# K. J. Somaiya College of Engineering, Mumbai-77

(Autonomous College Affiliated to University of Mumbai)

**Batch: B1** **Roll No.: 1711072**

**Experiment / assignment / tutorial No. 6**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

## Experiment No.:6

**TITLE:** To Simulate a simple Network and analyse the packet flow between the nodes using NS-2.

**AIM:** To Simulate a simple Network and analyze the packet flow between the nodes using NS-2.

**Expected Outcome of Experiment:**
**CO: Explain the fundamentals of the data communication networks, reference models, topologies, physical media, devices, simulators and identify their use in day to day networks.**

**Books/ Journals/ Websites referred:**

1. A. S. Tanenbaum, "Computer Networks", Pearson Education, Fourth Edition
2. B. A. Forouzan, "Data Communications and Networking", TMH, Fourth Edition
3. http://www.isi.edu/nsnam/ns

**Pre Lab/ Prior Concepts:** Simple Network flow, Basic NS2 concepts

**New Concepts to be learned:** TCL language Node, Queue, Network Layout

**Stepwise-Procedure:**

NS-2 simulation involves following steps:

o Initialization & termination aspects of ns simulator

o Definition of network nodes, links, queues & topology o Definition of agents & of applications
o The NAM visualization tool
o Tracing

Graphical plots using x-graph

1. **Initialization and termination**

Create simulator object:

set ns [new simulator]

Open output files with data on the simulation (trace files) set
tracefile1 [open out.tr w]

$ns trace-all $tracefile1

Open a file for writing data for input to nam (network animator) set
namfile [open out.nam w]

$ns namtrace-all $namfile

Finish procedure :

```
proc finish {} {

  global nstracefile1 namfile close
  $tracefile1
  close $namfile

  exec nam out.nam & exit 0

  }
```

Tell simulator object when to finish $ns at
5.0 “finish”

Start the simulation $ns run

2. **Definition of network of links and nodes**

Creating nodes

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

Creating link between nodes

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

Define buffer capacity of the queue related to each link
$ns queue-limit $n0 $n1 20

3. **Agents and applications**

In ns the data is transferred between agents & not nodes, therefore we need to define

1. routing (source,destinations)

2. agents (protocols)

3. Applications that use them

Setup a TCP connection

```
set tcp [new agent/TCP]
    $ns attach-agent $n0
    $tcpset sink [new agent/TCPSink]
    $ns attach-agent $n1 $sink $ns connect $tcp $sinkl

     $tcp set fid_ 1

     $tcp set packetSize_ 552  // by default 1000
```

Setup a FTP over TCP connection

```
set ftp [new Application/FTP] $ftp attach-agent $tcp
```

Telnet

```
set telnet [new Application/Telnet] $telnet
attach-agent $tcp0
```

Setup a UDP connection

```
set udp [new Agent/UDP]
$ns attach-agent $n0 $udp setnull [new Agent/Null]
$ns attach-agent $n1 $null
$ns connect $udp $null $udp set fid_ 2
```

Setup a CBR over UDP connection

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

$cbr set packetSize_ 500 $cbr set
interval_ 0.005 $cbr set random_
false
```

4. **Scheduling events**

Start and stop of data $ns at <time> <event>

