IJournals: International Journal of Software & Hardware Research in Engineering
ISSN-2347-4890
Volume 3 Issue 9 September, 2015

Present and Future Study on Large Scale Network Simulators

Dr. Vikash Kumar Singh
Head(I/C) Dept. of Computer Science IGNTU Amarkantak (M.P.)
drvksingh76@gmail.com

Devendra Singh Kushwaha
Assistant Professor Faculty of Vocational Educational IGNTU Amarkantak (M.P.)
devendra2904@gmail.com

Shaibya Singh
Assistant Professor Faculty of Vocational Educational IGNTU Amarkantak (M.P.)
shaibyaigntu@gmail.com

Sonal Sharma
Assistant Professor Faculty of Vocational Educational IGNTU Amarkantak (M.P.)
sharmasonal775@gmail.com

Abstract: In the networking, it is very hard to connect networked computers, routers manually. Using network simulator saves lots of money and time and it also remove the complexity. Network simulators are also particularly useful in allowing the network designers to test new networking protocols or to change the existing protocols in a controlled and reproducible manner. In this paper, we present a comprehensive survey on present and future network simulators. In this paper we focused on their features, advantages and disadvantages, and discuss the current and future developments. We hope this survey to be a good reference source for those who feel difficult to find the appropriate network simulators for their research or practical requirements.

Keywords— Network simulation, simulation techniques, NS2, NS3, OPNET, OMNeT++, etc

INTRODUCTION

Simulation is an important tool for engineering design to enable new and improved products, and, better and more efficient production processes. Simulation is widely recognized as an essential tool to analyze networks. Simulation can use for various application fields such as science, engineering and research for various purposes. Typical application areas include physics, chemistry, biology, and human-involved systems in economics, finance or even social science. Simulation has always been an indispensable tool in the design and analysis of telecommunication networks. Due to performance limitations of the majority of simulators, usually network simulations have been done for rather small network models and for short timescales. Simulation is often used to gain insight into the behavior of particular protocols and mechanisms under a variety of network conditions. Here, simulations of small to medium sized networks may be sufficient to gain critical insights into the behavior of the network. Simulation is used to understand the true effect of a new protocol, mechanism, network service, attack, or application when it is widely deployed on a large network such as the Internet. The large-scale nature of the Internet and how to account for it adequately when studying core Internet issues such as topology, routing, traffic, performance, and control.

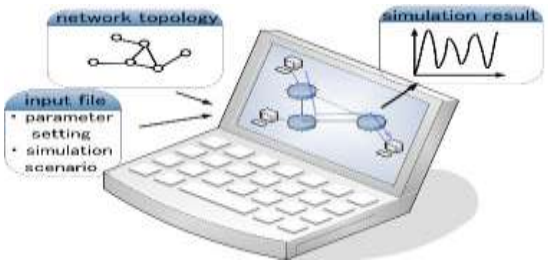

FIG 1- for simulation result

One approach is to use a combination of large-scale network measurement infrastructures and large-scale network simulation engines. A central problem in faithfully simulating large-scale networks is that packet-level descriptions scale poorly; while there exist more abstract models using, say, flows, the errors made in such approximations are not understood. Simulation of large-scale networks remains to be a challenge, although various network simulators are in place. In contrast, many difficult

design problems facing today's network engineers concern the behavior of very large hierarchical multihop networks carrying millions of multiprotocol flows over long timescales. Examples include scalability and stability of routing protocols, packet losses in core routers, of long-lasting transient behavior due to observed self-similarity of traffic patterns. Simulation of such systems would greatly benefit from application of parallel computing technologies, especially now that multiprocessor workstations and servers have become commonly available.

Few advantages of simulation include:

1. A simulation model helps us to gain the knowledge about the improvement of the system.

2. It can help in understanding how the system works

CHALLENGES IN LARGE-SCALE NETWORK SIMULATIONS

As simulation has established itself as an indispensable tool for Internet-related research, it naturally comes to the question: is simulation of an Internet-scale network feasible? As described at the beginning of this chapter, the current Internet has hundreds of millions of nodes, and its traffic still grows vigorously at a high rate. How much computation, memory, and disk space will be required if we want to simulate it? The calculation has been done in [2]. It conservatively estimates the number of hosts, routers, links, and traffic loads in the Internet. A discrete event packet-oriented network simulator (e.g., ns network simulator) is considered. The requirements on memory and disk space are roughly calculated based on measurements from the ns network simulator. It is estimated that simulating an Internet-scale network for a single second generates 2.9*10 POWER 11 packet events, and needs 290,000 seconds to finish if a 1GHz CPU is used; it also requires 2.9*10 POWER 14 bytes of memory, and 1.4 *10 POWER 13 bytes of disk space for logging the simulation results.

