



IMPLEMENTING RING AND STAR TOPOLOGY USING NS2

CSE1004(NETWORK AND COMMUNICATION)LAB:L53-L54



**MARCH 24, 2022
ANIRUDH VADERA
20BCE2940**

SIMULATION AND NS2:

Simulation is the process of *learning by doing*. Whenever there is something new in the world, we try to analyze it first by examining it and in the process get to learn a lot of things. This entire course is called **Simulation**.

Correlating to this process, in order to understand all the complexities one needs to model the entire role-play in form of computer simulation, the need is to build artificial objects and assign them roles dynamically.

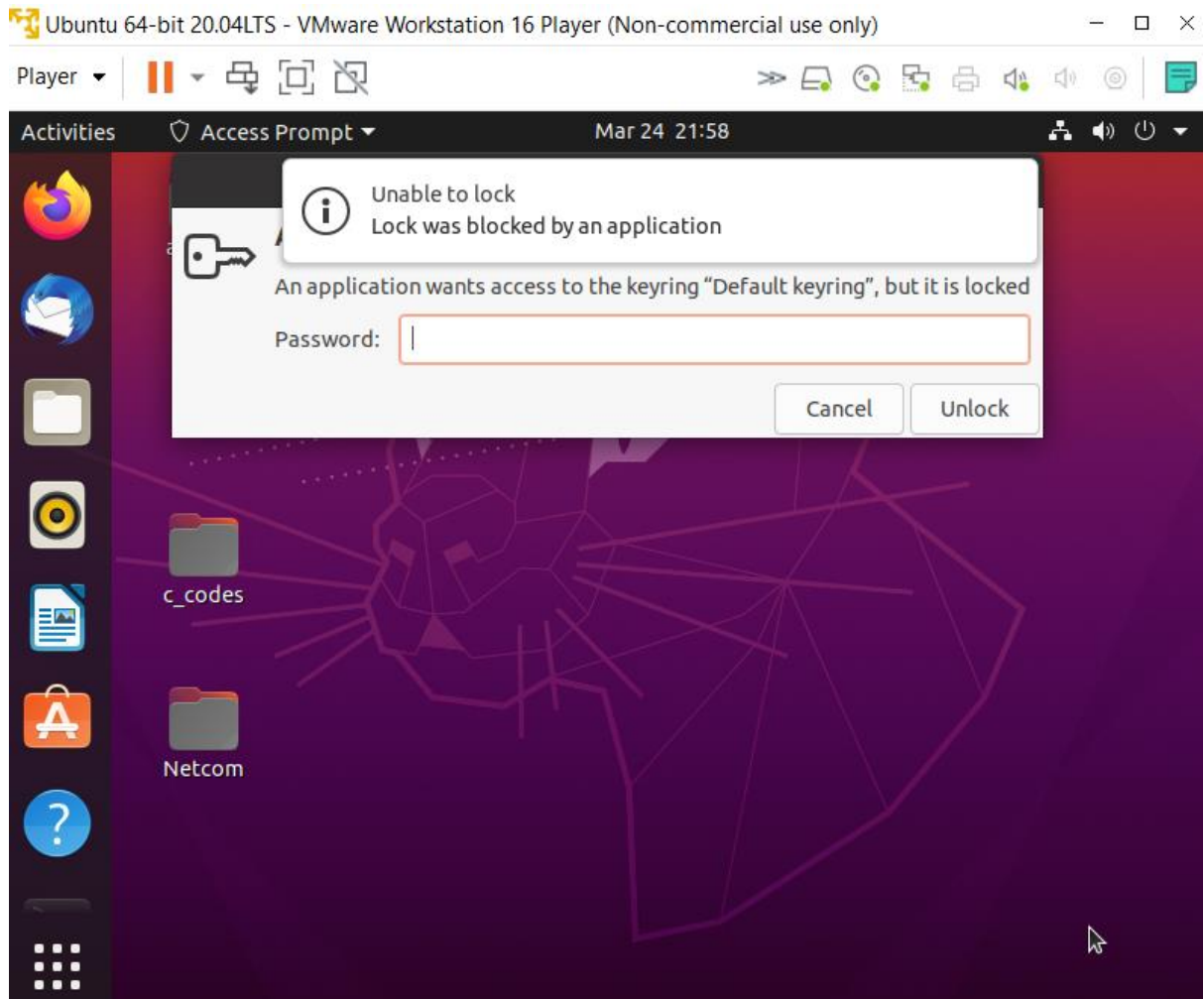
Computer simulation is the designing of a theoretical physical system on a digital computer with emphasis on model designing, execution, and analysis. After the creation of the mathematical model, the most important step is to create a computer program for updating the state and event variables through time (by time slicing or event scheduling). If this simulation is carried out successively in parallel computers, it is called *Parallel* or *Distributed simulation*.

Network simulation (NS) is one of the types of simulation, which is used to simulate the networks such as in MANETs, VANETs, etc. It provides simulation for routing and multicast protocols for both wired and wireless networks. NS is licensed for use under version 2 of the GNU (General Public License) and is popularly known as **NS2**. It is an object-oriented, discrete event-driven simulator written in C++ and Otcl/Tcl.

NS-2 can be used to implement network protocols such as TCP and UDP, traffic source behavior such as FTP, Telnet, Web, CBR, and VBR, router queues management mechanism such as Drop Tail, RED, and CBQ, routing algorithms, and many more. In ns2, C++ is used for detailed protocol implementation and Otcl is used for the setup. The compiled C++ objects are made available to the Otcl interpreter and in this way, the ready-made C++ objects can be controlled from the OTcl level.

