ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 2, Issue 4, April 2013

Open-Source Network Simulation Tools: An Overview

Suraj G. Gupta, Mangesh M. Ghonge, Parag D. Thakare, Dr. P. M. Jawandhiya

Abstract- **In the network research area, implementation of a whole network in real world is not easily possible because establishing of network in a real world is very difficult. A single test bed containing multiple networked computers, routers and data links to validate and verify a certain network protocol or a specific network algorithm that needed a large amount of time and cost. The simulator helps the network developer to check whether the network is able to work in the real world. Thus both the time and cost of testing the functionality of the network have been reduced and implementations are made easy. 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. In this paper, we discuss on different open-source simulators, present a comprehensive survey on current open-source network simulators and introduce the main features of different open-source network simulator and consider their advantages and disadvantages. We hope this survey for those people who feel difficult to select the appropriate open-source network simulators for their research.**

Keywords: Open-Source Technology, Network Simulation, Network Simulator, NS2, NS3, OMNeT++ and J-Sim

I. INTRODUCTION

Simulation is a very important modern technology. The simulation in computer can model hypothetical and real-life objects on a computer so that it can be studied. [1] It can be applied to different science, engineering, or other application fields for different purposes [3]. Computer simulation can be used to assist the modeling and analysis in many natural systems. A network simulator is a technique of implementing the network on the computer. Through this the behavior of the network is calculated either by network entities interconnection using mathematical formulas or by capturing and playing back observations from a production network [1].

"The Network Simulator provides an integrated, versatile, easy-to-use GUI-based network designer tool to design and simulate a network with SNMP, TL1, TFTP, FTP, Telnet and Cisco IOS device."[24]

Network simulator allows the researchers to test the scenarios that are difficult or expensive to simulate in real world. It's particularly useful to test new networking protocols or for changes the existing protocols in a controlled and reproducible environment. One can design different network topologies using various types of nodes (hosts, hubs, bridges, routers and mobile units etc.). Afterwards, the routing behavior can be easily studied in different topologies, given the fact that the network topology is merely a set of simulation parameters. Most available networks simulation toolkits are based on the paradigm of discrete event-based simulation [2] [25].

Advantages: [19]

- Sometimes cheaper
- Find bugs (in design) in advance
- Generality: over analytic/numerical techniques
- Detail: can simulate system details at an arbitrary level

Disadvantages: [19]

- Caution: does model reflect reality
- Large scale systems: lots of resources to simulate (especially accurately simulate)
- May be slow (computationally expensive – 1 min real time could be hours of simulated time)
- Art: determining the right level of model complexity
- Statistical uncertainty in results

The network simulators are of different types which can be compared on the basis of: range (from the very simple to the very complex), specifying the nodes and the links between those nodes and the traffic between the nodes, specify everything about the protocols used to handle traffic in a network, graphical applications (allow users to easily visualize the workings of their simulated environment.), text-based applications (permit more advanced forms of customization) and programming-oriented tools (providing a programming framework that customizes to create an application that simulates the networking environment to be tested.) [1] There are different network simulators with different features. Some of the open source network simulators are NS2, NS3, OMNeT++ and J-Sim. In this paper we are working on some of the simulator.

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.

II. OPEN-SOURCE SOFTWARE

Open source software as software that comes under an open source license. This implies that every recipient of the software can freely study its source code, modify it, and distribute it. [21]

This software typically does not require a license fee. There are open source software applications for a variety of different uses such as office automation, web design, content management, operating systems, and communications. Future uses of open source where could open source be used and how could its development processes help technology projects be more productive [20].

OSS is unique in that it is always released under a license that has been certified to meet the criteria of the Open Source Definition. These criteria include the right to: [23] [22]

- Redistribute the software without restriction;
- Access the source code;
- Modify the source code; and
- Distribute the modified version of the software.

Advantages of Open Source Software: [23] [22]

- Costs (little or no-charges)
- Flexibility
- Reliability and Quality
- Reduces "Vendor Lock-in"
- Availability of External Support
- Every one finds bugs

Some possible limitations: [23] [22]

- Lack of Personalized Support i.e. Documentation
- Restricted Choice (selected option).
- Speed of Change or Software Development.
- No warranty.

