**Name: Madhukumar Paka**

**Class: SEIT2**

**Roll No : 15**

**Batch: B3**

**NETWORK LAB**

**Index**

|  |  |
| --- | --- |
| **Experiment**  **No** | **Aim** |
| 1 | Basic commands of windows and UNIX operating system, editor commands (eg nano/vi editor etc) |
| 2 | Understanding basic networking commands |
| 3 | Installation and configuration of NS2. |
| 4 | Introduction to TCL hello programming. |
| 5 | A implementation of specific network topology with respect to number of nodes an physical network layer configuration. |
| 6 | Simulation of network with routing protocols and traffic consideration (TCP/ UDP) using NAM |
| 7 | Simulation of network with specific routing protocols (distance vector/ link state) |
| 8 | To write a ns2 program for implementing unicast routing protocol |
| 9 | Write and execute TCL scripts to create Bus topology, ring topology and star topology. |
| 10 | Simulating of a network with multicast routing protocol. |

**Experiment No 1**

**Aim**: Basic commands of windows and UNIX operating system, editor commands (eg nano/vi editor etc)

**Theory:** Basic commands for UNIX operating system are:

1. pwd: (Present Working Directory) pwd shows the directory currently opened.
2. ls: ls commands lists all files in current directory.
3. cd : cd command is used to change directory and navigate in filesystem.
4. file: file command is used to determine the type of the specified file.
5. head: head command prints the starting few lines of a file.
6. tail: tail command prints last few lines of a file.
7. cat: cat command is used to display the contents of a file. It is also used to change the contents of a file.
8. cp: cp command is used to copy a file to another directory.
9. mv: mv command is used to move a file from one directory to another.
10. mkdir: mkdir command is used to create a new directory.
11. rmdir: This command is used to delete a empty directory
12. rm: It is used to delete a file
13. which: which command is used to identify the location of a given executable
14. whereis: whereis command is used to find the location of source/binary file of a command and manuals sections for a specified file in Linux system.
15. locate: This command is used to search for files in filesystem very easily.
16. find: find command is used to search for files in filesystem but with some user specified criteria.
17. ps: ps command lists current running processes.
18. w: w command is used to show which users are logged in and what are they doing.
19. id: prints the user-id and groups id’s
20. df: It reports filesystem disk space usage
21. du: du command reports filesystem disk space usage in a particular directory

22 . top: top command displays CPU processes and activity of computer in real time.

1. free: This command displays amount of free and used memory in the system.
2. cat /proc/cpuinfo : Displays information about CPU
3. cat /proc/meminfo: Displays information about current memory usage
4. uname –a: Prints system information to screen like kernel version, machine type etc.
5. echo: echo command is used to print on the terminal . It is used to shell scripts
6. more: more command prints all the contents of a file
7. less: It displays the contents of a file or a command output, one page at a time.
8. grep: It is command used for searching plain-text data sets for lines that match a regular expression.

31 lpr: he lpr command transmits and manages print jobs

32. sort: prints contents of a file in sorted order.

# VI Editor commands:

Vim: It opens the VIM text editor

vim filename: It is used to open a file in vim. If file does not exists then an empty new file is opened.

i : This command is used to command mode to enter in insert mode.

Esc: Esc key is used to return to command mode from any other mode or command.

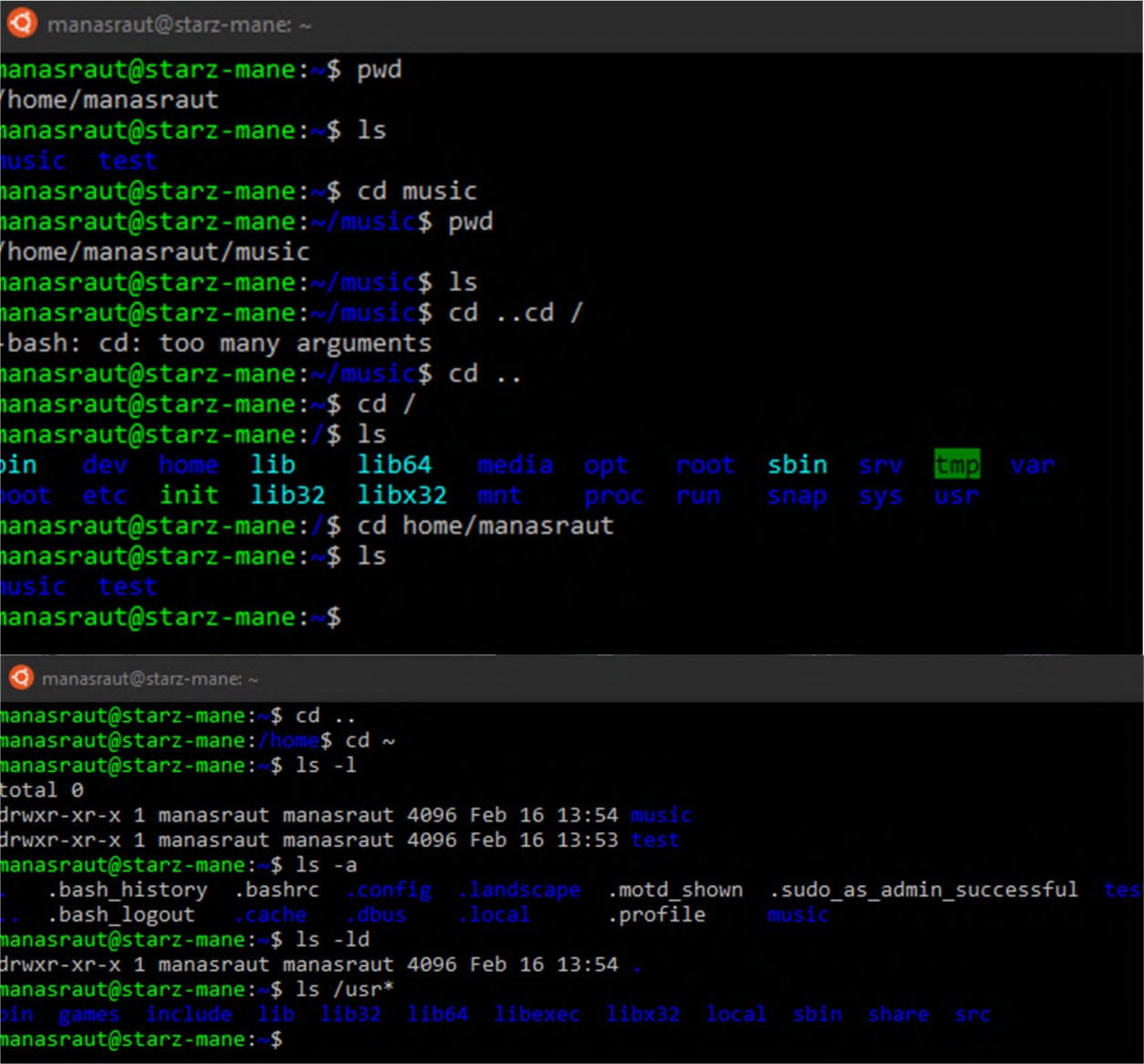
:q! : This command is used in command mode to exit vim editor without saving it.

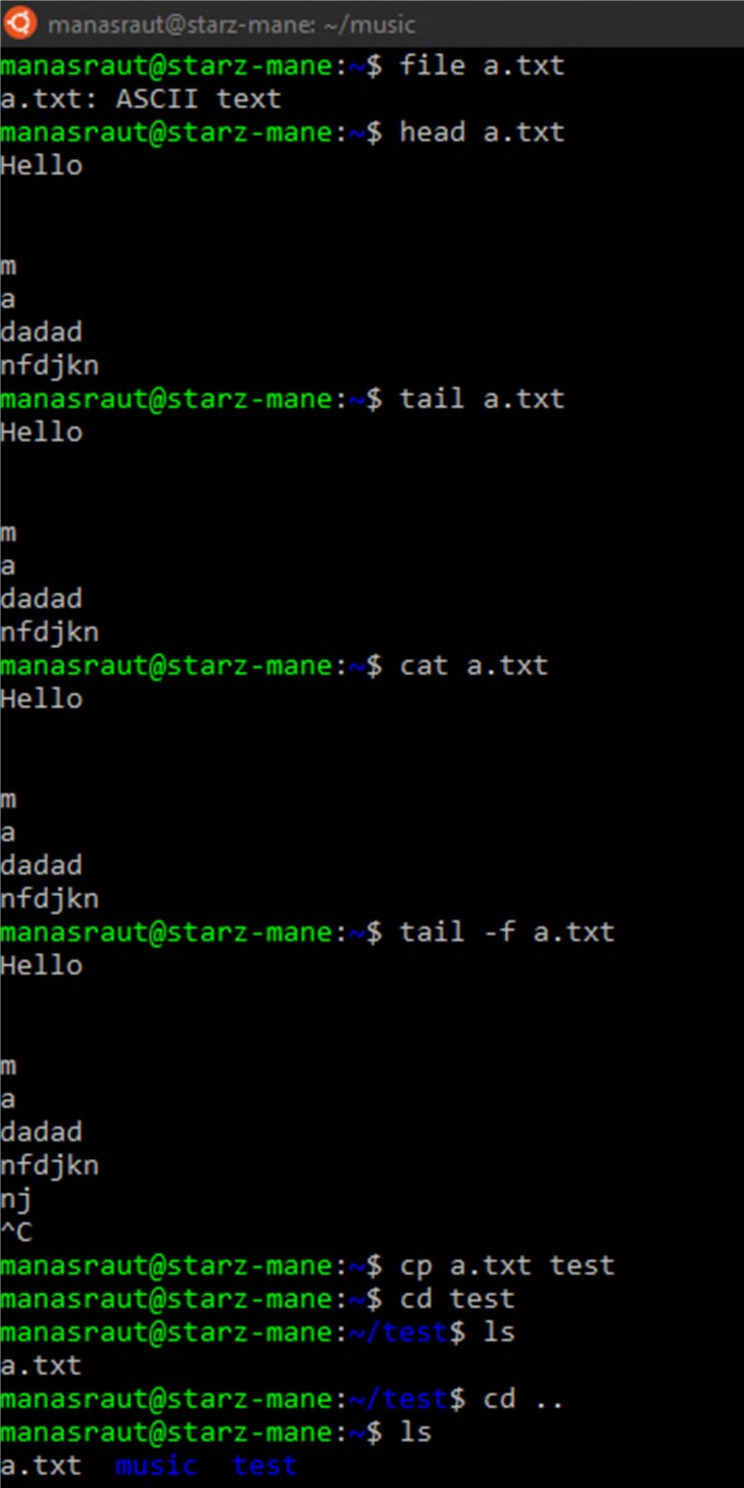
:w : This command is used to save the changes made in a file

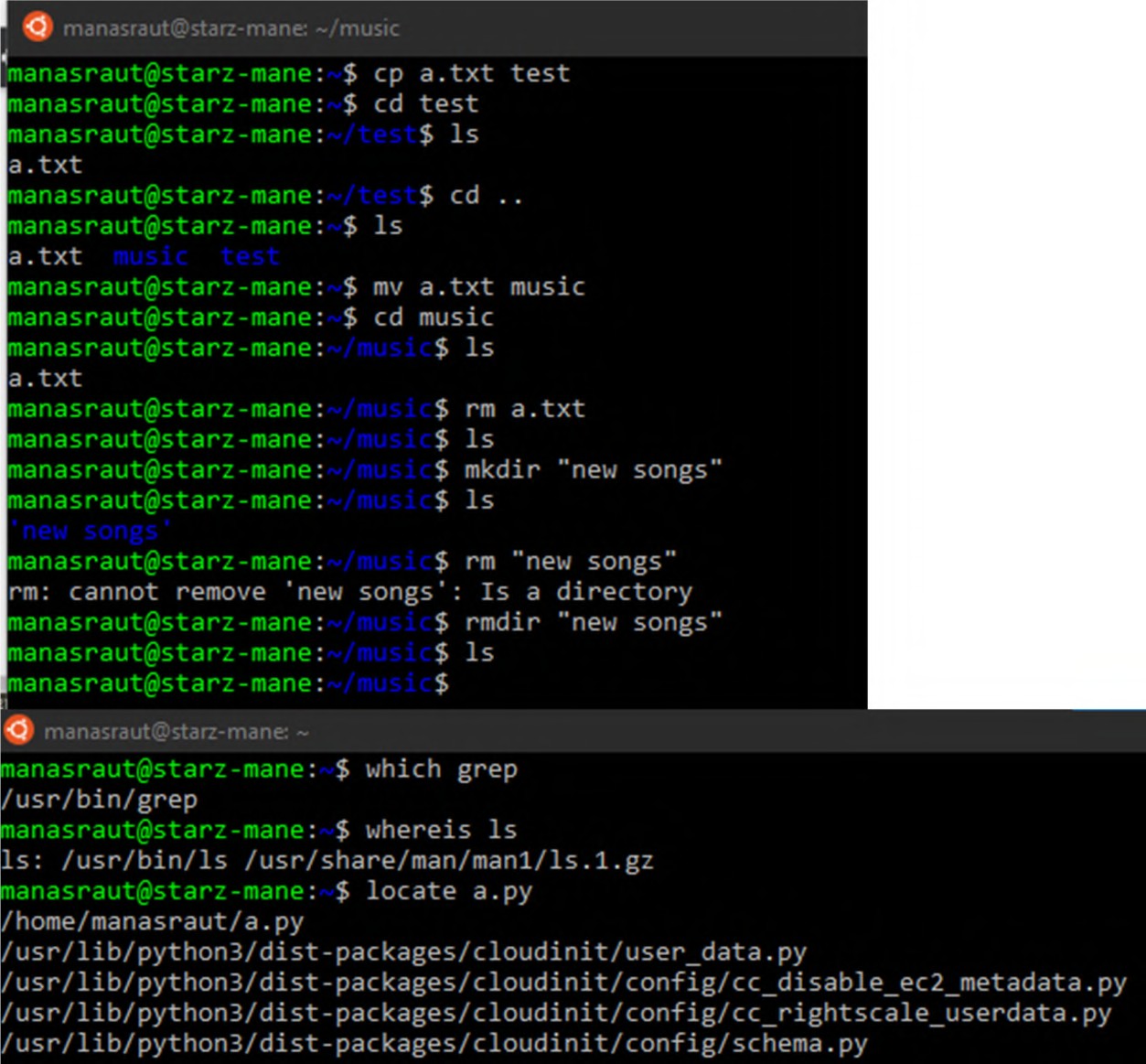
:wq : This command is used to save the changes and exit the vim editor. v : Use to select the text with arrow keys