PROCEDURE:

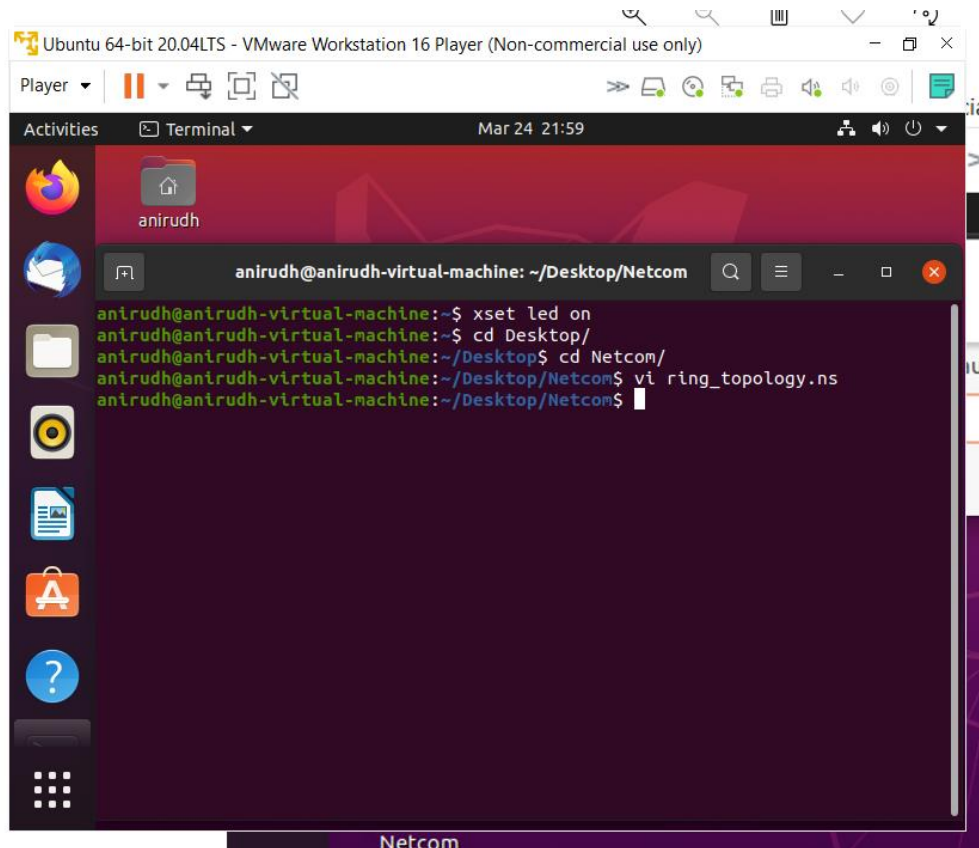
- **Open ur vmware having a Ubuntu linux distribution:**



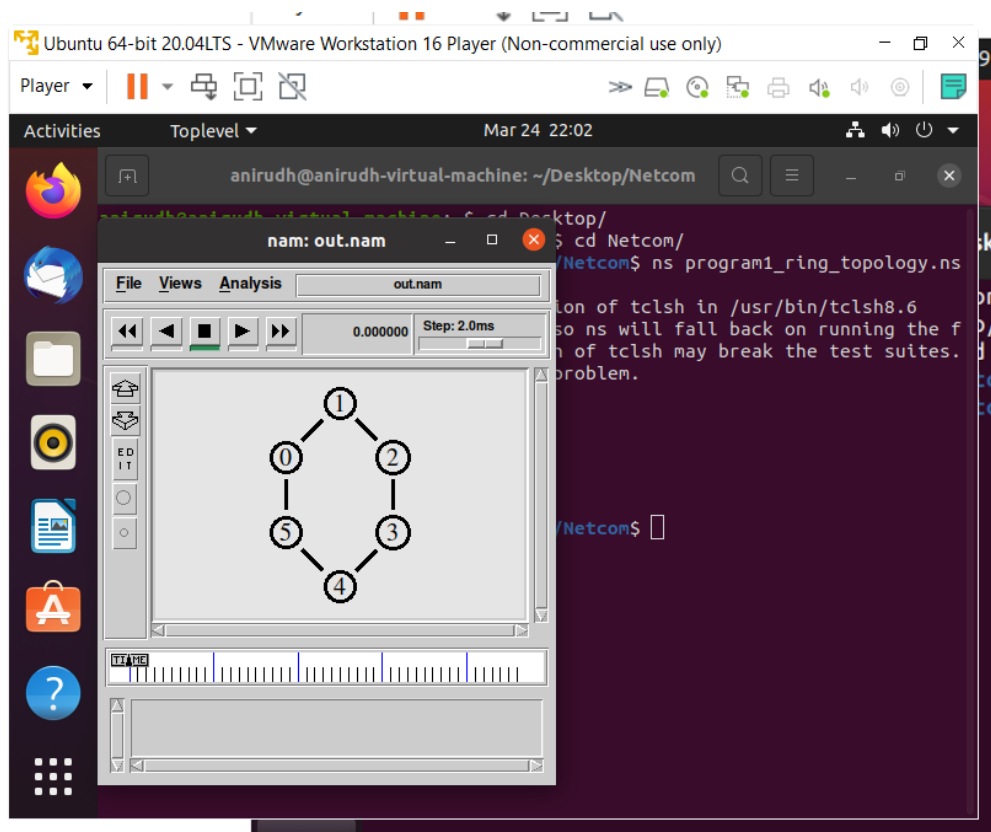
- **U must have all the necessary c and ns2 compilers already installed in your linux system.**
- **Using terminal open a vi editor and write ur ns2 code with the command `ns filename.ns`**

