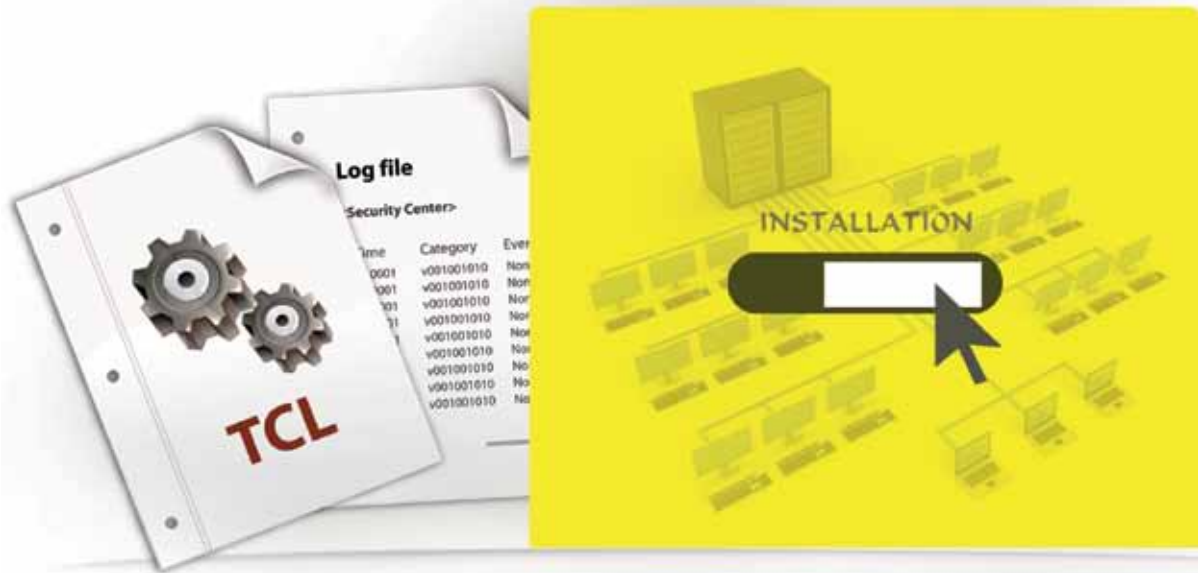


Simulate Your Network with NS2

Learn about Network Simulator-2, its installation and the execution of simple TCL files of network architecture.



Network Simulator-2 (NS2) is a discrete event network simulator that plays an important role in network research and development. It is one of the most widely used open source network simulators. It has been enhanced over the years, leading to a complete package consisting of various modules for functions such as routing, the transport layer protocol and applications.

To observe network performance, an easy scripting language can be used, by which you can configure the network as per your architecture and observe the results to check the correctness of your configuration.

NS2 plays the role of both emulator and simulator. The latter contains three types of discrete event schedulers: list, heap and hash-based calendar. NS2 provides default implementations for network nodes, links between nodes, routing algorithms, some transport level protocols (especially UDP and TCP) and some traffic generators. It also contains some useful utilities like the TCL debugger, simulation scenario generator and simulation topology generator. The TCL debugger is used to debug TCL scripts, which becomes necessary if one is using large scripts to control a simulation.

It is also possible to use NS2 as an emulator, which is currently supported by only FreeBSD. The NS2 emulator can be used to connect the tool to a live network. The

NS2 emulator works on two modes, i.e., the protocol mode and the opaque mode. In the protocol mode, the emulator interprets received traffic, whereas in the opaque mode, the received data is not interpreted.

An insight into the architecture of NS2

NS2 is primarily designed on two languages: C++ and Object-oriented Tool Command Language (OTCL). C++ is used for defining the internals of NS2 while OTCL is used to control the simulation as well as to schedule discrete events. C++ and OTCL are linked together using TCLCL. After the linking of C++ member variables to OTCL object variables by using the bind member function, C++ variables can be modified through OTCL directly. The main drawbacks of this approach are that the user has to know C++ as well as OTCL, and the debugging of simulations becomes more difficult.

