**NOIDA INSTITUTE OF ENGINEERING & TECHNOLOGY, GREATER NOIDA**

**(An Autonomous Institute)**



**Branch: Information Technology**

**Semester: VIth (3rd Year)**

**LAB FILE**

**Computer Network Lab  
 (ACSE0652)**

**BY**

**SHUBHAM MISHRA**

**(Roll No. 2201330130191)**

**Under the Supervision of**

**Mr Ahmad Nafis**

**(Academic Session: 2024-25)**

**NOIDA INSTITUTE OF ENGINEERING & TECHNOLOGY**

19, KNOWLEDGE PARK-II, INSTITUTIONAL AREA,

GREATER NOIDA, (U. P.) - 201 306, INDIA

|  |  |  |
| --- | --- | --- |
|  | **NOIDA INSTITUTE OF ENGINEERING & TECHNOLOGY, GR. NOIDA** | Session:  2024-25 |
| VI Semester |
| **DEPARTMENT OF INFORMATION TECHNOLOGY** | | |

**Computer Networks Lab (ACSE0652)**

**Index**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No.** | **Programs** | **Date** | **Signature** | **Grade** |
| 1 | To make an UTP cable with RJ45 connector and built and test simple network using UTP cable (crossover) and a hub based cable |  |  |  |
| 2 | Introduction to Network Devices (Repeater, Hub, Bridge, Switch, Router, Gateways, NIC . |  |  |  |
| 3 | Implementation of data link layer framing method such as bit stuffing in any language like C++ , Java or Python |  |  |  |
| 4 | Test the network connection using ping command and using ipconfig, netstat and trcert command provided by TCP/IP. |  |  |  |
| 5 | Implementation of CRC algorithm in any language like C++ , Java or Python. |  |  |  |
| 6 | Implementation of Hamming code (7, 4) code to limit the noise in any language like C++, Java or Python |  |  |  |
| 7 | Implementation of stop and wait protocol in any language like C++ , Java or Python |  |  |  |
| 8 | Write a program in java to find the IP address of the any site if name is given. |  |  |  |
| 9 | Write a program in java to find the IP address of the system |  |  |  |
| 10 | Implementation of Caesar cipher technique & RSA algorithm in any language like C++ , Java or Python. |  |  |  |
| 11 | Introduction to CISCO Packet Tracer. Design Bus, star, mesh, ring topology and check the connectivity using ping command. |  |  |  |
| 12 | Switch Configuration on CISCO packet tracer using CLI |  |  |  |

**Experiment-1**

**AIM:** To make an UTP cable with RJ45 connector and built and test simple network using UTP

cable(crossover) and a hub based cable

## Theory

**Physical layer** defines the cable or physical medium itself, e.g., thinnest, thicknet, unshielded twisted pairs (UTP). It transmits raw bit stream over physical cable, defines cables, cards, and physical aspects, defines NIC attachments to, hardware, how cable is attached to NIC, defines techniques to transfer bit stream to cable. All media are functionally equivalent. The main difference is in convenience and cost of installation and maintenance. **Network Cabling** Cable is the medium through which information usually moves from one network device to another.

There are several types of cable which are commonly used with LANs. In some cases, a network will utilize only one type of cable, other networks will use a variety of cable types. The type of cable chosen for a network is related to the network's topology, protocol, and size. Types of cables used in networks are:

1. Unshielded Twisted Pair (UTP) Cable
2. Shielded Twisted Pair (STP) Cable
3. Coaxial Cable
4. Fiber Optic Cable
5. wireless media

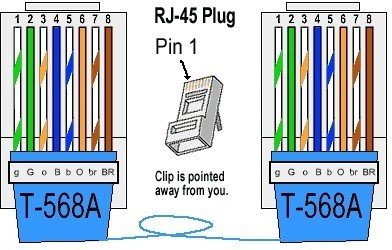
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Categories of Unshielded Twisted Pair defined by EIA/TIA** | | **Ethernet Cable Summary** | | |
| **Type** | **Use** | **Specifications** | **Cable Type** | **Maximum Length** |
| Category 1 | Voice Only (Telephone Wire) | 10BaseT | Unshielded Twisted Pair | 100 meters |
| Category 2 | Data to 4 Mbps (Local Talk) | 10Base2 | Thin Coaxial | 185 meters |
| Category 3 | Data to 10 Mbps (Ethernet) | 10Base5 | Thick Coaxial | 500 meters |

**Unshielded Twisted Pair Connector-**The standard connector for unshielded twisted pair cabling is an RJ-45 connector.



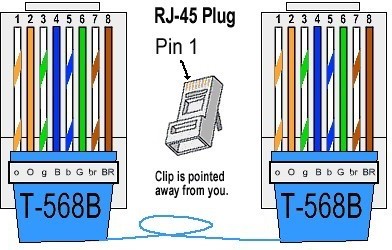
**Shielded Twisted Pair (STP) Cable -**A disadvantage of UTP is that it may be susceptible to radio and electrical frequency interference. Shielded twisted pair (STP) is suitable for environments with electrical interference; however, the extra shielding can make the cables quite bulky. Shielded twisted pair is often used on networks using Token Ring topology.

## Ethernet Cable: color – code Standards T-568A Straight-Through Ethernet Cable



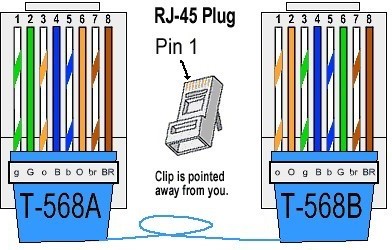
The T-568A standard is supposed to be used in new network installations. Most off-the-shelf Ethernet cables are still of the T-568B standard; however, it makes no functional difference. **T568B**

## Straight-Through Ethernet Cable



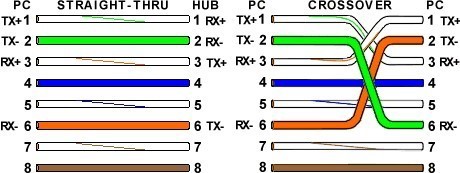
Both the T-568A and the T-568B standard Straight-Through cables are used most often as patch cords for Ethernet connections. If we require a cable to connect two Ethernet devices directly together without a hub or when connecting two hubs together, we need to use a Crossover cable instead.

## RJ-45 Crossover Ethernet Cable



|  |  |
| --- | --- |
| **Standard End** | **Crossover End** |
|  | |
| Pin 1 White/Orange | Pin 1 White/Green |
|  | |
| Pin 2 Orange | Pin 2 Green |
|  | |
| Pin 3 White/Green | Pin 3 White/Orange |
|  | |
| Pin 4 Blue | Pin 4 Blue |
| Pin 5 White/Blue | Pin 5 White/Blue |
|  | |
| Pin 6 Green | Pin 6 Orange |
|  | |
| Pin 7 White/Brown | Pin 7 White/Brown |
|  | |
| Pin 8 Brown | Pin 8 Brown |

Crossover Ethernet cable is to wire one end using the T-568A standard and the other end using the T-568B standard. Another way of remembering the color coding is to simply switch the Green set of wires in place with the Orange set of wires. Specifically, switch the solid Green (G) with the solid Orange, and switch the green/white with the orange/white.



By looking at a T-568A UTP Ethernet straight-thru cable and an Ethernet crossover cable with a T-568B end, we see that the TX (transmitter) pins are connected to the corresponding RX (receiver) pins, plus to plus and minus to minus. You can also see that both the blue and brown wire pairs on pins 4, 5, 7, and 8 are not used in either standard. What you may not realize is that, these same pins 4, 5, 7, and 8 are not used or required in 100BASE-TX as well. So why bother using these wires, well for one thing its simply easier to make a connection with all the wires grouped together. Otherwise you'll be spending time trying to fit those tiny little wires into each of the corresponding holes in the RJ-45 connector.

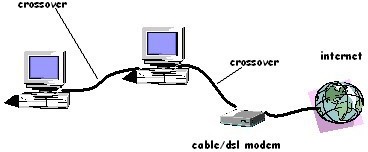
## Deciding which cable to use :

Here's a small chart that will give an idea which cable should go where:

|  |  |
| --- | --- |
| Computer to Computer | Crossover Cable |
| Computer to Hub | Straight through Cable |
| Hub Port to Port | Crossover Cable |
| Hub Uplink to Hub Port | Straight through Cable |
| Computer to Broadband Router Port | Straight through Cable |
| Computer to Switch | Straight through Cable |
| Computer to Cable Modem | Crossover Cable |

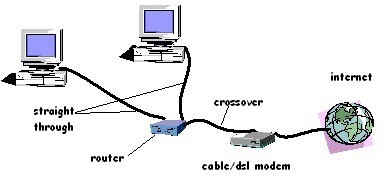
## Two Computers Scenario:

If we have only two computers, it will need two nic's and a crossover cable. If you also require both computers to have internet access, then you could either **(a)**. Install another NIC (connect it to the cable/DSL modem with a crossover cable) or fax modem in one and connect it to the internet, then use a software proxy, ICS, Wingate, etc to share the internet.

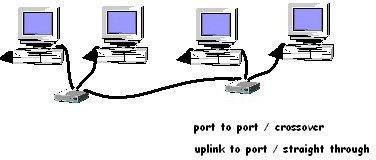


## OR

**(b).** Get a router (Connect it using a crossover cable to the cable modem and connect the computers to the built in switch on the router. Some routers don't have built in switches, in this case you will need to get a switch or hub.



**c) More than two computers scenario (hub/switch cascading):** With more computers we will need a hub/switch and straight through cables, connect all computers to the hub/switch with straight through cable. If you run out of ports then add another hub/switch to your network. Connect the uplink of one hub/switch to a port on the other hub/switch using a straight through cable, or connect port to port using a crossover cable.



If we have more than two hubs/switches, then connect all the hub/switches using uplink ports and straight through cables, to regular ports on a main hub/switch.

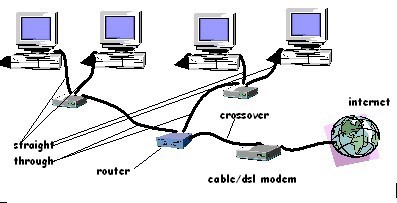
If you require internet connectivity:

1. Install another NIC (connect it to the cable/DSL modem with a crossover cable), or fax modem in one machine and connect it to the internet. Use a software proxy, ICS, Wingate, etc to share the internet.

Now connect the gateway computer and the rest of the computers to the hub's ports using straight through cables.

## OR

1. Using a broadband router, connect a cable modem with a crossover cable to the WAN port of the router, then connect the other computers to the router's LAN ports If you do not have enough ports then get more hub/switches. Connect the uplink ports to the router's LAN port with straight through cables and connect your computers to the hubs/switches.



## Procedure-

1. Start on one end of cable and strip the cable jacket off using a stripper.
2. Spread, untwist the pairs, and arrange the wires in the order of the desired cable end. Flatten the end between your thumb and forefinger. Trim the ends of the wires so they are even with one another.
3. Hold the RJ-45 plug with the clip facing down.Push the wires firmly into the plug. Inspect each wire is flat even at the front of the plug. Check the order of the wires. Check that the jacket is fitted right against the stop of the plug.Carefully hold the wire and firmly crimp the RJ-45 with the crimper.
4. Check the color orientation, check that the crimped connection is not about to come apart, and check to see if the wires are flat against the front of the plug
5. Similarly make end 2 according to the color coding for straight through cable or crossover cable on the basis of requirement.

