A Survey of Network Simulators Supporting Wireless Networks

Murat Miran Köksal
Department of Computer Science
Graduate School of Natural and Applied Sciences
Middle East Technical University
Ankara, TURKEY

October 22, 2008

Abstract

This paper contains a survey about network simulators supporting wireless networks. It aims to find a wireless network simulator which would help researchers develop and analyze simulation models in a reliable and not-too-complex way. So, it describes four popular network simulators, i.e. J-Sim, OMNeT++, ns-2 and OPNET Modeler, and includes information about each one's abilities, internal structure, development environment, support for wireless network simulation, documentation and graphical user interface. It depicts strengths and weaknesses of the simulators and gives several comparisons of them based on a number of papers and surveys. Finally, the paper concludes by presenting two simulators which seem to comply more with the needs of researchers studying on wireless networks.

1 Introduction

Simulation is an important part of research not only for the study of communications networks, but for the study of any system which is costly to build (as a testbed) and restricted in terms of flexibility. Network simulators provide researchers with a virtual and reproducible environment where they can design, configure, run and analyze different kinds of networks without getting their hands dirty with real-life hardware. Hence, researchers can concentrate on their study and test their developments easily.

The steps one should take for running a network simulation basically include development of a model (e.g. implementation of a protocol), creation of a simulation scenario (e.g. designing a network topology and traffic scenario), and selection of statistics to be collected. The last step is the visualization and analysis of simulation results which may be carried out after (or during, in some cases) the simulation execution.

The aim of this survey is to find a wireless network simulator that provides a good balance between features, efficiency, extendibility, accuracy, and easiness of use. Such a simulator will allow researchers to take the steps described above without much hassle, and allow them to concentrate on their research rather than the simulator. Therefore, this survey contains information about features, abilities, advantages/disadvantages and structures of different network simulators and investigates their feasibility as a research tool. The survey is based on a collection of papers and surveys, some of which compare different network simulators according to defined criteria and some of which analyze simulators' conformance to a specific project or area.

[1] presents a case study in which four popular wireless network simulators, i.e. J-Sim, OMNeT++, ns-2 and ShoX, were used to evaluate a topology control protocol. Not only the authors describe outstanding and desirable but missing features of the simulators, but they also compare the amount of ef-

fort needed for installation, familiarization, implementation and visualization from feature and usability point of view (instead of correlation of the individual simulation results).

[2] provides a map of the main characteristics that MANETs simulation tools should feature and the current support of these. It gives a description of a list of simulators (DIANEmu, GloMoSim, GTNets, J-Sim, Jane, NAB, ns-2, pdns, OMNeT++, OPNET Modeler, QualNet, and SWANS), provides an estimation of their popularity, and gives some hints on which simulator to use for what needs.

[3] gives an overview about different issues in wireless sensor networks on a general basis. Only at the end a table is presented comparing the considered simulators according to their language, the available modules, and whether they have GUI support or not.

[4] is a survey of a list of telecommunication networks simulators done for an industrial research of service availability and service resilience in the Next Generation Networks (NGNs). The authors present the criteria used in their evaluation of selected simulators (OPNET Modeler, ns-2, OMNet++/OMNEST, GLASS/SSNFNet, QualNet, J-Sim, TOTEM) and then the result of their survey in which those criteria have been applied. Even though [4] offers one of the most detailed comparisons of all papers which this survey is based on, it evaluates all simulators from an industrial research point of view. Therefore, it is better taken into consideration by the reader that a feature which is approved by other papers can be described as being insufficient by [4].

[5] simply presents a comparative study of two network simulators, OPNET Modeler and Ns-2, and provides a guide to researchers undertaking packet-level network simulations. The simulator outputs were compared to the output from a live network testbed.

[6] is a deliverable of a project called IRRIIS, Integrated Risk Reduction of Information-based Infrastructure Systems, the purpose of which is to identify, list and compare tools and components suitable for simulation of critical infrastructures. The simulators used are OPNET Modeler, NS-2, QualNet, OMNeT++, J-Sim, SSF and Backplane.

All of the network simulators discussed in this paper are free of charge or can be freely used by signing up for special university programs in case of OPNET Modeler.

2 J-Sim

J-Sim is a component-based, compositional simulation environment developed by a team at the Distributed Real Time Computing Laboratory of the Ohio State University and by Illinois University. It is written in Java. It has been built on the autonomous component architecture which mimics the integrated circuit design and manufacturing model in terms of how components are specified, designed, and assembled.

