**GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING**

**(AUTONOMOUS)**

**Madhurawada,Visakhapatnam-530048**



**CERTIFICATE**

Certified that this is a bonafide record of practical work done by **EARLE HARSHITH NAIDU**, Roll no:**322103310062** of B.Tech Vth Semester in the **Computer Networks Lab**, in the Department of **Computer Science and Engineering** during the academic year **2024-25**.

No.of Experiements done:12

Name of the faculty

**Dr.CH.SITA KUMARI**

(Associate Professor)

|  |  |  |
| --- | --- | --- |
| **S NO** | **AIM OF PROGRAM** | **PG NO** |
| 1 | Installation of Simulation Tools (packet tracer and NS2) | 3-8 |
| 2 | Configure the LAN using packet tracer | 9-16 |
| 3 | Design a sample topology using the NS2 Simulation Tool | 17-30 |
| 4 | Implement CRC and Hamming code for error handling | 31-33 |
| 5 | Implementation of stop and wait protocol | 34-35 |
| 6 |  |  |
| 7 |  |  |
| 8 |  |  |
| 9 |  |  |
| 10 |  |  |
| 11 |  |  |
| 12 |  |  |

**WEEK-1**

**AIM :** Installation of Simulation Tools (Packet tracer and NS2).

**CISCO PACKET TRACER INSTALLATION:**

**PROCEDURE :**

### Step 01 :

Go to the official Cisco Netacad website. Once you’re logged in click on “RESOURCES > DOWNLOAD PACKET TRACER”.

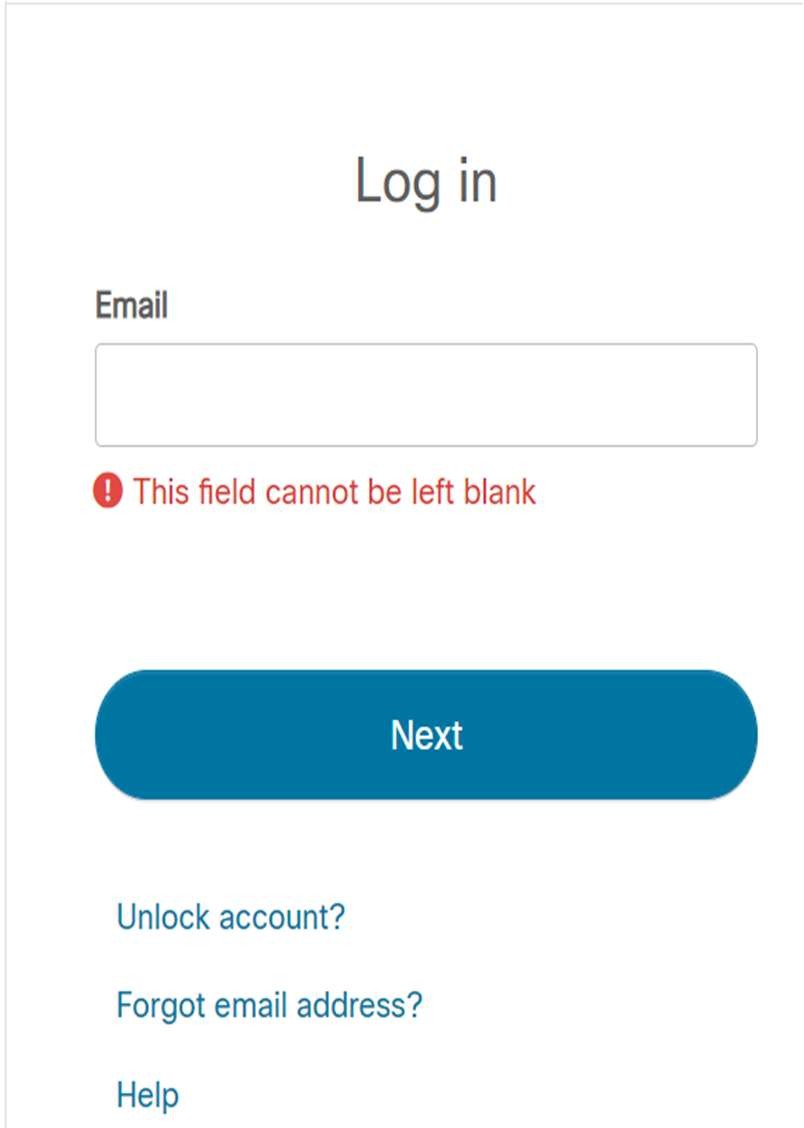
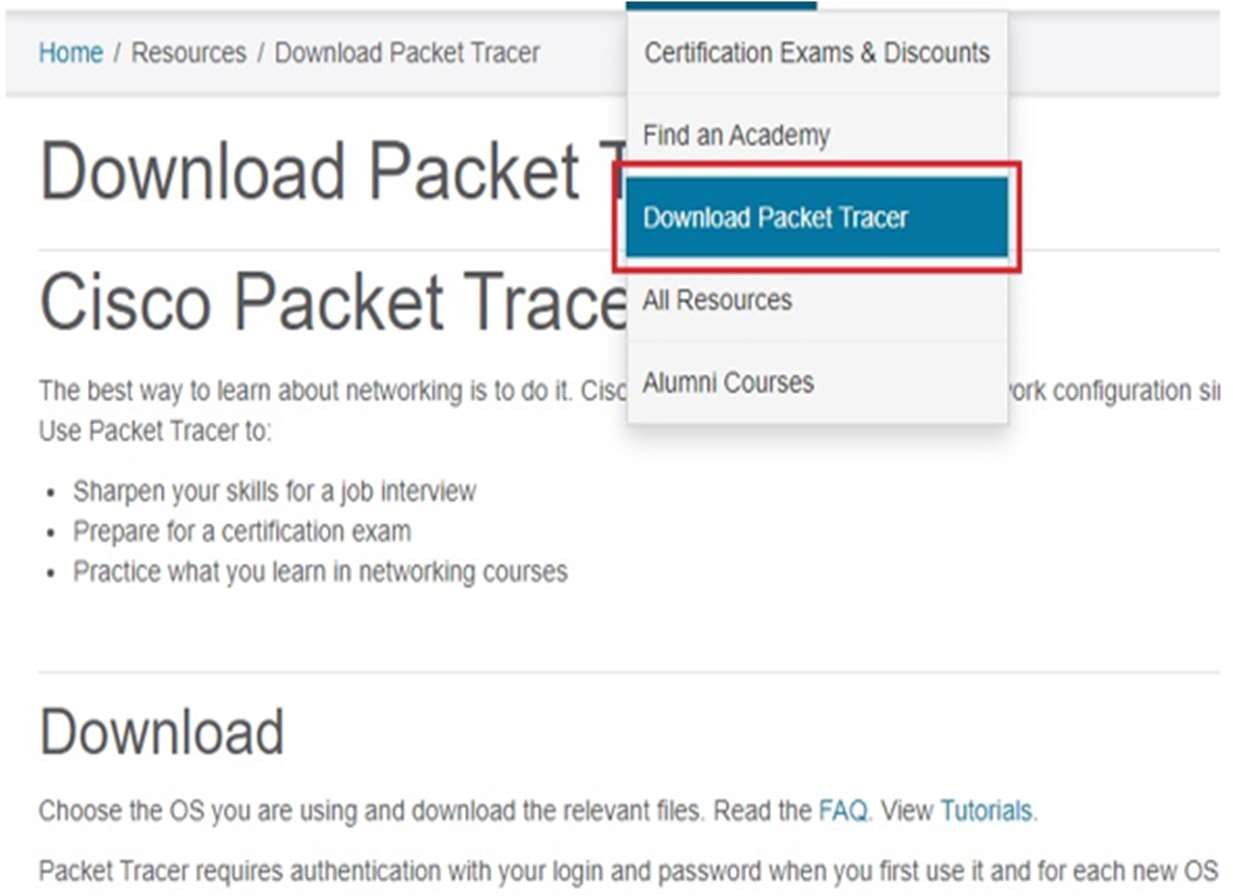
 

Fig 1:Packet tracer download

Now the file will be downloaded in the Downloads folder of the machine.

### Step 02 :

Next, open the folder where the Packet Tracer setup file has been downloaded to and double click on the setup file.

### Step 03 :

On the packet tracer software license agreement select the “I ACCEPT THE AGREEMENT” radio button and click on “NEXT”.

### Step 04 :

To choose a destination folder where Cisco Packet Tracer will be installed click on the “BROWSE” button. However it is always recommended to use the default folder. Click on “NEXT” to proceed.

### Step 05 :

* On the “SELECT START MENU” section choose a folder where Packet Tracer shortcuts will be created and click on “NEXT”
* On the “ADDITIONAL TASKS” section you can also choose to “CREATE A DESKTOP SHORTCUT” and to “CREATE A QUICK LAUNCH SHORTCUT”

Click on “NEXT” to proceed, on the “READY TO INSTALL” section click on the “INSTALL” button.

**Step - 06** : Now the Installation will be done.

**Step – 07 :** After the Installation Process is done, know double click on the Cisco Packet Tracer desktop icon and enter your NetAcad login credentials. The Packet Tracer user interface will be displayed and you can now start designing and simulating CISCO networks.

**Step – 08 :** After a successful login, the cisco packet interface appears as shown in the below image.

**Step – 09** : Once it is done, the user can access all the contents of the cisco packet tracer.

**INSTALLATION STEPS OF NS2 IN UBUNTU 14.04:**

**Step-1:** Download the latest version of ns-2.35 from this link to the desktop [www.isi.edu/nsnam/ns/](http://www.isi.edu/nsnam/ns/)

**Step-2:** Extract the zipped file by right clicking context menu “Extract here”

**Step-3:** Now we have to update the Ubuntu with its latest components. Open up a terminal and run the following commands:

1. sudo apt-get update

2. sudo apt-get upgrade

**Step4:** Now go to Desktop from terminal by typing the following commands

cd Desktop\ns-allinone2.35\

**Step-5:** Before installing the NS we have to install some essential packages required by the NS. So run the following commands:

1. sudo apt-get install tcl8.5-dev

2. sudo apt-get install tk8.5-dev

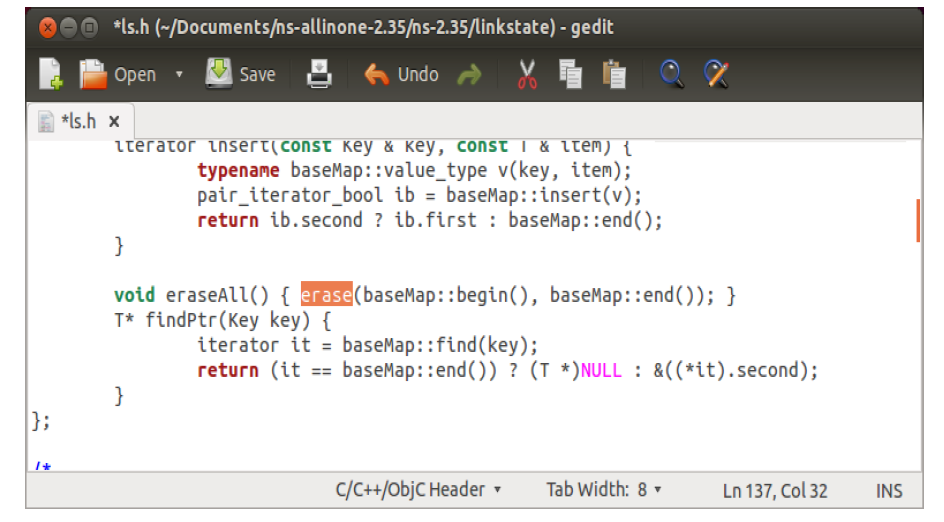
3. sudo apt-get install build-essential autoconf automake

4. sudo apt-get install perl xgraph libxt-dev libx11-dev libxmu-dev

**Step-6:** After extracting the "ns-allinone-2.35" folder, open up the file "/ns-allinone-2.35/ns-2.35/linkstate/ls.h" in an editor. You can do it either from the terminal or from the file explorer (Nautilus). We have to make some changes in the ls.h file else it will show an error while installing

the NS. Once you have opened the file move to the line 137 and replace the erase (Image 1) with this—>erase (image 2) and save the file.