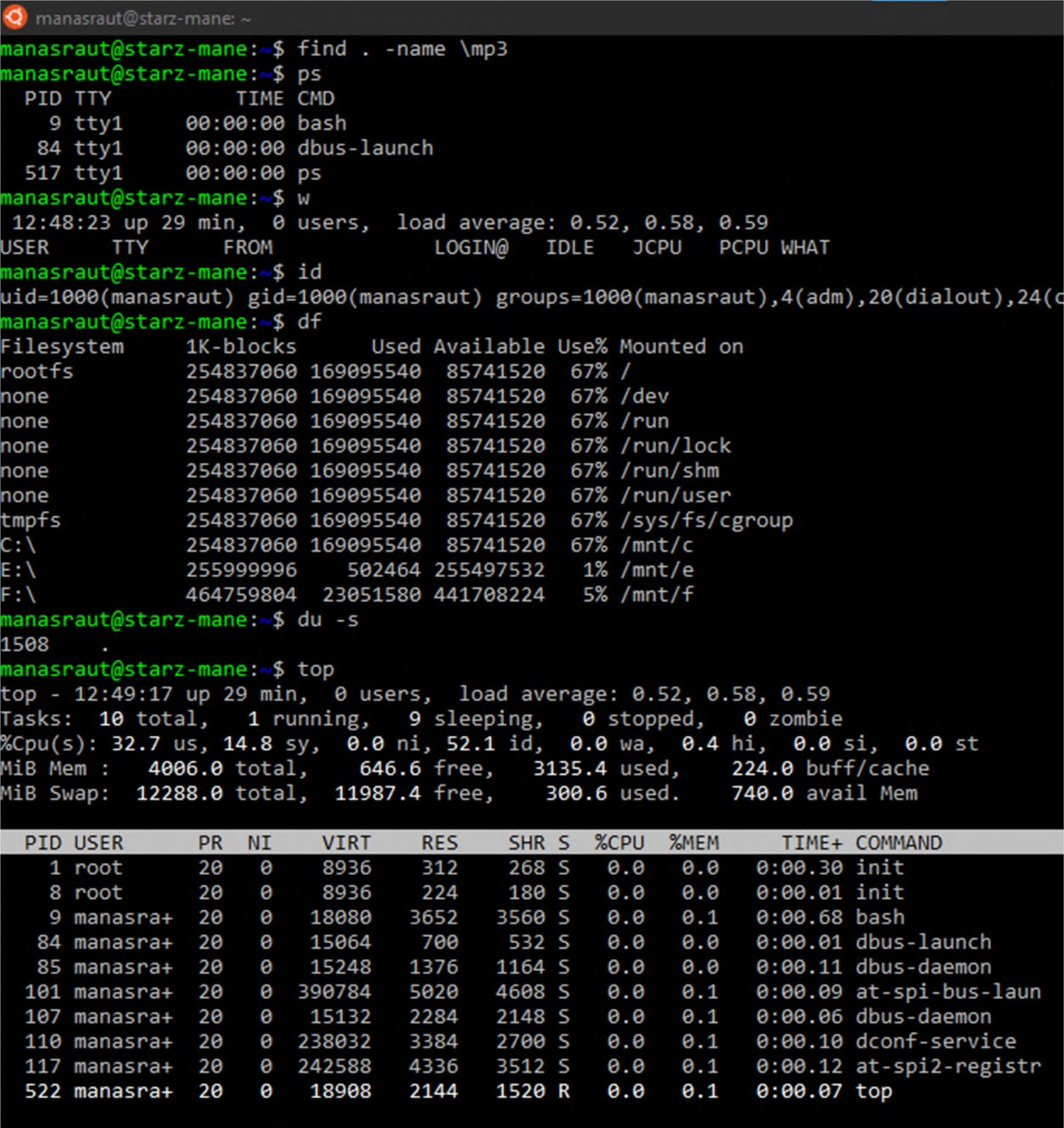
ggVG: Used to select all lines

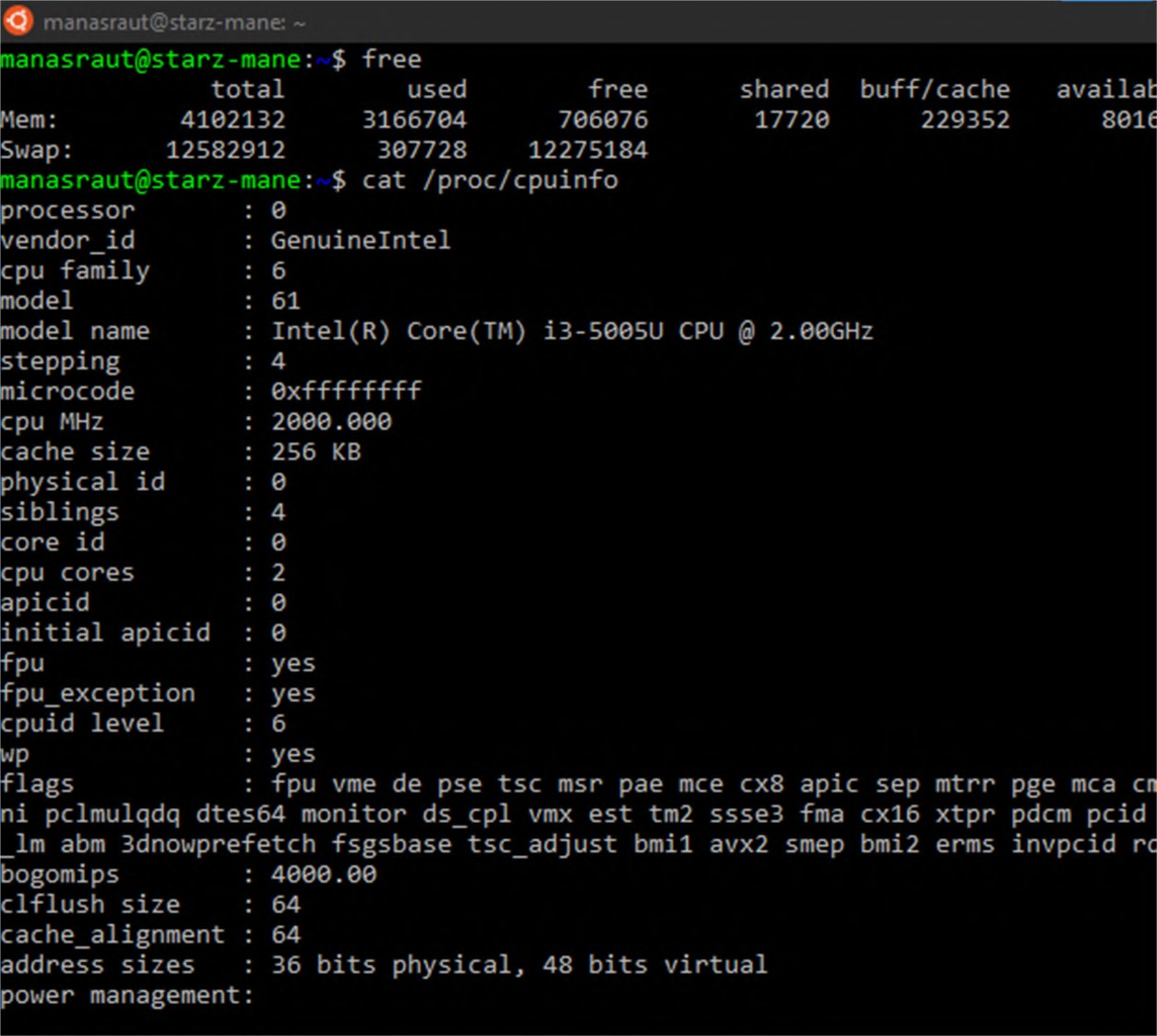
# Screenshots:

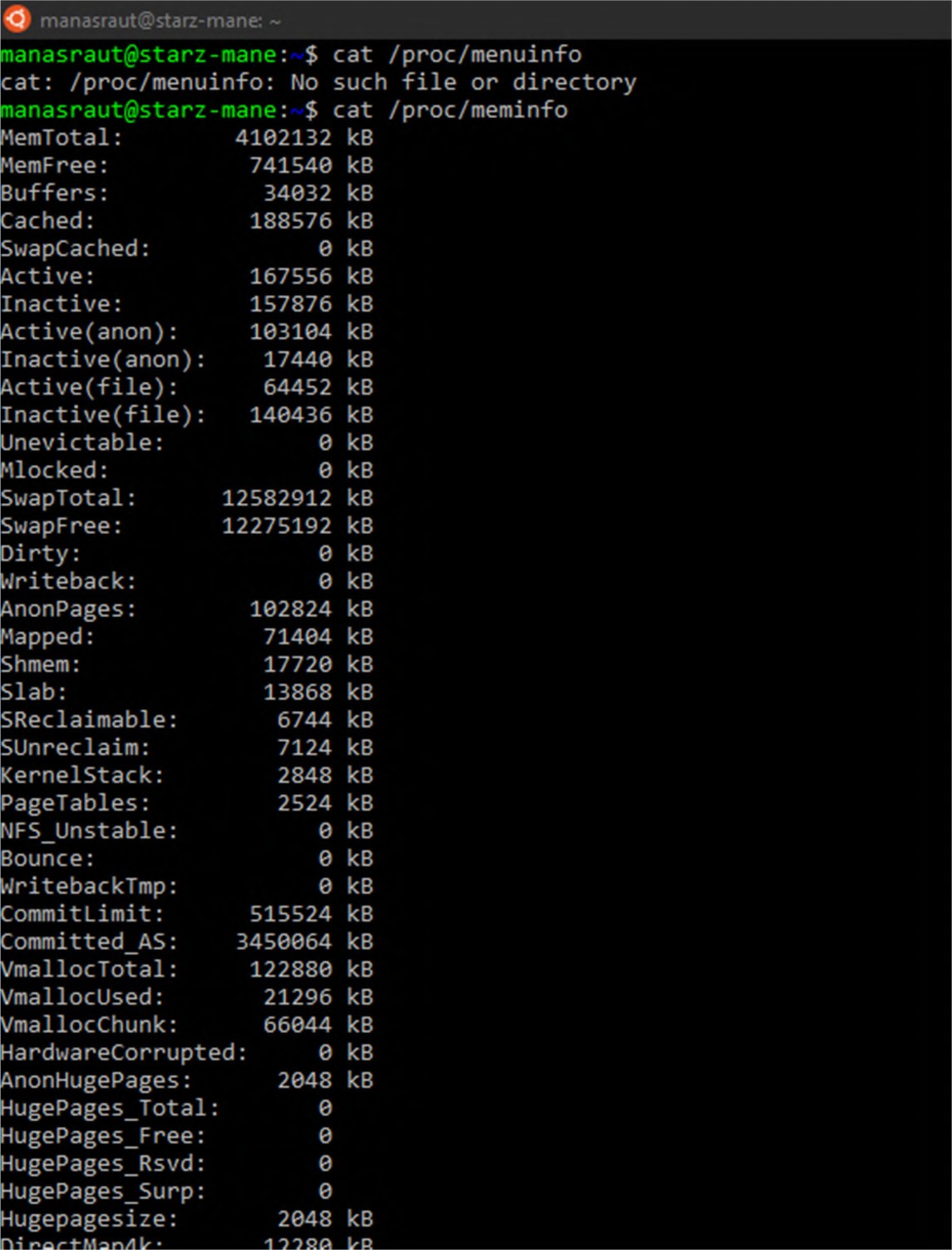
**Linus basic commands:**

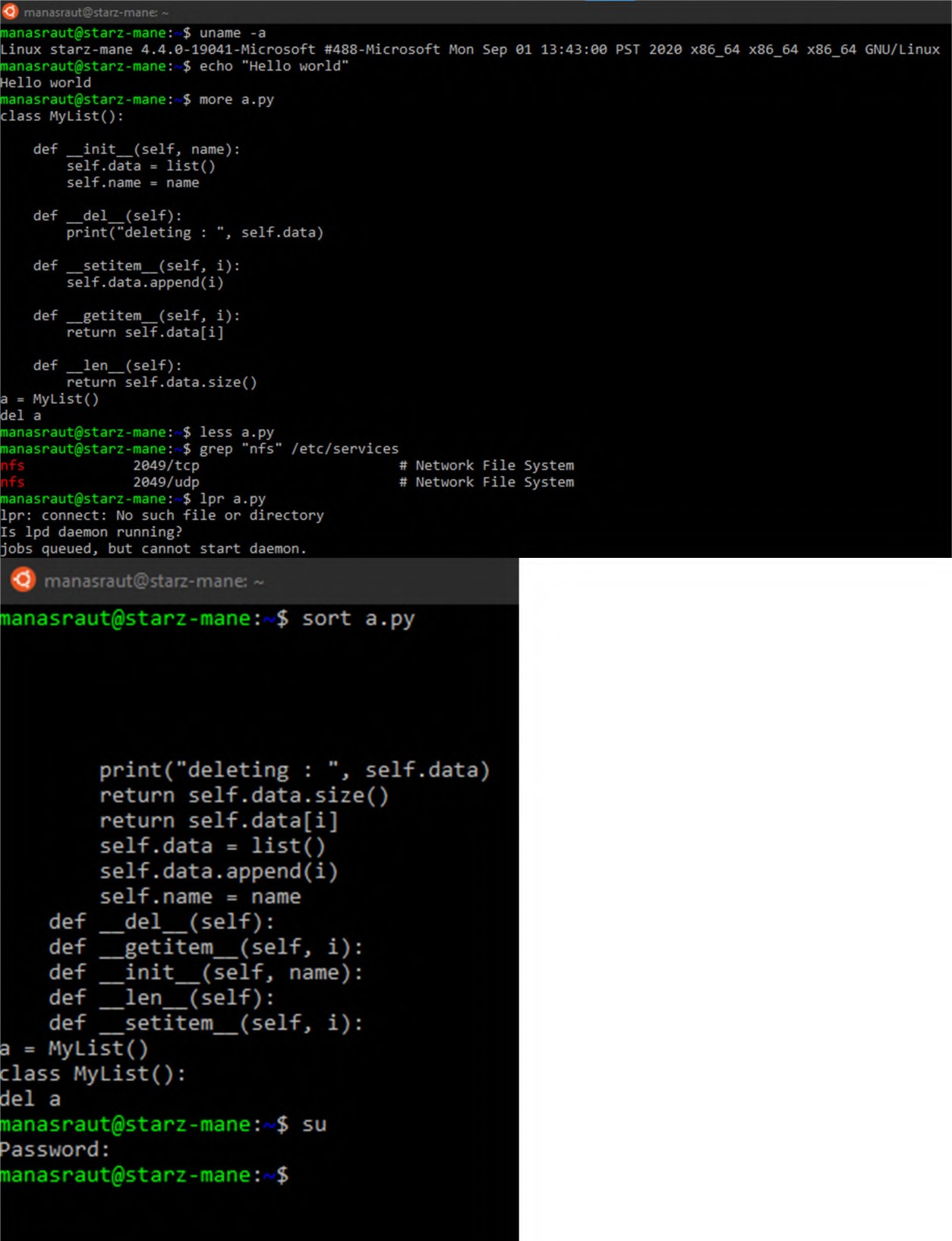




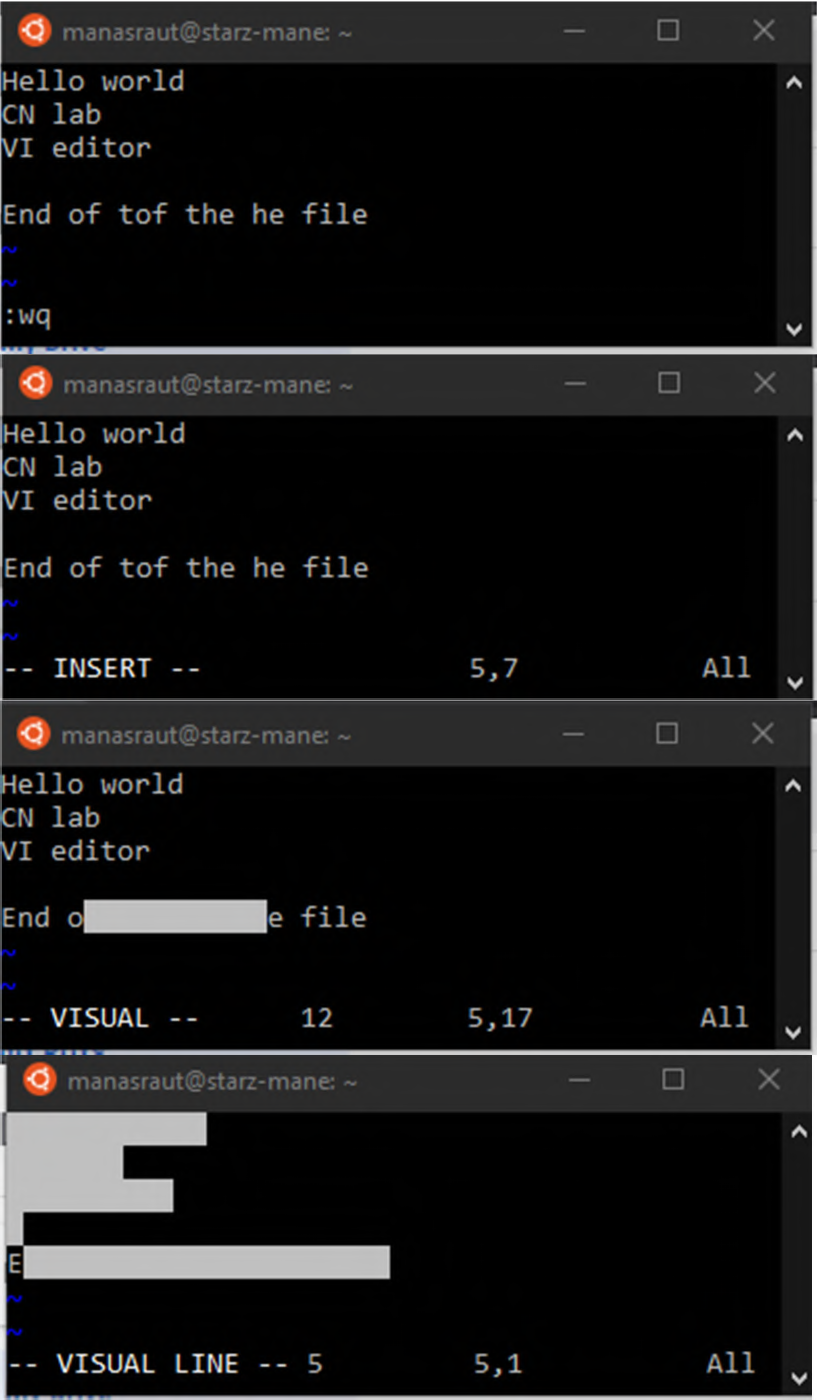








# Vim editor commands:



**Conclusion**: Thus we have performed basic commands for UNIX terminal and editor commands.

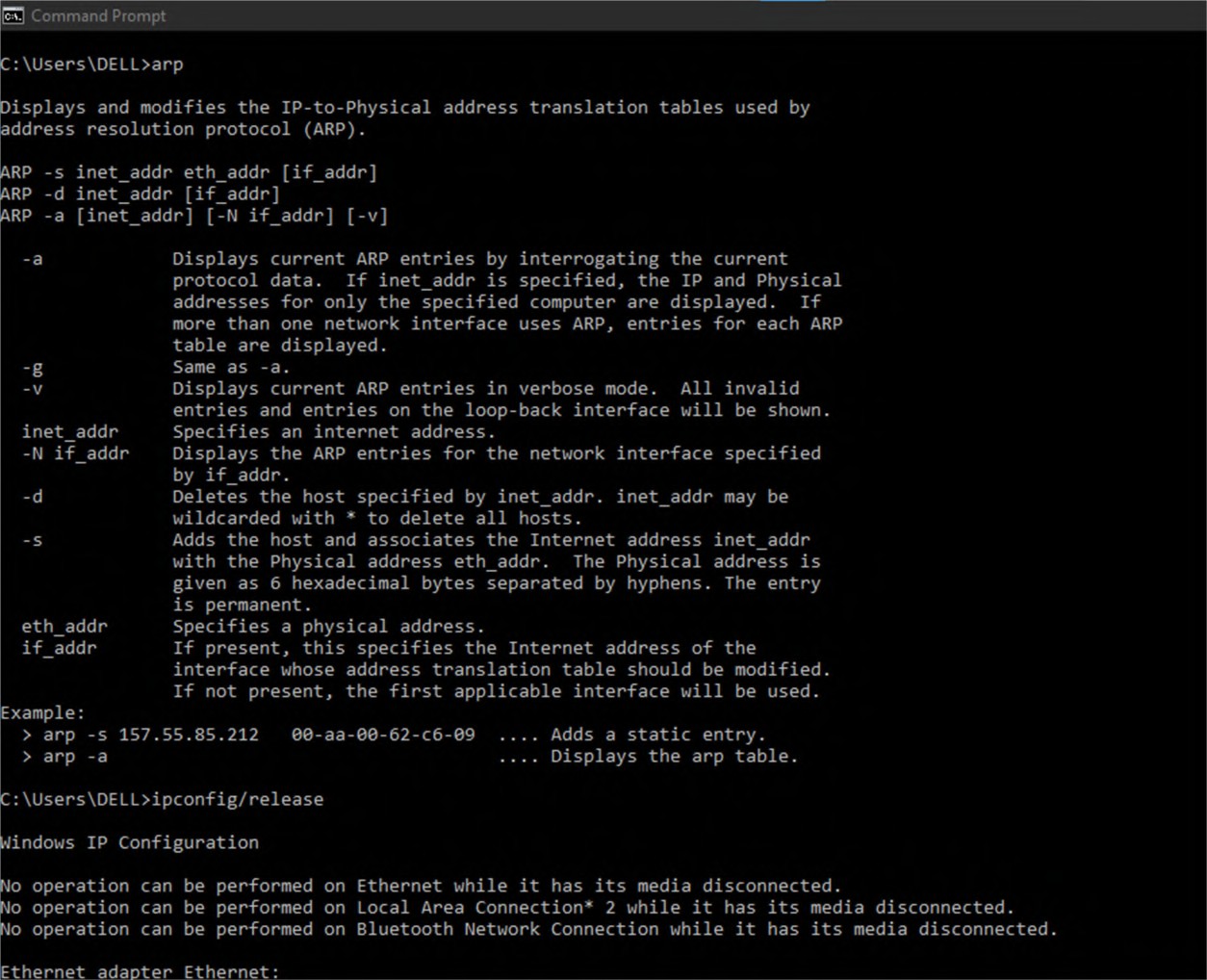
**Experiment No 2**

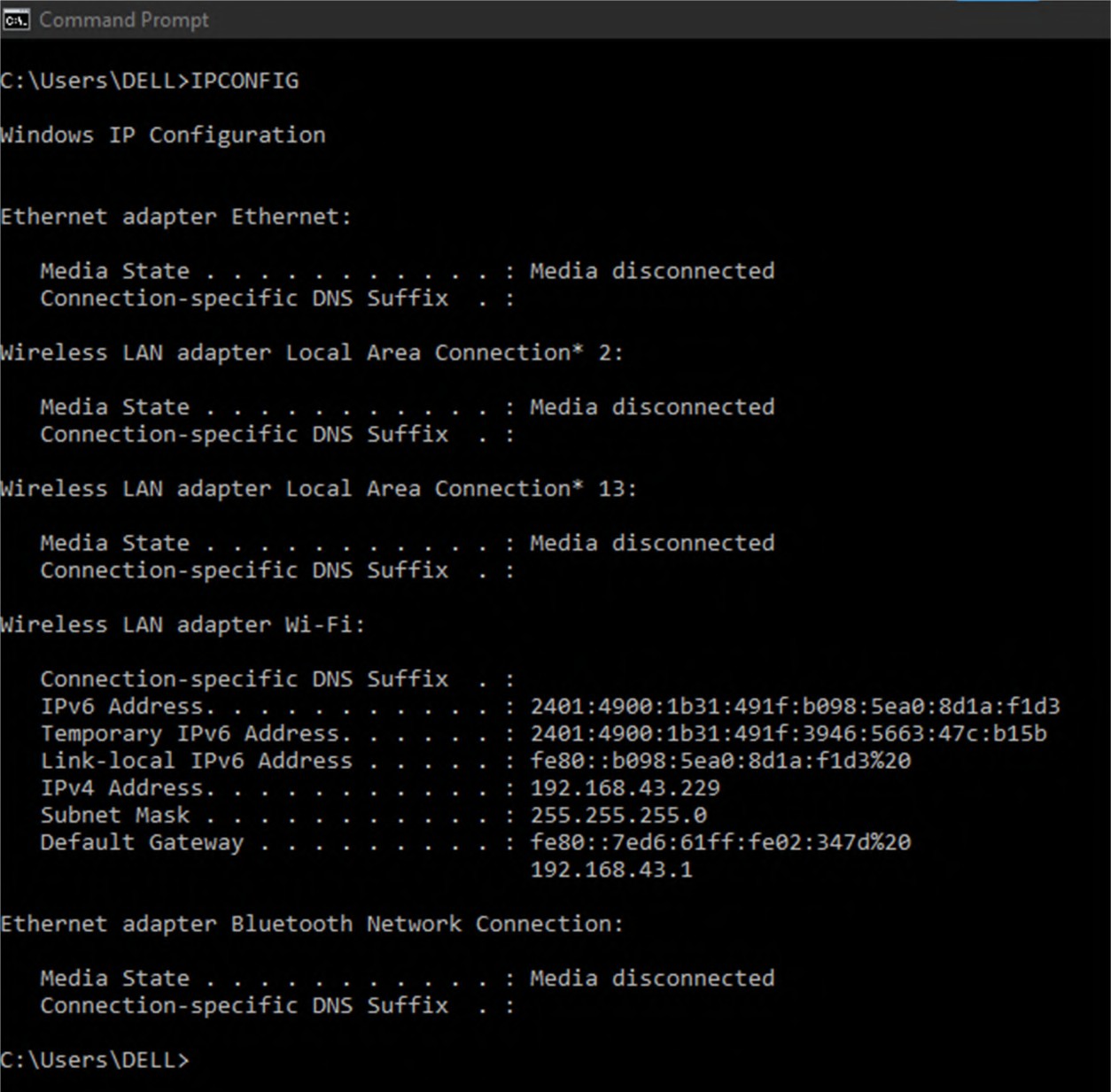
**Aim:** Understanding basic networking commands

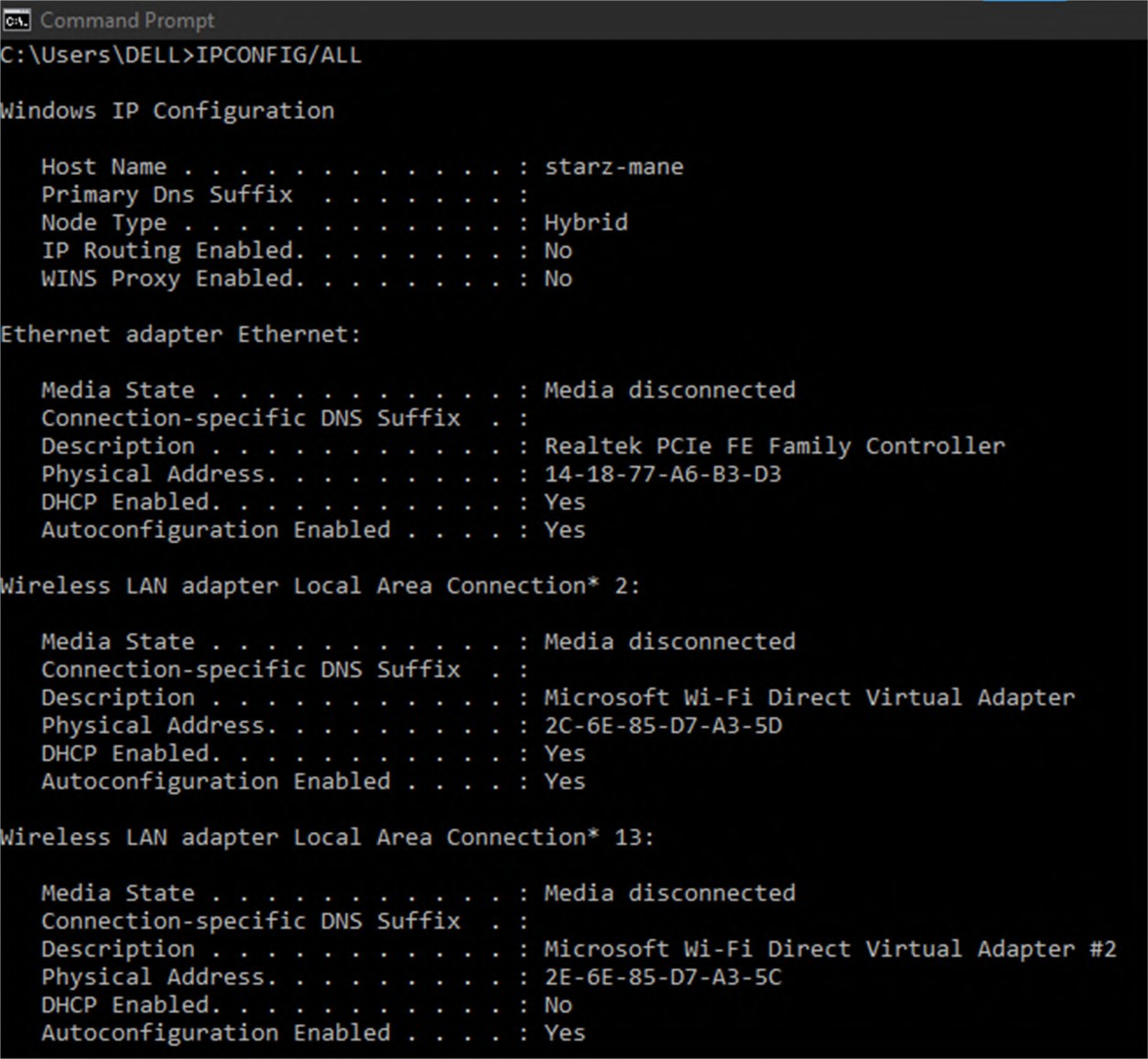
# Theory:

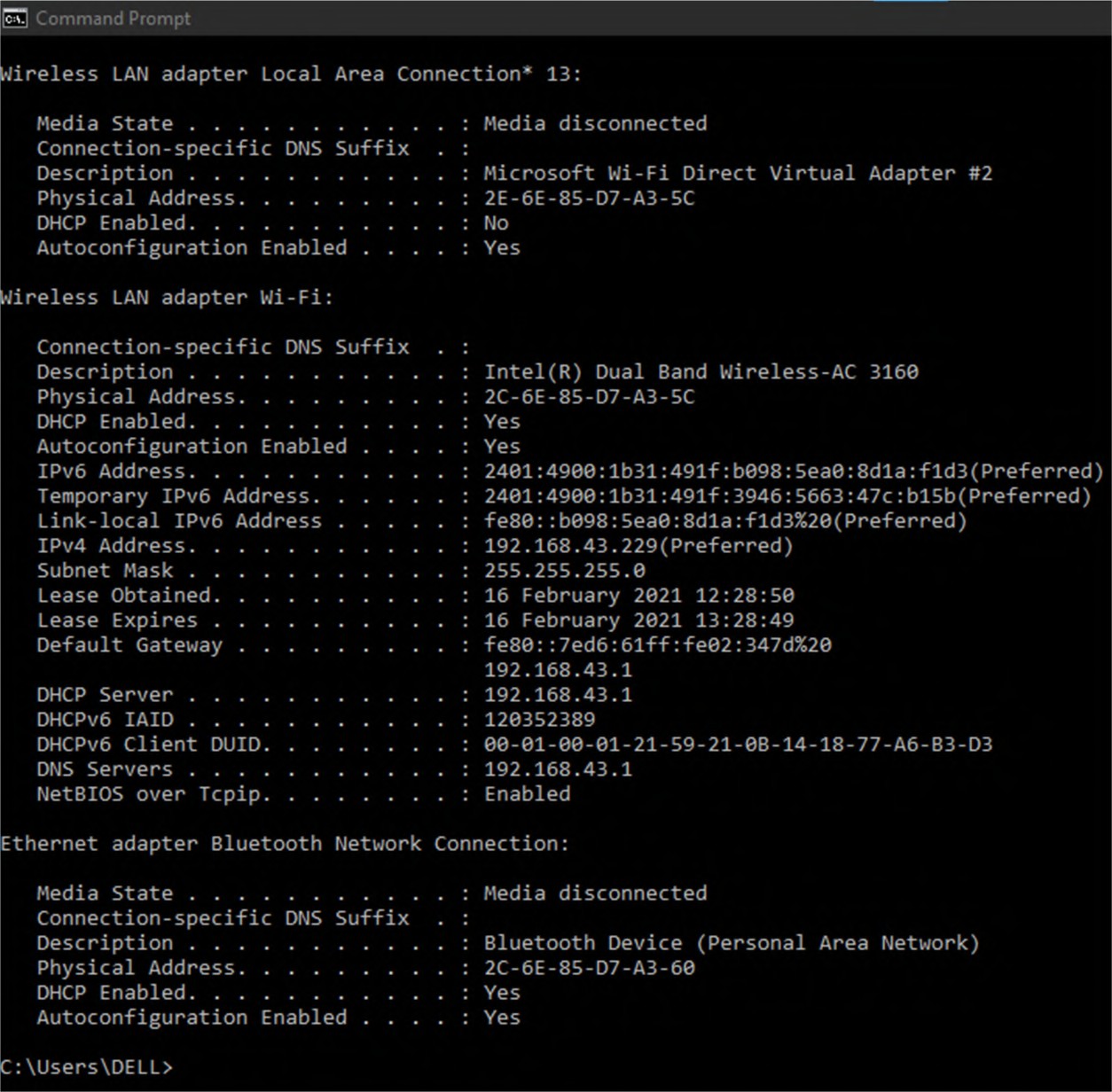
1. ipconfig: The command displays ip addresses and physical addresses of the computer and connected devices.
2. ipconfig/all : This command shows more details about the computer and its devices on the network. This details include device description, physical address, ip addresses, default gateway, etc.
3. tracert: tracert command is used to display the complete path or route taken to reach a specified server. It also displays the time taken in each step.
4. nslookup: The NSLOOKUP command shows server name and ip addresses of a domain name.
5. ping: Ping command is used to test the connection with a server by sending it some information and receiving some from the server.
6. arp: Address Resolution Protocol command translates the ip addresses to physical address of the computer.
7. rarp: This command is the reverse of arp command as it translates the physical address of the computer to the ip address.
8. ipconfig/release: This command tells the computer to give up the current ip address of active devices.

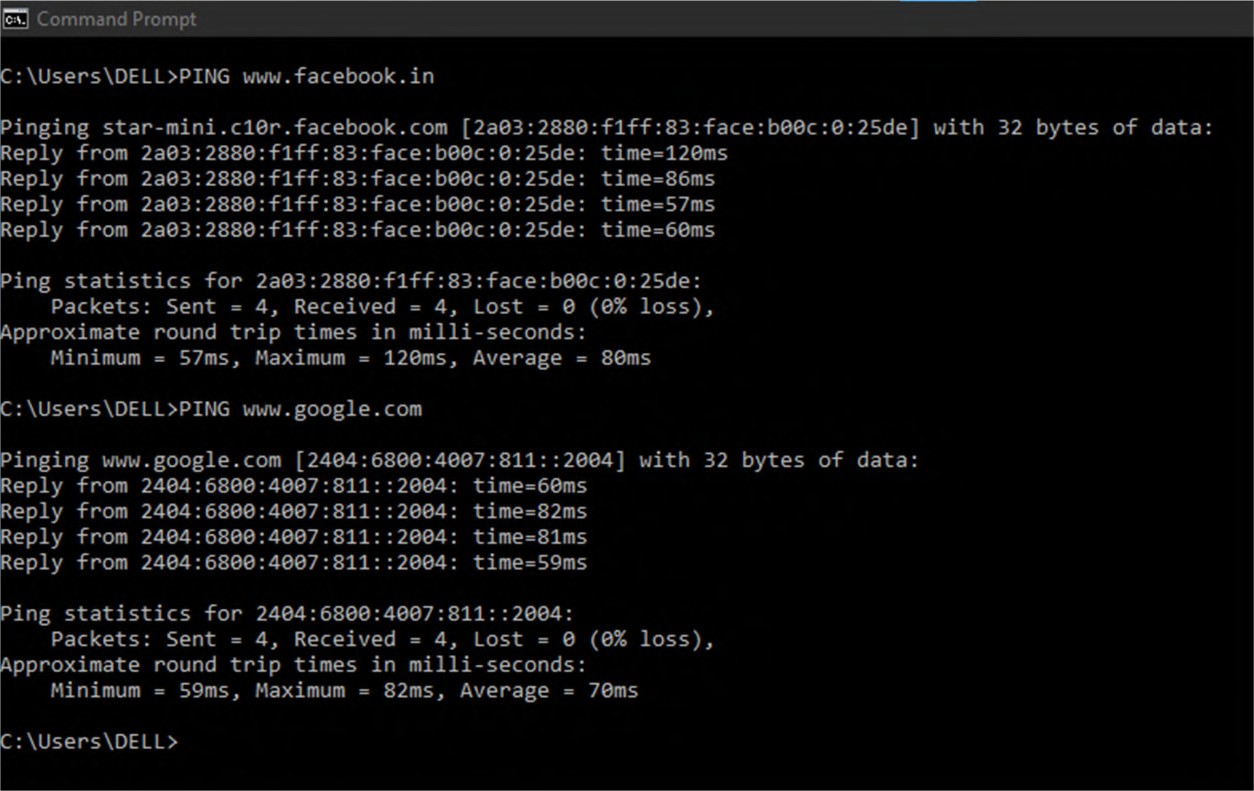
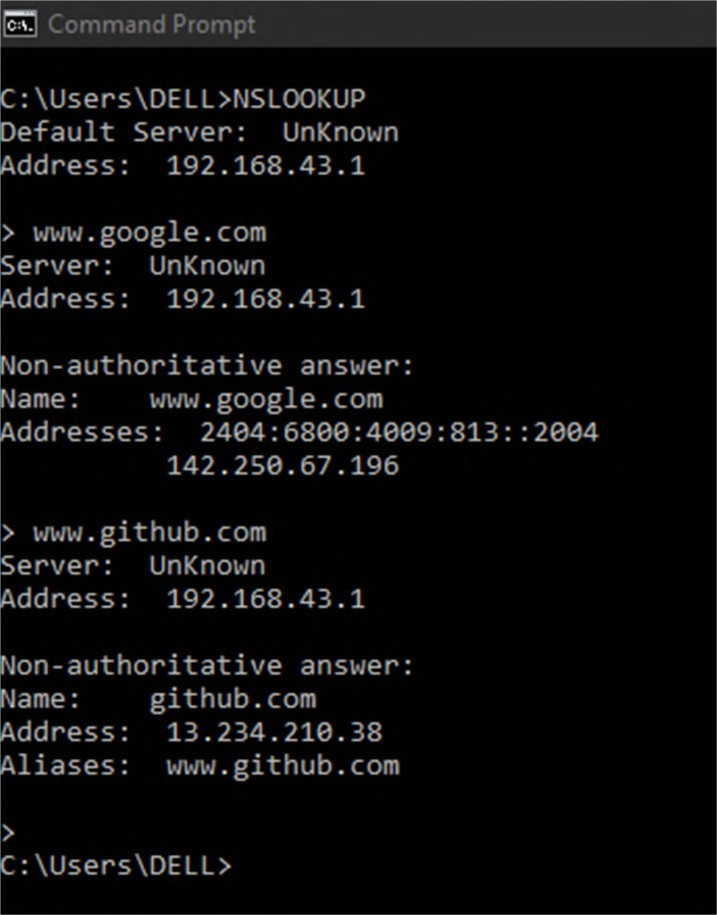
# Screenshots:













**Conclusion:** Thus we have successfully used some basic networking commands in windows command promp

**Experiment No 3**

**Aim:** Installation and configuration of NS2.

# Theory:

NS2 is a type of computer simulation software used to carry out simulation to study computer networks. It provides simulation for routing and multicast protocols for both wireless and wired networks. NS2 can be used to implement protocols such as TCP and UDP, traffic source behaviour such as FTP, TELNET etc.

To use NS2 we have to install it in a Linux Distribution and other required tools are:

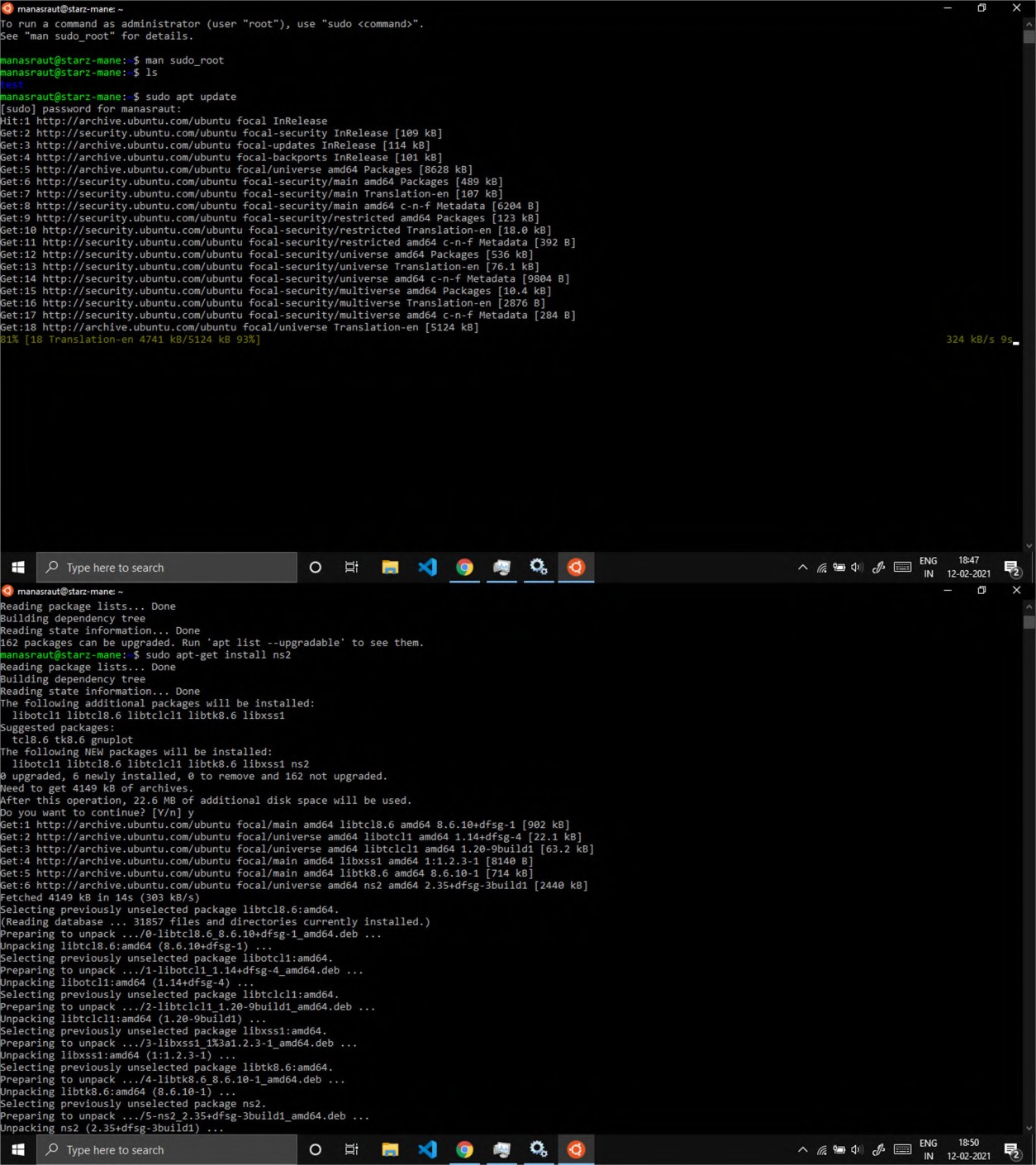
1. TCL
2. Gedit
3. Nam
4. NS2
5. Xming

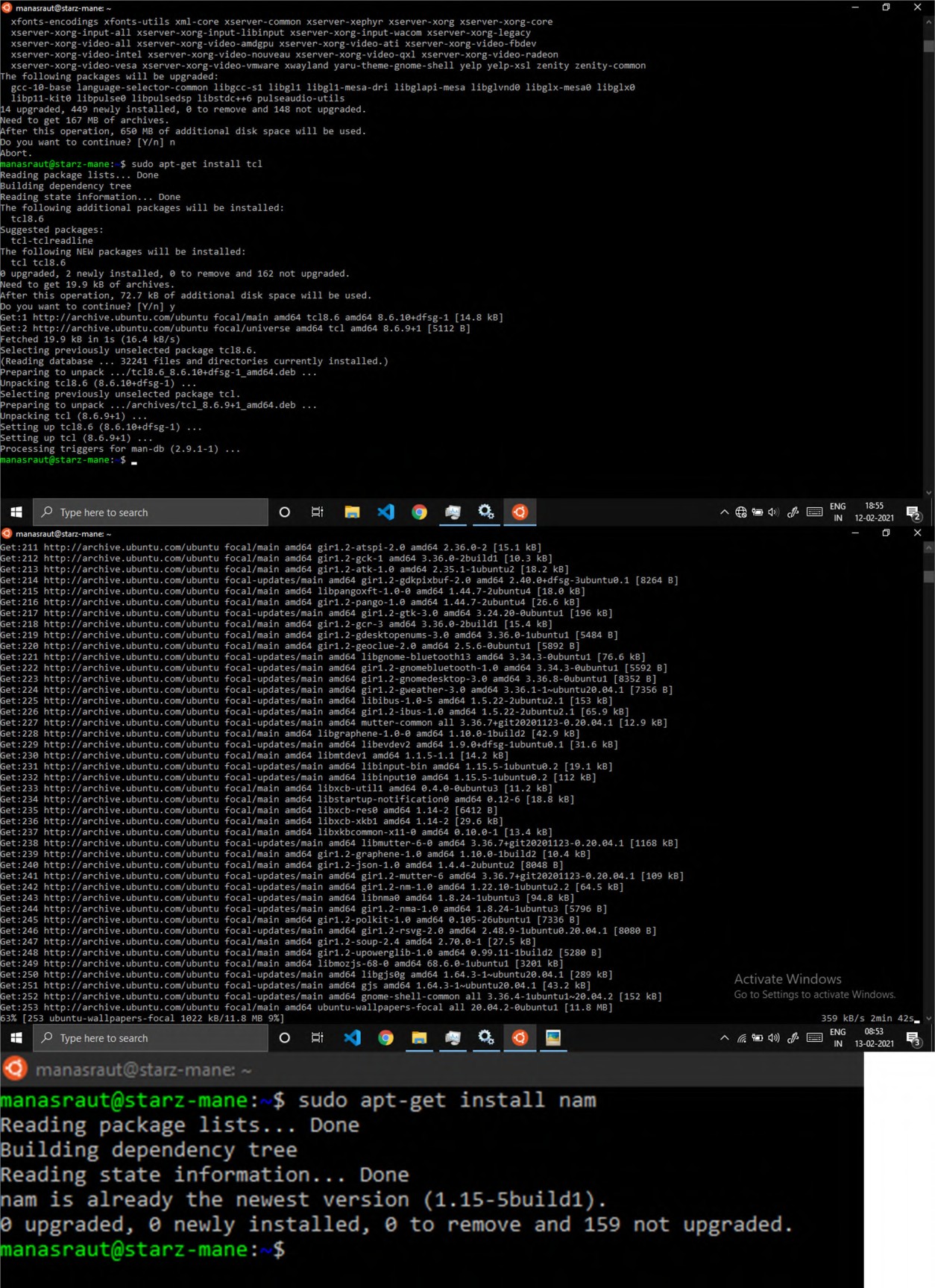
# Commands to install NS2 and other related tools are:

sudo apt update

sudo apt-get install ns2 sudo apt-get install tcl sudo apt-get install gedit sudo apt-get install nam

# Screenshots:





**Conclusion:** Thus we have successfully installed NS2 in Ubuntu operating system.

**Experiment No 4**

**Aim:** Introduction to TCL hello programming.

# Theory:

TCL is a high-level, interpreted dynamic programming language. TCl supports Object oriented paradigm. TCL programming language is used to write programs for creating network simulation in NS2 and Nam.

# Code for “Hello world” program in TCL is:

set ns [ new Simulator ]

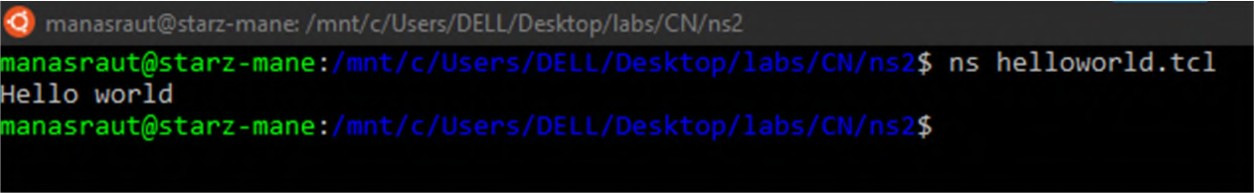
$ns at 1 "puts \"Hello world\""

$ns at 1.5 "exit"

$ns run

1. We create a new object of Simulator class named ns
2. To access values of the object $ is used
3. print “hello world” message at time 1
4. Exit at time 1.5

# Screenshots:



**Conclusion:** Thus we have successfully implemented a hello world program in TCL programming language.

**Experiment No 5**

**Aim:** A implementation of specific network topology with respect to number of nodes an physical network layer configuration.

# Theory:

Nam is a TCL/TK based animation tool that is used to visualize the ns simulation and real world packet trace data. In a wireless network, nodes communicate using the communication using the communication model that consists of UDP agent, Null agent and CBR traffic agent. The sender node is attached to the UDP agent while the receiver node is attached to the Null agent. Transport agent UDP and application CBR are connected.

proc: it is used to create a TCl procedure. A procedure are bunch of statements which can be executed when the procedure is called by its name. The proc command is also used to override other existing procedures with same name.

duplex-link: It creates two simplex links of specified bandwidth and delay, and connects the two specified nodes.

# Code:

set ns [ new Simulator ] set nf [ open s1.nam w ]

$ns namtrace-all $nf set nf1 [ open s1.tr w ]

$ns trace-all $nf1

proc finish { } {

global ns nf nf1

$ns flush-trace close $nf

close $nf1

exec nam s1.nam & exit 0

}

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

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

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

set cbr0 [ new Application/Traffic/CBR ]

$cbr0 set packet\_size\_ 500

$cbr0 set interval\_ 0.005

$cbr0 attach-agent $udp0 set null0 [ new Agent/Null ]

$ns attach-agent $n1 $null0

$ns connect $udp0 $null0

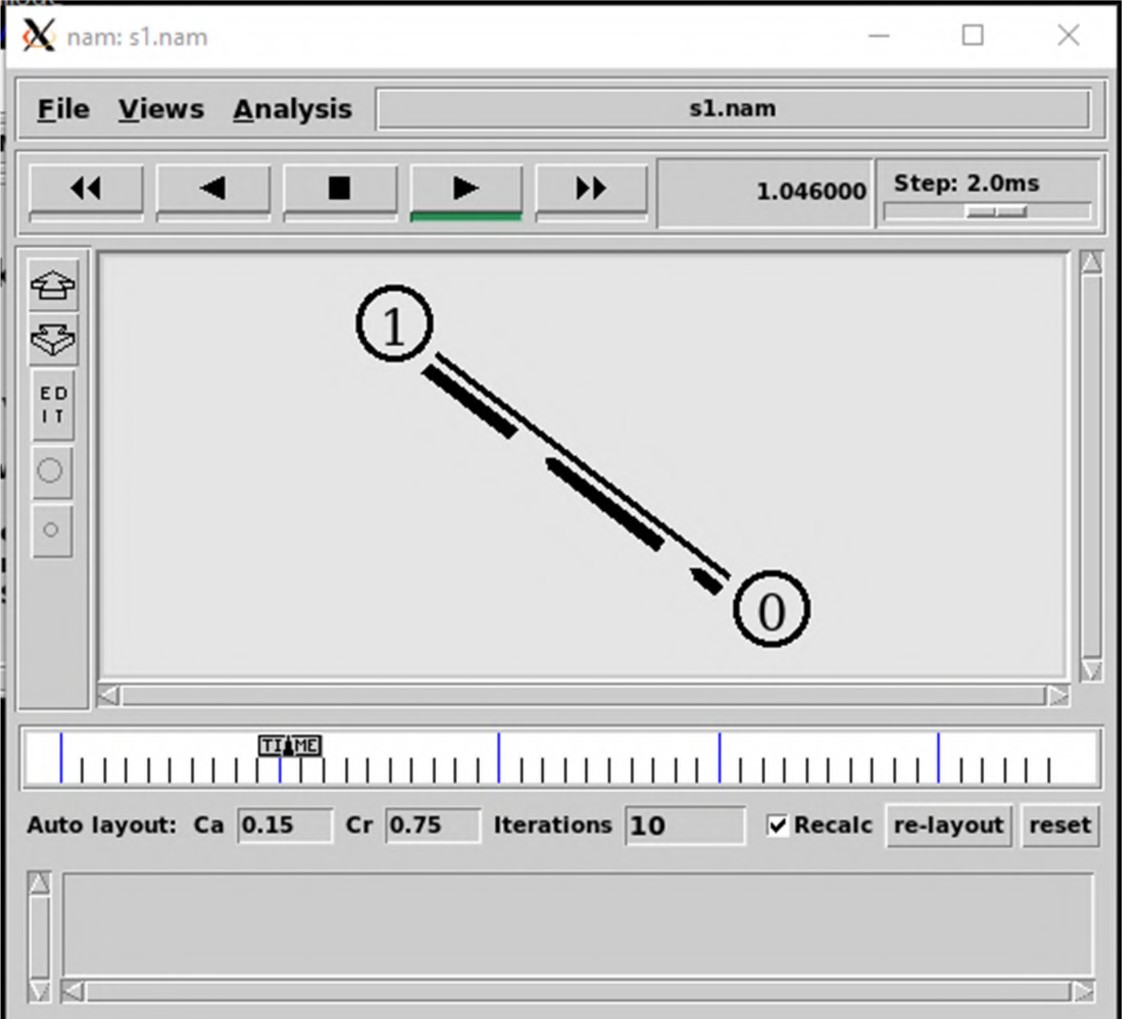
$ns at 0.5 "$cbr0 start"

$ns at 4.5 "$cbr0 stop"

$ns at 5.0 "finish"

$ns run

# Screenshots:



**Conclusion:** Thus we have successfully implemented a network topology in NS2 and Nam.

**Experiment No 6**

**Aim:** Simulation of network with routing protocols and traffic consideration (TCP/ UDP) using NAM

# Theory:

Nam is a TCL/TK based animation tool that is used to visualize the ns simulation and real world packet trace data. In a wireless network, nodes communicate using the communication using the communication model that consists of UDP agent, Null agent and CBR traffic agent. The sender node is attached to the UDP agent while the receiver node is attached to the Null agent. Transport agent UDP and application CBR are connected.

proc: it is used to create a TCl procedure. A procedure are bunch of statements which can be executed when the procedure is called by its name. The proc command is also used to override other existing procedures with same name.

duplex-link: It creates two simplex links of specified bandwidth and delay, and connects the two specified nodes.

In this practical we are setting the orientation of the nodes.

# Code:

set ns [ new Simulator ] set nf [ open out.nam w ]

$ns namtrace-all $nf set nf1 [ open s6.tr w ]

$ns trace-all $nf1

# set colors

$ns color 1 Blue

$ns color 2 Red

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 queue-limit $n2 $n3 10

# position the nodes

$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 duplex-link-op $n2 $n3 queuePos 0.5

set tcp [ new Agent/TCP ]

$tcp set class\_ 2

$ns attach-agent $n0 $tcp

set sink [ new Agent/TCPSink ]

$ns attach-agent $n3 $sink

$ns connect $tcp $sink

$tcp set fid\_ 1

set ftp [ new Application/FTP ]

$ftp attach-agent $tcp

$ftp set type\_ FTP

set udp [ new Agent/UDP ]

$ns attach-agent $n1 $udp set null [ new Agent/Null ]

$ns attach-agent $n3 $null

$ns connect $udp $null

$udp set fid\_ 2

set cbr [ new Application/Traffic/CBR ]

$cbr attach-agent $udp

$cbr set type\_ CBR

$cbr set packet\_size\_ 1000

$cbr set rate\_ 1mb

$cbr set random\_ false

$ns at 0.1 "$cbr start"

$ns at 1.0 "$ftp start"

$ns at 4.0 "$ftp stop"

$ns at 4.5 "$cbr stop"

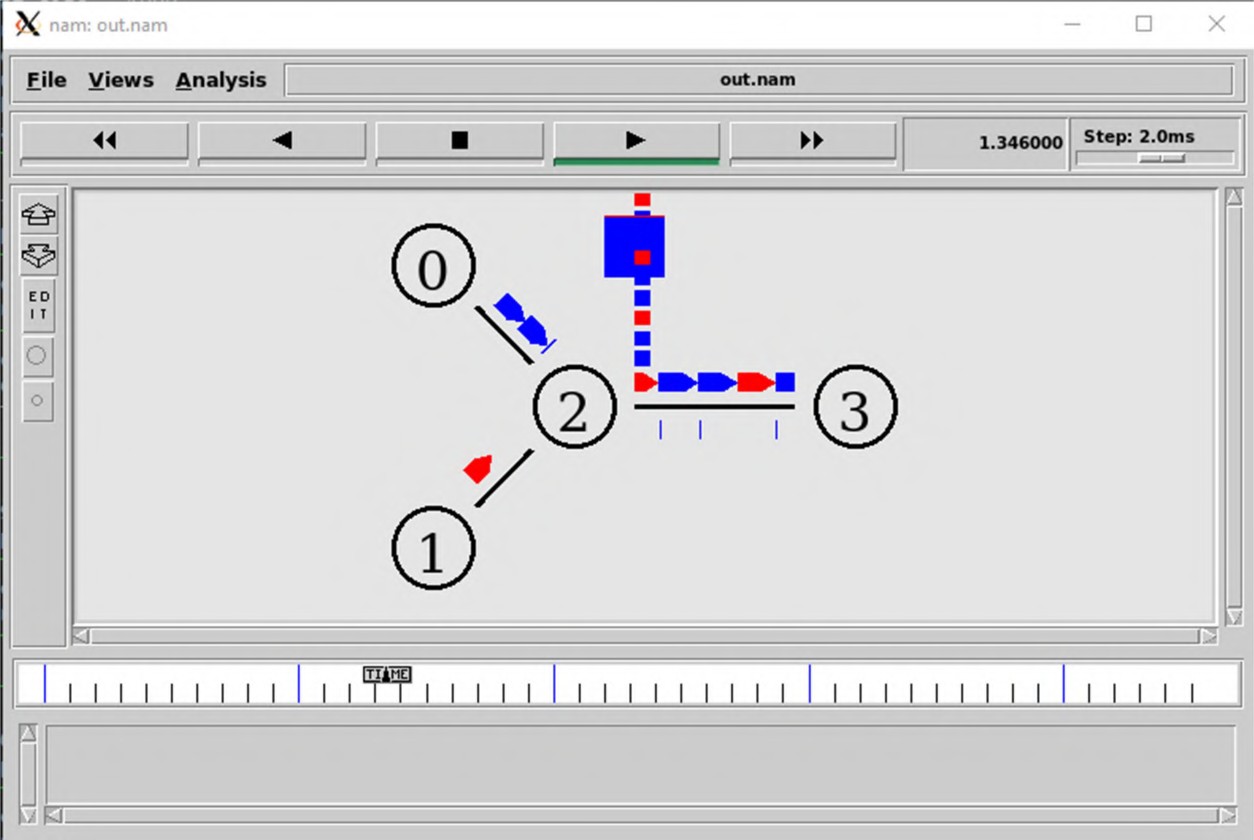
$ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink"