All work in J-Sim project is done voluntarily so nobody is responsible for bugs. J-Sim is not used often in research projects, so one can question the validity of its models before starting to work it.

In J-Sim, each network entity (a node, a link or a protocol) is a component. A component can be composite, meaning that it can be composed of several inner components. Components communicate through their ports, and connection types of 1-to-1, 1-to-many and many-to-many are supported. The behavior of a component is described with a contract. The port contract describes the communication pattern of a component with other components that are connected to an individual port. The component contract describes how a component responds to data that arrives at each of its ports.

Components are able to handle data in an execution context of their own, so that they can be designed, implemented and tested separate from the rest of the system. One nice feature is the ability to set flags for components which offer the options of enable, disable, and display. Using flags, failures can be set for nodes or protocols, for example, by simply disabling them.

The simulator can be extended by creating new components by using existing ones as subclasses and redefining their attributes and methods. It supports hierarchical networks of arbitrary depth. While a network consists of nodes, links and small networks, an internetwork contains networks, nodes and links.

J-Sim defines a generalized packet switched net-

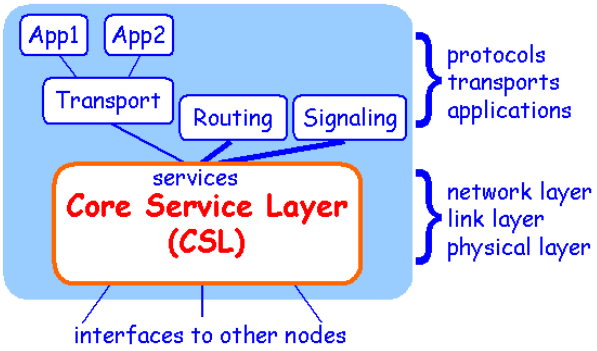

Figure 1: The internal structure of an Internetworking Simulation Platform (INET) node.

work model that contains the generic structure of a node and the generic network components, both of which can then be used as base classes to describe and implement new protocols at various layers. It distinguishes two layers as core service layer (lower layer) comprising layers from Network Layer to Physical Layer, and higher level comprising other layers of OSI model.

J-Sim uses Java's own utility [22] for generating random numbers which has an insufficient cycle length for avoiding repetition of randomness in simulation of large networks, as explained by [4]. Therefore, the simulator is adequate for small to medium scale simulations. Speaking of Java, it should be pointed out that, real java applications generating traffic can be plugged into the simulator.

Although models described in J-Sim are deterministic (i.e. the same results are obtained, regardless of how many times the model is re-calculated), they allow the use of stochastic variables (bringing uncertainty and variation) linked to the traffic model.

A script interface allows integration with different script languages like Tcl, which is currently fully integrated into the environment. The scripting language is used to create simulation scenarios that configure and control the simulation at runtime as well as to monitor and collect simulation data. It is stated by [1] that although Tcl requires some learning overhead, the binding between Java and Tcl to access Java objects and methods from Tcl is pretty intuitive.

For the simulation of wireless networks, J-Sim has a wireless extension. The only available MAC layer in this extension is IEEE 802.11. There are three radio propagation models which have been implemented in the extension: Free Space Model, Two-ray Ground Model and Irregular Terrain Model. Also implemented are two stochastic mobility models: Random Waypoint Mobility Model and Trajectory Based Mobility Model.

In J-Sim, execution contexts are implemented by Java threads with the thread scheduler of the Java Virtual Machine scheduling thread execution. Therefore, a simulation runs in the same manner a real system does. Event executions are carried out in real time so that the interactions and interferences among event executions take place as in real systems. This enhances the fidelity of the simulation.

The simulator has a good introductory material with overviews and examples for small scenarios but lacks a comprehensive manual and several more specific questions remain undocumented according to [1]. [4] supports this thoughts by stating that documents have not reached the level of details which is desirable.

A graphical editor exists for Tcl configuration files called gEditor. Also, a special plot component is provided to plot simulation statistics. Unfortunately, there are no tools provided for network visualization. Still, one can use ns-2's Network Animator (Nam) to visualize simulations since trace files generated by the simulator conform to Nam format.

J-Sim works on any operating system that runs Sun's Java SDK 1.5 or later.

