Journal of Multidisciplinary Engineering Science and Technology (JMEST)
ISSN: 2458-9403
Vol. 6 Issue 7, July - 2019

A Survey On Existing Network Simulators

Suleiman Abdullahi[1], Lawal Idris Bagiwa[2], Murtada Elmukashfi Eltaher[3], Abubakar Aliyu[4], Abubakar Muhammad[5]

[1 & 5] Department of Mathematics and Computer Sciences, Faculty of Natural and Applied Science, Al-Qlam University, Katsina, P.M.B. 2137 Dutsin-ma Road, Katsina State, Nigeria.

[2]Department of Computer Studies, College of Science and Technology, Hassan Usman Katsina Polytechnic P.M.B. 2052 Katsina State, Nigeria.

[3]Assistant Professor - Faculty of Science - University of Bakht Alruda - Elduwiam – Sudan

[4]Directorate of Information & Technology, National Open University of Nigeria, Jabi Abuja Nigeria.

***Abstract*—In the network research area, it is very costly to deploy a complete test bed containing multiple networked computers, routers and data links to validate and verify a certain network protocol or a specific network algorithm. The network simulators in these circumstances save a lot of money and time in accomplishing this task. 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.**

Keywords—Network, Simulator, Simulation, OPNET, Network Protocol

1. Introduction

There are many things that are very important in modern technology including the Simulation. Simulation can be widely applied to many side fields including in different science, engineering, or other application fields for different purposes. Hypothetical and real-life objects or activities on a computer can be modeled with the help from computer assisted simulation as a study to see how the system function can be done. Different variables can be used to predict the behavior of the system. Modeling and analysis in many natural systems can be assisted by the computer simulation. Physics, chemistry, biology, and human-involved systems in economics, finance or even social science are those examples of the typical application. Other important applications are in the engineering such as civil engineering, structural engineering, mechanical engineering, and computer engineering. Application of simulation technology into networking area such as network traffic simulation, however, is relatively new.

The network simulation in specific means the technologies of computer assisted simulation are being applied in the simulation of networking algorithms or systems by using software engineering. Field of network simulator is narrower than general simulation. It is said to be natural that more specific requirements will be placed on network simulations. For instance, the network simulations may put more emphasis on the performance or validity of a distributed protocol or algorithm rather than the visual or real-time visibility features of the simulations. Moreover, since network technologies is keeping developing very fast and so many different organizations participate in the whole process and they have different technologies or products running on different software on the Internet. That is the reasons that the network simulations always require open platforms which should be scalable enough to include different efforts and different packages in the simulations of the whole network. Internet has also a characteristic that it is structured with a uniformed network stack (TCP/IP) that all the different layers technologies can be implemented differently but with a uniformed interface with their neighbored layers. Hence, the network simulation tools have to be able to incorporate this feature and allow different future new packages to be included and run transparently without harming existing components or packages. Thus the negative impact of some packages will have no or little impact to the other modules or packages (Bilalb and Othmana, 2013).

There are many users for network simulators that come from different areas. It is widely used by people such as academic researchers, industrial developers, and Quality Assurance (QA). Usually, the QA will use this network simulator to help them to design, simulate, verify, and analyze the performance of different networks protocols. The network simulator can also be used to evaluate the effect of the different parameters on the protocols being studied. In brief, a network simulator will comprise of a wide range of networking technologies and protocols and help users to build complex networks from basic building blocks like clusters of nodes and links. With the help from this network simulator, different network topologies using various types of nodes including end-hosts, hubs, network bridges, routers, optical link-layer devices, and mobile units can be design.

Basic concepts in network simulation: Network simulation and simulator

Basically, the network simulators try to model the real world networks. The principal idea is that if a system can be modeled, then features of the model can be changed and the corresponding results can be analyzed. As the process of model modification is relatively cheap than the complete real implementation, a wide variety of scenarios can be analyzed at low cost (relative to making changes to a

real network). Nevertheless, network simulators are not perfect. They cannot perfectly model all the details of the networks. However, if well modeled, they will be close enough so as to give the researcher a meaningful insight into the network under test, and how changes will affect its operation(Font *et al.*, 2011).

