International Journal of Computer Science and Network (IJCSN)
Volume 1, Issue 5, October 2012 www.ijcsn.org ISSN 2277-5420

Evaluation of Discrete Event Wireless Sensor Network Simulators

[1]AnilKumar Patil , [2]Dr P. M .Hadalgi

[1]Research scholar, Dept of Applied Electronics, Gulbarga University Karnataka,India

[2]Dept of Applied Electronics Gulbarga University, Karnataka,India

Abstract

Simulation tools for wireless sensor networks are increasingly being used to study sensor webs and to test new applications and protocols in this evolving research field.There is always an overriding concern when using simulation that the results may not reflect accurate behavior. It is therefore essential to know the strengths and weaknesses of these simulators. This paper provides a comprehensive survey and comparisons of various popular sensor network simulators with a view to help researchers choose the best simulator available for a particular application environment. It also provides a detailed comparison describing the pros and cons of each simulator.

Keywords: Wireless Sensor Network, Simulator, NS-2, TOSSIM, OMNeT++, J-Sim, ATEMU, Avrora,OPNET, CASTALIA

1. Introduction

1.1 What is WSN

Sensor networks are composed of large numbers of tiny sensing and computing devices. Each of these devices, called motes, has very limited communication, computational and energy resources. Often embedded in uncontrolled physical environments, these networks require distributed algorithms for efficient data processing, while individual motes require highly concurrent and reactive behavior for efficient operation. Sensor networks face many problems that do not arise in other types of networks[1].Power constraints, limited hardware, decreased reliability, and a typically higher density and number of nodes than those found in conventional networks are few of the problems that have to be considered when developing protocols for use in sensor networks. Fig.1 shows a typical simple wireless sensor network. As can be seen, a complete wireless sensor network usually consists of one or more base stations (or gateway), a number of sensor nodes, and the end user. The topology of WSNs can vary among star network, tree network, and mesh network. Each node has the ability to communicate with every other node wirelessly, thus a typical sensor node has several components: a radio transceiver with an antenna which has the ability to send or receive packets, a microcontroller which could process the data and schedule relative tasks, sensors sensing the environment data, and batteries providing energy supply.
Sensor nodes measure physical quantities such as temperature, position, humidity, pressure etc. The output of those sensor nodes are wirelessly transmitted to the base station (or gateway) for data collection, analysis, and logging. End users may also be able to receive and manage the data from the sensor via a website from long-distance or applications in console terminal[2] . However due to the associated cost, time and complexity involved in implementation of such networks, developers prefer to have first-hand information on feasibility and reflectivity crucial to the implementation of the system prior to the hardware implementation.

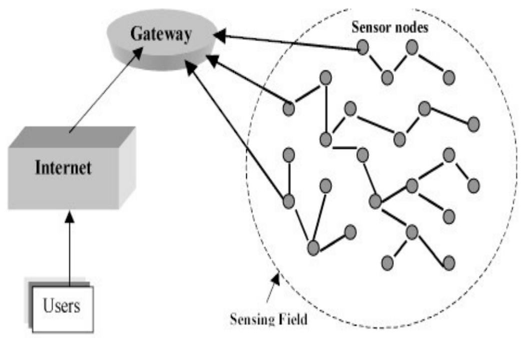

Fig. 1 A simple wireless sensor network

This is especially true in sensor networks, where hardware may have to be purchased in large quantities and at high cost. Even with readily available sensor nodes, testing the network in the desired environment can be a time consuming and difficult task. Simulation-based testing can help to indicate whether or not the time and monetary investments are worthwhile. Simulation is, therefore, the most common approach to developing and testing new protocol for sensor networks. There are a number of advantages to this approach including lower cost, ease of implementation, and practicality of testing large-scale networks.

In order to effectively develop any protocol based on simulations, it is important to know the different tools available and their benefits and drawbacks. Given the facts that simulation is not perfect and that there are a number of popular sensor network simulators available, thus making different simulators accurate and most effective for different situations/applications. It is crucial for a developer to choose a simulator that best fits the application[3] . However, without a working knowledge of the available simulators, this is can be a challenging task. Additionally, knowing the weaknesses of available simulators could help developers to identify drawbacks of their own models, when compared with these simulators, thus providing an opportunity for improvement. It is thus imperative to have a detailed description of a number of the more prominent simulators available. In this paper, we have compared various sensor network simulators with emphasis on their ease of use, key features, limitations, availability, and environments best supported[4].