III. OPEN-SOURCE NETWORK SIMULATORS

A. NS2 (Network Simulator version2)

NS2 is a discrete event simulator targeted at networking research. It provides support for simulation of TCP, routing, and multicast protocols over all networks (wired and wireless). Network simulator 2 has been developed under the VINT (Virtual Inter Network Testbed) project; in 1995 it is a joint effort by people from the University of California at Berkeley, University of Southern California's Information Sciences Institute, Lawrence Berkeley National Laboratory and Xerox Palo Alto Research Center. The main sponsors are the Defense Advanced Research Projects Agency and the National Science Foundation. It is a discrete event simulator that provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless networks [1]. The Monarch Project provides various modules for (mobile) wireless network simulation; e.g., radio propagation models, the IEEE 802.11 MAC protocol, mobility models, different ad-hoc routing protocols (e.g., AODV and DSR) and Mobile IP.

SensorSim [26, 27, 28], from UCLA, has further extended NS-2 by including the support for sensor network simulation [2] [3] [4].

B. NS3 (Network Simulator version3)

The NS-3 simulator is a discrete-event network simulator for Internet systems, targeted primarily for research and educational use. The NS-3 project, started in 2006, is an open-source project developing NS-3. Ns-3 is free software, licensed under the GNU GPLv2 license. It will rely on the ongoing contributions of the community to develop new models, debug or maintain existing ones, and share results [1] [2] [3] [4].

The major difference between NS3 and NS2: [4]

- Different software core: The core of NS3 is written in C++ and with Python scripting interface (compared with OTcl in NS2). Several advanced C++ design patterns are also used.
- Attention to realism: protocol entities are designed to be closer to real computers.
- Software integration: support the incorporation of more open-source networking software and reduce the need to rewrite models for simulation;
- Support for Virtualization: lightweight virtual machines are used. Figure 3 gives an example Virtualization test bed of NS3.
- Tracing architecture: NS3 is developing a tracing and statistics gathering framework trying to enable customization of the output without rebuilding the simulation core.

Through the comparison between NS2 and NS3, summarize the NS 3 features as follows: [4]

- Modular, documented core
- C++ programs and Python scripting
- Alignment with real systems
- Software integration
- Virtualization and test bed integration
- Attribute system
- Updated models

C. OMNET++ (Optical Micro-Networks Plus Plus)

OMNeT++ is not a network simulator by definition, but a general purpose it is a component-based, modular and open architecture discrete event based simulation framework. Yet it is mostly applied to the domain of network simulation, given

the fact that with its INET package it provides a comprehensive collection of Internet protocol models. OMNeT++ simulations consist of so-called simple modules which realize the atomic behavior of a model. It is an extensible, modular, component-based C++ simulation library and framework, primarily for building network simulators. The most common use of OMNeT++ is for simulation of computer networks, but it is also used for queuing network simulations and other areas as well. It is licensed under the own Academic Public License, which allows the GNU Public License like freedom but only in noncommercial settings [1] [2] [3].

D. JSIM (Java-based simulation)

It is a Java-based simulation system for building quantitative numeric models and analyzing them with respect to experimental reference data. JSim is an application development environment based on the component-based software architecture. This framework is built upon the autonomous component architecture (ACA) and the extensible internetworking framework (INET) of J-Sim. JSim has been developed by a team at the Distributed Real-time Computing Laboratory (DRCL). The project has been sponsored by the National Science Foundation (NSF), DARPA's Information Technology Office, Air Force Office of Scientific Research's Multidisciplinary University Research Initiative, the Ohio State University and the University of Illinois at Urbana-Champaign. J-Sim is free and available with source code [1] [2].

IV. Language Used and Component Diagram of Open-Source Simulators

In this section we are defining the languages that are used by different network simulators and Component Diagram of simulators.

A. NS2:

C++: C++ code, which is fast to run but slow to change, making it suitable for detailed protocol implementation. C++ code is used to model the behavior of the simulation nodes [17].

Article: OTcl runs much slower but can be changed very quickly (and interactively), making it ideal for simulation configuration. NS provides glue to make objects and variables appear in both languages.

The component diagram of ns2 is given herein fig-1, OTcl scripts that control the simulation and specify further aspects, for instance the network topology [1] [17].

