ISSN No. 0976-5697

Volume 3, No. 5, Sept-Oct 2012

International Journal of Advanced Research in Computer Science

CASE STUDY AND REPORT

Available Online at www.ijarcs.info

A Comparative Study of Simulation tools for implementing and improving energy efficiency of a Wireless Sensor Node

R.K.Nadesh*
Assistant Professor Senior, SITE,
VIT University Vellore, India
rknadesh@vit.ac.in

Varsha Singh
U.G.Scholar, SITE
VIT University, Vellore, India
varsha162@gmail.com

Apoorva Kishore Malewar
U.G.Scholar, SITE
VIT University, Vellore, India
mk.apoorva@gmail.com

Abstract: Wireless Sensor Networks is a novel field in the realm of research. These networks contain small battery driven nodes that are used to survey a physical area without constant human supervision. These nodes gather information regarding the physical environment, process it and send it back to the parent base station. Various simulation tools can be used to test algorithms and compare results rather than testing them as a real time experiment which is costlier and more time consuming. This paper contains a study of the various tools available which can be used to simulate WSN such as NS2, Omnet++ etc.

Keywords: Avrora, NS2, J-Sim, Omnet++, OPNET Modeler, TOSSIM, Wireless Sensor Networks.

I. INTRODUCTION

Recently there have been many advances in the areas of distributed and wireless networks. These require the use of low energy consuming devices that can work efficiently and also posses processing abilities. Wireless Sensor Networks (WSN) is one field that provides this functionality.

Wireless Sensor Networks consists of a number of autonomous devices that are used to keep a watch on the environmental conditions, such as temperature, humidity, motion or pollutants, pressure, sound and to collectively pass data through the network to the base station. These devices required to use optimal amount of power as they have a small battery life [1]. They also need a robust routing algorithm to navigate their data as they have minimal human supervision. WSN have a dynamic topology of network with static or mobile nodes that continuously interact with each other, allowing heterogeneity of nodes. It is also scalable yet providing ease of use. These nodes are designed to handle harsh environment conditions.

WSN was developed for military surveillance purposes, such as in the battlefield or to study the enemy camps. Later they were also used to study traffic patterns, weather monitoring and reporting. They are also used in the field of medicine, such as in hospital automation or medical assistance in emergency or in remote areas [1]. They are used to detect natural disasters, track animal life in the wild, monitor air pollution, data logging, machine health monitoring, and agriculture, waste water monitoring, and also tracking of materials. In structural health monitoring, you can use wireless sensors to effectively monitor highways, bridges, and tunnels. You also can deploy these systems to continually monitor office buildings, hospitals, airports, factories, power plants, or production facilities [2].

As the uses of WSN increase, they are being modified and tested for a lot of different functionalities. Running real time experiments on actual physical setting is very difficult as a lot of tests are run to reach perfection and some of them might not give the desired output. This can be very costly in the matter of finance, time and effort [3]. Many factors affect the output of a test run and isolating a single fault can be very tedious. Also the sensor node devices are expensive to set up and require a certain level of expertise. Thus simulation is very important in the area of WSNs [4].

Simulators are hence used to develop and test new routing algorithms, protocols, and applications, before implementing them in real time. They allow the developers to locate and isolate the errors in a short amount of time. They can also work towards improving a certain aspect of the product until the preferred results are achieved. This maximizes efficiency and promises better usability [5].

II. SIMULATORS

A. Network Simulator 2- NS2:

a. Overview:

NS2 is the most popular network simulator used by researchers. It is an open source discrete event simulator that can be used to simulate both wired and wireless networks. It supports a large number of protocols used for routing, providing simulations for both static and dynamic networks. NS (Network Simulator) is originally based on REAL network simulator. NS2 is the second version of NS. The first version of NS, developed in 1989 has evolved a lot over the years. In 1995, NS project was supported by DARPA through the Virtual Inter Network Testbed (VINT) project. Information Sciences Institute in California is currently developing the tool and is being supported through DARPA and NSF.

b. Features:

NS2 is an object-oriented network simulator developed in C++. Is supports simulation using C++, which is used to build protocols and OTcl. OTcl which basically is Tcl (Tool Command Language) with object-oriented extensions was developed by David Wetherall at Massachusetts Institute of Technology. It is used for manipulation of existing C++ objects and for scenario development and configuration purposes. The combination of these two languages proves to be very effective, as OTcl has the feature that C++ lacks. C++ is a very efficient language to define and develop protocols for routing, but it is difficult to visualize it and be represented graphically. It is difficult to gather connect modules and change parameter values in C++ visually. OTcl makes up for these lacking features. Thus C++ is used to implement the routing protocol and OTcl is used to control the simulation scenario and schedule the events. NS2 separates control path implementations from the data path implementation and hence is more efficient [6].

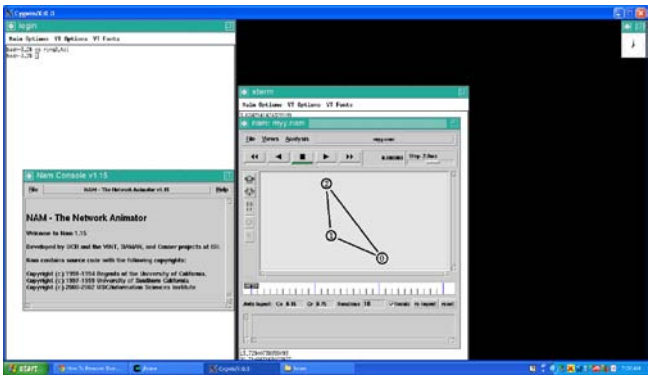

Figure 1: Xterm of NS2

c. Merits:

a) NS2 contains a variety of protocols in all the layers, especially the WSN and ad-hoc protocol.
b) It is open source and hence online example codes can be used as a part of component development.
c) It is object-oriented hence it has features like encapsulation, abstraction, modularity, inheritance, and ease of use.

d. Demerits:

a) It uses Tcl and hence users have to be acquainted with the scripting and modeling methods used. Tcl is a little difficult to write and understand.
b) NS2 has a poor GUI (Graphical User Interface) unlike the modern network simulators, and hence developing model is time consuming.
c) Due to the open source code base, the model may contain bugs [6].

B. OMNet++:

a. Overview:

OMNeT++ (Objective Modular Network Testbed in C++) is a discrete event simulation environment, which supports component-based, modular programming. It has an Eclipse-based IDE with a graphical run-time environment. What makes OMNet++ popular is its excellent GUI (Graphical User Interface) which provides robust debugging and integration capabilities. OMNeT++ is licensed under the Academic Public License. This allows GNU Public License-like freedom but only in noncommercial settings. It was developed mainly by Andras Varga from Technical University of Budapest. Its main application is the simulation of communication networks; but as it has a very flexible architecture, it can also be used in other areas like the simulation of queuing networks, hardware architectures or complex IT systems.

b. Features:

OMNeT++ was developed in C++ and supports object-oriented programming. It provides a component-based modular programming architecture with a good graphical and visual aid. Thus a new project can be built one small module at a time, which later can be integrated to give the whole functionality. It also provides a Simulation kernel library which can be reused for different simulations. The GUI consists of Graphical network editor for NED files (GNED) and a NED compiler. There are tools for graphical analysis of the simulation result called Plove and Scalar, which are used for vectors and scalars respectively. For the execution of the simulation two interfaces are provided: Tkenv for graphical visualization and Cmdenv for command line execution. A tool for model documentation (opp_neddoc), utilities such as makefile creation tool, also exists along with sample simulations and proper documentation.

To support for simulation of wireless networks, OMNeT++ has been provided with external extensions. Some of the most popular extensions are INET Framework, Mobility Framework and INET MANET for mobile ad-hoc networks. An extension called Mixim, is a modeling framework created for mobile and fixed wireless networks, and now includes the Mobility Framework. Another feature is that the simulation executables are actually independent programs that can be run on other machines without the simulator. OMNeT++ works on Linux, Unix-like systems and Windows XP/2K [7].