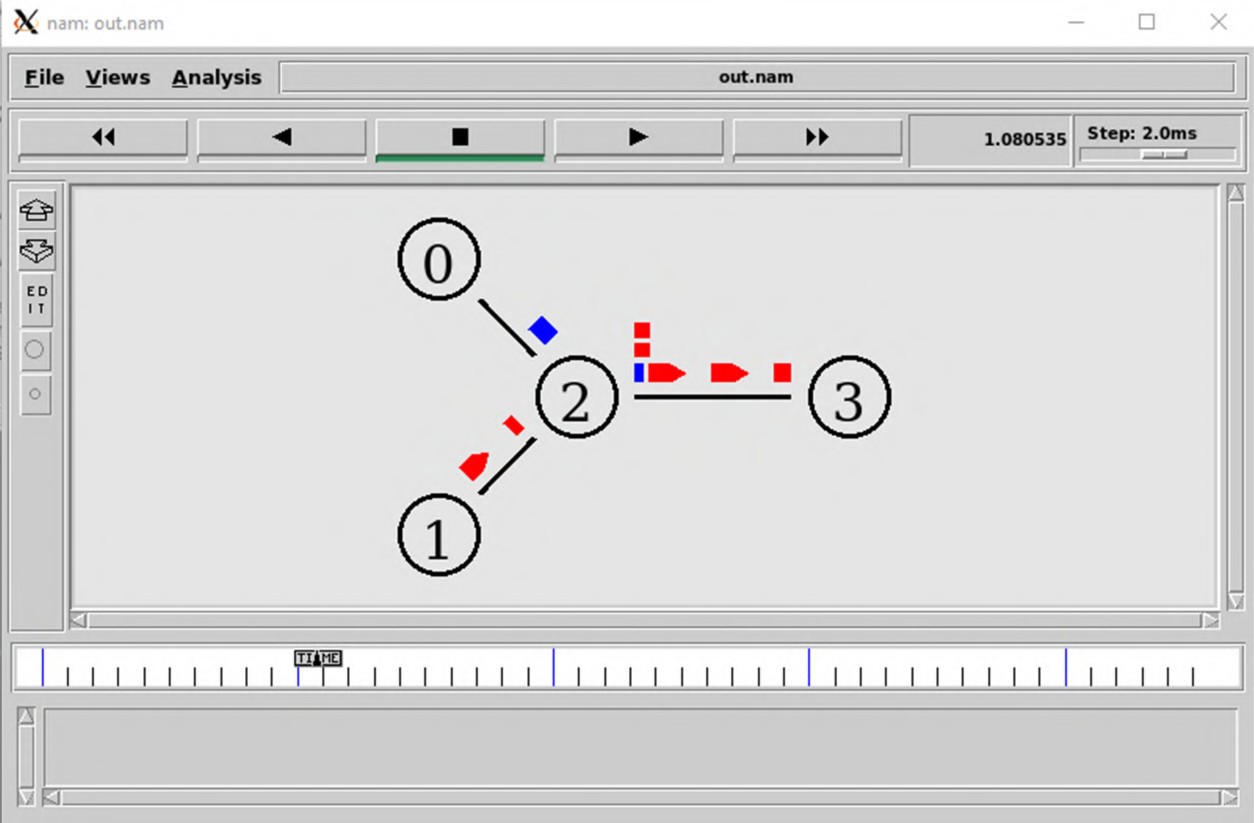
$ns at 5.0 "finish"

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

$ns run

# Screenshots:





**Conclusion:** Thus we have successfully implemented a network simulation of TCP/UDP in NS2 and Nam

**Experiment No 7**

**Aim:** Simulation of network with specific routing protocols (distance vector/ link state)

# Theory:

Distance Vector routing: It is a dynamic algorithm in which each router computes distance between itself and each possible destination. The router shares information about the whole network to its neighbours and performs updates from its neighbours. The routing algorithm uses Bellman Ford algorithm for making routing tables.

Link state routing: The link state routing is a dynamic algorithm in which each router shares knowledge with every other router in the network. Information sharing takes place only when there is change in information. It uses Dijkastra’s algorithm for making routing tables.

# Code:

set ns [ new Simulator ]

$ns rtproto LS

set nf [ open linkstate.nam w ]

$ns namtrace-all $nf

set f0 [ open linkstate.tr w ]

$ns trace-all $f0

proc finish { } {

global ns nf f0

$ns flush-trace close $nf

close $f0

exec nam linkstate.nam & exit 0

}

for { set i 0 } { $i<7 } { incr i } { set n($i) [ $ns node ]

}

for { set i 0 } { $i<7 } { incr i } {

$ns duplex-link $n($i) $n([ expr ($i+1)%7 ]) 1Mb 10ms DropTail

}

set udp0 [ new Agent/UDP ]

$ns attach-agent $n(0) $udp0

set cbr0 [ new Application/Traffic/CBR ]

$cbr0 set packetSize\_ 500

$cbr0 set interval\_ 0.005

$cbr0 attach-agent $udp0

set null0 [ new Agent/Null ]

$ns attach-agent $n(3) $null0

$ns connect $udp0 $null0

$ns at 0.5 "$cbr0 start"

$ns rtmodel-at 1.0 down $n(1) $n(2)

$ns rtmodel-at 2.0 up $n(1) $n(2)

$ns at 4.5 "$cbr0 stop"

$ns at 5.0 "finish"

$ns run

# Code 2:

set ns [ new Simulator ]

$ns rtproto LS

set nf [ open s7\_2.nam w ]

$ns namtrace-all $nf set f0 [ open s7\_2.tr w ]

$ns trace-all $f0

proc finish { } {

global ns nf f0

$ns flush-trace close $nf

close $f0

exec nam s7\_2.nam & exit 0

}

for { set i 0 } { $i<8 } { incr i } { set n($i) [ $ns node ]

}

for { set i 0 } { $i<8 } { incr i } {

$ns duplex-link $n($i) $n([ expr ($i+1)%8 ]) 1Mb 10ms DropTail

}

set udp0 [ new Agent/UDP ]

$ns attach-agent $n(0) $udp0

set cbr0 [ new Application/Traffic/CBR ]

$cbr0 set packetSize\_ 500

$cbr0 set interval\_ 0.005

$cbr0 attach-agent $udp0

set null0 [ new Agent/Null ]

$ns attach-agent $n(3) $null0

$ns connect $udp0 $null0

$ns at 0.0 "$n(0) label Source"

$ns at 0.0 "$n(3) label Destination"

$ns at 0.5 "$cbr0 start"

$ns rtmodel-at 1.0 down $n(2) $n(3)

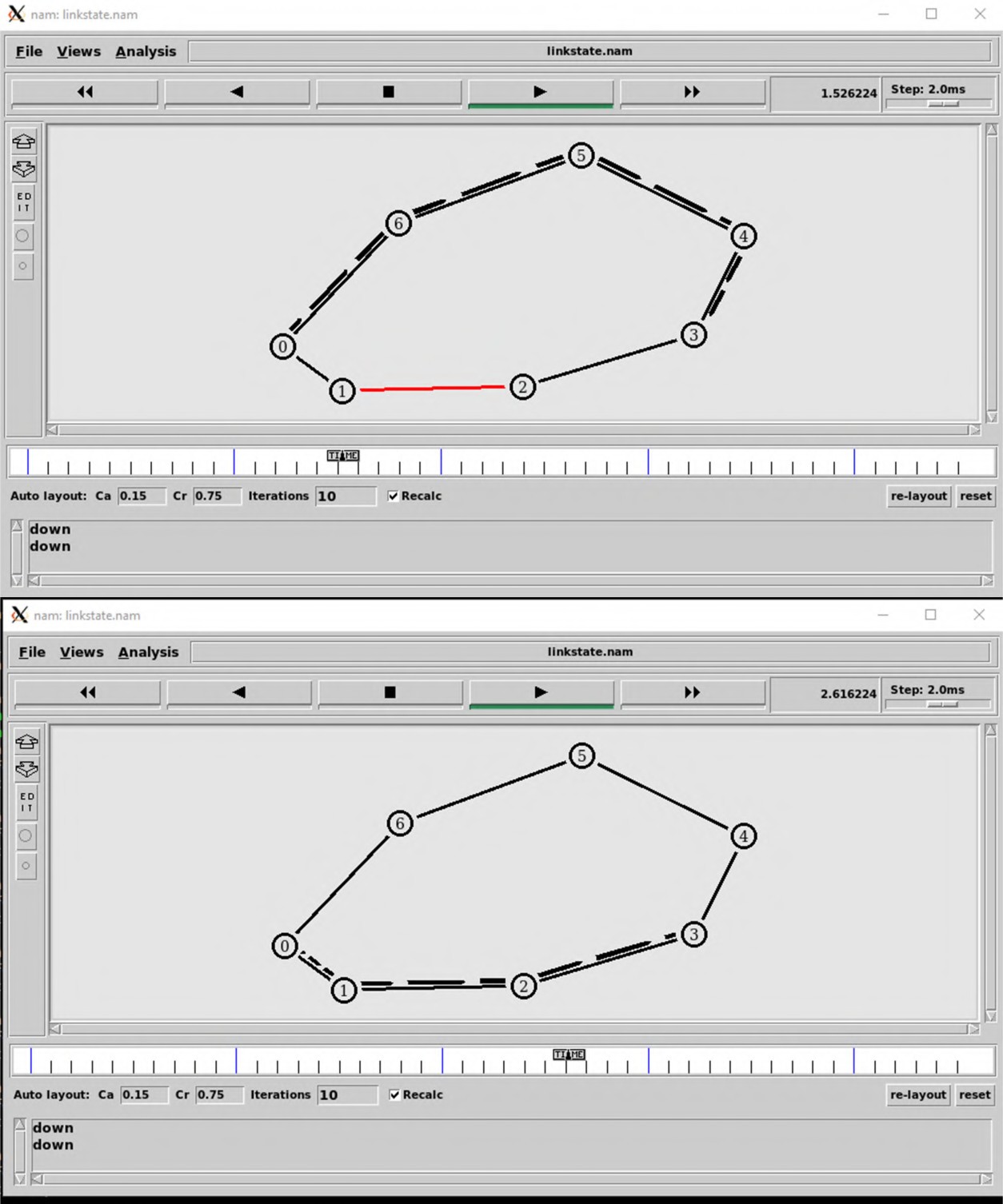
$ns rtmodel-at 2.0 up $n(2) $n(3)

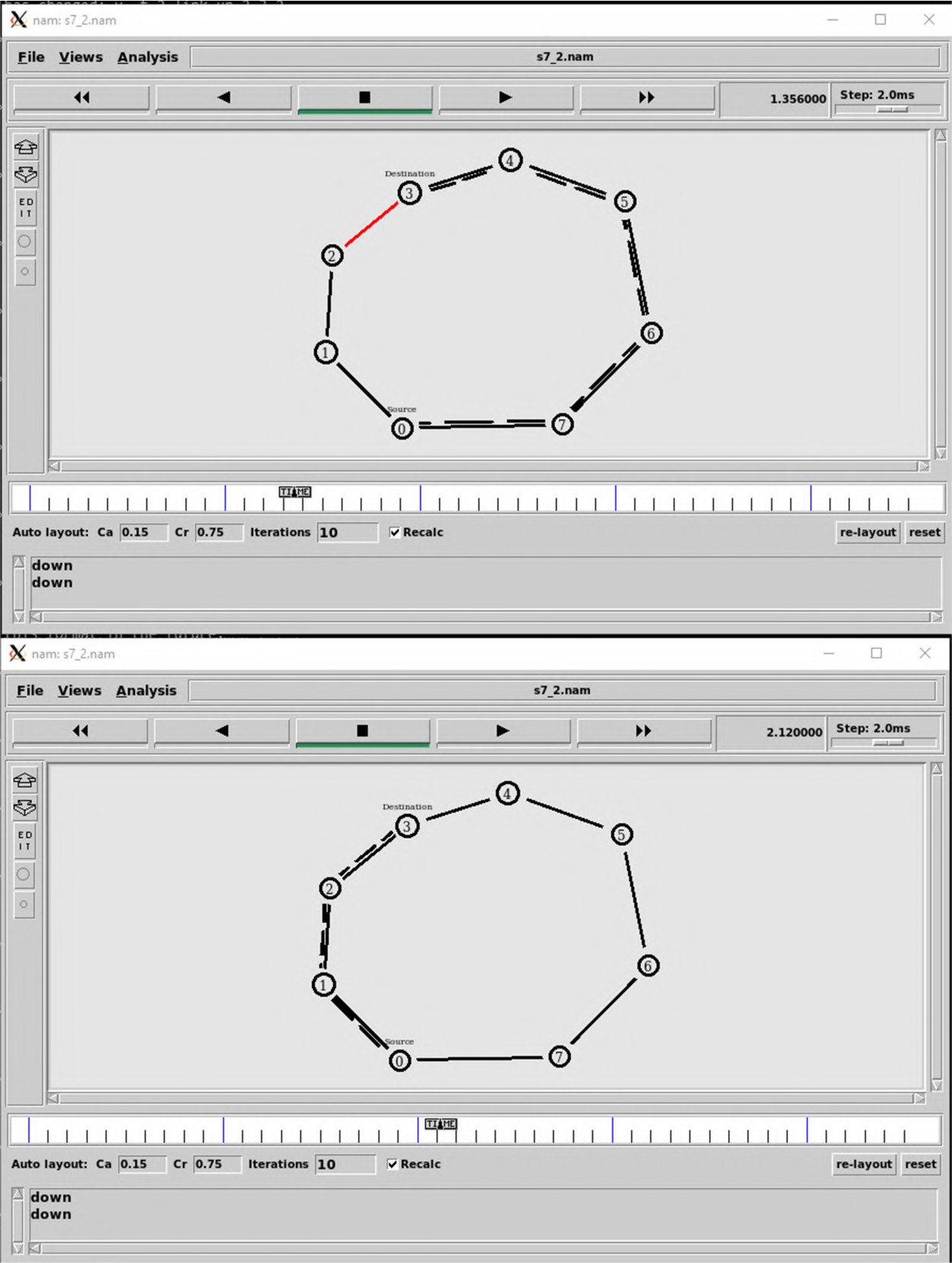
$ns at 4.5 "$cbr0 stop"

$ns at 5.0 "finish"

$ns run

# Screenshots: Screenshots for code 1:



**Screenshots for code 2:**

**Conclusion:** Thus we have successfully specific routing protocols (distance vector/ link state) in network animator (NAM).