1.2 Comparison of wired and wireless network

The wired network has been around for decades, as long as the internet itself. Compared with wireless networks, wired networks are more secure and faster in transfer speeds. However, wired networks contain one of the biggest growing problems, wires. Complicated wires and power cords are difficult to manage and hugely degrade the flexibility. Wiring and rewiring are the bottleneck of development of wired network. With the rapid development of wireless technology, more and more people prefer to use wireless network as their end-user network.

Compared with the traditional wireless network, WSN has its own features, such as low cost and low energy consumption. To reduce cost, each sensor board has very limited onboard resource, such as computing speed, storage and energy source. To achieve long lifetime with limited power supply usually batteries, onboard components are designed to consume energy as little as possible. For instance, the transmit power of radio is 1000 times smaller than the one in Wi-Fi routers. WSN is always deployed in difficult-access areas; the ability of self-configuration is another design goal.

1.3. Design of Sensor Network Simulator

The design of a Wireless Sensor Network (WSN) is a very application-specific task, especially because of the peculiarity of the considered deployment environment. Generic reliable predictive models for data correlation or radio propagation are seldom available. A thorough preliminary test phase is thus necessary, either by means of specifically crafted test beds, or via reliable simulations. WSN applications must be tested on a large scale, and under complex and varying conditions in order to capture a sufficiently wide range of interactions, both among nodes, and with the environment. A WSN simulator consists of various modules namely events, medium, environment, node, transceiver, protocols, and applications. Each category is represented by an interface that defines its methods and events generated and consumed.

1. Event
Event is an abstract base class that provides basic functionality for all events. It contains the time at which an event should work, and provides methods to: compare events based on their fire times, determine whether events are equal, print themselves to a string, and an abstract method to fire the event.

2. Medium
Medium models the wireless medium. It allows nodes to broadcast signals, and is responsible for
informing nodes of signals that affect it. In order to do this, Medium must be informed of the presence of every node, and any changes in position or radio properties such as transmitter power or receiver sensitivity. Medium has the properties of bandwidth and wavelength of the medium modeled and a reference to a propagation model that is given to it at the time of construction. The propagation model provides the strength at a particular receiver from a signal transmitted by a given transmitter.

3. Environment
The Environment module is similar to Medium module. The difference is that the implementation of Environment has properties that relate to the physical phenomenon modeled. Environment also has a propagation model that models the propagation of the physical phenomena modeled. Physical phenomena of interest in sensor networks include: temperature, light, humidity, magnetic field, sound, optical, chemical presence.

4. Node
It represents a single node in a wireless sensor network. As such, it serves as a container for all of the components, both hardware and software, in a node. These components should be included: processor, transceiver, sensors, actuators, energy source (such as a battery), network protocols, and applications. In addition each node has the properties of location and identification.

5. Transceiver
Transceiver models the hardware transceiver on each sensor node. It models the transceiver states (i.e.sleep,

standby, receive, and transmit), and their associated behavior and power consumption.Transceiver consumes events informing it of the beginning and ending of every signal it receives. It sums active signals to maintain the interference. Transceiver generates events for the beginning and ending of every signal it transmits. These events are all exchanged with an instance of the Medium module.

6. Physical Protocol
The Physical protocol is the lowest layer in a network stack. It is often implemented in the transceiver hardware. The Physical layer provides services for: changing the state of the transceiver, carrier sensing, sending and receiving packets, received energy detection on received packets, changing channels on physical layers that support multiple channels.

7. MAC Protocol
The MAC protocol is the next layer in a network stack. It is usually implemented in software running on the node's processor. The MAC layer provides services for: changing the state of the MAC layer (i.e. low power mode), setting and getting protocol parameters, sending and receiving packets, etc. A WSN simulator usually offers implementations for several sensor network MAC protocols.

8. Routing Protocol
The Routing protocol resides above the MAC protocol and provides services for routing messages over multiple hops between nodes that cannot communicate directly.

