Asian Journal of Information Technology 11 (2): 65-70, 2012
ISSN: 1682-3915
© Medwell Journals, 2012

A Comparative Analysis of Modern Day Network Simulators

Debajyoti Pal
Department of Information Technology,
Camellia Institute of Technology, 700129 Kolkata, India

Abstract: It is a very common practice to use network simulators for testing different network performance parameters before the real-life deployment of such a network. Simulation results across different simulators should be in close agreement with each other and must be dependable. Apart from ns-2, few other recent network simulators have come into existence today and are gaining in more popularity. Thus, it is evident that a qualitative comparison should be made about which simulator should be used. In this study, researchers survey some of the widespread network simulators that are in use today and try to evaluate their performance over certain parameters by setting up identical network simulation scenarios. The results that are obtained from the experiments conducted are analyzed in detail.

Key words: Network simulators, simulation, scalability, network-loss, topology, India

INTRODUCTION

Network simulators help us to reduce the cost, overhead and time that is involved with setting up a real test-bed containing multiple computers, routers and data-links. They allow researchers to implement practical test scenarios that otherwise might be difficult or expensive to emulate in real hardware for example while experimenting with a new routing protocol. In fact they are particularly useful in allowing researchers to test new network protocols or modify an existing protocol in a controlled and reproducible environment. Majority of the network simulators that are available are based upon the discrete event-based simulation type (Fisherman, 1978) in which a list of pending events is stored and those events are processed in order with some events triggering future events-such as the event of arrival of a packet at one node triggering the event of the arrival of that packet at a downstream node.

The first instance where discrete event-based simulation was applied towards simulating a computer network was published by Dupuy *et al*. (1990) and Keshav (1998). ns-2 was a direct successor of all those early efforts. In fact ns-2 is so widely popular that virtually it has become a benchmark for network simulations. The popularity of ns-2 amongst the research fraternity is primarily due to its ability to support a wide variety of protocol models and its support for both wired and wireless networks. However, studies (Xue *et al*., 2007; Henderson *et al*., 2006) reveal that with the growing number of nodes in a given simulated network ns-2 suffers from scalability problems. Under extreme conditions the problem of efficient memory usage and simulation run time become prominent. New research domains like Wireless Sensor Networks (WSN), Wireless Multimedia Sensor Networks (WMSN), grid architectures, etc. require simulating a large number of nodes where in the limitations of ns-2 become prominent. Some modifications have already been incorporated in the latest version of ns-2 (ns-2.35 released on 4th November, 2011) like the inclusion of parallelization. Other network specific features are still due that has ultimately paved its way to the next generation of the ns-2 simulator, namely ns-3 (Henderson *et al*., 2006). In fact, the main goal behind introducing ns-3 is to provide a newer and wider simulation platform that supports the latest networking technologies (both wired as well as wireless) and to improve the overall simulation performance when the number of network nodes becomes large.

Although, simulators model the real world but the way they simulate the realty varies significantly across different simulators. Thus, the only way to judge which simulator is the best is to go for a comparative analysis of the different available simulators which is the basic theme of this study. Apart from ns-2 and ns-3 several other network simulators are available in the market targeted both towards the research fraternity and commercial customers. Prominent among them are OMNeT++ (Varga and Hornig, 2008) Java in Simulation Time (JiST) (Barr *et al*., 2005) GloMoSim (Bagrodia *et al*., 1998) and TOSSIM (Levis *et al*., 2003).

COMPARISON OF DIFFERENT NETWORK SIMULATORS

In this study, researchers review the network simulators under consideration. ns-2 is included in the comparison list because it has been popular among all sections of academics and industry for a very long period of time and hence serves as the basis for the comparison. ns-3, OMNeT++ and JiST are gaining in popularity these days and have their own advantages and disadvantages.

ns-2: ns-2 is a discreet event simulator that supports a wide variety of protocols and can be used for both wired as well as wireless networks. It was built in C++ and provides a simulation interface through OTcl (an object oriented dialect of Tcl). Users describe a network topology by writing Otcl scripts and then the main ns-2 program simulates that topology with specified parameters. However, the problems those are associated with ns-2 while running a large simulation is well known with the main constraints being that of start-up time, memory and CPU usage (Huang and Heidemann, 2001). This problem is of prime importance these days because the simulations are often run for a scalable network size.

ns-3: Like its predecessor, ns-3 also uses C++ for the implementation of the simulation models. However, instead of using OTcl as the scripting language, ns-3 uses Python. It should be noted that ns-3 is not an extension of ns-2. ns-3 integrates architectural concepts and code from GTNetS (Riley, 2003) a simulator that has good scalability characteristics. The design decisions for ns-3 were made at the expense of compatibility. In fact, ns-2 models have to be ported to ns-3 in a manual way. ns-3 is a new software development effort focused on improving upon the core architecture, software integration, models and educational components of ns-2. It provides support for the integration of code by providing standard API's, like Berkley sockets or POSIX threads which can be easily mapped to the simulation.