3 OMNeT++

OMNeT++ (Objective Modular Network Testbed in C++) is a well-designed discrete event simulation environment. The principle author is Andras Varga from Technical University of Budapest, and there are some occasional contributors. Its primary application area is the simulation of communication networks; but because of its generic and flexible architecture, it is used in other areas like the simulation of complex

IT systems, queuing networks or hardware architectures, that is, any system that can be mapped to active components communicating by passing messages.

OMNeT++ provides a component-based, hierarchical, modular and extensible architecture. Components, or modules, are programmed in C++ and new ones are developed using the C++ class library which consists of the simulation kernel and utility classes for random number generation, statistics collection, topology discovery etc. New modules may be derived from basic object classes like module, gate or connection. A high-level language called Network Description (NED) is used to assemble individual components into larger components and models.

Alongside the simulation kernel library, the simulation environment contains a Graphical Network Editor (GNED), a NED compiler, graphical (Tkenv) and command line (Cmdenv) interfaces for simulation execution, graphical tools for simulation result analysis (Plove, Scalars), a model documentation tool, utilities (random number seed generation tool, etc.), documentation and sample simulations.

In OMNeT++, basic entity is a module. Modules can be atomic, which capture the actual behavior, or they can be composed of sub-modules. Modules communicate with each other by sending and receiving messages through their gates (equivalent to ports in J-Sim) which are linked to each other via connections. Gates can be associated with propagation delay, error rate and data rate, and they support only 1-to-1 correspondence. Connected modules form compound modules. Every simulation model is an instance of a compound module type, i.e. hierarchically nested modules. This level of components and topology is dealt with in NED files. Each module is able to generate, read or react to messages (also known as events).

As a feature-rich and powerful simulation tool, the simulator includes modules for Application Layer and Network Layer of OSI model as well as a Network Interface Card module which encapsulates MAC and PHY layers. Physical layer consists of one module for determining SNR characteristics and one module for deciding whether a packet can be passed to upper levels. A Mobility module provides and updates a node's current position and establishes communication channels.

Just like any discrete event simulator, OMNeT++ produces predictions at a low level which makes it accurate (but slow) to generate results. It does not only support deterministic modeling, but it also handles continuous and discrete stochastic variables that give randomness to models. There are available distributions for continuous random variables, and new ones can be defined in C++.

Some features of the simulator are failure modeling at specified times at runtime for nodes and links, built-in traffic scenarios with support for custom development, creation of any form of hierarchical topology using NED language and multiple open source pseudorandom number generators some of which are quite feasible for large scale simulations.

OMNeT++ has external extensions which enable it to provide support for simulation of wireless networks. Two most known and used extensions are INET Framework and Mobility Framework for mobile ad-hoc networks. One missing feature that may be required for wireless network research is energy models.

The simulator includes parallel simulation functionality (for executing simulations quicker and being able to model networks made of tens of thousands stations) through the use of either MPI or PVM3 communication libraries.

OMNeT++ has a well-written fairly large user manual, API documentation and some tutorials. Since the simulator has a complex structure, careful consultation of the available documentation is needed while working with it. INET Framework and Mobility Framework both has API documentation and Mobility Framework also has a user manual.

OMNeT++ has extensive GUI support. GNED allows building of network topologies graphically, therefore provides easy definition of network simulation scenarios. It can generate topologies as NED files. Tkenv supports interactive execution of the simulation, tracing and debugging. The user is able to start or stop the simulation execution in Tkenv and can change variables or objects inside the model at runtime. It provides the user with a detailed picture of the simulation state at any point of execution. For example, scheduled messages can be watched as the

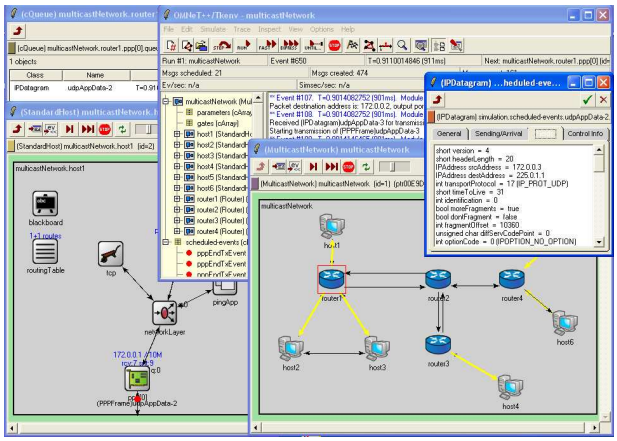

Figure 2: OMNeT++ GUI during simulation.

simulation progresses or simulation results can be displayed during execution. Whole environment can be customized while the simulation runs. Also, appearance of a node can be changed to reflect an inner state.