Simulation and emulation

In the research area of computer and communications networks, simulation is a useful technique since the behavior of a network can be modeled by calculating the interaction between the different network components (they can be end-host or network entities such as routers, physical links or packets) using mathematical formulas. They can also be modeled by actually or virtually capturing and playing back experimental observations from a real production networks. After we get the observation data from simulation experiments, the behavior of the network and protocols supported can then be observed and analyzed in a series of offline test experiments. All kinds of environmental attributes can also be modified in a controlled manner to assess how the network can behave under different parameters combinations or different configuration conditions. Another characteristic of network simulation that worth noticing is that the simulation program can be used together with different applications and services in order to observe end-to-end or other point-to-point performance in the networks(Kumar *et al.*, 2012).

Network emulation, however, means that network under planning is simulated in order to assess its performance or to predict the impact of possible changes, or optimizations. The major difference lying between them is that a network emulator means that end-systems such as computers can be attached to the emulator and will act exactly as they are attached to a real network. The major point is that the network emulator's job is to emulate the network which connects end-hosts, but not the end-hosts themselves. Typical network emulation tools include NS2 which is a popular network simulator that can also be used as a limited-functionality emulator. In contrast, a typical network emulator such as WANsim is a simple bridged WAN emulator that utilizes some Linux functionality(Kumar *et al.*, 2012).

Type of network simulators

There are different types of network simulators that can be categorized and explained based on some criteria such as if they are commercial or free, or if they are simple ones or complex ones.

Commercial and open source simulators

Some of the network simulators are commercial which means that they would not provide the source code of its software or the affiliated packages to the general users for free. All the users have to pay to get the license to use their software or pay to order specific packages for their own specific usage requirements. One typical example is the OPNET . Commercial simulator has its advantage and disadvantage. The advantage is that it generally has complete and up-to-date documentations and they can be consistently maintained by some specialized staff in that company. However, the open source network simulator is disadvantageous in this aspect, and generally there are not enough specialized people working on the documentation. This problem can be serious when the different versions come with many new things and it will become difficult to trace or understand the previous codes without appropriate documentations(Martinez *et al.*, 2011).

On the contrary, the open source network simulator has the advantage that everything is very open and everyone or organization can contribute to it and find bugs in it. The interface is also open for future improvement. It can also be very flexible and reflect the most new recent developments of new technologies in a faster way than commercial network simulators. We can see that some advantages of commercial network simulators, however, are the disadvantage for the open source network simulators. Lack of enough systematic and complete documentations and lack of version control supports can lead to some serious problems and can limit the applicability and life-time of the open source network simulators. Typical open source network simulators include NS2. We will introduce and analyze it in great detail in the following sections(Issariyakul and Hossain, 2012).

Table 1 Example of Network simulators

	Network simulators name
Commercial	OPNET, QualNet
Open source	NS2, NS3, OMNeT++, SSFNet, J-Sim

Simple vs. complex

Currently there are a great variety of network simulators, ranging from the simple ones to the complex ones. Minimally, a network simulator should enable users to represent a network topology, defining the scenarios, specifying the nodes on the network, the links between those nodes and the traffic between the nodes. More complicated systems may allow the user to specify everything about the protocols used to process network traffic. Graphical applications also allow users to easily visualize the workings of their simulated environment. Some of them may be text-based and can provide a less visual or intuitive interface, but may allow more advanced forms of customization. Others may be programming-oriented and can provide a programming framework that allows the users to customize to create an application that simulates the networking environment for testing(Pan and Jain, 2008).

Overview of current developments

Currently there are many network simulators that have different features in different aspects. A short list of the current network simulators includes OPNET,

NS-2, NS-3, OMNeT++, REAL, SSFNet, J-Sim, and QualNet. From the example list of those network simulators, only certain from it will be analyzed and compare in more detail including the NS-2, OMNet++, OPNET and J-Sim(Pan and Jain, 2008).

From the list selected, the OPNET is commercial software and is a little different from others and we will introduce in the first place. NS2 are the most popular one in academia because of its open-source and plenty of components library. A lot of non-benefit organizations contribute a lot in the components library and it has been proved that the development mode of NS2 is very successful. OMNet++ is another important network simulator which has very powerful graphical interface and modular core design. OMNet++ is also open sourced and widely acknowledged in academia(Pan and Jain, 2008).

