NS2 - GUIDE

KIRAN T S B160244CS | SNEHA K B160268CS | RAHUL M S B1602678CS | C P ROSHAN B160258CS

INTRODUCTION

NS2 is an open-source simulation tool that runs on Linux. Network Simulator is licensed for use under version 2 of the GNU (General Public License) and is popularly known as NS2. It is a discrete event simulator targeted at networking research and provides substantial support for simulation of routing, multicast protocols and IP protocols, such as UDP, TCP, RTP and SRM over wired and wireless (local and satellite) networks. It has many advantages that make it a useful tool, such as support for multiple protocols and the capability of graphically detailing network traffic. Additionally, NS2 supports several algorithms in routing and queuing.
NS is licensed for use under version 2 of the GNU (General Public License) and is popularly known as NS2. This guide is for installing NS2 on Ubuntu 18.10. NS2 is outdated compared to Ubuntu 18.10 and it may not support the latest C and C++ compilers.

INSTALLATION

Step 1

Download the latest NS2 package from Sourceforge. Download the file ***ns-allinone-2.35.tar.gz***, which contains NS2.35 and other necessary tools. Extract the file to the home directory.

```
~$ tar zxvf ns-allinone-2.35.tar.gz
```

Step 2

Install gcc 4.8 and g++ 4.8 which is supported by NS2.35. Also install build-essential, autoconf, automake & libxmu-dev which are essential for installing NS2.

```
~$ sudo apt install gcc-4.8 g++-4.8 build-essential autoconf
automake libxmu-dev
```

Step 3

Change the compiler version from current version to gcc 4.8 & g++ 4.8 in Makefile.in (/ns-allinone-2.35/ns-2.35/).

Step 4

Change erase to this->erase in line number 137, on file ls.h(/ns-allinone-2.35/ns-2.35/linkstate/)

Step 5
Install the package

```
~$ cd ns-allinone-2.35/ns-2.35/
~$ ./install
```

Step 6
Set the path in the bash file

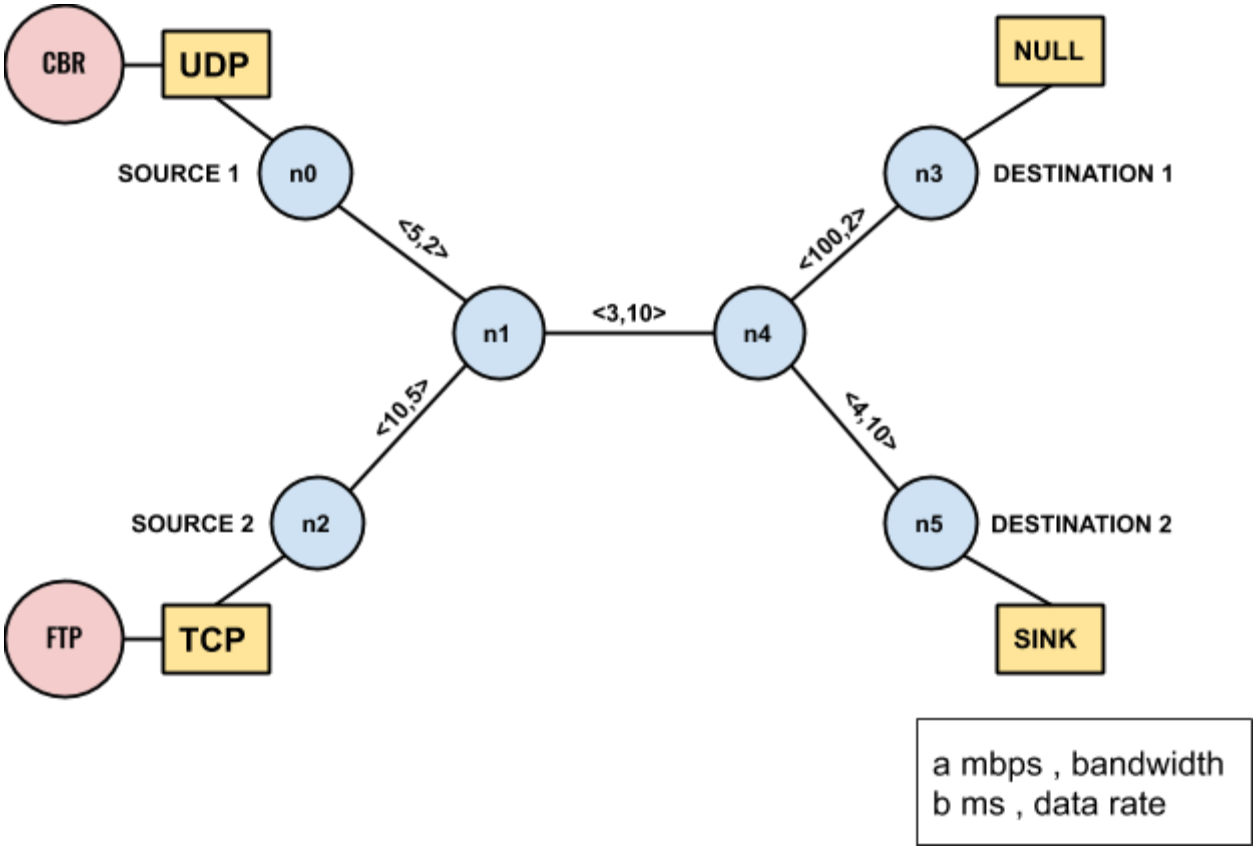
```
~$ gedit /home/username/.bashrc

export
PATH=$PATH:/home/username/ns-allinone-2.35/bin:/home/username/ns
-allinone-2.35/tcl8.5.10/unix:/home/username/ns-allinone-2.35/tk
8.5.10/unix

export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/username/ns-allinone-2.35
/otcl-1.14:/home/username/ns-allinone-2.35/lib
```

