**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| Serial  No. | AIM | DATE OF SUBMISSION | REMARKS |
| 01 | To Implement RING Topology in Cisco Packet Tracer. |  |  |
| 02 | To Implement STAR Topology in Cisco Packet Tracer. |  |  |
| 03 | To Implement BUS Topology in Cisco Packet Tracer. |  |  |
| 04 | To Implement MESH Topology in Cisco Packet Tracer. |  |  |
| 05 | To Implement TREE Topology in Cisco Packet Tracer. |  |  |
| 06 | Write a program to add a parity bit to a 7-bit data input by a user and the sender's site. |  |  |
| 07 | Write a program to add redundant bit using Hamming code at the sender's side. |  |  |
| 08 | Write a program to detect and correct redundant bit using Hamming code at the receiver's side. |  |  |
| 09 | Write a program to determine the class of given IPV4 address in dotted decimal or binary notation. |  |  |
| 10 | Set up a two-computer network using CPT and explore Ipconfig/all and ping commands. |  |  |
| 11 | To Set a six-computer network with a switch using CPT and show Unicast. Multicast and Broadcast addressing. |  |  |
| 12 | To Implement Checksum Algorithm using C++ |  |  |
| 13 | To Implement Pure and Slotted Aloha using C++ |  |  |
| 14 | Subnetting Mask. |  |  |

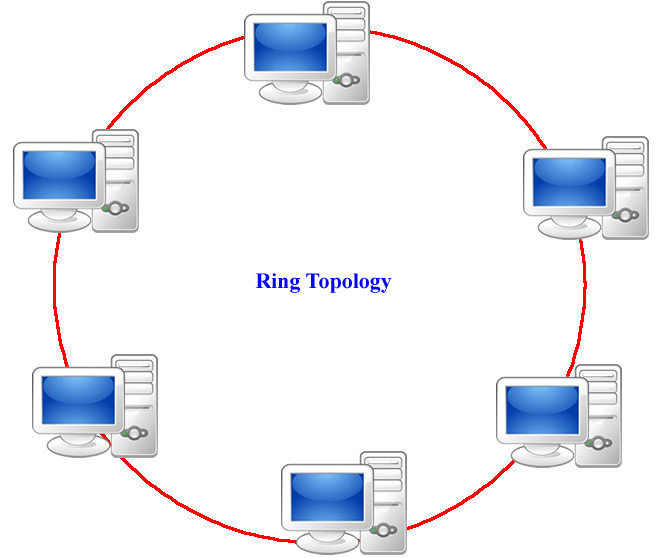
**PRACTICAL NO – 1**

**Aim:** To implement ring topology in cisco packet tracer.

**SOFTWARE REQURIED:** Cisco packet tracer network simulation tool.

**THEORY:** A ring topology is a [network](https://www.computerhope.com/jargon/n/network.htm) configuration where device connections create a circular [data](https://www.computerhope.com/jargon/d/data.htm) path. Each networked device is connected to two others, like points on a circle. Together, devices in a ring topology are called a **ring network.**

In a ring network, [packets](https://www.computerhope.com/jargon/p/packet.htm) of data travel from one device to the next until they reach their destination. Most ring topologies allow packets to travel only in one direction, called a **unidirectional** ring network. Others permit data to move in either direction, called **bidirectional**.



**Advantages of a ring topology**

* All data flows in one direction, reducing the chance of packet collisions.
* A network server is not needed to control network connectivity between each workstation.
* Data can transfer between workstations at high speeds.
* Additional workstations can be added without impacting performance of the network.

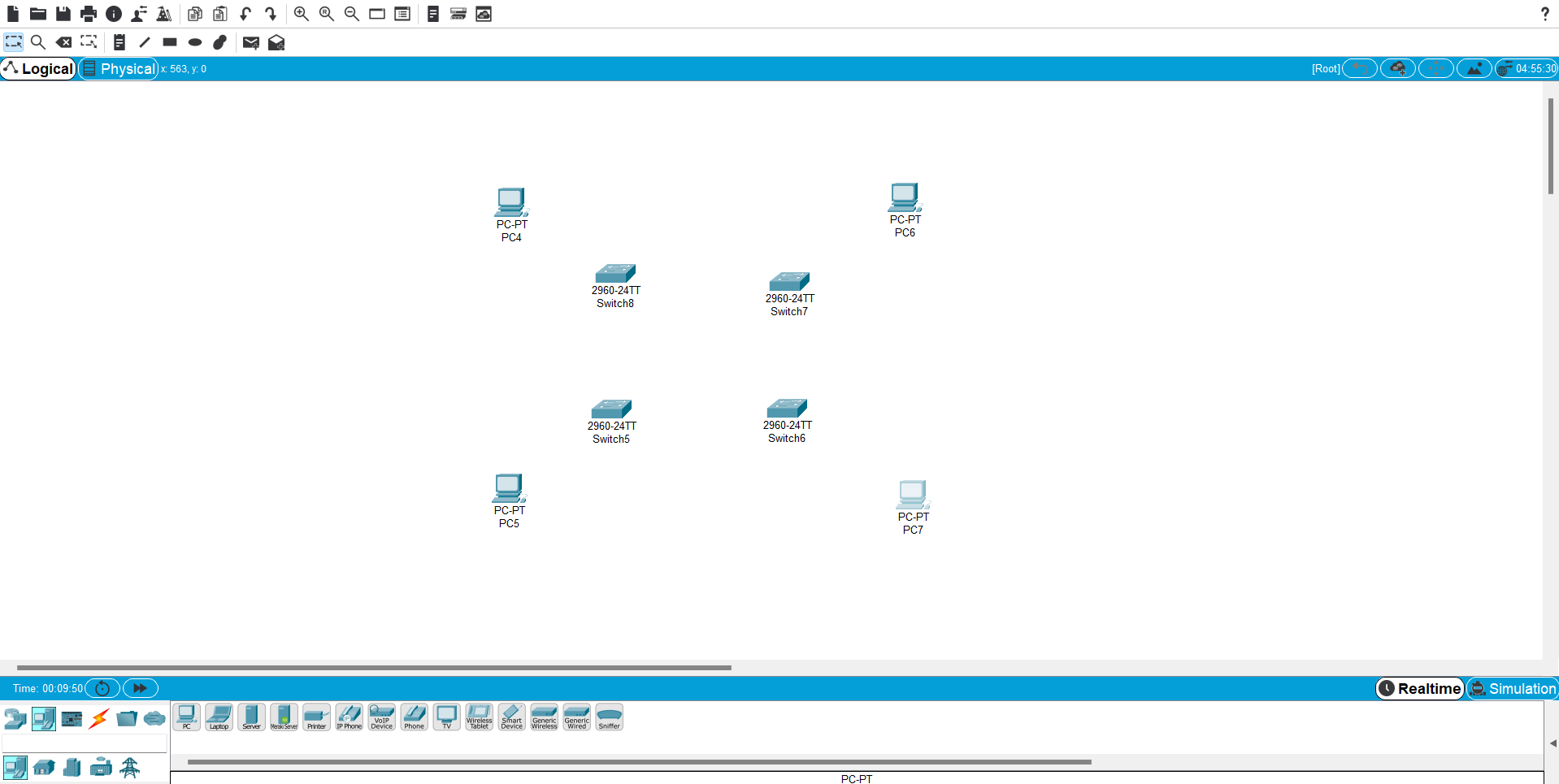
**Disadvantages of a ring topology**

* All data being transferred over the network must pass through each workstation on the network, which can make it slower than a [star topology](https://www.computerhope.com/jargon/s/startopo.htm).
* The entire network will be impacted if one workstation shuts down.
* The hardware needed to connect each workstation to the network is more expensive than Ethernet cards and hubs/switches.

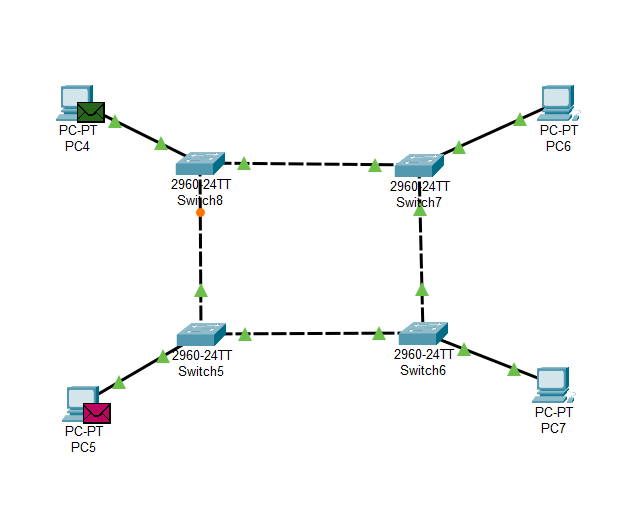
**PROCEDURE:**

Step 1: Go to device section

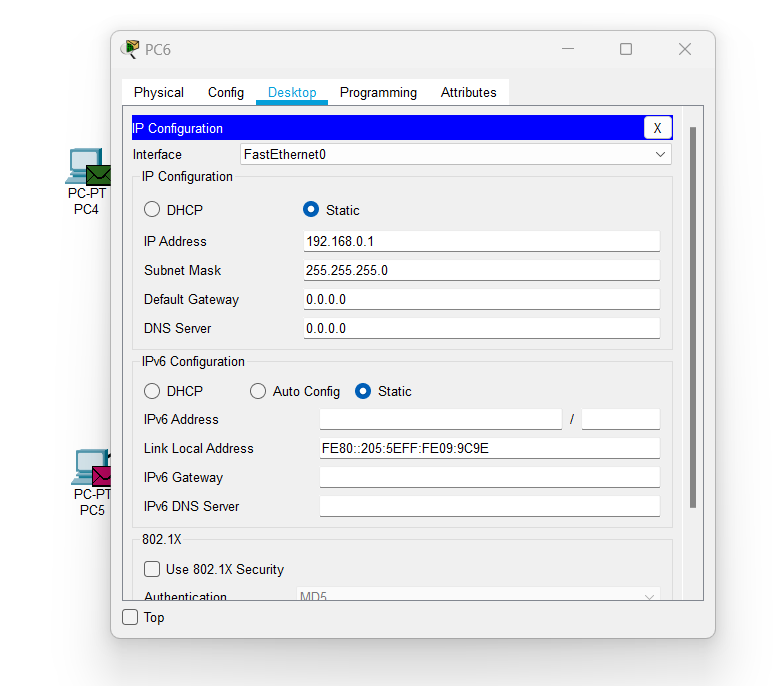
* 1. Select network devices
  2. Select switches
  3. Select and drag 4 2960 switches on the empty canvas
  4. Select end devices
  5. Select and drag 4 PCs on the canvas



Step 2: select copper cross - over and connects the switches as shown and select copper straight – through and connects PCs to switches as shown.



Step 3: Reconfigure the IP address of each PC, click on desktop, then select IP configuration.



