INTRODUCTION TO NS2

AIM:

To familiarise Network Simulator 2

DESCRIPTION:

Network simulator is a piece of software or hardware that predicates the behavior of a network without an actual network being present.NS2 is a simulation tool for networks. It supports several algorithms for routing and queuing. It can set up packet traffic similar to internet and measure various parameters. NS2 is very useful because it is very expensive to check feasibility of new algorithms, check architectures, check topologies etc. network simulator is a name for series of discrete event network simulator. Simulators are used in the simulation of routing protocols, and are heavily used in ad-hoc networking research, and support popular network protocols, offering simulation results for wired and wireless networks alike.

Features

It can be employed in most unix systems and windows. Most of NS2 code is in C++.It uses TCL as its scripting language, Otcl adds object orientation to TCL.NS(version 2) is an object oriented, discrete event driven network simulator that is freely distributed and open source.

- Protocols: TCP, UDP, HTTP, Routing algorithms, MAC etc
- Traffic Models: CBR, VBR, Web etc
- Error Models: Uniform, bursty etc
- Misc: Radio propagation, Mobility models, Energy Models
- Topology Generation tools
- Visualization tools (NAM), Tracing

NS Structure

- NS is an object oriented discrete event simulator
  – Simulator maintains list of events and executes one event after another
  – Single thread of control: no locking or race conditions
- Back end is C++ event scheduler
  – Protocols mostly
  – Fast to run, more control
- Front end is OTCL
  – Creating scenarios, extensions to C++ protocols
  – fast to write and change

Platforms

It can be employed in most unix systems (FreeBSD, Linux, Solaris) and Windows.

Source code

Most of NS2 code is in C++

Scripting language

It uses TCL as its scripting language OTcl adds object orientation to TCL. NS (version 2) is an object-oriented, discrete event driven network simulator developed at UC Berkely written in C++ and OTcl. NS is popularly used in the simulation of routing and multicast protocols, among others, and is heavily used in ad-hoc networking research. ns supports an array of popular network protocols, offering simulation results for wired and wireless networks alike. It can be also used as limited-functionality network emulator. It is popular in academia for its extensibility (due to its open source model) and plentiful online documentation. NS was built in C++ and provides a simulation interface through OTcl, an object-oriented dialect of Tcl. The user describes a network topology by writing OTcl scripts, and then the main NS program simulates that topology with specified parameters.

Protocols implemented in NS2

Transport layer (Traffic Agent) – TCP, UDP

Network layer (Routing agent)

Interface queue – FIFO queue, Drop Tail queue, Priority queue

Logic link control layer – IEEE 802.2, AR

How to use NS2

Design Simulation – Determine simulation scenario

Build ns-2 script using tcl.

Run simulation

Simulation with NS2

- Define objects of simulation.
- Connect the objects to each other
- Start the source applications. Packets are then created and are transmitted through network.
- Exit the simulator after a certain fixed time.

Accompanying Tools

nam

A tool that allows visualizing the motion of packets through the nodes and links of the network. It can either start nam with the command “nam <nam-file>”, where “<namfile>” is the name of a nam trace file that was generated by ns, or can execute it directly out of the Tcl simulation script for the simulation which want to visualize.

Xgraph

A tool that allows to plot the results of the simulation in the form of curves. xgraph is a plotting program which can be used to create graphic representations of simulation results. It can create output files in Tcl scripts, which can be used as data sets for xgraph. Call xgraph to display the results with the command "xgraph <data-file>".

<u>NS programming Structure</u>

● Create the event scheduler

● Turn on tracing

● Create network topology

● Create transport connections

● Generate traffic

● Insert errors

Creating Event Scheduler

● Create event scheduler: set ns [new simulator]

● Schedule an event: $ns at <time> <event>

– event is any legitimate ns/tcl function

```
$ns at 5.0 "finish"
 proc finish {} {
 global ns nf
 close $nf
 exec nam out.nam &
 exit 0
}
```

● Start Scheduler

```
$ns run
```

<u>Tracing</u>

● All packet trace

```
$ns traceall[open out.tr w]
```

```
<event> <time> <from> <to> <pkt> <size>...<flowid> <src> <dst> <seqno> <aseqno>
+  0.51  0  1 cbr  500 -----  0  0.0  1.0   0   2
_   0.51  0  1 cbr  500 ----- 0   0.0  1.0  0   2
R  0.514 0  1 cbr  500 -----0   0.0   1.0  0   0
```