OMNeT++: In contrast to ns-2 and ns-3 which primarily focuses on simulation of computer networks, OMNeT++ can be used for simulating any general network. In fact it is an extensible, modular, component-based C++ simulation library and framework, primarily for building network simulators. Network is meant in a broader sense that includes wired and wireless communication networks, on-chip networks, queuing networks and so on. Domain-specific functionality such as support for sensor networks, wireless ad-hoc networks, Internet protocols, performance modeling, photonic networks, etc. is provided by model frameworks, developed as independent projects. For example, packages such as OMNeT++ Mobility Framework and Castalia (Pham *et al*., 2007) facilitate the simulation of mobile ad-hoc networks and wireless sensor networks. OMNeT++ offers an Eclipse-based IDE, a graphical runtime environment and a host of other tools. There are extensions for real-time simulation, network emulation, alternative programming languages (Java, C#), database integration, SystemC integration and several other functions.

OMNeT++ simulations use the so-called simple modules. Modules can be connected with each other via gates and combined to form compound modules. Modules communicate through message passing where messages may carry arbitrary data structures. Modules may have parameters that can be used to customize module behavior and/or to parameterize the model's topology.

Like ns-2 and ns-3 OMNeT++ rests upon C++ for the implementation of simple modules. However, the combination of simple modules into compound modules and hence, the setup of the entire network topology takes place in a language calledNetwork Descriptor (NED). NED enables us to assign values to the various network parameters for example the total number of nodes in a network in a dynamic fashion or can later be configured during run-time. In fact OMNeT++ follows a strict object-oriented design philosophy.

JiST: JiST is a high-performance discrete event simulation engine that runs over a standard Java virtual machine. It is a prototype of a new general-purpose approach to building discrete event simulators, called virtual machine-based simulation that unifies the traditional systems and language-based simulator designs.

Simulation in JiST is made up of entities that actually represent the network elements for example the nodes while the simulation events being invoked by some method calls among those entities. Embedding the simulation semantics within the Java language enables JiST to inherit all the properties and libraries of java including the existing compilers. JiST benefits from the automatic garbage collection, type-safety, reflection and many other properties of the Java language. The use of a standard virtual machine provides an efficient, highly-optimized and portable execution platform and allows for important cross-layer optimization between the simulation kernel and running simulation.

Unfortunately, the official development of JiST has stalled and it is no longer maintained by its researcher, Rimon Barr. Still, researchers include it in the study due to the numerous advantages that it provides.

METHODOLOGY AND EXPERIMENTAL TEST BED

In this study, researchers define the experimental test bed that is used to conduct the simulation experiments across the four different simulators under consideration. The results that are obtained are analyzed there after and conclusions are drawn based upon the same.

With the aim to compare the performance of the different simulator tool kits, researchers implemented the same network simulation model in all of them. The simulation does not depend on any particular simulation model tied down to a specific simulator but is generic in nature that is applicable to all the simulators under consideration. Creating the own network model to evaluate the performance comparatives was preferred because researchers did not want to be tied down to the already existing network specific models as they tend to influence the efficiency and complexity of the entire simulation scenario in general.

The simulation models a very basic network which consists of different nodes that are arranged in a square topology as shown in Fig. 1. It should be clearly noted that researchers show 4 nodes for illustration purposes only. In reality while carrying out the experiment, the actual number of nodes is varied from a minimum of 16-2048. The communication process starts at the sending node which generates one packet every second that is broadcasted to all its immediate neighbors. The neighboring nodes in turn broadcast the unprocessed packets after a delay of 1 sec, to their immediate neighbors respectively, thereby flooding the entire network.

The propagation delay of 1 sec is implemented directly by delaying the simulation event's execution. Also researchers assume that the nodes do not implement any specific queuing policy. The probability of packet drop is assumed to be the same across every link. Researchers chose this model due to its simplicity without going into different network specific details.

The simulations were carried on a Intel core-i3-2310M CPU based system having 4 GB RAM, 500 GB hard disk and running dual operating systems (Windows 7 Ultimate and Ubuntu 10.04 LTS). The different observations were done on the latest simulator Versions viz. ns-2.35, ns-3.12.1, OMNeT++ 4.1 and JiST-1.0.6. Java version 6.0.290 provides the run time environment for OMNeT++ and JiST.

