Assignment No.1

**Aim** Install and Configure Network Simulator tool such as Network Simulator 2 or NetSim or QualNet and study its components and eco system.

**Theory:**

Simulation is the process of learning by doing. Whenever there is something new in the world, we try to analyze it first by examining it and in the process get to learn a lot of things. This entire course is called as Simulation. Correlating to this process, in order to understand all the complexities one need to model the entire role-play in form of computer simulation, the need is to build artificial objects and assign them roles dynamically.

Computer simulation is the designing of a theoretical physical system on a digital computer with emphasis on model designing, execution and analysis. After creation of the mathematical model the most important step is to create a computer program for updating the state and event variables through time (by time slicing or event scheduling). If this simulation is carried out successively in parallel computers, it is called Parallel *or* Distributed simulation.

The NS simulator has been distributed as free and open source software. The open source nature of the NS simulator has allowed thousands of students, engineers and researchers to contribute scripts and patches. The development of NS-2 has been funded by the DARPA VINT (Virtual Inter Network Testbed) project from 1997-2000, by DARPA SAMAN (Simulation Augmented by Measurement and Analysis for Networks) and NSF CONSER (Collaborative Simulation for Education and Research) from 2000-2004.The simulator NS-2 has had a smooth transition from the NS-1 version, which had a similar architecture; NS-2 was designed to be backward compatible with scripts written in NS-1. In contrast, the gap between the architectures of NS-2 and NS-3 is very large and NS-3 is not backward compatible. This suggests that NS-2 will remain for many years a useful tool, with an advantage of having huge amount of accessible open source software that had been developed during the last decade and not yet ported to NS-3.The open source nature of NS-2 and the community-based development practices of NS-2which were one of the main sources for its rapid development are expected to continue with theNS-3 version.

The following figure shows the basic architecture of NS2. NS2 provides users with executable command ns which take on input argument, the name of a Tcl simulation scripting file. Users are feeding the name of a Tcl simulation script (which sets up a simulation) as an input argument of an NS2 executable command ns. In most cases, a simulation trace file is created, and is used to plot graph and/or to create animation. NS2 consists of two key languages: C++ and Object-oriented Tool Command Language (OTcl). While the C++ defines the internal mechanism (i.e., a backend) of the simulation objects, the OTcl sets up simulation by assembling and configuring the objects as

well as scheduling discrete events (i.e., a frontend). The C++ and the OTcl are linked together using TclCL. Mapped to a C++ object, variables in the OTcl domains are sometimes referred to as handles. Conceptually, a handle (e.g., n as a Node handle) is just a string (e.g.,_o10) in the OTcl domain, and does not contain any functionality. Instead, the functionality (e.g., receiving a packet) is defined in the mapped C++ object (e.g., of class Connector). In the OTcl domain, a handle acts as a frontend which interacts with users and other OTcl objects. It may defines its own procedures and variables to facilitate the interaction. Note that the member procedures and variables in the OTcl domain are called instance procedures (instprocs) and instance variables (instvars), respectively. Before proceeding further, the readers are encouraged to learn C++ and OTcl languages. NS2 provides a large number of built-in C++ objects. It is advisable to use these C++ objects to set up a simulation using a Tcl simulation script. However, advance users may find these objects insufficient. They need to develop their own C++ objects, and use aOTcl configuration interface to put together these objects. After simulation, NS2 outputs either text-based or animation-based simulation results. To interpret these results graphically and interactively, tools such as NAM (Network AniMator) and XGraph are used. To analyze a particular behaviour of the network, users can extract a relevant subset of text-based data and transform it to a more conceivable presentation.

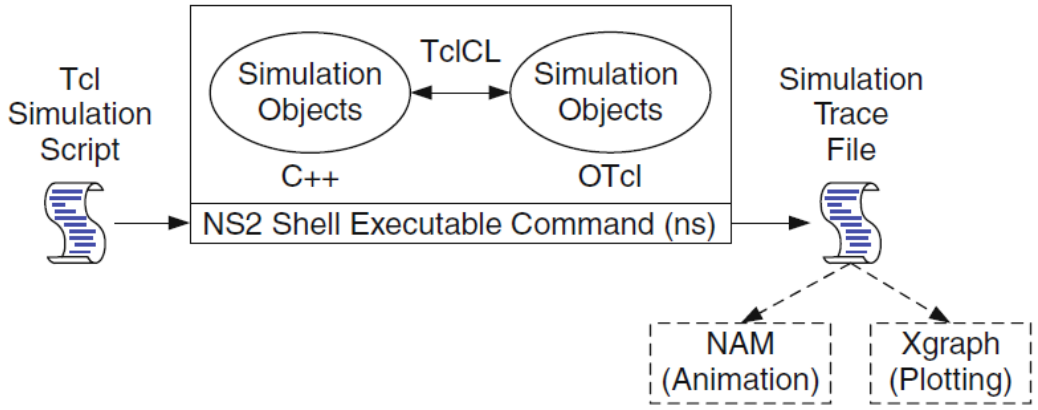


Fig: Basic Architecture of NS

**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 behavior

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.