**Step-7:** ./install



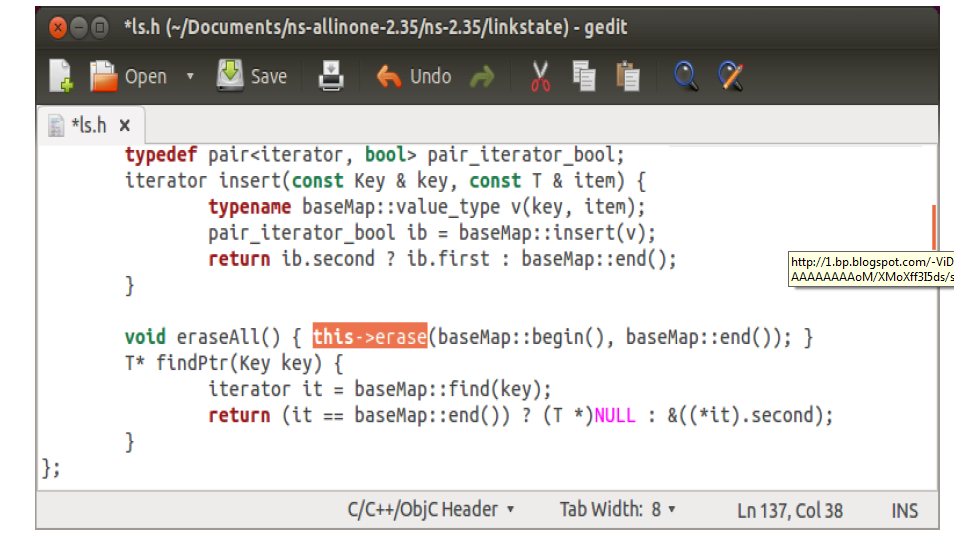


Fig. 2,3:NS-2 INSTALLATION PROCESS

### Steps to uninstall ns2

1     Delete the directory 'ns-allinone-2.35'.

2     Delete the ns related files from '/user/local/bin'.

3     Edit the '.bashrc file' and remove the paths that you have added during the  installation.

4     Restart the system and run following commands one by one:

5     sudo apt-get autoremove

6     sudo apt-get clean

# STEPS TO INSTALL NETWORK SIMULATOR NS2 ON UBUNTU 16.04

### Install Prerequisites

Type following commands on terminal  
  
sudo apt-get update  
sudo apt-get dist-upgrade  
sudo apt-get update  
sudo apt-get gcc  
sudo apt-get install build-essential autoconf automake  
sudo apt-get install tcl8.5-dev tk8.5-dev  
sudo apt-get install perl xgraph libxt-dev libx11-dev libxmu-dev

### Extract ns2

After extracting the "ns-allinone-2.35" folder, open up the file "/ns-allinone-2.35/ns-2.35/linkstate/ls.h" in an editor. You can do it either from the terminal or from the file explorer (Nautilus). We have to make some changes in the ls.h file else it will show an error while installing the NS. Once you have opened the file move to the line 137 and replace the erase (Image 1) with this—>erase (image 2) and save the file.

**Open bashrc file to Set the Environment Variables**

#### Type following commands on terminal sudo gedit ~/.bashrc

#### Copy the following lines AT THE END of the file.# LD\_LIBRARY\_PATHOTCL\_LIB=/opt/ns-allinone-2.35/otcl-1.14/NS2\_LIB=/opt/ns-allinone-2.35/lib/USR\_Local\_LIB=/usr/local/lib/export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:$OTCL\_LIB:$NS2\_LIB:$USR\_Local\_LIB

#### # TCL\_LIBRARYTCL\_LIB=/opt/ns-allinone-2.35/tcl8.5.10/library/USR\_LIB=/usr/lib/export TCL\_LIBRARY=$TCL\_LIBRARY:$TCL\_LIB:$USR\_LIB

#### # PATHXGRAPH=/opt/ns-allinone-2.35/xgraph-12.2/:/opt/ns-allinone-2.35/bin/:/opt/ns-allinone-2.35/tcl8.5.10/unix/:/opt/ns-allinone-2.35/tk8.5.10/unix/NS=/opt/ns-allinone-2.35/ns-2.35/NAM=/opt/ns-allinone-2.35/nam-1.15/export PATH=$PATH:$XGRAPH:$NS:$NAM

#### Type following commands on terminal ns  If you received the "%" sign, it means that NS is running correctly !

### Steps to uninstall ns2

1     Delete the directory 'ns-allinone-2.35'.

2     Delete the ns related files from '/user/local/bin'.

3     Edit the '.bashrc file' and remove the paths that you have added during the  installation.

4     Restart the system and run following commands one by one:

5     sudo apt-get autoremove

6     sudo apt-get clean

# WEEK-2

**Aim:** Configure the LAN using packet tracer.

# Procedure:

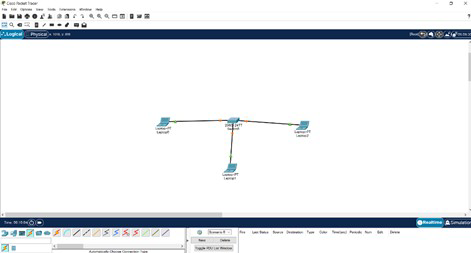
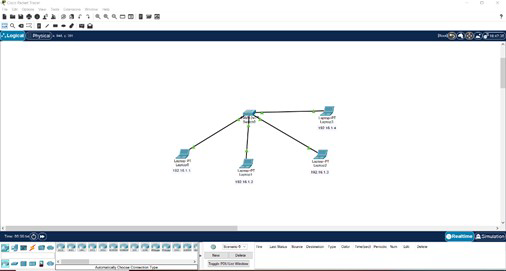
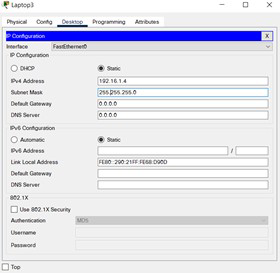
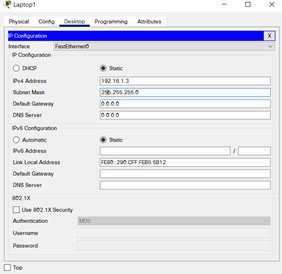
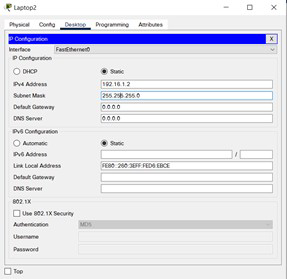
* Open packet tracer Select the end devices , pick up the switch (2690) placed in the work space.
  + Connect the switch and end Devices through Copper Straight-Through cable using fast ethernet
* Assigning IP address of each system by opening the end device go to Desktop configure the IP Address While assigning the IP address make sure all the system are in same network

Fig.4 LAN IN NS-2

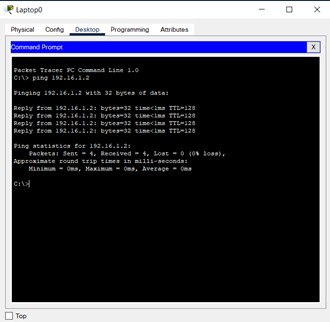
**EX :** System 1 IP : 192.16.1.1

System 2 IP : 192.16.1.2 … ect



Now Using the ping command we can check, the end devices are not. Open any one of the end device go to Desktop -> Command prompt type ping IP address of another end device

EX: Open device 1 type ping 192.16.1.2



$ - Hence the Network established successfully if the above message is displayed.

# TOPOLOGIES USING PACKET TRACER

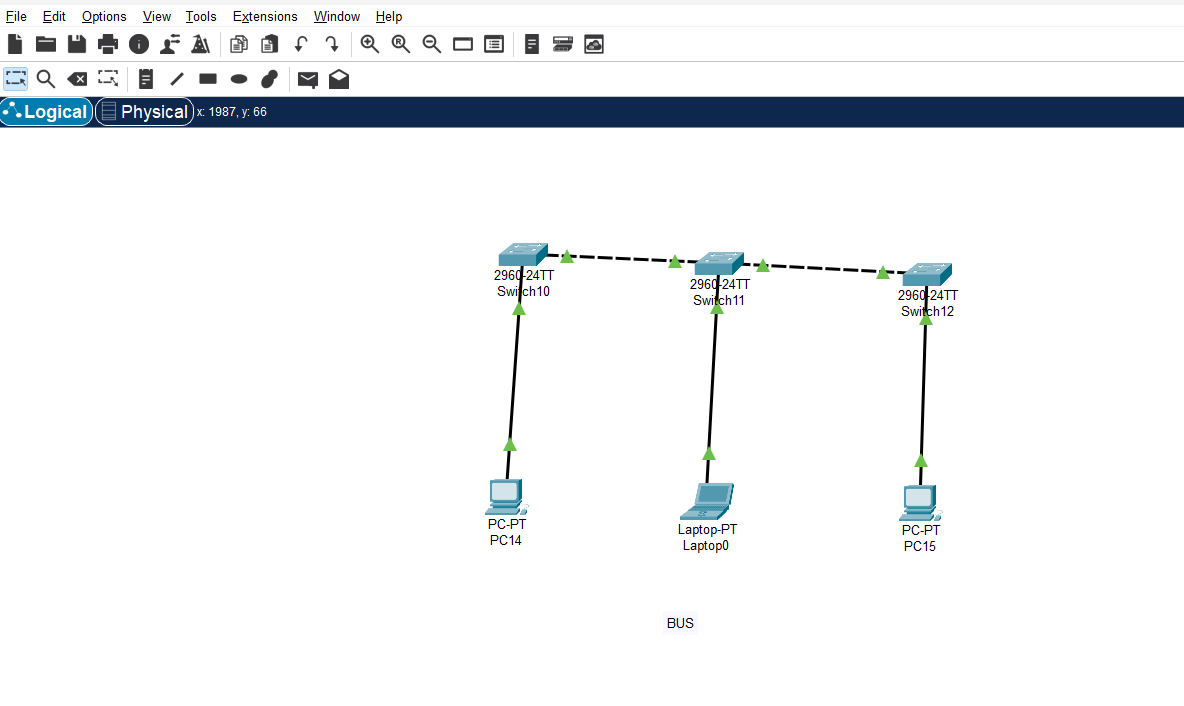
Network topology refers to the manner in which the links and nodes of a network are arranged to relate to each other.

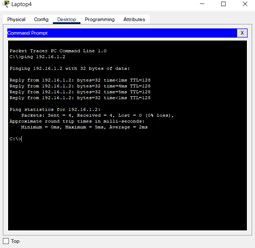
Physical network topology examples

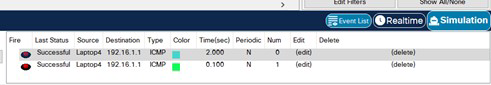
1. Bus
2. Star
3. Mesh
4. Ring

# Bus Topology

Bus topology is a network type in which every computer and network device is connected to a single cable. It transmits the data from one end to another in a single direction. No bi-directional feature is in bus topology.



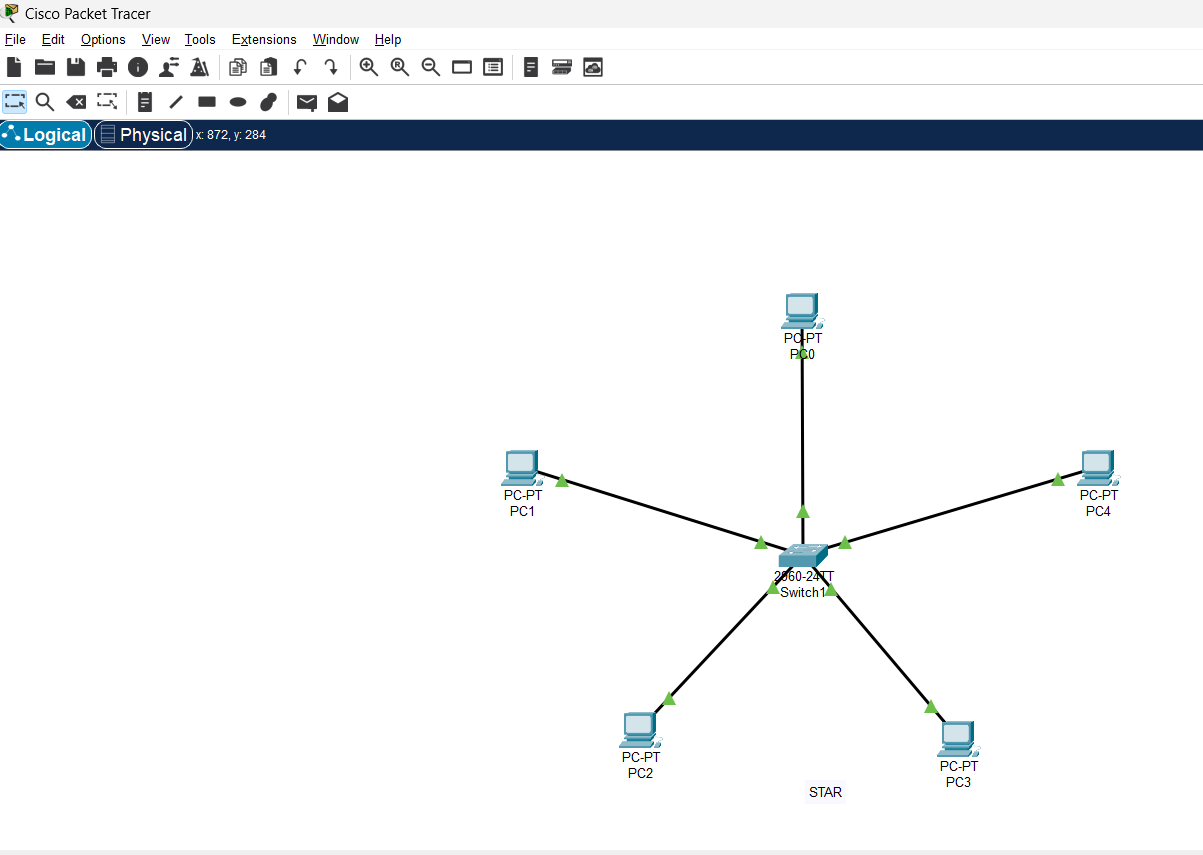
* Configure each and every end device with IP address and subnet mask. Then open any one of the system open command prompt type ping IP address of another system.
* Message sending successful prompt will be displayed at the bottom of the Window.