Cmdenv is designed primarily for batch execution of simulations. Plove visualizes output vector files of network simulations and it can plot one or more output vectors in a single graph. Similarly, Scalar visualizes output scalar files by creating bar charts or XY plots.

A welcome feature is that simulation executables created by the simulator are actually standalone programs that can be run on other machines without the simulator.

OMNeT++ is described by [4] as a well-organized, flexible, easy to use simulation environment with consistent object-oriented design; however, it is found to be poor in simulation result reporting, therefore users should code many useful measurements themselves.

[6] mentions that models already exist for telecommunications but OMNeT++ can be easily used and extended to simulate other kinds of infrastructures.

OMNeT++ works on Linux, Unix-like systems and Windows XP/2K.

4 ns-2

ns-2 is a discrete event simulator which provides support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. It is the most popular network simulator used by researchers.

The Network Simulator began as a variant of the REAL network simulator in 1989 and has evolved over the past years. In 1995, its development was supported by DARPA through the Virtual InterNetwork Testbed (VINT) project. Currently the development is carried out by Information Sciences Institute in California and is supported through DARPA and NSF.

ns-2 is written in C++ and is based on two languages: C++ which is used to extend the simulator (i.e. to define protocol behaviors), and OTcl [12], an object-oriented extension of Tcl developed by David Wetherall as part of the VUsystem project at Massachusetts Institute of Technology, which is used for scenario configuration, manipulation of existing C++ objects, periodic or triggered actions, etc.

ns-2 provides OSI layers excluding presentation and session layers. It has a huge pool of available features, offering a large number of external protocols already implemented. Behavior of the simulator is highly trusted within the network community.

The simulator has a complex structure. This makes adding new components a hard task since it requires a good knowledge of the simulator. While composing a simulation scenario, one needs to construct a binding between OTcl and the actual C++ classes by writing a corresponding OTcl class for each one of C++ classes.

ns-2 supports deterministic or probabilistic packet loss in queues attached to network nodes as well as it supports deterministic and stochastic modeling of traffic distribution. It also allows defining disturbances and corruptions that may occur in a simulated network like link interruptions, node stoppage and recovery by modifying scenario scripts.

In ns-2, one may create a topology for simulation using Inet Topology Generator [14], GT-ITM (Georgia Tech Internetwork Topology Models) topology generator [15] or Tiers Topology Generator [16] and

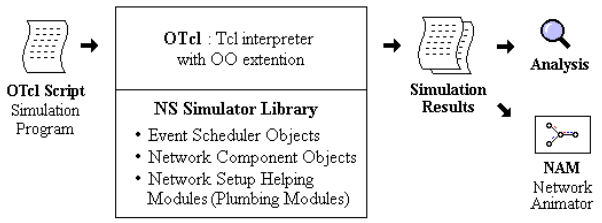

Figure 3: Simplified user's view of ns-2.

convert their outputs to ns-2 format. Of course, generation of topologies by hand is another option.

The simulation event scheduler of the simulator, contained in OTcl script interpreter, is either a non real-time scheduler or a real time scheduler which is mainly used for real-time synchronization of an emulated network. The user indicates in the event scheduler when network elements should start or stop transmitting packets.

With its emulation facility, ns-2 can be connected to a real network and capture live packets just like a common node. It can also inject packets into the live network.

The simulator can generate personalized trace files by allowing users to select parameters to be traced, therefore saves CPU resource.

Other features of the simulator include models for different network architectures including Wireless LAN, MANET and satellite, built-in traffic models with support for development of new ones, plugging of new pseudorandom number generators, and network state estimation .

Although ns-2 was primarily designed for simulating wired networks, it has the ability to simulate wireless networks using the CMU Monarch Project's Wireless and Mobility Extensions to NS [11] for IEEE 802.11 and several other extensions for Bluetooth. The simulator includes an energy model and it allows user to easily generate traffic and movement patterns. It also provides a set of randomized mobility models and there are several projects to bring advanced mobility models to the simulator which, therefore, bring realistic simulations. For the wireless part, mobile IP is also provided.