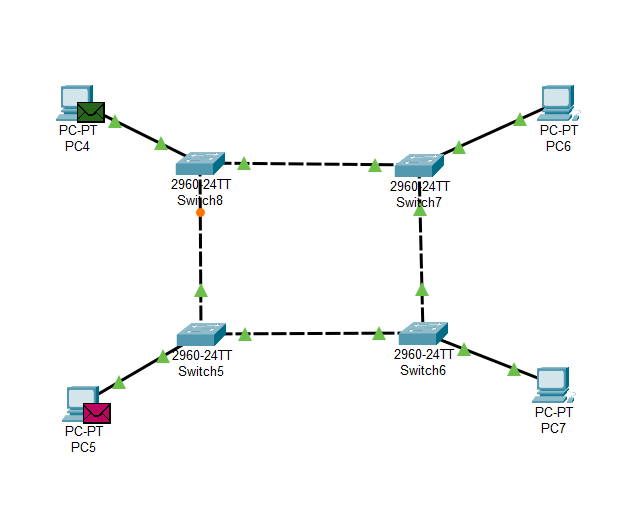
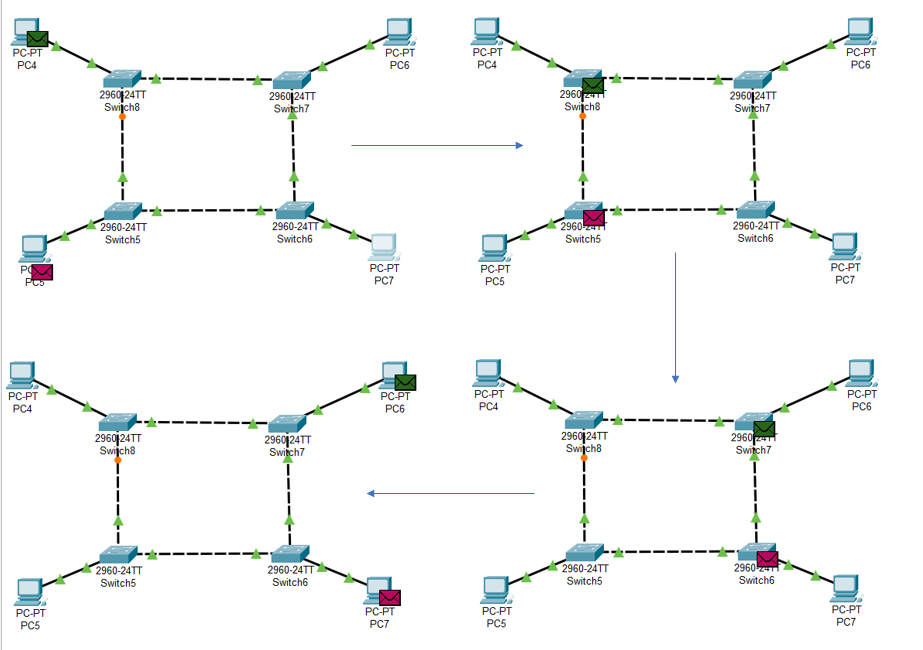
Step 4: Select simple PDU from tool bar and then

1. Click on PC4 to set as sender
2. Then click on PC6 as receiver

Repeat the same process with PC5 and PC7

**Observation:**

1. The packet travel from PC4 to PC6 and PC5 to PC7 and reach their destination through switches switch5, switch6, switch7, switch8.
2. A message confirmation is then revert back from PC6 and PC7 to PC4 and PC5.



**Precautions and Sources of Error:**

1. Make sure to use the correct type of wiring for connecting the Networking Devices and the End Devices
2. Double check the IP Address configuration of all the End Devices in case of some error.

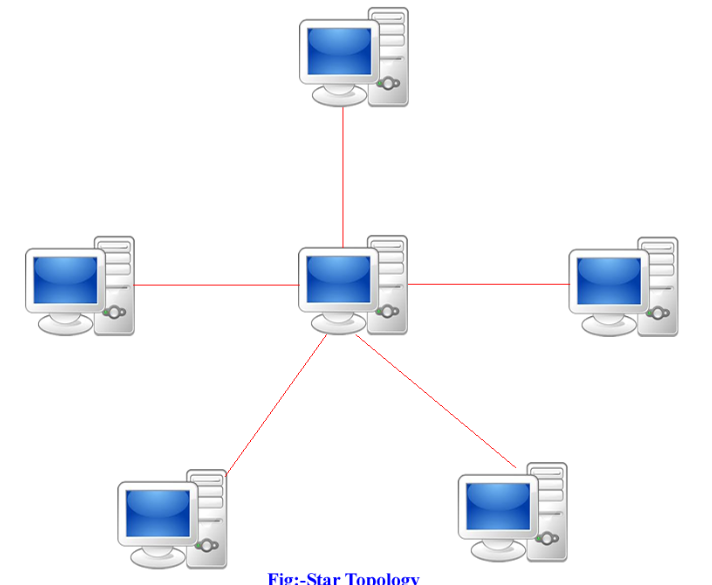
**PRACTICAL NO – 2**

**Aim:** To implement star topology in cisco packet tracer.

**SOFTWARE REQURIED:** Cisco packet tracer network simulation tool.

**THEORY: Star topology** is one of the most common [network](https://www.computerhope.com/jargon/n/network.htm) setups. Every [node](https://www.computerhope.com/jargon/n/node.htm) connects to a central network device in this configuration, like a [hub](https://www.computerhope.com/jargon/h/hub.htm), [switch](https://www.computerhope.com/jargon/s/switch.htm), or computer. The central network device acts as a [server](https://www.computerhope.com/jargon/s/server.htm), and the peripheral devices act as [clients](https://www.computerhope.com/jargon/c/client.htm).

There technically is no limit to how many computers can connect in a star topology. However, network performance can decrease as more computers are connected, resulting in slower network speeds.



**Advantages of star topology**

* Centralized management of the network through the use of the central computer, hub, or switch.
* Easy to add another computer to the network.
* If one computer on the network fails, the rest of the network continues to function normally.

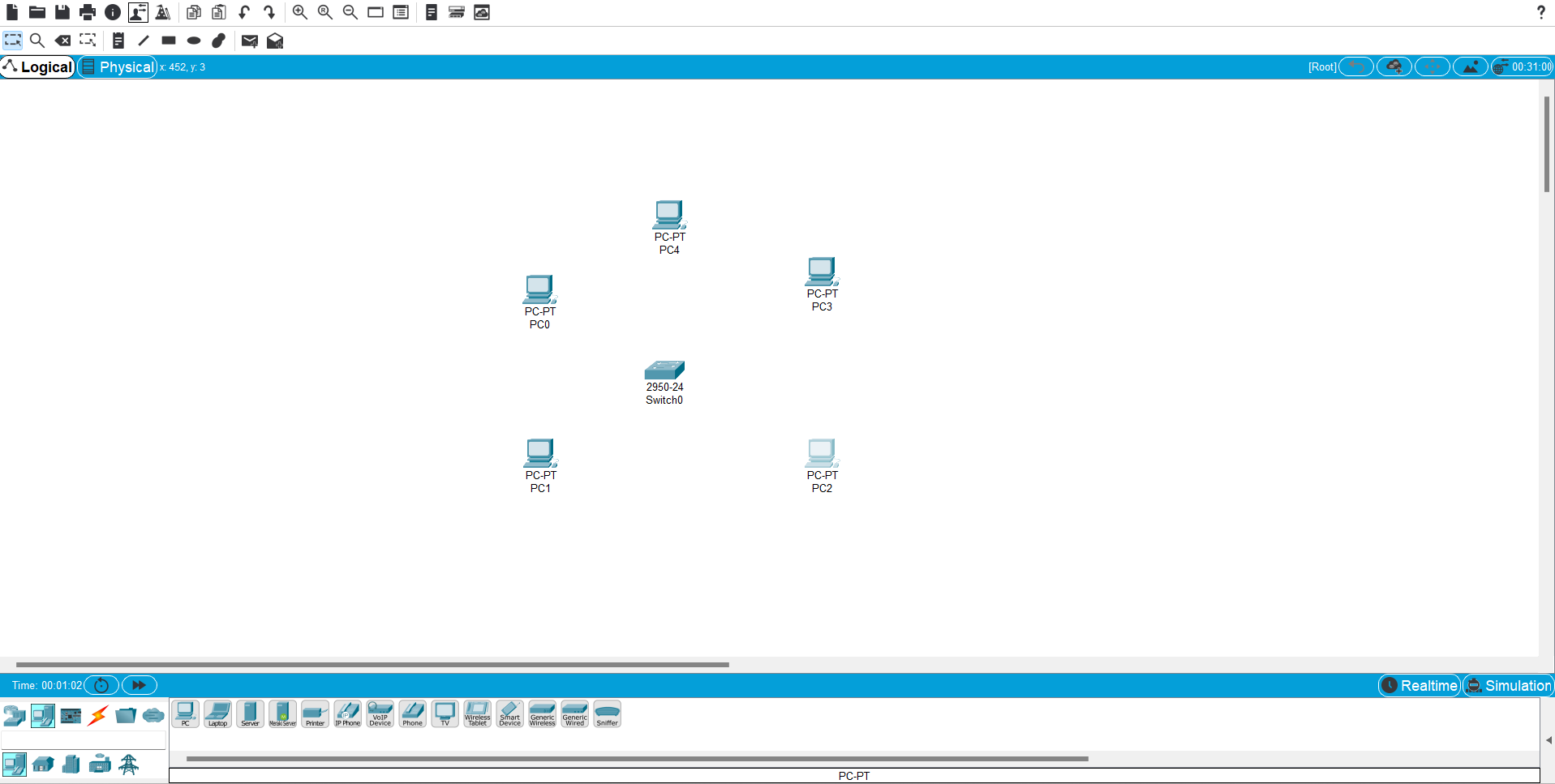
**Disadvantages of star topology**

* It may have a higher cost to implement, especially when using a switch or router as the central network device.
* The central network device determines the performance and number of nodes the network can handle.
* If the central computer, hub, or switch fails, the entire network goes down, and all computers are disconnected from the network.

**PROCEDURE:**

Step 1: Go to device section

* 1. Select network devices
  2. Select switches
  3. Select and drag 2954-24 switche on the empty canvas
  4. Select end devices
  5. Select and drag 5 PCs on the canvas

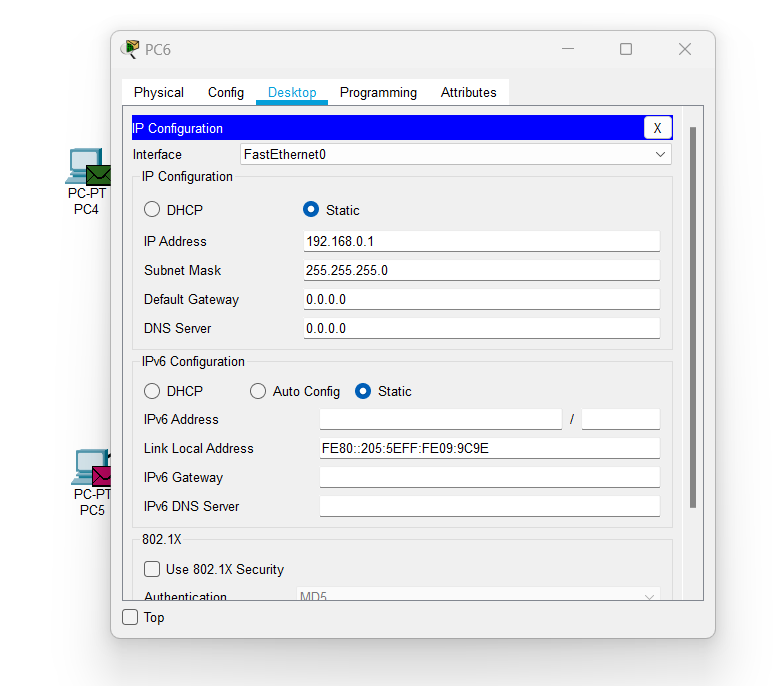


Step 2: select copper straight – through and connects PCs to switche as shown.