After simulation, NS2 outputs either text-based or animation-based simulation results. To interpret these results graphically and interactively, tools such as NAM and XGraph are used, which are explained in the next section. To analyse some particular behaviour of the network, extract a relevant subset of the text-based data and transform it into a more understandable presentation. The basic architecture is shown in Figure 1, pictorially:

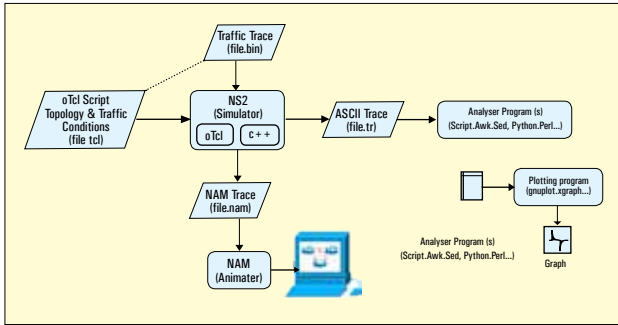


Figure 1: Network Simulator 2 Architecture

Supporting tools with NS2

Let's explore the supporting tools that come with the NS2 installation, which help set up your environment and run your simulations.

NAM (Network AniMator): NAM is a TCL-based animation tool for viewing network simulation traces and real traces of packet data. To use NAM, you have to first generate a trace file that contains topological information like nodes, links and packet traces.

Once the trace file is generated, NAM will read it, create a topology, pop up a window, do the layout if necessary, and then pause at the time of the first packet in the trace file. NAM provides control over many aspects of animation through its user interface, and does animation using the following building blocks: node, link, queue, packet, agent and monitor. The official NAM user manual can be found at <http://www.isi.edu/nsnam/nam/>

XGraph: XGraph is a plotting program that is used to create graphic representations of simulation results. It is important because it allows some basic animation of data sets. The animation only pages through data sets in the order in which they are loaded. It is quite crude, but useful if all the data sets are in one file in the time order, and are put out at uniform intervals. Also, the code will take derivatives of your data numerically and display these in a new XGraph window.

NS2 functionalities

Wired world

Routing: distance vector (DV), link state (LS), and multicast

Transport protocols: TCP, UDP, RTP and SCTP

Traffic sources: WEB, FTP, telnet, CBR, and Stochastic

Queuing disciplines: Drop-Tail, RED, FQ, SFQ, and DRR

QoS: IntServ and Diffserv emulation

Wireless

Ad hoc routing (AODV, DSDV) and mobile IP

Directed diffusion, and sensor-MAC

Installing NS2 on Ubuntu

- Download the latest release of the ns-allinone package from the official NS2 weblink <http://www.isi.edu/nsnam/ns/ns-build.html>

- Open the terminal and change the working directory to where you downloaded the ns-allinone package.
- Now untar (uncompress) the package using the following command:

```
tar xzvf filename
```

Change the directory to *ns-allinone2.29*, run the installation script using the following command and wait until the installation is successfully completed:

```
./install
```

- After successfully installing the NS2 package, configure the *.bashrc* file, which is present in the following path:

```
home/ns2username
```

Edit the *.bashrc* file using the following command:

```
gedit .bashrc
```

Set the following path in the last line of *.bash*

```
# export PATH="$PATH:/home/ns2username/ns-allinone-2.29/bin:/home/ns2username/ns-allinone-2.29/tcl 8.4.11/unix:/home/ns2username/ns-allinone-2.29/tk8.4.11/unix"
export LD_LIBRARY_PATH="/home/ns2username/ns-allinone-2.29/otcl-1.11,/home/ns2username/ns-allinone-2.29/lib"
export TCL_LIBRARY="/home/ns2username/ns-allinone-2.29/tcl8.4.11/library"
```

- If the path set is correct, open the new terminal and run the following in the home directory:

```
ns/
```

If you see a % sign displayed in the terminal, congratulations, you have installed NS2 successfully!