# Star Topology

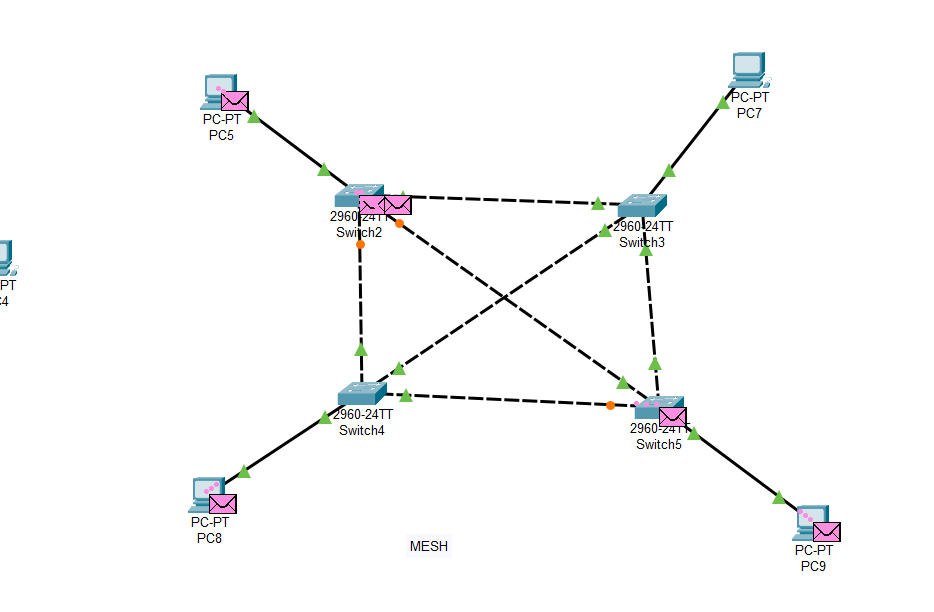
Star topology is a network topology in which each network component is physically connected to a central node such as a router, hub or switch. ...

When the central node receives a packet from a connecting node, it can pass the packet on to other nodes in the network. A star topology is also known as a star network .

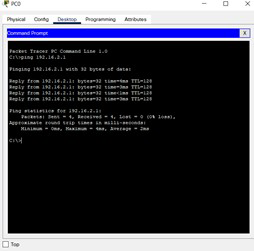


# Mesh Topology

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 network Below is a visual example of a simple computer setup on a network using a **mesh topology**.

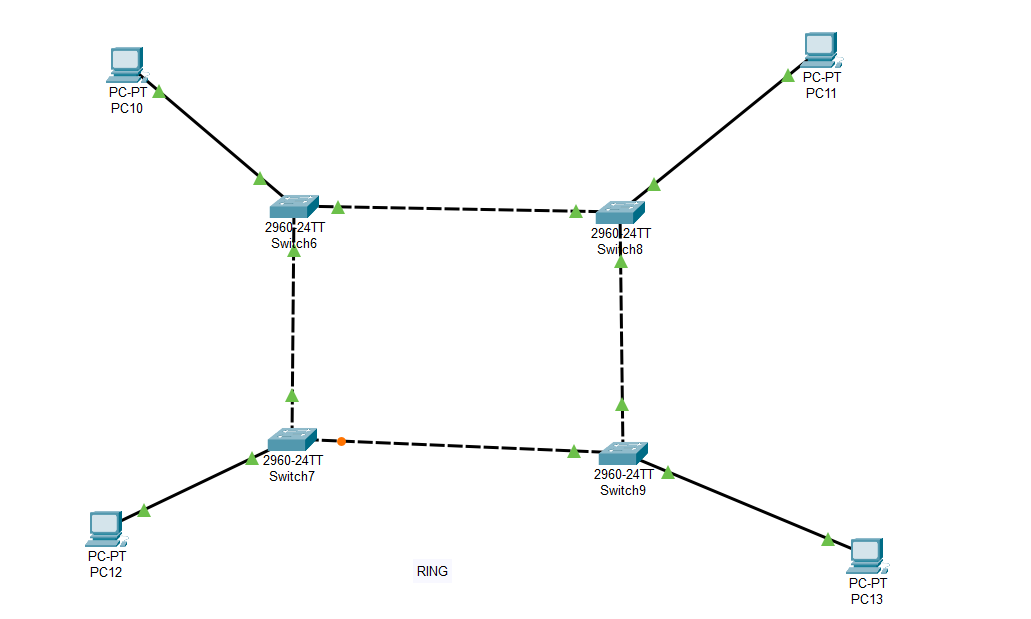


* Configure each and every end device with IP address and subnet mask. Then open any one of the system open command prompt type ping IP address of another system.



# Ring Topology

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 referred to as a ring network.



**WEEK-3**

**Aim:** Design a sample topology using NS2 Simulation Tool.

# Bus Topology:

#Create a simulator object set ns

set ns[new Simulator]

#Open file

set nf [open bus.nam w]

$ns namtrace-all $nf

$ns color 1 Red

$ns color 2 Blue

$ns color 3 Green

$ns color 4 Yellow

$ns color 5 Violet

$ns color 6 Pink

$ns color 7 Brown

$ns color 8 White

# #Define a 'finish' procedure

proc finish {} {

global ns nf

$ns flush-trace

#close the trace file

close $nf

# #Execute nam on the trace file

exec nam bus.nam &

exit 0

}

# #Create Nodes for the simulation

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

# #Create a duplex link between the links

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

$ns duplex-link $n1 $n2 10Mbps 10ms DropTail

$ns duplex-link $n2 $n3 10Mbps 10ms DropTail

$ns duplex-link $n3 $n4 10Mbps 10ms DropTail

# #Labelling the nodes

$ns at 0.0 "$n0 label Client1"

$ns at 0.0 "$n1 label Client2"

$ns at 0.0 "$n2 label Client3"

$ns at 0.0 "$n3 label Client4"

$ns at 0.0 "$n4 label Client4"

# #Orient

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

$ns duplex-link-op $n1 $n2 orient right$ns duplex-link-op $n2 $n3 orient right

$ns duplex-link-op $n3 $n4 orient null

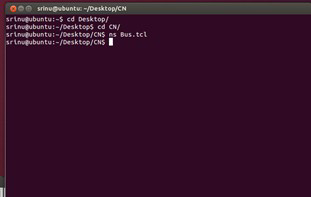
# #Specifying the start and stop time

$ns at 1.0 "finish"

# #Start simulation

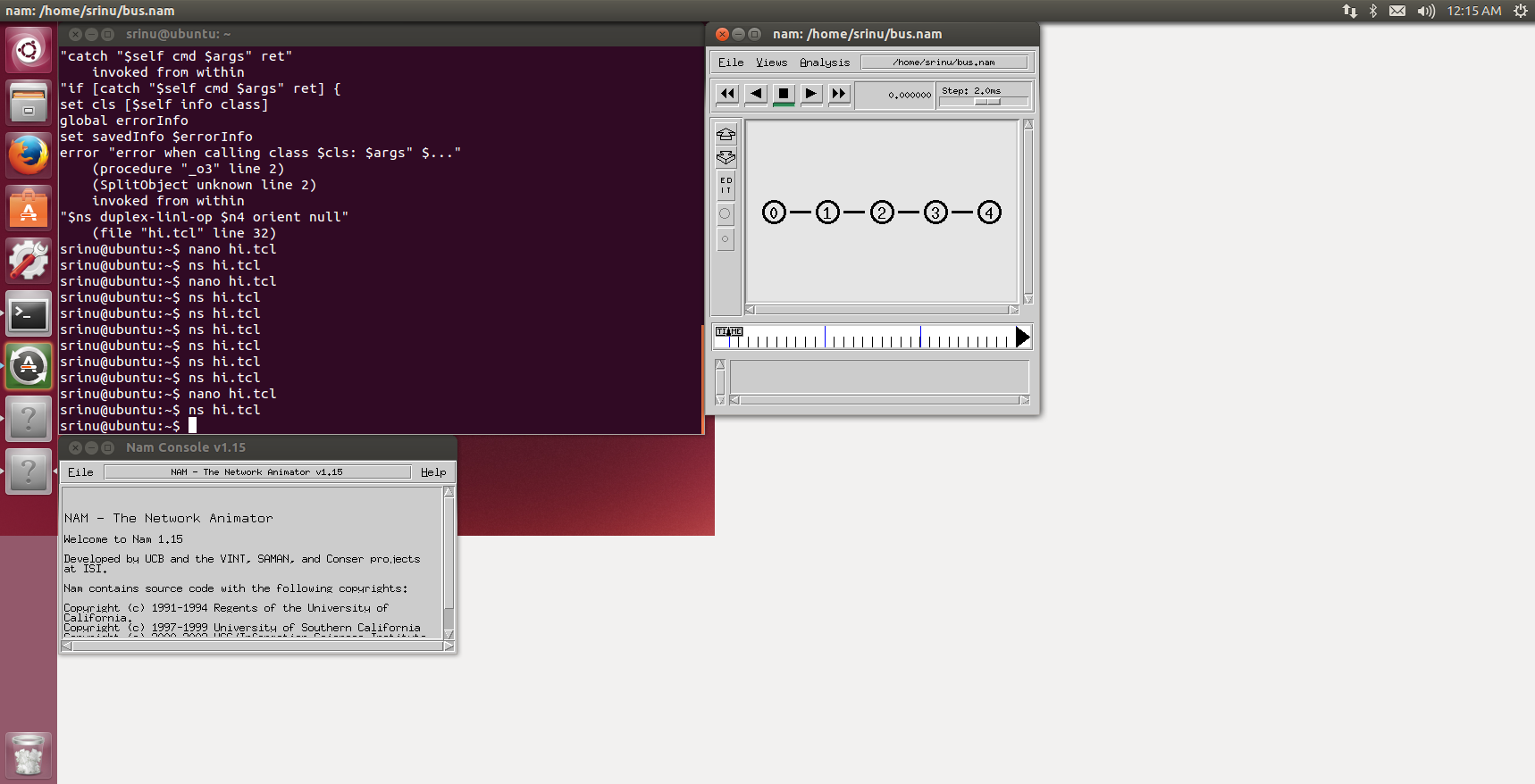
$ns run

* Open Terminal go to the file location type ns file.tcl



# Output:

Bus Topology in nam



# Ring Topology :

#Create a simulator object set ns

set ns [new Simulator]

#Open file for nam tracing

set nf [open ring.nam w]

$ns namtrace-all $nf

# #Set up different colors for dataflow

$ns color 1 Red

$ns color 2 Blue

$ns color 3 Green

$ns color 4 Yellow

$ns color 5 Violet

$ns color 6 Pink

$ns color 7 Brown

$ns color 8 White

# #Define a 'finish' procedure

proc finish {} {

global ns nf

$ns flush-trace

#close the trace file

close $nf

# #Execute nam on the trace file

exec nam ring.nam &

exit 0

}

# #Create Nodes for the simulation

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

set n5 [$ns node]

set n6 [$ns node]

set n7 [$ns node]

set n8 [$ns node]

# #Create a duplex link between the links

$ns duplex-link $n1 $n2 1Mb 10ms DropTail

$ns duplex-link $n2 $n3 1Mb 10ms DropTail

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

$ns duplex-link $n4 $n5 1Mb 10ms DropTail

$ns duplex-link $n5 $n6 1Mb 10ms DropTail

$ns duplex-link $n6 $n7 1Mb 10ms DropTail

$ns duplex-link $n7 $n8 1Mb 10ms DropTail

$ns duplex-link $n8 $n1 1Mb 10ms DropTail

# #Labelling the nodes

$ns at 0.0 "$n1 label Client1"

$ns at 0.0 "$n2 label Client2"

$ns at 0.0 "$n3 label Client3"

$ns at 0.0 "$n4 label Client4"

$ns at 0.0 "$n5 label Client5"

$ns at 0.0 "$n6 label Client6"

$ns at 0.0 "$n7 label Client7"

$ns at 0.0 "$n8 label Client8"

# #Orient

$ns duplex-link-op $n1 $n2 orient right-down

$ns duplex-link-op $n2 $n3 orient down-right

$ns duplex-link-op $n3 $n4 orient left-down

$ns duplex-link-op $n4 $n5 orient left-down

$ns duplex-link-op $n5 $n6 orient left-up

$ns duplex-link-op $n6 $n7 orient left-up

$ns duplex-link-op $n7 $n8 orient right-up

$ns duplex-link-op $n8 $n1 orient right-up

# #Coloring the link

$ns duplex-link-op $n1 $n2 color red

$ns duplex-link-op $n2 $n3 color blue

$ns duplex-link-op $n3 $n4 color green

$ns duplex-link-op $n4 $n5 color yellow

$ns duplex-link-op $n5 $n6 color violet

$ns duplex-link-op $n6 $n7 color pink

$ns duplex-link-op $n7 $n8 color brown

$ns duplex-link-op $n8 $n1 color white

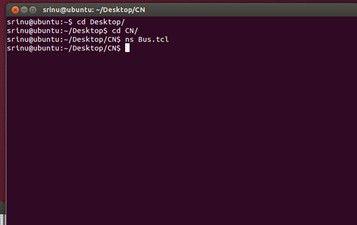
# #Specifying the start and stop time

$ns at 1.0 "finish"

# #Start simulation

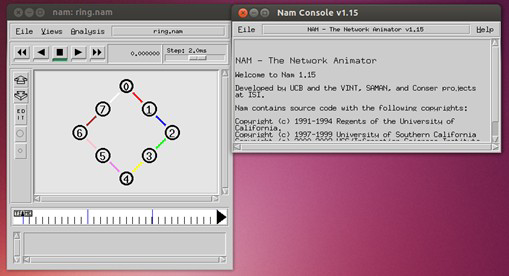
$ns run

* Open Terminal go to the file location type ns file.tcl ex : ns Ring.tcl



# Output:

Ring Topology in nam



# Mesh Topology :

#Create a simulator object set ns

set ns[new Simulator]