Step 7
Install NAM for visualization.

```
~$ sudo apt install nam
```

Step 8
Restart your machine

WIRED NETWORK DESIGN



IMPLEMENTATION

1. Create a .tcl file.

```
~$ gedit network.tcl
```

2. Create a new simulator object. **Simulator** is a C++ class which was precompiled.

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

3. Create a tracefile for logging purposes and link it to the simulator object.

```
set tracefile [open network.tr w]
$ns trace-all $tracefile
```

4. Create a NAM file for storing animation information.

```
set namfile [open network.nam w]
$ns namtrace-all $namfile
```

5. Create nodes.

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

6. Create links between the nodes using duplex link and Droptail Queue.

```
$ns duplex-link $n0 $n1 5Mb 2ms DropTail
$ns duplex-link $n1 $n2 10Mb 5ms DropTail
$ns duplex-link $n1 $n4 3Mb 10ms DropTail
$ns duplex-link $n3 $n4 100Mb 2ms DropTail
$ns duplex-link $n4 $n5 4Mb 10ms DropTail
```

7. Create agents (TCP, UDP, NULL, SINK).

```
# Setup  UDP connection
# node_0 - node_3
set udp [new Agent/UDP]
set null [new Agent/Null]
$ns attach-agent $n0 $udp
$ns attach-agent $n3 $null
$ns connect $udp $null
```

```
# Setup  TCP connection
# node_2 - node_5
set tcp [new Agent/TCP]
set sink [new Agent/TCPSink]
$ns attach-agent $n2 $tcp
$ns attach-agent $n5 $sink
$ns connect $tcp $sink
```

8. Create Applications(CBR, FTP).

```
# Setup Constant Bit Rate over UDP connection
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
```

```
# Setup File Transfer Protocol over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
```

9. Schedule the network traffic.

```
$ns at 1.0 "$cbr start"
$ns at 2.0 "$ftp start"

$ns at 10.0 "finish"
```

10. Define the procedure ‘finish’.

```
proc finish {} {
        global ns tracefile namfile
        # flush all the log files to tracefile
        $ns flush-trace
        # Close the NAM trace file
        close $namfile
        # Close the trace file
        close $tracefile
        exit 0
}
```