9. Application Layer
The Application layer resides at the top of the network stack. It interfaces with the lower layers in the network stack as well as the sensors and actuators to implement a wireless sensor network application.
Most of the WSN simulators are based on the design described above. In addition to including the different modules, a WSN simulator should also have the following capabilities:
i. Reusability and availability
Simulation is used to test novel techniques in realistic and controlled scenarios. Researchers are usually interested in comparing the performance of a new technique against existing proposals[5] .
ii. Performance and scalability
Performance and scalability is a major concern when facing WSN simulation. The former is usually bounded to the programming language effectiveness. The latter is constrained to the memory, processor and logs storage size requirements[6] .
iii. Support for rich-semantics scripting languages to define experiments and process results
The vast amount of variables involved in the definition of a WSN experiment requires the use of specific input scripting languages, with high-level semantics. Additionally, it is likely that large quantities of output data will also be generated through many replicas of the experiments[7] . Therefore, a suitable output scripting language, which helps to obtain the results from the experiments quickly and precisely is desirable.
iv. Graphical, debug and trace support.
Graphical support for simulations is interesting in three aspects:
(a) As a debugging aid. The primary and more practical way to quickly detect a bad behavior is to "watch" and follow the execution of a simulation. The key features that a graphical interface should support are: Capability of inspection of modules, variables and event queues at real time, together with "step-by-step" and "run-until" execution possibilities. These features make graphical interfaces a very powerful debugging tool. Note that the key is the ability to interact with the simulation.
(b) As a visual modeling and composition tool. This feature usually facilitates and speeds the design of small experiments or the composition of basic modules. However, for large scale simulations, it is not very practical.
(c) Finally, it allows quick visualization of results without a post-processing application[8] .
However, there are various challenges associated with the available WSN simulators. For instance, some simulator lack of available protocol models, which causes the increase of developing time, some simulators limit the scalability, etc. Additionally, modeling problems arise when considering the new environment and the energy components. They also compromise scalability and accuracy. A deep study of these issues is mandatory for a better understanding and characterization of sensor networks and their corresponding simulators

2. Basic Concepts

There are three types of simulation: Monte Carlo Simulation, Trace-Driven Simulation and Discrete-Event Simulations[9] . The last two simulations are used commonly in WSN.
In this paper discrete event simulators are compared.

2.1 Discrete-Event Simulations

Discrete-event simulation is widely used in WSNs, because it can easily simulate lots of jobs running on different sensor nodes. Discrete-event simulation includes some of components. This simulation can list pending events, which can be simulated by routines. The global variables, which describe the system state, can represent the simulation time, which allow the scheduler to predict this time in advance. This simulation includes input routines, output routines, initial routines, and trace routines. In addition, this simulation provides dynamic memory management, which can add new entities and drop old entities in the model. Debugger breakpoints are provided in discrete-event simulation, thus users can check the code step by step without disrupting the program operation.

2.2 Simulator and Emulator

Simulator[10] is universally used to develop and test protocols of WSNs, especially in the beginning stage of these designs. The cost of simulating thousands of nodes networks is very low, and the simulation can be finished within very short execution time. Both general and specialized simulators are available for uses to simulate WSNs. The tool, which is using firmware as well as hardware to perform the simulation, is called emulator[10] . Emulation can combine both software and hardware implementation. Emulator implements in real nodes, thus it may provide more precision performance. Usually emulator has highly scalability, which can emulate numerous sensor nodes at the same time. In this survey, seven simulation tools are also categorize into this two types, and their advantage and disadvantage will be discussed in section 3.

3. Simulation Tools

This section illustrates simulation tools used in WSNs: NS-2, TOSSIM, OMNeT++, J-Sim, ATEMU, Avrora,OPNET and Castalia and analyzes the advantage and disadvantage of each simulation tool.

3.1 NS-2

The introduction of NS-2 [11-17]and the comparison with other simulation tools will be discussed in this subsection.

3.1.1 Overview

NS-2 is the abbreviation of Network simulator version two, which first been developed by 1989 using as the REAL network simulator. Now, NS-2 is supported by Defense Advanced Research Projects Agency and National Science Foundation. NS-2 is a discrete event network simulator built in Object-Oriented extension of Tool Command Language and C++[18] People can run NS-2 simulator on Linux Operating Systems or on Cygwin, which is a Unix-like environment and command-line interface running on Windows. NS-2 is a popular non-specific network simulator can be used in both wire and wireless area. This simulator is open source and provides online document.

