

#Schedule events for the CBR agent

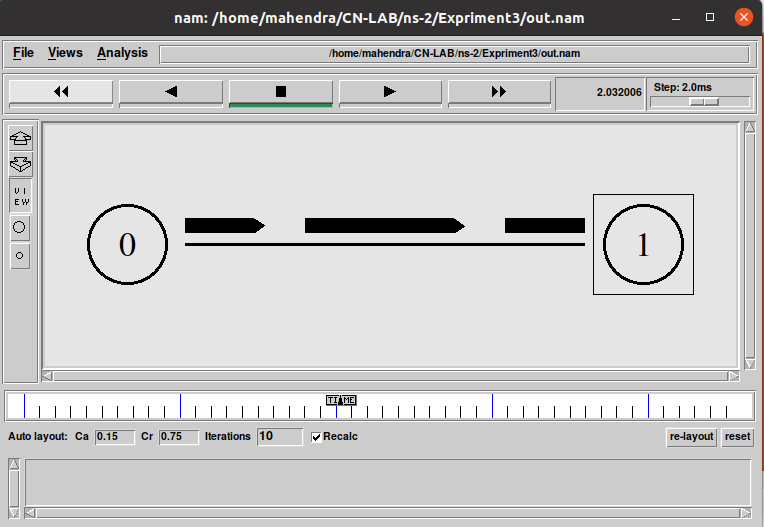
$ns at 0.5 "$cbr0 start"

$ns at 4.5 "$cbr0 stop"

#Run the simulation

$ns run

**Output:**



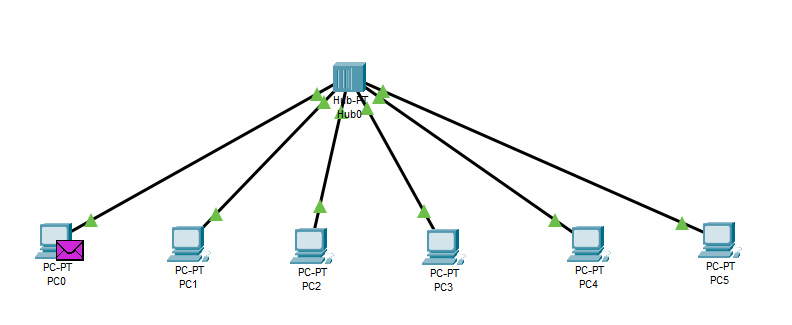
**Lab-10**

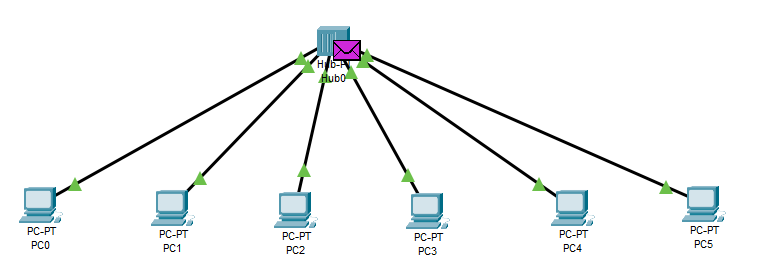
**(Cisco Packet Tracer)**

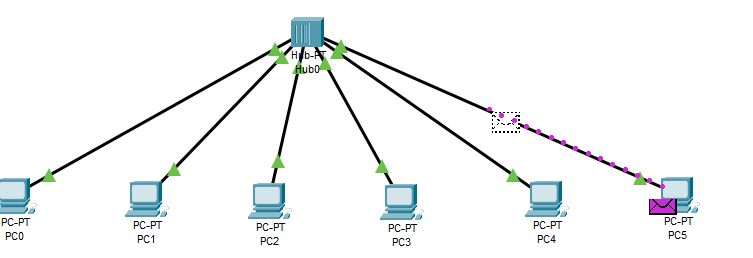
**I. Building a local area Network (LAN) using hub in Cisco Packet Tracer.**

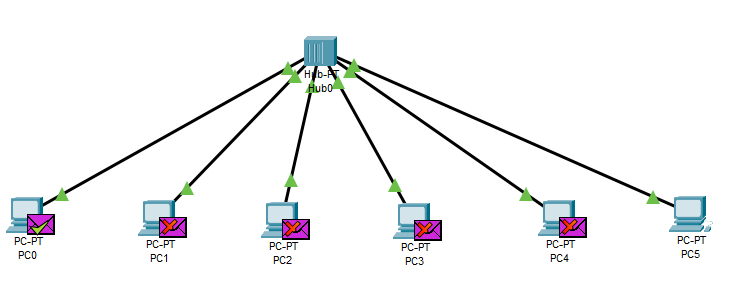
Hub is connecting device which is used to connect and stablish communication b/w nodes in a local area network. Hub has multiple port to connect multiple systems in a LAN.

Hub accepts data from one station and transfer it to the all connecting devices.