11. Run the simulation.

```
$ns run
```

12. Compile the file.

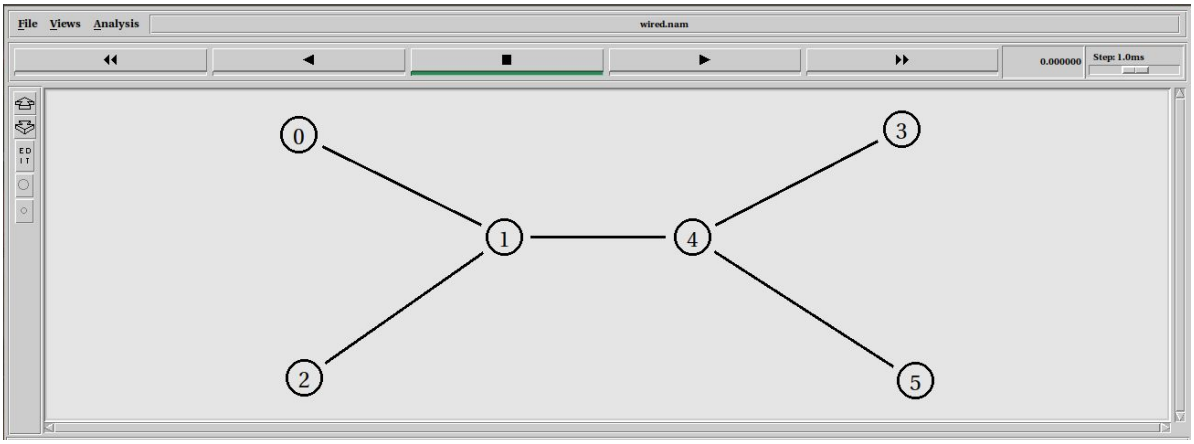
```
~$ ns network.tcl
```

13. Run the NAM file.

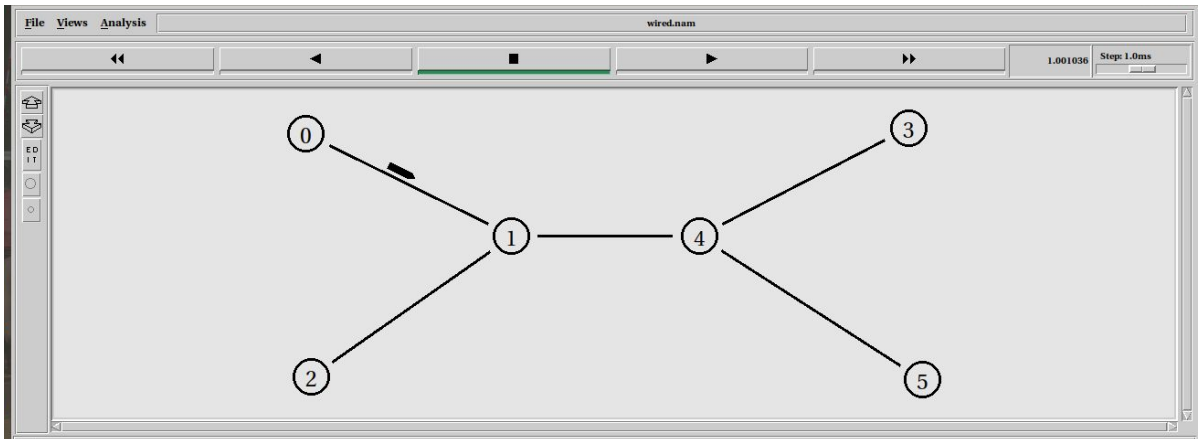
```
~$ nam network.nam
```

NETWORK ANIMATOR(NAM) RESULT

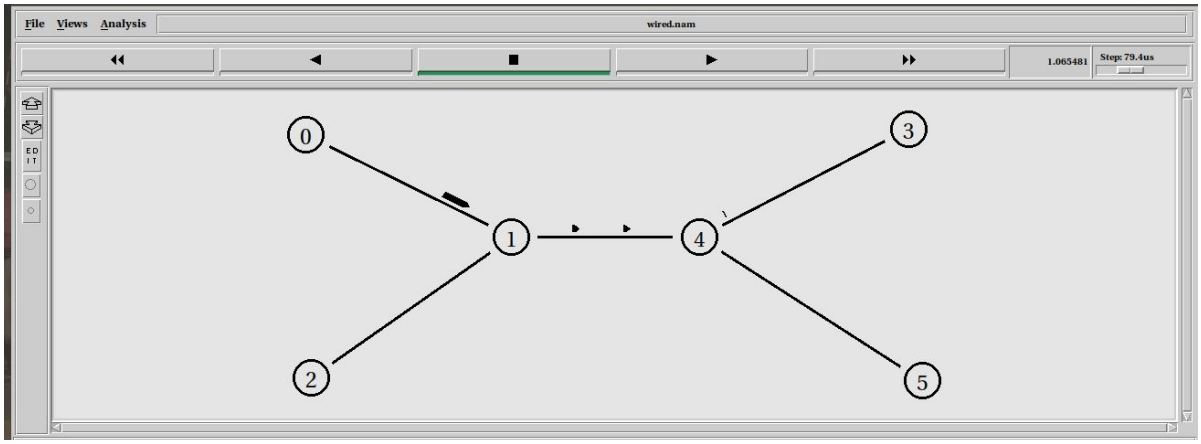
1. Time = 0s, No Traffic.