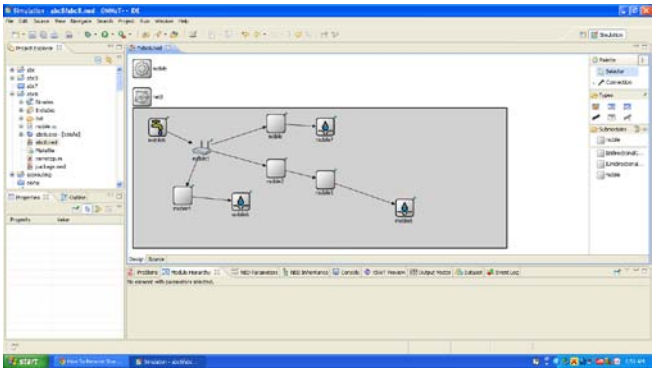

Figure 2: Workspace of Omnet++

c. Merits:

a. It offers a powerful GUI that helps in the easy and simplified tracing and debugging of the programs.
b. The GUI makes the designing of the network easier and quicker with its drag and drop and other visual aids.
c. The configuration file can be modified easily to simulate the network with different parameters.
d. With the contribution of the supporting team, OMNeT++ also has a mobility framework that helps in designing of wireless networks.

e. The execution of the simulation is simplified for the user. They can choose to run it graphically or through command line.
f. The result analysis of the simulated output can be visualized graphically and is separate for scalars and vectors [7].

d. Demerits:

a. The number of available protocols is not large.
b. Compatibility problems may arise in modules developed by different teams.
c. Some of the open source modules may contain bugs [8].

C. Avrora:

a. Overview:

Avrora, built in Java, is a simulator specifically designed for WSNs. It is a set of simulation and analysis tool written by the University of California, Los Angeles Compilers Group. It was developed mainly by Ben L. Titzer and his team. It is written to simulate AVR-based micro-controller MICA2 sensor nodes. This simulator has open sources and online documents which can be used as help. Avrora contains a flexible framework for simulating and analyzing assembly programs, providing a clean Java API and infrastructure for experimentation, profiling, and analysis.

b. Features:

The simulator allows one to specify a-monitors option, which allows monitors to collect information while it executes the program at the same time. The trace monitor will print out each instruction along the path of execution. It then generates a report once the execution is complete. The GDB debugger hooks provide source-level debugging and integrated development and testing. The profiling utilities are used to study the simulation's behavior during execution. The instrumentation capabilities allow the observation of program behavior without disturbing the simulation, and without modifying the simulator source code. The control flow graph tool can create a graphical representation of your program's instructions and the energy analysis tool is used to analyze energy consumption and to determine the battery life of your device. There is a stack checker tool that is used to bind the maximum stack size used by the program [9].

c. Merits:

a. Avrora is an instruction-level simulator where the code is run instruction by instruction, which provides faster speed and better scalability.
b. It is built in Java language hence imbibes the salient features of the language such as flexibility.
c. It can support thousands of nodes simulation, and can save much more execution time with similar accuracy.

d. Demerits:

a. It does not have a GUI, which may make designing the network more difficult.
b. It does not provide network communication tools, and thus can not simulate network management algorithms [9].

D. J-Sim:

a. Overview:

J-Sim is a simulation environment written in Java. It provides an open-source, component based network simulation environment. It gives a Simula-like simulation environment with a lot of additional functionalities. It was developed by a team at the Distributed Real Time Computing Laboratory of the Ohio State University and by Illinois University. It is being used in a rapidly emerging area of simulation research that is simulation and animation environment supporting Web-Based Simulation.

b. Features:

The simulation is executed in a step-by-step manner. During a step, just one process is given a chance to run. The basic building blocks of every J-Sim simulation are processes and queues. Processes are active, while queues and other elements of the simulation are passive. The simulation models may be built using either the event package (Event-Scheduling Paradigm) or the process package (Process-Interaction Paradigm). The scripting language is used to create simulation scenarios, which are used to configure and control the simulation at run-time and also to monitor and collect simulation data. A script interface allows integration with different script languages like Tcl, which is currently fully integrated into the environment. A graphical editor exists for Tcl configuration files called gEditor and a special plot component is provided to plot simulation statistics. The execution components are implemented by Java threads which uses the Java Virtual Machine (JVM) scheduling thread execution. Therefore, a simulation runs in the same manner a real system does. This framework is built upon the extensible inter-networking framework (INET) and the autonomous component architecture (ACA) of J-Sim. The ACA enables new components to be included into J-Sim in a plug-and-play fashion. Also available are sensor and sink nodes along with an object-oriented definition. Also it has wireless communication channels, with physical media. It also provides mobility model and power model (both energy-producing and energy-consuming components) [10].

c. Merits:

a. J-Sim provides a GUI library, which helps in debugging and tracing programs.
b. It contains large number of protocols.
c. It has facilities for easy simulation as it provides good re-usability and interchangeability of program modules.
d. It supports routing and localization simulations in WSNs data diffusion.

d. Demerits:

a. The execution of the simulation takes much more time than other simulators like NS2.
b. J-Sim has a lot new protocols or node components added to it as it was not originally designed for use with WSNs, and hence is a little difficult to use [10].

E. OPNET Modeler:

a. Overview:

Opnet, standing for Optimized Network Engineering Tools, is used to analyze computer networks and applications based on their performance and properties. It was developed by Alain Cohen in 1987 as his graduate project at MIT. He later co-founded the company Opnet Technologies Inc. Along with his brother Marco Cohen and classmate Steven Baraniuk. The company's first product was OPNET Modeler, a software tool for network modeling and simulation. It is a well-established commercial discrete-event simulator developed in C++. The simulator offers an environment for designing protocols and technologies and also for the testing and demonstrating designs in realistic scenarios.

b. Features:

OPNET Modeler uses hierarchical modeling. It defines a network as a collection of sub-models representing sub-networks or nodes. The topology that one needs can be manually created, imported or selected from the pool of predefined topologies.

The simulator has both built-in and external tools for statistics analysis. Some of the graphical tools of the simulator are described as follows. First is the Probe Editor with which one can find out the runtime statistics of any particular point in the network by simply placing a probe at it. Global statistics, node statistics, attribute statistics and animation statistics are some statistics types collected by different probes. Opnet provides Simulation Tool that one can use to give input/output optins, runtime options and other attributes. We can also define a sequence of simulation and set option for parallel simulations. Analysis Tool graphically displays simulation results, creates scalar graphs and can save analysis configurations for future use. It has an additional function called High-level Architecture that can support distributed simulation. OPNET Modeler runs on Windows XP/2K, Linux and Solaris platforms [11].

c. Merits:

a. OPNET Modeler provides a good manual and an introductory tutorial.
b. The Modeler has an advanced graphical interface which is used for model creation, simulation execution and data analysis.
c. It can concurrently execute several simulation scenarios.
d. It provides realistic mobility models.

d. Demerits:

a. It is not open source hence additional external tools are not supported.
b. If a specific component has to be developed, the simulator maybe quite complex [11].

F. Tossim:

a. Overview:

TOSSIM is a simulator specifically designed for WSN running on TinyOS, an open source operating system. It was developed by UC Berkeley's TinyOS project team in 2003 . It is a bit-level discrete event network simulator built in Python and C++. It works on Linux Operating Systems or on Cygwin on Windows. It was designed specifically for TinyOS applications to be run on MICA Motes and can simulate entire TinyOS applications. TOSSIM captures the behavior and interactions of networks not on the packet level but at network bit granularity [12].

b. Features:

TOSSIM can replace a packet-level communication component for packet-level simulation, or replace a low-level radio chip component for a more precise simulation of the code execution. It works by replacing components with simulation implementations. The accuracy and complexity of the model necessary for the simulations can be chosen by the developers. The simulation provides several mechanisms for interacting with the network. The packets can be statically or dynamically injected into the network and the packet traffic can be monitored. It simulates the TinyOS network stack at the bit level, allowing experimentation with low-level protocols in addition to top-level application systems. The transmission is simulated at the bit level [13].