#Open file for nam tracing

set nf [open mesh2.nam w]

$ns namtrace-all $nf

# #Set up different colors for dataflow

$ns color 1 Red

$ns color 2 Blue

$ns color 3 Green

$ns color 4 Yellow

$ns color 5 Violet

$ns color 6 Pink

$ns color 7 Brown

$ns color 8 White

# #Define a 'finish' procedure

proc finish {} {

global ns nf

$ns flush-trace #close the trace file close $nf

# #Execute nam on the trace file

exec nam mesh2.nam &

exit 0 }

# #Create Nodes for the simulation

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

# #Create a duplex link between the links

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

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

$ns duplex-link $n0 $n3 10Mbps 10ms DropTail

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

$ns duplex-link $n1 $n2 10Mbps 10ms DropTail

$ns duplex-link $n1 $n3 10Mbps 10ms DropTail

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

$ns duplex-link $n2 $n1 10Mbps 10ms DropTail

$ns duplex-link $n2 $n3 10Mbps 10ms DropTail

$ns duplex-link $n3 $n0 10Mbps 10ms DropTail

$ns duplex-link $n3 $n1 10Mbps 10ms DropTail

$ns duplex-link $n3 $n2 10Mbps 10ms DropTail

# #Labelling the nodes

$ns at 0.0 "$n0 label Client1"

$ns at 0.0 "$n1 label Client2"

$ns at 0.0 "$n2 label Client3"

$ns at 0.0 "$n3 label Client4"

# #Orient

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

$ns duplex-link-op $n0 $n2 orient right

$ns duplex-link-op $n0 $n3 orient right-down

$ns duplex-link-op $n1 $n0 orient left-down

$ns duplex-link-op $n1 $n2 orient right-down

$ns duplex-link-op $n1 $n3 orient down

$ns duplex-link-op $n2 $n0 orient left

$ns duplex-link-op $n2 $n1 orient left-up

$ns duplex-link-op $n2 $n3 orient left-down

$ns duplex-link-op $n3 $n0 orient left-up

$ns duplex-link-op $n3 $n1 orient up

$ns duplex-link-op $n3 $n2 orient right-up

# #Coloring the link

$ns duplex-link-op $n0 $n1 color red

$ns duplex-link-op $n0 $n2 color yellow

$ns duplex-link-op $n0 $n3 color pink

$ns duplex-link-op $n1 $n0 color red

$ns duplex-link-op $n1 $n2 color blue

$ns duplex-link-op $n1 $n3 color black

$ns duplex-link-op $n2 $n0 color yellow

$ns duplex-link-op $n2 $n1 color blue

$ns duplex-link-op $n2 $n3 color purple

$ns duplex-link-op $n3 $n0 color pink

$ns duplex-link-op $n3 $n1 color black

$ns duplex-link-op $n3 $n2 color purple

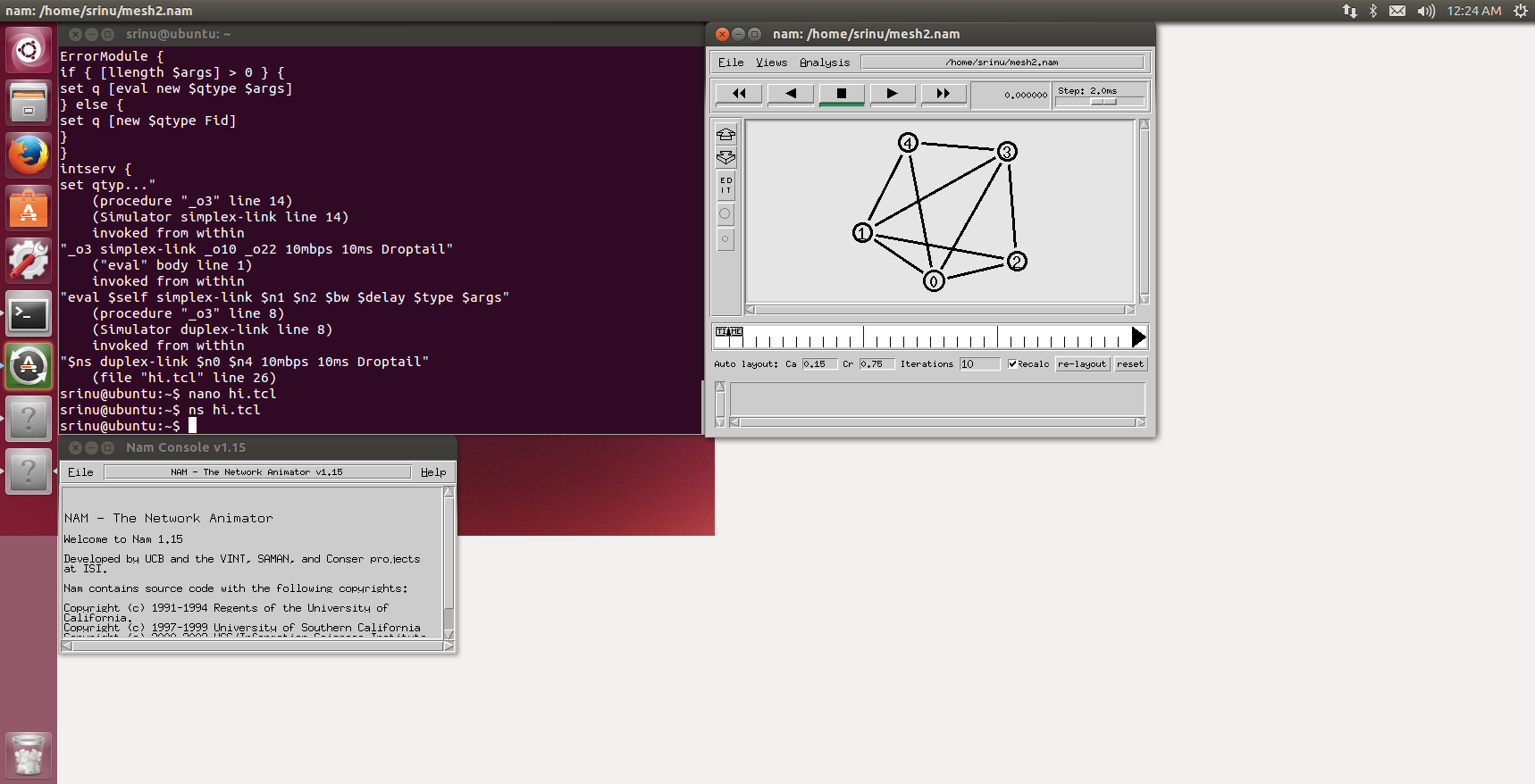
# #Specifying the start and stop time

$ns at 1.0 "finish"

# #Start simulation

$ns run

**Output:** nam



# Star Topology :

#Create a simulator object set ns

set ns[new Simulator]

#Open file for nam tracing

set nf [open star.nam w]

$ns namtrace-all $nf

# #Set up different colors for dataflow

$ns color 1 Red

$ns color 2 Blue

$ns color 3 Green

$ns color 4 Yellow

# 

# #Define a 'finish' procedure

proc finish {} {

global ns nf

$ns flush-trace

#close the trace file

close $nf

# #Execute nam on the trace file

exec nam star.nam &

exit 0

}

# #Create Nodes for the simulation

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

# #Create a duplex link between the links

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

$ns duplex-link $n0 $n2 1Mb 10ms DropTail

$ns duplex-link $n0 $n3 1Mb 10ms DropTail

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

$ns at 0.0 "$n0 label Router1"

$ns at 0.0 "$n1 label Client1"

$ns at 0.0 "$n2 label Client2"

$ns at 0.0 "$n3 label Client3"

$ns at 0.0 "$n4 label Client4"

# #Orient

$ns duplex-link-op $n0 $n1 orient left-up

$ns duplex-link-op $n0 $n2 orient left down

$ns duplex-link-op $n0 $n3 orient right-up

$ns duplex-link-op $n0 $n4 orient down-right

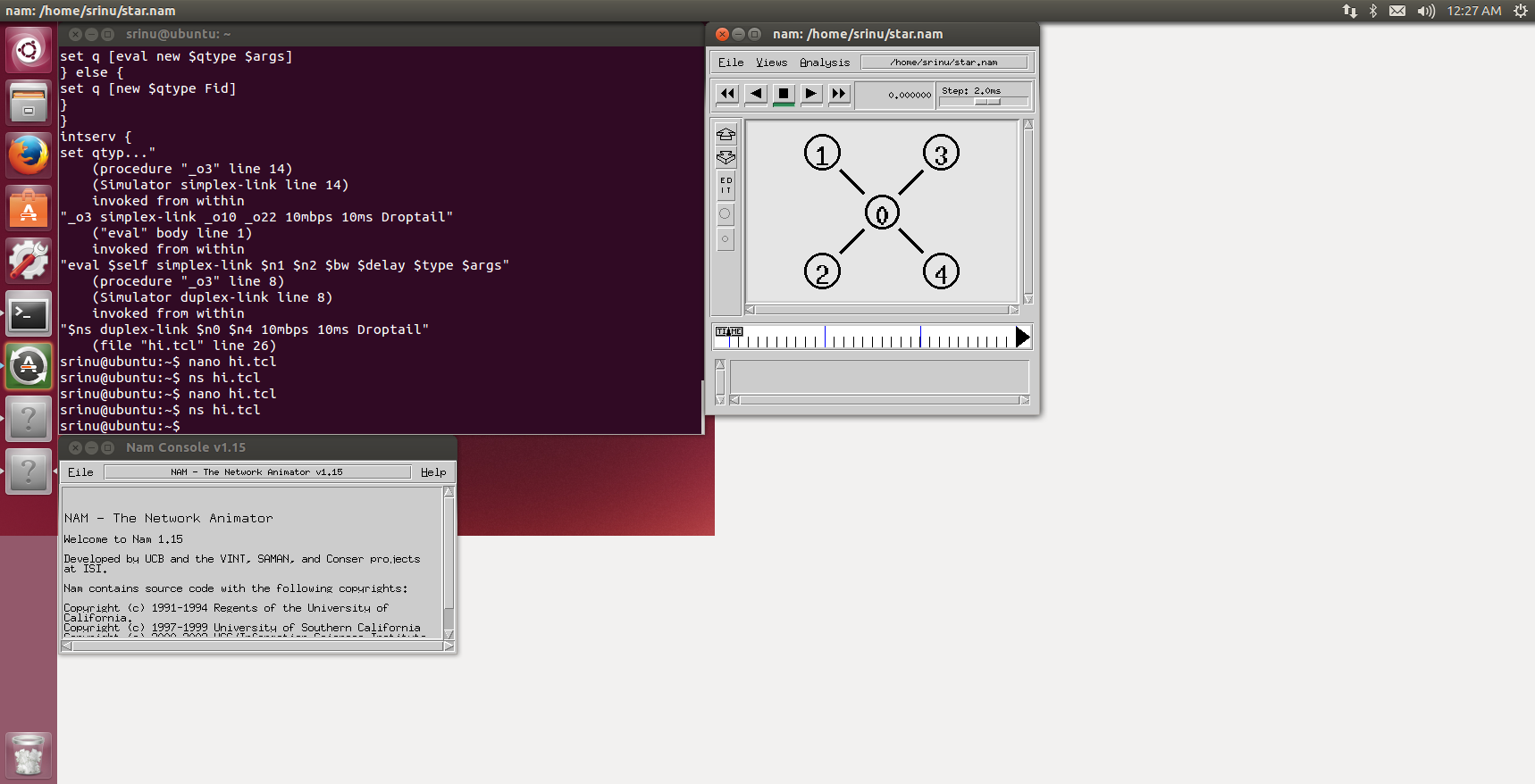
# #Specifying the start and stop time

$ns at 1.0 "finish"

# #Start simulation

$ns run

# Output:

****

**WEEK-4**

**Aim:** Implement CRC and Hamming code for error handling.

# Program:

**Hamming Code:**

def calcRedundantBits(m):

    for i in range(m):

        if(2\*\*i >= m + i + 1):

            return i

def posRedundantBits(data, r):

    j = 0

    k = 1

    m = len(data)

    res = ''

    for i in range(1, m + r + 1):

        if(i == 2\*\*j):

            res = res + '0'

            j += 1

        else:

            res = res + data[-1 \* k]

            k += 1

    return res[::-1]

def calcParityBits(arr, r):

    n = len(arr)

    for i in range(r):

        val = 0

        for j in range(1, n + 1):

            if(j & (2\*\*i) == (2\*\*i)):

                val = val ^ int(arr[-1 \* j])

        arr = arr[:n-(2\*\*i)] + str(val) + arr[n-(2\*\*i)+1:]

    return arr

def detectError(arr, nr):

    n = len(arr)

    res = 0

    for i in range(nr):

        val = 0

        for j in range(1, n + 1):

            if(j & (2\*\*i) == (2\*\*i)):

                val = val ^ int(arr[-1 \* j])

        res = res + val\*(10\*\*i)

    return int(str(res), 2)

data = '1011001'

m = len(data)

r = calcRedundantBits(m)

arr = posRedundantBits(data, r)

arr = calcParityBits(arr, r)

print("Data transferred is " + arr)

print("case1")

arr = '10101001110'

print("Error Data is " + arr)

correction = detectError(arr, r)

if correction == 0:

    print("There is no error in the received message.")