OPNET

OPNET is the registered commercial trademark and the name of product presented by OPNET Technologies incorporation. It is one of the most famous and popular commercial network simulators by the end of 2008. Because of it has been used for a long time in the industry, it become mature and has occupied a big market share (Sarkar and Halim, 2011).

Overview of OPNET

OPNET is specially used in network research and development. By using OPNET, user will be able to study the communication of network, the devices, protocols or even the application flexibly. It can be flexibly used to study communication networks, devices, protocols, and applications. OPNET offers relatively much powerful visual or graphical support for the users as it has a fact of being a flexible and commercial software provider. In order to build network topology and entities from the application layer to the physical layer user may used the graphical editor interface and for process of mapping from graphical design to the implementation of the real systems can be done by using the object-oriented programming technique . It can be seen from the figure 1 that shows the example of the graphical GUI of OPNET. The configuration and simulation results of all topology can be presents intuitively and visually. Easy operation through the GUI causes the parameters to be adjusted and also the experiments can be repeated easily.

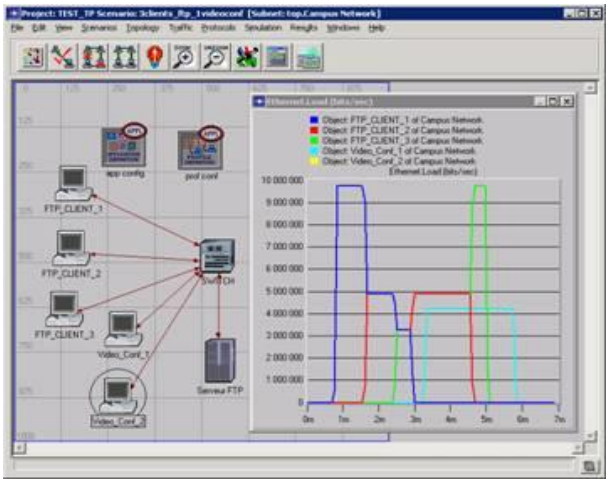

Figure 1. OPNET GUI [OPNET]

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. Organization of networks is done by hierarchical structure. OPNET also provide programming tools to allow users to define the protocol's packet format as like what the other network simulators provides. Besides to define the protocol's packet format, the programming tools provides by the OPNET are also required in order to help in accomplish several tasks as such of defining the state transition machine, defining network model and also the process module (Sarkar and Halim, 2011).

All in all, OPNET is a popular simulator used in industry for network research and development. The GUI interface and the programming tools helps a lot user to build the system that they want.

Main features of OPNET

Three main functions of OPNET are such as: modeling, simulating, and analysis. In modeling, the **intuitive** graphical environment is provides in order to create all kinds of models of protocols. There are 3 different advanced simulations technologies used in simulating as it can be used to address a wide range of studies and for analysis, the simulation results and data can be analyzed and displayed very easily. Graphs, charts, statistics, and even animation are very user friendly and can be generated by OPNET for users ' convenience (Sundani *et al.*, 2011).

According to the OPNET whitepaper, OPNET's detailed features include those that listed below:

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
7. Customizable wireless modeling

8. Discrete Event, Hybrid, and Analytical simulation

9. 32-bit and 64-bit parallel simulation kernel

10. Grid computing support

11. Integrated, GUI-based debugging and analysis

12. Open interface for integrating external component libraries.

Recent development of OPNET and its future

On August 7, 2008, OPNET Technologies announced the addition of two major application performance management capabilities. It includes end-to-end visibility into application performance for organizations using *WAN* optimization solutions and the ability to capture and analyze *NetFlow* data (Martinez *et al.*, 2011).

ACE Analyst software has been upgraded by OPNET recently including its 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 (Martinez *et al.*, 2011).

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 users's requirements and keeps improving their product which make it very competitive compared with other commercial network simulators in the near expectable future (Martinez *et al.*, 2011).