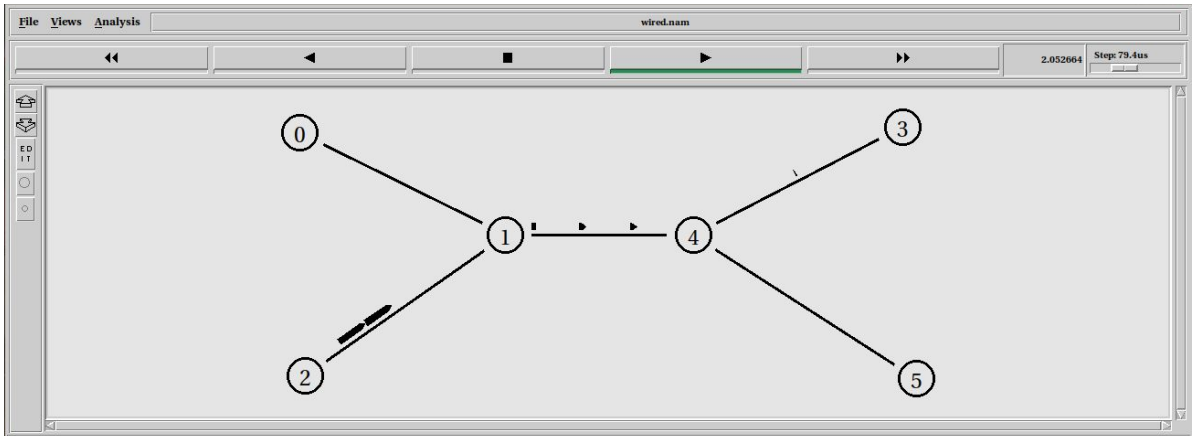
2. Time = 1s, Traffic starts from node 1.



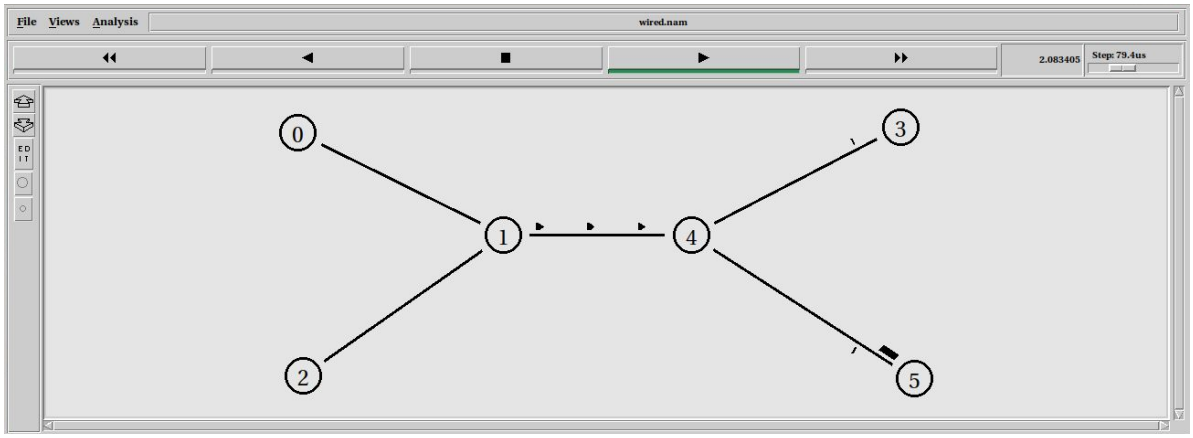
You can notice the difference in the size of packets on each duplex link depending on their bandwidth and the speed at which packets move based on their data rates