else:

    print("The position of error is", len(arr) - correction + 1, "from the left")

print("case2")

arr = '11101001110'

print("Error Data is " + arr)

correction = detectError(arr, r)

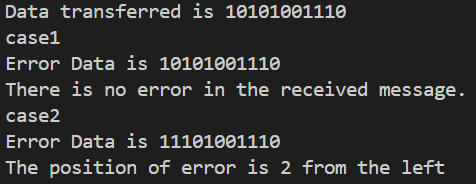
if correction == 0:

    print("There is no error in the received message.")

else:

    print("The position of error is", len(arr) - correction + 1, "from the left")

**output:**

****

**CRC:**

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(dividend, divisor):

    pick = len(divisor)

    tmp = dividend[0:pick]

    while pick < len(dividend):

        if tmp[0] == '1':

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

        else:

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

        pick += 1

    if tmp[0] == '1':

        tmp = xor(divisor, tmp)

    else:

        tmp = xor('0'\*pick, tmp)

    checkword = tmp

    return checkword

def encodeData(data, key):

    l\_key = len(key)

    appended\_data = data + '0'\*(l\_key-1)

    remainder = mod2div(appended\_data, key)

    codeword = data + remainder

    return codeword

data = "1101011011"

key = "1001"

print("Original Data: ", data)

encoded\_data = encodeData(data, key)

print("Encoded Data: ", encoded\_data)

**Output:**

****

**WEEK-5**

**Aim:** Implementation of stop and wait protocol.

# Program:

import time

import random

WINDOW\_SIZE=1

TIMEOUT=2

def send\_packet(packet):

    print(f"Sending packet {packet}")

    if random.random()<0.1:

        print(f"Packet {packet} lost")

        return False

    return True

def receive\_packet(packet):

    print(f"Receiver received packet {packet}")

    if random.random()<0.9:

        return True

    return False

def stopandwait(packets):

    for packet in packets:

        while True:

            if send\_packet(packet):

                start\_time=time.time()

                ack\_received=False

                while time.time()-start\_time<TIMEOUT:

                    if receive\_packet(packet):

                        ack\_received=True

                        print(f"Acknowledgment for packet {packet} received")

                        break

                if ack\_received:

                    break

                else:

                    print(f"No acknowledgment for packet {packet}. Retrying...")

            else:

                print(f"Failed to send packet {packet}. Retrying...")

        time.sleep(1)

packets=[1,2,3,4,5]

stopandwait(packets)

# Output:

# 

**WEEK-6**

**AIM:** Implementation of stop and wait protocol.

Client.py

import socket as s def client():

host = s.gethostname() port = 1234 client\_socket = s.socket()

client\_socket.connect((host, port))

n = int(input("Enter Number of Packets : ")) input\_data = "Enter Packet of Data : "

c = 0

while c < n:

datapacket = input(input\_data) datapacket = str(datapacket) client\_socket.send(datapacket.encode())

acknowledgement = client\_socket.recv(1024).decode() if acknowledgement == "Not Received":

input\_data = "Data not received!! Resend the previous data : " else:

input\_data = "Enter Packet of Data : " print(acknowledgement)

c += 1

client\_socket.close() client()

Server.py

import socket as s import time as t def server():

host = s.gethostname() port = 1234

server\_socket = s.socket()

server\_socket.bind((host, port)) server\_socket.listen(2)

connec, address = server\_socket.accept() print("Connection from : " + str(address)) c = 1

while True:

data = connec.recv(1024).decode() if not data:

break

if c % 4 == 0: t.sleep(2)

data = "Not Received" connec.send(data.encode()) c += 1

continue

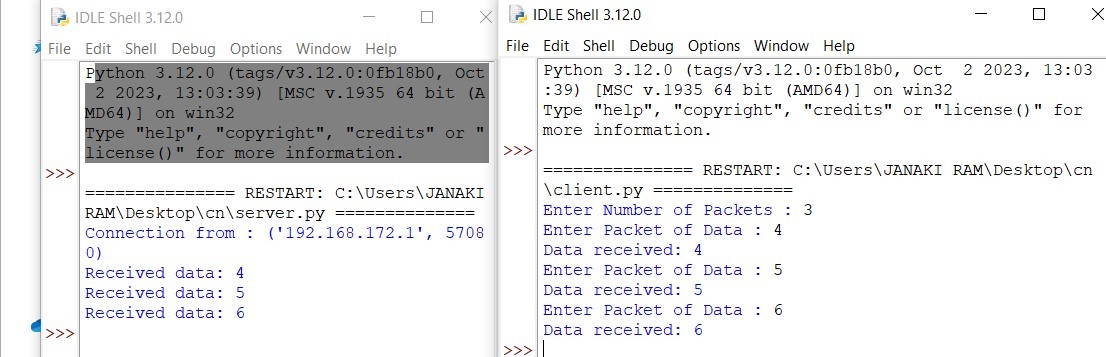
print("Received data: " + str(data)) response = "Data received: " + str(data) connec.send(response.encode())

c += 1

connec.close()

server()

OUTPUT:



**WEEK – 7**

**AIM:** Implementation of goback-N protocol and selective repeat protocol using NS2 Simulation Tool

Goback-N protocol :

set ns [new Simulator]

set nf [open "Go-Back-N.nam" w]

$ns namtrace-all $nf

set f [open "Go-Back-N.tr" w]

$ns trace-all $f

proc finish {} { global ns

$ns flush-trace

puts "running nam..."

exec nam "Go-Back-N.nam" & exit 0

}

set no [$ns node] set n1 [$ns node] set n2 [$ns node] set n3 [$ns node]

$ns at 0.0 "$no label Sender"

$ns at 0.0 "$n1 label Receiver"

$no color brown

$n1 color red

$ns duplex-link $no $n2 1Mb 20ms DropTail

$ns duplex-link $n2 $n3 1Mb 20ms DropTail

$ns duplex-link $n3 $n1 1Mb 20ms DropTail

Agent/TCP set nam\_tracevar\_ true set tcp [new Agent/TCP]

$tcp set windowInit\_ 2

$tcp set maxcwnd\_ 2

$ns attach-agent $no $tcp

set sink [new Agent/TCPSink]

$ns attach-agent $n1 $sink

$ns connect $tcp $sink

$ns rtmodel-at 1.80 down $n2 $n3

$ns rtmodel-at 1.91 up $n2 $n3

set ftp [new Application/FTP]

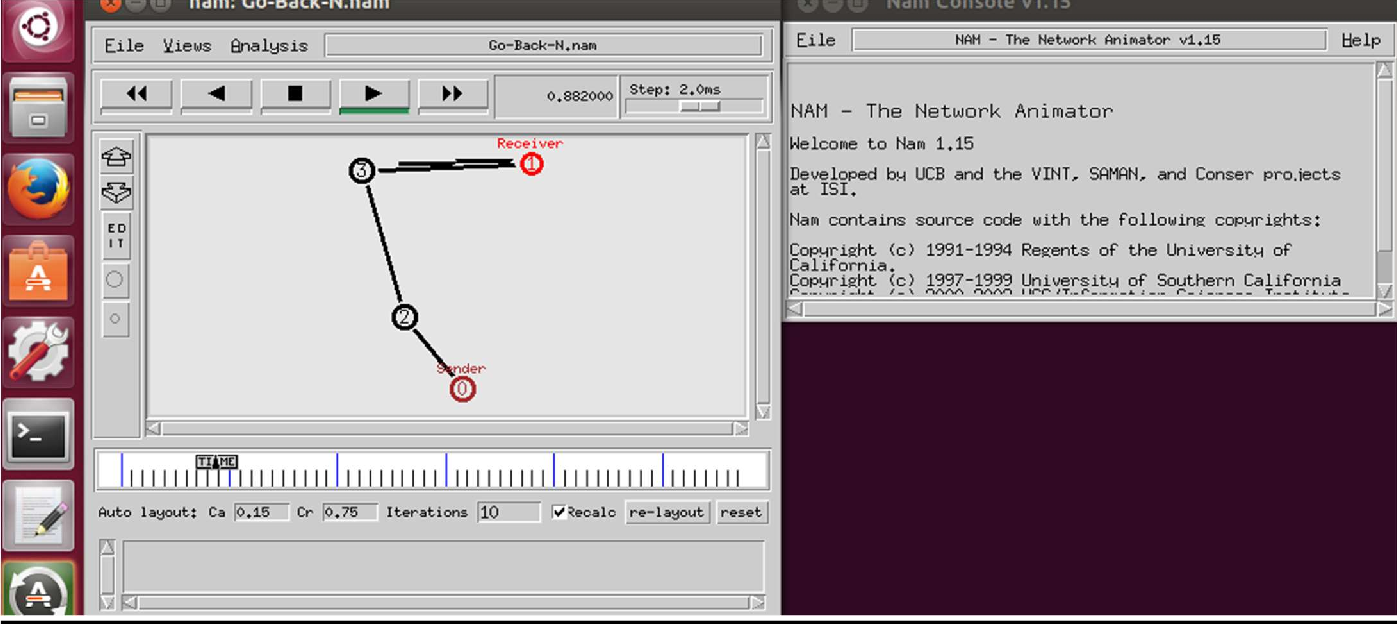
$ftp attach-agent $tcp

$ns at 0.1 "$ftp start"

$ns at 5.68 "$ftp stop"

$ns at 6.8 "$ns detach-agent $no $tcp; $ns detach-agent $n1 $sink"

$ns at 7.0 "finish"

$ns run OUTPUT:

**Selective repeat protocol :**

set ns [new Simulator] set n0 [$ns node]

set n1 [$ns node] set n2 [$ns node] set n3 [$ns node] set n4 [$ns node] set n5 [$ns node]

$n0 color "red"

$n1 color "red"

$n2 color "green"

$n3 color "green"

$n4 color "black"

$n5 color "black"

$n0 shape circle ;

$n1 shape circle ;

$n2 shape circle ;

$n3 shape circle ;

$n4 shape circle ;

$n5 shape circle ;

$ns at 0.0 "$n0 label SYS1"

$ns at 0.0 "$n1 label SYS2"

$ns at 0.0 "$n2 label SYS3"

$ns at 0.0 "$n3 label SYS4"

$ns at 0.0 "$n4 label SYS5"

$ns at 0.0 "$n5 label SYS6" set nf [open Srepeat.nam w]

$ns namtrace-all $nf

set f [open Srepeat.tr w]

$ns trace-all $f

$ns duplex-link $n0 $n2 1Mb 10ms DropTail

$ns duplex-link-op $n0 $n2 orient right-down

$ns queue-limit $n0 $n2 5

$ns duplex-link $n1 $n2 1Mb 10ms DropTail

$ns duplex-link-op $n1 $n2 orient right-up

$ns duplex-link $n2 $n3 1Mb 10ms DropTail

$ns duplex-link-op $n2 $n3 orient right

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

$ns duplex-link-op $n3 $n4 orient right-up

$ns duplex-link $n3 $n5 1Mb 10ms DropTail

$ns duplex-link-op $n3 $n5 orient right-down Agent/TCP set\_nam\_tracevar\_true

set tcp [new Agent/TCP]

$tcp set fid 1

$ns attach-agent $n1 $tcp

set sink [new Agent/TCPSink]

$ns attach-agent $n4 $sink

$ns connect $tcp $sink

set ftp [new Application/FTP]

$ftp attach-agent $tcp

$ns at 0.05 "$ftp start"

$ns at 0.06 "$tcp set windowlnit 8"

$ns at 0.06 "$tcp set maxcwnd 8"

$ns at 0.25 "$ns queue-limit $n3 $n4 0"

$ns at 0.26 "$ns queue-limit $n3 $n4 10"

$ns at 0.30 "$tcp set windowlnit 1"

$ns at 0.30 "$tcp set maxcwnd 1"

$ns at 0.30 "$ns queue-limit $n3 $n4 10"

GRT INSTITUTE OF ENGINEERING AND TECHNOLOGY

$ns at 0.47 "$ns detach-agent $n1 $tcp;$ns detach-agent $n4 $sink"

$ns at 1.75 "finish"

$ns at 0.0 "$ns trace-annotate \"Select and repeat\""

$ns at 0.05 "$ns trace-annotate \"FTP starts at 0.01\""

$ns at 0.06 "$ns trace-annotate \"Send 8Packets from SYS1 to SYS4\""

$ns at 0.26 "$ns trace-annotate \"Error Occurs in 4th packet \""

$ns at 0.30 "$ns trace-annotate \"Retransmit Packet\_4 from SYS1 to SYS4\""