3.1.2 Merits and Limitations

NS-2 contains both merits and limitations when people use it to simulate WSNs. To the merits, firstly as a non-specific network simulator, NS-2 can support a considerable range of protocols in all layers. For example, the ad-hoc and WSN specific protocols are provided by NS-2. Secondly, the open source model saves the cost of simulation, and online documents allow the users easily to modify and improve the codes.
However, this simulator has some limitations. Firstly, people who want to use this simulator need to familiar with writing scripting language and modeling technique; the Tool Command Language is somewhat difficult to understand and write. Secondly, sometimes using NS-2 is more complex and time-consuming than other simulators to model a desired job. Thirdly, NS-2 provides a poor graphical support, no Graphical User Interface (GUI) [19]; the users have to directly face to text commands of the electronic devices. Fourthly, due to the continuing changing the code base, the result may not be consistent, or contains bugs.
In addition, since NS-2 is originally targeted to IP networks , there are some limitations when apply it to simulate WSNs. Firstly, NS-2 can simulate the layered protocols not application behaviors. However, the layered protocols and applications interact and can not be strictly separated in WSNs. So, in this situation, using NS-2 is inappropriate, and it can hardly to acquire correct results. Secondly, because NS-2 is designed as a general network simulator, it does not consider some unique characteristics of WSN. For example, NS-2 can not simulate problems of the bandwidth, power consumption or energy saving in WSN. Thirdly, NS-2 has a scalability problem in WSN, it has trouble to simulate more than 100 nodes. As the increasing of the number of nodes, the tracing files will be too large to management. Finally, it is difficult to add new protocols or node components due to the inherently design of NS-2. In sum, NS-2 as a simulator of WSN contains both advantages and disadvantages.

3.2 TOSSIM

The introduction of TOSSIM[10,12,13,20-24] and the comparison with other simulation tools will be discussed in this subsection.

3.2.1 Overview

TOSSIM is an emulator specifically designed for WSN running on TinyOS, which is an open source operating system targeting embedded operating system. In 2003, TOSSIM was first developed by UC Berkeley's TinyOS project team. TOSSIM is a bit-level discrete event network emulator built in Python[25], a high-level programming language emphasizing code readability, and C++. People can run TOSSIM on Linux Operating Systems or on Cygwin on Windows. TOSSIM also provides open sources and online documents.

3.2.2 Merits and Limitations

TOSSIM contains both merits and limitations when people use it to emulate WSNs. To the merits, the open source model free online document save the emulation cost. Also, TOSSIM has a GUI, TinyViz, which is very convenience for the user to interact with electronic devices because it provides images instead of text commands.
In addition, TOSSIM is a very simple but powerful emulator for WSN. Each node can be evaluated under perfect transmission conditions, and using this emulator can capture the hidden terminal problems. As a specific network emulator, TOSSIM can support thousands of nodes simulation. This is a very good feature, because it can more accurately simulate the real world situation. Besides network, TOSSIM can emulate radio models and code executions. This emulator may be provided more precise simulation result at component levels because of compiling directly to native codes.
However, this emulator still has some limitations. Firstly, TOSSIM is designed to simulate behaviors and applications of TinyOS, and it is not designed to simulate the performance metrics of other new protocols. Therefore, TOSSIM can not correctly simulate issues of the energy consumption in WSN; people can use PowerTOSSIM[26] , another TinyOS simulator extending the power model to TOSSIM, to estimate the power consumption of each node. Secondly, every node has to run on NesC code, a programming language that is event-driven, component-based and implemented on TinyOS, thus TOSSIM can only emulate the type of homogeneous applications. Thirdly, because TOSSIM is specifically designed for WSN simulation, motes-like nodes are the only thing that TOSSIM can simulate. In sum, TOSSIM as an emulator of WSN contains both advantages and disadvantages.

3.3 OMNeT++

The introduction of OMNeT++[12,27,28] and the comparison with other simulation tools will be discussed in this subsection.

3.3.1 Overview

OMNeT++ is a discrete event network simulator built in C++. OMNeT++ provides both a noncommercial license, used at academic institutions or non-profit research organizations, and a commercial license, used at "for-profit" environments. This simulator supports module programming model. Users can run OMNeT++ simulator on Linux Operating Systems, Unix-like system and Windows. OMNeT++ is a popular non-specific network simulator, which can be used in both wire and wireless area. Most of frameworks and simulation models in OMNeT++ are open sources.

3.3.2 Merits and Limitations

OMNeT++ contains both merits and limitations when people use it to simulate WSNs. To the merits, firstly, OMNeT++ provides a powerful GUI. This strong GUI makes the tracing and debugging much easier than using other simulators. Although initial OMNeT++ do not support the module library which is specifically used for WSNs simulation, with the consciously contribution of the supporting team, now OMNeT++ has a mobility framework. This simulator can support MAC protocols as well as some localized protocols in WSN. People can use OMNeT++ to simulate channel controls in WSNs. In addition, OMNeT++ can simulate power consumption problems in WSNs. However, there are still some limitations on OMNeT++ simulator. For example, the number of available protocols is not larger enough. In addition, the compatible problem will rise since individual researching groups developed the models separately, this makes the combination of models difficult and programs may have high probability report bugs. In sum, both advantages and disadvantages are included in the OMNeT++ design.