```
e.g. $ns at 0.1 “$cbr start”
     $ns at 1.0 “$ftp start”
     $ns at 15.0 “$ftp stop”
     $ns at 20.0 “$cbr stop
```

5. **Visulisation using nam**

 Give node position

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

 To obtain color

$ns color 1 Blue

 Other things

1. Coloring nodes - $n0 color red

2. Shape of nodes - $n1 shape box

   (shape can be box/ hexagon/ circle, default is circle)

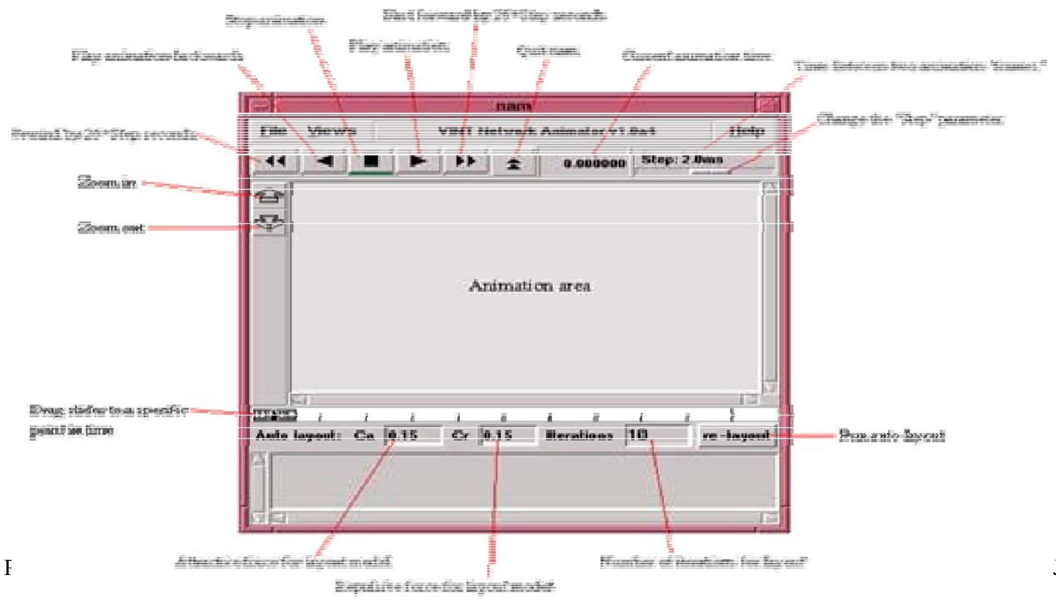
3. Coloring links- $ns duplex-link-op $n0 $n2 color "green"

4. Adding and removing marks

   $ns at 2.0 "$n3 add-mark m3 blue box" $ns at
   30.0 "$n3 delete-mark m3"

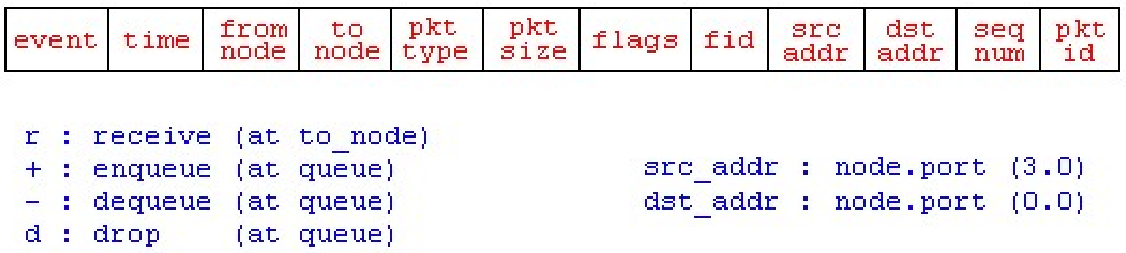
5. Adding labels
   At node - $ns at 1.2 "$n3 label \" active node\" "
   To link – $ns duplex-link-op $n0 $n2 label "TCP input link "
6. Adding text:(Annotation)
   $ns at 5 "$ns trace-annotate \"packet drop\" "
7. Monitoring queue size

   $ns simplex-link-op $n2 $n3 queuePos 0.5


Stop animation
Fast forward by 25*Step seconds
Play animation backwards
Play animation
Quit nam
Current animation time
Time between two animation "frames"
nam
File
Views
VINT Network Animator v1.0a4
Help
Rewind by 25*Step seconds
0.000000
Step: 2.0ms
Change the "Step" parameter
Zoom in
Zoom out
Animation area
Drag slider to a specific point in time
Auto layout: Ca 0.15 Cr 0.15 Iterations 10
re-layout
Run auto-layout
Attractive force for layout model
Repulsive force for layout model
Number of iterations for layout

6. **Tracing**

Tracing into an o/p ASCII file is organized into 12 fields as shown-



| event | time | from node | to node | pkt type | pkt size | flags | fid | src addr | dst addr | seq num | pkt id |
|---|---|---|---|---|---|---|---|---|---|---|---|