$ns at 1.5 "$ns trace-annotate \"FTP stops\"" proc finish {} {

global ns nf

$ns flush-trace close $nf

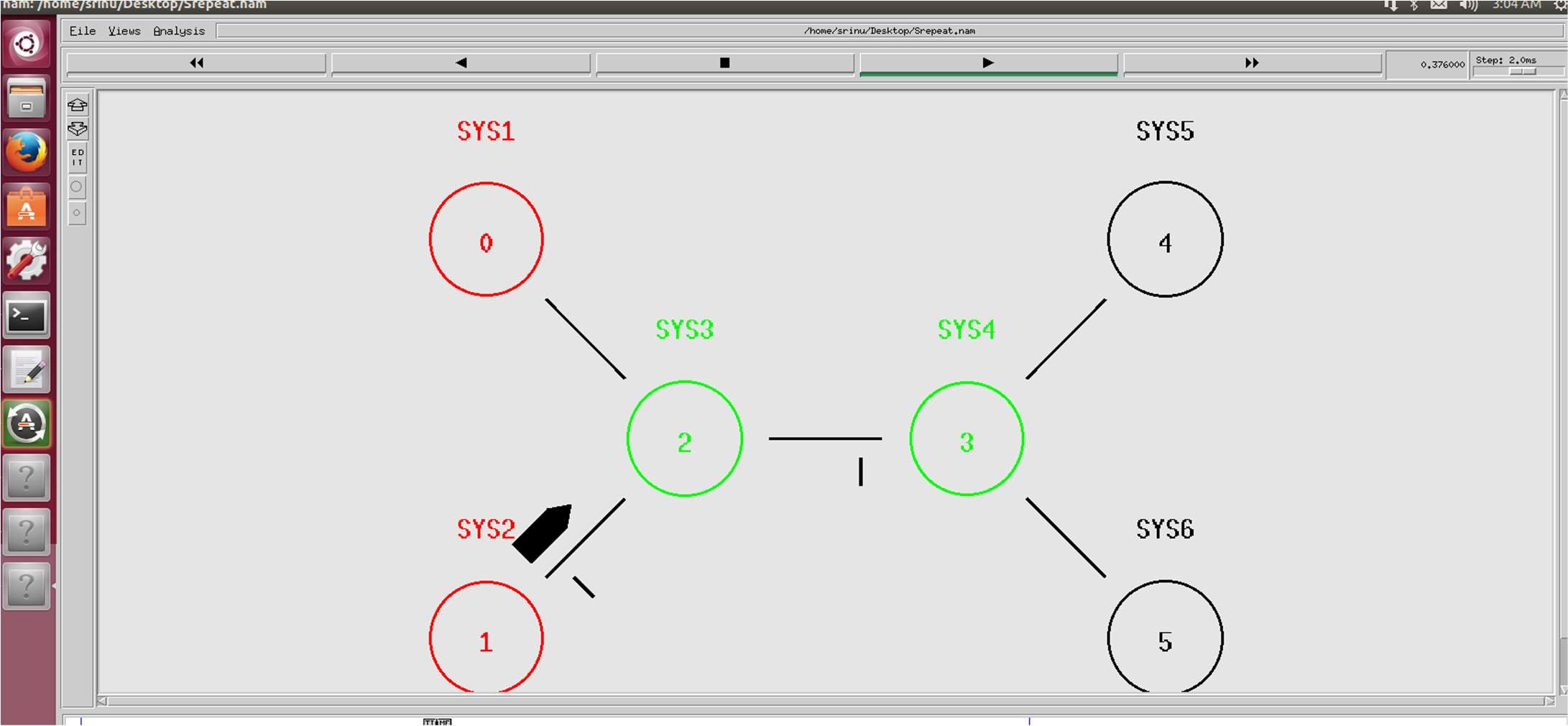
puts "filtering..."

#exec tclsh../bin/namfilter.tcl srepeat.nam #puts "running nam..."

exec nam Srepeat.nam & exit 0

}

$ns run OUTPUT:



**WEEK – 8**

**AIM:** Implementation of distance vector routing algorithm

**Program:**

class Router:

    def \_\_init\_\_(self, name):

        self.name = name

        self.distance\_vector = {name: (name, 0)}

        self.neighbors = {}

    def add\_neighbor(self, neighbor, cost):

        self.neighbors[neighbor] = cost

        self.distance\_vector[neighbor.name] = (neighbor.name, cost)

    def update\_distance\_vector(self):

        updated = False

        for neighbor, cost\_to\_neighbor in self.neighbors.items():

            for destination, (next\_hop, cost\_from\_neighbor) in neighbor.distance\_vector.items():

                new\_cost = cost\_to\_neighbor + cost\_from\_neighbor

                if destination not in self.distance\_vector or new\_cost < self.distance\_vector[destination][1]:

                    self.distance\_vector[destination] = (neighbor.name, new\_cost)

                    updated = True

        return updated

    def \_\_str\_\_(self):

        result = f"Router {self.name} Distance Vector Table:\n"

        for destination, (next\_hop, cost) in self.distance\_vector.items():

            result += f"Destination: {destination}, Next Hop: {next\_hop}, Cost: {cost}\n"

        return result

A = Router("A")

B = Router("B")

C = Router("C")

D = Router("D")

A.add\_neighbor(B, 1)

A.add\_neighbor(C, 4)

B.add\_neighbor(A, 1)

B.add\_neighbor(C, 2)

B.add\_neighbor(D, 5)

C.add\_neighbor(A, 4)

C.add\_neighbor(B, 2)

C.add\_neighbor(D, 1)

D.add\_neighbor(B, 5)

D.add\_neighbor(C, 1)

def simulate\_dvr(routers):

    converged = False

    while not converged:

        converged = True

        for router in routers:

            if router.update\_distance\_vector():

                converged = False

routers = [A, B, C, D]

simulate\_dvr(routers)

for router in routers:

    print(router)

**OUTPUT**

Router A Distance Vector Table:

Destination: A, Next Hop: A, Cost: 0

Destination: B, Next Hop: B, Cost: 1

Destination: C, Next Hop: B, Cost: 3

Destination: D, Next Hop: B, Cost: 4

Router B Distance Vector Table:

Destination: B, Next Hop: B, Cost: 0

Destination: A, Next Hop: A, Cost: 1

Destination: C, Next Hop: C, Cost: 2

Destination: D, Next Hop: C, Cost: 3

Router C Distance Vector Table:

Destination: C, Next Hop: C, Cost: 0

Destination: A, Next Hop: B, Cost: 3

Destination: B, Next Hop: B, Cost: 2

Destination: D, Next Hop: D, Cost: 1

Router D Distance Vector Table:

Destination: D, Next Hop: D, Cost: 0

Destination: B, Next Hop: C, Cost: 3

Destination: C, Next Hop: C, Cost: 1

Destination: A, Next Hop: C, Cost: 4

# WEEK- 9

**AIM:** Implementation of Link state routing algorithm using NS2 Simulation Tool # Create a new Simulator object

set ns [new Simulator]

# Open the Nam trace file for writing set nf [open thro.nam w]

$ns namtrace-all $nf

# Procedure to finish the simulation and generate the Nam trace file proc finish {} {

global ns nf

$ns flush-trace close $nf

puts "Running Nam..." exec nam thro.nam & exit 0

}

# Create nodes set n0 [$ns node] set n1 [$ns node] set n2 [$ns node] set n3 [$ns node] set n4 [$ns node]

# Create duplex links between the nodes

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

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

$ns duplex-link $n1 $n2 1Mb 10ms DropTail

$ns duplex-link $n3 $n2 1Mb 10ms DropTail

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

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

# Create UDP agent and traffic source set udp0 [new Agent/UDP]

$ns attach-agent $n0 $udp0

set null0 [new Agent/Null]

$ns attach-agent $n1 $null0

$ns connect $udp0 $null0

set cbr0 [new Application/Traffic/CBR]

$cbr0 set packetSize\_ 500

$cbr0 set interval\_ 0.005

$cbr0 attach-agent $udp0

# Define the routing protocol (Link State)

$ns rtproto LS

# Down and up routes at specific times

$ns rtmodel-at 1.5 down $n0 $n1

$ns rtmodel-at 2.0 down $n4 $n1

$ns rtmodel-at 2.5 up $n1 $n4

$ns rtmodel-at 2.9 up $n1 $n0

# Start the CBR traffic source

$ns at 1.0 "$cbr0 start"

# Schedule the finish procedure to run at 3.2 seconds

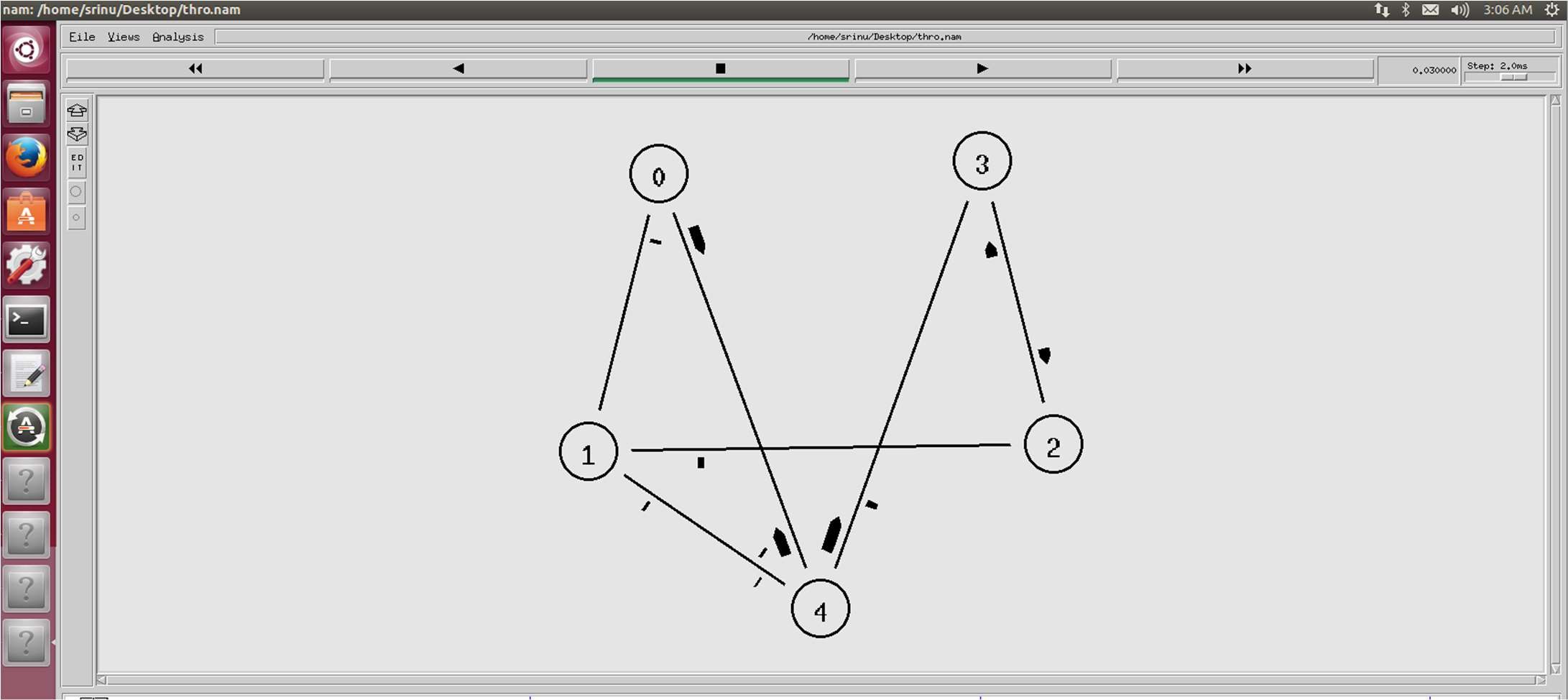
$ns at 3.2 "finish"

# Run the simulation

$ns run

# Debugging information (optional) debug

OUTPUT:



# WEEK-10

**AIM:** Implement TCP and UDP on different pairs of systems using NS2 Simulation Tool. set ns [new Simulator]

$ns color 0 blue

$ns color 1 red

set nf [open out.nam w]

$ns namtrace-all $nf proc finish {} { global ns nf

$ns flush-trace close $nf

exec nam out.nam & exit 0

}

set n0 [$ns node] set n1 [$ns node] set n2 [$ns node] set n3 [$ns node]

$ns duplex-link $n0 $n2 2Mb 10ms DropTail

$ns duplex-link $n1 $n2 2Mb 10ms DropTail $ns duplex- link $n2 $n3 1.7Mb 20ms DropTail $ns duplex-link-op $n0

$n2 orient right-down $ns duplex-link-op $n1 $n2 orient right-up $ns duplex-link-op $n2 $n3 orient right $ns queue- limit $n0 $n2 10

$ns queue-limit $n1 $n2 10

$ns queue-limit $n2 $n3 10 set tcp1 [new Agent/TCP]

$tcp1 set fid\_ 0

$ns attach-agent $n1 $tcp1

set sinkTCP3 [new Agent/TCPSink] $ns attach- agent $n3 $sinkTCP3

$ns connect $tcp1 $sinkTCP3 set udp0 [new Agent/UDP]

$udp0 set fid\_ 1

$ns attach-agent $n0 $udp0 set null0 [new Agent/Null]

$ns attach-agent $n3 $null0

$ns connect $udp0 $null0

set ftp0 [new Application/FTP]

$ftp0 attach-agent $tcp1

set cbr0 [new Application/Traffic/CBR] $cbr0 set packetSize\_ 1000

$cbr0 set interval 0.010

$cbr0 attach-agent $udp0

$ns at 0.5 "$ftp0 start"

$ns at 4.0 "$ftp0 stop"

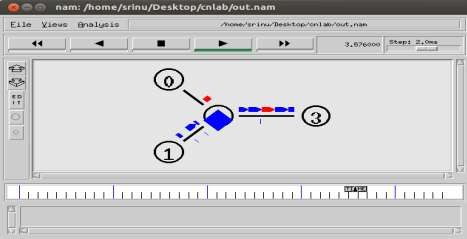
$ns at 0.1 "$cbr0 start"

$ns at 4.5 "$cbr0 stop"

$ns at 5.0 "finish"