A diagram of a computer network

Description automatically generated

Step 3: Reconfigure the IP address of each PC, click on desktop, then select IP configuration.



Step 4: Select simple PDU from tool bar and then

1. Click on PC4 to set as sender
2. Then click on PC0 as receiver

Repeat the same process with PC3 and PC2

**Observation:**

1. The packet travel from PC4 to PC0and PC3 to PC2 and reach their destination through switch.

2.A message confirmation is then revert back from PC0 and PC2 to PC4 and PC3.

A diagram of a computer network

Description automatically generated

A diagram of a computer network

Description automatically generated

1. The status bar can be confirmed from the status bar at the bottom of the screen

A screenshot of a computer

Description automatically generated

**Precautions and Sources of Error:**

1. Make sure to use the correct type of wiring for connecting the Networking Devices and the End Devices
2. Double check the IP Address configuration of all the End Devices in case of some error.

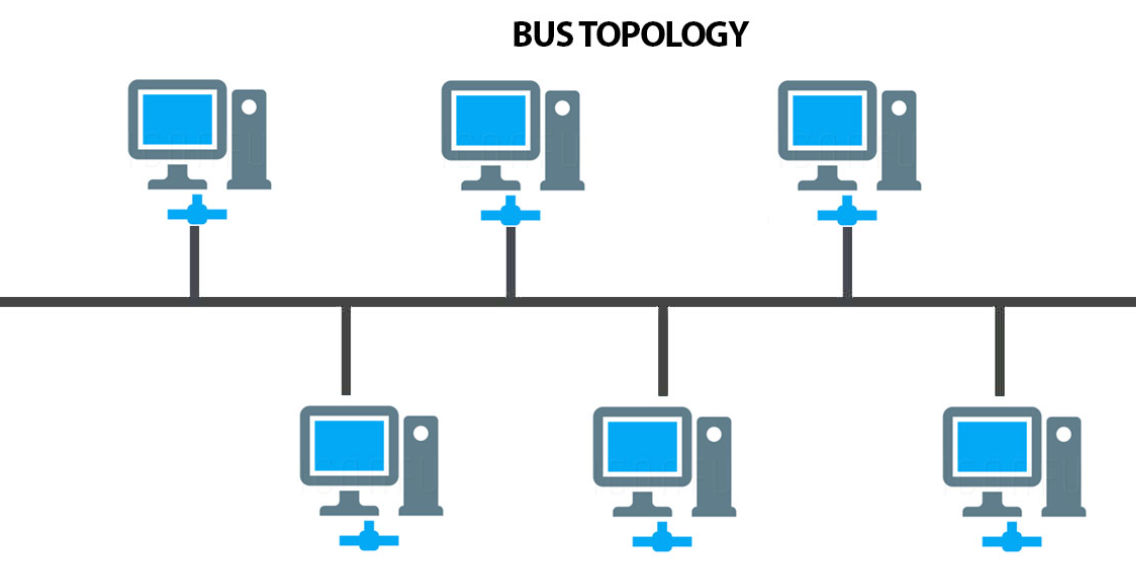
**PRACTICAL NO – 3**

**Aim:** To implement bus topology in cisco packet tracer.

**SOFTWARE REQURIED:** Cisco packet tracer network simulation tool.

**THEORY:** This is one of the simplest physical topology used for the network. This topology is famously used for the Local Area Network. In this topology, all the nodes are connected through a single cable known as ‘Backbone’. If this Backbone cable is damaged the entire network breakdowns.

Here, the node that transmits data is known as Host. All the computers connected in the network will receive all the network traffic. Each node is given equal priority for data transmission. The nodes use Media Access Technology such as a bus master to share the bus.



### **Advantages and Disadvantages**

**Advantages –**

* It is very simple to design.
* Require less cabling compared to other topologies.
* Each to implement for small networks.
* It is easy to expand by simply joining two cables together.
* Very cost-effective.

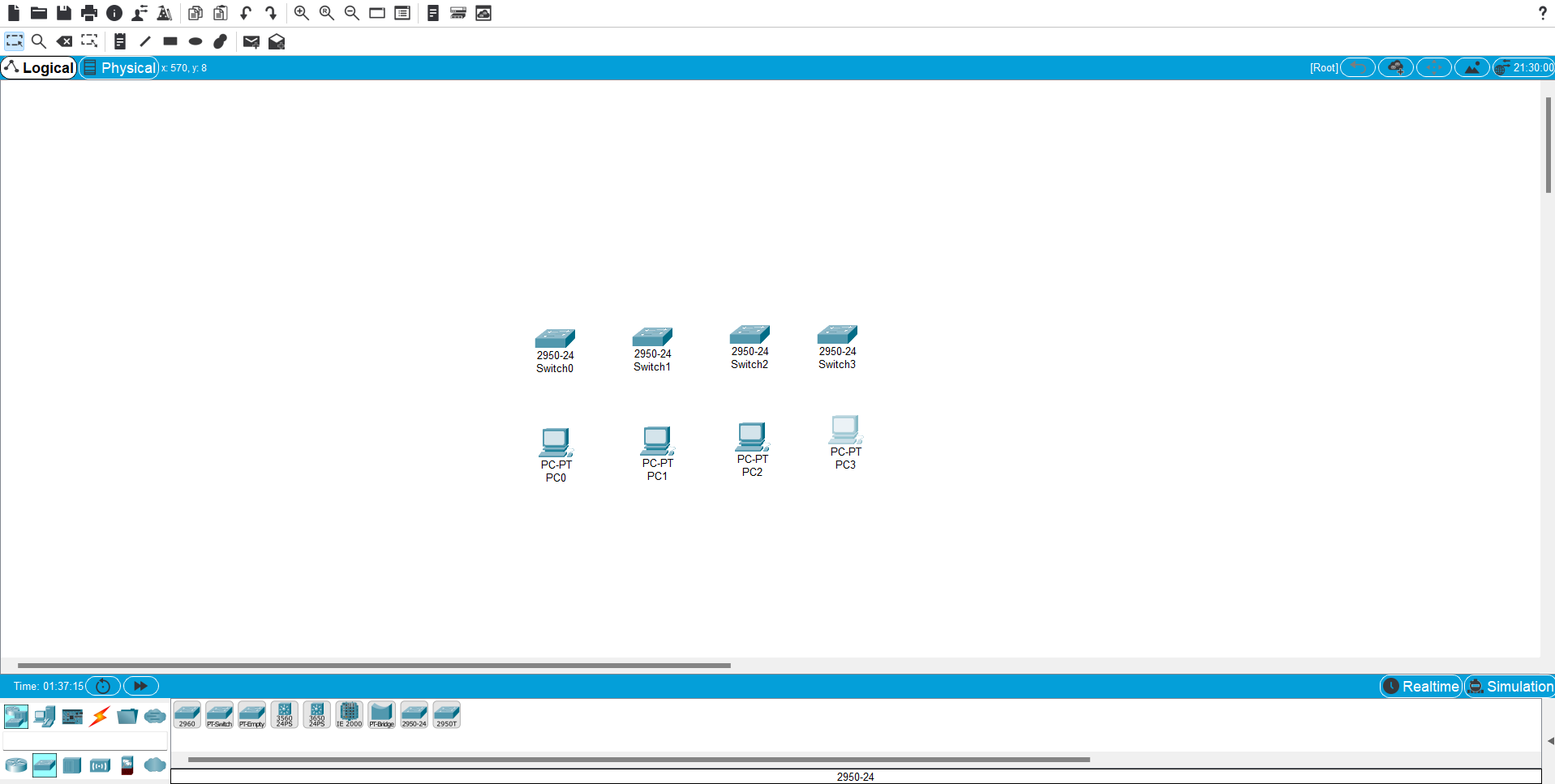
**Disadvantages –**

* It is difficult to find the flaws and faults in the network connected with this method.
* Packet loss is high.
* This topology is very slow compared to other topologies.
* If the central computer, hub, or switch fails, the entire network goes down, and all computers are disconnected from the network

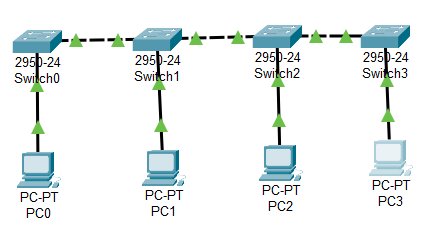
**PROCEDURE:**

Step 1: Go to device section

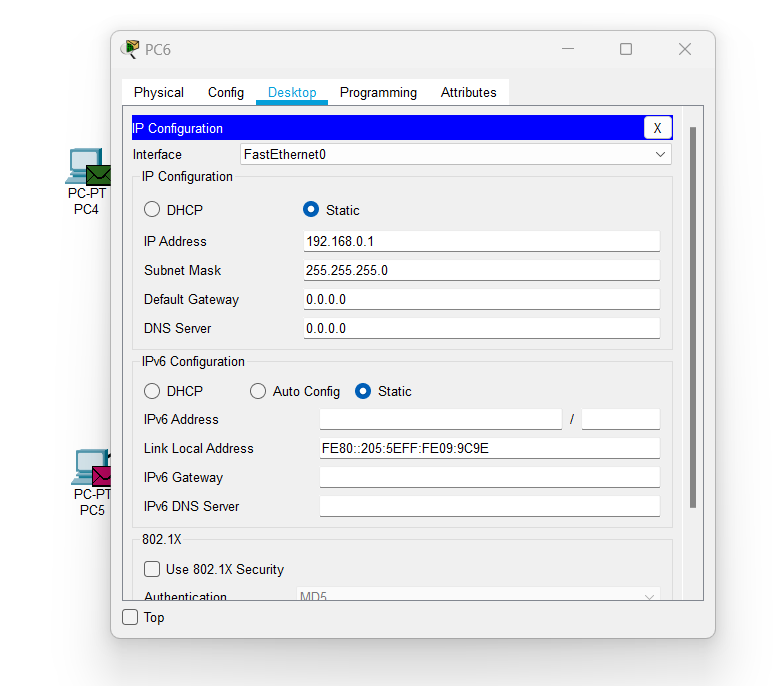
* 1. Select network devices
  2. Select switches
  3. Select and drag 4 2954-24 switche on the empty canvas
  4. Select end devices
  5. Select and drag 4 PCs on the canvas



Step 2: select copper cross - over and connects the switches as shown and select copper straight – through and connects PCs to switch as shown.



Step 3: Reconfigure the IP address of each PC, click on desktop, then select IP configuration.