DETAILS ON NETWORK SIMULATOR USED FOR SIMULATION

Simulation models can be classified into two broad categories, continuous models and discrete event models, based on how the state variables in a model are updated throughout the simulation. In a continuous simulation model, state changes occur continuously as simulation time elapses; by contrast, a discrete-event model changes its state's only at discrete time points in simulation. It is important to distinguish means of simulation modeling from the properties of real-world systems. Much of the current simulators follow discrete event simulation technique. Hence, here we only focus on this technique

A continuous system like the water in a river can be modeled with both continuous models and discrete-event models; discrete systems such as banks and transportation systems are not restricted to discrete-event simulation models – they can also be characterized by appropriate continuous simulation models. Sometimes, the essential features of a complex system are captured more effectively and efficiently if hybrid simulation models are used. The decision on model selection is contingent on both the properties of the system being modeled and the goal of the simulation study.

Simulation models can also be distinguished by the time management mechanisms used in simulation:

Event-driven simulation (or discrete-event simulation), time-stepped simulation, and real time simulation [3]. In event-driven simulation, simulation events are organized into a list in non descending order of the fire time of each event. The list is often called future event list (FEL). The event at the head of the list (i.e., the one with the earliest fire time) is always processed first. After that, the event is removed from the list, the simulation state is updated, and the simulation time advances to the fire time of the next event. Such a process iterates until the event list becomes empty or the simulation time exceeds the intended simulation length. Event-driven simulation has been applied in simulation of many real systems, such as transportation systems and communication networks, because of its ability to handle asynchrony among objects or entities in a system. In time-stepped simulation, simulation time advances periodically by a constant simulation time

unit, which is often called a time step. Simulation time moves to the next time step only when all the simulation activities associated with the current time step have been finished. Time-stepped simulation is particularly useful when simulating continuous systems whose dynamics can hardly be characterized with discrete events. The last type of simulation models is driven by wall clock time, a computer's hardware clock time during the execution of a simulation. Real time simulation progresses in

synchrony with wall clock time, pacing either in exact real time or in scaled real time; it is mostly suitable for simulation projects in which simulators interact with the real world, such as hardware-in-loop simulation and human-in-loop simulation.

There are some other ways to classify simulations models. For instance, a simulation model can be identified as either static or dynamic based on whether the simulation time advances in the simulation, or either deterministic or stochastic based on whether random processes are used in the simulation. When simulation modelers develop a discrete-event simulation model, there are three choices of simulation modeling paradigm: event scheduling, process-interaction, and activity-scanning [3]. The event scheduling approach centers on events and how system states change after an event is processed. The event scheduling view conforms to the basic nature of discrete-event simulation and thus easy to understand. However, modelers may find it difficult to abstract the behavior of a complex system into events directly. The list of such network simulators include OPNET, NS2, NS3, OMNeT++, REAL, Here we discuss some of the most famous network simulators which follow discrete event simulation such as OPNET, NS2, NS3, OMNeT++[4].

NETWORK SIMULATOR

Network simulator is a name for series of discrete event network simulators. The goal of the ns-3 project is to create an open simulation environment for computer networking research that will be preferred inside the research community:

It should be aligned with the simulation needs of modern networking research.

It should encourage community contribution, peer review, and validation of the software.

The most commonly used in simulation is ns-2 and ns-3.

NS2: NS2 is an open source network simulator. NS2 is mostly used in academia because of its open-source and plenty of components library. In 1996-97, ns version 2 (ns-2) was initiated based on a refactoring by Steve McCanne. Use of Tcl was replaced by MIT's Object Tcl (OTcl), an object-oriented dialect Tcl. The core of ns-2 is also written in C++, but the C++ simulation objects are linked to shadow objects in OTcl and variables can be linked between both language realms. Simulation scripts are written in the OTcl language, an extension of the Tcl scripting language. A simplified user's view of NS2 is shown in figure 2.