3.4 J-Sim

The introduction of J-Sim[11,12,24,29] and the comparison with other simulation tools will be discussed in this subsection.

3.4.1 Overview

J-Sim is a discrete event network simulator built in Java. This simulator provides GUI library, which facilities users to model or compile the Mathematical Modeling Language, a “text-based language” written to J-Sim models. J-Sim provides open source models and online documents. This simulator is commonly used in physiology and biomedicine areas, but it also can be used in WSN simulation. In addition, J-Sim can simulate real-time processes.

3.4.2 Merits and Limitations

J-Sim contains both merits and limitations when people use it to simulate WSNs. To the merits, firstly, models in J-Sim have good reusability and interchangeability, which facilities easily simulation. Secondly, J-Sim contains large number of protocols; this simulator can also support data diffusions, routings and localization simulations in WSNs by detail models in the protocols of J-Sim. J-Sim can simulate radio channels and power consumptions in WSNs. Thirdly, J-Sim provides a GUI library, which can help users to trace and debug programs. The independent platform is easy for users to choose specific components to solve the individual problem. Fourth, comparing with NS-2, J-Sim can simulate larger number of sensor nodes, around 500, and J-Sim can save lots of memory sizes. However, this simulator has some limitations. The execution time is much longer than that of NS-2. Because J-Sim was not originally designed to simulate WSNs, the inherently design of J-Sim makes users hardly add new protocols or node components.

3.5 ATEMU

The introduction of ATEMU[12,13,21,24,30] and the comparison with other simulation tools will be discussed in this subsection.

3.5.1 Overview

ATEMU is an emulator of an AVR processor for WSN built in C; AVR is a single chip microcontroller commonly used in the MICA platform. ATEMU provides GUI, Xatdb; people can use this GUI to run codes on sensor nodes, debug codes and monitor program executions. People can run ATEMU on Solaris and Linux operating system. ATEMU is a specific emulator for WSNs; it can support users to run TinyOS on MICA2 hardware. ATEMU can emulate not only the communication among the sensors, but also every instruction implemented in each sensor. This emulator provides open sources and online documents.

3.5.2 Merits and Limitations

ATEMU contains both merits and limitations when people use it to simulate wireless sensor network. To the merits, firstly, ATEMU can simulate multiple sensor nodes at the same time, and each sensor node can run different programs. Secondly, ATEMU has a large library of a wide rage of hard devices. Thirdly, ATEMU can provide a very high level of detail emulation in WSNs. For example, it can emulate different sensor nodes in homogeneous networks or heterogeneous networks. ATEMU can emulate different application run on MICA. Also users can emulate power consumptions or radio channels by ATEMU. Fourthly, the GUI can help users debug programs, and monitor program executions. The open source saves the cost of simulation. ATEMU can provide an accurate model, which helps users to give unbiased comparisons and get more realistic results. The ATEMU components architecture is shown in Figure 6. However, this emulator also has some limitations. For instance, although ATEMU can give a highly accuracy results, the simulation time is much longer than other simulation tools. In addition, ATEMU has fewer functions to simulate routing and clustering problems. Therefore, both merits and limitation contains in ATEMU.

3.6 Avrora

The introduction of Avrora[13,24,31] and the comparison with other simulation tools will be discussed in this subsection.

3.6.1 Overview

Avrora is a simulator specifically designed for WSNs built in Java. Similar to ATEMU, Avrora can also simulate AVR-based microcontroller MICA2 sensor nodes. This simulator was developed by University of California, Los Angeles Compilers Group. Avrora provides a wide range of tools that can be used in simulating WSNs. This simulator combines the merits of TOSSIM and ATEMU, and limits their drawbacks. Avrora does not provide GUI. Avrora also supports energy consumption simulation. This simulator provides open sources and online documents. However, this simulator has some drawbacks. It does not have GUI. In addition, Avrora can not simulate network management algorithms because it does not provide network communication tools.

3.6.2 Merits and Limitations

Avrora contains both merits and limitations when people use it to simulate WSNs. To the merits, firstly, Avrora is an instruction-level simulator, which removes the gap between TOSSIM and ATEMU. The codes in Avrora run instruction by instruction, which provides faster speed and better scalability. Avrora can support thousands of nodes simulation, and can save much more execution time with similar accuracy. Avrora provides larger scalability than ATEMU does with equivalent accuracy; Avrora provides more accuracy than TOSSIM does with equivalent scales of sensor nodes. Unlike TOSSIM and ATEMU, Avrora is built in Java language, which provides much flexibility. Avrora can simulate different programming code projects, but TOSSIM can only support TinyOS simulation.