ANIRUDH VADERA

IMPLEMENTING RING AND STAR TOPOLOGY USING NS2



- After saving the file run `ns filename.ns` command to execute it using `nam`

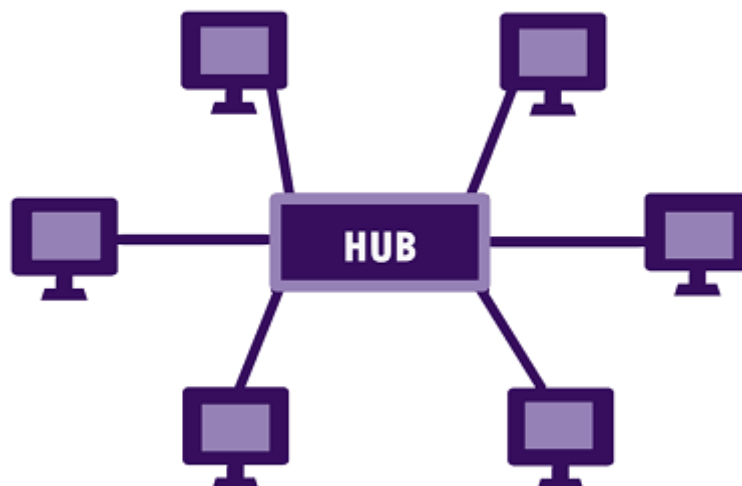


QUESTION:

1. Design and study the performance of a topology for a Local Area Network (LAN) in which all nodes are individually connected to a central connection point, like a hub or a switch with NS2 simulator

Star Topology:

A star topology, sometimes known as a star network, is a network topology in which each device is connected to a central hub. It is one of the most prevalent computer network configurations, and it's by far the most popular Network Topology. In this network arrangement, all devices linked to a central network device are displayed as a star.



EXPLANATION:

- Consider a small network with five nodes n0, n1, n2, n3, n4, forming a star topology.
- The node n0 is at the center.
- Node n2 is a TCP source, which transmits packets to node n4 (a TCP sink) through the node n0(Blue).

ANIRUDH VADERA
IMPLEMENTING RING AND STAR TOPOLOGY USING NS2

- Node n3 is another TCP source, and sends TCP packets to node n5 through n0(Green).
- Node n1 is another TCP source, and sends TCP packets to node n3 through n0(Red).
- The duration of the simulation time is 5 seconds.

CODE:

```
# ANIRUDH VADERA 20BCE2940
```

```
#Create a simulator object
```

```
set ns [new Simulator]
```

```
#Routing Protocol used is Distance Vector
```

```
$ns rtproto DV
```

```
#Open the nam trace file
```

```
set nf [open out.nam w]
```

```
$ns namtrace-all $nf
```

```
#Define a 'finish' procedure
```

```
proc finish {} {
```

```
    global ns nf
```

```
    $ns flush-trace
```

```
    #Close the trace file
```

```
    close $nf
```

```
    #Execute nam on the trace file
```

```
    exec nam out.nam &
```

```
    exit0
```

```
}
```

```
#Create nodes
```

ANIRUDH VADERA
IMPLEMENTING RING AND STAR TOPOLOGY USING NS2

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 shape square

#Create links between the nodes

\$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 duplex-link \$n0 \$n5 1Mb 10ms DropTail

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

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

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

\$ns duplex-link-op \$n0 \$n4 orient left-down

\$ns duplex-link-op \$n0 \$n5 orient left-up

#Create a TCP agent and attach it to node n0

set tcp0 [new Agent/TCP]

\$tcp0 set class_ 1

\$ns attach-agent \$n1 \$tcp0

set sink0 [new Agent/TCPSink]

ANIRUDH VADERA
IMPLEMENTING RING AND STAR TOPOLOGY USING NS2

```
$ns attach-agent $n3 $sink0
```

```
#Connect the traffic sources with the traffic sink
```

```
$ns connect $tcp0 $sink0
```

```
# Create a CBR traffic source and attach it to tcp0
```

```
set cbr0 [new Application/Traffic/CBR]
```

```
$cbr0 set packetSize_ 500
```

```
$cbr0 set interval_ 0.01
```

```
$cbr0 attach-agent $tcp0
```

```
$ns color 1 red
```

```
$ns at 4.5 "$cbr0 stop"
```

```
#Schedule events for the CBR agents
```

```
$ns at 0.5 "$cbr0 start"
```

```
set tcp1 [new Agent/TCP]
```

```
$tcp1 set class_ 2
```

```
$ns attach-agent $n2 $tcp1
```

```
set sink1 [new Agent/TCPSink]
```

```
$ns attach-agent $n4 $sink1
```

```
#Connect the traffic sources with the traffic sink
```

```
$ns connect $tcp1 $sink1
```

```
$ns color 2 blue
```

```
# Create a CBR traffic source and attach it to tcp1
```

```
set cbr1 [new Application/Traffic/CBR]
```

```
$cbr1 set packetSize_ 300
```


ANIRUDH VADERA
IMPLEMENTING RING AND STAR TOPOLOGY USING NS2

\$cbr1 set interval_ 0.02

\$cbr1 attach-agent \$tcp1

\$ns at 3.5 "\$cbr1 stop"

#Schedule events for the CBR agents

\$ns at 0.3 "\$cbr1 start"

set tcp2 [new Agent/TCP]

\$tcp2 set class_ 3

\$ns attach-agent \$n3 \$tcp2

set sink2 [new Agent/TCPSink]