c. Merits:

a) It provides open sources and online documents.
b) Each node can be evaluated under perfect transmission conditions.
c) Besides network, TOSSIM can simulate radio models and code executions
d) One can capture the hidden terminal problems.

d. Demerits:

a) It was designed to simulate behaviors and applications of TinyOS, and not to simulate other new protocols.
b) It can not correctly simulate issues of the energy consumption in WSN; one needs to use another version PowerTOSSIM.
c) Motes-like nodes are the only thing that it can simulate [13].

III. CONCLUSION

Simulation is an essential tool to study Wireless Sensor Networks due to the difficulties of setting up real experiments. This survey provides guidelines to help selecting a suitable simulation model for a WSN based on various aspects. Each tool has both advantages and disadvantages. One can decide on a particular tool based on his requirements.

NS2 is the most popular and widely used simulator amongst all. One can find numerous tutorials and examples online. It is powerful and efficient but doesn't have a good GUI. Omnet++, on the other hand provides an excellent GUI. Hence designing a network is easy and fast. Also it is open source and additional frameworks like Mobility Framework. But it fails to provide enough number of protocols in its library. Avrora, built in Java, includes all the salient features of the language. It is also is very flexible and scalable. But it doesn't have a good GUI. J-Sim provides a Simula-like environment along with Java as the development language. It provides an excellent GUI, but the execution of the simulation takes more time compared to the other simulators. OPNET Moduler has a good graphical interface and can execute several simulations in a concurrent manner. It is not open sourced. TOSSIM, built specifically

for WSN, runs on TinyOS. simulate radio models and code executions.

Thus one has quite a large number of simulators to choose from. A deep study of these is mandatory for a better understanding and characterization the simulators.

IV. ACKNOWLEDGMENT

We would like to thank our graduate project guide Prof. R.K. Nadesh, who has guided throughout the duration of our project. We were privileged to experience a sustained enthusiastic and involved interest from his side. This fueled our enthusiasm even further and encouraged us to boldly step into what was a totally dark and unexplored expanse before us.

V. REFERENCES

[1]. Muhammad Saleem, Gianni A. Di Caro, Muddassar Farooq. Swarm Intelligence Based Routing Protocol for Wireless Sensor Networks: Survey and Future Directions, Information Sciences, 2011, pp 4597–4624.

[2]. E. Egea-López, J. Vales-Alonso, A. S. Martínez-Sala, P. Pavón-Mariño, J. García-Haro. Simulation Tools for Wireless Sensor Networks, Summer Simulation Multiconference– SPECTS, 2005, pp 2-9.

[3]. Guodong Teng Kougen Zheng, Wei Dong. A Survey of Available Tools for Developing Wireless Sensor Networks. Systems and Networks Communications, 2008. ICSNC '08. 3rd International Conference, Oct. 2008.

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

[5]. Harsh Sundani, Haoyue Li, Vijay K. Devabhaktuni, Mansoor Alam, & Prabir Bhattacharya. Wireless Sensor Network Simulators A Survey and Comparisons. International Journal of Computer Networks (IJCN), Volume (2).

[6]. The Network Simulator, NS–2 [Online]. Available:http://www.isi.edu/nsnam/ns/

[7]. OMNeT++ Community Site [Online]. Available: http://www.omnetpp.org/

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

[9]. Avrora Community Site [Online]. Available: http:// compilers.cs.ucla.edu/avrora/index.html

[10]. J-Sim Official website [Online]. Available: https://sites.google.com/site/jsimofficial/downloads

[11]. OPNET Technologies, Inc [Online]. Available: http://www.opnet.com/

[12]. TOSSIM tutorial and information [Online]. Available: http://www.tinyos.net/nest/doc/tutorial/tossim-lesson.html

[13]. TOSSIM [Online]. Available: http://www.cs.berkeley.edu/~pal/research/tossim.html