Network Simulator 2 (NS-2)

NS2 is one of the most popular open source network simulators. The original NS is a discrete event simulator targeted at networking research. In this section, we will give a brief introduction to the NS2 system (Font *et al.*, 2011).

Overview of NS-2

NS2 is the second version of NS (Network Simulator). NS is originally based on REAL network simulator. The first version of NS was developed in 1989 and evolved a lot over the past few years. The current NS project is supported through DARPA. The current second version NS2 is widely used in academic research and it has a lot of packages contributed by different non-benefit groups. For NS2 documentation on recent changes, refer to the NS-2 official webpage (Font *et al.*, 2011).

Main features of NS-2

First and foremost, NS2 is an object-oriented, discrete event driven network simulator which was originally developed at University of California-Berkely. The programming it uses is C++ and OTcl (Tcl script language with Object-oriented extensions developed at MIT). There are some reasons on the usage of these two programming language. The main reason is due to the internal characteristics of these two languages. C++ is efficient to implement a design but it has minor problem as it is difficult to be shown visual and graphically. Without a very visual and easy-to-use descriptive language, it's not easy to modify and assembly different components and to change different parameters without a very visual and easy-to-use descriptive language. Moreover, NS2 separates control path implementations from the data path implementation as for efficiency reason. In order to reduce packet and event processing time, the event scheduler and the basic network component objects in the data path are written and compiled using C++. The feature that C++ lacks seems that had and happen by the OTcl . Hence, it is prove that the combination of these two languages will be very effective. The implementation of the detailed protocol by using the C++ and users used to control the simulation scenario and schedule the events by using the OTcl. Figure 2 shows a simplified user's view of NS2. In order to initiate the event scheduler, the OTcl script is used and the network topology isset up. It then will tell traffic source when to start and stop sending packets through event scheduler. The OTcl script will be programmed to allow the scenes to be changing easily. User can make a new network object by write the new object or assemble a compound object from the existing object library. Later, user can plumb the data path through the object. This plumbing makes NS2 very powerful (Issariyakul and Hossain, 2012).

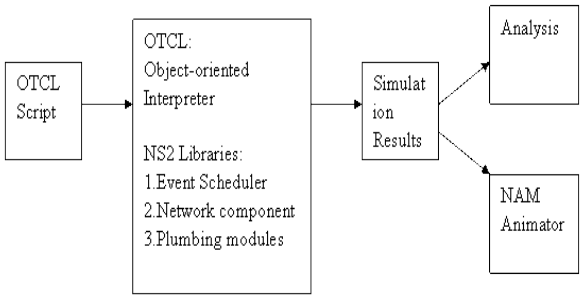

Figure 2. Simplified User's View o f NS2

Event scheduler is another feature of NS2. The event scheduler keeps track of simulation time and releases all the events in the event queue In NS2. This is done by invoking appropriate network components. All the network components use the event scheduler by issuing an event for the packet and waiting for the event to be released before doing further action on the packet.

Recent developments of NS-2 and its future

NS 2.33 is the most recent version of NS2. It 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 and multirate 802.11 library from Nicola Baldo and Federico Maguolo of the SIGNET group, University of Padova. NS is now developed in collaboration between some different researchers and institutions, including SAMAN (supported by DARPA), CONSER (through the NSF), and ICIR (formerly ACIRI). Contributions have also come from Sun Microsystems and the UCB and Carnegie Mellon Monarch projects. Generation 3 of NS (NS3) has begun development as of July 1 , 2006 and is projected to take four years (Issariyakul and Hossain, 2012).

OMNeT++

Similar with NS-2, OMNeT++ is also a public-source, component-based network simulator with GUI support. Its primary application area is communication networks. 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.

Overview of OMNeT++

Like NS-2, OMNeT++ is also a discrete event simulator. It is a component-based architecture. 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 NS-2 and Python in NS3. OMNeT++ also provides GUI support, and due to its modular architecture, the simulation kernel can be embedded into all kinds of different user s' applications. Figure 5 is an OMNeT++ GUI screenshot (Pan and Jain, 2008).