There are available extensions for ns-2 to allow parallel and distributed execution of simulations, like COMPASS (Composable and Parallel Simulation of Internetworks) project [13]. SNS project [21] applies staged simulation to ns-2 which improves the performance by identifying and eliminating redundant computations. It boasts a runtime 30 times faster than the original one.

ns-2 offers a comprehensive documentation and a regularly updated manual as well as an API for C++ and OTcl classes. There are also mailing lists and many web pages for its large user community which makes it easy to access information about the simulator.

In order to visualize a network simulation in ns-2, traffic and movement patterns should be generated and references as inputs into the OTcl code configuring the simulation scenario. The simulation can then be visualized by Nam [19] using the trace file generated by the simulator. However, [6] states that Nam can not be used for accurate simulation analysis. Node appearance can be changed according to node's inner state, but it is limited. Extended NamEditor [20] can be used to graphically create simple scenarios and save them as script files.

For statistics plotting, external tools like gnuplot [17] or xgraph [18] must be used. There is a function in OTcl configuration file which can periodically call itself to collect statistical data of simulation and write them into a file. The file generated by the function can be referenced as input into external tools for plotting of data.

One downside of ns-2 is that, ns-2 needs to be recompiled every time there is a change in the user code. As a solution to this problem, the authors of [1] compiled their user code into a separate shared library which is linked to the ns-2 kernel.

[2] and [6] complain about the simulator's inherent complexity, and high consumption of resources that leads to lack of scalability, preventing execution of simulations with more than a few hundred nodes. [2] also mentions it's lack of modularity.

ns-2 works on UNIX, Free BSD and Windows platforms.

5 OPNET Modeler

OPNET (Optimized Network Engineering Tools) Modeler [23] is a well-established commercial discrete-event simulator which can be used free of charge by researchers applying to University Program of the product. It was first proposed in 1986 and initially developed by Massachusetts Institute of Technology in 1987 using C++. It is the most widely used network simulator.

The simulator provides an environment for designing protocols and technologies as well as testing and demonstrating designs in realistic scenarios.

OPNET Modeler defines a network as a collection of sub-models representing sub-networks or nodes, therefore it employs hierarchical modeling. The topology used in a simulation can be manually created, imported or selected from the pool of predefined topologies.

A vast number of protocol models are available in OMNET Modeler and users can implement their own models. Models are developed in a hierarchical way using a four-level structure. Network level, which is not the OSI layer of the same name, handles topology modeling and overall configuration. Node level deals with internal structures of nodes like transmitters and receivers while functionalities of node level devices are modeled as finite state machines at the process layer. Proto-C layer, being the lowest layer, is where the coding of model behavior takes place in Proto-C language which is an extension of C. The layer contains a large number of kernel procedures available and it allows access to source codes of built-in models.

Nodes are configured by setting their parameters which define their internal structure as a set of fields or probability density functions. So, it is possible to implement stochastic models. Nodes contain a set of transmission and reception modules, representing a protocol layer or physical resource, to ensure their connection to communication links. Interactions between modules are handled by exchanging messages. Users are able to configure applications installed on a node, and set nodes and links to fail or recover during simulation at specified times.

Before simulation execution, one should make a selection of desirable output statistics. It is possible to specify a set of network simulations and pass a range of input parameters or traffic scenarios (which can be characterized by models for various applications like FTP, HTTP, etc.) to them. Statistics about performance of simulated networks can be collected at runtime.

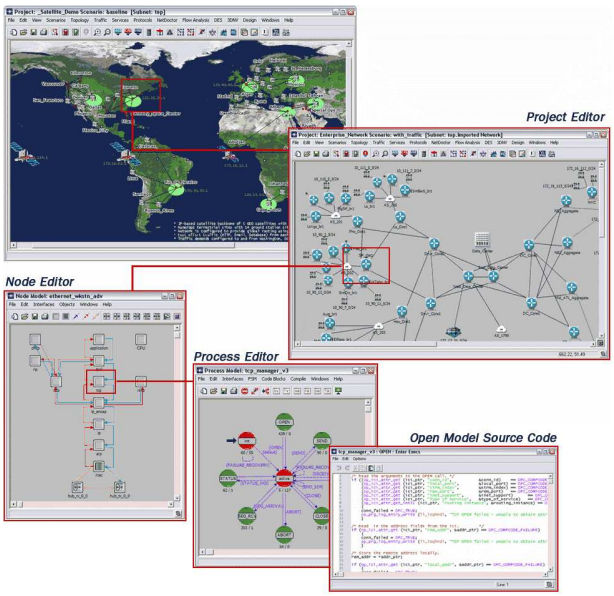

Figure 4: OPNET Modeler's editors shown hierarchically.