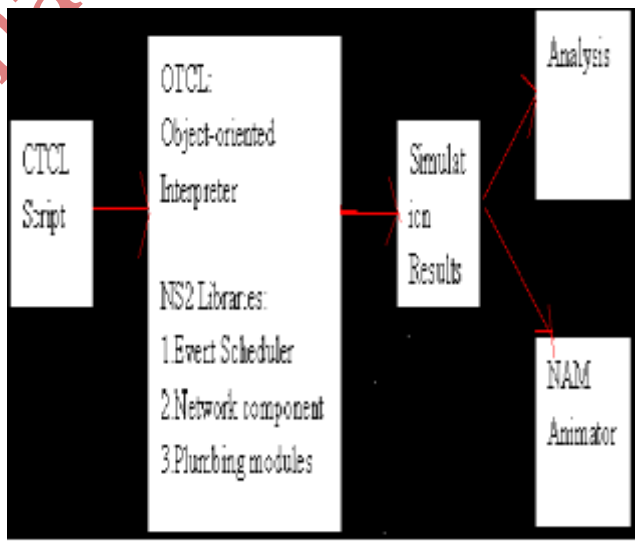

Fig 2: Simplified User's View of NS2 [4]

NS is based on REAL network simulator. NS2 is widely used because of the lot of packages contributed by different no benefit groups that can be widely used for network simulation [1].

NS3: NS3 is designed to replace the current popular NS2. However, NS3 is not an updated version of NS2

since that NS3 is a new simulator and it is not backward-compatible with NS2. A team led by Tom Henderson, George Riley, Sally Floyd, and Sumit Roy, applied for and received funding from the U.S. National Science Foundation (NSF) to build a replacement for ns-2, called ns-3. This team collaborated with the Planete project of INRIA at Sophia Antipolis, with Mathieu Lacage as the software lead, and formed a new open source project.

In the process of developing ns-3, it was decided to completely abandon backward-compatibility with ns-

2. The new simulator would be written from scratch, using the C++ programming language.. NS3 put more emphasis on the documentation works and some specialized people are volunteered to manage different components. NS3 redesigns a lot of mechanisms based on the successful and unsuccessful experiences of NS2 [1].

The main features of Network Simulator 3 (NS3) include:

a) Different software core: The core of NS3 is written in C++ and with Python scripting interface.

b) Software integration: support the incorporation of more open-source networking software and reduce the need to rewrite models for simulation

c) Tracing architecture: NS3 is developing a tracing and statistics gathering framework trying to enable customization of the output without rebuilding the simulation core.

Optimized Network Engineering Tools (OPNET): The OPNET is considered as one of the most famous and commercial network simulators because of its wide uses in the industry and the name of product presented by OPNET Technologies incorporation. It can be flexibly used to study communication networks, devices, protocols, and applications. Because of the fact of being a commercial software provider, OPNET offers relatively much powerful visual or graphical support for the users [4]. OPNET's software environment is specialized for network research and development. OPNET offers powerful visual or graphical support for the users. Object-oriented programming technique is used to create the mapping from the graphical design to the implementation of the real systems. It has the wide variety of possibilities to simulate heterogeneous networks with various protocols. OPNET is based on a mechanism called discrete event system. We can see all the topology configuration and simulation results can be presented very intuitively and visually. Through the GUI the parameters can be adjusted and the experiments can be repeated easily [1]. The parameters can also be adjusted and the experiments can be repeated easily through easy operation through the GUI. OPNET is based on a mechanism called discrete event system which means that the system behavior can simulate by modeling the events in the system in the order of the scenarios the user has set up.

OPNET's detailed features include [4]:

1. Fast discrete event simulation engine

2. Lot of component library with source code

3. Object-oriented modeling

4. Hierarchical modeling environment

5. Scalable wireless simulations support

6. 32-bit and 64-bit graphical user interface

OMNeT++: Like NS2 and NS3, OMNeT++ is also a discrete event simulator. OMNet++ is open source, component-based network simulator with GUI support and widely acknowledged in academia. OMNeT++ has generic and flexible architecture which makes it successful also in other areas like the IT systems, queuing networks, hardware architectures, or even business processes as well. Components are also called modules and are programmed in C++. The components are then assembled into larger components and models by using a high-level language. Its function is similar to that of OTcl in NS2 and Python in NS3. OMNeT++ is a discrete event simulator. It is a component-based architecture and is programmed in C++. The components are then assembled into larger components and models by using a high-level language. OMNeT++ provides GUI support, and due to its modular architecture, the simulation kernel can

be embedded into all kinds of different user s' applications [4].