Step 4: Select simple PDU from tool bar and then

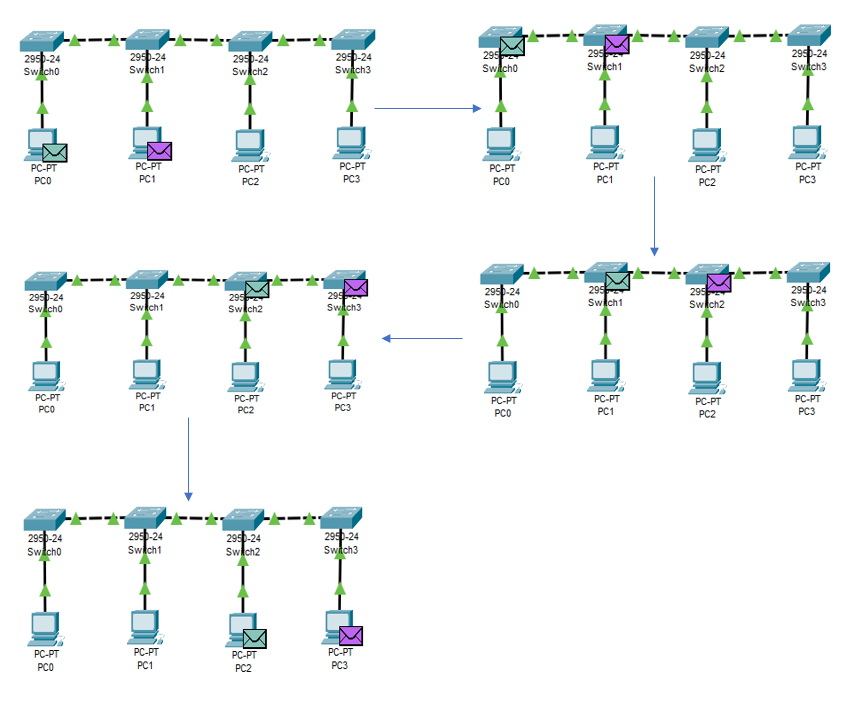
1. Click on PC0 to set as sender
2. Then click on PC2 as receiver

Repeat the same process with PC1 and PC3

**Observation:**

1. The packet travel from PC0 to PC2 and PC1 to PC3 and reach their destination through switches switch0, switch1, switch2, switch3.

2.A message confirmation is then revert back from PC2 and PC3 to PC0 and PC1.



3.The status bar can be confirmed from the status bar at the bottom of the screen

A screenshot of a computer

Description automatically generated

**Precautions and Sources of Error:**

1.Make sure to use the correct type of wiring for connecting the Networking Devices and the End Devices

2.Double check the IP Address configuration of all the End Devices in case of some error.

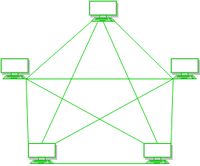
**Practical 4**

**Aim:** To Implement Mesh Topology in Cisco Packet Tracer

**Software Required:** Cisco Packet Tracer Network Simulation Tool

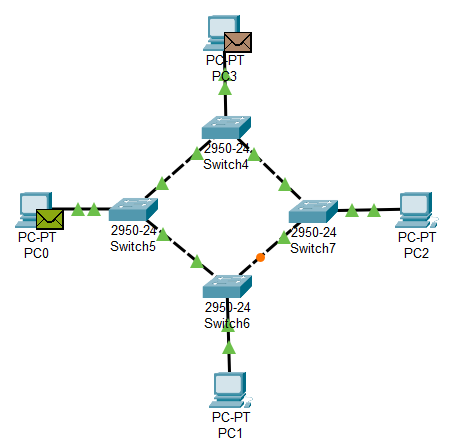
**Theory:**

A mesh topology is a network setup where each computer and network device is interconnected with one another. This topology setup allows for most transmissions to be distributed even if one of the connections goes down. It is a topology commonly used for wireless networks. Below is a visual example of a simple computer setup on a network using a mesh topology.

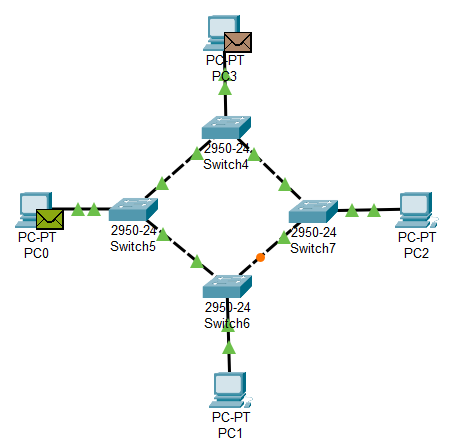


**Procedure: -**

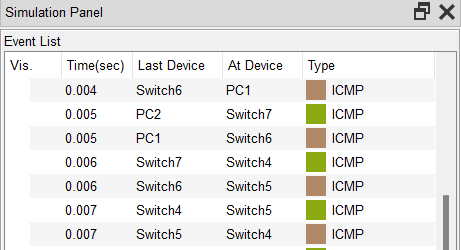
**Step 1:** Take 4 switches and 4 PCs and connect them with each other as shown in fig.



**Step 2:** Select source and destination PCs with a data packet and run the simulation.



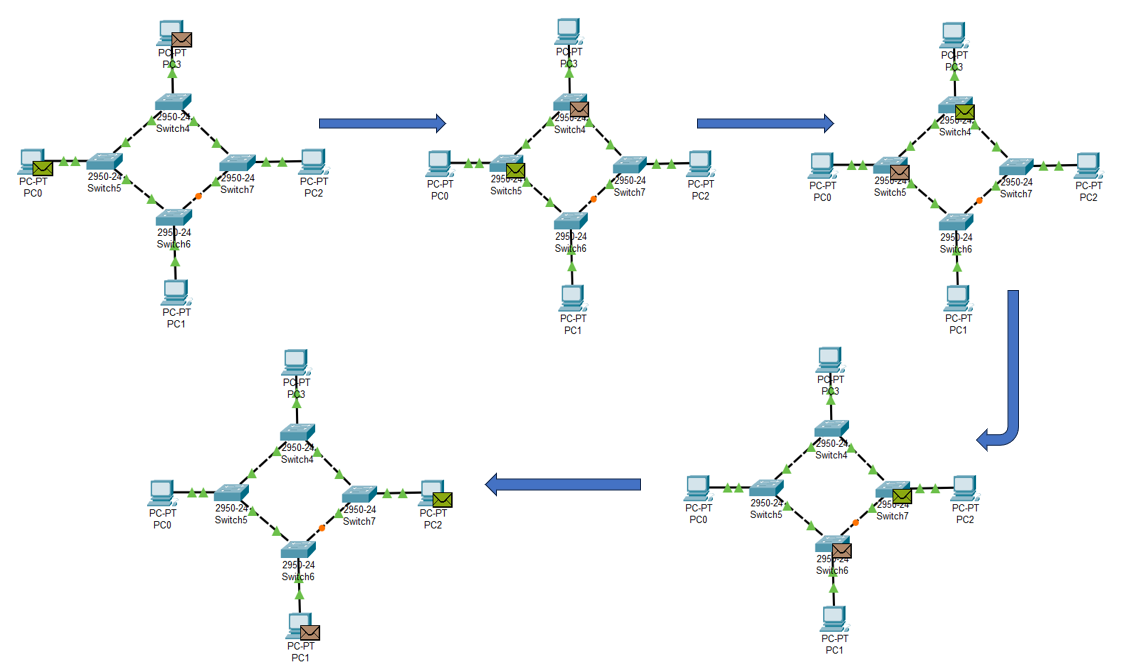
**Step 3:** The status of message transfer can be confirmed from the status bar at the bottom of the screen.



**Observation:**

1. The packet travel from PC0 to PC2 and PC3 to PC1 and reach their destination through switches switch0, switch1, switch2, switch3.

2.A message confirmation is then revert back from PC2 and PC1 to PC0 and PC3.



**Precautions and Sources of Error:**

1. Make sure to use the correct type of wiring for connecting the Networking Devices and the End Devices

2. Double check the I.P Address configuration of all the End Devices in case of some error

**Result:**

A demonstration of setting up and simulating Mesh Network Topology using Cisco Packet Tracer was completed successfully.

**Practical 5**

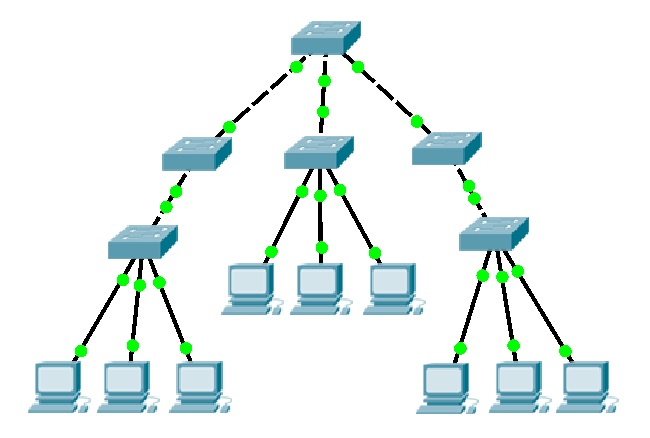
**Aim:**

To Implement Tree Network Topology in Cisco Packet Tracer

**Software Required:** Cisco Packet Tracer Network Simulation Tool

**Theory:**

In computer networks, a tree topology is sometimes known as a star bus topology because it combines features of both star and bus topologies to create a tree-like structure. In this topology. Every branch contains Star Network, and its main structure is designed in the form of a bus backbone cable. Therefore, the primary bus is attached to one or more buses and switches, which further connects to one or more Network Devices and Network Nodes. It is a very flexible computer networking method that allows you to add Network Devices to this network by simply expanding the Star Network at each branch of a tree. However, you may need to add or remove devices at certain times.



**Procedure:**

**Step 1:** Take switches and PCs and connect them in a tree manner as shown in fig

A diagram of a network

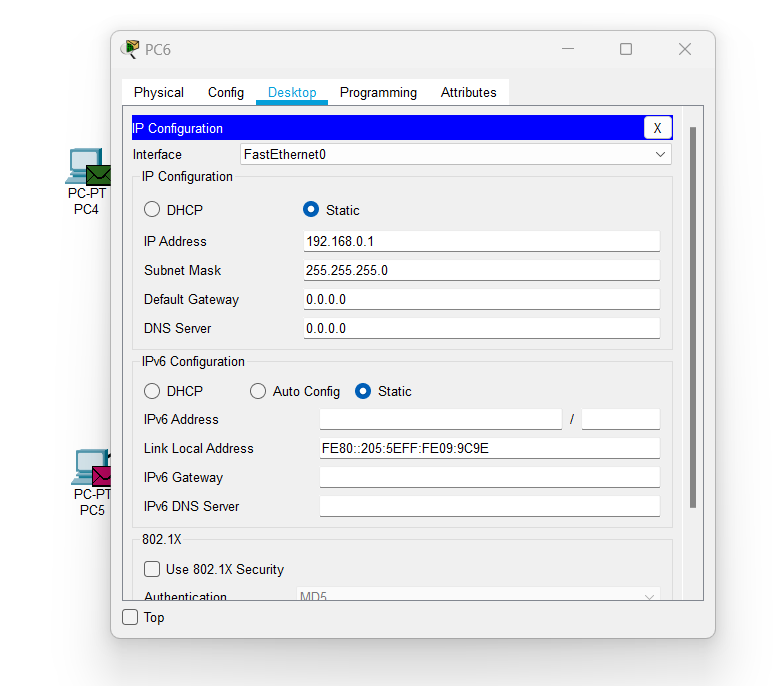
Description automatically generated

**Step 2:** Select the source and destination PCs with a data packet and run the simulation.

A computer network diagram with blue boxes

Description automatically generated

**Step 3:** Reconfigure the IP address of each PC, click on desktop, then select IP configuration.



**Step 4:** The status of message transfer can be confirmed from the status bar at the bottom of the screen.

A screenshot of a computer

Description automatically generated

**Precautions and Sources of Error:**

1. Make sure to use the correct type of wiring for connecting the Networking Devices and the End Devices

2. Double check the IP Address configuration of all the End Devices in case of some error

**Result:**

A demonstration of setting up and simulating Tree Network Topology using Cisco Packet Tracer was completed successfully.

