Analysis and comparison of different network simulators

Vinita Mishra¹, Smita Jangale²

¹Vinita Mishra
Department of Information technology, V.E.S
Institute of technology
University of Mumbai, India
Vinitamayank3@gmail.com

²Smita Jangale
Department of Information technology, V.E.S
Institute of technology
University of Mumbai, India
smitajangale@yahoo.com

ABSTRACT

A network simulator is a software program that imitates the working of computer network. In simulators, the computer network is typically modeled with devices, traffic etc. and the performance are analyzed. Typically, users can then customize the simulator to fulfill their specific analysis needs. This paper gives analysis and comparison study of different wireless network simulators. Simulators in consideration are Glomosim/Qualnet, Omnet++, Ns-2, Opnet, and J-Sim. The information about each of the simulator abilities, their internal structure advantages and disadvantages has been discussed in this paper. This paper analyses situations wherein simulators can be used for their best need. Finally, the paper contains analysis, comparison study and conclusion about making the suitable choice of network simulator based on the researchers needs.

Keywords: a Glomosim/Qualnet, Omnet++, Ns-2, Opnet, J-Sim

1. INTRODUCTION

In communication and computer network research, network simulation is a technique where a program models the behavior of a network either by calculating the interaction between the different network entities (hosts/packets, etc.) using mathematical formulas, or actually capturing and playing back observations from a production network. The behavior of the network and the various applications and services it supports can then be observed in a test lab; various attributes of the environment can also be modified in a controlled manner to assess how the network would behave under different conditions. When a simulation program is used in conjunction with live applications and services in order to observe end-to-end performance to the user desktop, this technique is also referred to as network emulation. In this paper, we have analyzed both open source and commercial simulators. In Section II, we provide a brief overview of network simulators. The main contribution of this paper is a comparison study incorporating five simulation tools, namely, Glomosim/Qualnet, Ns-2, Omnet++, Opnet, and J-Sim. By survey and analysis for all of them, we are able to compare the simulators. The comparison study is discussed in Section III. Actually, this is not the first comparison of network simulators. Many Performance comparisons have been done. We conclude in Section IV with the lessons we learned from the survey and analysis.
2. BACKGROUND OF PROPOSED WORK

There are a wide variety of network simulators, used in different situations as per researcher’s requirements. Simulators may be complex, may be simple. As the matter of fact all the simulators must enable a user to represent a network topology, specifying the nodes on the network, the links between those nodes and the traffic between the nodes. More complex simulators may allow the user to specify everything about the protocols used to handle traffic in a network. Simulators can have graphical applications which allow users to easily visualize the workings of their simulated environment. or text-based applications that may provide a less intuitive interface, but may permit more advanced forms of customization. In this section we have discussed two commercial and three open source simulators.

2.1 GLOMOSIM/QUALNET

GloMosim is an open source, component-based, modular and open architecture environment for discrete-event simulation. It has scalable environment for large wireless and the communication for wire line networks. GloMoSim is used for parallel discrete-event simulation. This capability is provided by Parsec. Commercial tool of GloMosim is QuaNet. It is derived from the GloMoSim that was first released in 2000 by SNT. The main differences between QualNet and GloMoSim are (1) QualNet is based on C++ but GloMoSim is based on the C Parsec language. (2) QualNet is a Commercial Product, but the GloMoSim is distributed under an open source license. (3) QualNet is maintained by SNT, but the GloMoSim is maintained by Parallel Computing. QualNet Architecture is composed of (1) QualNet kernel: The kernel of QualNet is a parallel discrete-event scheduler. It provides the scalability and portability to run hundreds and thousands of nodes with high-fidelity models on a variety of platforms, from laptops and desktops to high performance computing systems. (2) QualNet Model Libraries: QualNet supports a number of model libraries that enables to design networks using various protocol models. QualNet includes the Developer, Multimedia and Enterprise, and Wireless Model Libraries. Additional libraries such as cellular networks, satellite networks, UMTS, WiMAX, sensor networks, military radio networks, and advanced propagation model libraries are also available. (3) QualNet Graphical User Interface: QualNet GUI consists of Architect, Analyzer, Packet Tracer, and File Editor. Architect is a network design and visualization tool. Analyzer is a statistical graphing tool that displays hundreds of metrics collected during simulation of a network scenario. Packet Tracer provides a visual representation of packet trace files generated during the simulation of a network scenario. File editor is a text editing tool. (4) QualNet Command Line Interface: The QualNet command line interface enables a user to run QualNet from a DOS prompt (in Windows) or from a command window (in Linux). QualNet enables users to design new protocol models, optimize new and existing models, Design large wired and wireless networks using pre-configured or user-designed models and analyze the performance of networks. QualNet can support real-time speed to enable software-in-the-loop, network emulation, and hardware-in-the-loop modeling. QualNet can model thousands of nodes. QualNet can run on cluster, multi-core, and multi-processor systems to model large networks with high fidelity. QualNet can communicate with other simulators through HLA module. The QualNet advantages are the complete GUI provided, which is, in our opinion, simpler to use than the OPNET one. It shows a very good scalability, simulation time being reasonable, even on a laptop or desktop computer (tens of thousands of nodes including mobility and high traffic can be simulated in relatively short simulation times). However, some features are not clearly detailed. For example, QualNet allows the modelling of new protocols with the GUI as well as by directly coding the protocol. Nevertheless, it makes not clear that new components can be created.

