**Computer Networks** **Kunal Sachdeva**

**4th Semester CSE NITRR**

**19115045**

# Term Paper

# Network Simulation

## 1. Abstract

In this Paper, I have discussed about Network Simulation. Explanation of NS2, key features of NS2 and its components. Explanation and steps to execute its installation, necessary steps required along with the commands required to execute it and discussed related to the implementation of creating links between the source and destination with an example. I have implemented a code for Simulation of LAN and LAN topologies.

## 2. Introduction

Simulation is the process of learning by doing. Whenever there is something new in the world, we try to analyse it first by examining it and in the process get to learn a lot of things. This entire course is called as Simulation. Most of the commercial simulators are GUI driven, while some network simulators are CLI driven. The network model / configuration describes the network (nodes, routers, switches, links) and the events (data transmissions, packet error etc.).

Output results would include network level metrics, link metrics, device metrics etc. Further, drill down in terms of simulations trace files would also be available. Trace files log every packet, every event that occurred in the simulation and are used for analysis. Most network simulators use discrete event simulation, in which a list of pending “events” is stored, and those events are processed in order, with some events triggering future events—such as the event of the arrival of a packet at one node triggering the event of the arrival of that packet at a downstream node.

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 UPD, traffic source behaviour such as FTP, Telnet, Web, CBR and VBR, router queue 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.

**3. Features of NS2**

Below are some of the key features of NS2:

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

*NS Components:*

- Ns, the simulator itself
- Nam, the network animator
  - Visualize ns (or other) output
  - Nam editor: GUI interface to generate ns scripts
    - Since we only run ns2 in remote Unix server, we will not introduce Nam usage in this class
  - It is not essential to simulation and analysis
- Pre-processing:
  - Traffic and topology generators (use Tcl to write)
- Post-processing:
  - Simple trace analysis, often in Awk, Perl, or Tcl
  - You can also use grep (under linux), or C/java

---

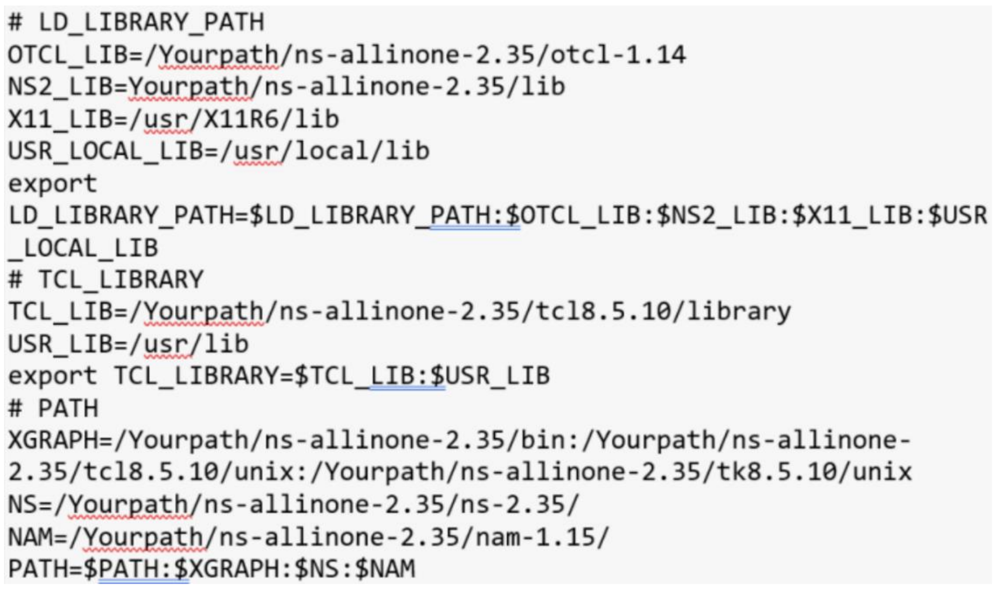
**4. Installation of NS2:**

1. Update and install prerequisites packages using these commands:
   ```
   $sudo apt-get update
   $sudo apt-get dist-upgrade
   $sudo apt-get update
   $sudo apt-get gcc
   $sudo apt-get install build-essential autoconf automake
   $sudo apt-get install tcl8.5-dev tk8.5-dev
   $sudo apt-get install perl xgraph libxt-dev libx11-dev libxmu-dev
   ```