The above mentioned network topology along with the constraints are run in the four simulators under consideration for network loss ranging between 0-1. The simulation is carried out for 800 sec in all the cases. The total number of network nodes is increased from 0-1400 gradually.

Figure 2a-c show the graph of the total number of packets that are dropped in the network v/s the total number of network nodes for network loss corresponding to 0.2, 0.5 and 1.0, respectively.

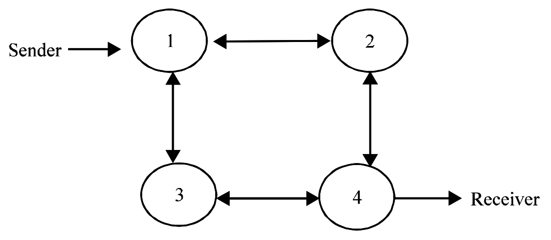

Fig. 1: Different nodes arranged in a square topology

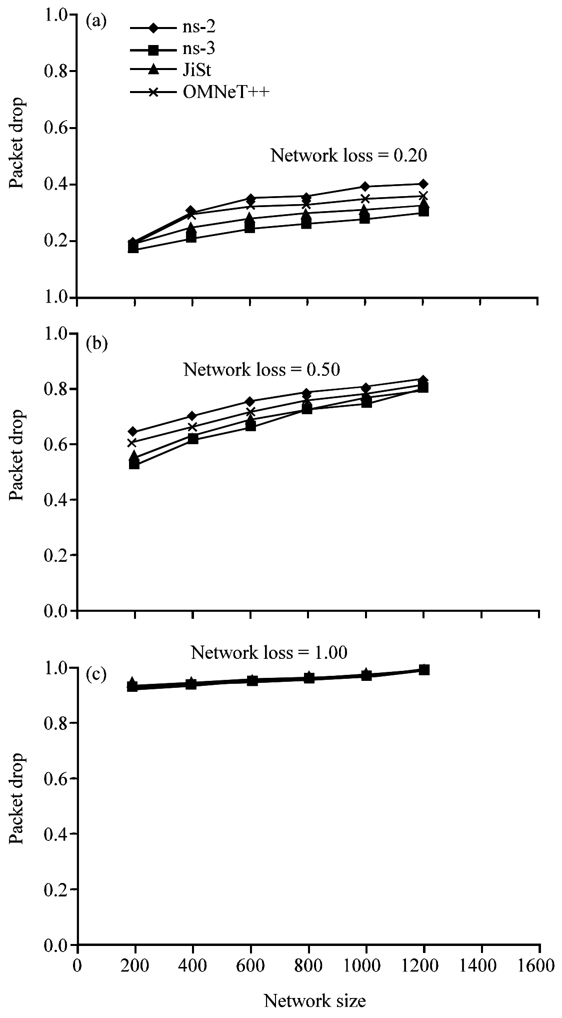

Fig. 2: Total number of packets dropped in the network; a) network loss = 0.20; b) network loss = 0.5; c) network loss = 1.0

From the Fig. 2a-c, it is evident that keeping the network loss constant with increasing the number of network nodes, the total number of packets dropped also increases. This is quite obvious and as seen with increase probability of network loss, the number of packets dropped also increases. In fact, corresponding to network loss = 1.0, almost 100% of the packets are dropped in all the simulation scenarios.

The percentage of packet dropped for different values of network loss is consistent across all the simulation platforms although, ns-2 in particular is more prone to the effect however, still within the safe limits of tolerance. Thus, it can be concluded that the reference simulation scenario that researchers use in all the four cases produce almost equivalent results.

Now, researchers compare the different simulation tools with respect to three very important performance metrics, namely, simulation run-time, memory usage and computation time. For this purpose researchers consider 2 cases:

Case 1: The network loss is kept constant at 0.02 and the total number of nodes is gradually increased from 0-3000.

Case 2: The network loss is varied from 0.0-1.0 keeping the total number of nodes constant at 3000.

In both the cases, the simulation time is fixed to 800 sec. Figure 3 shows the graph for simulation run-time v/s the network size. It is evident that for simulating the same number of nodes the time taken by ns-2 is maximum while that taken by JiST is minimum. For a network size of 3000 nodes while the average time taken by ns-2 is about 440 sec that taken by JiST is about 80 sec. Hence, the simulation run-time for JiST is about 5.5 times faster as compared to ns-2. In fact it is quite surprising that JiST despite being based on a Java platform outperforms all other simulation tools. Primarily, this is due to the JiST architecture. Apart from parallel execution of different entities, JiST also has provisions of various run-time optimizations based upon the analysis of the executed byte-code (Zygmunt, 2004). Thus, the slowness of Java is purely a myth as compared to a C++ platform. The simulation run-time for ns-3 (about 110 sec) is also far better as compared to ns-2. This is primarily due to the abolishment of oTCL/C++ duality in ns-3. The performance of OMNeT++ is more or less similar to that of ns-3. Thus, apart from ns-3 all the other 3 simulators are equally scalable with respect to the simulation run-time.