**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
3. Basic Architecture

**Tcl scripting:**

Tcl is a general purpose scripting language. [Interpreter]

• Tcl runs on most of the platforms such as Unix, Windows, and Mac.

• The strength of Tcl is its simplicity.

• It is not necessary to declare a data type for variable prior to the usage.

**Basics of TCL Syntax:**

command arg1 arg2 arg3

**Hello World!**

putsstdout{Hello, World!} Hello, World!

Variables Command Substitution

set a 5 set len [string length foobar] set b $a set len [expr [string length foobar] + 9]

**Wired TCL Script Components**

Create the event scheduler

Open new files & turn on the tracing

Create the nodes Setup the links

Configure the traffic type (e.g., TCP, UDP, etc)

Set the time of traffic generation (e.g., CBR, FTP)

Terminate the simulation

**NS Simulator Preliminaries.**

1. Initialization and termination aspects of the ns simulator.
2. Definition of network nodes, links, queues and topology.
3. Definition of agents and of applications.
4. The nam visualization tool.
5. Tracing and random variables

**NS2 INSTALLATION ON UBUNTU 16.04**

The following are the steps to install NS-2 on Ubuntu 16.04:

1) Download 'ns-allinone-2.35' from:
http://sourceforge.net/projects/nsnam/files/allinone/ns-allinone-2.35/ns-allinone-2.35.tar.gz/download

2) Copy the downloaded zip file 'ns-allinone-2.35.tar.gz file' to home Folder and extract.

3) A) Type following commands on terminal:

i) sudo apt-get update

ii) sudo apt-get install build-essential autoconfautomake

iii) sudo apt-get install tcl8.5-dev tk8.5-dev

iv) sudo apt-get install perlxgraphlibxt-dev libx11-dev libxmu-dev

3) Now open file ls.h from /home/ns-allinone-2.35/ns-2.35/linkstatein the file go to line no. 137 (void eraseAll() { erase(baseMap::begin(), baseMap::end()); } )
Replace withvoid eraseAll() { **this**->erase(baseMap::begin(), baseMap::end()); }

4) Now change your directory(here i have already extracted the downloaded files to desktop,so my location is desktop) type the following codes in the command window to install NS2.

cd ns-allinone-2.35

./install

The installation procedure will take a few minutes.....

5) After completing the installation type the following command in the command window

gedit ~/.bashrc

6) Now an editor window appears,please copy and paste the following codes **in the end of the text** file (notethat '/home/c05l0202/ns-allinone-2.35/otcl-1.14' in each line in the below code should be replaced withyour location where the 'ns-allinone-2.35.tar.gz'file is extracted)

# LD_LIBRARY_PATH

OTCL_LIB=/home/c05l0202/ns-allinone-2.35/otcl-1.14

NS2_LIB=/home/c05l0202/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=/home/c05l0202/ns-allinone-2.35/tcl8.5.10/library

USR_LIB=/usr/lib
export TCL_LIBRARY=$TCL_LIB:$USR_LIB
# PATH

XGRAPH=/home/c05l0202/ns-allinone-2.35/bin:/home/c05l0202/ns-allinone-2.35/tcl8.5.10/unix:/home/c05l0202/ns-allinone-2.35/tk8.5.10/unix

NS=/home/c05l0202/ns-allinone-2.35/ns-2.35/
NAM=/home/c05l0202/ns-allinone-2.35/nam-1.15/
PATH=$PATH:$XGRAPH:$NS:$NAM

7) Save and close the text editor and then type the following command on the terminal
source ~/.bashrc

8) Close the terminal window and start a new terminal window and now change the directory to ns-2.35and validate ns-2.35 by executing the following command
cd ns-allinone-2.35
cd ns-2.35
./validate
9) If the installation is successful, then you will be able to see % at the command prompt while typing thefollowing command
ns
10) checknam, xgraph too
11) Now type
Exit

**Conclusion:** Thus, we studied, install and configure NS2 on ubuntu.

FAQs:

1. Distinguish emulation and simulation with example.
2. Compare complier and interpreter.
3. List the different simulation tools and compare it with network simulator-2 (NS-2).
4. Why two language (oTCL and C++) used by NS2?
5. What are different advantages and disadvantages of NS2?
6. Draw and explain three kinds of formats for wired networks in NS-2.