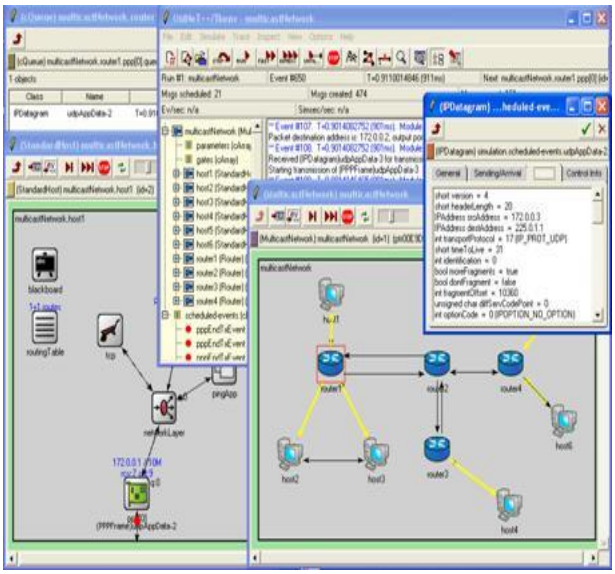

Figure 5. OMNeT++ GUI

Main features of OMNeT++

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 include (Pan and Jain, 2008):

1. Simulation kernel library

2. Compiler for the NED topology description language (nedc)

3. Graphical network editor for NED files (GNED)

4. GUI for simulation execution, links into simulation executable (Tkenv)

5. Command-line user interface for simulation execution (Cmdenv)

6. Graphical output vector plotting tool (Plove)

7. Graphical output scalars visualization tool (Scalars)

8. Model documentation tool (opp_neddoc)

9. Utilities (random number seed generation tool, makefile creation tool, etc.)

10. Documentation, sample simulations, etc.

As the key feature of OMNeT++, the simulation kernel C++ class library consists of the simulation kernel and utility classes which will be used to create simulation components. The library also includes the infrastructure to assemble simulations from different components. Besides these, there are also runtime user interfaces or environments for simulations, and tools to facilitate and manage simulations. OMNeT++ can run on Linux, other Unix-like systems and on Windows (XP, Win2K).

OMNeT++ represents a framework approach. It provides an infrastructure for writing different simulations. Specific application areas' requirements are met by different simulation models and frameworks, most of which are open sourced. More important, these models are developed completely independently of OMNeT++, and follow their own release cycles. This is another important feature of OMNeT++ (Pan and Jain, 2008).

Recent developments of OMNeT++ and its future

Currently, OMNeT++ is popular in academia for its extensibility since it is also open sourced and there are plentiful online documentations. There is also a mailing list for the general discussion (Kumar *et al.*, 2012).

OMNeT++ is being used in the academia as well as in industry. Several open source simulation models have been published in the field of network simulations such as IP, IPv6, MPLS, mobility and ad-hoc simulations (Kumar *et al.*, 2012).

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 (Rachedi *et al.*, 2012).

J-Sim

J-Sim (formerly called JavaSim, the former name conflicted with Sun's Java trademark) has been developed by a team at the Distributed Realtime Computing Laboratory (DRCL) of the Ohio State University. Additional third-party packages are also available. The project has been sponsored by the National Science Foundation (NSF), DARPA's Information Technology Office (DARPA/ITO, through the Quorum global distributed computing program and the Network Modeling and Simulation program), Air Force Office of Scientific Research's Multidisciplinary University Research Initiative (AFORS MURI), the Ohio State University and the University of Illinois at Urbana-Champaign (Sundani *et al.*, 2011).

Overview of J-Sim

J-Sim is free and available with source code, examples, tutorials and white papers. Additionally, there is a publicly available mailing list for J-Sim users.

J-Sim is implemented in Java and uses Tcl binding in the form of Jacl. Java is used to create simulator's objects, called *components*, while Tcl enables topology setup and provides a limited means of simulation control. Similarly to NS-2, J-Sim offers higher-level Tcl-based programming interface (Sundani *et al.*, 2011).

Main features of J-Sim

J-Sim is based on a so-called Autonomous Component Architecture (ACA). The components are written in Java and expose some well-defined interfaces. The components communicate with each other through *ports*, and are bound to *contracts*. Contracts define component interactions at layer boundaries (e.g., how an FTP application creates TCP sockets); they allow to make use of lower layer's services but at the same time introduce tight coupling between layers (e.g., an FTP socket application must use TCP socket transport component).

