**Introduction:**

NS is a discrete event simulator targeted at networking research. NS provides substantial

support for simulation of TCP, routing and multicast protocols over wired and wireless

networks.

A simulation is an imitation of some real thing, state of affairs, or process. The act of

simulating something generally entails representing certain key characteristics or

behaviour of a selected physical or abstract system.

Key issues in simulation include acquisition of valid source information about referent

selection of key characteristics and behaviours, the use of simplifying approximations

and assumptions within the simulation and fieldility and validity of the simulation

outcomes.

NS is a public domain simulator boating a rich set of Internet protocols, including

terrestrial, wireless and satellite networks.

NS began as a variant of the REAL network simulator in 1989 and has evolved

substantially over the past few years. In 1995 NS development was supported by DARPA

through the VINT project at LBL, Xerox PARC, UCB and USC/IS.

NS – 2 is written in C++ and an Object Oriented version of Tcl called OTcl.

NS is constantly maintained and updated by its large user base and a small group of

developers at ISI.

* Main directories
  + bin, ns-2xx, lib, man, include, etc in ns2 home
* ns-2.xx
  + Readme file
  + Makefile, installation file, tutorial, etc
  + Source files related to the protocols
    - All .cpp and .h files related needed for editing
  + Need understanding of interaction among the functions/sub routines
  + Not fully layered like QualNet

***Network Animator – NAM:***

Nam is a Tcl/TK based animation tool for viewing network simulation traces and real

world packet traces. It is mainly intended as a companion animator to the NS simulator.

It supports topology layout, packet level animation, and various data inspection tools.

Nam began at LBL. It has evolved substantially over the past few years. The nam

development effort was an ongoing collaboration with the VINT project.

Currently, it is being developed at ISI as part of the SAMAN and Conser projects.

NS together with its companion, Nam form a very powerful set of tools for teaching

networking concepts.

NS contains all the IP protocols typically covered in undergraduate and most graduate

courses, and many experimental protocols contributed by its ever-expanding users. With

nam, these protocols can **Visualized** as animations.

Below is the screenshot of a nam window where the most important functions are being

explained.



**Installing NS:**

**1. Requirements:**

To build NS we need a computer and a C++ compiler. We develop NS on several kinks

of Unix (FreeBSD, LINUX, SunOS, Solaris), so it installs smoothest there, but it should

run on a Posix- like computer, possibly with some thwacking.

NS also builds and runs under Windows.

Simple scenarios should run on reasonable machine, but very large scenarios benefit from

large amounts of memory.

NS is fairly large. The allinonc package requires about 320MB of disk space to build.

Building NS from pieces can save some disk space.

**2. Downloading And Building NS:**

NS requires a modestly up-to-date installation of Tcl/TK (with header files), and two

additional packages: tclcl and otcl.

TCL (Tools Command Language) is a very powerful but easy to learn dynamic

programming language, suitable for a very wide range of uscs, including web and

desktop applications, networking, administration, testing and many more.

TK is a graphical user interface toolkit. TK is the standard GUI not only for Tcl, but also

for many other dynamic languages, and can produce rich, native applications that run

unchanged across Windows, Mac OS X, Linux and more.

Most OS installations do not come with full Tcl/TK installations or with these other

packages, so we will most likely need to install several packages.

**Windows Specific Instructions:**

NS runs on windows platforms using Cygwin emulation.

Cygwin is a Linux- like environment for Windows. It consists of two parts:

A DLL (cygwin1.dll), which acts as a Linux API emulation layer providing substantial

Linux API functionality.

A Collection of tools, which provide Linux look and feel.

**3. Protocols Supported By NS2:**

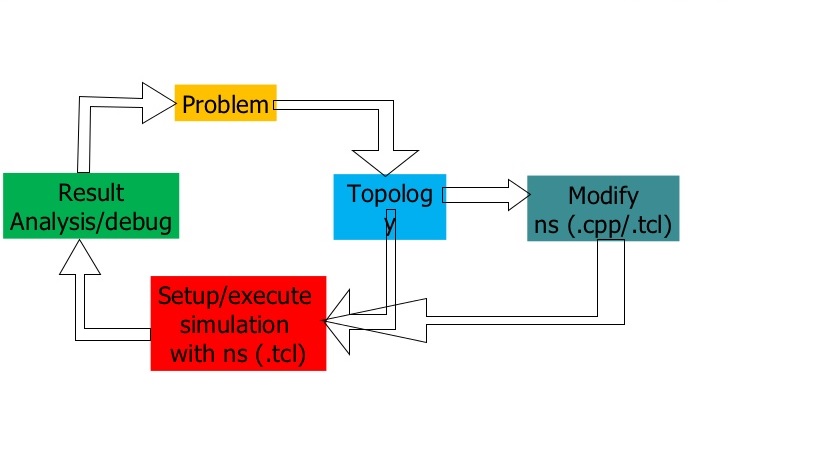
A lot almost all variants of TCP, several forms of multicast, wired networking, several

adhoc routing protocols and propagation models (but not cellular phones), data diffusion,

satellite, and other stuff.

Executing NS2:-

* Create your .tcl script as per your topology
* Run the .tcl file using *ns* command
  + Check which ns2 you are using
  + Create a huge output file (trace file) to analyze
  + Need to understand the file contents
  + Perl scripts are also available to analyze the trace file
* Analyze using *nam*
  + Visual network animator
* Single thread of control
  + No locking or race conditions to worry about



**Functional Diagram of NS2**

**4. Documentation:**

**Core Documentation**

NS Manual (formerly called ―NS Notes and Documentation‖)

The NS Manual (formerly known as NS Notes and Documentation) is the main

source of documentation. It is available in three formats.

**Functions Of NS:**

**Education Uses**

General information about using NS / nam for networking education.

Web index of education scripts.

**Contributed Code and Contributing Code**

Contributed Modules.

How to contribute our NS code.

Research using NS.

Parallel/Distributed NS (PDns) from the COMPASS research group at Georgia

Tech.

**Other Applications**

Topology Generation for large simulations.

Scenario generation in NS.

NS Network Emulation Capability.

.