## Connecting PC’s using direct cable UTP –

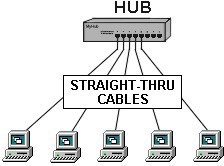
1. use crossover UTP cable.
2. Insert RJ45 connector to both PC’s NIC card.
3. restart both computers and both of them detect the network after logon.



## Connecting PC to hub-

1 use straight through UTP cable.

2.Insert RJ45 connector to computer’s NIC card and other end to HUB. 3.Switch o power of HUB and computer.



## Result-

The study of an UTP cable with RJ-45 connector is completed, and build the simple network using UTP cable and a hub based network successfully.

## Precautions-

* A straight-thru cable has identical ends.
* A crossover cable has different ends.
* A straight-thru is used as a patch cord in Ethernet connections.
* A crossover is used to connect two Ethernet devices without a hub or for connecting two hubs.
* No more than 1/2" of the Ethernet cable should be untwisted otherwise it will be susceptible to crosstalk.
* Do not deform, do not bend, do not stretch, do not staple, do not run parallel with power cables, and do not run Ethernet cables near noise inducing components.

# Experiment No.- 2

**AIM-** Introduction to Network Devices (Repeater, Hub, Bridge, Switch, Router, Gateways, NIC .

1. **Repeater** – A repeater operates at the physical layer. Its job is to regenerate the signal over the same network before the signal becomes too weak or corrupted so as to extend the length to which the signal can be transmitted over the same network. An important point to be noted about repeaters is that they do not amplify the signal. When the signal becomes weak, they copy the signal bit by bit and regenerate it at the original strength. It is a 2 port device.



1. **Hub** – A hub is basically a multiport repeater. A hub connects multiple wires coming from different branches, for example, the connector in star topology which connects different stations. Hubs cannot filter data, so data packets are sent to all connected devices. In other words, collision domain of all hosts connected through Hub remains one. Also, they do not have intelligence to find out best path for data packets which leads to inefficiencies and wastage.

## Types of Hub

**Active Hub:-** These are the hubs which have their own power supply and can clean, boost and relay the signal along with the network. It serves both as a repeater as well as wiring center.

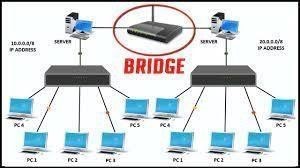
These are used to extend the maximum distance between nodes.





**Passive Hub :-** These are the hubs which collect wiring from nodes and power supply from active hub. These hubs relay signals onto the network without cleaning and boosting them and can’t be used to extend the distance between nodes.

1. **Bridge** – A bridge operates at data link layer. A bridge is a repeater, with add on the functionality of filtering content by reading the MAC addresses of source and destination. It is also used for interconnecting two LANs working on the same protocol. It has a single input and single output port, thus making it a 2 port device.



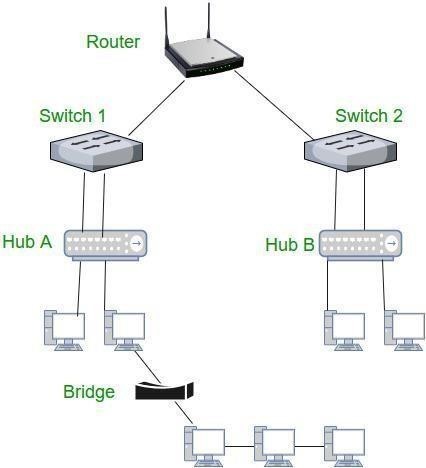
**Types of Bridges**

**Transparent Bridges: -** These are the bridge in which the stations are completely unaware of the bridge’s existence i.e whether or not a bridge is added or deleted from the network, reconfiguration of the stations is unnecessary. These bridges make use of two processes i.e bridge forwarding and bridge learning.

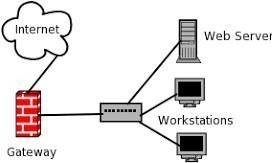
**Source Routing Bridges: -** In these bridges, routing operation is performed by source station and the frame specifies which route to follow. The hot can discover frame by sending a special frame called discovery frame, which spreads through the entire network using all possible paths to destination.

1. **Switch** – 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. A switch is a data link layer device. The switch can perform error checking before forwarding data, that makes it very efficient as it does not forward packets that have errors and forward good packets selectively to correct port only. In other words, switch divides collision domain of hosts, but broadcast domain remains same.
2. **Routers** – A router is a device like a switch that routes data packets based on their IP addresses. 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. Router divide broadcast domains of hosts connected through it. A router is connected to two or more data lines from different IP networks. When a data packet comes in on one of the lines, the router reads the network address information in the packet header to determine the ultimate destination. Then, using information in its routing table or routing policy, it directs the packet to the next network on its journey.

The most familiar type of IP routers are home and small office routers that simply forward IP packets between the home computers and the Internet. More sophisticated routers, such as enterprise routers, connect large business or ISP networks up to the powerful core routers that forward data at high speed along the optical fiber lines of the Internet backbone.



1. **Gateway** – A gateway, as the name suggests, is a passage to connect two networks together that may work upon different networking models. They basically work as the messenger agents that take data from one system, interpret it, and transfer it to another system. Gateways are also called protocol converters and can operate at any network layer. Gateways are generally more complex than switch or router.



1. **NIC** - A network interface card (NIC) is a hardware component without which a computer cannot be connected over a network. It is a circuit board installed in a computer that provides a dedicated network connection to the computer. It is also called network interface.



**Results-**

The study of important network devices is accomplished successfully.

# Experiment No.- 3

**Aim -** Implementation of data link layer framing method such as bit stuffing in any language like

C++ , Java or Python.

# Code-

def bitStuffing(N, arr):

brr = [0 for \_ in range(30)] k = 0

i = 0

j = 0

count = 1 while (i < N):

if (arr[i] == 1):

brr[j] = arr[i] k = i + 1 while True:

if not (k < N and arr[k] == 1 and count < 5): break

j += 1

brr[j] = arr[k] count += 1

if (count == 5): j += 1

brr[j] = 0 i = k

k += 1

else:

brr[j] = arr[i] i += 1

j += 1

for i in range(0, j): print(brr[i], end = "")

if name == " main ":

N = 6

arr = [ 1, 1, 1, 1, 1, 1 ]

bitStuffing(N, arr)

**Input and Output-**



# Result-

The Implementation of data link layer framing method such as bit stuffing is done successfully.

# Experiment No.- 4

**AIM-** Test the network connection using ping command and using ipconfig, netstat and trcert command provided by TCP/IP.

## Theory and Procedure-

1. **ping command**

The ping command is a Command Prompt command used to test the ability of the source computer to reach a specified destination computer. The ping command is usually used as a simple way to verify that a computer can communicate over the network with another computer or network device.

The ping command operates by sending ***Internet Control Message Protocol (ICMP) Echo Request*** messages to the destination computer and waiting for a response. How many of those responses are returned, and how long it takes for them to return, are the two major pieces of information that the ping command provides.

For example, you might find that there are no responses when pinging a network printer, only to find out that the printer is offline and its cable needs replaced. Or maybe you need to ping a router to verify that your computer can connect to it, to eliminate it as a possible cause for a networking issue.

|  |  |
| --- | --- |
| **Ping Command Options** | |
| **Item** | **Explanation** |
| **-t** | Using this option will ping the target until you force it to stop by using Ctrl-C. |
| **-a** | This ping command option will resolve, if possible, the hostname of an IP address target. |
| **-n**count | This option sets the number of ICMP Echo Requests to send, from 1 to 4294967295. The ping command will send 4 by default if **-n** isn't used. |
| **-l** size | Use this option to set the size, in bytes, of the echo request packet from 32 to 65,527. The ping command will send a 32-byte echo request if you don't use the **-l** option. |
| **-f** | Use this ping command option to prevent ICMP Echo Requests from being fragmented by routers between you and the target. The **-f** option is most often used to troubleshoot Path Maximum Transmission Unit (PMTU) issues. |
| **-i TTL** | This option sets the Time to Live (TTL) value, the maximum of which is 255. |
| **-v TOS** | This option allows you to set a Type of Service (TOS) value. Beginning in Windows 7, this option no longer functions but still exists for compatibility reasons. |
| **-rcount** | Use this ping command option to specify the number of hops between your computer and the target computer or device that you'd like to be recorded and displayed. The |

|  |  |
| --- | --- |
|  | maximum value for count is 9, so use the tracert command instead if you're interested in viewing all the hops between two devices. |
| **-scount** | Use this option to report the time, in Internet Timestamp format, that each echo request is received and echo reply is sent. The maximum value for count is 4,  meaning that only the first four hops can be time stamped. |
| **-wtimeout** | Specifying a timeout value when executing the ping command adjusts the amount of  time, in milliseconds, that ping waits for each reply. If you don't use the -w option, the default timeout value of 4000 is used, which is 4 seconds. |
| **-R** | This option tells the ping command to trace the round trip path. |
| **-Ssrcaddr** | Use this option to specify the source address. |
| **-p** | Use this switch to ping a Hyper-V Network Virtualization provider address. |
| **-4** | This forces the ping command to use IPv4 only but is only necessary if target is a hostname and not an IP address. |
| **-6** | This forces the ping command to use IPv6 only but as with the -4option, is only necessary when pinging a hostname. |
| **Target** | This is the destination you wish to ping, either an IP address or a hostname. |
| **/?** | Use the help switch with the ping command to show detailed help about the command's several options. |

## ping 127.0.01

In the above example, we're pinging 127.0.0.1, also called the IPv4 localhost IP address or IPv4 loopback IP address, without options.

Using the ping command to ping 127.0.0.1 is an excellent way to test that Windows' network features are working properly but it says nothing about your own network hardware or your connection to any other computer or device. The IPv6 version of this test would be **ping::1**.

## ipconfig command

This command details how IP is setup on the machine you are logged into. At the command prompt, type: ipconfig

There are also a variety of switches for *ipconfig* that add functions. These are invoked by entering "ipconfig /{*switch*}". To obtain a list of switches, enter "ipconfig /?" or "ipconfig -?".

These are shown in the figure below.

|  |  |
| --- | --- |
| **ipconfig [/? | /all | /release [adapter] | /renew [adapter]]** | |
| /? | Display this help message. |
| /all | Display full configuration information. |
| /release | Release the IP address for the specified adapter. |
| /renew | Renew the IP address for the specified adapter. |

For a detailed output of network parameters, use the command "ipconfig /all". Unless you are experienced with networks, however, this may be more than we want to know.

## tracert command

This command shows you all the routers that your TCP/IP message goes through in order to make a connection to another IP address.

At the command prompt, type: tracert xxx Where xxx is the IP address or URL of the machine that we are trying to connect to.