**Experiment No 8**

**Aim:** To write a ns2 program for implementing unicast routing protocol

# Theory:

Unicast routing protocol is the term used for transmission from a single sender to single receiver. It is a point to point communication between sender and receiver. There are various unicast protocols like TCP, HTTP, etc.

Three major protocols for unicast routing are:

1. Distance Vector routing
2. Link state routing
3. Path vector routing.

# Code:

set ns [ new Simulator ]

$ns color 1 Blue

$ns color 2 Red

set nf [ open s8.nam w ]

$ns namtrace-all $nf set nfl [ open s8.tr w ]

$ns trace-all $nfl

$ns rtproto DV

proc finish { } {

global nf ns nfl

$ns flush-trace close $nf

close $nfl

exec nam s8.nam & exit 0

}

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

$ns duplex-link $n0 $n1 0.3Mb 10ms DropTail

$ns duplex-link $n1 $n2 0.3Mb 10ms DropTail

$ns duplex-link $n2 $n3 0.3Mb 10ms DropTail

$ns duplex-link $n1 $n4 0.3Mb 10ms DropTail

$ns duplex-link $n3 $n5 0.3Mb 10ms DropTail

$ns duplex-link $n4 $n5 0.3Mb 10ms DropTail

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

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

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

$ns duplex-link-op $n1 $n4 orient up-left

$ns duplex-link-op $n3 $n5 orient left-up

$ns duplex-link-op $n4 $n5 orient right-up

set tcp [ new Agent/TCP/Newreno ]

$ns attach-agent $n0 $tcp

set sink [ new Agent/TCPSink/DelAck ]

$ns attach-agent $n5 $sink

$ns connect $tcp $sink

$tcp set fid\_ 1

set ftp [ new Application/FTP ]

$ftp attach-agent $tcp

$ftp set type\_ FTP

$ns at 0.1 "$ftp start"

$ns rtmodel-at 1.0 down $n1 $n4

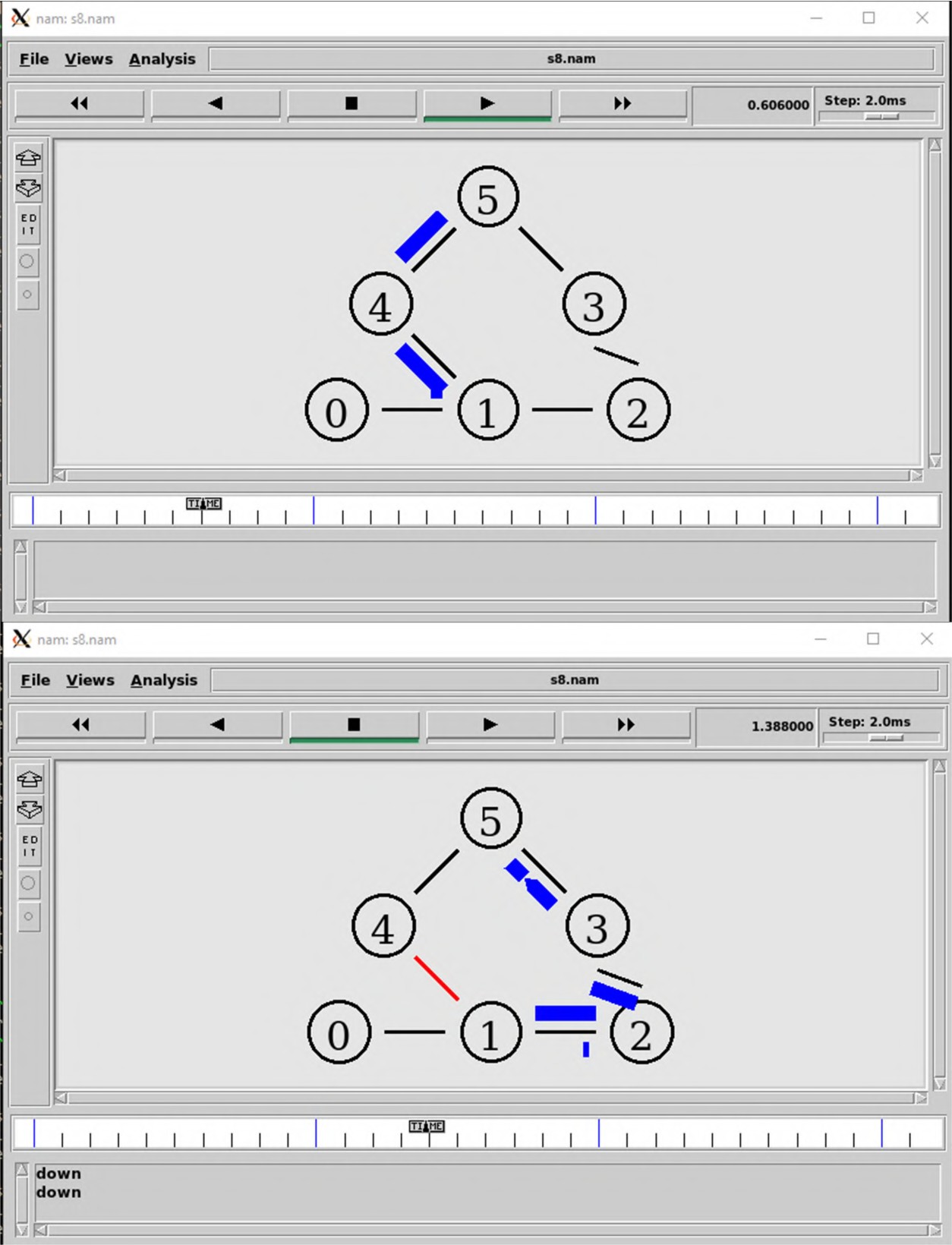
$ns rtmodel-at 2.0 up $n1 $n4

$ns at 3.0 "$ftp stop"

$ns at 3.1 "finish"

$ns run

# Screenshots:



**Conclusion:** Thus we have successfully program for implementing unicast routing protocol in network animator (NAM).

**Experiment No 9**

**Aim:** Write and execute TCL scripts to create Bus topology, ring topology and star topology.

# Theory:

**Bus topology:** A bus topology is a topology for a local area network (LAN) in which all the nodes are connected to a single cable called as bus. If the Bus cable stops working then the entire system stops working

**Ring topology:** Ring topology is also used in Local Area Network (LAN). In this all nodes are connected to two of its adjacent nodes. If any of the nodes fails then the network still works.

**Start topology:** Star topology is used in Local Area Network (LAN). In this topology all the nodes are connected to a single centre node. This main node is called as the hub or switch. In this topology if any of the node fails then other nodes still works.

# Code:

**Bus Topology:**

set ns [ new Simulator ]

set nf [ open st9\_1.nam w ]

$ns namtrace-all $nf

set nfl [ open st9\_1.tr w ]

$ns trace-all $nfl

proc finish { } {

global ns nf nfl

$ns flush-trace close $nf

close $nfl

exec nam st9\_1.nam & exit 0

}

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

set lan0 [ $ns newLan "$n0 $n1 $n2 $n3 $n4" 0.5Mb 40ms LL Queue/DropTail MAC/Csma/Cd Channel ]

set tcp0 [ new Agent/TCP ]

$tcp0 set class\_ 1

$ns attach-agent $n1 $tcp0

set sink0 [ new Agent/TCPSink ]

$ns attach-agent $n3 $sink0

$ns connect $tcp0 $sink0

set cbr0 [ new Application/Traffic/CBR ]

$cbr0 set packetSize\_ 500

$cbr0 set interval\_ 0.01

$cbr0 attach-agent $tcp0

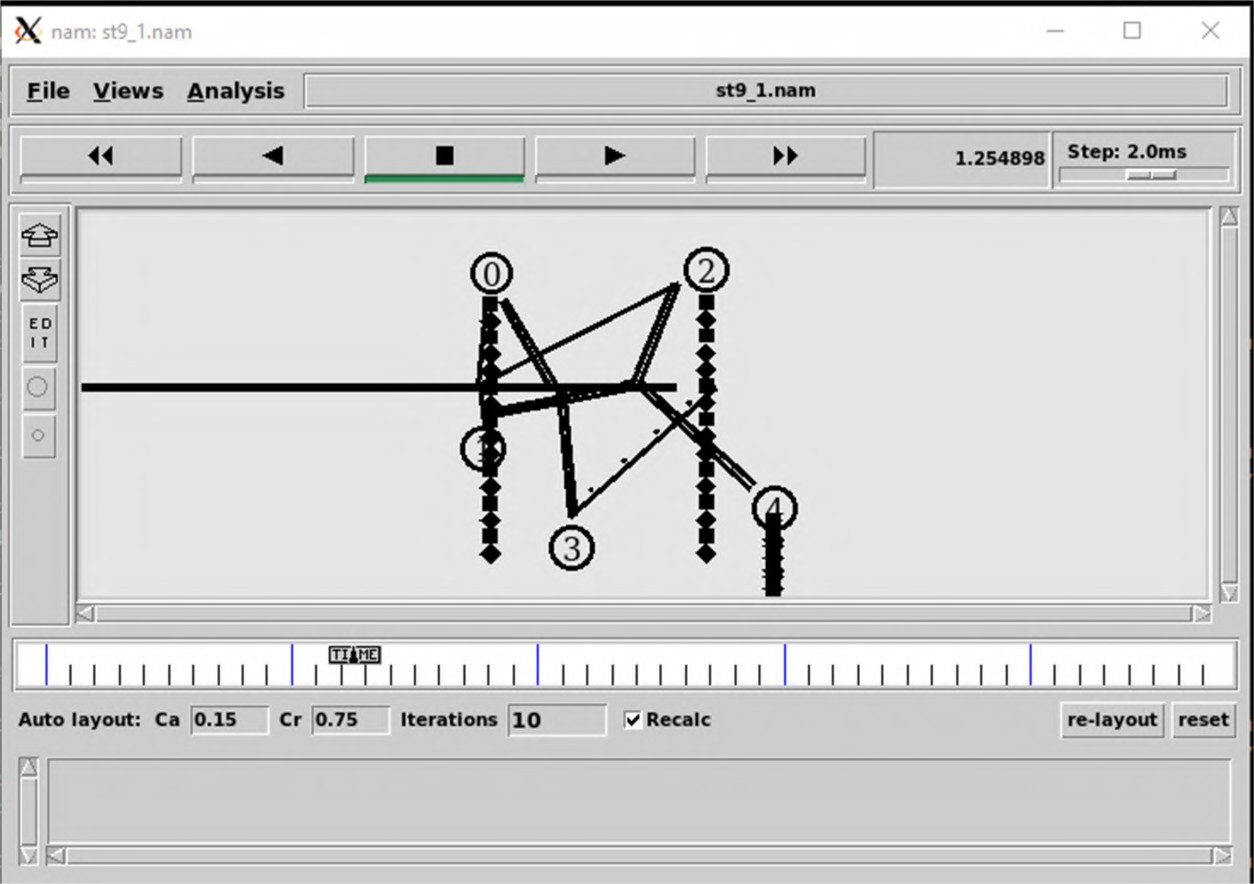
$ns at 0.5 "$cbr0 start"

$ns at 4.5 "$cbr0 stop"

$ns at 5.0 "finish"

$ns run

# Screenshots:



**2). Ring Topology:**

set ns [ new Simulator ]

set nf [ open st9\_2.nam w ]

$ns namtrace-all $nf

set nfl [ open st9\_2.tr w ]

$ns trace-all $nfl

proc finish { } {

global ns nf nfl

$ns flush-trace close $nf

close $nfl

exec nam st9\_2.nam & exit 0

}

set n0 [ $ns node ] set n1 [ $ns node ] set n2 [ $ns node ] set n3 [ $ns node ] set n4 [ $ns node ] set n5 [ $ns node ] set n6 [ $ns node ] set n7 [ $ns node ]

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

$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 $n0 1Mb 10ms DropTail

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

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

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

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

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

$ns duplex-link-op $n5 $n6 orient down

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

$ns duplex-link-op $n7 $n0 orient left

set tcp0 [ new Agent/TCP ]

$tcp0 set class\_ 1

$ns attach-agent $n0 $tcp0

set sink0 [ new Agent/TCPSink ]

$ns attach-agent $n3 $sink0

$ns connect $tcp0 $sink0

set cbr0 [ new Application/Traffic/CBR ]

$cbr0 set packetSize\_ 500

$cbr0 set interval\_ 0.01

$cbr0 attach-agent $tcp0

$ns at 0.0 "$n0 label Source"

$ns at 0.0 "$n3 label Destination"