OPNET Modeler uses a pseudorandom number generator based on the random() function found in Unix BSD source. Like the one in J-Sim, this random number generator suffers from its low cycle length which is not enough to prevent repetition of randomness. This causes a potential problem with producing accurate final estimates of estimates with large variance (when a vast amount of events are needed to be simulated) and is a handicap for large scale network simulations.

The simulator uses an extension called the OPNET Modeler Wireless Suite to provide modeling, simulation, and analysis of wireless networks. It offers full protocol stack modeling capability with the ability to model all aspects of wireless transmis-

sions, including RF propagation, interference, transmitter/receiver characteristics, node mobility including handover, and the interconnection with wired transport networks. There is an admirable list of supported wireless networking technologies such as MANET, IEEE 802.11, 3G, Ultra Wide Band, IEEE 802.16, Bluetooth, and Satellite.

OPNET Modeler can execute several simulation scenarios in a concurrent manner. Also, it supports distributed simulation via an additional function called High-level Architecture which allows communication with various simulators, and system-in-the-loop simulation that provides communication to live hardware or software.

OPNET Modeler provides a good manual and an introductory tutorial. Also, technical support is effective in answering questions; however, it is not included in University Program, therefore should be purchased.

The Modeler has an advanced graphical interface which is used for model creation, simulation execution and data analysis. There are both built-in and external tools for statistics analysis. Graphical tools of the simulator are described below.

Probe Editor selects the statistics to be collected at runtime by letting users to place probes at arbitrary points in their network models. Global statistics, node statistics, attribute statistics and animation statistics are some statistics types collected by different probes.

Simulation Tool defines a sequence of simulation and is used to specify input/output options, simulation duration, simulation attributes, runtime options such as parallel simulation, etc.

Analysis Tool graphically displays simulation results, creates scalar graphs and can save analysis configurations for future use. Simulation results of several network scenarios belonging to the same project can be compared in the same graph. The simulator has a number of available filters to mathematically process, reduce or combine statistical simulation data and allows creation of new ones using Filter Editor.

An advantage of the simulator is that developers of the product take full responsibility for the credibility of simulation models.

OPNET Modeler, and its wireless suite, are available free of charge to researches who apply to their University Program. However, they require the completed work that is produced through the use of donated product to be free for all to use, and the participants to provide progress reports each year. They would also like to receive copies of published papers or theses upon completion of the project. The requirements are available in OPNET's website [23].

OPNET Modeler runs on Windows XP/2K, Linux and Solaris platforms.

6 Results of Comparisons and Surveys

It is for sure that each network simulator has its own strengths and weaknesses; therefore, each one is more suitable for some specific purpose than others. Our choice of a simulator is driven by our purpose of finding a well-balanced network simulator for the research of wireless networks. Conclusions drawn by the papers and surveys will effect our decision. This section is divided into two parts, first one of which sums up weaknesses and strengths of the simulators, and second one gives comparative information.

6.1 Weaknesses and Strengths

ns-2 has advantages of large number of available models [1], realistic mobility models [2], powerful and flexible scripting and simulation setup [4], large user community [1, 4], and ongoing development [4]. [2] pronounces the simulator as the wisest choice for the ones who are interested in the level of high-precision PHY layers. [2] also suggests consideration of ns-2 with stage simulation for large scale simulations. While [1] lists OTcl language and overall architecture of the simulator as its weak points, [4] extends this list by stating that some protocols and features are not well-documented, patching of extensions is not easy, and there is no clean separation between OTcl and C++.

The authors of [6] describe OMNeT++ as the most complete generic simulation tool, and specify that it includes a lot of functionalities. OMNeT++'s GUI support receives praise from [1] and [6]; however, [4]

asserts that model design GUI is not detailed enough to be useful and lists this as a shortcoming. [4] also complains about the still-evolving documentation and poor analysis of typical performance measures. The authors of [2] find mobility extension of the simulator fairly incomplete.