Figure 4 shows the graph for different values of network loss v/s the simulation run-time keeping the total number of nodes fixed at 3000.

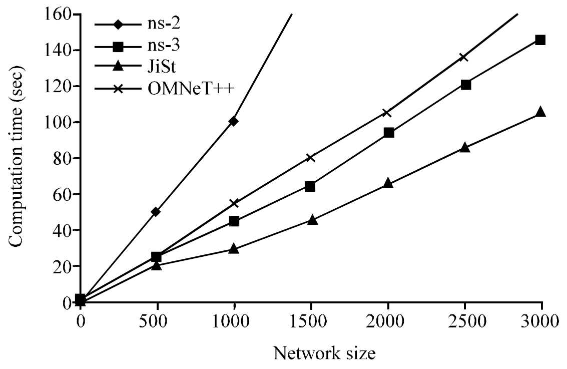

Fig. 3: Graph for simulation run-time v/s the network size

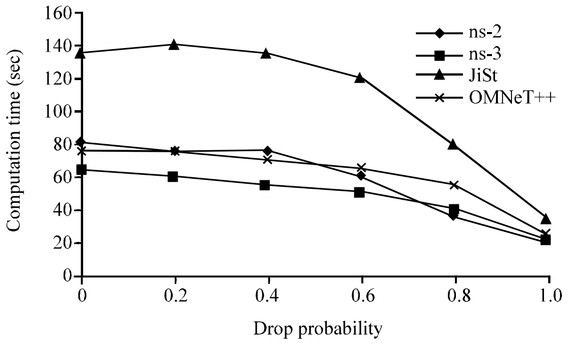

Fig. 4: Graph for different values of network loss v/s the simulation run-time

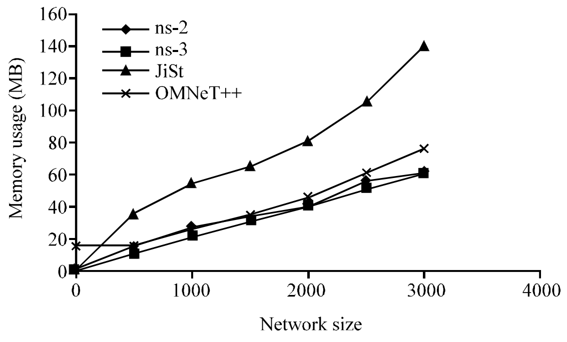

Fig. 5: Graph between network size v/s memory usage

As it is evident from the figure withan increase in network loss, the simulation run-time also decreases. This is quite obvious because as researchers increase the network loss, more and more number of packets are removed from the simulation scenario thereby causing the simulation run-time to drop. For lower order range of network loss, the computation time for ns-2 is maximum while that of JiST is minimum.

Similar to the analysis of the simulation run-time researchers also investigate the memory usage of the different network simulator tools. Figure 5 shows the

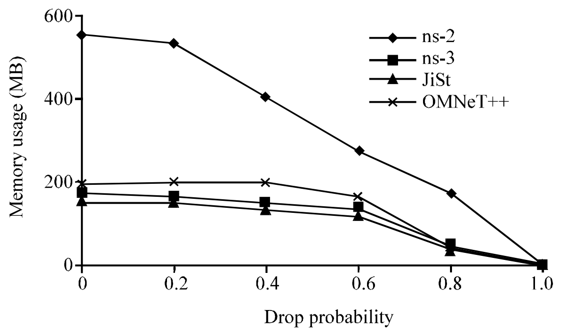

Fig. 6: Graph between network loss and memory usage

graph between network size v/s memory usages keeping the network loss constant. For network size up to 500 the memory usage of the different simulator tools are comparable but thereafter and especially as the number of network nodes becomes large the memory consumption of JiST increases to a great deal. In fact it exhibits the worst performance with respect to memory usage. This is quite surprising due to the fact that Java supports automatic garbage collection mechanism. Memory usage of the remaining simulator tools exhibit more or less the same pattern (increasing in a linear fashion with the number of network nodes) with ns-3 being the most efficient amongst all of them in this regard.