\$ns attach-agent \$n5 \$sink2

#Connect the traffic sources with the traffic sink

\$ns connect \$tcp2 \$sink2

\$ns color 3 green

Create a CBR traffic source and attach it to tcp2

set cbr2 [new Application/Traffic/CBR]

\$cbr2 set packetSize_ 500

\$cbr2 set interval_ 0.01

\$cbr2 attach-agent \$tcp2

\$ns at 4.5 "\$cbr2 stop"

#Schedule events for the CBR agents

\$ns at 0.5 "\$cbr2 start"

ANIRUDH VADERA IMPLEMENTING RING AND STAR TOPOLOGY USING NS2

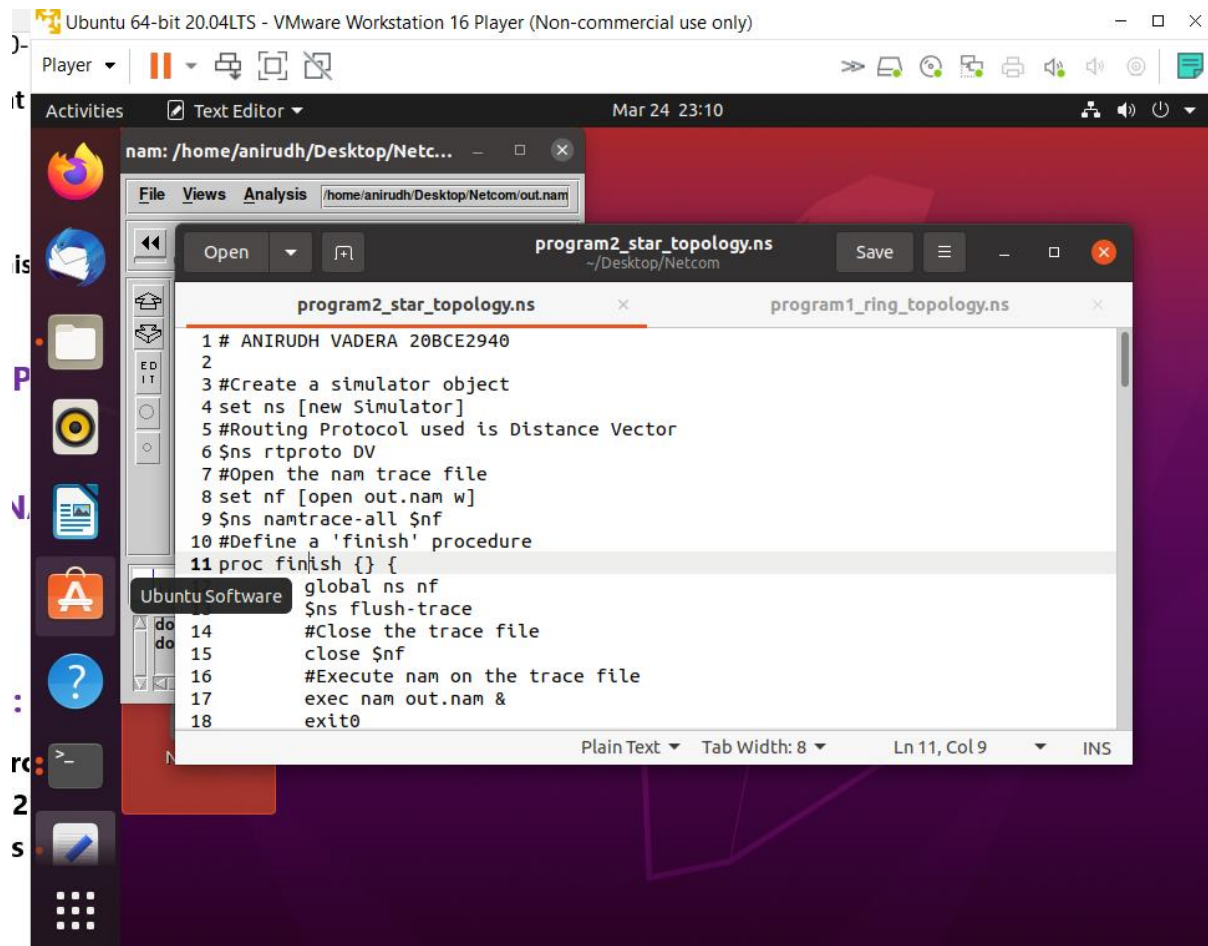
Break link n0---n3 at 4ms

\$ns rtmodel-at 4 down \$n0 \$n3

\$ns at 5.0 "finish"

\$ns run

CODE SNAPSHOTS:



The screenshot shows a Netcom NS2 script editor window titled "program2_star_topology.ns" with the following code:

```
1 # ANIRUDH VADERA 20BCE2940
2
3 #Create a simulator object
4 set ns [new Simulator]
5 #Routing Protocol used is Distance Vector
6 $ns rtproto DV
7 #Open the nam trace file
8 set nf [open out.nam w]
9 $ns namtrace-all $nf
10 #Define a 'finish' procedure
11 proc finish {} {
12     global ns nf
13     $ns flush-trace
14     #Close the trace file
15     close $nf
16     #Execute nam on the trace file
17     exec nam out.nam &
18     exit 0
19 }
```