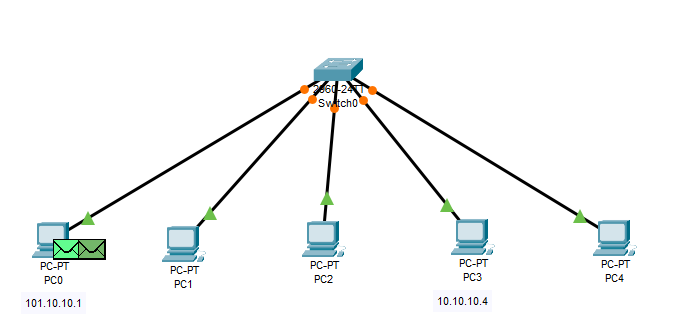


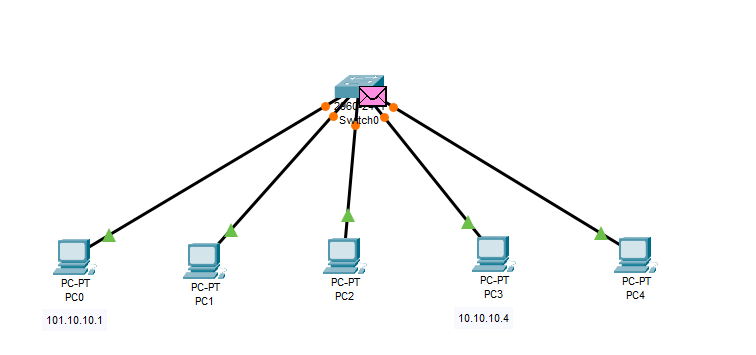
**II. Build a LAN in Cisco Packet Tracer using switch.**

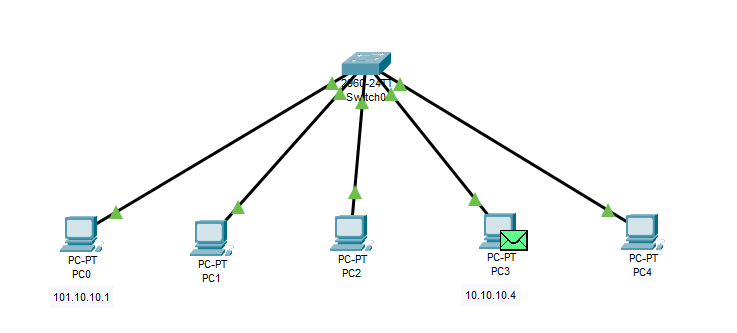
Switch is a networking device operating at layer 2 or data link layer.

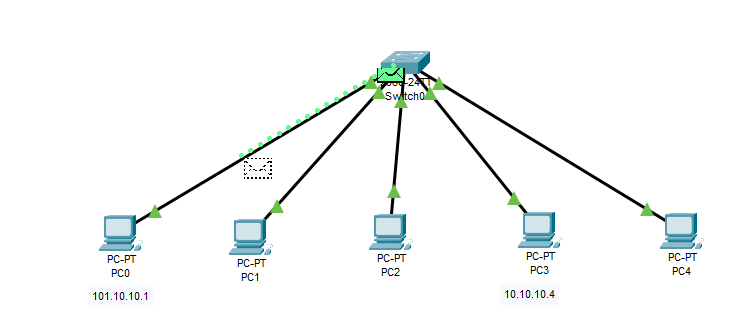
A switch has many ports from which computers are plugged in. A switch

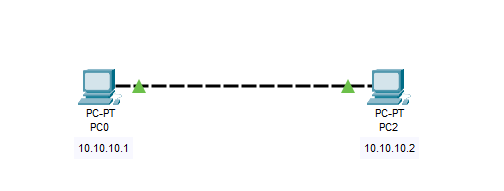
Has its own memory to store MAC addresses of systems in LAN. When it receives a message from a system then forward it to only intended system only.

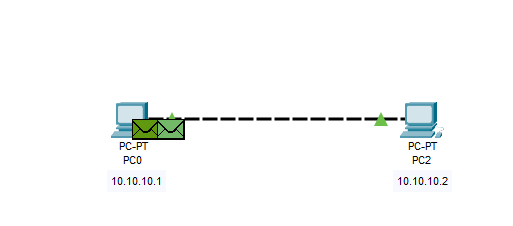


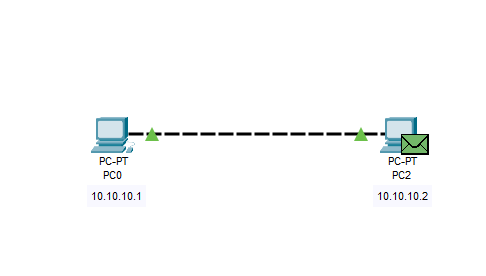


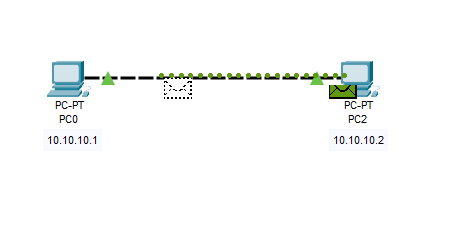




**III. Build a peer-to-peer network b/w two nodes in Cisco Packet Tracer.** 







**IV. Connect two local area networks using a repeater.**

A repeater is implemented in computer networks to expand the coverage area of the network, repropagate a weak or broken signal and or service remote nodes. Repeaters amplify the received/input signal to a higher frequency domain so that it is reusable, scalable and available.