● Variable trace

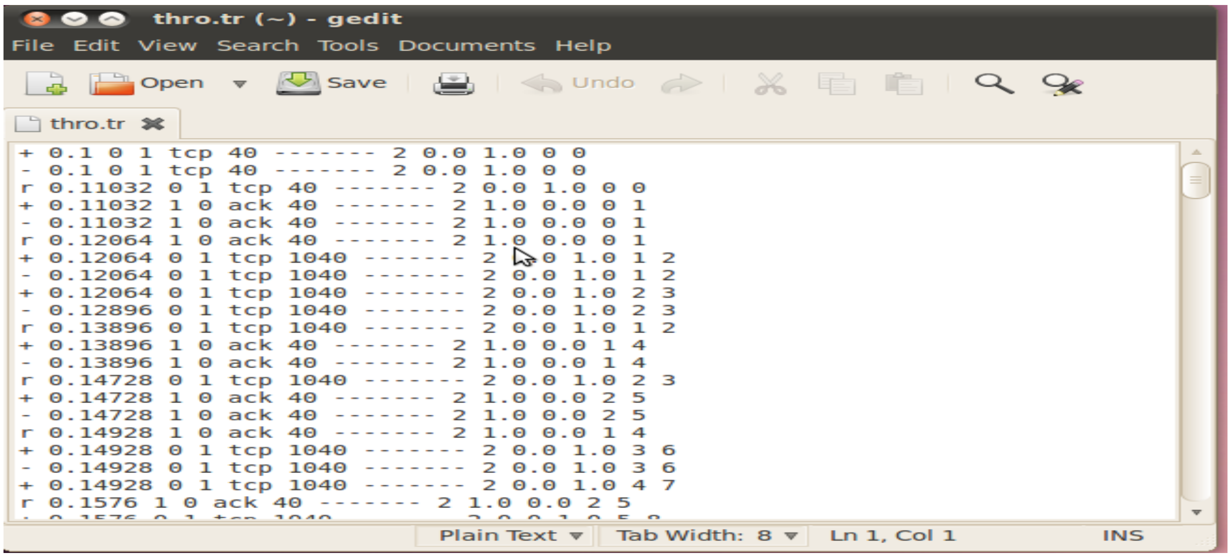
```
set par [open output/param.tr w]
$tcp attach $par
$tcp trace cwnd_
```

```
$tcp trace maxseq_

$tcp trace rtt_
```

Tracing and Animation

● Network Animator

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

$ns namtraceall

$nf

proc finish {} {

global ns nf

close $nf

exec nam out.nam &

exit 0
```



```
+ 0.1 0 1 tcp 40 ------- 2 0.0 1.0 0 0
- 0.1 0 1 tcp 40 ------- 2 0.0 1.0 0 0
r 0.11032 0 1 tcp 40 ------- 2 0.0 1.0 0 0
+ 0.11032 1 0 ack 40 ------- 2 1.0 0.0 0 1
- 0.11032 1 0 ack 40 ------- 2 1.0 0.0 0 1
r 0.12064 1 0 ack 40 ------- 2 1.0 0.0 0 1
+ 0.12064 0 1 tcp 1040 ------- 2 0.0 1.0 1 2
- 0.12064 0 1 tcp 1040 ------- 2 0.0 1.0 1 2
+ 0.12064 0 1 tcp 1040 ------- 2 0.0 1.0 2 3
- 0.12896 0 1 tcp 1040 ------- 2 0.0 1.0 2 3
r 0.13896 0 1 tcp 1040 ------- 2 0.0 1.0 1 2
+ 0.13896 1 0 ack 40 ------- 2 1.0 0.0 1 4
- 0.13896 1 0 ack 40 ------- 2 1.0 0.0 1 4
r 0.14728 0 1 tcp 1040 ------- 2 0.0 1.0 2 3
+ 0.14728 1 0 ack 40 ------- 2 1.0 0.0 2 5
- 0.14728 1 0 ack 40 ------- 2 1.0 0.0 2 5
r 0.14928 1 0 ack 40 ------- 2 1.0 0.0 1 4
+ 0.14928 0 1 tcp 1040 ------- 2 0.0 1.0 3 6
- 0.14928 0 1 tcp 1040 ------- 2 0.0 1.0 3 6
+ 0.14928 0 1 tcp 1040 ------- 2 0.0 1.0 4 7
r 0.1576 1 0 ack 40 ------- 2 1.0 0.0 2 5
```

```
}
```

Creating topology

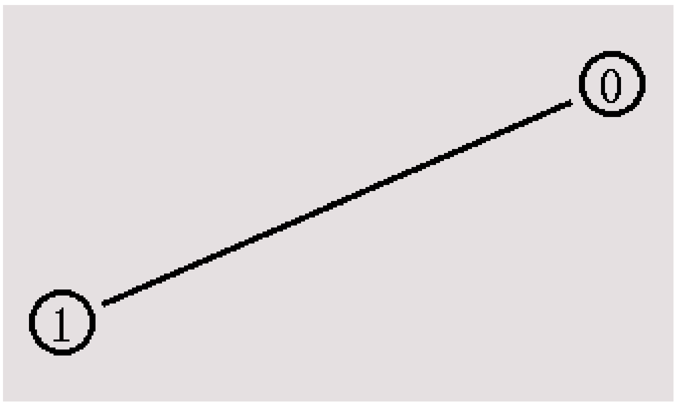
● Two nodes connected by a link

● Creating nodes

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

- Creating link between nodes