J-Sim has advantages of Java base [4], attractive flexible component-based architecture [1] and realistic mobility models [2]. On the downside, J-Sim has inadequate documentation and result reporting (many measurements should be custom coded) [4].

OPNET Modeler is powerful and user friendly, has a large library of simulation models, provides easy modeling, and has a good documentation [4]. It also has realistic mobility models [2] like J-Sim and ns-2. External tools are not supported since kernel is not open source as stated by [4] and the simulator is quite complex if a specific component has to be developed [6].

6.2 Comparisons

It is known by software architects that modeling everything completely is not achievable. On the other hand, it is important to pay attention to some details, absence of which has serious impact on results. Therefore, there should be a happy medium of granularity (the level of detail) for a software. [2] describes granularity of ns-2 as finest while J-Sim and OPNET Modeler are said to have fine granularity. OMNeT++'s granularity is medium.

According to [1], OMNeT++ and ns-2 are the most mature ones in a group of simulators including J-Sim and ShoX (a new simulator targeted at wireless networks from the beginning) and they both are productive. On the other hand, concerning the amount of effort it takes to become familiar with a simulator, the authors of [1] observed a clear order from ns-2, over OMNeT++ (OMNeT++ was said to be relatively easy to learn by [4]) to J-Sim and ShoX. This is probably caused by the feature richness of ns-2 and OMNeT++, especially regarding their scenario configuration capabilities. The authors of [6] say that J-Sim is similar to OMNeT++ but does not integrate as many features as OMNeT++. Common features are less efficient and OMNeT++'s GUI integrates more tools and functionalities than J-Sim.

[23] contains a comparison of ns-2 and OMNeT++ (It is a web source instead of a paper or survey). OMNeT++'s documentation is explained to be well-written and up-to-date while ns-2 documentation is explained to be fragmented. OMNeT++'s simulation API is said to be more mature and much more powerful than ns-2's. For ns-2, no clear dividing line is found between the models and the ns-2 simulation library.

[6] states that except for ns-2, the simulators (including OPNET Modeler, OMNeT++ and J-Sim) include a wide variety of features as well as a GUI for model and scenario definition and for simulation display and result analysis.

The authors of [5], who compared ns-2 and OPNET Modeler, conclude their study by explaining that both simulators provide very similar simulation results. They say that the complete set of OPNET Modeler modules provides more features than ns-2, and makes it more attractive to networks operators. Although the simulators had no significant problem in terms of accurately modeling the testbed behavior of simple constant bit rate data traffic, the simulators failed at adequately modeling the dynamic behavior of FTP. Only OPNET Modeler managed to produce more accurate results with fine tuning.

For the simulation of wireless sensor networks, [3] presents a summary of general purpose simulation frameworks. The authors express that J-Sim has a rich wireless sensor network support (with environment and battery models), and very detailed sensor extension while ns-2 has support for some common wireless sensor network protocols. OMNeT++'s Mobility Framework is also mentioned for its possibility to be used for sensor simulation.

Finally, ns-2 and OMNeT++ can be linked to Akaroa2 [4, 25], which runs multiple instances of an ordinary serial simulation program simultaneously on different processors. These instances run independently of one another, and continuously send back observations of the simulation model parameters which are of interest to a central controlling process. The central process calculates from these observations an overall estimate of the mean value of each parameter.

Hence, Akaroa2 brings more speed and reliability to the simulators.

7 Conclusion

In this survey, four network simulators that support simulation of wireless networks (J-Sim, OMNeT++, ns-2 and OPNET Modeler) have been described, and their strengths and weaknesses have been depicted based on a couple of papers and surveys. The purpose was to find a wireless network simulator that would provide a researcher with an efficient and easy-to-use development environment to benefit his or her research. As seen from the survey, simulators have a long list of features to be considered, and none of them offer good support for all of those features. Therefore, what had been searched for here was actually a balanced simulator that would offer a decent user experience.

From the descriptions and the conclusions drawn by the reference papers, it can be concluded that ns-2 and OMNeT++ are the best choices for research. ns-2, the most popular simulator for academic research, is generally criticized for its complicated architecure. But, its large use community makes up for it since there are lots of people helping each other with their problems through the use of mailing lists and forums. OMNeT++ is gaining popularity in academic and industrial world. Unlike ns-2, OMNeT++ has a well-designed simulation engine and supports hierarchical modeling, so it is better for development. Also, OMNeT++'s powerful GUI gives it a certain edge. However, OMNeT++ lacks the abundance of external models and user base ns-2 has. OPNET Modeler is also a good, complete solution; but, it caters to industrial researchers, people who need an extensive set of built-in reliable models for constructing credible simulations in a quick way, rather than academic researchers.