2.2 OMNeT++

OMNeT++ is an open source, component-based, modular and open architecture environment for discrete event simulation. It is free of charge for academic and non-profit use. Its primary application area is the simulation of communication networks, but its generic and flexible architecture makes it possible to use it in other areas like the simulation of complex IT systems, queuing networks or hardware architectures. OMNeT++ is not a network simulator; however, it is currently gaining popularity as a network simulation platform in the scientific community as well as in industrial ones, and building up a large user community. OMNeT++ runs on Linux, other Unix-like systems and on Windows (XP, Win2K) OS platforms. OMNeT++ is composed of (1) Graphical network editor: A graphical network editor (GNED) to allow graphical topology build, creating files in the Network Description (NED) language (2) Kernel library: A simulation Kernel library contains definitions of objects used for the topology creation (3) Command line interface: Includes Graphical and command line interfaces for simulation execution (4) A model documentation tool for documentation. Two kinds of modules exist: simple modules and compound modules. Compound modules are a set of simple modules. These modules are assembled into larger components and models using a high-level Network
Description (NED) language, allowing a hierarchical organization of the network. These compound modules allow simulation of large-scale networks. Modules communicate by message exchanges. These messages may represent, for instance, packets in communication networks or jobs in queuing networks. They are sent from a simple module to another simple module, either directly to their destination or along a predefined path. Messages are exchanged through gates and connections. OMNeT++ offers the possibility either to modify existing models or to create new object classes that may be derived from basic object classes (module, gate, connection, etc.). Modules types can be stored in separate files. This allows the user to group existing module types and to compose component libraries. OMNeT++ implements a deterministic modeling formalism. But it also handles continuous and discrete stochastic variables that give randomness to the model. OMNeT++ has a separate window for each modules text output. Scheduled messages can be watched as the simulation progresses, Event–by-event execution. Execution animations. Graphical display of simulation results during execution.

2.3 NS-2

It is a public domain event-driven network simulator developed at UC Berkeley. It is available on UNIX, Free BSD and Windows OS platforms. NS-2 is currently a part of the VINT (Virtual Inter-Network Test bed) project [VINT]. NS-2 is designed to simulate small-scale networks. NS-2 is based on three languages: TCL writes simulation script, OTCL defines simulation parameter, C++ implements the schedulers. The outputs produced by NS-2 can be: General format trace files, NAM format trace files, Personalized trace files. NS-2 is cheap, complex scenarios can be easily tested, results can be quickly obtained, supports platform and protocols and popular. Main disadvantage is that real time system too complex to model and scalability issue.

2.4 OPNET

It provides a global environment to model, simulate and evaluate performances of all kinds of wired and wireless communication networks and distributed systems. It is available on Windows 2000, XP, Linux and Solaris platforms. The OPNET environment includes graphical tools for scenarios and models conception, scenarios simulation, data collection and data analysis. A simulation within OPNET is represented by a project including a set of scenarios. This project is created through the project editor also known as the OPNET central interface. All the available functionalities may be accessed from this editor. It provides an access to other editors that propose functions including node and process model creation, building packet formats, and creating filters and parameters. OPNET provides many additional functions including a High-Level Architecture (HLA) module, which allows communication between various simulators. OPNET allows hierarchical modeling by defining a network as a collection of sub-models representing sub-networks or nodes. Modeling is done within the modeling environment, which consists of three domains. (1) The network domain. It defines the communication network topology to simulate. (2) The node domain. The node domain instantiates the nodes defined in the network domain, i.e. elements connected to the network that can send and receive data. (3) The process domain. The process domain describes every module (processors or queues) that is programmable by the user. They execute processes or tasks. The visualization and analysis of results is achieved with the Analysis Tool and the Filter Editor. The Analysis Tool displays the results of a simulation or series of simulations as graphs. Although simulation results can be viewed in the Project Editor. OPNET modeling modularity is an asset to model critical infrastructures. It offers the possibility to define and add new models specific to the infrastructure to simulate. New parameters can also be specified to model actions that might occur within the infrastructure (management, processing, etc.). Infrastructures can be distinguished by the process models (i.e.: the codes or programs that characterize the state of the modules associated to a node or a link). However, this tool is quite complex, especially if specific component have to be developed.