$ns run

**OUTPUT :**



**AIM:**

**WEEK-11**

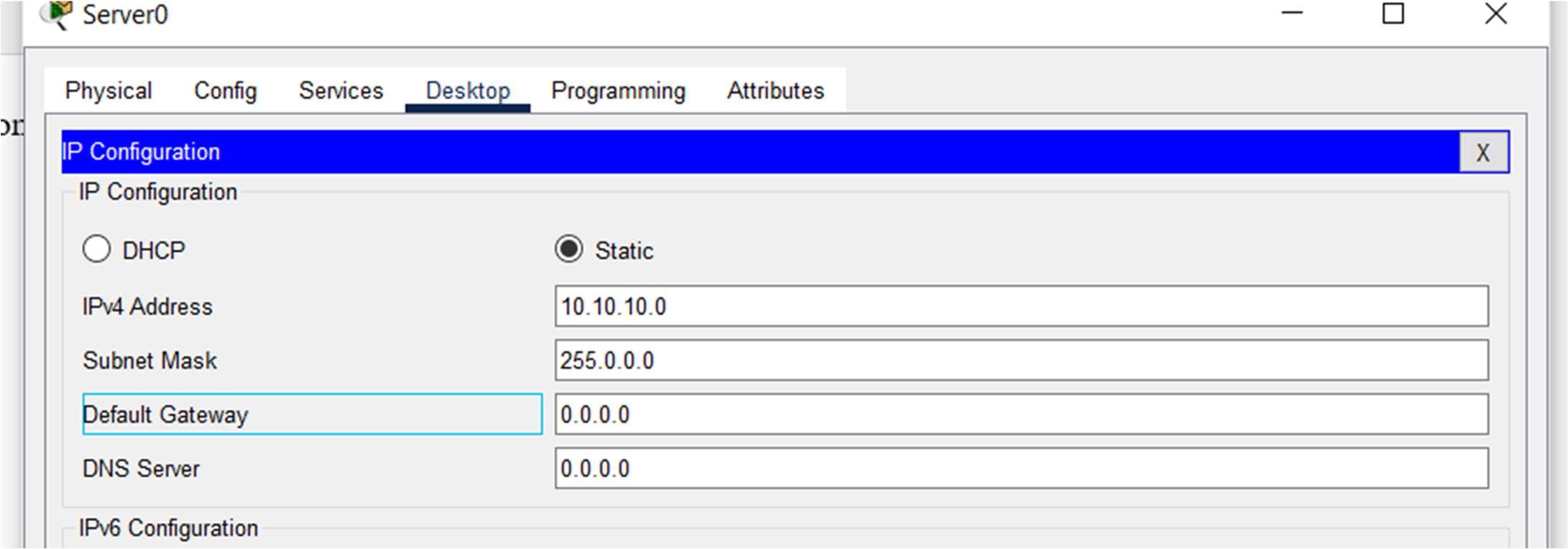
Configuring any two application layer protocols using packet tracer.

**FTP PROTOCOL:-**

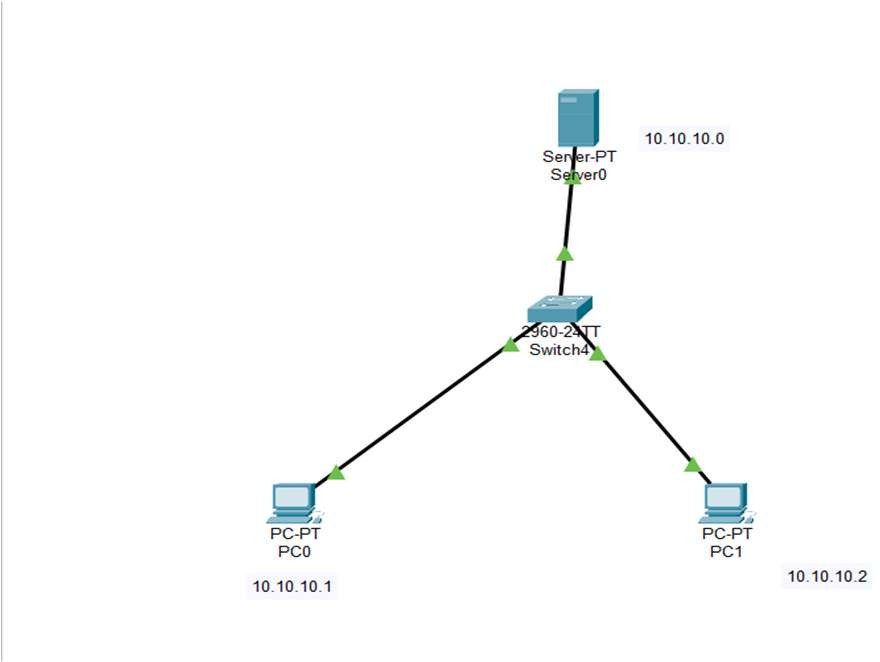
1. Open packet tracer and place the server,switch and 2 pc’s from the workspace.



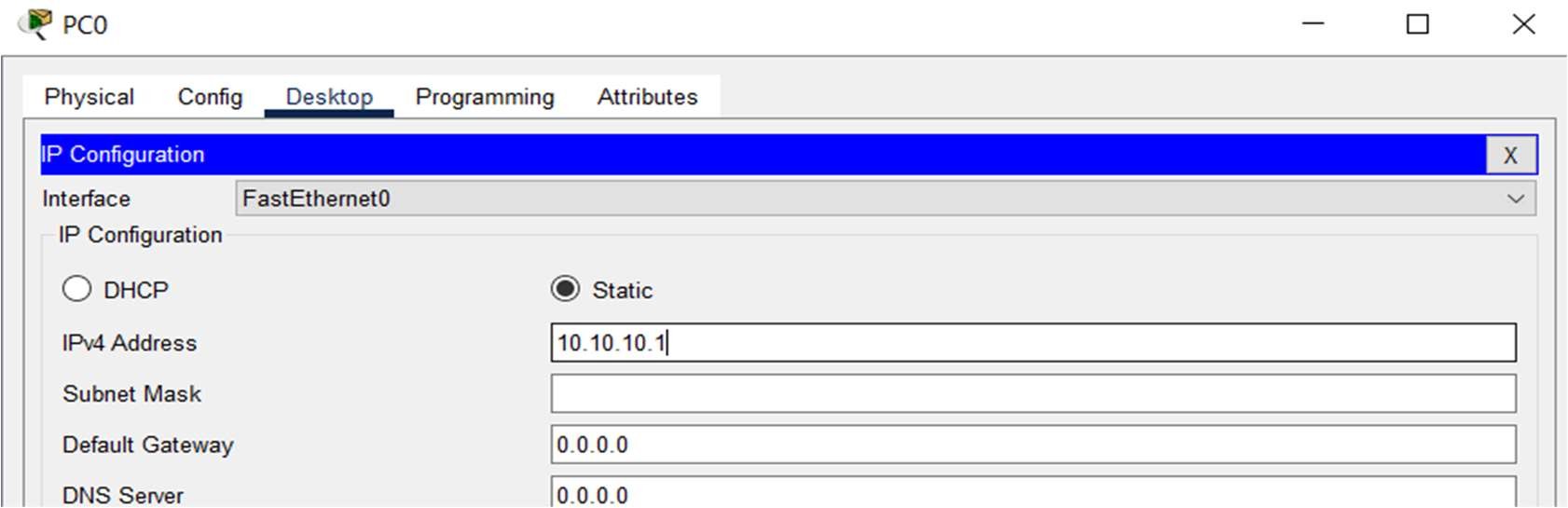
1. Assign ip address for server.



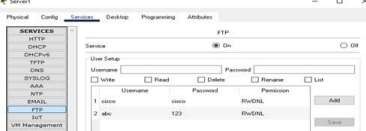
1. Connect all of the devices with straight through cable.



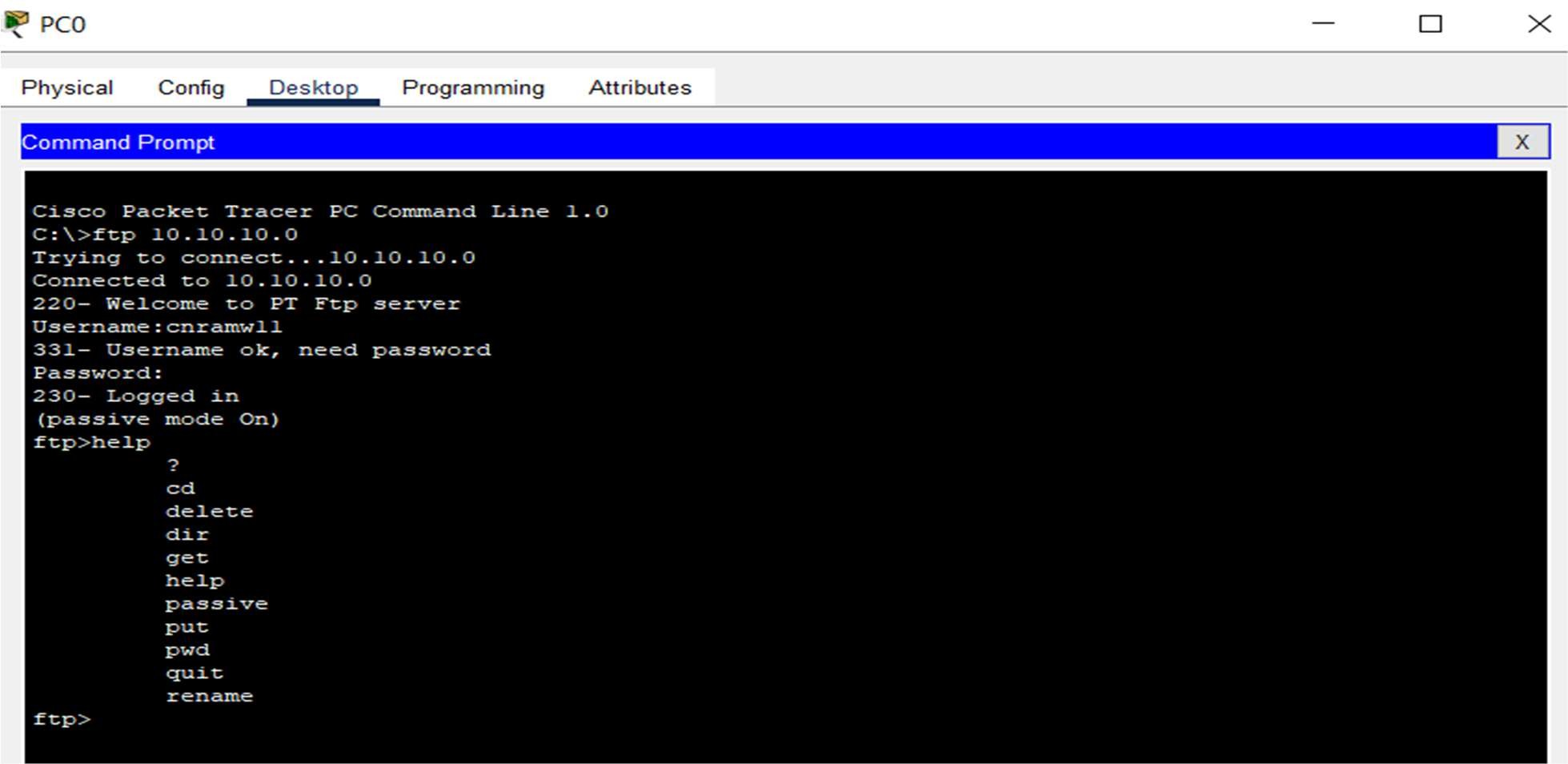
1. Assign ip’s for both the systems.



1. And in server configuration off all the protocols and on the ftp protocol and assign username and password and add it.



1. Now open the command prompt of any pc and type ftp 10.10.10.0,Then server is connected, then we can use commands like delete a file or directory.