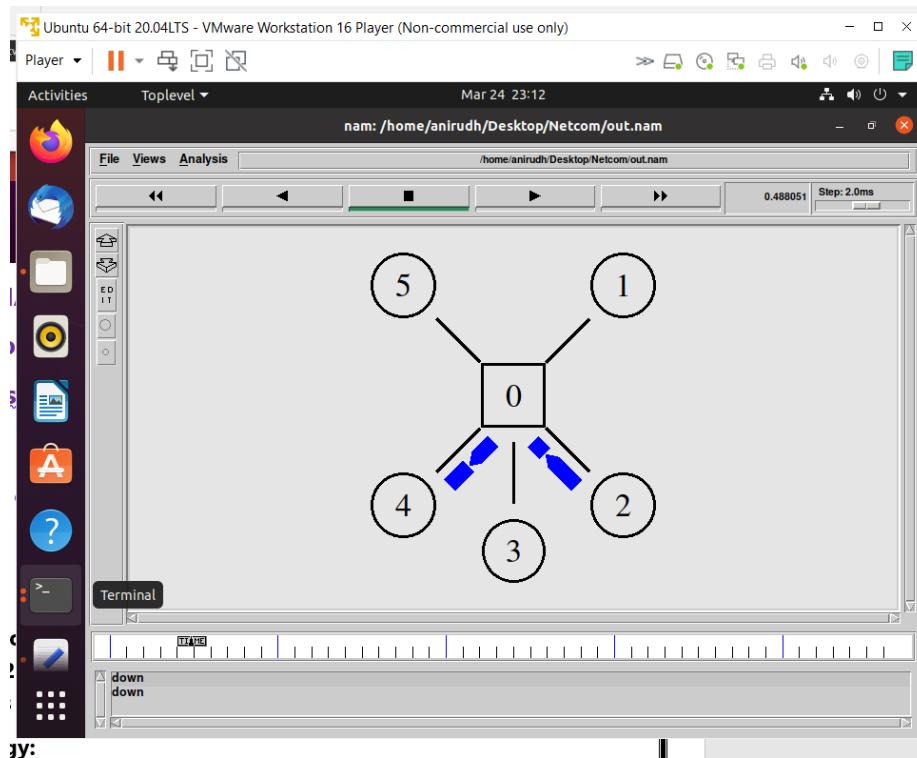
The editor also shows a second tab titled "program1_ring_topology.ns". The status bar at the bottom indicates "Ln 11, Col 9" and "INS".

ANIRUDH VADERA
IMPLEMENTING RING AND STAR TOPOLOGY USING NS2

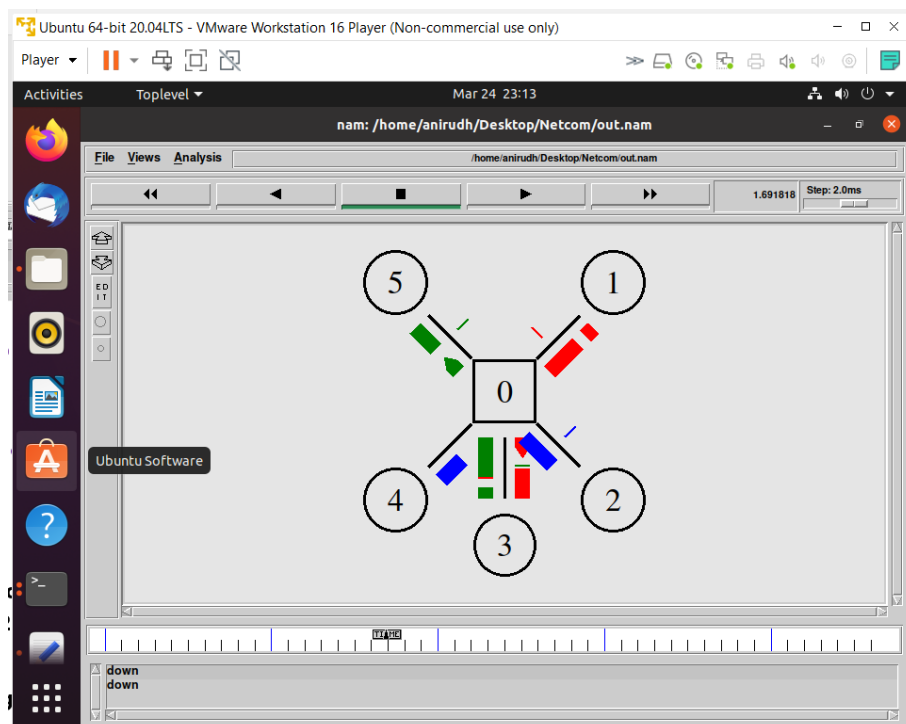
OUTPUT (NAM SCREENSHOTS):

Sending from 0 sec to 5 sec:

Before 0.5 sec(only blue packets are sent):