```
$ns <link_type> $n0 $n1 <bandwidth> <delay><queue-type>
 $ns duplex-link$n0 $n1 1Mb 10ms DropTail
```



Sending data

- Create UDP agent

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

- Create CBR traffic source for feeding into UDP agent

```
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent$udp0
```

- Create traffic sink

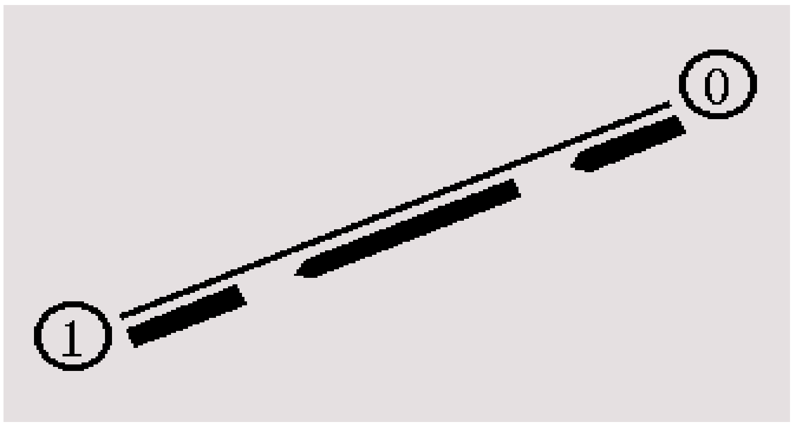
```
set null0 [new Agent/Null]
$ns attach-agent$n1 $null0
```

- Connect two agents

```
$ns connect $udp0 $null0
```

- Start and stop of data

```
$ns at 0.5 “$cbr0 start”
$ns at 4.5 “$cbr0 stop”
```



Creating TCP Connections

● Create TCP agent and attach it to the node

```
set tcp0 [new Agent/TCP]
$ns attach-agent$n0 $tcp0
```

● Create a Null Agent and attach it to the node

```
set null0 [new Agent/TCPSink]
$ns attach-agent$n1 $null0
```

● Connect the agents

```
$ns connect $tcp0 $null0
```

Traffic on top of TCP

● FTP

```
set ftp [new Application/FTP]
$ftp attach-agent$tcp0
```

● Telnet

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

PROCEDURE

STEP 1: Start

STEP 2: Create the simulator object ns for designing the given simulation

STEP 3: Open the trace file and nam file in the write mode

STEP 4: Create the nodes of the simulation using the ‘set’ command

STEP 5: Create links to the appropriate nodes using $ns duplex-link command

STEP 6: Set the orientation for the nodes in the simulation using ‘orient’ command

STEP 7: Create TCP agent for the nodes and attach these agents to the nodes

STEP 8: The traffic generator used is FTP for both node0 and node1

STEP 9: Configure node1 as the sink and attach it

STEP10: Connect node0 and node1 using ‘connect’ command

STEP 11: Setting color for the nodes

STEP 12: Schedule the events for FTP agent 10 sec

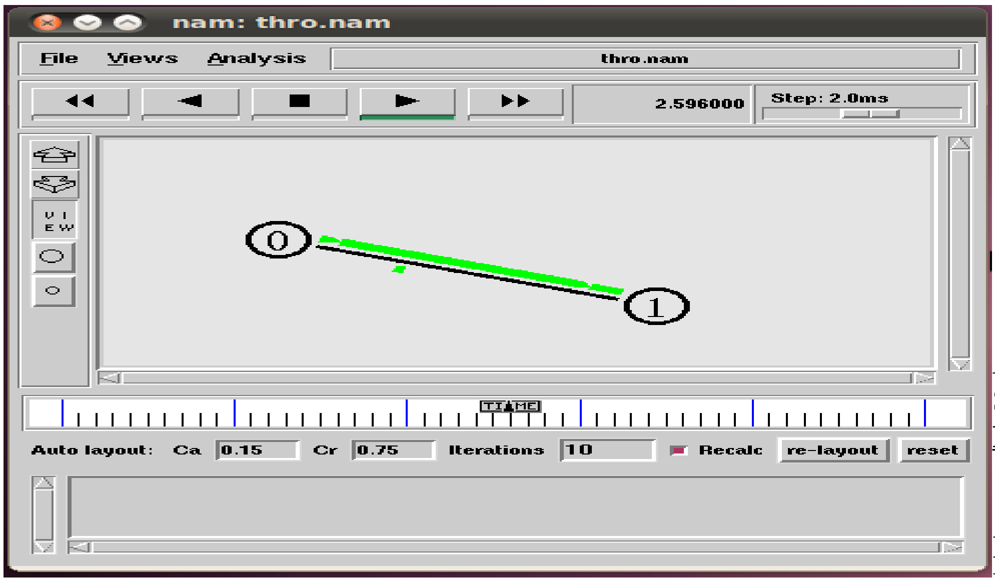
STEP 13: Schedule the simulation for 5 minutes

PROGRAM