3.7 OPNET

The introduction of OPNET and the comparison with other simulation tools will be discussed in this subsection.

3.7.1 Overview

OPNET Modeler is a discrete event, object oriented, general purpose network simulator. Modeler was introduced in 1987 as the first commercial network simulator [34]. Originally, the software was developed for military purposes, but it has grown to be the world's leading commercial network simulation and modeling tool. OPNET is a large and powerful software with a wide variety of possibilities. OPNET can be used as a research tool and also as a network design/analysis tool. OPNET was originally built for the simulation of fixed networks, and therefore, it contains extensive libraries of accurate models from commercially available fixed network hardware and protocols.Recent versions also include wide possibilities for wireless network simulations including support for Zigbee compatible 802.15.4 MAC.

3.7.2 Merits and Limitations

Strength of OPNET in wireless network simulations is the accurate modeling of the radio transmission. Different characteristics of physical-link transceivers, antennas and antenna patterns are modeled in detail. With Wireless suite for Defence extension OPNET can model 3D outdoor scenarios and take into account different kinds of obstacles like terrain shape and buildings[35] . OPNET can also be used to define custom packet formats.
A weak point is that there exists only a few ready models for recent wireless systems.OPNET uses a hierarchical model to define each aspect of the system. Hierarchical structure is divided into three levels. The top level consists of the project editor, where network topology is designed.
The next level is the node level, where individual network nodes and data flow models are defined. A third level is the process editor, which uses a finite state machine approach to support specification of protocols, resources, applications and queuing policies. Finally, a simulation tool is included to support the three higher levels. OPNET also has so-called ESD (External System Domain) for communicating with external software and systems. Via ESD external software can exchange data and influence running simulation in OPNET. [34,36]

3.8 Castalia

The introduction of Castalia and the comparison with other simulation tools will be discussed in this subsection

3.8.1 Overview

Castalia is an application-level simulator for Wireless Sensor Network based on
OMNeT++. It can be used to evaluate different platform characteristics for specific applications, since it is highly parametric, and can simulate a wide range of platforms. In Castalia, sensor nodes are implemented as compound modules, consisting of sub-modules that represent, for instance, network stack layers, application, and sensor. Node modules are connected to wireless channel and physical process modules[32] . It is a generic simulator with realistic wireless channel and radio model based on measured data. Since it is based on the OMNeT++ platform, it can be used by researchers and developers who want to test their distributed algorithms and/or protocols in realistic wireless channel and radio models, with a realistic node behavior especially relating to access of the radio. It is developed in C++ at the National ICT Australia.

3.8.2 Merits and Limitations

Castalia merits are physical process modeling, sensing device bias and noise, node clock drift, and several MAC and routing protocols implemented. Castalia has a highly tunable Medium access Control(MAC) protocol and a flexible parametric physical process model. Distinct physical process modules in Castalia represent different sensing devices(e.g. temperature, pressure, light, and acceleration).Castalia can consider sensing device noise, bias and node clock drift[33]. It should be noted that Castalia is not sensor-platform specific. Castalia is meant to provide a generic reliable and realistic framework for the first order validation of an algorithm before moving to implementation on a specific sensor platform. It is not useful if one would like to test code compiled for a specific sensor node platform.

4. Summary

The purpose of this survey is to give a general picture of discrete event simulation tools using in WSNs, and help people to choose different simulation tools according to different needs. In the beginning part, this survey illustrates what is WSNs, why they need simulation, and what specific features should be considered when simulating WSNs. Then, this survey analyzes the simulators: NS-2, TOSSIM, OMNeT++, J-Sim, ATEMU, Avrora,OPNET,and Castalia and compares their merits and limitations, shown in Table 1. Both general simulators and specific simulators are evaluated in this survey. The general simulators usually lack some functions to provide specific simulations in WSNs, however specific simulators with more comprehensive functions may perform better. According to different targets to choose different simulation tools in WSNs will be more efficient and effective.