```
r : receive (at to_node)
+ : enqueue (at queue)                src_addr : node.port (3.0)
- : dequeue (at queue)                dst_addr : node.port (0.0)
d : drop    (at queue)
```

Fig above shows the Trace file format with the following fields

*1.event*: Event _Type: r, +, -, d : receive , enqueued ,dequeued & dropped packets

*2. time:* time at which event occurs

*3. from node*: i/p node of link at which event occurs
*4. to node:* o/p node of link at which event occurs

*5. pkt type:* cbr or tcp (depending upon application)

*6. pkt size:* size in bytes

*7. flags:* some flags
*8. fid:* Flow ID of IPv6, Used to specify stream color for nam display
*9. src addr:* Source address in "node.port" form

*10.dst addr:* destination address in the same format
*11.seq num:* N/w layer protocol's sequence number
*12.Pkt id:* unique id of packet

**IMPLEMENTATION:** (printout of code)

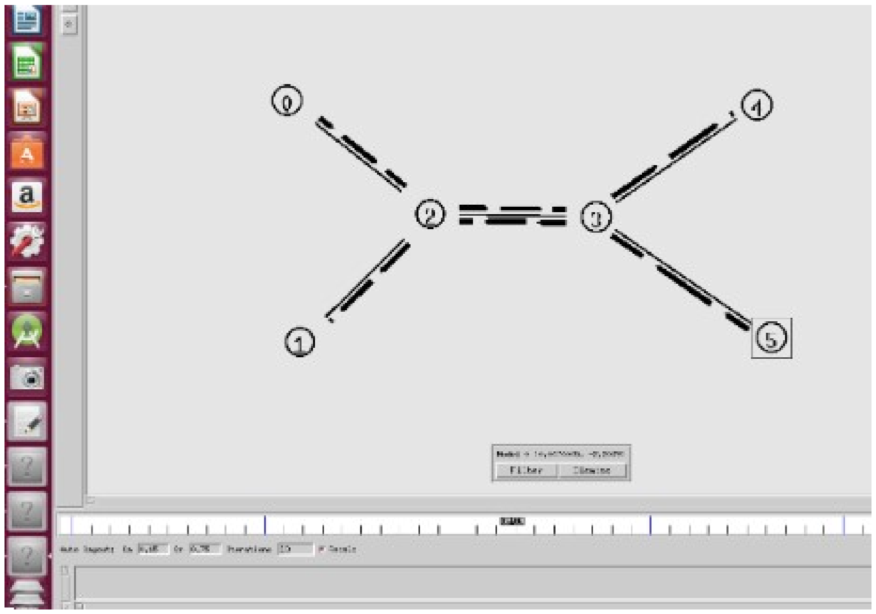
**Program 1 (UDP, CBR):**

```
#Create a simulator object
set ns [new Simulator]

#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 &
exit 0
}

#Create two 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]
#Create a duplex link between the nodes
$ns duplex-link $n0 $n2 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 $n3 $n5 1Mb 10ms DropTail

#Create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
set udp1 [new Agent/UDP]
$ns attach-agent $n0 $udp0
$ns attach-agent $n5 $udp1
# Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
set cbr1 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1
#Create a Null agent (a traffic sink) and attach it to node n1
set null0 [new Agent/Null]
set null1 [new Agent/Null]
$ns attach-agent $n4 $null0
$ns attach-agent $n1 $null1
#Connect the traffic source with the traffic sink
```

```
$ns connect $udp0 $null0
$ns connect $udp1 $null1
#Schedule events for the CBR agent
$ns at 0.5 "$cbr0 start"
$ns at 2.5 "$cbr0 stop"
$ns at 2.5 "$cbr1 start"
$ns at 4.5 "$cbr1 stop"
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"

#Run the simulation
$ns run
```

**Output Screen:**



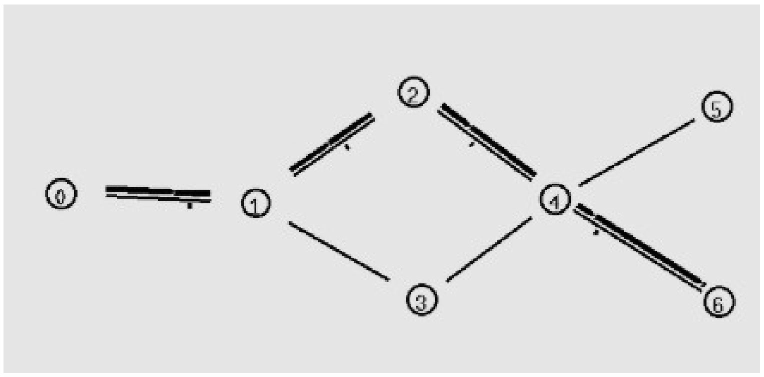
**Program 2 (TCP-FTP):**

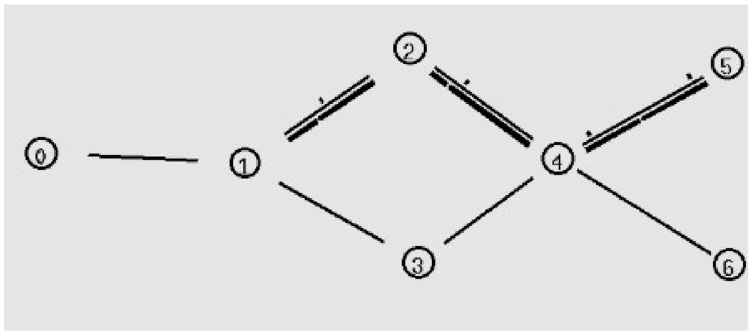
```
#Create a simulator object
set ns [new Simulator]