- Now to validate the installation, run the following command (optional):

```
cd /ns-allinone-2.29/ns-2.29 / ./validate
```

Execution of a simple TCL file in NS2

In this section, let's cover the execution of TCL files for a simple network architecture.

```
set ns [new Simulator] //This creates an NS2 simulator object
```

```
set tracefile [open out.tr w]
```

```
$ns trace-all $tracefile //Open trace file
```

```

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

proc finish {}
{
    global ns tracefile nf
    $ns flush-trace
    close $nf
    close $tracefile
    exec nam out.nam &
    exit 0
}
//'finish' procedure

set n0 [$ns node]
set n1 [$ns node]
$ns simplex-link $n0 $n1 1Mb 10ms DropTail // Create your
topology
    - set n0 nodes...
    - make the link between node0 and node1

set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $tcp0
set tcpsink0 [new Agent/TCPSink]
$ns attach-agent $n1 $tcpsink0
$ns connect $tcp0 $tcpsink0 // Create your agents
-the nodes will follow the TCP at CBR(Constant Bit Rate)

$ns at 1.0 "$cbr start"
$ns at 3.0 "finish" //Scheduling Events
- $ns at 1.0 start
  and at 3.0 finish

$ns run //starts the simulation

```

This is a basic example of a .TCL file where you create two nodes and observe the transfer of packets according to the Transmission Control Protocol (TCP) at Constant Bit Rate (CBR), which is a traffic generator. In this example, we have used the TCP but other protocols and traffic sources can be used depending on your network architecture.

Execution of the .TCL file in NS2

.TCL files contain the code of network simulation and can be executed by following the steps given below:

- Once you have installed the NS2 successfully, open a terminal and run the following:

```
/ns filename.TCL.
```

- Once you run that command, .tr and .nam files will be created in the same directory that contains the .TCL file.

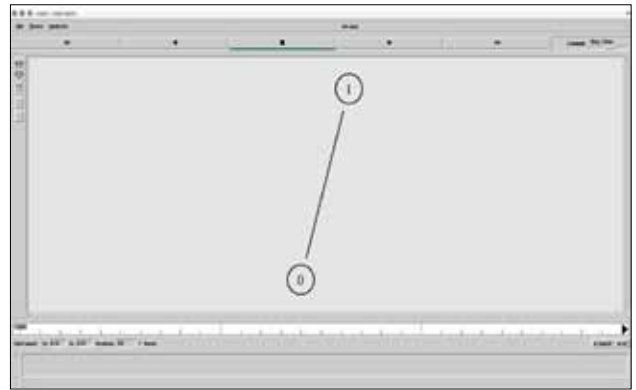


Figure 2: Execution of a simple TCL file


- To interpret the results in the form of animation, you need to just run the command given below:

```
/nam filename.nam
```

- And to see the results in numerical form, you need to open the .tr file in the editor.

```
/gedit filename.tr
```

The main aim of this article was to give readers an insight into the architecture and working of NS2 and how it could be used to simulate complex networking problems with simple programming syntax.

As the next step, start with designing a different network architecture consisting of three nodes, write a simple script using the reference given above and check its execution. This would give you a good idea about writing and executing scripts in NS2. Check the 'References' section for more details about simulating your network using Network Simulator 2. **END** 

References

- [1] Introduction to Network Simulator 2, Springer Publications.
- [2] The NS2 project page, http://nsnam.isi.edu/nsnam/index.php/User_Information
- [3] For more examples, visit, www.nsnam.com

By: Naman Jain and Tejaswi Agarwal

Naman Jain is keenly interested in open source programming and has a good command over Linux systems administration. His research interests lie in the areas of computer networks, simulations, and cloud computing. You can contact him at naman.jain2010@vit.ac.in

Tejaswi Agarwal is a FOSS enthusiast, who is passionate about compute power utilisation, run time and memory utilisation of algorithms. Computer architecture, parallel programming and performance engineering are some of his research areas. He can be contacted at tejaswi.agarwal2010@vit.ac.in