Here is the result of tracert from college computer to yahoo.com *( DNS has resolved the name “yahoo.com” to 129.64.99.33)*

Other options available in tracert command

|  |  |
| --- | --- |
| **tracert [-d] [-h maximum\_hops] [-j host-list] [-w timeout] target\_name** | |
| -d | Do not resolve addresses to hostnames. |

## netstat command

The netstat command is used to display the TCP/IP network protocol statistics and information

|  |  |
| --- | --- |
| **NETSTAT [-a] [-e] [-n] [-s] [-p proto] [-r] [interval]** | |
| -a | Displays all connections and listening ports. (Server-side connections are normally not shown). |
| -e | Displays Ethernet statistics. This may be combined with the -s option. |
| -n | Displays addresses and port numbers in numerical form. |
| -p proto | Shows connections for the protocol specified by proto; proto may be tcp or udp. If used with the -s option to display per-protocol statistics, proto may be tcp, udp, or ip. |
| -r | Displays the contents of the routing table. |
| -s | Displays per-protocol statistics. By default, statistics are shown for TCP, UDP and IP; the  -p option may be used to specify a subset of the default. |
| interval | Redisplays selected statistics, pausing interval seconds between each display. Press CTRL+C to stop redisplaying statistics. If omitted, netstat will print the current configuration information once. |

## nslookup command

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

## nslookup google.com:

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

## nslookup 192.168.0.10:

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

## nslookup -type=any google.com

Lookup for any record we can also view all the available DNS records using*- type=any* option.

## nslookup -type=soa redhat.com:

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

## nslookup -type=ns google.com

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

## nslookup -type=a google.com

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

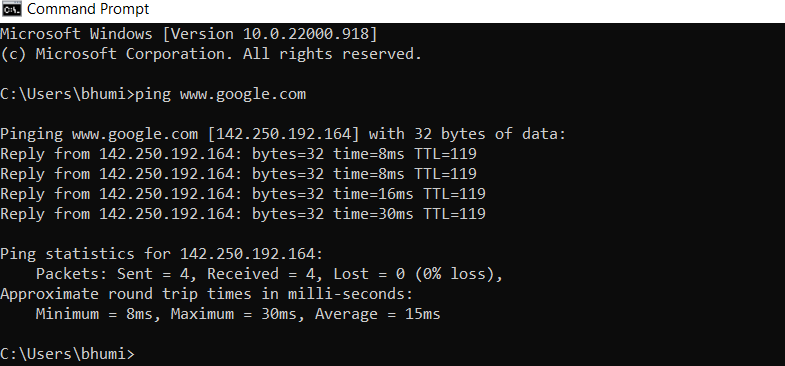
## nslookup -type=mx google.com:

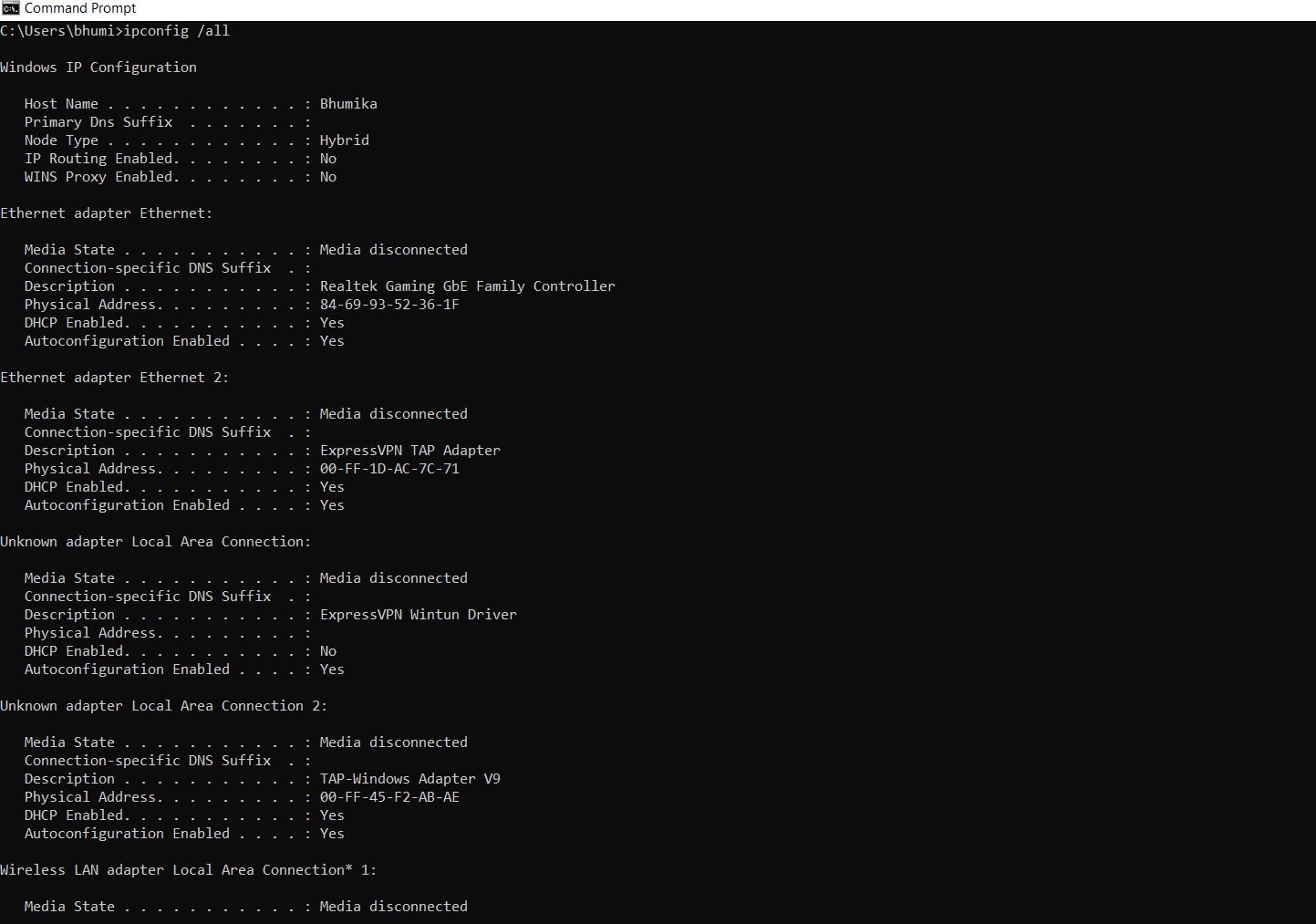
Lookup for an mx record MX (Mail Exchange) record maps a domain name to a list of mail exchange servers for that domain. The MX record tells that all the mails sent to “google.com” should be routed to the Mail server in that domain.

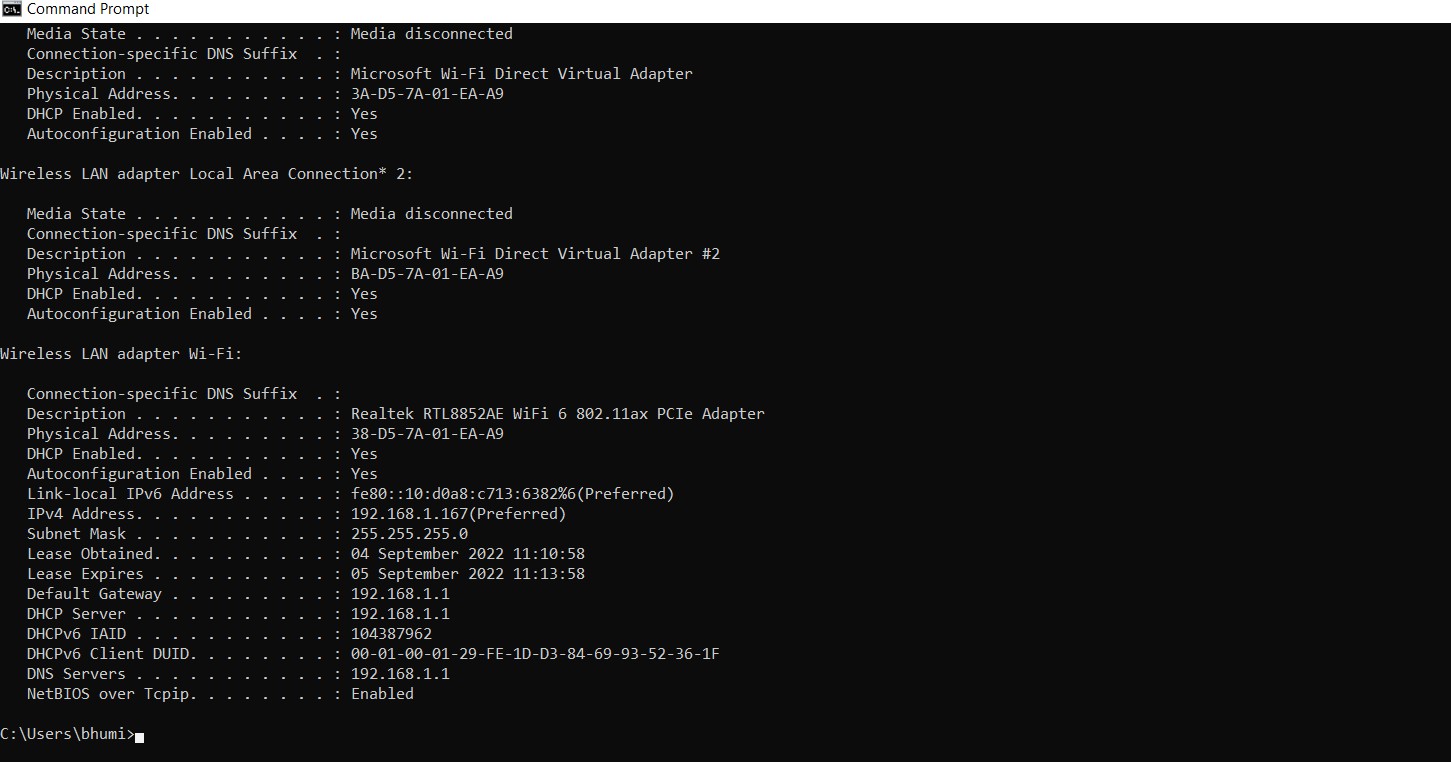
## nslookup -type=txt google.com:

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

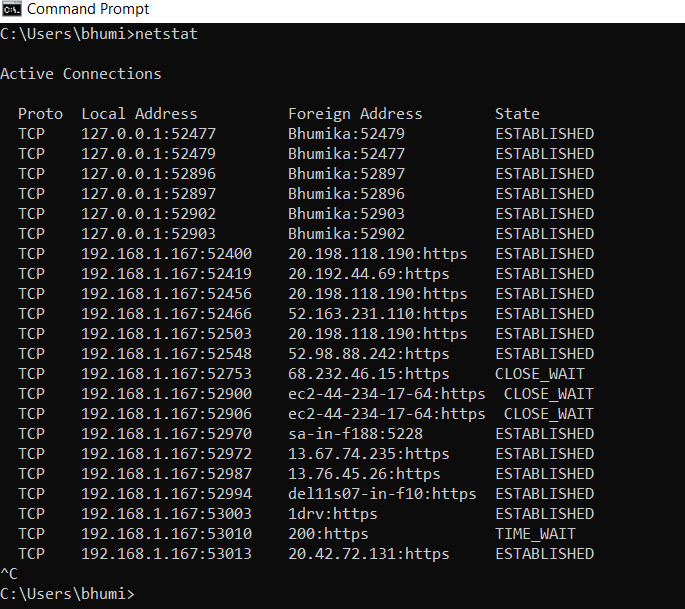
## Outputs: -

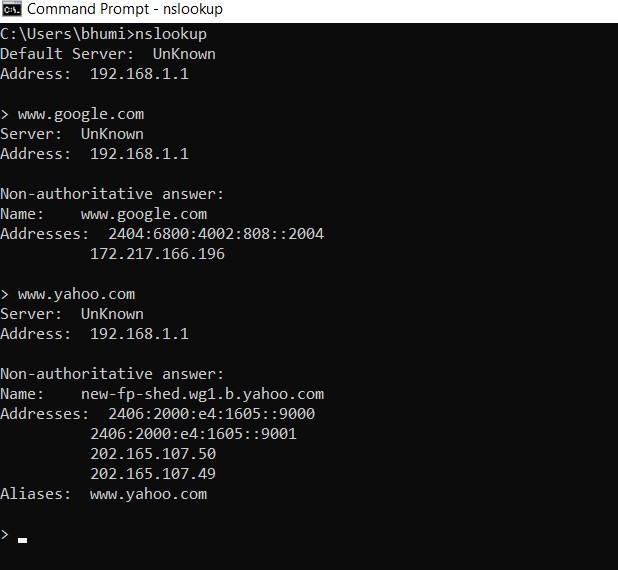












1. **ping command Example:**

## >> ping [www.google.com](http://www.google.com/)

**>> ping** [**www.niet.co.in**](http://www.niet.co.in/)

## >> ping 192.168.4.1

1. **ipconfig command**

## >> ipconfig

**>> ipconfig/all**

## tracert command Example:

**>> tracert 192.168.0.18**

## >>tracert yahoo.com

1. **netstat command**

## >> netstat-n

**>> netstat-a**

## >> netstat-?

1. **nslookup command Example:**

## >> nslookup google.com

>> **nslookup niet.co.in Result: -**

All the required network commands are studied and successfully test the network connection using ping command and use of ipconfig, netstat and tracert command provided by TCP/IP.

# Experiment No.-5

**Aim -** Implementation of CRC algorithm in any language like C++ , Java or Python.

# Code-

#------ Convert string data to binary string data----------

input\_string = "EVN"

data = (''.join(format(ord(x), 'b') for x in input\_string)) print (data)

#-------Sender Side--------

import socket def xor(a, b):

result = []

for i in range(1, len(b)):

if a[i] == b[i]:

result.append('0')

else:

result.append('1')

return ''.join(result)

def mod2div(divident, divisor): pick = len(divisor)

tmp = divident[0 : pick] while pick < len(divident):

if tmp[0] == '1':

tmp = xor(divisor, tmp) + divident[pick]

l\_key = len(key)

appended\_data = data + '0'\*(l\_key-1) remainder = mod2div(appended\_data, key) codeword = data + remainder

return codeword s = socket.socket()

port = 12345 s.connect(('127.0.0.1', port))

input\_string = input("Enter data you want to send->") data =(''.join(format(ord(x), 'b') for x in input\_string)) print("Entered data in binary format :",data)

key = "1001"

ans = encodeData(data,key)

print("Encoded data to be sent to server in binary format :",ans) s.sendto(ans.encode(),('127.0.0.1', 12345))

print("Received feedback from server :",s.recv(1024).decode()) s.close()

#-------Receiver Side-------

import socket def xor(a, b):

result = []

for i in range(1, len(b)):

if a[i] == b[i]:

result.append('0')

else:

result.append('1')

return ''.join(result)

def mod2div(divident, divisor): pick = len(divisor)

tmp = divident[0: pick] while pick < len(divident):

if tmp[0] == '1':

tmp = xor(divisor, tmp) + divident[pick]

else:

tmp = xor('0'\*pick, tmp) + divident[pick]

pick += 1 if tmp[0] == '1':

tmp = xor(divisor, tmp)

else:

tmp = xor('0'\*pick, tmp)

checkword = tmp return checkword

def decodeData(data, key):

l\_key = len(key)

appended\_data = data.decode() + '0'\*(l\_key-1) remainder = mod2div(appended\_data, key) return remainder

s = socket.socket()

print("Socket successfully created") port = 12345

s.bind(('', port))

print("socket binded to %s" % (port)) s.listen(5)

print("socket is listening") while True:

c, addr = s.accept()

print('Got connection from', addr) data = c.recv(1024)

print("Received encoded data in binary format :", data.decode()) if not data:

break key = "1001"

ans = decodeData(data, key) print("Remainder after decoding is->"+ans) temp = "0" \* (len(key) - 1)

if ans == temp:

c.sendto(("THANK you Data ->"+data.decode() +

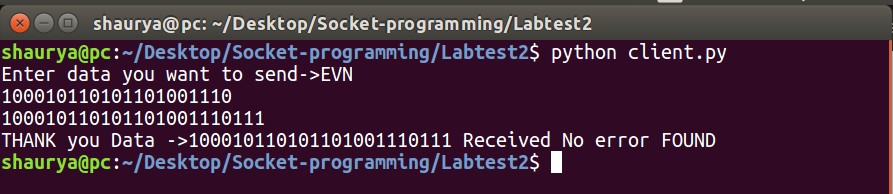
" Received No error FOUND").encode(), ('127.0.0.1', 12345))

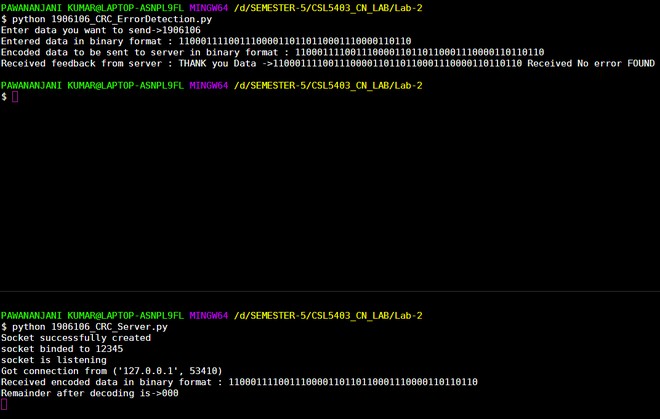
else:

c.sendto(("Error in data").encode(), ('127.0.0.1', 12345))

c.close()

**Input and Output-**





# Result-

The Implementation of CRC algorithm is done successfully.

# Experiment No.-6

**Aim -** Implementation of Hamming code (7, 4) code to limit the noise in any language like C++,

Java or Python

## Theory –

Let n=1010 and consider ODD parity error control protocol

Part-1. Calculation of Hamming code (7, 4) construction at the transmitter end. Part-2. Detection and correction of Hamming code (7, 4) at the receiver end.

## Code-

option=int(input('Press 1 for generating hamming code \nPress 2 for finding error in hamming code\n\t Enter your choice:--\n'))

if(option==1): # GENERATE HAMMING CODE

print('Enter the data bits') d=input()

data=list(d) data.reverse() c,ch,j,r,h=0,0,0,0,[]

while ((len(d)+r+1)>(pow(2,r))): r=r+1

for i in range(0,(r+len(data))): p=(2 c)

if(p==(i+1)): h.append(0) c=c+1

else:

h.append(int(data[j])) j=j+1

for parity in range(0,(len(h))): ph=(2 ch) if(ph==(parity+1)):

startIndex=ph-1 i=startIndex toXor=[]

while(i<len(h)):

block=h[i:i+ph] toXor.extend(block) i+=2\*ph

for z in range(1,len(toXor)): h[startIndex]=h[startIndex]^toXor[z]

ch+=1

h.reverse()

print('Hamming code generated would be:- ', end="") print(int(''.join(map(str, h))))

elif(option==2): # DETECT ERROR IN RECEIVED HAMMING CODE

print('Enter the hamming code received') d=input()

data=list(d) data.reverse()

c,ch,j,r,error,h,parity\_list,h\_copy=0,0,0,0,0,[],[],[]

for k in range(0,len(data)): p=(2 c) h.append(int(data[k])) h\_copy.append(data[k]) if(p==(k+1)):

c=c+1

for parity in range(0,(len(h))): ph=(2 ch) if(ph==(parity+1)):

startIndex=ph-1 i=startIndex toXor=[]

while(i<len(h)): block=h[i:i+ph] toXor.extend(block) i+=2\*ph

for z in range(1,len(toXor)): h[startIndex]=h[startIndex]^toXor[z]

parity\_list.append(h[parity]) ch+=1

parity\_list.reverse()

error=sum(int(parity\_list) \* (2 i) for i, parity\_list in enumerate(parity\_list[::-1]))

if((error)==0):

print('There is no error in the hamming code received')

elif((error)>=len(h\_copy)): print('Error cannot be detected')

else:

print('Error is in',error,'bit')

if(h\_copy[error-1]=='0'): h\_copy[error-1]='1'

elif(h\_copy[error-1]=='1'): h\_copy[error-1]='0'

print('After correction hamming code is:- ') h\_copy.reverse()

print(int(''.join(map(str, h\_copy))))

else:

print('Option entered does not exist')

## Input and Output-



**Result-**

The Implementation of Hamming code (7, 4) transmitter and receiver end is done successfully.

# Experiment No.- 7

**Aim -** Implementation of stop and wait protocol in any language like C++ , Java or Python

## Theory –

As name stop and wait suggest the data that sender wants to send, he sends the data to the receiver. After sending the data, he stops and waits until he receives the acknowledgment from the receiver. The stop and wait protocol is a flow control protocol where flow control is one of the services of the data link layer.

Primitives of stop and wait protocol are:

1. Sender side :-

Rule 1: Sender sends one data packet at a time.

Rule 2: Sender sends the next packet only when it receives the acknowledgment of the previous packet.

1. Receiver side

Rule 1: Receive and then consume the data packet.

Rule 2: When the data packet is consumed, receiver sends the acknowledgment to the sender.

## Code-

import \_thread import random import string

flag = 1 data = '' i = 0

def sender(a): global flag global data global i file\_create() while(True):

if data[i] == '1': break

if flag == 1:

print("Sending : ", data[i]) i += 1

flag = 0

def receiver(a): global flag

global i

while(True): if flag == 0:

print("Recieved : ", data[i-1]) flag = 1

if data[i] == '1': break

def file\_create():

file = open('prac4\_input.txt', 'w') n = random.randint(10, 20)

res = ''.join(random.choices(string.ascii\_uppercase, k=n)) res = res + '1'

file.write(res) # print("hi")

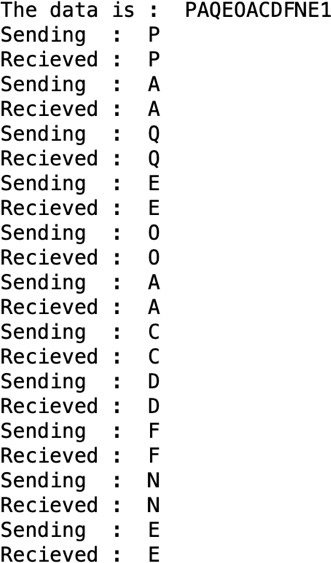
if name == ' main ': file\_create()

file = open("Prac4\_input.txt", "r") data = file.readline()

print("The data is : ", data) a = 0

t1 = \_thread.start\_new\_thread(sender, (a,)) t2 = \_thread.start\_new\_thread(receiver, (a,))

## Input and Output-



**Result-**

The Implementation of stop and wait protocol is done successfully.

