UNIT-V: Simulator Software

INTRODUCTION TO NS-2

Network Simulator (Version 2), widely known as NS-2, is simply an event-driven simulation tool that has proved useful in studying the dynamic nature of communication networks. NS-2 is use to simulate wired as well as wireless network functions and protocols such as routing algorithm, UDP, TCP. In general, NS-2 provides users with a way of specifying such network protocols and simulating their corresponding behaviors. Due to its flexibility and modular nature, NS-2 has gained stable popularity in the networking research community since its birth in 1989. Ever since, numerous revolutions and revisions have marked the rising maturity of the tool.

In general network simulators attempt to model the actual world networks. We can analyze the results of model network by changing their features which is the principle idea of system modeling. System modeling and their modification are cheaper than complete real implementation. So at low cost we can analyze a wide variety of scenarios.

The Defense Advanced Research Projects Agency (DARPA) supported growth of NS since 1995 through the Virtual Inter Network Tested (VINT). Presently the National Science Foundation (NSF) has tied the ride in development. In the community the group of researchers and developers are continuously working to keep NS-2 tough and versatile.

However, all the details of the network cannot be modeled by network simulator so network simulator is not perfect. However, if model is fine then they will be sufficient to give the researcher a significant insight into the network.

Basic Architecture of NS-2

The basic architecture of NS-2 is shown in Figure 5-1. An executable command ns is provided by NS-2 to the users which takes the name of a Tcl as input argument.

NS-2 consists of two key languages like C++ and Object-oriented Tool Command Language (OTcl). The C++ defines the interior mechanism that is a backend of the simulation objects and the Object-oriented Tool Command Language (OTcl) sets up simulation by assembling and configuring the objects as well as scheduling discrete events that is a frontend. The C++ and the OTcl are connected together using Tcl.

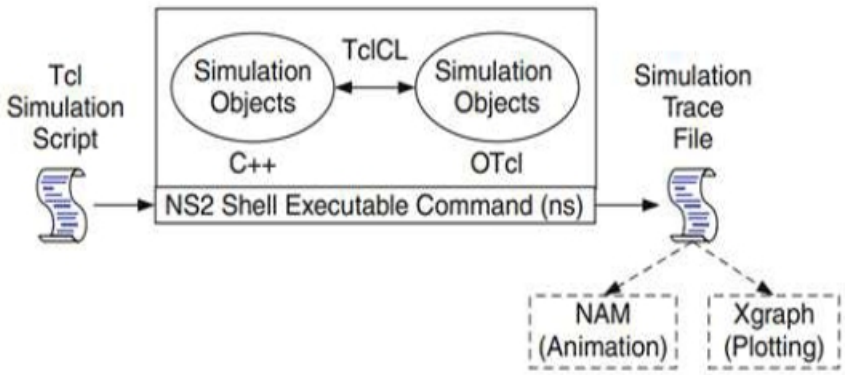

Figure 5-1 basic Architecture of NS-2

These objects are not enough for advance users they need to build up their own C++ objects and an OTcl configuration interface is use to put together these developed objects. NS-2 outputs after simulation is either animation-based or text-based simulation results. NAM (Network Animator) and x graph are used to interpret these results interactively and graphically. To analyze a particular performance of the network, users can extract an associated subset of text-based data and transform it to a more conceivable presentation.

Installation

A free simulation tool is NS-2 which can be runs from vivacious platforms including UNIX (or Linux), Windows, and Mac systems. Source codes of NS-2 are distributed in two forms one are the all-in-one suite and other is the component wise.

After the installation and/or recompilation, an executable file NS is created in the NS-2 home directory. NS-2 can be invoked by executing the following statement from the shell environment >>ns [<file>] [<args>]

Where<file> and <args> are optional input argument.

If no argument is given, the command will bring up an NS-2 environment, where NS-2 waits to take commands from the standard input that is from keyboard line-by-line.

The three key step guideline in defining a simulation scenario in a NS-2

Step 1: Simulation Design

In network simulation the first step is to design the simulation. In this first step users should determine the purposes of simulation, network configuration and assumption, performance measures and kind of expected results.

Step 2: Configuring and Running Simulation

Configuring and Running Simulation implements the design in the first step. It consists of two phases.

- **Network configuration phase:**

In this phase network components like node, UDP and TCP are created and configured according to the simulation design. Also, the events are scheduled to begin at a certain time.