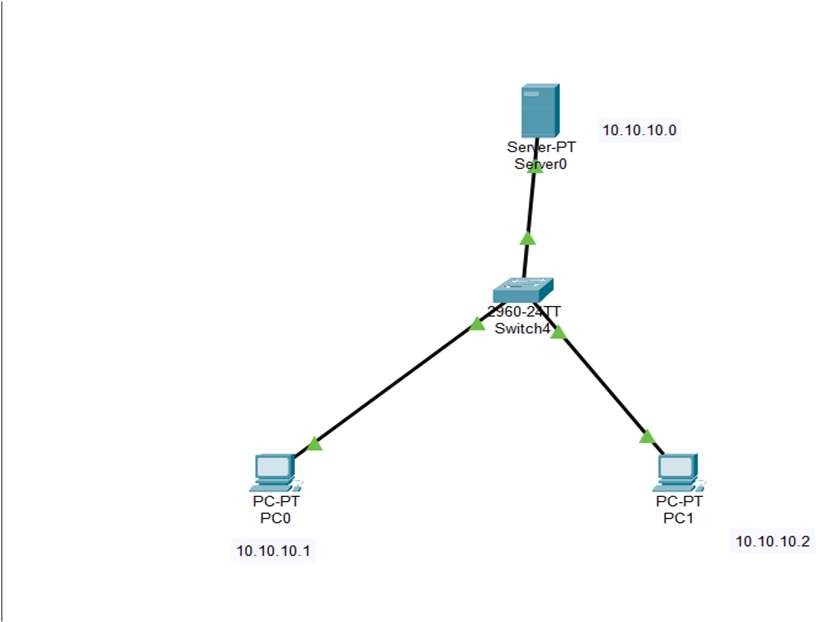


1. Finally messages are shared.

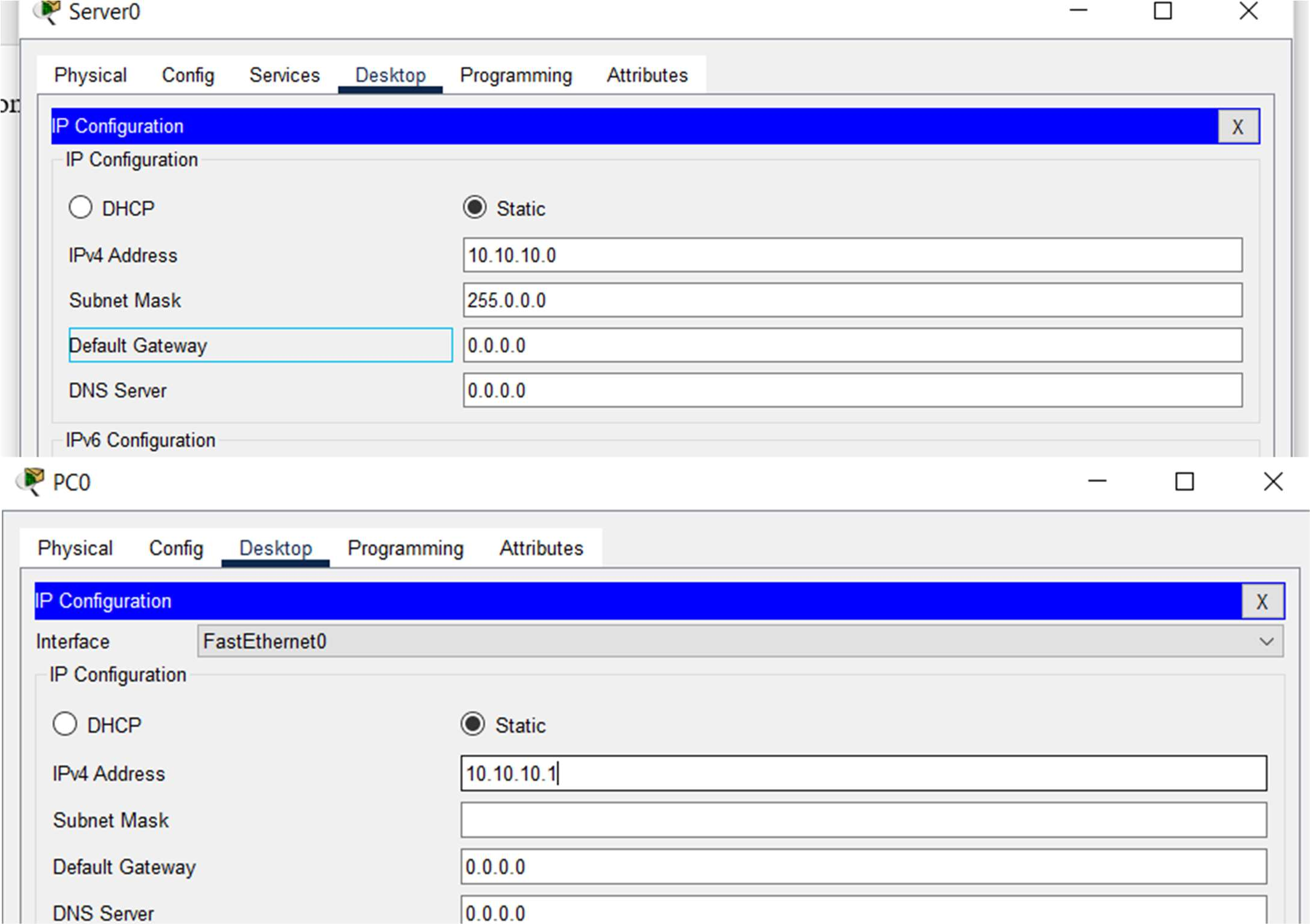


**STEPS : HTTP PROTOCOL**

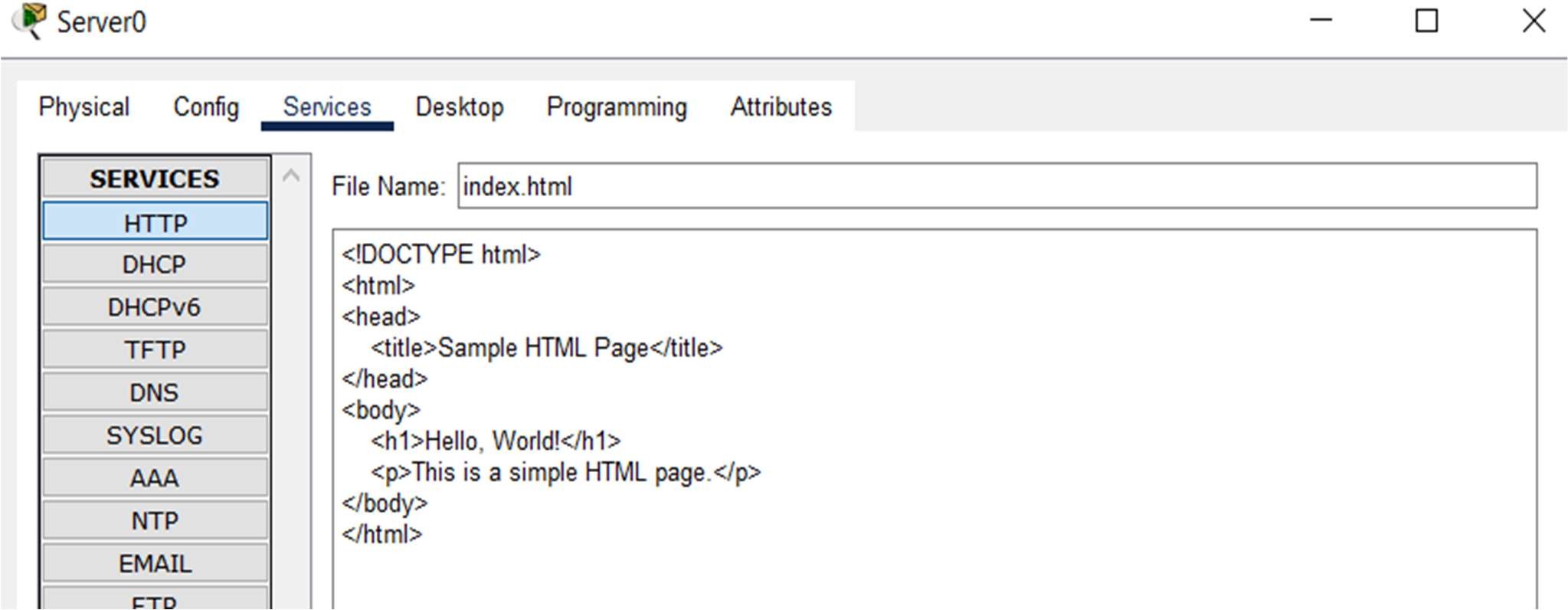
1. Open packet tracer and place the server,switch and 2 pc’s from the workspace and Connect all of the devices with straight through cable.



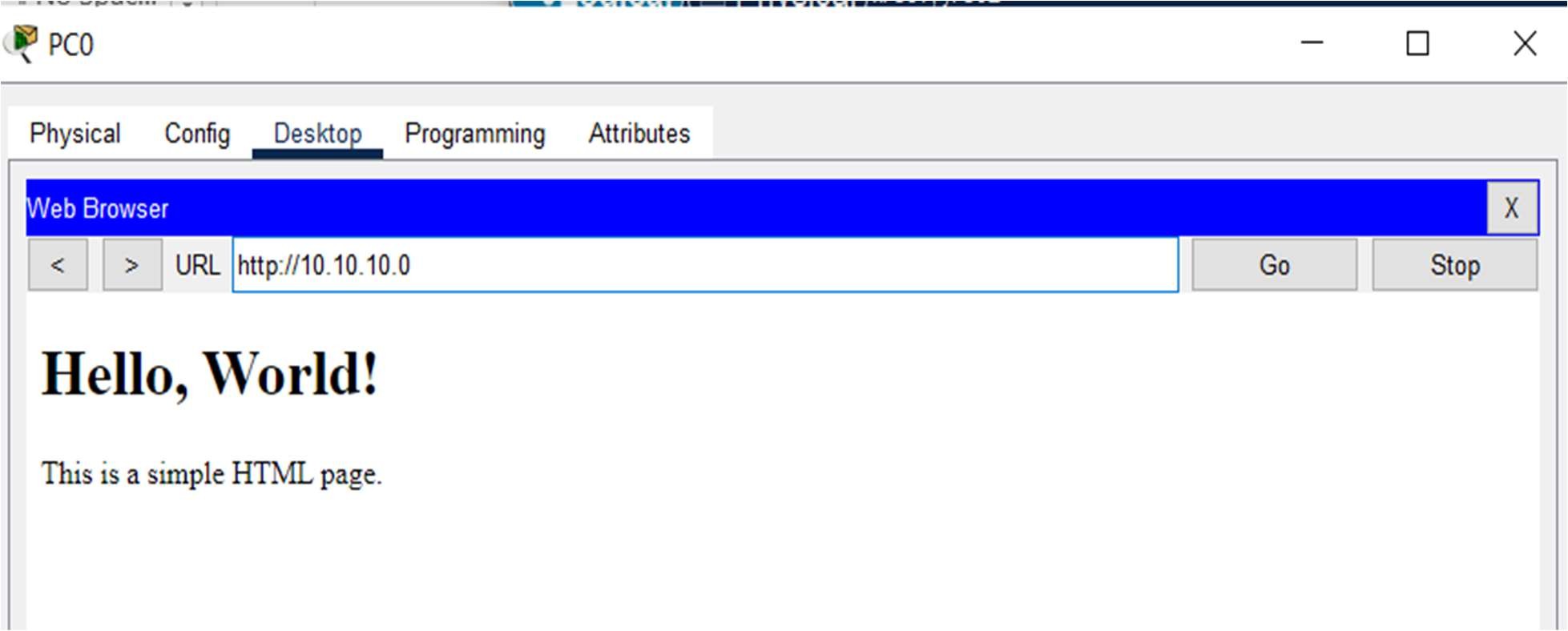
1. Assign ip’s to server and pc’s



1. Off all protocols except Http and edit one html file in http window and save it.



1. Then go to any pc’s web browser and type the server url then u get the text written in htmlfile.



**WEEK-12**

**AIM :** Implementation of wireless networks using NS2 Simulation Tool.

**CODE :**

set ns [new Simulator]

set tracefile [open wireless.tr w]

$ns trace-all $tracefile

set namfile [open wireless.nam w]

$ns namtrace-all-wireless $namfile 500 500

set topo [new Topography]

$topo load\_flatgrid 500 500

create-god 6

set val(chan) Channel/WirelessChannel ;

$ns node-config -adhocRouting AODV -llType LL \

-macType Mac/802\_11 -ifqType Queue/DropTail/PriQueue \ -ifqLen 50 - antType Antenna/OmniAntenna \

-propType Propagation/TwoRayGround -phyType Phy/WirelessPhy \ -channel [new

$val(chan)] -topoInstance $topo \

-agentTrace ON -routerTrace OFF\

-macTrace ON \

-movementTrace OFF

set n0 [$ns node] set n1 [$ns node] set n2 [$ns node] set n3 [$ns node] set n4 [$ns node] set n5 [$ns node]

$n0 random-motion 0 $n1 random- motion 0 $n2 random-motion 0 $n3 random-motion 0 $n4 random- motion 0 $n5 random-motion 0

|  |  |  |  |
| --- | --- | --- | --- |
| $ns initial\_node\_pos | $n0 | 20 | $ns |
| initial\_node\_pos $n1 |  | 20 | $ns |
| initial\_node\_pos $n2 |  | 20 | $ns |
| initial\_node\_pos $n3 |  | 20 | $ns |

initial\_node\_pos $n4 20

$ns initial\_node\_pos $n5 50

$n0 set X\_ 10.0

$n0 set Y\_ 20.0

$n0 set Z\_ 0.0

$n1 set X\_ 210.0

$n1 set Y\_ 230.0

$n1 set Z\_ 0.0

$n2 set X\_ 100.0

$n2 set Y\_ 200.0

$n2 set Z\_ 0.0

$n3 set X\_ 150.0

$n3 set Y\_ 230.0

$n3 set Z\_ 0.0

$n4 set X\_ 430.0

$n4 set Y\_ 320.0

$n4 set Z\_ 0.0

$n5 set X\_ 270.0

$n5 set Y\_ 120.0

$n5 set Z\_ 0.0

$ns at 1.0 "$n1 setdest 490.0 340.0 25.0" $ns at 1.0 "$n4

setdest 300.0 130.0 5.0" $ns at 1.0 "$n5 setdest 190.0

440.0 15.0"

$ns at 5.0 "$n5 setdest 100.0 200.0 30.0" set tcp [new Agent/TCP]

set sink [new Agent/TCPSink]

$ns attach-agent $n0 $tcp

$ns attach-agent $n5 $sink

$ns connect $tcp $sink

set ftp [new Application/FTP]

$ftp attach-agent $tcp

$ns at 1.0 "$ftp start";

set udp [new Agent/UDP] set null [new Agent/Null]

$ns attach-agent $n2 $udp

$ns attach-agent $n3 $null

$ns connect $udp $null

set cbr [new Application/Traffic/CBR] $cbr attach- agent $udp

$ns at 1.0 "$cbr start"

$ns at 10.0 "finish" proc finish {} {

global ns tracefile namfile

$ns flush-trace close $tracefile close $namfile

exec nam wireless.nam & exit 0

}

puts "Starting Simulation"

$ns run

**OUTPUT**