2. Download ns-allinone-2.35.tar.gz.
3. Select the file and click the right mouse button and choose option extract here.
4. After extracting type on the command prompt.
   ```
   $cd ns-allinone-2.35
   ```

5. Run Command ./install see in the image.
6. Step-6 Set the path of the NS-2 for global access.
   Use command
   ```
   $sudo gedit ~/.bashrc
   ```
   It will open a file .bashrc
   Set the path

7. When you set the path, save the file and restart the system and type the command “ns”. The symbol % will appear that means NS-2.35 is correctly installed.



```
# LD_LIBRARY_PATH
OTCL_LIB=/Yourpath/ns-allinone-2.35/otcl-1.14
NS2_LIB=Yourpath/ns-allinone-2.35/lib
X11_LIB=/usr/X11R6/lib
USR_LOCAL_LIB=/usr/local/lib
export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OTCL_LIB:$NS2_LIB:$X11_LIB:$USR
_LOCAL_LIB
# TCL_LIBRARY
TCL_LIB=/Yourpath/ns-allinone-2.35/tcl8.5.10/library
USR_LIB=/usr/lib
export TCL_LIBRARY=$TCL_LIB:$USR_LIB
# PATH
XGRAPH=/Yourpath/ns-allinone-2.35/bin:/Yourpath/ns-allinone-
2.35/tcl8.5.10/unix:/Yourpath/ns-allinone-2.35/tk8.5.10/unix
NS=/Yourpath/ns-allinone-2.35/ns-2.35/
NAM=/Yourpath/ns-allinone-2.35/nam-1.15/
PATH=$PATH:$XGRAPH:$NS:$NAM
```

## 5. Creating Links between the Source and Destination

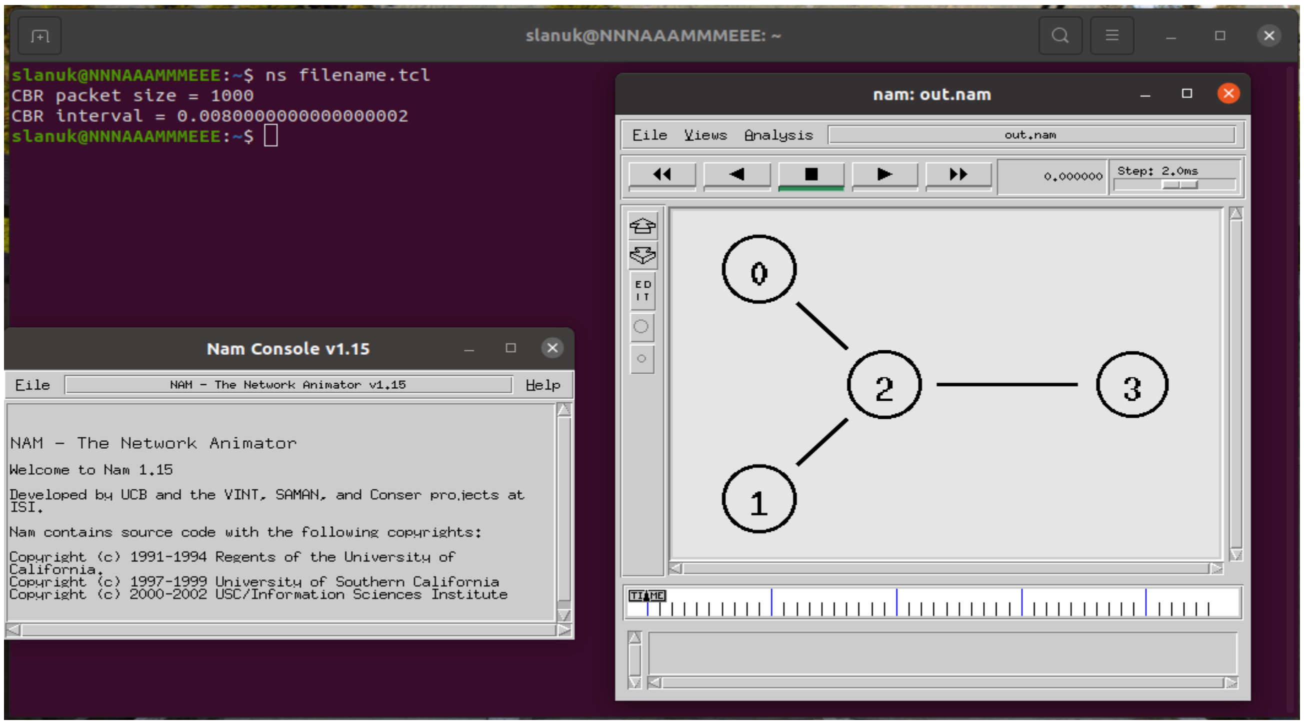
Below is the implementation of creating links between the source and destination:

*Code:*

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

# Define different colors
# for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red

# 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 NAM trace file
        close $nf
        # Execute NAM on the trace file
        exec nam out.nam &
        exit 0
}

# Create four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

# Create links between the nodes
$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

# Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n2 $n3 10

# Give node position (for NAM)
$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

# Monitor the queue for link (n2-n3). (for NAM)
$ns duplex-link-op $n2 $n3 queuePos 0.5


# Setup a TCP connection
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

# Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP


# Setup a UDP connection
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

# Setup a CBR over UDP connection
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


# Schedule events for the CBR and FTP agents
$ns at 0.1 “$cbr start”
$ns at 1.0 “$ftp start”
$ns at 4.0 “$ftp stop”
$ns at 4.5 “$cbr stop”

# Detach tcp and sink agents
# (not really necessary)
$ns at 4.5 “$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink”

# Call the finish procedure after
# 5 seconds of simulation time
$ns at 5.0 “finish”

# Print CBR packet size and interval
puts “CBR packet size = [$cbr set packet_size_]”
puts “CBR interval = [$cbr set interval_]”

# Run the simulation
$ns run
```

*Output:*



```
slanuk@NNNAAAMMMEEE:~$ ns filename.tcl
CBR packet size = 1000
CBR interval = 0.0080000000000000002
slanuk@NNNAAAMMMEEE:~$
```

```
NAM - The Network Animator
Welcome to Nam 1.15
Developed by UCB and the VINT, SAMAN, and Conser projects at
ISI.
Nam contains source code with the following copyrights:
Copyright (c) 1991-1994 Regents of the University of
California.
Copyright (c) 1997-1999 University of Southern California
Copyright (c) 2000-2002 USC/Information Sciences Institute
```

Links between the Source and Destination

**6 .Simulating LAN and LAN topologies**

*Instructions for execution:*

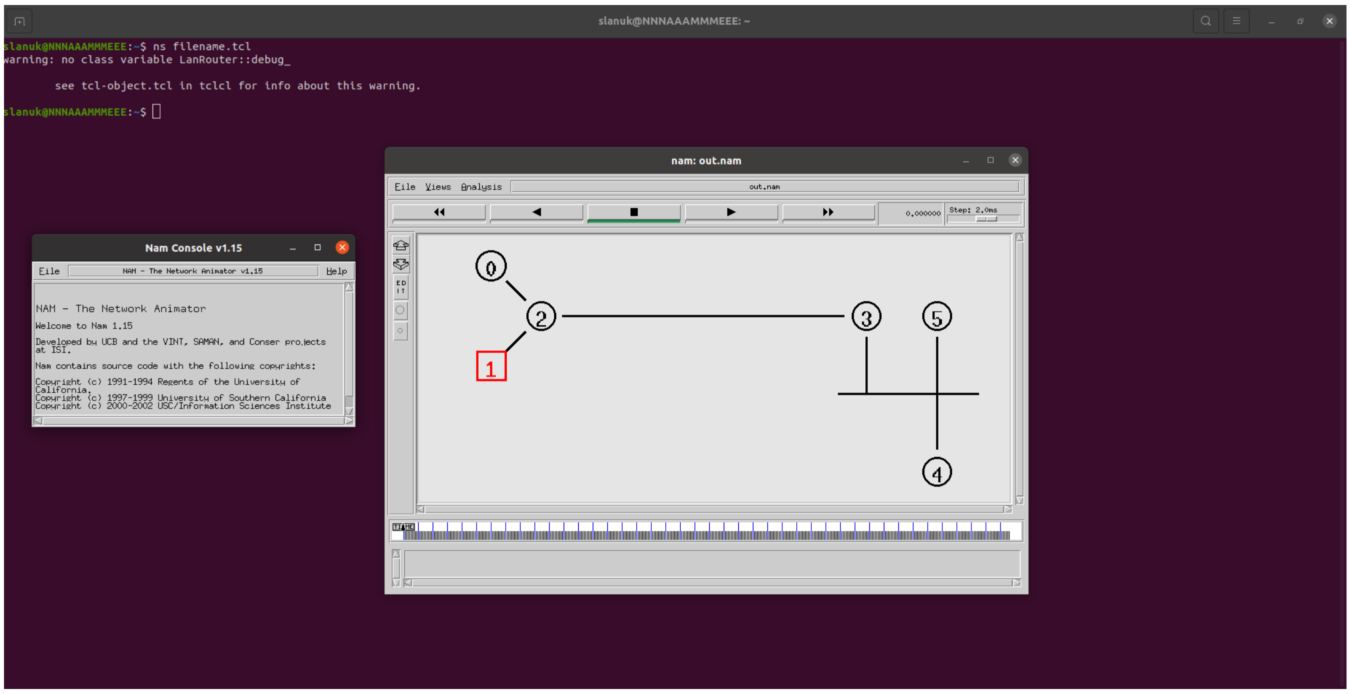
- To analyse the given problem you have to write a Tcl script and simulate with ns2
- Begin by specifying the trace files and the nam files to be created
- Define a finish procedure
- Determine and create the nodes that will be used to create the topology. Here in our experiment we are selecting 6 nodes namely 0, 1, 2, 3, 4, 5
- Create the links to connect the nodes
- Set up the LAN by specifying the nodes, and assign values for bandwidth, delay, queue type and channel to it
- Optionally you can position and orient the nodes and links to view a nice video output with Nam
- Set up the TCP and/or UDP connection(s) and the FTP/CBR (or any other application) that will run over it
- Schedule the different events like simulation start and stop, data transmission start and stop
- Call the finish procedure and mention the time at what time your simulation will end
- Execute the script with ns

*Code:*