Figure 6 shows the same memory usage for case 2, i.e., a graph between network loss and memory usage keeping the total, number of nodes fixed at 3000.

As expected with an increase in network loss since, more and more packets are being removed from the simulation scenario hence the memory utilization of the simulator tools also keeps on decreasing. Again JiST shows the worst memory usage characteristics among all the others. It should also be noted that even for network loss equal to 100% some residual memory is used by all the simulators. This can be attributed to the simulator process itself that keeps on running in the background at all points of time.

CONCLUSION

A few papers have been found on performance comparison of network simulators. One example is the research provided by Nicol (2003) where the performance of a TCP-based simulation is implemented in ns-2, SSFNet and J-Sim. Similarly, another study (Albeseder and Fuegger, 2005) analyzes the run time performance of ns-2, OMNeT++ and SimPy coming to the same conclusion as researchers. Another example is that of which uses some more network simulators.

After conducting an in depth studies of the different network simulators researchers come to a conclusion that ns-3, OMNeT++ and JiST are all capable of carrying out large scale simulation scenarios. In fact the execution speed of JiST is the best although it is worst with respect to memory utilization. Thus, a tradeoff has to be done between simulation run-time and memory usage and often it depends upon the circumstances and the environment in which researchers are going to run the simulation. Also the official development and maintenance of JiST has been stopped. The overall performance of ns-2 is worst among all.

Thus, researchers can conclude that the answer to the question which simulator is the best is a very difficult one to be answered and is heavily dependent upon the specific use case. However, if memory requirement is not a constraint and scalability is the primary issue then ns-3, OMNeT++ and JiST are the obvious choices.

REFERENCES

Albeseder, D. and M. Fuegger, 2005. Small PC-network simulation a comprehensive performance case study. Research Report 77/2005, TU Wien, Institute of Technology Computer Science, http://www.vmars.tuwien.ac.at/php/pserver/extern/docdetail.php?DID=1847&viewmode=paper&year=2005.

Bagrodia, R., R. Meyer, M. Takai, Y.A. Chen and X. Zeng *et al*., 1998. Parsec: A parallel simulation environment for complex systems. Computer, 31: 77-85.

Barr, R., Z.J. Haas and R.V. Renesse, 2005. JiST: An efficient approach to simulation using virtual machines. J. Software Pract. Experience, 35: 539-576.

Dupuy, A., J. Schwartz, Y. Yemini and D. Bacon, 1990. NEST: A network simulation and prototyping test bed. Communication, 33: 63-74.

Fisherman, G.S., 1978. Principles of Discrete Event Simulation. John Wiley and Sons Inc., New York.

Henderson, T.R., S. Roy, S. Floyd and G.F. Riley, 2006. *ns-3* project goals. Proceedingas of the Workshop on *ns-2*: The IP Network Simulator, October 10, 2006, ACM Press, New York, USA., pp:13-13.

Huang, P. and J. Heidemann, 2001. Minimizing routing state for light-weight network simulation. Proceedings of the International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems Cincinnati, August 15-18, 2001, Ohio, USA.

Keshav, S., 1998. Real: A network simulator. Technical Report, University of California at Berkley, CA, USA.

Levis, P., N. Lee, M. Welsh and D. Culler, 2003. TOSSIM: Accurate and scalable simulation of entire TinyOS applications. Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, November 5-7, 2003, Los Angeles, CA., USA., pp: 126-137.

Nicol, D.M., 2003. Scalability of network simulators revisited. Proceedings of the Communication Networks and Distributed Systems Modelling and Simulation Conference, February, 2003, Orlando, FL, USA.

Pham, H.N., D. Pediaditakis and A. Boulis, 2007. From simulation to real deployments in WSN and back. Proceedings of the 8th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, June 18-21, 2007, Espoo, Finland, pp: 1-6.

Riley, G.F., 2003. Large scale network simulations with GTNetS. Proc. Winter Simul. Conf., 1: 676-684.

Varga, A. and R. Hornig, 2008. An overview of the OMNeT++ simulation environment. Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems, March 3-7, 2008, Marseille, France.

Xue, Y., H.S. Lee, M. Yang, P. Kumarawadu, H.H. Ghenniwa and W. Shen, 2007. Performance evaluation of ns-2 simulator for wireless sensor networks. Proceedingas of the Canadian Conference on Electrical and Computer Engineering, April 22-26, 2007, Canada, pp: 1372-1375.

Zygmunt, R.B., 2004. An efficient, unifying approach to simulation using virtual machines. Ph.D. Thesis, Cornell University, Ithaca, New York, USA.