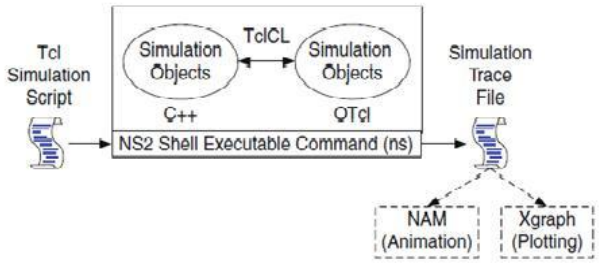

Fig-1: Basic architecture of NS [16]

B. NS3:

C++: C++ used for implementation of simulation and core model. Ns-3 is built as a library which may be statically or dynamically linked to a C++ main program. These libraries define the start of simulation and simulation topology.

Python: C++ wrapped by Python. Python programs to import an "ns3" module.

The component diagram of ns3 is given in fig-2.

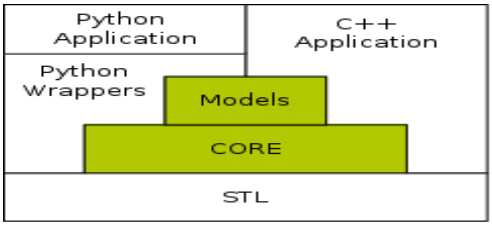

Fig-2: Architecture of NS3 [45].

C. OMNET++:

C++: A C++ class library which consists of the simulation kernel and utility classes (for random number generation, statistics collection, topology discovery etc) this one you will use to create simulation components (simple modules and channels); infrastructure to assemble simulations from these components and configure them (NED language, ini files); runtime user interfaces or environments for simulations (Tkenv, Cmdenv); an Eclipse-based simulation IDE for designing, running and evaluating simulations; extension interfaces for real-time simulation, emulation, MRIP, parallel distributed simulation, database connectivity and so on.[1][29].

The component diagram of OMNET++ is given in Fig-3 [29].

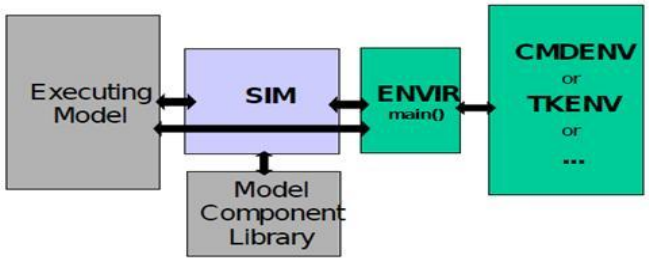

Fig-3: Architecture of OMNeT++ simulation programs

D. JSIM:

Java: Java is easy to learn and easy to use. In case of any problems, source texts provided with J-Sim can be used to generate new code, compiled in the target environment, thus 100-percent compatible with JVM used. Java provides a class called Thread whose instances run parallel with other such instances. Java is a fully object-oriented language, providing the concepts of classes, instances, encapsulation, inheritance and polymorphism. JSim provides basic classes for simulation, process and queue. These classes can be either directly used or extended according to specific user's requirements.

Tcl: Scripting is an essential part of J-Sim, use it to "glue" all the components and define how the system operates. It makes it possible to manipulate Java objects in the Tcl environment, such as creating an object from a Java class, invoking a method of a Java object, or accessing a field variable of a Java object.

Name of Simulator	Language	Platform	Cost & licenses	Network Support Type	User interface	API
NS2	C++, Otcl	Windows, Linux	Free, Open Source	Wired Network, Wireless Ad-Hoc mode, Wireless Managed mode, Wired cum Wireless, Cannot simulate problems of the bandwidth or the power consumption in Wireless Sensor Network	Command Line Interface	Pure Event Base
NS3	C++, Python	Windows, Linux, Mac OS	Free, GNU General Public License	Wired Network, Wireless Network, Wireless Sensor Network	Command Line Interface	low-level, users can mix and match between the simpler API
OMNeT++	C++	Windows, Unix-Based, Mac OS X 10.6 and 10.7	Free, Noncommercial license, Commercial license	Wired Network, Wireless Managed mode,	Graphical User Interface	Event Base
JSIM	Java, Tcl	Truly platform independent	Free, Open Source	Wired Network, Wireless Network, Wireless Sensor Network, radio channels and power consumptions	GUI, Command line interface on Linux	Completely Process Driven including Thread Synchronization

Table-1: Comparison of different open source simulator (NS2, NS3, OMNeT++ and J-Sim).[1][2][3][4][5][7][18][6][35][37][39] [41][43][45] [46][47].