```
#create simulator object
set ns [new Simulator]
#open xgraph in write mode
set nr [open thro.tr w]
$ns trace-all $nr
#open the nam file in write mode
set nf [open thro.nam w]
$ns namtrace-all $nf
#define a  finish procedure
        proc finish { } {
        global ns nr nf
        $ns flush-trace
        #close the nam trace file
        close $nf
        close $nr
        exec nam thro.nam &

        exit 0
        }
#Creation of Nodes:
set n0 [$ns node]
set n1 [$ns node]

#Creation of Links:
$ns duplex-link $n0 $n1 1Mb 10ms DropTail

#Connection of node0 with node1
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set ftp [new Application/FTP]
$ftp set packetSize_ 500
$ftp set  interval_ 0.005
$ftp attach-agent $tcp0
set sink [new Agent/TCPSink]
$ns attach-agent $n1 $sink
$ns connect $tcp0 $sink

#Setting color for the nodes
$tcp0 set fid_ 2
$ns color 2 Green

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

```
$ns at 0.1 "$ftp start"
 $ns at 5.0 "finish"
$ns run
```

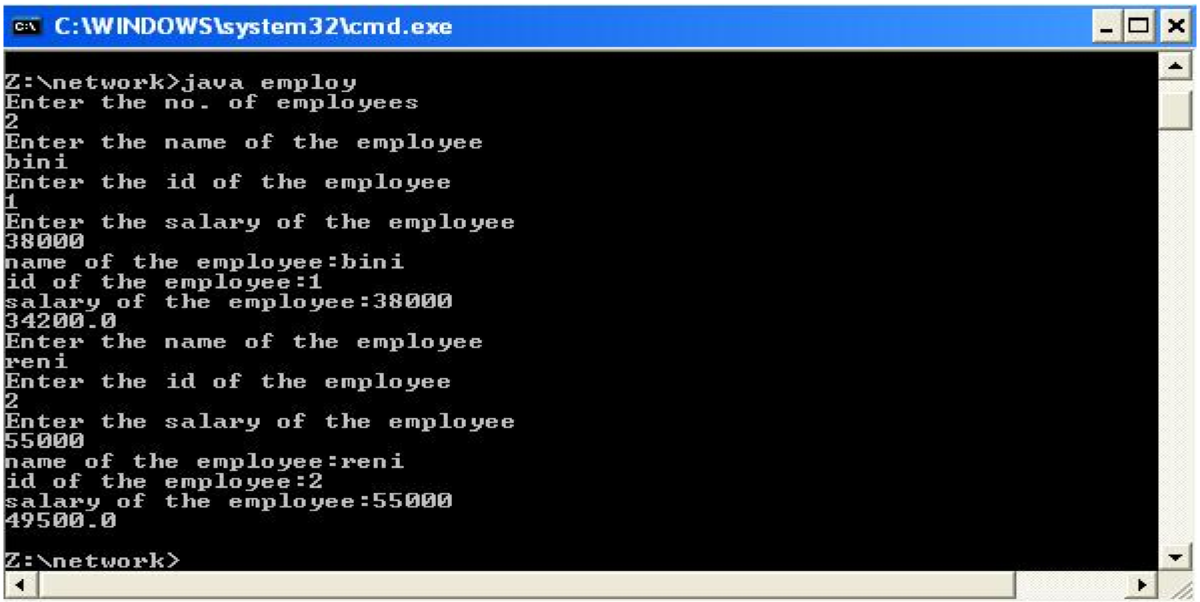
OUTPUT



A view of NS2 Simulation Output

RESULT

The Network Simulator NS2 is familiarised


C:\WINDOWS\system32\cmd.exe

```
Z:\network>java employ
Enter the no. of employees
2
Enter the name of the employee
bini
Enter the id of the employee
1
Enter the salary of the employee
38000
name of the employee:bini
id of the employee:1
salary of the employee:38000
34200.0
Enter the name of the employee
reni
Enter the id of the employee
2
Enter the salary of the employee
55000
name of the employee:reni
id of the employee:2
salary of the employee:55000
49500.0

Z:\network>
```