# Experiment No.- 8

**Aim -**Write a program in java to find the IP address of the any site if name is given.

# Code-

import java.net. \*; import java. \*;

class GFG {

public static void main (String args[]) throws Unknown Host Exception

{

String s = "https:// [www.google.com/](http://www.google.com/)";

try {

InetAddress ip = InetAddress.getByName(new URL(s).getHost());

System.out.println("Public IP Address of: " + ip);

}

catch (MalformedURLException e) { System.out.println("Invalid URL");

}

}

}

**Input and Output-**



# Result-

The IP address of the [www.google.com](http://www.google.com/) site is found successfully.

# Experiment No.- 9

**Aim -** Write a program in java to find the IP address of the system.

# Code-

import java.net.\*; import java.io.\*; import java.util.\*;

import java.net.InetAddress;

public class JavaProgram

{

public static void main(String args[]) throws Exception

{

InetAddress localhost = InetAddress.getLocalHost(); System.out.println("System IP Address : " +

(localhost.getHostAddress()).trim());

String systemipaddress = ""; try

{

URL url\_name = new URL("[http://bot.whatismyipaddress.com](http://bot.whatismyipaddress.com/)");

BufferedReader sc =

new BufferedReader(new InputStreamReader(url\_name.openStream()));

systemipaddress = sc.readLine().trim();

}

catch (Exception e)

{

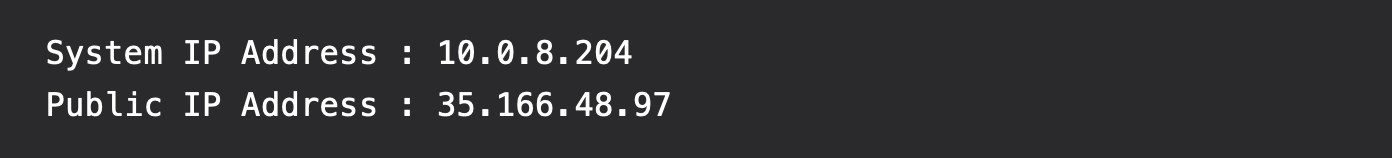
systemipaddress = "Cannot Execute Properly";

}

System.out.println("Public IP Address: " + systemipaddress +"\n");

}

**Input and Output-**



# Result-

The IP address of the system is found successfully.

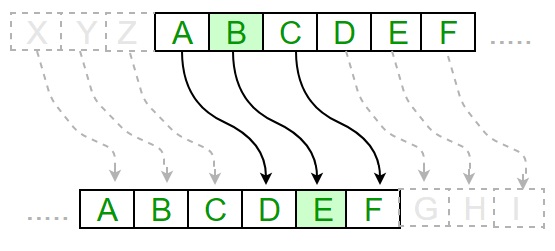
# Experiment No.- 10

**AIM-**Implementation of Ceaser Cipher and RSA algorithm in any language like C++. Java or Python and explain the RSA algorithm.

1,Encrypting with the Caesar Cipher

In the Caesar cipher, the key is a number from 0 to 25, because there are 26 letters in the alphabet. This means that for any given message, there are 26 different ways we can encrpyt the message.

For each letter, the key determines **which letter is replacing the current letter, by counting down the alphabet**. In the following example, let’s say we wanted to encrypt the letter B with a key of 3, we would find the 3rd letter that appears after B - which is C, D, then finally E.



Encryption:

C=(P+K)MOD 26

Decryption:

P=(C-K)MOD 26

CODE:

A python program to illustrate Caesar Cipher Technique

def encrypt(text,s):

result = ""

# traverse text

for i in range(len(text)):

char = text[i]

# Encrypt uppercase characters

if (char.isupper()):

result += chr((ord(char) + s-65) % 26 + 65)

# Encrypt lowercase characters

else:

result += chr((ord(char) + s - 97) % 26 + 97)

return result

#check the above function

text = "ATTACKATONCE"

s = 4

print ("Text : " + text)

print ("Shift : " + str(s))

print ("Cipher: " + encrypt(text,s))

2.RSA Algorithm

The RSA algorithm is a widely used public-key encryption algorithm named after its inventors Ron Rivest, Adi Shamir, and Leonard Adleman. It is based on the mathematical concepts of prime factorization and modular arithmetic.

The algorithm for RSA is as follows:

1. Select 2 prime numbers, preferably large, p and q.
2. Calculate n = p\*q.
3. Calculate phi(n) = (p-1)\*(q-1)
4. Choose a value of e such that 1<e<phi(n) and gcd(phi(n), e) = 1.
5. Calculate d such that d = (e^-1) mod phi(n).

Here the public key is {e, n} and private key is {d, n}.

If M is the plain text then the cipher text C = (M^e) mod n.

This is how data is encrypted in RSA algorithm. Similarly, for decryption, the plain text M = (C^d) mod n.

**Example:** Let p=3 and q=11 (both are prime numbers).

* Now, n = p\*q = 3\*11 = 33
* phi(n) = (p-1)\*(q-1) = (3-1)\*(11-1) = 2\*10 = 20
* Value of e can be 7 since 1<7<20 and gcd(20, 7) = 1.
* Calculating d = 7^-1 mod 20 = 3.
* Therefore, public key = {7, 33} and private key = {3, 33}.

Suppose our message is M=31. You can encrypt and decrypt it using the RSA algorithm as follows:

**Encryption:** C = (M^e) mod n = 31^7 mod 33 = 4

**Decryption:** M = (C^d) mod n = 4^3 mod 33 = 31

Since we got the original message that is plain text back after decryption, we can say that the algorithm worked correctly.

Program:

**import** math

# step 1

p **=** 3

q **=** 7

# step 2

n **=** p**\***q

print("n =", n)

# step 3

phi **=** (p**-**1)**\***(q**-**1)

# step 4

e **=** 2

**while**(e<phi):

**if** (math.gcd(e, phi) **==** 1):

**break**

**else**:

        e **+=** 1

print("e =", e)

# step 5

k **=** 2

d **=** ((k**\***phi)**+**1)**/**e

print("d =", d)

print(f'Public key: {e, n}')

print(f'Private key: {d, n}')

# plain text

msg **=** 11

print(f'Original message:{msg}')

# encryption

C **=** pow(msg, e)

C **=** math.fmod(C, n)

print(f'Encrypted message: {C}')

# decryption

M **=** pow(C, d)

M **=** math.fmod(M, n)

print(f'Decrypted message: {M}')

**OUTPUT**

n = 21

e = 5

d = 5.0

Public key: (5, 21)

Private key: (5.0, 21)

Original message:11

Encrypted message: 2.0

Decrypted message: 11.0

**Experiment No.-11**

**AIM-** Introduction to CISCO Packet Tracer. Design Bus, star, mesh, ring topology and check the connectivity using ping command.

**Introduction:**

Cisco Packet Tracer is Cisco's simulation software. It can be used to create complicated network typologies, as well as to test and simulate abstract networking concepts. It acts as a playground for you to explore networking and the experience is very close to what you see in computer networks.

**Simulation:**

A **network simulator** is a [software](https://en.wikipedia.org/wiki/Software) program that can predict the performance of a computer network or a wireless communication network. Since communication networks have become too complex for traditional analytical methods to provide an accurate understanding of system behavior, network simulators are used.

**List of Simulators in Market:**

* Ns2 (Network Simulator 2).
* Ns3 (Network Simulator 3).
* OPNET.
* OMNeT++.
* NetSim.
* REAL.
* QualNet.
* J-Sim.

**Use of Cisco packet tracer:**

It is basically used for helping students learn the principles of networking with hands-on experience so that they can develop skills related to Cisco technologies. It is basically used for helping students learn the principles of networking with hands-on experience so that they can develop skills related to Cisco technologies.

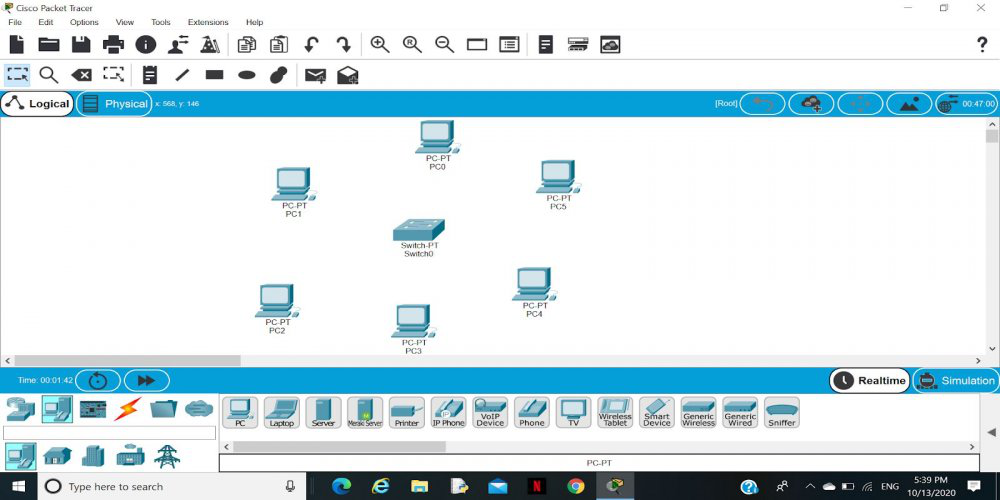
**Topology:**

A Network Topology is the arrangement with which computer systems or network devices are connected to each other.

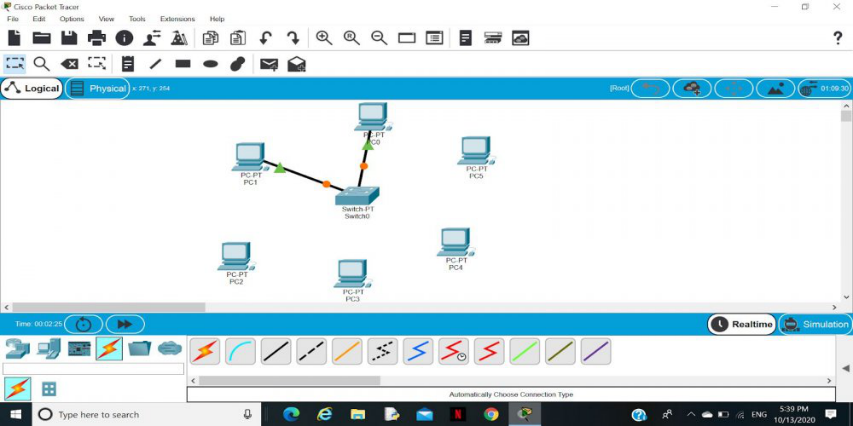
**Star Topology:**

A star topology for a Local Area Network (LAN) is one in which each node is connected to a central connection point, such as a hub or switch. Whenever a node tries to connect with another node then the transmission of the message must be happening with the help of the central node. The best part of star topology is the addition and removal of the node in the network but too many nodes can cause suffering to the network.