Table 1: Comparison of Simulation Tools

	Simulator or Emulator	Discrete-Event Simulations	GUI	Open sources and Online documents	General simulator or Specific simulator	Detail
NS-2	Simulator	Discrete-Event Simulation	No	Yes	general simulator	1.can not simulate more than 100 nodes, 2 can not simulate problems of the bandwidth or the power consumption in WSNs
TOSSIM	Emulator	Discrete-Event Simulation	Yes	Yes	specifically designed for WSNs	1.can support thousands of nodes simulation 2.can emulate radio models and code executions 3.only emulate homogeneous applications 4.have to use PowerTOSSIM to simulate power consumption
OPNET	Simulator	Discrete-Event Simulation	Yes	Yes	General simulator	1.can not support large number of sensors simulation 2.can support Zigbee compatible 802.15.4 MAC protocols 3. 3D radio modelling
OMNeT++	Simulator	Discrete-Event Simulation	Yes	noncommercial license,commercial license	general simulator	1.can support MAC protocols and some localized protocols in WSN 2.simulate power consumptions and channel controls 3. limited available protocols
J-Sim	Simulator	Discrete-Event Simulation	Yes	Yes	general simulator	1. can simulate large number of sensor nodes, around 500 2. can simulate radio channels and power consumptions 3. its execution time is much longer
ATEMU	Emulator	Discrete-Event Simulation	Yes	Yes	specifically designed for WSNs	1.can emulate different sensor nodes in homogeneous networks or heterogeneous networks 2.can emulate power consumptions or radio channels 3. the simulation time is much longer
Avrora	Simulator	Discrete-Event Simulation	No	Yes	specifically designed for WSNs	1. can support thousands of nodes simulation 2.can save much more execution time
Castalia	Simulator	Discrete-Event Simulation	yes	noncommercial license,commercial license	General simulator	1.can support Physical process modeling, sensing device bias and noise, node clock drift 2.several MAC and routing protocols supported.

5.References

[1]. M. Ilyas and I. Mahgoub, Handbook of sensor networks: compact wireless and wired sensing systems, BocaRaton, FL., CRC Press, 2004.

[2]. J. Liu, et. al., “Simulation modeling of large-scale ad-hoc sensor networks,” European Simulation Interoperability Workshop 2001, London, Eng land, June 2001.

[3]. I.F. Akyildiz and W. Su and Y. Sankarasubramaniam and E. Cayirci, “A Survey on Sensor Networks,” IEEE Communication Magazine, vol. 40, no. 8, pp. 102-116, Aug. 2002.

[4]. David Curren, “A survey of simulation in sensor networks,” University of Binghamton, NY, 2005.

[5]. John Heidemann, Kevin Mills, Sri Kumar, Expanding Confidence in Network Simulations.

[6]. E. Egea-López, J. Vales-Alonso, A. S. Martínez Sala,Pavón-Mariño, J. García-Haro,Simulation Tools for Wireless Sensor Networks.

[7]. Mekni, M. Moulin, A Survey on Sensor Webs Simulation Tools.

[8]. WeiChung,Hu MingLun, Lee TzungShian, Tsai Hewijin, Christine Jiau, A GUI Simulation Model in Supporting Embedded Software Design.

[9.] [Jain91]Raj Jain, "Art of Computer Systems Performance Analysis Techniques For Experimental Design Measurements Simulation And Modeling", Wiley Computer Publishing, John Wiley & Sons, Inc, 1991, ISBN: 0471503363.

[10]. [Imran10]Muhammad Imran, Abas Md Said,Halabi Hasbullah, "A Survey of Simulators, Emulators and Testbeds for Wireless Sensor Networks", Information Techonology(ITSim), 2010 International Symposium in, June 2010, pp. 897 – 902. ISBN: 978-1-4244-6715-0.

[11] .[Sinha09]Sourendra Sinha, Zenon Chaczko, Ryszard Klempous, "SNIPER: A WirelessSensor Network Simulator", Compute Aided Systems Theory- EUROCAST , 2009 Volume 5717/2009, pp. 913-920.

[12] .[Egea05]E. Egea-Lopez, J. Vales-Alonso, A. S. Martinez-Sala, P. Pavon-Marino, J. Garcia- Haro;Simulation Tools for Wireless Sensor Networks", Summer Simulation Multiconference, SPECTS, 2005, pp.2-9, URL: http://ait.upct.es/~eegea/pub/spects05.pdf

[13] .[Yi08]Sangho Yi, Hong Min, Yookun Cho, Jiman Hong,"SensorMaker: A Wireless Senso Network Simulator for Scalable and Fine-GrainedInstrumentation",computational science and its application-ICCSA, 2008, Volume5072/2008, pp. 800-810.

[14].[Xue07]Yunjiao Xue, Ho Sung Lee, Ming Yang, Kumarawadu, P., Ghenniwa, H.H., Weiming Shen, "Performance Evaluation of NS-2 Simulator for Wireless Sensor Networks", Electrical and Computer Engineering, CCECE Canadian Conference on, 22-26 April 2007, pp.1372 – 1375, ISBN: 1-4244-1020-7.