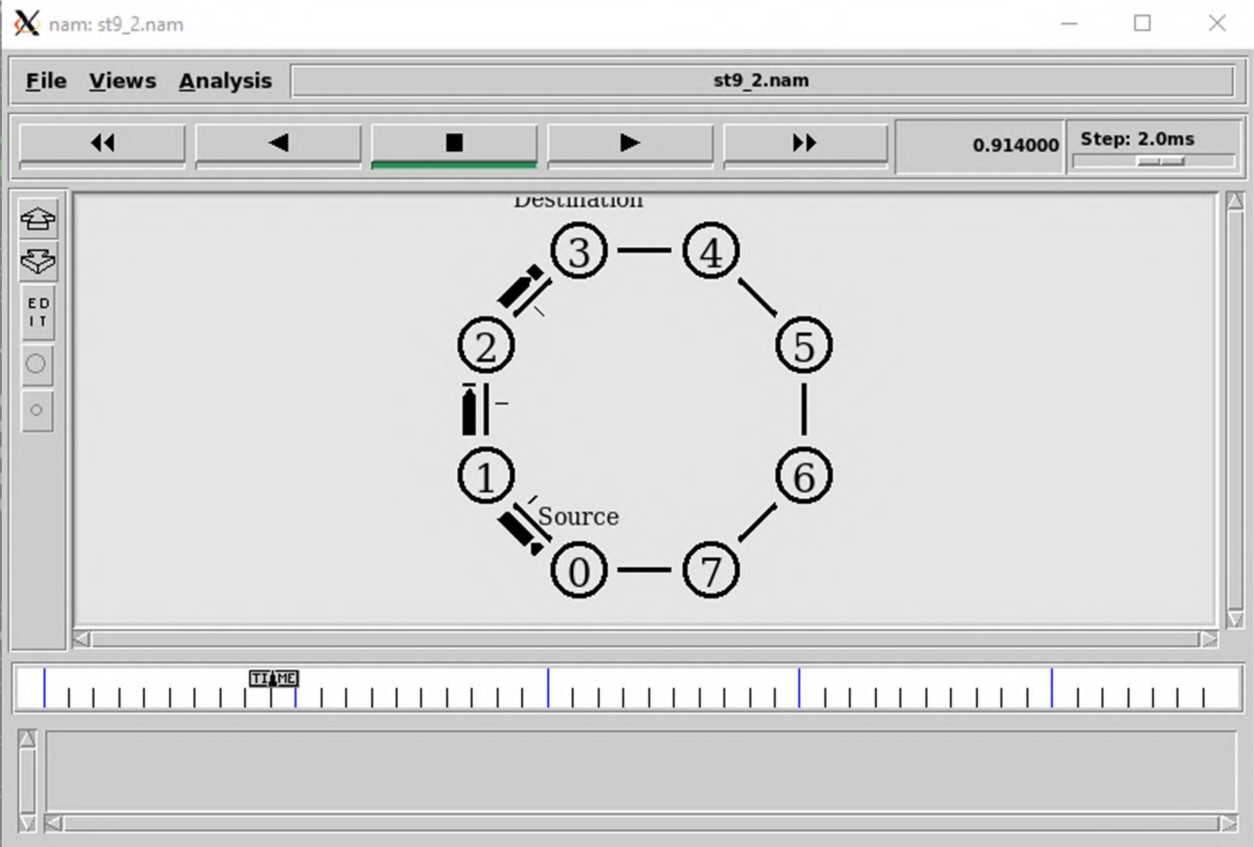
$ns at 0.5 "$cbr0 start"

$ns at 4.5 "$cbr0 stop"

$ns at 5.0 "finish"

$ns run

# Screenshots:



**3) Star Topology**

set ns [ new Simulator ]

set nf [ open st9\_2.nam w ]

$ns namtrace-all $nf

set nfl [ open st9\_2.tr w ]

$ns trace-all $nfl

proc finish { } {

global ns nf nfl

$ns flush-trace close $nf

close $nfl

exec nam st9\_2.nam & exit 0

}

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

$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

set tcp0 [ new Agent/TCP ]

$tcp0 set class\_ 1

$ns attach-agent $n2 $tcp0

set sink0 [ new Agent/TCPSink ]

$ns attach-agent $n5 $sink0

$ns connect $tcp0 $sink0

set cbr0 [ new Application/Traffic/CBR ]

$cbr0 set packetSize\_ 500

$cbr0 set interval\_ 0.01

$cbr0 attach-agent $tcp0

$ns at 0.0 "$n2 label Source"

$ns at 0.0 "$n5 label Destination"

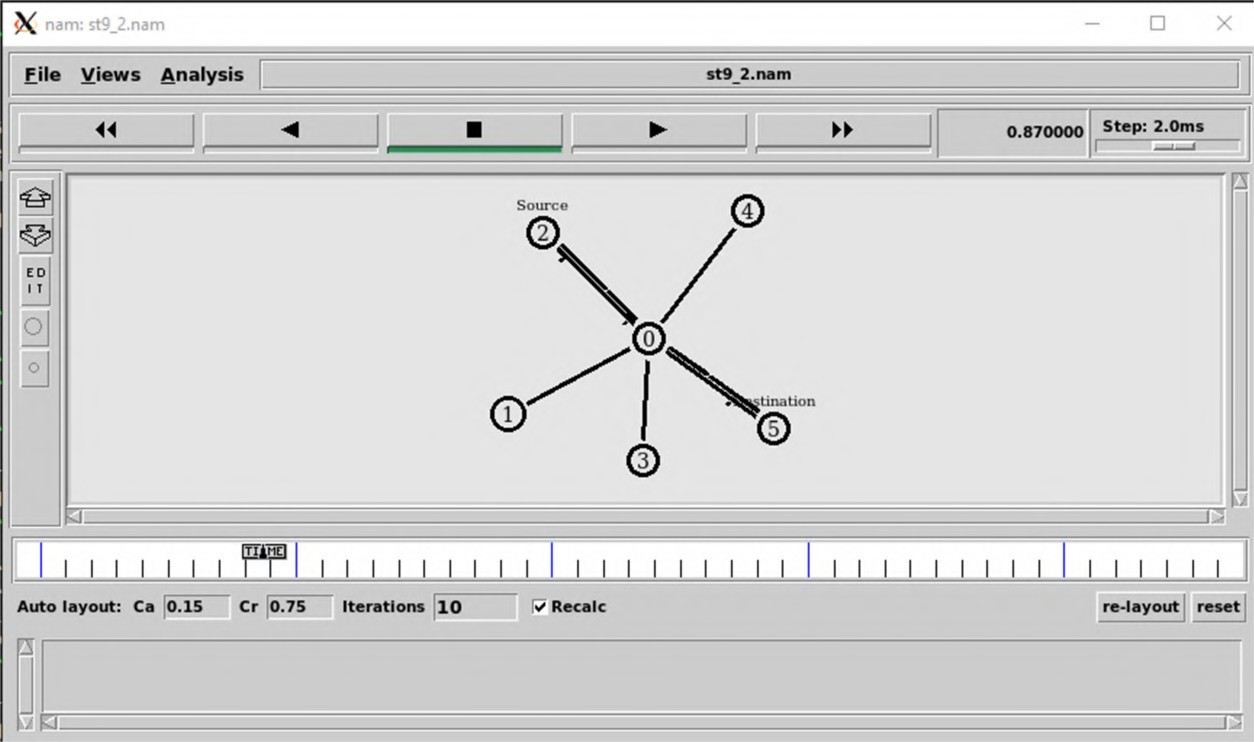
$ns at 0.5 "$cbr0 start"

$ns at 4.5 "$cbr0 stop"

$ns at 5.0 "finish"

$ns run

# Screenshots:



**Conclusion:** Thus we have create Bus , start and ring topology using TCL scripts.

**Experiment No 10**

**Aim:** Simulating of a network with multicast routing protocol.

# Theory:

Multicast routing is a networking method for efficient distribution of one to many traffic. A multicast source such as live video conference sends traffic in one stream to a multicast group. These multicast group contains receivers such as computers, devices and IP phones.

Common uses of these technologies includes: Voice over IP (VOIP), Video on Demand (VOD), Video conferencing and IP television.

# Code:

set ns [ new Simulator -multicast on ]

$ns multicast

set nf [ open st10.nam w ]

$ns namtrace-all $nf

set nfl [ open st10.tr w ]

$ns trace-all $nfl

proc finish { } {

global ns nf nfl

$ns flush-trace close $nfl

exec nam st10.nam & exit 0

}

set n0 [ $ns node ] set n1 [ $ns node ] set n2 [ $ns node ] set n3 [ $ns node ] set n4 [ $ns node ] set n5 [ $ns node ] set n6 [ $ns node ] set n7 [ $ns node ]

$ns duplex-link $n0 $n2 1.5Mb 10ms DropTail

$ns duplex-link $n1 $n2 1.5Mb 10ms DropTail

$ns duplex-link $n2 $n3 1.5Mb 10ms DropTail

$ns duplex-link $n3 $n4 1.5Mb 10ms DropTail

$ns duplex-link $n3 $n7 1.5Mb 10ms DropTail

$ns duplex-link $n4 $n5 1.5Mb 10ms DropTail

$ns duplex-link $n4 $n6 1.5Mb 10ms DropTail

set mproto DM

set mrthandle [ $ns mrtproto $mproto {} ]

# allocate group address

set group1 [ Node allocaddr ] set group2 [ Node allocaddr ]

set udp0 [ new Agent/UDP ]

$ns attach-agent $n0 $udp0

$udp0 set dst\_addr\_ $group1

$udp0 set dst\_port\_ 0

set cbr1 [ new Application/Traffic/CBR ]

$cbr1 attach-agent $udp0

set udp1 [ new Agent/UDP ]

$ns attach-agent $n1 $udp1

$udp1 set dst\_addr\_ $group2

$udp1 set dst\_port\_ 0

set cbr2 [ new Application/Traffic/CBR ]

$cbr2 attach-agent $udp1

#create receiver

set revr1 [ new Agent/Null ]

$ns attach-agent $n5 $revr1

$ns at 1.0 "$n5 join-group $revr1 $group1"

set revr2 [ new Agent/Null ]

$ns attach-agent $n6 $revr2

$ns at 1.5 "$n6 join-group $revr2 $group1"

set revr3 [ new Agent/Null ]

$ns attach-agent $n7 $revr3

$ns at 2.0 "$n7 join-group $revr3 $group1"

set revr4 [ new Agent/Null ]

$ns attach-agent $n5 $revr4

$ns at 2.5 "$n5 join-group $revr4 $group2"

set revr5 [ new Agent/Null ]

$ns attach-agent $n6 $revr5

$ns at 3.0 "$n6 join-group $revr5 $group2"

set revr6 [ new Agent/Null ]

$ns attach-agent $n7 $revr6

$ns at 3.5 "$n7 join-group $revr6 $group2"

$ns at 4.0 "$n5 leave-group $revr1 $group1"

$ns at 4.5 "$n6 leave-group $revr2 $group1"

$ns at 5.0 "$n7 leave-group $revr3 $group1"

$ns at 5.5 "$n5 leave-group $revr4 $group2"

$ns at 6.0 "$n6 leave-group $revr5 $group2"

$ns at 6.5 "$n7 leave-group $revr6 $group2"

# scheduling events

$ns at 0.5 "$cbr1 start"

$ns at 9.5 "$cbr1 stop"

$ns at 0.5 "$cbr2 start"

$ns at 9.5 "$cbr2 stop"

$ns at 10.0 "finish"

$ns color 10 red

$ns color 11 green

$ns color 30 purple

$ns color 31 green

$udp0 set fid\_ 10

$n0 color red

$n0 label "source 1"

$udp1 set fid\_ 11

$n1 color green

$n1 label "source 2"

$n5 label "receiver 1"

$n5 color blue

$n6 label "receiver 3"

$n6 color blue

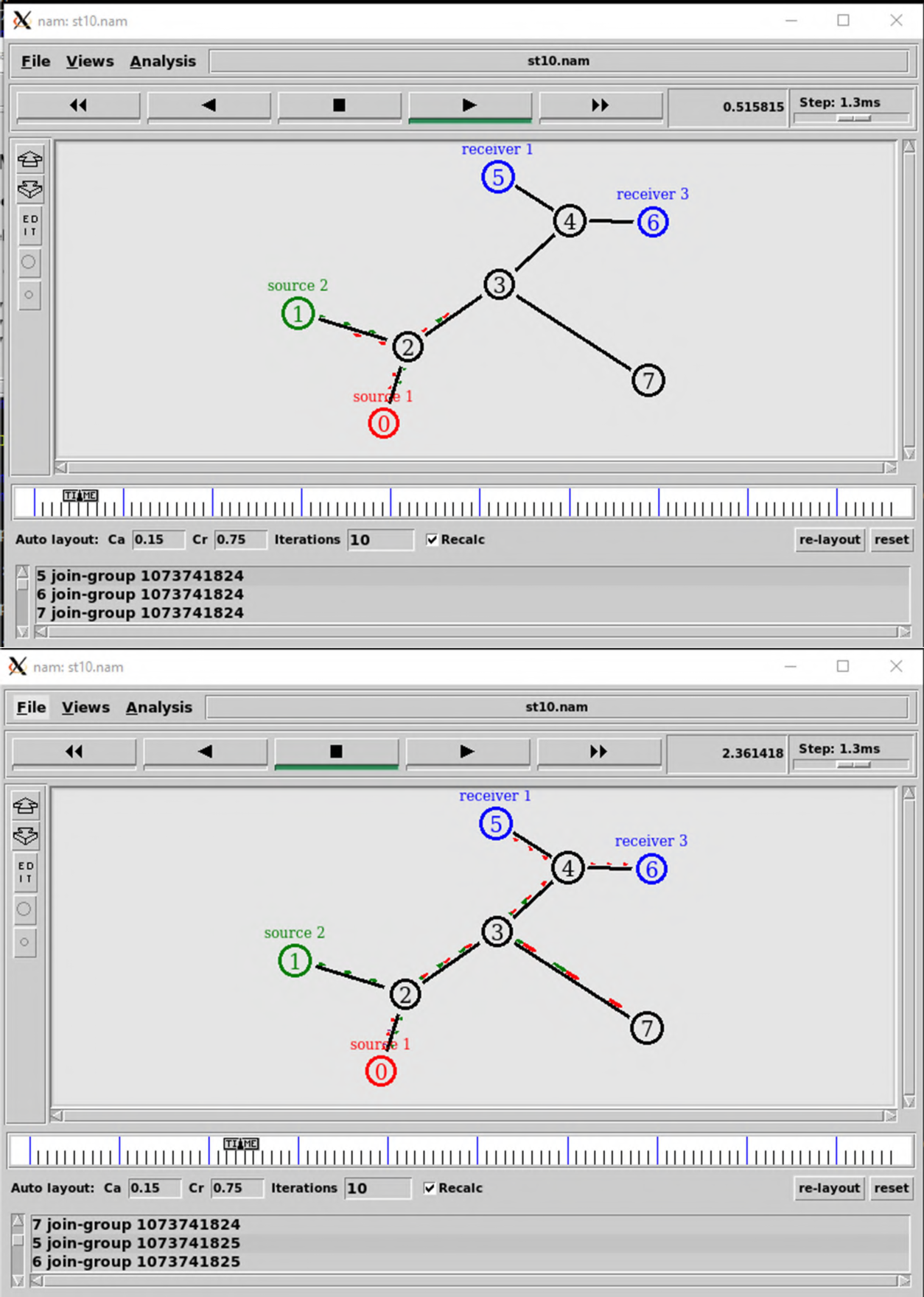
$n2 add-mark m0 red

$n2 delete-mark m0

$ns set-animation-rate 3.0ms

$ns run

# Screenshots:



**Conclusion:** Thus we have successfully implemented multicast routing protocol.