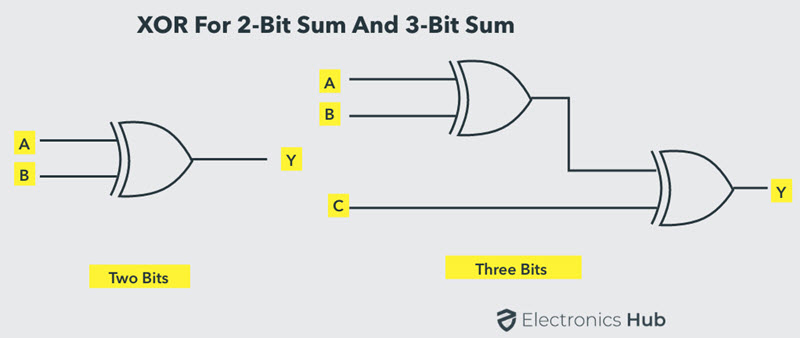
**Practical 6**

**Aim:** Write a program to add a parity bit to a 7-bit data input by a user and the sender's sites.

**Software Required:** Online C- Compiler

**Theory:**

The parity generating technique is one of the most widely used error detection techniques for the data transmission. In digital systems, when binary data is transmitted and processed, data may be subjected to noise so that such noise can alter Os (of data bits) to Is and 1s to Os.



A parity generator is a combinational logic circuit that generates the parity bit in the transmitter. On the other hand, a circuit that checks the parity in the receiver is called parity checker. A combined circuit or devices of parity generators and parity checkers are commonly used in digital systems to detect the single bit errors in the transmitted data word. The sum of the data bits and parity bits can be even or odd. In even parity, the added parity bit will make the total number of Is an even amount whereas in odd parity the added parity bit will make the total number of 1s odd amount.

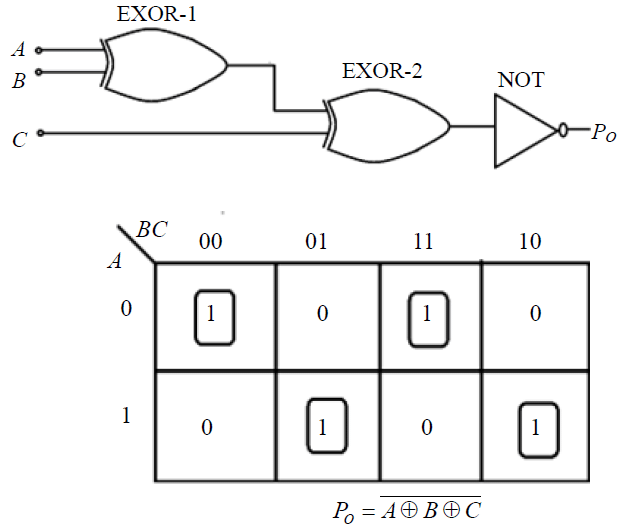
To produce two bits, sum, one Ex-OR gate is sufficient whereas for adding three bits two Ex-OR gates are required as shown in figure.

A table with numbers and symbols

Description automatically generated

**Odd Parity Generator:** Let us consider that the 3-bit data is to be transmitted with an odd parity bit. The three inputs are A, B and C and P is the output parity bit. The total number of bits must be odd to generate the odd parity bit. In the given truth table below, I is placed in the parity bit in order to make the total number of bits odd when the total number of Is in the truth table is even.

The truth table of the odd parity generator can be simplified by using K-map as



The output parity bit expression for this generator circuit is obtained as

P=(A+B+C) A diagram of a power generator

Description automatically generated with medium confidence

**Algorithm:**

Step 1

Firstly, user input a 7bit binary data from sender side as what to send.

Step 2

Sender get the 7bit data and then there will be run a modulo- two addition process i.e. XOR Gate,

XOR operation on the corresponding binary digits of each operand.

0+-0-0, 0+1=1, 1+0=1, 1+1=0

Step 3

So, if number of ones in the data is even then we get o/p as 0 through XOR gate and 0 is appended on the data. Similarly, if number of ones in the data is odd then we get o/p as I through XOR gate and I is appended on the data.

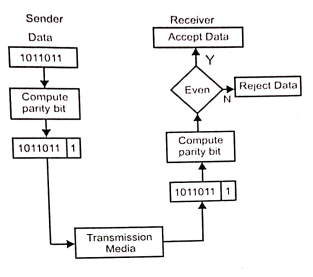
Step 4

Then the data is sent to the receiver, receiver will consume it and then receiver gets the acknowledgement whether the data is correct or having some redundant bits.

Step 5

After getting successfully reached acknowledgement, if you want to send a new data you could continue else exit.

**Flow chart :-**



**Code: -**

#include <stdio.h>

void main() {

// Declaration of Variables

int binary\_num1, binary\_num2, rem1, rem2, count=0, out=1;

int final[100];

while (out==1){

printf("Enter a binary number with the combination of Os and 1s \n");

scanf("%d", &binary\_num1); // Input the binary number (Os and Is)

printf("Enter the binary number to add \n");

scanf("%d", &binary\_num2); // Input the binary number (Os and Is)

while (binary\_num1 > 0){

rem1= binary\_num1 % 10;

rem2= binary\_num2 % 10;

if (rem1 == 1 && rem2== 1)

{

final[count] = 0;

}

else if(rem1==1 && rem2 ==0)

{

final[count] = 1;

}

else if (rem1 == 0 && rem2==1)

{

final[count] = 1;

}

else{

final[count] = 0;

}

binary\_num1 = binary\_num1/10;

binary\_num2 = binary\_num2/10;

count++;

}

printf("Current Data: ");

for (int i = 0; i < count; ++i) {

printf("%d", final[i]);

}

printf("\nTo add more binary data enter 1, else enter 0 \n");

scanf("%d", &out);

}

/\* Printing the final data \*/

printf ("Final Data: ");

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

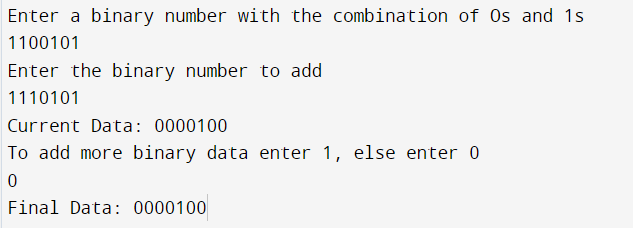
{

printf("%d", final[i]);

}

}

**OUTPUT :-**

****

**Practical 7**

**Aim:** Write a program to add redundant bit using Hamming code at the sender's side.

**Software Required:** Online C- Compiler

**Theory:**

Hamming code is a set of error-correction codes that can be used to detect and correct the errors that can occur when the data is moved or stored from the sender to the receiver. It is a technique developed by R. W. Hamming for error correction.

Redundant bits are extra binary bits that are generated and added to the information-carrying bits of data transfer to ensure that no bits were lost during the data transfer. The number of redundant bits can be calculated using the following formula:

2^r2m+r+1

where, r = redundant bit, m = data bit

Suppose the number of data bits is 7, then the number of redundant bits can be calculated using: 2^427+4 +1 Thus, t he number of redundant bits= 4 Parity bits. A parity bit is a bit appended to a data of binary bits to ensure that the total number of 1's in the data is even or odd. Parity bits are used for error detection. There are two types of parity bits:

**1. Even parity bit**: In the case of even parity. for a given set of bits, the number of I's are counted. If that count is odd, the parity bit value is set to 1, making the total count of occurrences of 1's an even number. If the total number of I's in each set of bits is already even, the parity bit's value is 0.

**2. Odd Parity bit:** In the case of odd parity, for a given set of bits, the number of I's are counted. If that count is even, the parity bit value is set to 1. making the total count of occurrences of I's an odd number. If the total number of 1's in each set of bits is already odd, the parity bit's value is 0.

**Algorithm:**

1. Ask user to enter 7-bit data

2. Calculate number of redundant bits (m-7, 1-4)

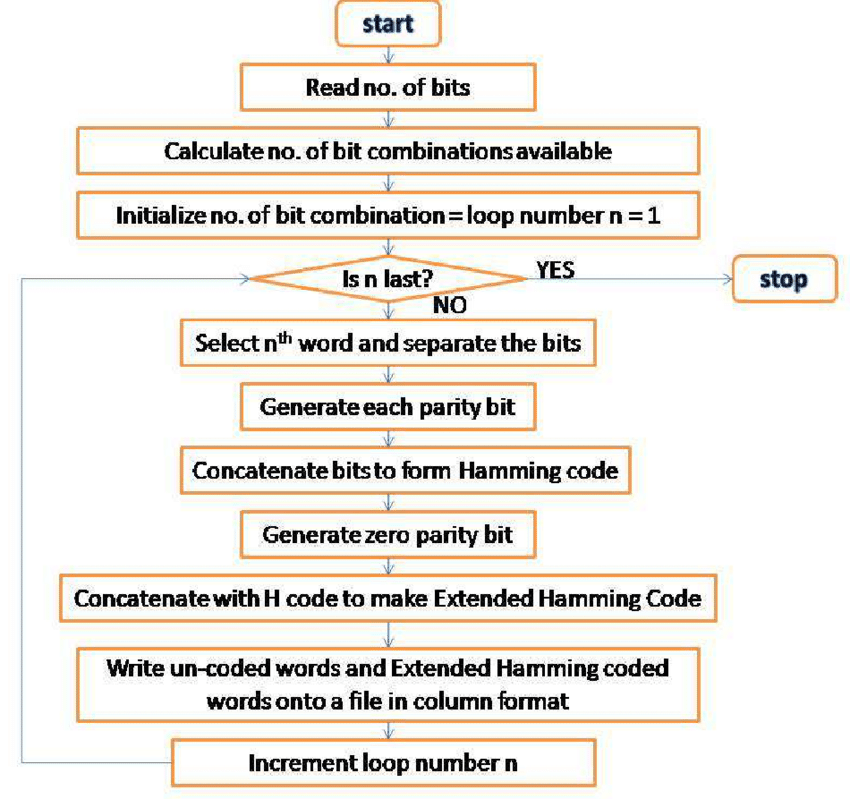
3. Initialize array of size 11

4. Put redundant bits in their position (2^i where i is number of redundant bits)

5. Put 0 initially and put the data in the rest position 6. Calculate value of redundant bits using parity [Even or Odd]

7. If user wants to continue, repeat from step 1, else end the program

**Flow Chart:**



**Code: -**

#include <math.h>

#include <stdio.h>

// Store input bits

int input[32];

// Store hamming code

int code[32];

int ham\_calc(int, int);

void solve(int input[], int):

// Function to calculate bit for //ith position

int ham\_calc(int position, int c\_1)

{

int count = 0, i, j;

i = position - 1:

// Traverse to store Hamming Code

while (i <c\_l) {

for(j=i;j<i+position; j++) {

// If current boit is 1

if (code[j] == 1)

count++;

}

// Update i

i = i +2 position;

}

if (count%20)

return 0;

else

return 1;}

// Function to calculate hamming code void solve(int input[], int n)

int i, p\_n 0, c\_l.j, k;

i=0;

// Find msg bits having set bit

// at x'th position of number

while (n> (int)pow(2, i) - (i + 1)) {

p\_n++;

i++;

}

c\_l=p\_n+n;

j=k=0;

// Traverse the msgBits

for (i=0; i<c\_l; i++) {

// Update the code

if (i == ((int)pow(2, k) - 1)) {

code[i]= 0; k++;}

// Update the code[i] to the

// input character at index j

else {

code[i] = input[i];

j++;

}}

// Traverse and update the

// hamming code

for (i=0; i<p\_n; i++) {

// Find current position

int position = (int)pow(2, i):

// Find value at current position

int value = ham\_calc(position, c\_1);

// Update the code

code[position - 1] = value;

}

// Print the Hamming Code

printf("\nThe generated Code Word is: ");

for (i=0; i<c\_l; i++)

{

printf("%d", code[i]):

}

}

**OUTPUT :-**

**A screenshot of a computer code

Description automatically generated**

**Practical 8**

**Aim:** Write a program to detect and correct redundant bit using Hamming code at the receiver's side.

**Software Required:** Online C- Compiler

**Theory:**

Hamming code is a set of error-correction codes that can be used to detect and correct the errors that can occur when the data is moved or stored from the sender to the receiver. It is a technique developed by R. W. Hamming for error correction.