#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf

#Define a 'finish' procedure
```

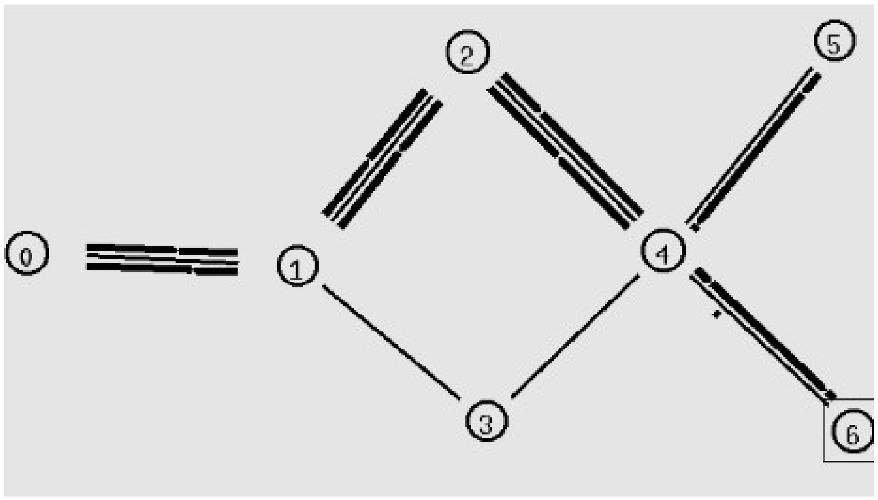
```
proc finish {} {
global ns nf
$ns flush-trace

#Close the trace file
close $nf

#Execute nam on the trace file
exec nam out.nam &
exit 0
}

#Create two 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]
set n6 [$ns node]
#Create a duplex link between the nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n3 1Mb 10ms DropTail
$ns duplex-link $n2 $n4 1Mb 10ms DropTail
$ns duplex-link $n3 $n4 1Mb 10ms DropTail
$ns duplex-link $n4 $n5 1Mb 10ms DropTail
$ns duplex-link $n4 $n6 1Mb 10ms DropTail
#Create a TCP agent and attach it to node n0
set tcp0 [new Agent/TCP]
set tcp1 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
$ns attach-agent $n5 $tcp1
# Create a FTP source and attach it to tcp0
set ftp0 [new Application/FTP]
set ftp1 [new Application/FTP]
#$ftp0 set packetSize_ 500
#$ftp0 set interval_ 0.005
$ftp0 attach-agent $tcp0
#$ftp1 set packetSize_ 500
#$ftp1 set interval_ 0.005
$ftp1 attach-agent $tcp1
#Create a TCPSink agent (a traffic sink) and attach it to node n1,n6
set tcpsink0 [new Agent/TCPSink]
set tcpsink1 [new Agent/TCPSink]
$ns attach-agent $n6 $tcpsink0
```

```
$ns attach-agent $n1 $tcpsink1
#Connect the traffic source with the traffic sink
$ns connect $tcp0 $tcpsink0
$ns connect $tcp1 $tcpsink1
#Schedule events for the ftp agent
$ns at 0.5 "$ftp0 start"
$ns at 3.5 "$ftp0 stop"
$ns at 1.5 "$ftp1 start"
$ns at 4.5 "$ftp1 stop"
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"

#Run the simulation
$ns run
```

**Output Screens:**







**CONCLUSION: The network was simulated using NS2 software and we learned how to code in TCL.**

**<u>Post Lab Questions</u>**

1. What is use of NS2?
2. State different features of NS2.
3. List different network simulators?

**Ans 1.** Network simulator - 2 (ns-2) is a popular open source network simulator for carrying out network experimentation. NS2 is an open-source simulation tool that runs on Linux. It is a discreet 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. LAN routing and broadcasts are part of routing algorithms. Queuing algorithms include fair queuing, deficit round-robin and FIFO. Way back when it was being designed, it's primary usage was to analyse the performance of congestion control algorithms implemented in TCP. Even today, it remains the most widely used network simulator for TCP research. Over the period of time, it gained wide acceptance in industry, and now supports simulation of latest networking paradigms such as MANETs, VANETs, etc.

**Ans 2.**

1. It is a discrete event simulator for networking research.

2. It provides substantial support to simulate bunch of protocols like TCP, FTP, UDP, https and DSR.

3. It simulates wired and wireless network.

4. It is primarily Unix based.

5. Uses TCL as its scripting language.

6. Otcl: Object oriented support

7. Tclcl: C++ and otcl linkage

8. Discrete event scheduler.

**Ans 3.** Some examples of network simulators are:

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