```
#lan.tcl
#Lan simulation
set ns [new Simulator]

#define color for data flows
$ns color 1 Blue
$ns color 2 Red

#open tracefiles
set tracefile1 [open out.tr w]
set winfile [open winfile w]
$ns trace-all $tracefile1

#open nam file
set namfile [open out.nam w]
$ns namtrace-all $namfile

#define the finish procedure
proc finish {} {
        global ns tracefile1 namfile
        $ns flush-trace
        close $tracefile1
        close $namfile
        exec nam out.nam &
        exit 0
}

#create six 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]
$n1 color Red
$n1 shape box

#create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns simplex-link $n2 $n3 0.3Mb 100ms DropTail
$ns simplex-link $n3 $n2 0.3Mb 100ms DropTail
set lan [$ns newLan “$n3 $n4 $n5” 0.5Mb 40ms LL Queue/DropTail MAC/Csma/Cd Channel]

#Give node position
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns simplex-link-op $n2 $n3 orient right
$ns simplex-link-op $n3 $n2 orient left

#set queue size of link(n2-n3) to 20
$ns queue-limit $n2 $n3 20

#setup TCP connection
set tcp [new Agent/TCP/Newreno]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink/DelAck]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
$tcp set packet_size_ 552

#set ftp over tcp connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp

#setup a UDP connection
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n5 $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 type_ CBR
$cbr set packet_size_ 1000
```

```
$cbr set rate_ 0.01Mb
$cbr set random_ false

#scheduling the events
$ns at 0.1 “$cbr start”
$ns at 1.0 “$ftp start”
$ns at 124.0 “$ftp stop”
$ns at 125.5 “$cbr stop”
proc plotWindow {tcpSource file} {
        global ns
        set time 0.1
        set now [$ns now]
        set cwnd [$tcpSource set cwnd_]
        puts $file “$now $cwnd”
        $ns at [expr $now+$time] “plotWindow $tcpSource $file”
}

$ns at 0.1 “plotWindow $tcp $winfile”
$ns at 125.0 “finish”
$ns run
```

*Output:*



```
slanuk@NNNAAAMMMEEE:~$ ns filename.tcl
warning: no class variable LanRouter::debug_

        see tcl-object.tcl in tclcl for info about this warning.

slanuk@NNNAAAMMMEEE:~$ 
```

```
NAM - The Network Animator

Welcome to Nam 1.15

Developed by UCB and the VINT, SAMAN, and Conser projects
at ISI.

Nam contains source code with the following copyrights:

Copyright (c) 1991-1994 Regents of the University of
California.
Copyright (c) 1997-1999 University of Southern California
Copyright (c) 2000-2002 USC/Information Sciences Institute
```

LAN Simulation

## References

- Class Provided material for CN LABs
- **https://www.geeksforgeeks.org/**