I. MERIT, DEMERIT, RECENT DEVELOPMENTS AND ITS FUTURE SCOPE

A. NS2:

- NS-2 can be used for parallel and distributed simulation: PDNS. NS-2 provides emulation functionalities.
- NS-2 comes closer to reality than other simulators [18].
- A good simulation design, good results can be achieved with NS-2 [18].
- If the simulator has to come at no charge or the GUI is only optional, then NS-2 is the right choice [18].
- NS-2 has the rich collection of models than others simulators [18].
- A major shortcoming of NS-2 is its limited scalability [30, 31] in terms of memory usage and simulation run-time that as new research domains in the field of computer networks, such as wireless sensor networks (WSNs), peer-to-peer networks or grid architectures and very large networks, potentially with hundreds of thousands of nodes [3].
- To face those challenges, a couple of enhancements of NS-2 have been proposed, for instance the incorporation of parallelization [32].
- The memory usage performances of NS-2 linear growth of memory usage [3].
- The rich collection of models for ns-2 still needs to be ported from ns- 2 to ns-3, OMNeT++ can be considered as viable alternative [3].

B. NS3:

- Users of ns-3 can construct simulations of computer networks using models of traffic generators, protocols such as TCP/IP, and devices and channels such as Wi-Fi, and analyze or visualize the results [1].
- NS-3 is an active open-source project and it is still under development [1].
- Ns-3 is not an extension of NS-2; it is a new simulator. The two simulators are both written in C++ but ns-3 is a new simulator that does not support the ns-2 APIs. Some models from ns-2 have already been ported from ns-2 to ns-3. The project will continue to maintain NS-2 while NS-3 is being built, and will study transition and integration mechanisms.
- The first stable version of NS-3 (release candidate 1) was released and considered as an eventual replacement for NS-2. [18]
- NS-2 is currently undergoing a major redesign [30]. One of the main development goals of its successor, NS-3, is the improvement of simulation performance [3].
- NS-3 still is in the early stages, and just a few simulation models exist which one can use off the shelf. NS-3 does not have all of the models that NS-2 currently has [3] [4].
- NS-3 does have new capabilities (such as handling multiple interfaces on nodes correctly, use of IP addressing and more alignment with Internet protocols and designs, more detailed 802.11 models, etc.).
- NS-3 integrates architectural concepts and code from GTNetS[33], a simulator with good scalability characteristics [3].
- NS-2 models need to be ported to NS-3 in a manual way [3].
- NS-3 being the most efficient simulation tool in memory usage performances [2] [3].
- Capable of carrying out large-scale network simulations in an efficient way [2].
- The feature set of the simulator is also about to be extended. For example, NS-3 is slated to support the integration of real implementations' code by providing standard APIs, such as Berkeley sockets or POSIX threads, which are transparently mapped to the simulation [34] [3].
- NS3 needs a lot of specialized maintainers in order to let the NS3 have the advantages as the commercial OPNET network simulators which are documented well [4].
- NS3 is intended to replicate the successful mode of NS 2 in which a lot of different organizations contributed to the models and components based on the framework of NS2 [4].
- The support available on the user mailing list, and the developer and maintainer activity, are higher for NS-3.
- Limitations of simulations, in general, are that it often suffers from lack of credibility, NS3 simulation solve this problem with the help of following four points [4].

1) Hosting NS3 code and scripts for published work.
2) Tutorials on how to do things right.
3) Flexible means to configure and record values.
4) Support for ported code should make model validation easier and more credible.

- If scalability is the main concern, NS-3 and OMNeT++ are smart choices [2].

