Research paper © 2012 IJFANS. All Rights Reserved, UGC CARE Listed (Group -I) Journal Volume 11, Iss 10, October 2022

Simulators for Wireless Networks: A Study

Rahul Vishnoi, Assistant Professor,
Department of Electrical and Communication Engineering, Teerthanker Mahaveer University, Moradabad, Uttar Pradesh, India
Email Id- ra_v@yahoo.com

ABSTRACT: Unlike wired networks, wireless networks are one of the finest fields for study. Simulators are extremely excellent concessions between cost and complexity, as well as accuracy of the findings, in ad hoc wireless networks to test the performance and behavior of different routing methods and protocols. It may be tough to select which simulator to use for ad hoc wireless networks since there are so many options. The article includes a review of simulators that enable ad hoc wired networks. Its goal is to create a simulator that will assist researchers in developing and analyzing different strategies and models in a more reliable and efficient manner. Unlike previous simulator comparisons, we do not concentrate on the correlation of individual simulation outcomes, but instead attempt to evaluate simulators based on features and usability. Ad hoc wireless networks have grown in popularity during the past several years. Simulation is a widely used technique for evaluating the performance of different routing methods and models in ad hoc wireless networks.

KEYWORDS: Wireless Networks, Simulations, OPNET Modeler, NS-2, J-Sim.

1. INTRODUCTION

New routing methods and network designs are also developed via simulation. They are typically built in network simulators to explore any features for routing methods in ad hoc wireless networks. Setting simulation settings allows you to quickly study the behavior of network architectures and routing topologies. The majority of network simulation toolkits are based on the discrete event-based simulation paradigm[1]. The goal of this survey is to identify a wireless network simulator that strikes a fair mix of functionality, efficiency, extendibility, accuracy, and usability. Researchers will be able to complete the processes outlined above with little effort, allowing them to focus on their study rather than the simulator. As a result, this study collects data on the features, capabilities, advantages/disadvantages, and topologies of various network simulators in order to determine their suitability as a research tool. The survey is based on a collection of articles and surveys, some of which evaluate various network simulators based on predetermined criteria, and others of which examine simulators' compliance with a particular project or area[2].

To assess the simulator's performance and make comparisons, we use a variety of parameters. Because wireless network modeling is more challenging than wired network simulation in terms of deciding the underlying details of a specific routing strategy and parameters to utilize in simulation. Significant abstractions have been possible in wired networks during the past 30 to 40 years of study. Point-to-point connections, for example, are often represented by bandwidth and a queue delay, as well as framing and transmission faults. On the other hand, in the area of ad hoc wireless networking, there is little advice on what abstractions to use. Low-level details in ad hoc wireless networks may have a big impact on routing strategy performance. Simulation is the greatest tool for studying such features and characteristics in ad hoc wireless networks. We provide a case study in which topology control routing methods were evaluated using widely used wireless network simulators such as NS-2, OPNET Modeler, J-Sim, and OMNET++. From a feature and usability standpoint, we also identify the missing aspects of each simulator, as well as the amount of work required for installs, familiarizations, implementations, and visualizations. It also includes descriptions of a list of simulators (as listed above), as well as estimates of their popularity and some clues as to which simulators are utilized for certain purposes. Finally, a table is given that compares the chosen simulators based on their own languages, modules, and whether or not they include GUI support [1].

2. DISCUSSION

There may be some major differences in how simulators work. It is impossible to say that these variances can be represented accurately. No network simulator is technically accurate. A simulator may be described as reliable and realistic at best. Researchers that need extreme precision will want to use test beds to perform their research on actual equipment. If this is not feasible, they will have to rely on simulation and therefore accept a certain degree of uncertainty. For instance, the effect of granularity, node mobility, radio propagation models, simulation sizes, and so on. Simulators are useful because they offer software frameworks for creating a computerized model that includes all of the relevant information[3].

2.1. OPNET MODELER

The heart of the OPNET (Optimized Network Engineering Tools) Modeler is a finite state machine. A balance between speed and accuracy is reached when the analytical model, which is responsible for speeding up the simulation by utilizing mathematical models that come with modeler, is used in conjunction with the analytical model. The goal of the modelers was to do diagnostics on a company's network and then assist in its restructuring. It has a number of preset functions, protocols, devices, and behaviors that make the Modeler a powerful application right out of the box and with little effort. It was first suggested and built in 1987 at MIT for simulation purposes, using C++.

OPNET Modeler uses hierarchical modeling to describe a network as a collection of sub-models representing sub-networks or nodes. A simulation's topology may be manually built, imported, or chosen from a pool of preset topologies. As previously stated, OPNET simulations are event-driven, with an event being a request for a certain action to occur at a specific moment. It may be time-based, which implies that various techniques could be used to sample at regular intervals but advance when an event happens. A single global event list is maintained by an OPNET simulator, and all objects have access to a common simulation time clock. The events are listed in chronological order, with the first item in the list being regarded the head[4].