Components are assembled at runtime by "wiring" their ports. The components are organized in a tree-like structure (i.e., a parent component is a directory for its children). The Runtime Virtual System (RUV) is a set of Tcl commands, built atop Tcl/Jacl, which resembles file system commands in UNIX, used to manipulate the component tree (e.g., mkdir, cp, mv, rm, cd, pwd, ls). Also, hierarchical components' names look similarly to file paths in UNIX (e.g., /net/router/port0@group0) (Sundani *et al.*, 2011).

Recent developments of J-Sim and its future

Two simulation models are available for J-Sim which is the multithreaded, "real-time process-based," and the "classical" discrete event based. The simulation within J-Sim is by definition multithreaded. The *runtime* manages a pool of *worker threads* to provide a new *execution context* for a component to handle and process incoming data. Because of this mode, and unlike in ns-2 and earlier OPNET releases, developers should pay special attention to proper synchronization of critical regions in a component's code. Note that due to possible various thread scheduling in different Java Virtual Machines, subsequent runs of the same simulation scenario (with no randomness) may yield different results (Sarkar and Halim, 2011).

The simulation time is advanced proportionally to the wall time when at least one thread is active, otherwise it is advanced to the moment at which at least one thread may be woken (a "jump in time" is made). Thus, while the simulation is based on events (usually, data reception on a port), time management is completely opposite to discrete event simulation in ns-2 or OPNET, where events are scheduled at fixed time points (and the simulation time is frozen between two events). However, the larger value of a time scale, the closer the time management in J-Sim to discrete event simulation is (event processing times become negligible). Note that this so-called "real-time process-based" approach has some severe consequences: Helper actions, not directly related to simulation (e.g., log file formatting, component state validation, etc.) may affect the still elapsing simulation time. Additionally, if a component code needs to add some artificial delay (e.g., a slow router is simulated), then the delay cannot be guaranteed, due to thread scheduling within a JVM (and due to process scheduling within an operating system). For example, a component developer needs to use J-Sim-specific thread synchronization and delay primitives, to enable the kernel to make a "jump in time" when all threads are suspended for some timeout; otherwise the simulation may be frozen when a thread sleeps (Sarkar and Halim, 2011).

COMPARISON

The aim of this comparatively study is to find wirelessvnetwork simulator that establish a good balancing scenario between the features, efficiency, accuracy, and easy to use. Such a simulator will allow bresearchers to take the steps to describe above without much difficulty (Sarkar and Halim, 2011).

The GloMosim simulator has the lack of various TCP implementations. It does not support GUI for large number of nodes.Quelnet is a commercial simulator that grew out of Glomosim while the weakness of the Qualnet it is not the OPEN source like GloMosim. OMnet++ includes the lot of functionality so it is the complete Tool for generic simulation. It also support the GUI which receives in the Praise form and weakness of this simulator the GUI is not detailed enough to be useful. OpNet Modeler is very powerful and user friendly simulator and it proves also the easy modeling with good

documentation with its large library for simulation models. It also has the mobility models like ns-2 and J-Sim simulators. External tools are not supported since kernel is not open source. NS-2 is the most used simulator by the research community besides this it has the large numbers of available models like realistic mobility models, powerful and flexible scripting and simulation setup, large user community. The main weakness in ns-2 is OTCL language and overall architecture of the simulator and patching of extension is not easy. It also did not differentiate the working between OTCL and C++. J-Sim is the Java based simulator. It has the facility of component based architecture with realistic mobility models. It is also called the autonomous component architecture (ACA). On the other hand it has the drawback of inadequate documentation. Abbreviation used in table detailed is given below (Martinez *et al.*, 2011):-

- GloMosim: Global Mobile Information System.
- Quelnet: Commercial Version of GloMosim Simulator.
- OMNett++: Objective Modeler Network Testbed in C++.
- NS-2: Network Simulator-2.
- Opnet Modeler: Optimized Network Engineering Tools Modeler.
- J-Sim: Java –Simulator.
- NET/SYS: Network System.
- OS: Operating System.
- GUI: Graphical User Interface.
- T/W/A: Time warp.