C. OMNET++:

- OMNeT++ is a C++-based discrete event simulator for modeling communication networks, multiprocessors and other distributed or parallel systems [1].
- OMNeT++ modules are reusable and can be combined in various ways which is one of the main features [4].
- OMNeT++ has generic and flexible architecture which makes it successful also in other areas like the IT systems, queuing networks, hardware architectures [4].
- OMNeT++ is not recommended because most scenarios could not be implemented in the simulator as certain features are still missing. Relatively young and only few simulation models [18] [25].
- OMNeT++ performed well in the scenarios that could be implemented [18].
- GUI is mandatory, used OMNeT++ [18].
- Apparently some of the simulations could not be performed in OMNeT++ [18].
- The memory usage performances of OMNeT++ linear growth of memory usage [3].
- Well structured, highly modular, not limited to network protocol simulations (e.g., like ns2) [25].
- Capable of carrying out large-scale network simulations in an efficient way [3].
- OMNeT++ can run on Linux, other UNIX-like systems and on Windows (XP, Win2K), MAC-OS [11].
- Several open source simulation models have been published in the field of network simulations such as IP, IPv6, MPLS, mobility and ad-hoc simulations [4].
- OMNeT++ can have greater development if it could persuade more organizations to participate in and to contribute [4].

D. JSIM:

- JSIM, a Java-based simulation and animation environment supporting Web-Based Simulation, a rapidly emerging area of simulation research and development [25].
- Simulating several typical WSN scenarios in J-Sim and NS-2. The simulation study indicates that the proposed WSN simulation framework in J-Sim is much more scalable than ns-2.[2]
- J-Sim provides a loosely-coupled component architecture, i.e., a component can be individually designed, implemented and tested independently [2][35][36].
- Wireline and wireless network component and protocols have been implemented in J-Sim [2].
- J-Sim a truly platform independent, extensible, and reusable environment [2].
- Chosen J-Sim as the base environment to be augmented with a simulation framework for WSNs [2].
- WSN framework in J-Sim exhibits good scalability [2].
- J-Sim incurs a comparable though longer execution, even though the number of events generated is almost the same because Java program is inherently slower than a C/C++ program [2].
- If scalability is the main concern, ns-3 and OMNeT++ are smart choices [2].

VI. CONCLUSION

In this paper, we have presents the general overview on open source network simulators for those who are not familiar with it. In this paper we gave brief introduction of some key concepts Open-Source Technology, Network Simulation and different open source network simulator, we analyzed and discussed advantages, disadvantages and future scope of Open-Source Technology and its used, it is more efficient for academia, industries, new researchers and student, it seems that the easier for the users to learn and to use, then we introduce four different open source network simulators NS2, NS3, OMNeT++ and J-Sim. We also analyzed and discussed current feature, merits, demerits, challenges, future scope of all simulators. Currently NS2 is best option, for GUI interested OMNeT++, for large network simulation remain three are better than ns2 but reaming three are on development stage and they are not cover all models like ns2 but special type simulation they can perform like jsim best for wireless sensor

network simulation . Popularity of ns2 is more than others but may be in future ns3 will be replace ns2.

REFERENCES

[1] Mrs. Saba Siraj, Mr. Ajay Kumar Gupta, Mrs Rinku-Badgujar, "Network Simulation Tools Survey", International Journal of Advanced Research in Computer and Communication Engineering Vol. 1, Issue 4, June 2012, ISSN : 2278 – 1021

[2]Ahmed Sobeih, Wei-Peng Chen, Jennifer C. Hou, Lu-Chuan Kung, Ning Li, Hyuk Lim, Hung-Ying Tyan, and Honghai Zhang, "J-Sim: A Simulation Environment forWireless Sensor Networks", Proceedings of the 38th Annual Simulation Symposium (ANSS'05), 2005 IEEE

[3] Elias Weing¨artner, Hendrik vom Lehn and Klaus Wehrle, "A performance comparison of recent network simulators",

[4] Jianli Pan. "A Survey of Network Simulation Tools: Current Status and Future Developments",

[5]Hung-ying Tyan, "Design, Realization and Evalution of a Component-Based Compositional Software Architecture for Network Simulation" dissertation, Ohio State University, 2002

[6] http://j-sim.cs.uiuc.edu/drcl.inet/ex_echoer.html

[7] ns-3 project, "ns-3 simulator", ns-3 Tutorial Release ns-3.15, November 13, 2012

[8] ns-3 project, "ns-3 simulator", ns-3 Model Library Release ns-3.15, November 13, 2012

[9] ns-3 project, "ns-3 simulator", ns-3 Manual Release ns-3.15, November 13, 2012

[10] "OMNeT++ IDE Customization Guide Version 4.2.2", 2011 Andras Varga and OpenSim Ltd.

[11] "OMNeT++ Installation Guide Version 4.2.2", 2011 András Varga and OpenSim Ltd.