[15].[NS-2_wiki]"NS-2", URL: http://en.wikipedia.org/wiki/Ns-2, Description: an introduction of NS-2 in wiki webpage.

[16]. [NS-2_isi]"NS-2",URL: http://www.isi.edu/nsnam/ns/, Description: a webpage introduced NS-2.

[17]. [Stevens09]Clay Stevens, Colin Lyons, Ronny Hendrych, Ricardo Simon Carbajo, Meriel Huggard, Ciaran Mc Goldrick, "Simulating Mobility in WSNs: Bridging the gap between ns- 2 and TOSSIM 2.x", 13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications, 2009, ISBN: 978-0-7695-3868-6.

[18].[C++]"C++", URL: http://en.wikipedia.org/wiki/C++, Description:an introduction of C++ in wiki webpage.

[19]. Description: an introduction of GUI in wiki webpage.

[20]. [TOSSIM]"TOSSIM", URL: http://docs.tinyos.net/index.php/TOSSIM, Description: a webpage introduced TOSSIM.

[21]. [Polley04]J. Polley, D. Blazakis, J. McGee , D. Rusk, J.S. Baras, "ATEMU: A Fine-grained Sensor Network Simulator", First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, Santa Clara, CA, October 4-7, 2004.

[22]. [Stevens09]Clay Stevens, Colin Lyons, Ronny Hendrych, Ricardo Simon Carbajo, Meriel Huggard, Ciaran Mc Goldrick, "Simulating Mobility in WSNs: Bridging the gap between ns-2 and TOSSIM 2.x", 13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications, 2009, ISBN: 978-0-7695-3868-6

[23]. [Levis03]Philip Levis, Nelson Lee, Matt David Culler, "TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications", SenSys, 2003, ISBN:1-58113-707-9, URL: http://portal.acm.org/citation.cfm?id=958506

[24]. [Shu08]Lei Shu,Chun Wu,Yan Zhang,Jiming Chen,Lei Wang,Manfred Hauswirth, "NetTopo: Beyond Simulator and Visualizer for Wireless Sensor Networks", Future Generation Communication and Networking - FGCN , 2008, Volume1, pp. 17-20, ISBN: 978-0-7695-3431-2.

[25].[Python]"Python", URL: http://en.wikipedia.org/wiki/Python, Description: an introduction of Python in wiki webpage.

[26].[PowerTOSSIM]"PowerTOSSIM", URL: http://www.eecs.harvard.edu/~shnayder/ptossim/ Description: a webpage introduced PowerTOSSIM.

[27].[Omnet++_wiki]"Omnet++", URL: http://en.wikipedia.org/wiki/Omnet%2B%2B, Description: an introduction of Omnet++ in wiki webpage.

[28].[Omnet++]"Omnet++", URL: http://www.omnetpp.org/home/what-is-omnet, Description: a webpage introduced Omnet++.

[29].[J-sim]"J-sim" , URL http://sites.google.com/site/jsimofficial/, Description: a webpage introduced J-sim.

[30].[ATEMU]"ATEMU", URL: http://www.hynet.umd.edu/research/atemu/, Description: a webpage introduced ATEMU.

[31].[Avrora]"Avrora", URL: http://compilers.cs.ucla.edu/avrora/, Description: a webpage introduced Avrora.

[32].S. Park, A. Savvides, and M. B. Srivastava, "Simulating

networks of wireless sensors," Winter Simulation Conference, Arlington, Virginia, Dec. 2001.A. Boulis, "Castalia, a simulator for wireless sensor networks and body area networks," version 2.0,User's manual, May 2009 [Online]. Available: http://castalia.npc.nicta.com.au/. Retrieved:02/04/2010 B. Boulis, "Castalia: Revealing pitfalls in designing distributed algorithms in WSN," 5th Int. Conf. on Embedded Networked Sensor Systems, Sydney, Australia, Nov. 2007.

[33]. L. Girod, et al., "EmStar: An environment for developing wireless embedded system software," USENIX Technical Conference, Boston, MA, June 2004.

[34] www.opnet.com/solutions/brochures/Modeler.pdf (Read 5.2.2008)

[35] www.opnet.com/solutions/network_rd/mode ler_ wi reless_defense.html (Read 10.2.2008

[36].Prokkola, J.(2006), "OPNET – Network simulator",VTT Technical Research Center of Finland 106