Redundant bits are extra binary bits that are generated and added to the information-carrying bits of data transfer to ensure that no bits were lost during the data transfer. The number of redundant bits can be calculated using the following formula:

2^r2m+r+1

where, r redundant bit, m = data bit

Suppose the number of data bits is 7, then the number of redundant bits can be calculated using: =2^427+4 +1 Thus, the number of redundant bits 4 Parity bits. A parity bit is a bit appended to a data of binary bits to ensure that the total number of I's in the data is even or odd. Parity bits are used for error detection. There are two types of parity bits:

**1. Even parity bit**: In the case of even parity, for a given set of bits, the number of I's are counted. If

that count is odd, the parity bit value is set to 1. making the total count of occurrences of 1's an

even number. If the total number of 1's in a given set of bits is already even, the parity bit's value

is 0.

**2.Odd Parity bit -** In the case of odd parity, for a given set of bits, the number of I's are counted. If that count is even, the parity bit value is set to 1. making the total count of occurrences of I's an odd number. If the total number of 1's in each set of bits is already odd, the parity bit's value is 0.

**Algorithm:**

1. Ask user to enter 11-bit data (Hamming Code)

2. Calculate number of redundant bits (m=7, r=4)

3. Initialize array of size 11

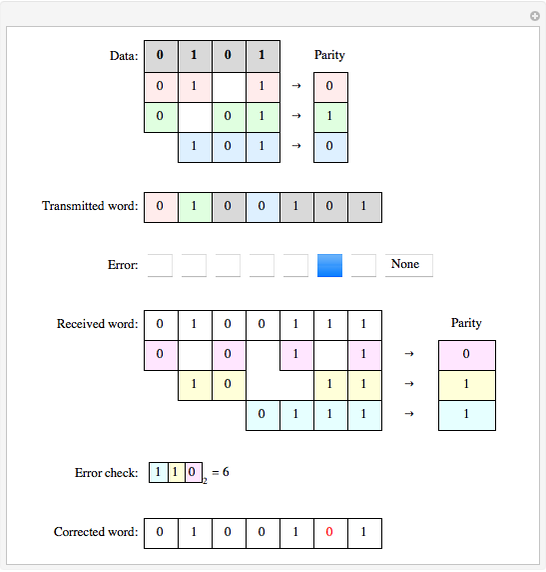
4. Put redundant bits in their position (2^i where i is number of redundant bits)

5. Put 0 initially and put the data in the rest position

6. Calculate value of redundant bits using parity [Even or Odd]

7. If user wants to continue, repeat from step 1, else end the program

**Flow Chart:**



**Code: -**

#include <math.h>

#include <stdio.h>

int d[12];

int pl,p2,p4,p8;

void main(){

printf("Enter 11 bits of data one by one\n");

scanf("%d",&d[1]);

scanf("%d",&d[2]);

scanf("%d",&d[3]);

scanf("%d",&d[4]);

scanf("%d",&d[5]);

scanf("%d",&d[6]);

scanf("%d",&d[7]);

scanf("%d",&d[8]);

scanf("%d",&d[9]);

scanf("%d",&d[10]);

scanf("%d",&d[11]);

int c=0;

for(int i=1;i>0; i--){

printf("%d",d[i]);

}

d[1]=d[1]^d[3]^d[5]^d[7]^d[9]^d[11];

d[2]=d[2]^d[3]^d[6]^d[7]^d[10]^d[11];

d[4]-d[4]^d[5]^d[6]^d[7];

d[8]=d[8]^d[10]^d[11];

for (int i=1;i<=11;i++){

if(i==pow(2,c)){

if(d[i]==1)

{

d[i]=1;

}

else{

d[i]=0;

}

c++;

}

}

printf("\nThe correct hamming code is : ");

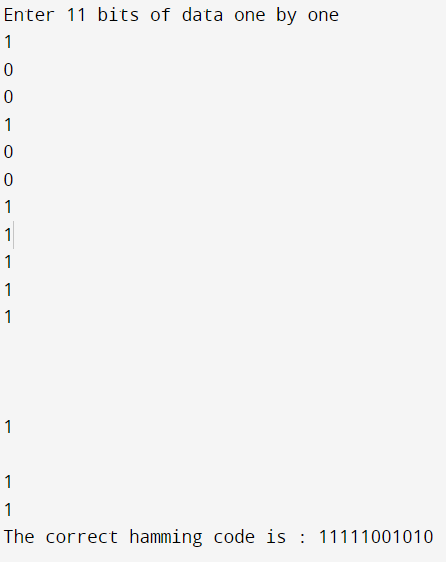
for(int i=11; i>0; i--){

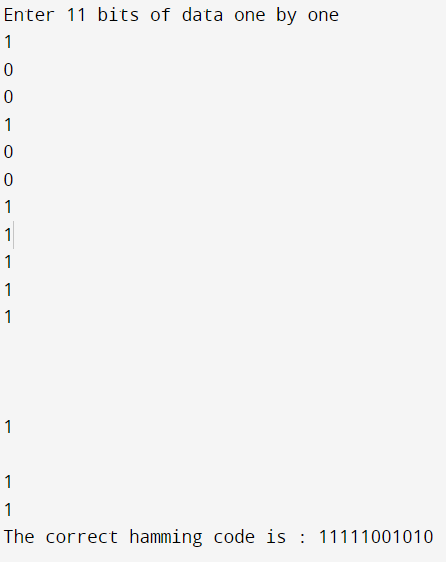
printf("%d", d[i]);

}

}

**Output:**





**Practical 9**

**Aim:** Write a program to determine the class of given IPV4 address in dotted decimal or binary notation.

**Software Required:** -Online C-Compiler

**Theory:**

There are three common notations to show an IPv4 address:

1. Binary notation (base 2)

2. Dotted-Decimal notation (base 256)

Binary Notation: Base 2 In binary notation, an IPv4 address is displayed as 32 bits. To make the address more readable, one or more spaces is usually inserted between each octet (8 bits). Each octet is often referred to as a byte. So, it is common to hear an IPv4 address referred to as a 32-bit address, a 4-octet address, or a 4-byte address.

Dotted-Decimal Notation: Base 256 To make the IPv4 address more compact and easier to read, an IPv4 address is usually written in decimal form with a decimal point (dot) separating the bytes. This format is referred to as dotted-decimal notation.

**Algorithm:**

1. Ask the user to enter the IP4 address

2. Define the type of input

2.1 Check if dotted decimal input

2.1.1 check the class of decimal input

2.2 Check if binary input

2.2.1 check the class of binary input

3. define the function for checking the class of decimal input

3.1 if input contains and leading "0"

3.1.1 display invalid input

3.2 elif any number in input > 255

3.2.1 display invalid input

3.3 if first byte input is in range of 0-127

3.3.1-display Class A

3.4 elif first byte input is in range of 128-191

3.4.1 display Class B

3.5 elif first byte input is in range of 192-223

3.5.1 display class C

3.6 elif first byte input is in range of 224-239

3.6.1 display class D

3.7 elif first byte input is in range of 240-255

3.7.1 display class E

4 define the function for checking the class of binary input

4.1 check the bytes in input has only has 0 or 1.

4.2 check each byte in input must be less than or equal to 255

4.3 if first bit of first byte in input is '0'

4.3.1 display Class A

4.4 elif first bit of first byte in input 1 and second bit is '0'

4.4.1 display Class B

4.5 elif first two bits are 1 and the third bit is '0'of first byte in input

4.5.1 dusplay Class C

4.6 elif first three bits are 'I' and the third bit is '0' of first byte in input

4.6.1 display Class D

4.7 elif first 4 bits of first byte in input is 'I'

4.7.1 display class E

5 Ask the users choice

5.1 if want to continue?

5.1.1 Repeat step 1

5.2 if want to exit

5.2.1 Exit the program

**Code :-**

#include <math.h>

#include <string.h>

void extractIpAddress(unsigned char \*sourceString,short \*ipAddress)

{

unsigned short len=0;

unsigned char oct[4]={0},cnt=0,cnt1=0,i,buf[5];

len=strlen(sourceString);

for(i=0;i<len;i++)

{

if(sourceString[i]!='.'){

buf[cnt++]=sourceString[i];

}

if(sourceString[i]=='.' || i==len-1){ buf[cnt]='\0';

cnt=0;

oct[cnt++]=atoi(buf);

}

}

ipAddress[0]=oct[0];

ipAddress[1]=oct[1];

ipAddress[2]=oct[2];

ipAddress[3]=oct[3];

}

int main(){

unsigned char ip[20]={0};

short ipAddress[4];

printf("Enter IP Address (xxx.xxx.xxx.xxx format): ");

scanf("%s", ip);

extractIpAddress(ip,&ipAddress[0]);

printf("\nlp Address: %03d.%03d.%03d. %03d\n", ipAddress[0], ipAddress[1], ipAddress[2],ipAddress[3]);

if(ipAddress[0]>=0 && ipAddress[0]<=127)

printf("Class A Ip Address.\n");

if(ipAddress[0]>127 && ipAddress[0]<191)

printf("Class B Ip Address.\n");

if(ipAddress[0]>191 && ipAddress[0]<224) printf("Class C Ip Address.\n");

if(ipAddress[0]>224 && ipAddress[0]<=239)

printf("Class D Ip Address \n");

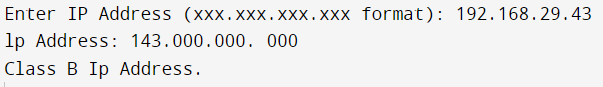
if(ipAddress[0]>239)

printf("Class E Ip Adress\n");

return 0;

}

**Output:**



**Practical 10**

**Aim**: Set up a two-computer network using CPT and explore ipconfig/all and ping commands.

**Apparatus Required:**

1. Hardware - Laptop/PC

2. Software-Android/Windows/IOS Operating System, Cisco Packet Tracer.

**Theory:**

Ipconfig Command (internet Protocol Configuration) A Windows command line utility that is used to manage the IP address assigned to the machine it is running in. Used without any additional parameters, it displays the computer's currently assigned IP. Subnet mask and default gateway addresses.

**Ipconfig/all:** Displays the full TCP/IP configuration for all adapters. Adapters can represent physical interfaces, such as installed network adapters, or logical interfaces, such as dial-up connections.

**The Ping Command:** The ping command is a very common method for troubleshooting the accessibility of devices. It uses a series of Internet Control Message Protocol (ICMP) Echo messages to determine

1. Whether a remote host is active or inactive.
2. The round-trip delay in communicating with the host
3. Packet loss

The ping command first sends an echo request packet to an address then waits for a reply The ping is successful only if:

1. The echo request gets to the destination and
2. The destination is able to get an echo reply back to the source within a predetermined time called a timeout. The default value of this timeout is two seconds on Cisco router.