All of the events have data connected with them, and when one of them is finished, it is deleted from the list. When an event reaches the top of the event list, it becomes an interrupt and is sent to the simulator's assigned module by the simulation kernel. When an interrupt occurs, the module may receive data connected with the event, and specific modules, processes, and queues are also chosen to put initial interruptions on the event list. The Simulation Kernel (SK) is in charge of managing the whole event list, delivering each event in order to the relevant module, as well as receiving requests from modules and inserting new events into existing event lists[5].

The OPNET Modeler wireless suit is used by the simulator to offer modeling, simulations, and analysis of wireless networks. It also has complete protocol stack modeling capabilities, allowing it to simulate all elements of wireless communications such as RF propagation, interferences, sender/receiver characteristics, node mobility, handover, and so on. OPNET may run several simulation situations at the same time. It also enables distributed simulations through a feature called High level architecture, which allows connection with other simulations as well as system in the loop simulations that communicate with real hardware and software. For result analysis, OPNET modeler includes a source code editor, Network model editor, Node model editor, Process model editor, Antenna pattern editor, Packet format editor, Simulation tools, and an OPNET animation viewer. These are the simulator's major features, but it does have some flaws. For example, with the OPNET modeler, the accuracy of the findings is limited by the sampling resolution, and the simulation is wasteful if nothing occurs for extended periods of time[6].

2.2. OMNET++

OMNET++ (Objective Modular Network Testbed in C++) is a C++ simulation toolkit and framework that is mainly used to create network simulations. It is extensible, modular, and component-based. OMNET++ is an environment for discrete event simulation. Its main application field is the simulation of communication networks, but it has also been effectively utilized in other areas such as the modeling of complex IT systems, queuing networks, and hardware designs due to its generic and flexible architecture. For OMNET++, there are currently two main network simulation model frameworks: the Mobility framework and the INET framework. At TU Berlin, the mobility framework was created to offer strong foundations for the creation of wireless and mobile networks. While the IP Suite was created at the University of Karlsruhe, the INET framework emerged from it. It has a detailed protocol model for a variety of protocols [7].

Model component architecture is provided by OMNET++. C++ is used to write components and modules, which are subsequently combined into bigger components and models using a high-level language. Model reusability is provided for free. The simulation kernel (and models) may be simply integrated into your applications thanks to OMNET++'s strong GUI support and modular design. OMNET++ is open-source software that may be used for non-profit purposes under the Academic Public License. The goal of creating OMNET++ was to provide a sophisticated open-source discrete event simulation tool that could be used to simulate computer networks and distributed or parallel systems by academic, educational, and research-oriented commercial organizations. OMNET++ aims to bridge the gap between open-source simulation tools like NS-2 and more costly commercial equivalents like OPNET[8].

2.3. NS – 1 NS – 2

The second edition of the famous network simulator Network Simulator (NS) is designed for wired networks. NS - 2 is an open source event driven simulator that is used by wireless network researchers, academics, and others. NS - 2 may be used to simulate wired and wireless network operations and protocols, routing methods, TCP, UDP, and other protocols. Since its inception in 1989, NS2 has maintained its prominence in the networking research field due to its flexibility and modular nature. Since then, numerous revolutions and modifications have characterized the tool's maturation, due to significant contributions from the field's participants.

LBNL created the initial version of ns, known as ns – 1, which was adapted from an older simulator called as REAL. The MIT group then published version 2 of the simulator in 1995. Although ns-2 was intended to mimic wired networks, it can also simulate wireless networks utilizing the Wireless and Mobility Extensions to NS for IEEE 802.11 and many additional Bluetooth extensions developed by the CMU Monarch Project. The simulator contains an energy model and enables users to create traffic and movement patterns with ease. It also includes a set of randomized movement models, and many initiatives are underway to add sophisticated mobility models to the simulator, resulting in more realistic simulations. Mobile IP is also available for the wireless component [9].

Except for the presentation and session levels, NS - 2 offers OSI layers. It has a vast number of features available, including a big number of external protocols that are already implemented. Within the network community, the simulator's behavior is well regarded. NS -

2 allows for deterministic or probabilistic packet loss in network node queues, as well as deterministic and stochastic traffic distribution models. By changing scenario scripts, it is also possible to define disturbances and corruptions that may occur in a simulated network, such as link disruptions, node halt, and recovery. The OTcl script interpreter's simulation event scheduler is either a non-real-time scheduler or a real-time scheduler that is primarily utilized for real-time synchronization of an emulated network. The user specifies when network components should begin or stop sending packets in the event scheduler. NS-2 may be linked to a real network and collect live packets exactly like a regular node thanks to its emulation capability. It also has the ability to inject packets into a live network. By enabling users to choose parameters to be tracked, the simulator may create customized trace files, which saves CPU resource [10].