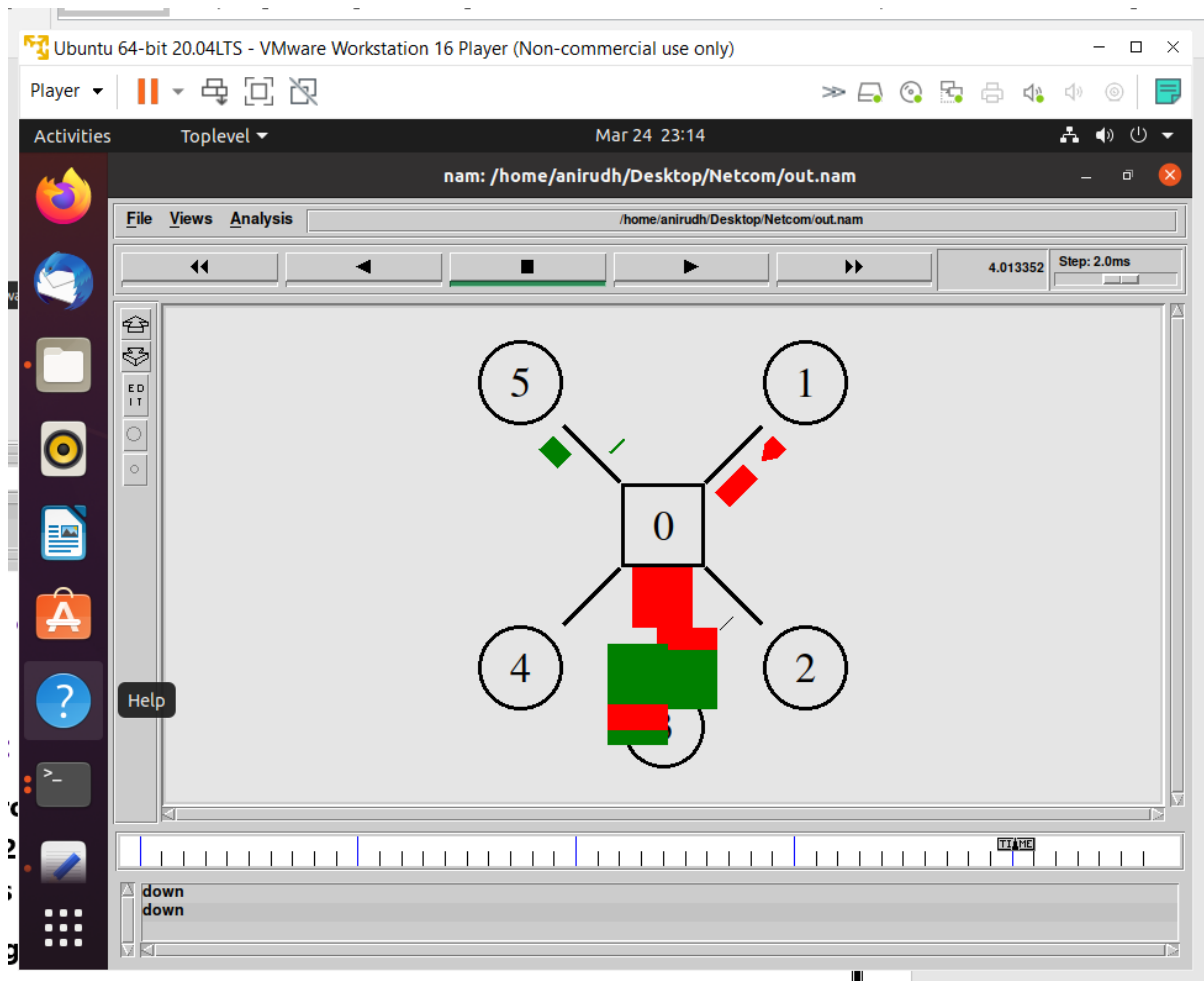
From 0.5 to 4 sec:



ANIRUDH VADERA

IMPLEMENTING RING AND STAR TOPOLOGY USING NS2

Packet loss at 4 sec:

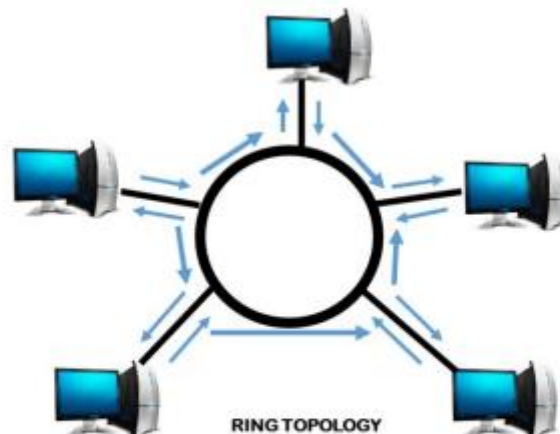


QUESTION:

2. Write a program to implement a small network with five nodes (n0, n1, n2, n3, n4) forming a ring topology. Simulate the transfer of packets from node n0 to n3 through n4 for 10 seconds.

Ring Topology:

A ring topology is a network architecture in which devices are connected in a ring structure and send information to each other based on their ring node's neighbouring node. As compared to the bus topology, a ring topology is highly efficient and can handle heavier loads. Because packets may only travel in one direction, most Ring Topologies are referred to as one-way unidirectional ring networks. Generally, Bidirectional and Unidirectional are the two types of ring topology. On the basis of devices that are being linked together to form a network, several kinds of ring topology setups work differently.



CODE:

```
#Create a simulator object
```

```
set ns [new Simulator]
```

```
#Routing Protocol used is Distance Vector
```

```
$ns rtproto DV
```

ANIRUDH VADERA
IMPLEMENTING RING AND STAR TOPOLOGY USING NS2

#Open the nam trace file

set nf [open out.nam w]

\$ns namtrace-all \$nf

#Define a 'finish' procedure

proc finish {} {

global ns nf

\$ns flush-trace

#Close the trace file

close \$nf

#Execute nam on the trace file

exec nam out.nam &

exit0

}

#Create nodes

set n0 [\$ns node]

set n1 [\$ns node]

set n2 [\$ns node]

set n3 [\$ns node]

set n4 [\$ns node]

#Create links between the nodes

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

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

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

ANIRUDH VADERA
IMPLEMENTING RING AND STAR TOPOLOGY USING NS2

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

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

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

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

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

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

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

#Create a TCP agent and attach it to node n0

set tcp0 [new Agent/TCP]