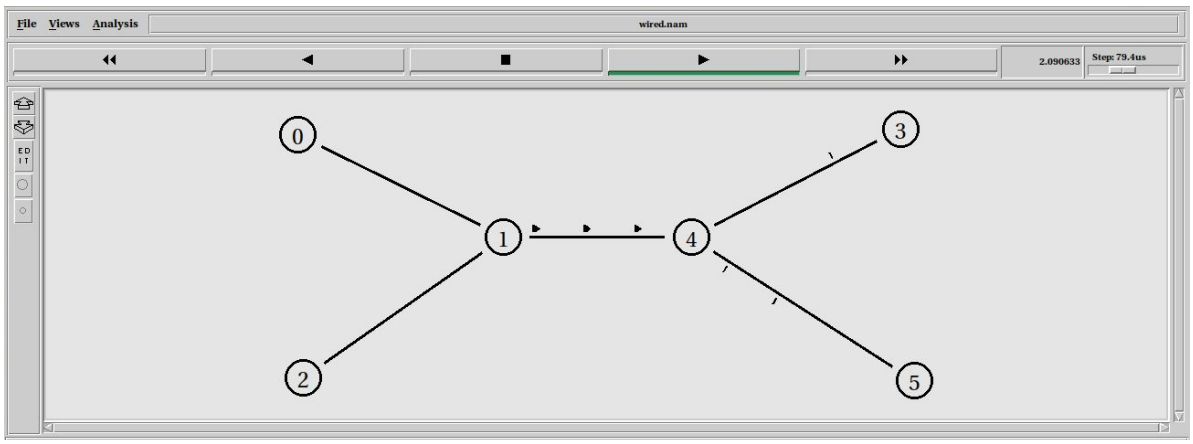


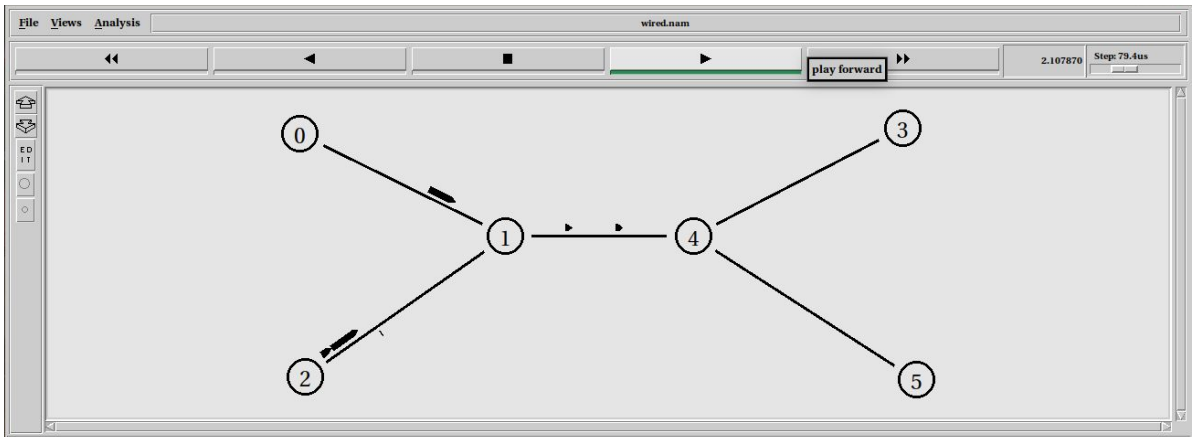
3. Time = 2s, node 2 performs TCP handshake with node 5 and upon receiving acknowledgment, the traffic starts from node 2.

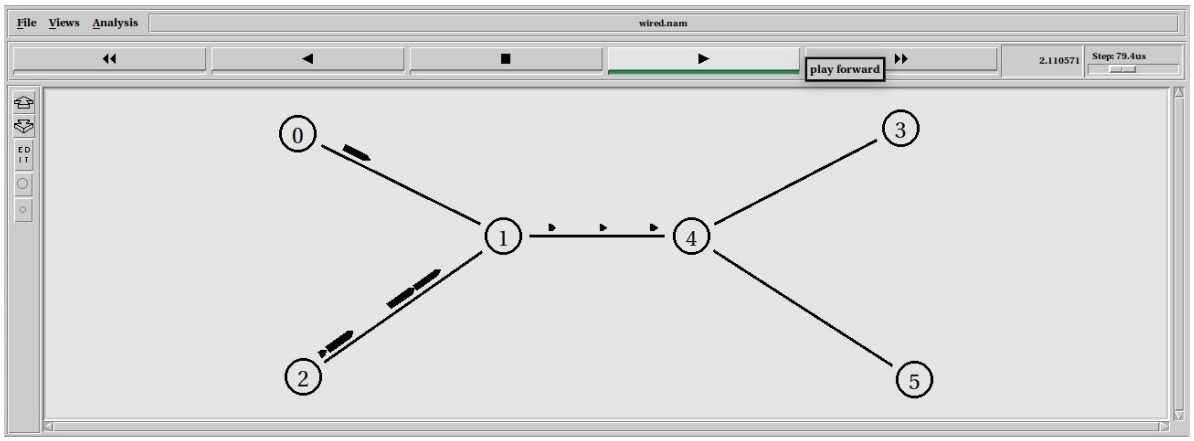


node 2 sends the next packet only after receiving acknowledgment bits from node 5