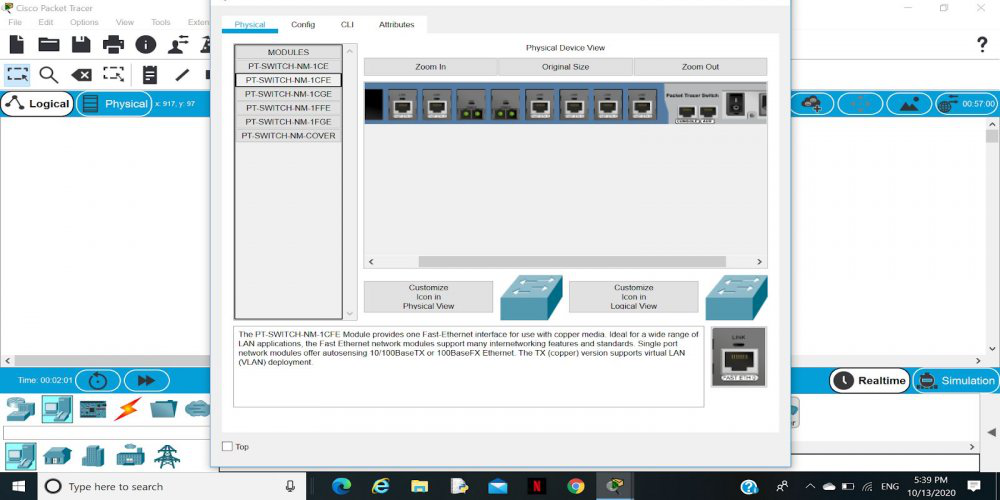
**Step 1:**We have taken a switch and linked it to six end devices.



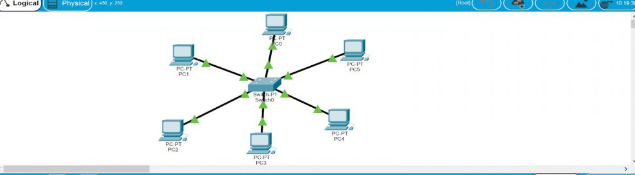
**Step 2:** Link every device with the switch.

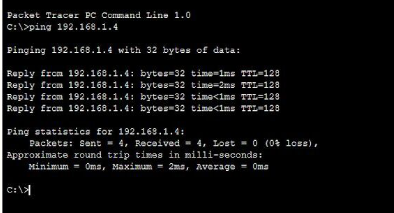


**Step 3:** Provide the IP address to each device.



**Step 4:**Transfer message from one device to another and check the Table for Validation.





**Ring Topology:**

Ring topology is a kind of arrangement of the network in which every device is linked with two other devices. This makes a circular ring of interconnected devices which gives it its name. Data is usually transmitted in one direction along the ring, known as a unidirectional ring. The data is delivered from one device to the next until it reaches the decided destination. In a bidirectional ring, data can travel in either direction.

### Steps to Configure and Setup Ring Topology in Cisco Packet Tracer :

**Step 1:** First, open the cisco packet tracer desktop and select the devices given below:

| **S.NO** | **Device** | **Model Name** |
| --- | --- | --- |
| **1.** | PC | PC |
| **2.** | Switch | PT-Switch |

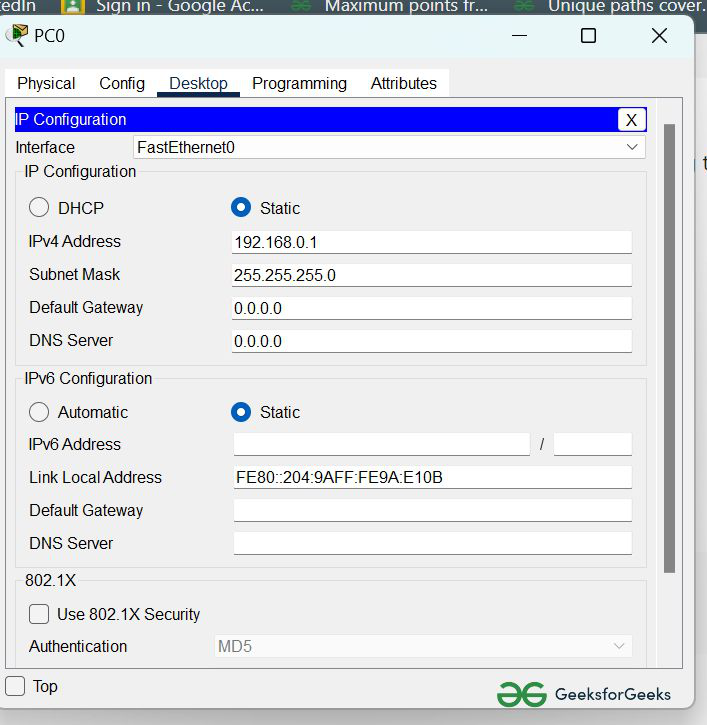
**IP Addressing Table**

| **S.NO** | **Device** | **IPv4 Address** | **Subnet Mask** |
| --- | --- | --- | --- |
| **1.** | **pc0** | 192.168.0.1 | 255.255.255.0 |
| **2.** | **pc1** | 192.168.0.2 | 255.255.255.0 |
| **3.** | **pc2** | 192.168.0.3 | 255.255.255.0 |
| **4.** | **pc3** | 192.168.0.4 | 255.255.255.0 |

* Then, create a network topology as shown below the image.
* Use an Automatic connecting cable to connect the devices with others

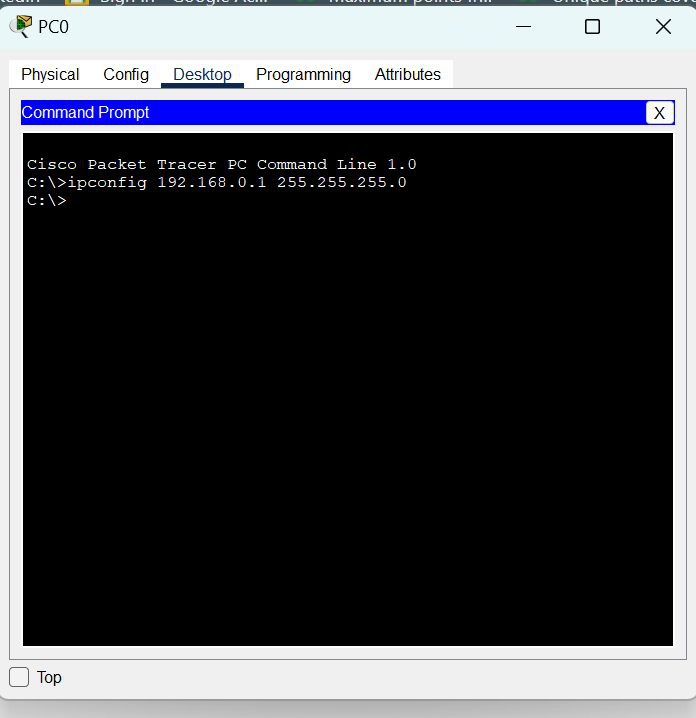
**Step 2:** Configure the PCs (hosts) with IPv4 address and Subnet Mask according to the IP addressing table given above.

* To assign an IP address in PC0, click on PC0.
* Then, go to desktop and then IP configuration and there you will IPv4 configuration.
* Fill IPv4 address and subnet mask.



* Assigning IP address using the ipconfig command, or we can also assign an IP address with the help of a command.
* Go to the command terminal of the PC.
* Then, type ipconfig <IPv4 address><subnet mask><default gateway>(if needed)

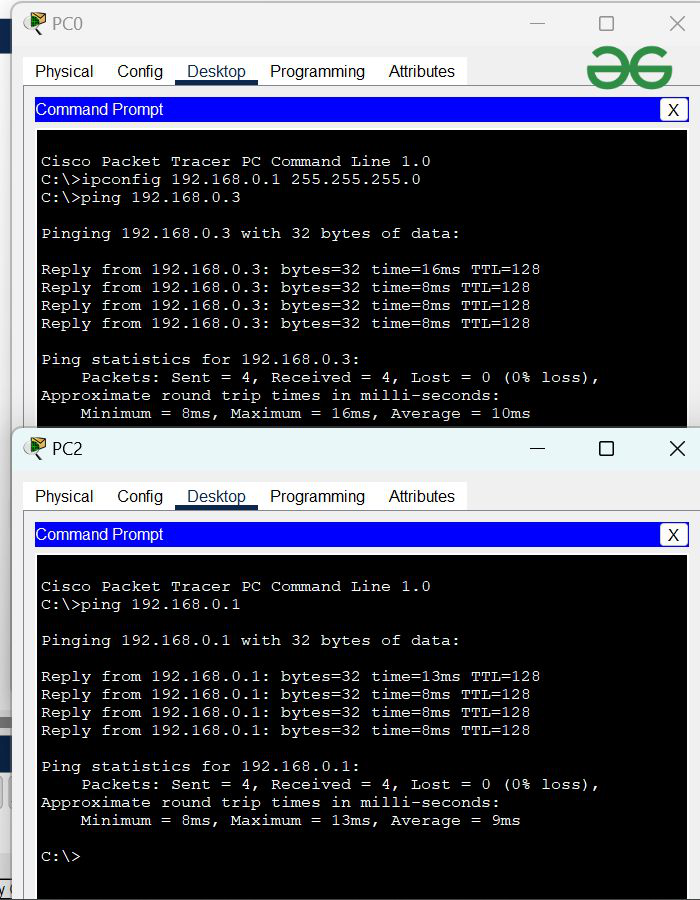
Example: ipconfig 192.168.0.1 255.255.255.0



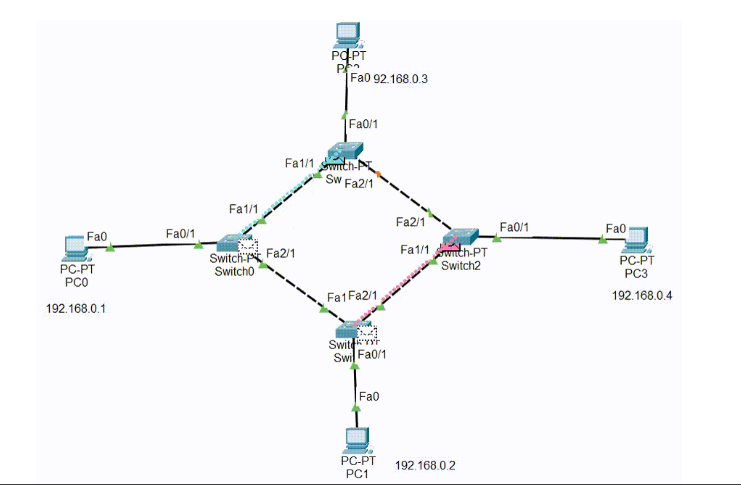
* Repeat the same procedure with other PCs to configure them thoroughly.

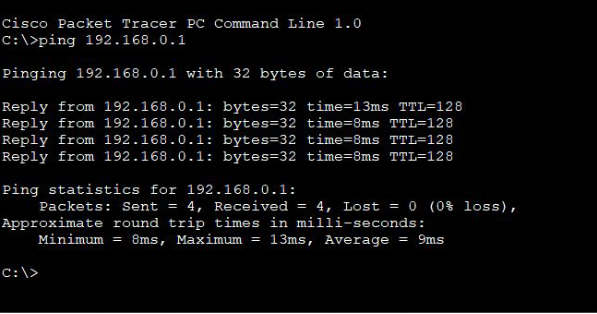
**Step 3:** Verify the connection by pinging the IP address of any host in PC0.