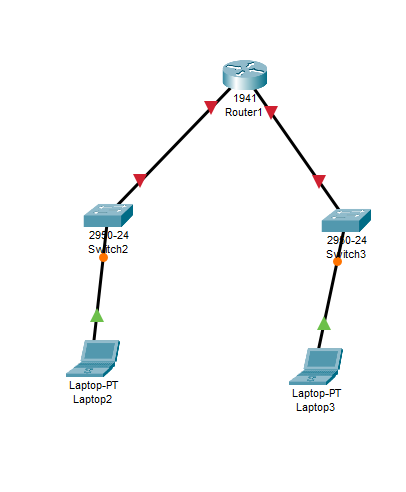
**Procedure:**

Step 1: Go to device section

* 1. Select network devices
  2. Select router.
  3. Select switches
  4. Select end devices
  5. Select and drag PCs on the canvas

Step 2: select copper straight – through and connects PCs to switches.

Step 3: Reconfigure the IP address of each PC, click on desktop, then select IP configuration.



A screenshot of a computer

Description automatically generated

1. Ipconfig for laptop 2

A computer screen shot of a black screen

Description automatically generated

1. Ipconfig all for laptop

A computer screen shot of a black and white screen

Description automatically generated

1. Ping command for laptop 2

A black background with white text

Description automatically generated

1. Ipconfig command for laptop 3

A computer screen shot of a computer program

Description automatically generated

1. Ipconfig all for laptop 3

A computer screen shot of a computer

Description automatically generated

1. Ping command for laptop 3

A black background with white text

Description automatically generated

**Precautions and Sources of Error:**

1. Make sure to use the correct type of wiring for connecting the Switches and the End Devices configuration of all the End Devices in case of some error

2. Double check the I.P Address

**Result:**

demonstration to Set up a two-computer network using CPT and explore ipconfig / all and ping commands was completed successfully.

**Practical 11**

**Aim:**

To Set a six computer network with a switch using CPT and show Unicast . Multicast and Broadcast addressing.

**Apparatus Required:**

1. Hardware - Laptop/PC
2. Software- Cisco Packet Tracer

**Theory:**

There are three types of addresses in Computer Networks

1) Unicast addressing

2) Multicast addressing

3) Broadcast addressing

**Unicast Addressing:** This type of information transfer is useful when there is a participation of a single

sender and a single recipient. So, in short, you can term it as a one-to-one transmission. For example. ifa

device having IP address 10.1.2.0 in a network wants to send the traffic stream(data packets) to the device

with IP address 20.12.4.2 in the other network, then unicast comes into the picture. This is the most common form of data transfer over the networks

**Multicast Addressing:** Multicasting in computer network is a group communication. where a sender(s) send data to multiple receivers simultaneously. It supports one - to - many and many -to -- many data transmission across LANs or WANs. Through the process of multicasting, the communication and processing overhead of sending the same data packet or data frame in minimized. Multicasting in computer networks is a group communication, where a sender(s) send data to multiple receivers simultaneously. It supports s one- 1 to- many and many -to - many data transmission across LANs or WANs

**Broadcast Addressing:** Broadcasting in computer network is a group communication. where a sender sends data to receivers simultaneously. This is an all - to - all communication model where each sending device transmits data to all other devices in the network domain.

The ways of operation of broadcasting may be-

* A high Level operation in a program, like broadcasting in Message Passing Interface
* A low level networking operation, like broadcasting on Ethernet.

**Procedure:**

1. Open Cisco Packet Tracer software
2. On the bottom left corner, go to devices section

a. Select Network Devices

b. Select Switches

c. Select and drag one 2960 switches on the empty canvas.

d. Select End Devices

e. Select and drag six PCs on the canvas

f. Select Connections

g. Select Copper Straight Through and connect the switch with End Device.

1. Change the 1Pv4 Address as follows:

PC0 - 192.168.0.1

PC1 - 192.168.0.2

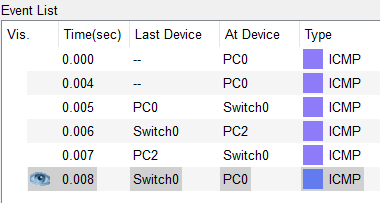
PC2 -192.168.0.3

PC3-192.168.0.4

PC4-192.168.0.5

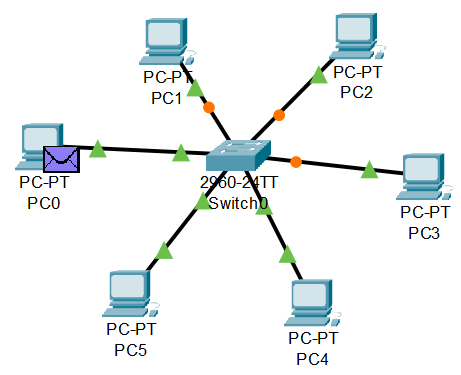
PC5 - 192.168.0.6

1. The Subnet mask will be automatically set. In this case, 255.255.255.0

A computer network diagram with many computers

Description automatically generatedA computer network diagram with words

Description automatically generated**A computer network diagram with letters and numbers

Description automatically generated****For Unicast Address:-**

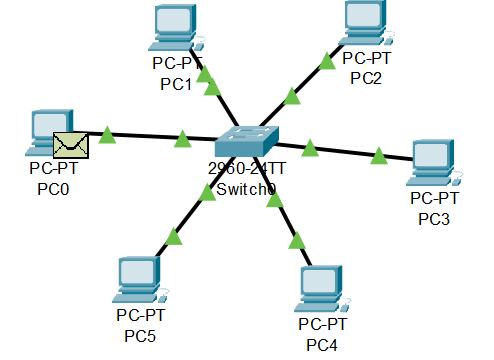
A table with numbers and symbols

Description automatically generatedA diagram of a computer network

Description automatically generated**A computer network diagram with words

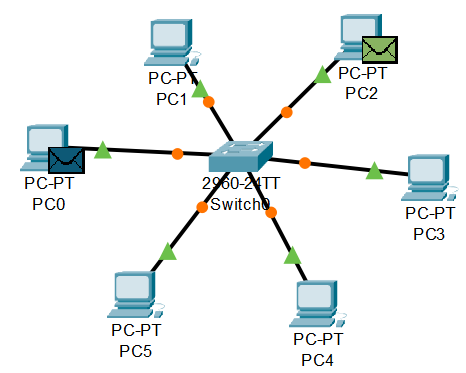
Description automatically generated**A computer network diagram with letters and numbers

Description automatically generatedA computer network diagram with letters and numbers

Description automatically generated**For Broadcast address:**

A computer network diagram with letters and numbers

Description automatically generatedA computer network diagram with letters and words

Description automatically generated**For Multicast Address:**

A diagram of a computer network

Description automatically generatedA computer network diagram with letters and numbers

Description automatically generatedA diagram of a network

Description automatically generated

A screen shot of a computer

Description automatically generated

**Precautions and Sources of Error:**

1. Make sure to use the correct type of wiring for connecting the Switches and the End Devices

2. Double check the 1.P Address configuration of all the End Devices in case of some error

**Result:**

A demonstration of setting up a six computer network with a switch using CPT and show Unicast, Multicast and Broadcast addressing was completed successfully.

**EXPERIMENT-12**

**Aim:**

To Implement Checksum Algorithm using C++

**Apparatus Required:**

1. Hardware - Laptop/PC

2. Software-C++ compiler

**Theory:-**

Checksum is the error detection method used by upper layer protocols and is more reliable than LRC, VRC and CRC. This method makes the use of Checksum Generator on Sender side and Checksum Checker on Receiver side.

At the Sender side, the data is divided into equal subunits of n bit length by the checksum generator. This bit is generally of 16-bit length. These subunits are then added together using one's complement method. This sum is of n bits. The resultant bit is then complemented. This complemented sum which is called checksum is appended to the end of original data unit and is then transmitted to Receiver.

The Receiver after receiving data + checksum passes it to checksum checker. Checksum checker divides this data unit into various subunits of equal length and adds all these subunits. These subunits also contain checksum as one of the subunits. The resultant bit is then complemented. If the complemented result is zero, it means the data is error-free. If the result is non-zero it means the data contains an error and Receiver rejects it.

A diagram of a computer algorithm

Description automatically generated with medium confidence

**CODE:**

// C++ implementation of the above approach

#include <bits/stdc++.h>

using namespace std;

// Function to find the One's complement

// of the given binary string

string Ones\_complement(string data)

{

for (int i = 0; i < data.length(); i++) {

if (data[i] == '0')

data[i] = '1';

else

data[i] = '0';

}

return data;

}

// Function to return the checksum value of

// the given string when divided in K size blocks

string checkSum(string data, int block\_size)

{

// Check data size is divisible by block\_size

// Otherwise add '0' front of the data

int n = data.length();

if (n % block\_size != 0) {

int pad\_size = block\_size - (n % block\_size);

for (int i = 0; i < pad\_size; i++) {

data = '0' + data;

}

}

// Binary addition of all blocks with carry

string result = "";

// First block of data stored in result variable

for (int i = 0; i < block\_size; i++) {

result += data[i];

}

// Loop to calculate the block

// wise addition of data

for (int i = block\_size; i < n; i += block\_size) {

// Stores the data of the next block

string next\_block = "";

for (int j = i; j < i + block\_size; j++) {

next\_block += data[j];

}

// Stores the binary addition of two blocks

string additions = "";

int sum = 0, carry = 0;

// Loop to calculate the binary addition of

// the current two blocls of k size

for (int k = block\_size - 1; k >= 0; k--) {

sum += (next\_block[k] - '0')

+ (result[k] - '0');

carry = sum / 2;

if (sum == 0) {

additions = '0' + additions;

sum = carry;

}

else if (sum == 1) {

additions = '1' + additions;

sum = carry;

}

else if (sum == 2) {

additions = '0' + additions;

sum = carry;

}

else {

additions = '1' + additions;

sum = carry;

}

}

// After binary add of two blocks with carry,

// if carry is 1 then apply binary addition

string final = "";

if (carry == 1) {

for (int l = additions.length() - 1; l >= 0;

l--) {

if (carry == 0) {

final = additions[l] + final;

}

else if (((additions[l] - '0') + carry) % 2

== 0) {

final = "0" + final;

carry = 1;

}

else {

final = "1" + final;

carry = 0;

}

}

result = final;

}

else {

result = additions;

}

}

// Return One's complements of result value

// which represents the required checksum value

return Ones\_complement(result);}

// Function to check if the received message

// is same as the senders message

bool checker(string sent\_message,

string rec\_message,

int block\_size)

{

// Checksum Value of the senders message

string sender\_checksum

= checkSum(sent\_message, block\_size);

// Checksum value for the receivers message

string receiver\_checksum = checkSum(

rec\_message + sender\_checksum, block\_size);

// If receivers checksum value is 0

if (count(receiver\_checksum.begin(),

receiver\_checksum.end(), '0')

== block\_size) {

return true;

}

else {

return false;

}

}

// Driver Code

int main()

{

string sent\_message

= "10000101011000111001010011101101";

string recv\_message

= "10000101011000111001010011101101";

int block\_size = 8;

if (checker(sent\_message,

recv\_message,

block\_size)) {

cout << "No Error";

}

else {

cout << "Error";

}

return 0;

}

**Output:**

A close-up of a number

Description automatically generated

**Precautions and Sources of Error:**

1. Make sure to check the code for any syntax or logical errors

2. Use multiple test cases to calculate the correctness of the implemented method

**Result:**

Successfully implemented Checksum Algorithm using C++ Programming Language.