- **Simulation Phase**

Simulation Phase starts the simulation which was configured in the Network Configuration Phase. Simulation clock are maintain in this phase and executes events chronologically. Generally this phase runs until the simulation clock reached a threshold value specified in the Network Configuration Phase. In most of cases it is convenient to define a simulation scenario in a Tcl scripting file (e.g., <file>) and feed the file as an input argument of an NS-2 invocation (e.g., executing "ns <file>").

Step 3: Post Simulation Processing

The main tasks in this step include verifying the integrity of the program and evaluating the performance of the simulated network. While the first task is referred to as debugging and the second one is achieved by properly collecting and compiling simulation results.

NS-2 Descriptor File

The NS-2 descriptor file is defined in a file Make file located in the home directory of NS-2. It contains the details required to recompile and relink NS-2.

The key relevant details are those beginning with the following keywords.

- INCLUDES =: The items behind this keyword are the directory which should be included into the NS-2 environment.
- OBJ_CC = and OBJ_STL =: The items behind these two keywords constitute the entire NS-2 object files. When a new C++ module is developed, its corresponding object file name should be added here.
- NS_TCL_LIB =: The items behind these keywords are the Tcl file of NS-2. Again, when a new OTcl module is developed, its corresponding Tcl file name should be added here.

Software Structure and Mechanism of NS-2

In NS-2 network physical activities are translated to events, events are queued and processed in the order of their scheduled occurrences. Simulation time progress accordingly with the events processed. It can model important network components, traffic models and applications. Typically, it can configure transport layer protocols, routing protocols, interface queues, and also link layer mechanisms. All parts of network is simulated with the help of this software and ultimately the great amount of cost which is required for network construction is save.

After finish of simulation NS-2 present the most details information about network. It provides us a huge trace file in which the record of all the events line by line are available. It really could preserve the things ever happened as records. And we can evaluate the performance of particular stuffs in our network by tracing these records

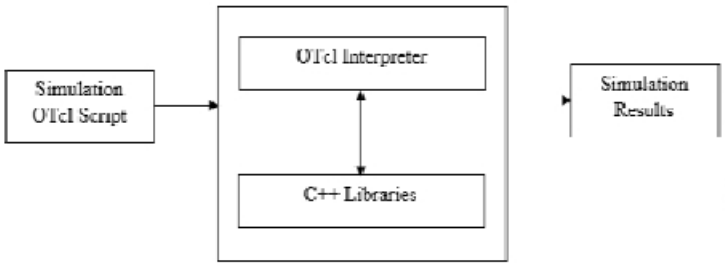

Figure 5-2: Data flow for one time simulation.

The data flow of one time simulation in NS-2 is as shown in Figure 5-2. The OTcl source file is the user input and the OTcl script perform the major work. It initiates an event scheduler and by the use of network objects and plumbing functions, it sets up the network topology. Also it tells the traffic sources about packets transmitting time that is start and stop time of packets transmission through event scheduler. And then, this OTcl script file will be passed to NS-2, in this view, we can treat NS-2 as Object-oriented Tcl (OTcl) script interpreter that has a simulation event scheduler and network component object libraries, and network setup module libraries. And then the detail network construction and traffic simulation will be actually done in NS-2. After a simulation is finished, NS produces one or more text-based output files that contain detailed simulation data, and the data can be used for simulation analysis.

From the NS-2 developer view, Figure 5-3 shows the layered architecture of ns. The event schedulers and most of the network components are implemented in C++ and available to Tcl Script, thus the lowest level of NS-2 is implemented by C++, and the Tcl script level is on top of it to make simulation stuffs much easier to be conducted. Then, upon the Tcl level, we see the overview of the network. That is the simulation scenario. These all things combined as so called NS-2 software.

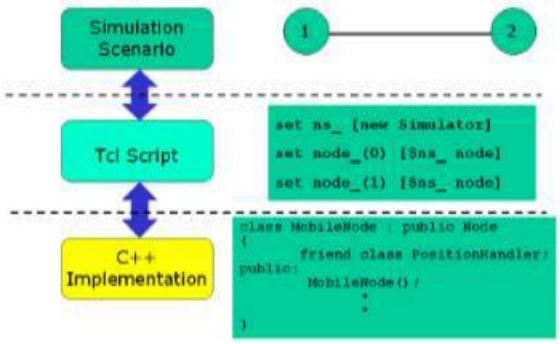

Figure 5-3: Layered structure from the NS-2 developer view.