\$tcp0 set class_ 0

\$ns attach-agent \$n0 \$tcp0

set tcp4 [new Agent/TCP]

\$tcp4 set class_ 4

\$ns attach-agent \$n0 \$tcp4

#Create a TCP Sink agent (a traffic sink) for TCP and attach it to node n3

set sink0 [new Agent/TCPSink]

\$ns attach-agent \$n3 \$sink0

set sink4 [new Agent/TCPSink]

\$ns attach-agent \$n3 \$sink4

ANIRUDH VADERA
IMPLEMENTING RING AND STAR TOPOLOGY USING NS2

#Connect the traffic sources with the traffic sink

\$ns connect \$tcp0 \$sink0

\$ns connect \$tcp4 \$sink4

Create a CBR traffic source and attach it to tcp0

set cbr0 [new Application/Traffic/CBR]

\$cbr0 set packetSize_ 500

\$cbr0 set interval_ 0.01

\$cbr0 attach-agent \$tcp0

\$ns color 0 red

\$ns color 4 green

Create a CBR traffic source and attach it to tcp0

set cbr4 [new Application/Traffic/CBR]

\$cbr4 set packetSize_ 500

\$cbr4 set interval_ 0.01

\$cbr4 attach-agent \$tcp4

#Schedule events for the CBR agents

\$ns at 0.5 "\$cbr0 start"

\$ns at 2.0 "\$cbr4 start"

\$ns at 8.0 "\$cbr4 stop"

\$ns at 9.5 "\$cbr0 stop"

ANIRUDH VADERA IMPLEMENTING RING AND STAR TOPOLOGY USING NS2

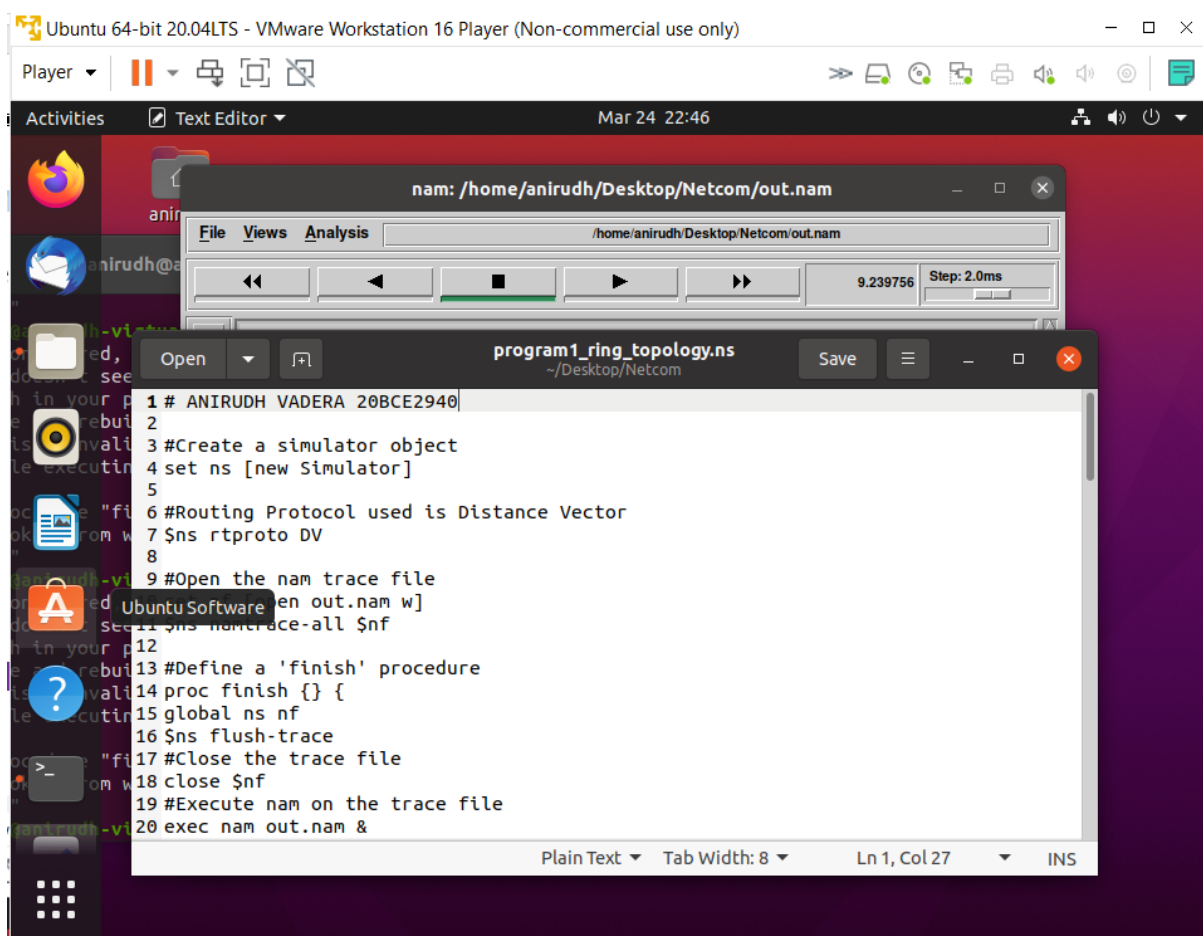
#Call the finish procedure after 10 seconds of simulation time

\$ns at 10.0 "finish"