* Use the ping command to verify the connection.
* As we can see we are getting replies from a targeted node on both PCs.
* Hence the connection is verified.



* A simulation of the experiment is given below we have sent two PDU packets one targeted from PC0 to PC2 and another targeted from PC1 to PC3.





**Bus Topology:**

A bus topology is a network in which nodes are directly linked with a common half-duplex link. A host on a bus topology is called a station. In a bus network, every station will accept all network packets, and these packets generated by each station have equal information priority. A bus network includes a single network segment and collision domain.

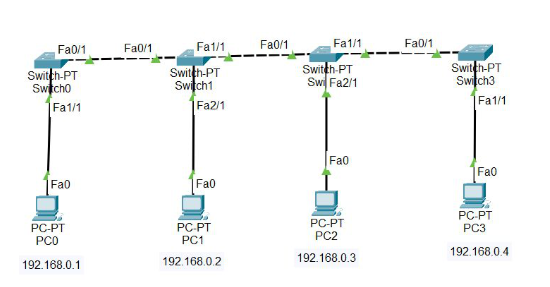
**Step 1:** First, open the cisco packet tracer desktop and select the devices given below:

| **S.NO** | **Device** | **Model-Name** |
| --- | --- | --- |
| **1.** | PC | PC |
| **2.** | Switch | PT-Switch |

**IP Addressing Table**

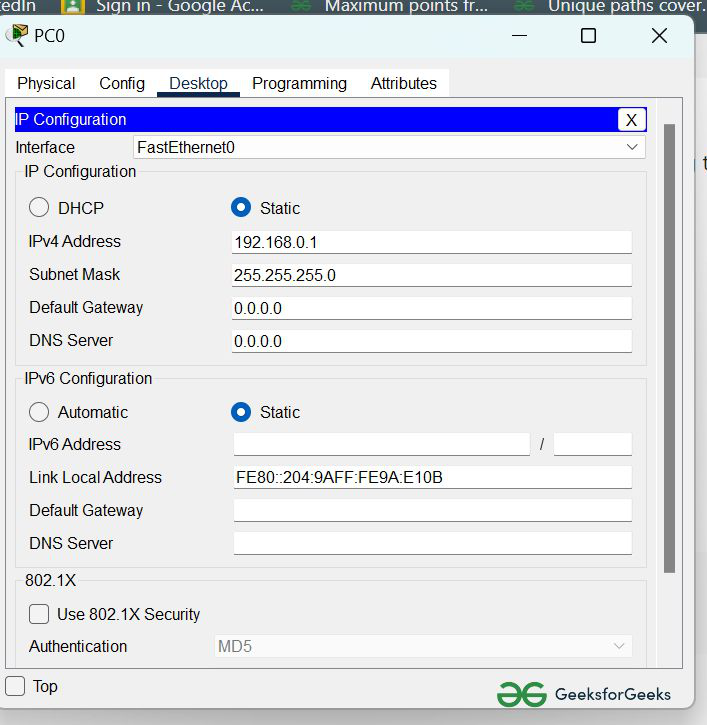
| **S.NO** | **Device** | **IPv4 Address** | **Subnet Mask** |
| --- | --- | --- | --- |
|  | **pc0** | 192.168.0.1 | 255.255.255.0 |
|  | **pc1** | 192.168.0.2 | 255.255.255.0 |
|  | **pc2** | 192.168.0.3 | 255.255.255.0 |
|  | **pc3** | 192.168.0.4 | 255.255.255.0 |

* Then, create a network topology as shown below image:
* Use an Automatic connecting cable to connect the devices with others.



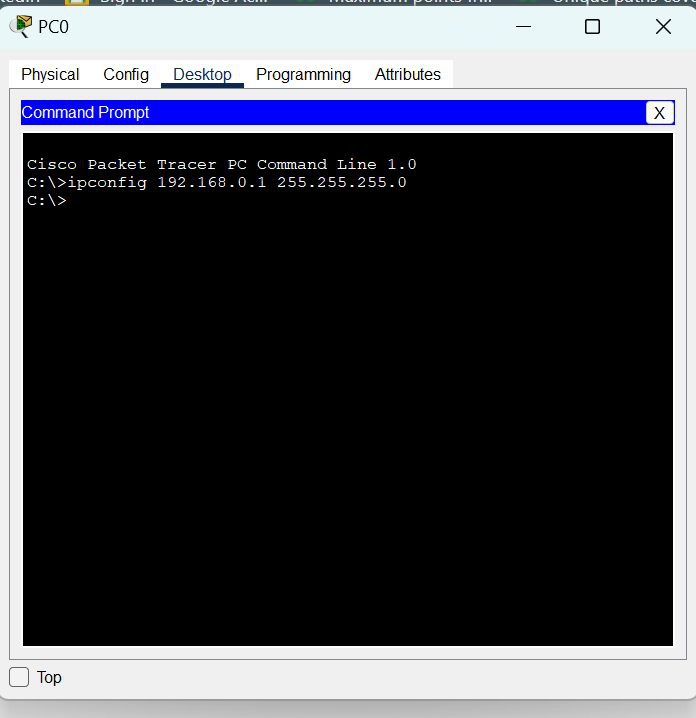
**Step 2:** Configure the PCs (hosts) with IPv4 address and Subnet Mask according to the IP addressing table given above.

* To assign an IP address in PC0, click on PC0.
* Then, go to desktop and then IP configuration and there you will IPv4 configuration.
* Fill IPv4 address and subnet mask.



* Assigning an IP address using the ipconfig command, or we can also assign an IP address with the help of a command.
* Go to the command terminal of the PC.
* Then, type ipconfig <IPv4 address><subnet mask><default gateway>(if needed)

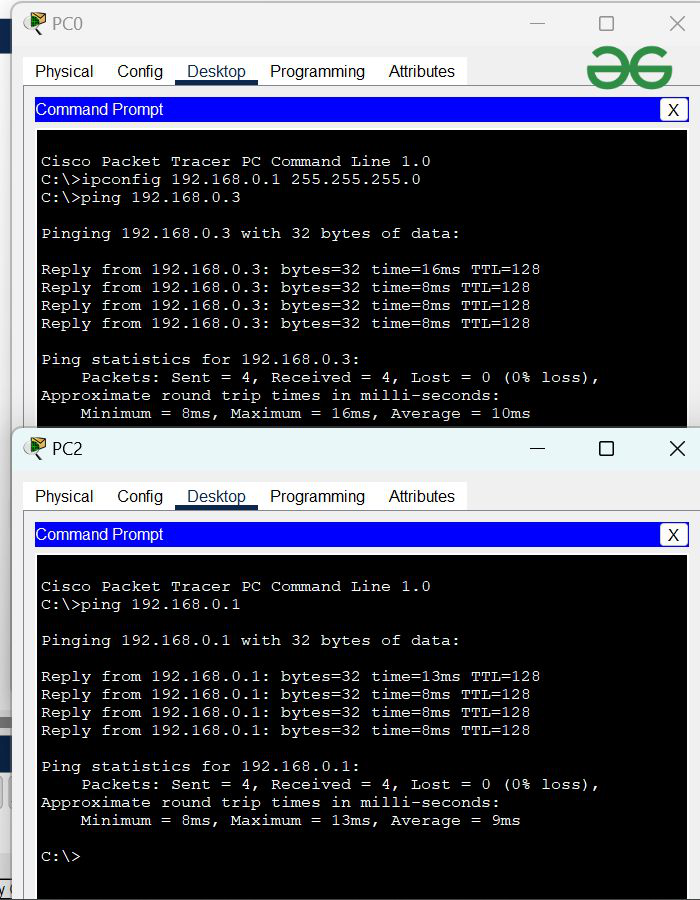
Example: ipconfig 192.168.0.1 255.255.255.0

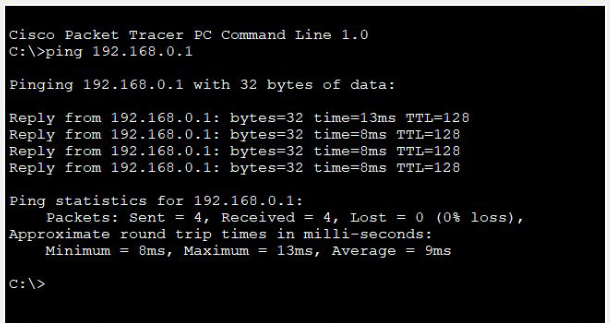


* Repeat the same procedure with other PCs to configure them thoroughly.

**Step 3:** Verify the connection by pinging the IP address of any host in PC0.

* Use the ping command to verify the connection.
* As we can see we are getting replies from a targeted node on both PCs.
* Hence the connection is verified.





**Mesh Topology:**

In the mesh topology of networking, each and every device sends its own signal to the other devices that are present in the arrangement of the network.

**Step 1:**First, open the Cisco packet tracer desktop and select the devices given below:

| **S.NO** | **Device** | **Model name** |
| --- | --- | --- |
| **1.** | PC | PC |
| **2.** | Switch | PT-switch |

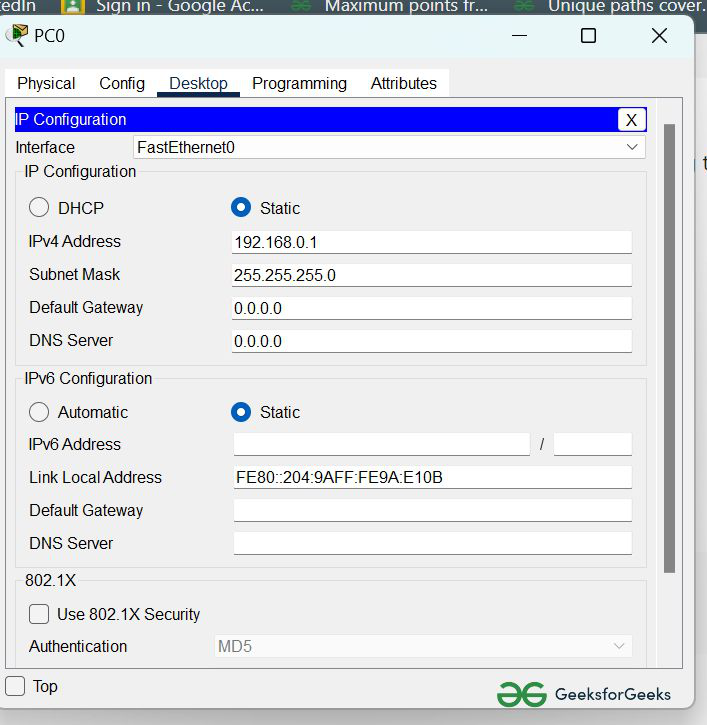
**IP Addressing Table:**

| **S.NO** | **Device** | **IPv4 Address** | **Subnet Mask** |
| --- | --- | --- | --- |
| **1.** | pc0 | 192.168.0.1 | 255.255.255.0 |
| **2.** | pc1 | 192.168.0.2 | 255.255.255.0 |
| **3.** | pc2 | 192.168.0.3 | 255.255.255.0 |
| **4.** | pc3 | 192.168.0.4 | 255.255.255.0 |

* Then, create a network topology as shown below the image.
* Use an Automatic connecting cable to connect the devices with others.