Main features

Since OMNeT++ is designed to provide a component-based architecture, the models or modules of OMNeT++ are assembled from reusable components. Modules are reusable and can be combined in various ways which is one of the main features of OMNeT++. The OMNeT++ components [OMNET] include:

The main features of OMNeT++ include:

1. Simulation kernel library

2. Compiler for the NED topology description language .

3. Graphical network editor for NED files.

4. GUI for simulation execution links into simulation executable.

5. Graphical output vector plotting tool.

RECENT DEVELOPMENTS AND ITS FUTURE

Recently, about at August 7, 2008, OPNET Technologies announced the addition of two major application performance management capabilities. These capabilities include end-to-end visibility into application performance for organizations using WAN optimization solutions and the ability to capture and analyze NetFlow data.[5]

OPNET recently upgrades its ACE Analyst software includes functionality and it is announced to allow end-user organizations using Riverbed, Cisco, or Juniper WAN optimization appliances to maintain end-to-end visibility into application performance while deploying WAN acceleration solutions. OPNET also provides a module to collect and analyze NetFlow data. Because of the consistent endeavor and operation of OPNET Inc., OPNET is becoming mature and its product maintain a high acknowledge in the industry. Moreover, OPNET always keeps an eye on the most recent user's requirements and keeps improving their product which make it very competitive compared with other commercial network simulators in the near future [4].

NS2: The most recent version of NS2 is NS 2.33 version which was released on Mar 31, 2008. Compared with the previous version, this newest version [NS2] has integrated the most recent extension on new 802.11 models which include the Ilango Purushothaman's infrastructure mode extensions, the 802.11Ext models from a Mercedes-Benz R&D, NA and University of Karlsruhe team, and the dynamic libraries patch andmultirate 802.11 library from Nicola Baldo and Federico Maguolo of the SIGNET group, University of Padova. Generation 3 of NS (NS3) has begun development as of July 1, 2006 and is projected to take four years. It is deemed as the future of NS2.

NS3: NS3 is still in the process and some major challenges still remain for NS3 to solve. The biggest challenge is that NS3 needs participation from the research community. Firstly, the simulation credibility needs to be improved. We know that one of the limitations of simulations, in general, is that it often suffers from lack of credibility. Generally there are four points that are important for NS3 to solve this problem. These are:

a) Hosting NS3 code and scripts for published work

b) Flexible means to configure and record values

c) Support for ported code should make model validation easier and more credible

Secondly, NS3 is intended to replicate the successful mode of NS 2 in which a lot of different organizations contributed to the models and components based on the framework of NS2.

OMNET++: OMNeT++ is being used in the academia as well as in industry. For the future of OMNeT++, we need to note that OMNeT++ is not a network simulator itself. Actually it is currently popular as a network simulation platform in the academia as well as in industry, and build up a large user community. So we have the reason to believe that using OMNeT++ as a basic platform but not an overall single solution. OMNeT++ can have greater development if it could persuade more organizations

to participate in and to contribute.[6] At the Swedish Defence Research Agency (FOI) there is ongoing research, targeting the role of network/web based technologies in M&S, to support defence communities in their work. Our vision comprises an environment supporting the entire M&S-process, including conceptualization, scenario definition, design, development and execution. All these tasks should be maintained by a framework for collaboration, which lets users; developers, analysts, administrators etc, jointly work on a project.

CONCLUSIONS

In this paper we discuss on network simulators and its different types. We also include a study of simulation techniques namely discrete event simulation, parallel discrete event simulation. We discussed on basic types of network simulators that is OPNET, NS2, NS3, and OMNet++. These are very popular for research work and study of simulators. .

REFERENCES

[1] Gayatry Borboruah and Gypsy Nandi. A Study on Large Scale Network Simulators

[2] G. F. Riley and M. H. Ammar. Simulating large networks - how big is big enough? In Proceedings of First International Conference on Grand Challenges for Modeling and Simulation, January 2002.

[3] J. Banks, II J. S. Carson, B. L. Nelson, and D. M. Nicol. Discrete-event system simulation. Prentice Hall, December 2004.

[4] Jianli Pan, Prof. Raj Jain, Project, "A Survey of Network Simulation Tools: current Status and Future Developments", report.

[5] Mrs. Saba Siraj, Mr. Ajay Kumar Gupta, Mrs Rinku-Badgujar, "Network Simulation Tools Survey", International Journal of Advanced Research in Computer and Communication Engineering Vol. 1, Issue 4, June 2012, ISSN : 2278 – 1021

[6]Ahmed Sobeih, Wei-Peng Chen, Jennifer C. Hou, Lu-Chuan Kung, Ning Li, Hyuk Lim, Hung-Ying Tyan, and Honghai Zhang, "J-Sim: A Simulation Environment forWireless Sensor Networks", Proceedings of the 38th Annual Simulation Symposium (ANSS'05), 2005 IEEE