References

[1] J. Lessmann, P. Janacik, L. Lachev, and D. Orfanus. *Comparative Study of Wireless Network Simulators.* The Seventh International Conference on Networking, pages 517-523, 2008. IEEE.

[2] J. L. Hogie, P. Bouvry, and F. Guinand. *An Overview of MANETs Simulation.* In Electronic Notes in Theoretical Computer Science, Proc. of 1st International Workshop on Methods and Tools for Coordinating Concurrent, Distributed and Mobile Systems (MTCoord 2005), LNCS, pages 81101, April 2005. Elsevier.

[3] E. Egea-Lopez, J. Vales-Alonso, A. Martinez-Sala, P. Pavon-Mari, and J. Garcia-Haro. *Simulation Scalability Issues in Wireless Sensor Networks.* IEEE Communications Magazine, 44(7):6473, July 2006.

[4] L. Begg,W. Liu, K. Pawlikowski, S. Perera, and H. Sirisena. *Survey of Simulators of Next Generation Networks for Studying Service Availability and Resilience.* Technical Report TR-COSC 05/06, Department of Computer Science & Software Engineering, University of Canterbury, Christchurch, New Zealand, February 2006.

[5] G. F. Lucio, M. Paredes-Farrera, E. Jammeh, M. Fleury, and M. J. Reed. *OPNET Modeler and ns-2 - Comparing the Accuracy of Network Simulators for Packet-level Analysis Using a Network Testbed.* WSEAS Transactions on Computers, 2(3):700707, July 2003.

[6] S. Duflos, G. L. Grand, A. A. Diallo, C. Chaudet, A. Hecker, C. Balducelli, F. Flentge, C. Schwaegerl, and O. Seifert. *Deliverable d 1.3.2: List of Available and Suitable Simulation Components.* Technical report, Ecole Nationale Supieure des Tommunications (ENST), September 2006.

[7] M. Fleury, G. F. Lucio and M. J. Reed. *Clarification of the "OPNET NS-2 Comparison" Paper with regards to OPNET Modeler.* July 2003.

[8] *J-sim Official.* http://sites.google.com/site/jsimofficial/

[9] *OMNeT++ Community Site.* http://www.omnetpp.org/

[10] *The Network Simulator - ns-2.* http://www.isi.edu/nsnam/ns/

[11] *The Rice University Monarch Project: Mobile Networking Architectures.* http://www.monarch.cs.cmu.edu/

[12] *OTcl.* http://otcl-tclcl.sourceforge.net/otcl/

[13] *COMPASS Project.* http://www.cc.gatech.edu/computing/compass/

[14] *Inet Topology Generator.* http://topology.eecs.umich.edu/inet/

[15] *Modeling Topology of Large Internetworks.* http://www.cc.gatech.edu/fac/Ellen.Zegura/graphs.html

[16] *Tiers Topology Generator.* http://www.isi.edu/nsnam/dist/topogen/tiers1.0.tar.gz

[17] *Gnuplot Homepage.* http://www.gnuplot.info/

[18] *Xgraph.* http://www.isi.edu/nsnam/xgraph/index.html

[19] *Nam: Network Animator.* http://www.isi.edu/nsnam/nam/

[20] *Extended NamEditor.* http://www.grid.unina.it/grid/ExtendedNamEditor/

[21] *SNS: Staged Simulation in ns-2.* http://www.cs.cornell.edu/people/egs/sns/

[22] *J-Sim Users, Google Groups.* http://groups.google.com/group/j-sim-users/browse_thread/thread/2f533783bccf4932

[23] *OPNET Technologies, Inc.* http://www.opnet.com/

[24] *OMNeT++ Vs ns-2: A comparison.* http://ctieware.eng.monash.edu.au/twiki/bin/view/Simulation/OMNeTppComparison

[25] *Project Akaroa.* http://www.cosc.canterbury.ac.nz/research/RG/net_sim/simulation_group/akaroa/about.chtml

[26] *NS by Example.* http://nile.wpi.edu/NS/