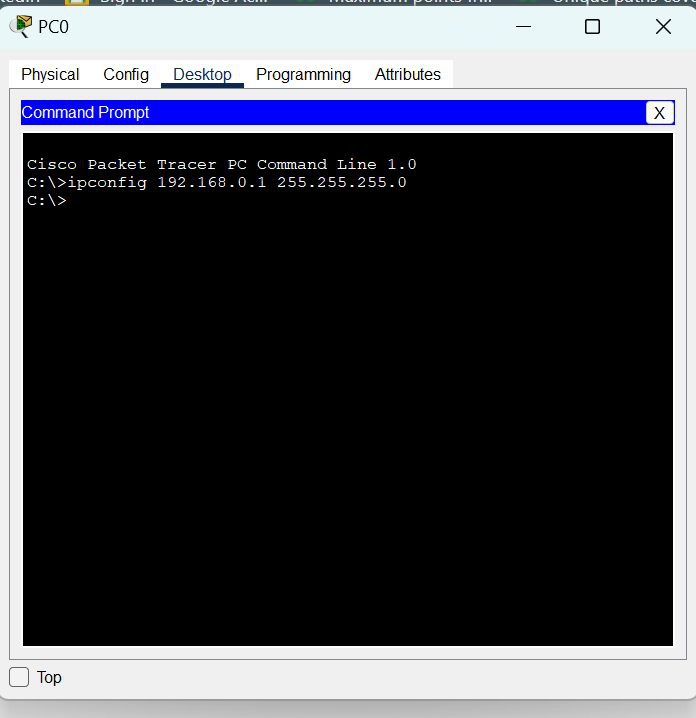
**Step 2:** Configure the PCs (hosts) with IPv4 address and Subnet Mask according to the IP addressing table given above.

* To assign an IP address in PC0, click on PC0.
* Then, go to desktop and then IP configuration and there you will IPv4 configuration.
* Fill IPv4 address and subnet mask.



* Assigning IP address using the ipconfig command.
* Also, we can also assign an IP address with the help of a command.
* Go to the command terminal of the PC.
* Then, type ipconfig <IPv4 address><subnet mask><default gateway>(if needed)

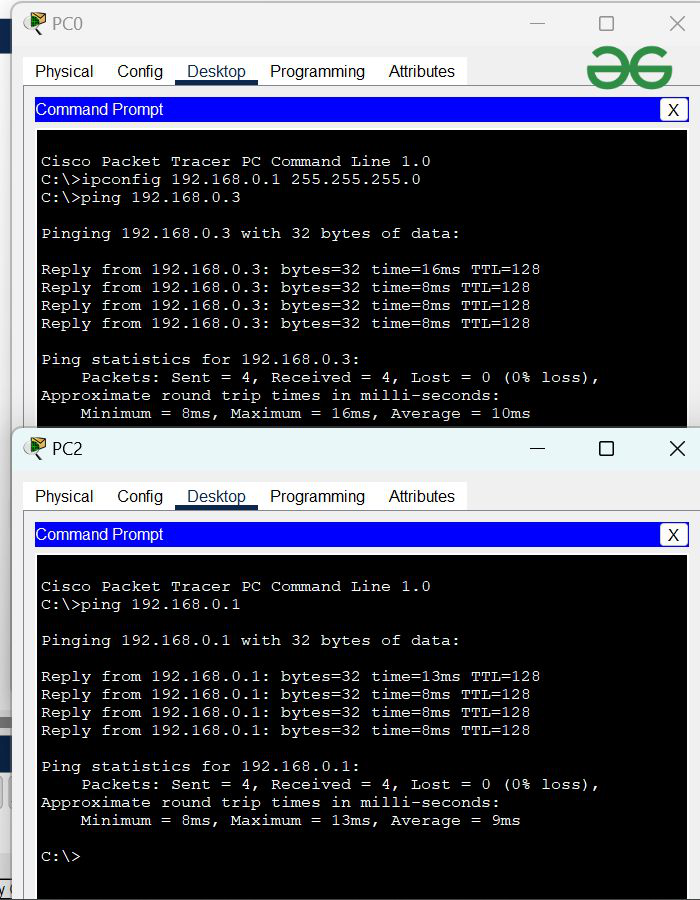
Example: ipconfig 192.168.0.1 255.255.255.0

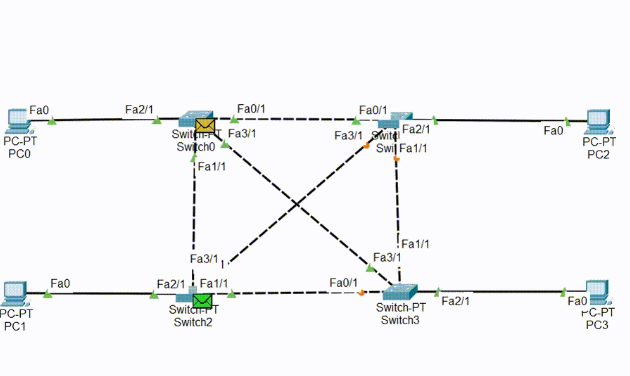


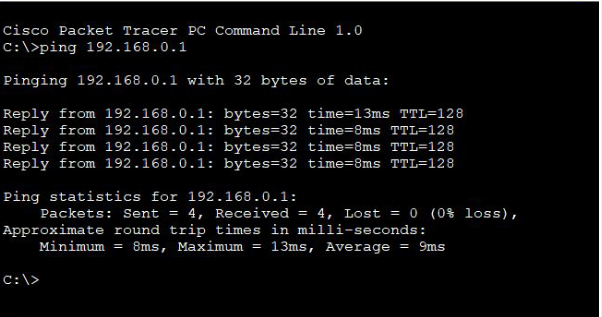
* Repeat the same procedure with other PCs to configure them thoroughly.

**Step 3:** Verify the connection by pinging the IP address of any host in PC0.

* Use the ping command to verify the connection.
* We will check if we are getting any replies or not.
* Here we get replies from a targeted node on both PCs.
* Hence the connection is verified.







**Experiment No.-12**

**AIM**-Switch Configuration on CISCO packet tracer using CLI

The switch is a network device that is used to segment the networks into different subnetworks called subnets or LAN segments. It is responsible for filtering and forwarding the packets between LAN segments based on the MAC address.

In this article, we will see the switch configuration in the Cisco packet tracer.

**Steps to Configure the Switch:**

**Step 1.** Open the packet tracer desktop and take a switch (PT-Switch) from the devices.



**Step 2:** Configure the Host name of the swicth0.

* Click on switch0 and go to Command Line Interface.
* Then change the hostname to “sh”

**Command:**

switch>

switch>en

switch#conf t

switch(config)#hostname sh

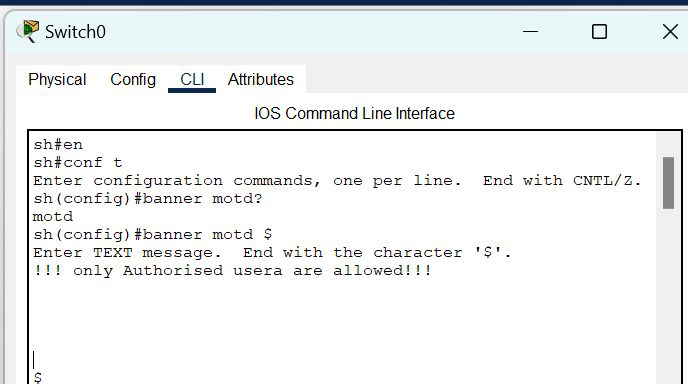
sh(config)exit

**Step 3:** Set a message of the day (MOTD) banner for the users.

**Command:**

sh(config)#banner motd $

* Then, enter MOTD and end it with ‘$’ to exit.



**Step 4:** Set up line control password and enable secret password.

To configure the Line Control password and Enable secret follow the below commands:

sh#conf t

sh(config)#

sh(config)#line con 0

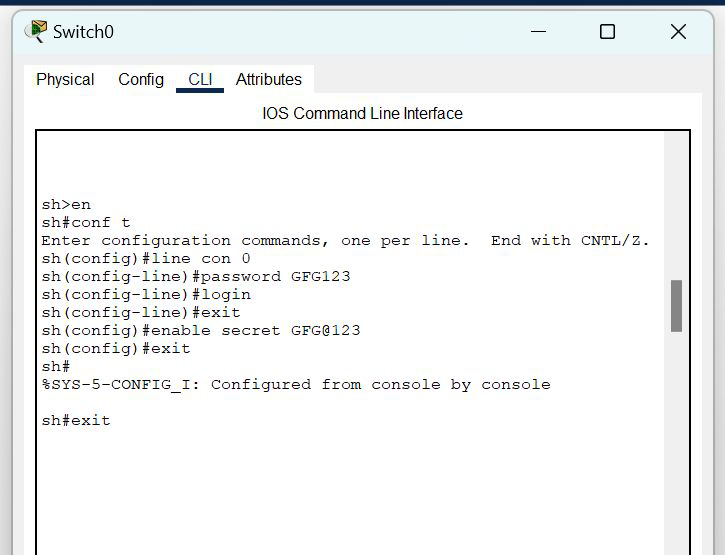
sh(config-line)#password GFG123

sh(config-line)#login

sh(config-line)#exit

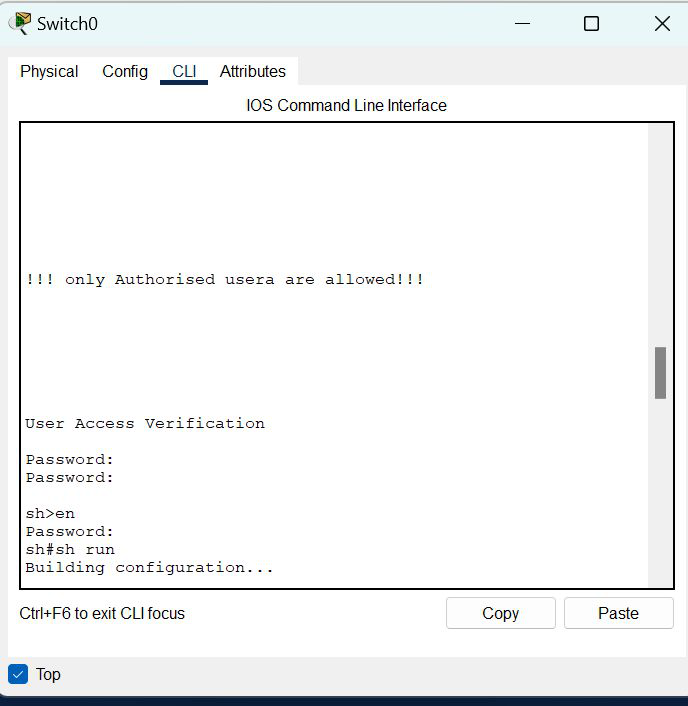
sh(config)#enable secret GFG@123

sh(config)#exit



**Step 5:** Verify the password

* When you try to log in first, it will ask for the **line control password.**
* Then, to configure the terminal it will ask to **enable a secret password.**



 To save the configuration use the below command:

**Command:**

sh#copy run startup-config