2.5 J-SIM

J-Sim (Java Simulator) is a free of charge implementation for simulation of a component-based architecture, the Autonomous Component Architecture (ACA). The ACA mimics the integrated circuit design and manufacturing model in terms of how components are specified, designed and assembled [J-Sim]. J-Sim includes a specific platform, dedicated to network simulation, the Internetworking Simulation Platform called INET, but is not limited to this field. Its organization is similar to the one of OMNeT++. J-Sim is a real-time process-driven simulator, in other words, a simulation runs in the same manner as a real system does, in the sense that event executions are carried out in real time as opposed to at fixed time points in discrete event simulation. As in NS-2, two languages are used in J-Sim: Java to describe and implement models and a script language to construct, configure and/or control the simulation at run-time. J-Sim has been designed to support Tcl, Perl or Python script languages, however, the available implementation is based on Tcl. J-Sim provides specific Tcl commands, the Runtime Virtual (RUV) commands, to simplify the manipulation and the configuration of the network components during simulation runtime. J-Sim includes the INET platform that is dedicated to network simulation the simulation results can be collected of three ways: Trace files, X-Y graph and NAM.
format. J-Sim is similar to OMNeT++ on several points; For example, it has a component-based architecture enabling generic modeling and it can store models in XML files. Its advantage, compared with OMNeT++, is that it is free of charge. Unfortunately, it does not integrate as many features as OMNeT++. Common features are less efficient. The OMNeT++ GUI integrates more tools and functionalities than J-Sim. In addition, the J-Sim simulation performance is not documented.

3. RESULTS AND CONCLUSION

Simulation of electricity and telecommunication can be done by Qualnet, Omnet++, Opnet, Ns-2 and J-Sim. Analysis tool for state estimation is available in Qualnet, Omnet++, Opnet, Ns-2 and J-Sim. All the simulators discussed in paper supports network visualization tools and can modify scenarios. Trace files are created by all the simulators discussed in the paper. Qualnet and Opnet communicates with other simulators whereas Omnet++, Ns-2 and J-Sim does not. Fast simulation capabilities are only supported by Qualnet and Opnet. Qualnet and Opnet are commercial whereas Ns-2, Omnet++ and J-Sim are open source.

4. CONCLUSION

The comparison with respect platform, interaction with other simulators, visualizes action capabilities and scalabilities of different network simulator have been discussed. Each has some advantages and disadvantages and each can be appropriate in different situation. The choice of a simulator should be based on the motive of study. Researchers must consider the pros and cons of different programming language, which simulation is driven like event vs. time, component-based or object-oriented architecture, the level of complexity of the simulator, features to include and not include and other design choices. The ns-2 and OMNeT++ in terms of open source must be the best choices in most of situation for research. Ns-2 is most popular simulator for academic research but it is normally criticized by its complicated architecture. Nevertheless, it is using largely by the research communities. OMNeT++ is getting popularity in educational and industrial area. Instead of that the ns-2, OMNeT++ has a well- designed simulation engine and powerful GUI. In terms of commercial simulator, Qualnet satisfies almost all important features.

<table>
<thead>
<tr>
<th>features</th>
<th>Qualnet</th>
<th>Omnet++</th>
<th>NS-2</th>
<th>OPnet</th>
<th>J-Sim</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language supported</td>
<td>Parsec C++</td>
<td>C++</td>
<td>C++/OTCL</td>
<td>C++/Java</td>
<td>Java</td>
</tr>
<tr>
<td>License</td>
<td>Commercial</td>
<td>Open source</td>
<td>Open source</td>
<td>Commercial</td>
<td>Open source</td>
</tr>
<tr>
<td>GUI support</td>
<td>Excellent</td>
<td>Good</td>
<td>Poor</td>
<td>Excellent</td>
<td>Good</td>
</tr>
<tr>
<td>Time taken to learn</td>
<td>Very easy</td>
<td>Moderate</td>
<td>Long</td>
<td>Long</td>
<td>Moderate</td>
</tr>
<tr>
<td>Platform</td>
<td>Linux</td>
<td>Linux,mac-os,unix</td>
<td>Unix,mac-OS Microsoft window Cygwin</td>
<td>C,C++,Opnet modeler software</td>
<td>Mat lab</td>
</tr>
<tr>
<td>Availability of analysis tool</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Network visualization tool</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Possibility to design and modify scenarios</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Create trace files</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Interaction with</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 1: Analysis and comparison of different network simulators
real systems

<table>
<thead>
<tr>
<th>Communication with other modules</th>
<th>✓</th>
<th>✓</th>
<th>✓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast simulation capabilities</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

References