**Practical 13**

**Aim:**

To Implement Pure and Slotted Aloha using C++

**Apparatus Required:**

1.Hardware- Laptop/PC

2. Software- C++ compiler

**Theory:**

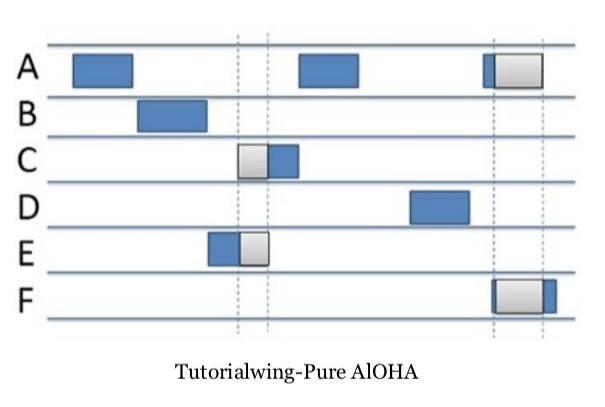
The Aloha Protocol allows several stations to send data frames over the same communication channel at the same time. This protocol is a straightforward communication method in which each network station is given equal priority and works independently.

Aloha is a medium access control (MAC) protocol for transmission of data via a shared network channel. Using this protocol, several data streams originating from multiple nodes are transferred through a multi-point transmission channel.

There are two types of Aloha protocols - Pure Aloha and Slotted Aloha.

* In Pure Aloha, the time of transmission is continuous. Whenever a station has an available frame, it sends the frame. If there is
* Collision and the frame is destroyed, the sender waits for a random amount of time before retransmitting it. In Slotted Aloha, time is divided into discrete intervals called slots, corresponding to a frame.

Pure Aloha is the basic form of Aloha contention mechanism, in which demand-driven data frames from numerous VSATs are sent to the satellite through a shared channel. It was first used at the University of Hawaii in 1970, under the direction of Norman Abramson.

1. In Pure Aloha, the time of transmission is continuous. Whenever a station has an available frame, it sends the frame.
2. A collision occurs if more than one frame tries to occupy the channel at the same time. If there is

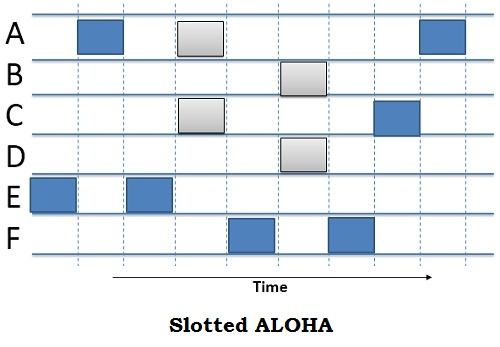
collision and the frame is destroyed, the sender waits for a random amount of time before retransmitting it.

1. After transmitting a frame, a station waits for a finite period of time to receive an acknowledgement. If the acknowledgement is not received within this time, the station assumes that the frame has been destroyed due to collision and resends the frame.

Slotted Aloha was introduced in 1972 Robert as ang improvement over Purc Aloha.

1. In stotted aloha, successful data transmission occurs

only when each slot sends just one data frame at a lime.

1. The chance of a collision is considerably reduced by doing

so.

1. Here. time is divided into discrete intervals called stots,

corresponding to a frame. The communicating stations

agree upon the slot boundaries.

Any station can send only one frame in catch slot. Also,

1. the stations cannot transmit at any time

whenever a frame is available. They should wait for the beginning of the next slot.

1. It will stay idle if no data packets are sent in any of the slots. It should be noted that if a packet does not get acknowledgment after a collision, it is deemed lost and is retransmitted in a different slot after back- off time is taken into account.

**Code:**

#include <iostream>

#include <vector>

#include <cstdlib>

#include <ctime>

using namespace std;

class AlohaSystem {

private:

vector<bool> channel; // Represents the communication channel

int numStations; // Number of stations in the system

public:

AlohaSystem(int n) : numStations(n) {

channel.resize(numStations, false);

srand(time(0));

}

void pureAloha() {

for (int i = 0; i < numStations; ++i) {

if (channel[i]) {

cout << "Station " << i + 1 << ": Collision detected, retrying...\n";

} else {

if (rand() % 2 == 0) {

cout << "Station " << i + 1 << ": Frame transmitted successfully!\n";

channel[i] = true;

} else {

cout << "Station " << i + 1 << ": Transmission failed, retrying...\n";

}

}

}

}

void slottedAloha() {

for (int i = 0; i < numStations; ++i) {

if (channel[i]) {

cout << "Station " << i + 1 << ": Collision detected, retrying...\n";

} else {

if (rand() % 2 == 0) {

cout << "Station " << i + 1 << ": Frame transmitted successfully!\n";

channel[i] = true;

} else {

cout << "Station " << i + 1 << ": Transmission failed, retrying...\n";

}

}

}

}

};

int main() {

int numStations;

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

cin >> numStations;

AlohaSystem aloha(numStations);

cout << "Pure Aloha:\n";

aloha.pureAloha();

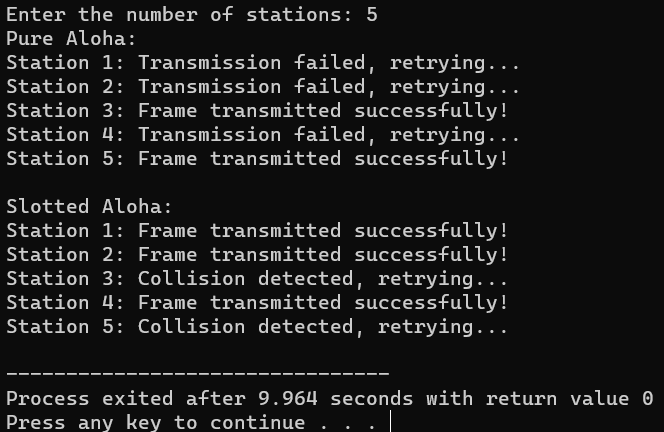
cout << "\nSlotted Aloha:\n";

aloha.slottedAloha();

return 0;

}

**Output:**

****

**Practical 14**

**Aim:** **To implement Subnetting Mask.**

**Apparatus required:**

Hardware- Laptop/Mobile phone/Pc

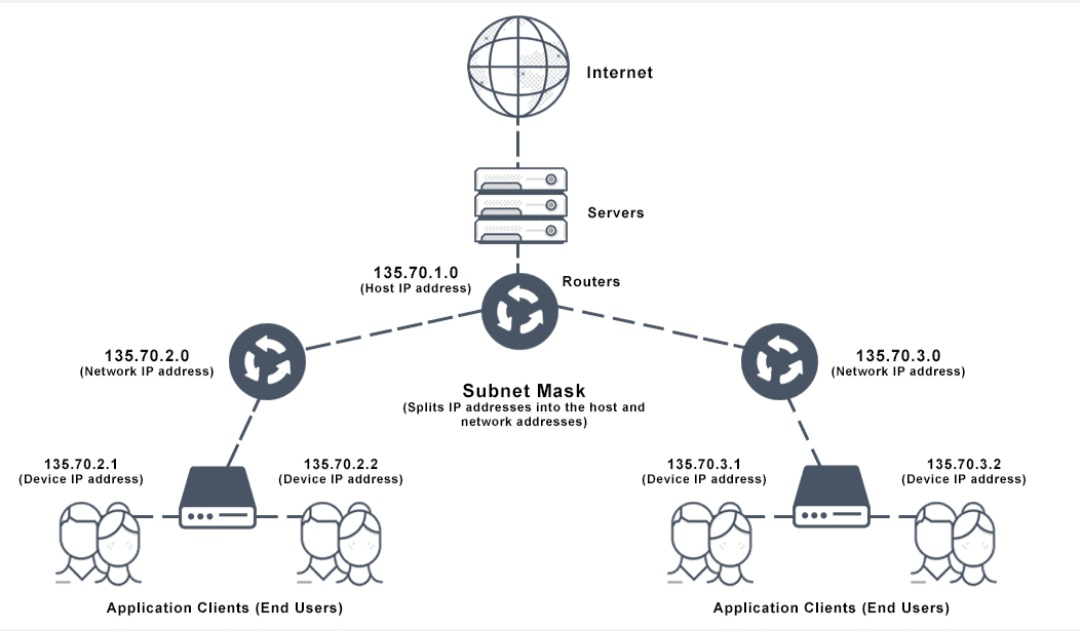
Software -VS code 1DE for java

**Theory:**

**Subnet Mask-**Every device has an IP address with two pieces: the client or host address and the server or network address. IP addresses are either configured by a DHCP seryer or manually configured (static IP addresses).The subnet mask splits the IP address into the host and network addresses, thereby defining which part of the IP address belongs to the device and which part belongs to the network.

The device called a gateway or default Gateway connects local devices to other networks. This means that when a local device wants to send information to a device at an IP address on another network, it first sends its packets to the gateway, which then forwards the data on to its destination outside of the local network.

A subnet mask is a 32-bit number created by setting host bits to all 0s and setting network bits to all 1s. In this way, the subnet mask separates the IP address into the network and host addresses.



Subnetting is the technique for logically partitioning a single physical network into multiple smaller sub-networks or subnets.

Subnetting enables an organization to conceal network complexity and reduce network traffic by adding subnets without a new network number. When a single network number must be used across many segments of a local area network (LAN), subnetting S essential.

The benefits of subnetting include:

* Reducing broadcast volume and thus network traffic
* Enabling work from home
* Allowing organizations to surpass LAN constraints such as maximum number of hosts

**Code:**

#include <stdio.h>

// Function to perform subnetting

void subnetting(char ip[], int subnetBits) {

// Convert IP address to binary

unsigned long ipAddress = 0;

sscanf(ip, "%lu", &ipAddress);

// Create subnet mask

unsigned long subnetMask = (~0UL) << (32 - subnetBits);

// Calculate network address

unsigned long networkAddress = ipAddress & subnetMask;

// Display results

printf("IP Address: %s\n", ip);

printf("Subnet Mask: %lu\n", subnetMask);

printf("Network Address: %lu\n", networkAddress);

}

int main() {

char ip[16];

int subnetBits;

// Get IP address and subnet bits from the user

printf("Enter IP address (in decimal format): ");

scanf("%s", ip);

printf("Enter Subnet Bits: ");

scanf("%d", &subnetBits);

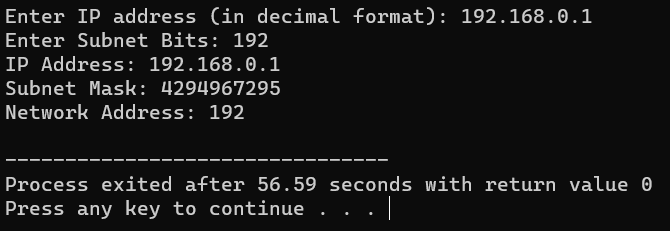
// Perform subnetting

subnetting(ip, subnetBits);

return 0;

}

**Output:**

****

**Precautions and Sources of Error:**

1. Make sure to check the code for any syntax or logical errors

2. Use multiple test cases to calculate the correctness of the implemented method

**Result:**

Successfully implemented Subnetting Mask using C++ Programming Language.