#Run the simulation

\$ns run

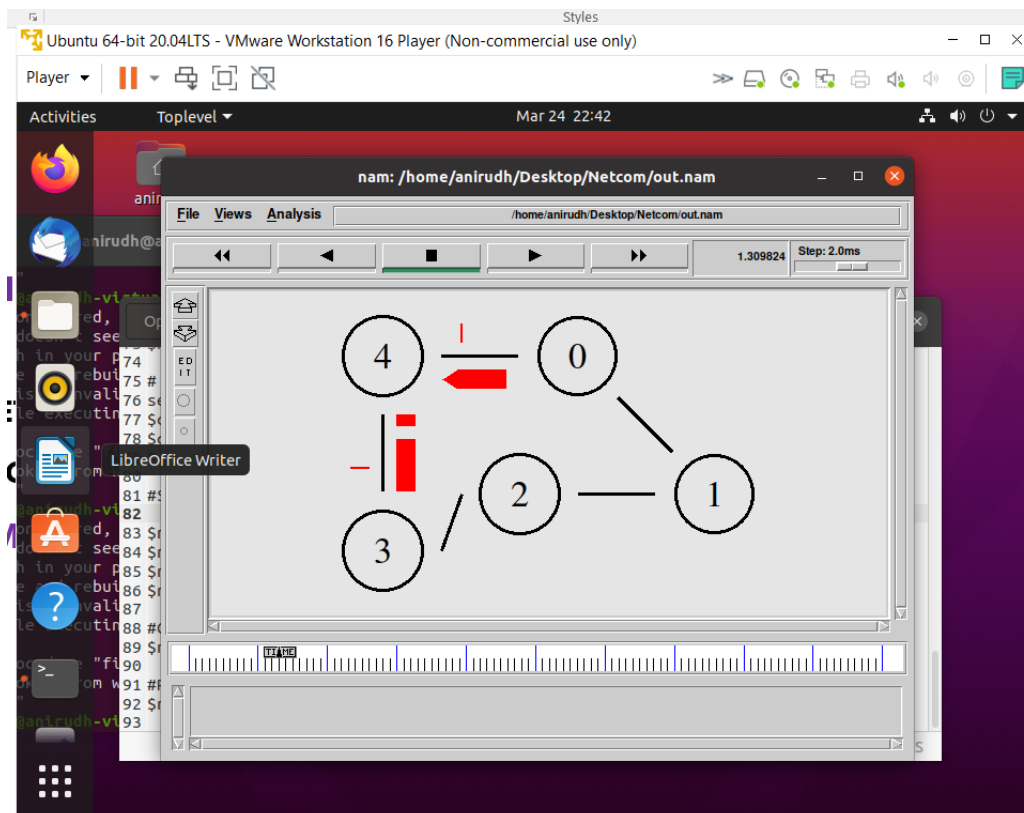
CODE SNAPSHOTS:



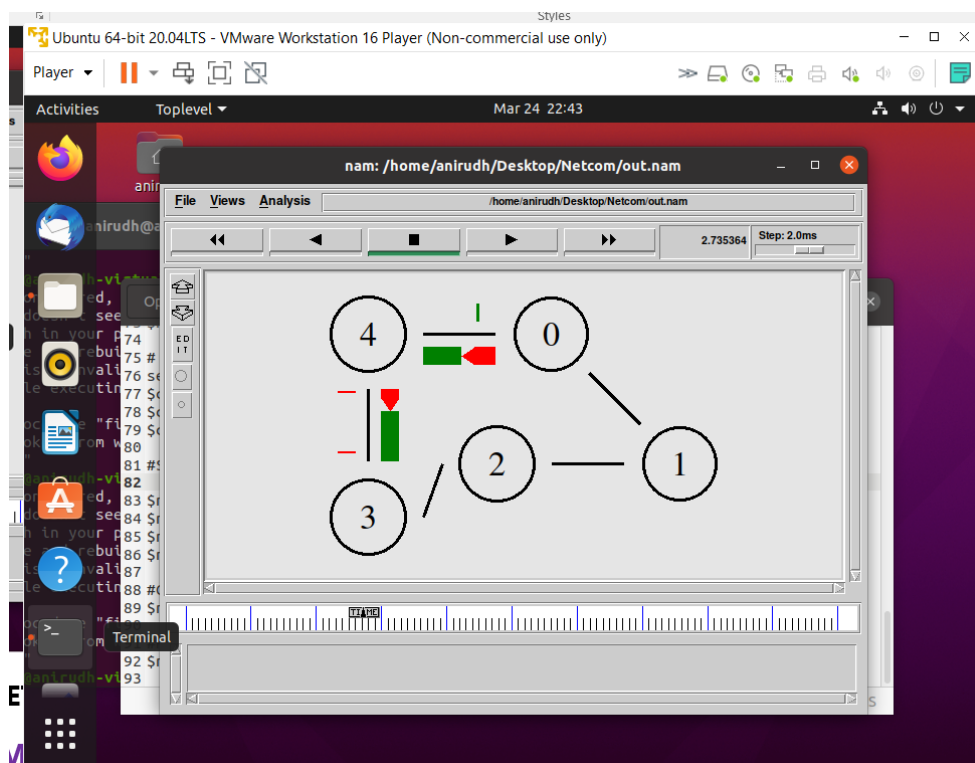
ANIRUDH VADERA
IMPLEMENTING RING AND STAR TOPOLOGY USING NS2

OUTPUT (NAM SCREENSHOTS):

BEFORE 2 SECS:



SENDING PACKETS: FROM 2 TO 8 SECS:



ANIRUDH VADERA
IMPLEMENTING RING AND STAR TOPOLOGY USING NS2

AFTER 8 SECS WHEN ONE LINK IS OVER(GREEN ONE IS FINISHED):