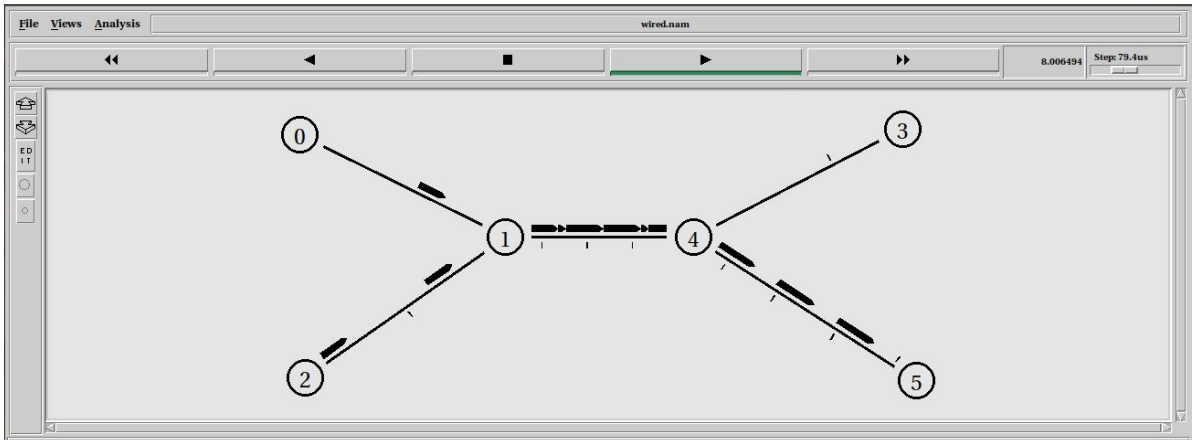




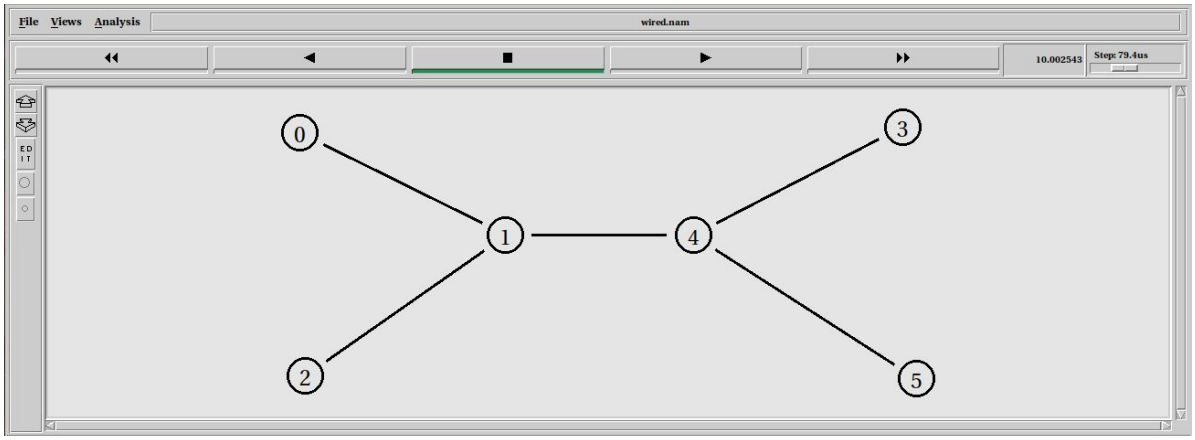




4. Time = 8s, the traffic continues without any queue overflows.



5. Time = 10s, the traffic stops.



6. If we change the bandwidth of duplex link node 1 to node 4 from 3 Mbps to 1 Mbps, we can notice packet loss at node 1 at Time = 4s. Since we have implemented the Droptail queue, the packets at the end of the queue are discarded.