[12] "OMNeT++ User Guide Version 4.2.2", 2011 Andras Varga and OpenSim Ltd.

[13] http://www.odu.edu/engr/networking/Tools.html

[14] Virginia Tech, "Beginner's Guide to ns2 – Installation and Basic Usage", ECE 5984: Network Performance, Design, and Management, springer 2002.

[15] "NS-2 Tutorial (1)", Multimedia Networking Group, the Department of Computer Science, UVA Jianping Wang, 2004.

[16] Teerawat Issariyakul, Ekram Hossain, "Introduction to Network Simulator NS2", 2009 Springer Science+Business Media, LLC, ISBN: 978-0-387-71759-3.

[17] Kevin Fall, Kannan Varadhan, "The ns Manual (formerly ns Notes and Documentation)", The VINT Project A Collaboration between researchers at UC Berkeley, LBL, USC/ISI, and Xerox PARC, November 4, 2011.

[18] Karsten M. Reineck. "Evaluation and Comparison of Network Simulation Tools", Master Thesis. 29th August 2008

[19] http://www.nsnam.org/overview/key-technologies/

[20] "OPEN SOURCE", http://www.parliament.uk/post, September 2004

[21] Joachim Henkel, "Open Source Software from Commercial Firms – Tools, Complements, and Collective Invention", Munchener betriebswirtschaftlich -e Beitrage Munich Business Research 2002-07, Revised version, May 2003.

[22] Ministry of Economic Development and Trade, "Open Source Software", Queen's Printer for Ontario, 2010, www.ontario.ca/economy.

[23] Josh Lerner, Jean Tirole, "THE SCOPE OF OPEN SOURCE LICENSING", NBER Working Paper No. 9363 December 2002 JEL No. O3, K3.

[24]http://www.webnms.com/simulator/network-simulator-ds.html

[25] J-Sim. http://www.j-sim.org/.

[26] SensorSim: A simulation framework for sensor networks. http://nesl.ee.ucla.edu/projects/sensorsim/.

[27] S. Park, A. Savvides, and M. Srivastava. "Simulating networks of wireless sensors", In Proc. of the 2001 Winter Simulation Conference.

[28] S. Park, A. Savvides, and M. Srivastava. "SensorSim: A simulation framework for sensor networks", In Proc. of the ACM international Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems, 2000.

[29] "OMNeT++ User Manual Version 4.2.2", 2011 Andras Varga and OpenSim Ltd.

[30] T. R. Henderson, S. Roy, S. Floyd, and G. F. Riley. "ns-3 project goals. In WNS2" 06: Proceeding from the 2006 workshop on ns-2: the IP network simulator, page 13, New York, NY, USA, 2006. ACM.

[31] Y. Xue, H. S. Lee, M. Yang, P. Kumarawadu, H. Ghenniwa, and W. Shen. "Performance evaluation of ns-2 simulator for wireless sensor networks". Proceedings of the Canadian Conference on Electrical and Computer Engineering (CCECE 2007), pages 1372–1375, April 2007.

[32]G. Riley. PDNS project website http://www.cc.gatech.edu/computing/compass/pdns/.

[33] G. Riley. "Large scale network simulations with GTNetS", I Proceedings of the 2003 Winter Simulation Conference, 2003.

[34] ns-3 Overview (June 2008).

[35] J-Sim. http://www.j-sim.org/.

[36] H.-Y. Tyan and J. C. Hou. "JavaSim: A component-based compositional network simulation environment", In Proc. of Western Simulation Multi conference, CNDS'01.

[37] http://en.wikipedia.org/wiki/Network_simulation

[38] http://www.webnms.com/simulator/network-simulator-ds.html

[39] http://www.omnetpp.org/

[40] https://sites.google.com/site/jsimofficial/j-sim-tutorial

[41] http://physiome.org/jsim/

[42] http://www.icir.org/models/simulators.html

[43] Fei Yu, "A Survey of Wireless Sensor Network Simulation Tools", http://www.cse.wustl.edu/~jain/ cse567- 11/ftp/sensor/index.html

[44] http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf

[45] http://www.nsnam.org/

[46] http://j-sim.cs.uiuc.edu/drcl.inet/inet_tutorial.html

[47] http://www.cs.cornell.edu/skeshav/real/ overview. Html