NS2 delivers either text-based or animation-based simulation results after simulation. Tools like NAM (Network AniMator) and X-Graph are used to analyze these findings visually and interactively. Users may take a relevant portion of text-based data and convert it into a more plausible display to evaluate a certain network activity. One of NS – 2's main flaws is that it requires recompilation every time a user code modification is made. As a result, the user must create his or her own user code in a separate shared library connected to the NS – 2 kernel. The second major point is that simulations with more than a hundred nodes are difficult to run due to a lack of scalability. Although NS-2 has a "small suite," large-scale networks need numerous changes and additional caution in memory allocation and CPU time management. NS-2 outperformed OPNET Modeler in terms of bandwidth estimate accuracy for pure CBR-type traffic. The NS -2 simulators operate correctly on GNU/LINUX, UNIX, Solaris, and Mac OS platforms. We can either create NS from scratch or obtain a "all-in-one package." The all-in-one programs provide additional capabilities for simulation, particularly in wireless scenarios, but the major drawback is that they only operate on LINUX / UNIX systems.

2.4. J-SIM

J-Sim (previously known as Java Sim) is a JAVA-based network simulator. It is designed using the component-based software architecture. It offers a compositional simulation environment created by a team at Ohio State University's Distributed Real Time Computing Laboratory and Illuinois University. It is based on the autonomous component architecture (ACA), which is a design and manufacturing paradigm for integrated circuits. In J-Sim, everything is a component, such as a node, a connection, or a protocol. Ports are used to connect the various components. The ports may be connected in three different ways: one-to-one, one-to-many, and many-to-many. Because J-Sim isn't often utilized in research, some may have doubts about its models' validity. The J-Sim component contracts, which defines how a component reacts to data arriving at each of its ports. Components may handle data in their own execution context, allowing them to be planned, built, and tested independently of the rest of the system. Setting flags for components is a useful feature of J-Sim, since it gives you more choices for enable, disable, and display.

3. CONCLUSION

Different network simulators for ad hoc wireless networks were detailed in this study, along with their strengths, weaknesses, and features. The goal is to find a wireless network simulator that will assist academics by providing an efficient and easy-to-use development environment for their study. As can be observed from the survey, simulators provide a large range of characteristics, but none of them can be supported by all of them. So the goal here was to find a well-balanced simulator that would provide a good user experience. It may be inferred from the preceding descriptions and findings made in different reference articles that ns-2 is the best option for the study job. Although OMNET++ offers several useful capabilities, ns-2 is the most often used simulator in academic research.

Although ns-2 has a complex structure that is difficult to grasp, the widespread usage of the community makes it simpler to use since many people assist each other with their issues through mailing lists and forums. OMNET++ gained a lot of traction in the corporate world rather than in academic research. OMNET++, unlike ns-2, has a well-designed simulation engine and enables hierarchical modeling, making it a superior choice for development. Unlike ns-2, OMNET++ also has a powerful GUI. However, OMNET++ lacks the huge number (loads) of external models and users that ns-2 can readily handle.

REFERENCES:

[1] M. M. Koksal, “A survey of network simulators supporting wireless networks,” *Middle east Tech. Univ.*, 2008.

[2] A. S. Toor and A. K. Jain, “A survey on wireless network simulators,” *Bull. Electr. Eng. Informatics*, 2017, doi: 10.11591/eei.v6i1.568.

[3] A. Augustine, “A Comparison of Network Simulators for Wireless Networks,” *Int. J. Adv. Res. Electr. Electron. Instrum. Eng.*, 2017.

[4] R. Khan, S. M. Bilal, and M. Othman, “A Performance Comparison of Network Simulators for Wireless Networks,” *A Perform. Comp. Netw. Simulators Wirel. Networks*, 2013.

[5] A. Kumar, S. K. Kaushik, R. Sharma, and P. Raj, “Simulators for wireless networks: A comparative study,” 2012. doi: 10.1109/ICCS.2012.65.

[6] P. Owczarek and P. Zwierzykowski, “Review of simulators for wireless mesh networks,” *Journal of Telecommunications and Information Technology*. 2014.

[7] A. Boulis, “Castalia A simulator for Wireless Sensor Networks and Body Area Networks - User’s Manual,” *Version 3.0*, 2010.

[8] M. Jaganath, R. Vasanth, and G. Malarselvi, “An exhaustive consideration of wired and wireless network simulators,” *Int. J. Recent Technol. Eng.*, 2019, doi: 10.35940/ijrte.B1017.0782S419.

[9] J. M. Yi, M. J. Kang, and D. K. Noh, “SolarCastalia: Solar energy harvesting wireless sensor network simulator,” *Int. J. Distrib. Sens. Networks*, 2015, doi: 10.1155/2015/415174.

[10] S. Kang, M. Aldwairi, and K. Il Kim, “A survey on network simulators in three-dimensional wireless ad hoc and sensor networks,” *Int. J. Distrib. Sens. Networks*, 2016, doi: 10.1177/1550147716664740.