CONCLUSION

The comparison with respect to installation implementation issues, visualize action capabilities and scalabilities of different network simulator have been discussed for researchers to find a simulator having efficient and easy development environment during research. Each has some advantages and disadvantages and each can be appropriate in different situation. The choice of a simulator from the discussed available simulators should be driven by the research requirements it is the motive of this 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++ 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 welldesigned simulation engine and powerful GUI, so these are better for simulation environment development.

Reference

Bilalb and Othmana (2013). A Performance Comparison of Network Simulators for Wireless Networks. *arXiv preprint arXiv:1307.4129.*

Font *et al.* (2011). Analysis of source code metrics from ns-2 and ns-3 network simulators. *Simulation Modelling Practice and Theory.* 19 (5.), 1330-1346.

Issariyakul and Hossain (2012).*An introduction to network simulator NS2*, Springer.

Kumar *et al.* (2012). Simulators for Wireless Networks: A Comparative Study. *Computing Sciences (ICCS), 2012 International Conference on.* 338-342.

Martinez *et al.* (2011). A survey and comparative study of simulators for vehicular ad hoc networks (VANETs). *Wireless Communications and Mobile Computing.* 11 (7.), 813-828.

Pan and Jain (2008). A survey of network simulation tools: Current status and future developments. *Email: jp10@ cse. wustl. edu.*

Rachedi *et al.* (2012). Wireless network simulators relevance compared to a real testbed in outdoor and indoor environments. *International Journal of Autonomous and Adaptive Communications Systems.* 5 (1.), 88-101.

Sarkar and Halim (2011). A review of simulation of telecommunication networks: simulators, classification, comparison, methodologies, and recommendations. *Cyber Journals: Multidisciplinary Journals in Science and Technology-Journal of Selected Areas in Telecommunications (JSAT).* 2 (03.), 10-17.

Sundani *et al.* (2011). Wireless Sensor Network Simulators A Survey and Comparisons. *International Journal of Computer Networks (IJCN).* 2 (6.), 249-265.

Appendix

Result and Comparison among the Simulators.

Sr.no	Tools/feature	Glomosim	Quelnet	OMnet++	Ns-2	Opnet Modeler	J-sim
1	Interface	Parsec C-based language	Parsec C++-based language	C++	C++/OTCL	C or C++	Java
2	User support	Poor	Excellent	Good	Excellent	Excellent	Excellent
3	License	Open source	Commercial	Free for educational use only	Open source	Free for Academic limited use	Open source
4	Scalability	Large	Very large	Large	Small	Medium	Small
5	Learning time	Moderate	Easy to learn	Moderate	Long	Long	Moderate
6	Availability	T/W/AD	T/W/AD/WSNA	T/W/AD	T/W/AD	T/W/WSN	T/W/AD/WSNA
7	Developed	Developed at University of California,It is written in the Bargrodia, & Gerla, 1999	Derived from the GloMoSim that was first released in 2000 by SNT	The Principle author is Andre Varge from Technical university of Budapest and it is publicly available since 1997	REAL network simulator 1989	First proposed by MIT in 1986 and initially developed by MIT of technology 1987 using C++	Distributed Real Time Computing Laboratory of the Ohio State University and by Illinois University in 2005
8	Mobility	Support	Support	No	Support	Support	Support
9	Applicability	Net./Sys	Net. /Sys.	Net. /Sys.	Net. /Sys.	Net. /Sys.	Network
10	Platform	Linux of Java 1.3 or higher versions, Parsec Compiler, GlomoSim software	Qualnet Simulator and SDL protocol developer.	OmNet++Runs on Linux, Mac OS x, other Unix-like systems and on and window XP, Win2K, Vista, 7	UNIX, Mac OS X, Microsoft Window Cygwin	C, C++ and Opnet modeler software.	VI. a GUI platform for real-time sharing of Mat lab designs and simulations
11	Graphic Interface support	Limited GUI	Excellent GUI	Good GUI	Limited GUI	Excellent GUI	Good GUI