A Survey of Wireless Sensor Network Simulation Tools

Fei Yu, feiyu8643@gmail.com (A paper written under the guidance of Prof. Raj Jain)

Download

Abstract

Recently there has been growing interest in providing fine-grained metering and control of living environments using low power devices. Wireless Sensor Networks (WSNs), which consist of spatially distributed self-configurable sensors, perfectly meet the requirement. Since running real experiments is costly and time consuming, simulation is essential to study WSNs, being the common way to test new applications and protocols in the field. This survey illustrates some main-stream WSNs simulators including: NS-2, TOSSIM, EmStar, OMNeT++, J-Sim, ATEMU, and Avrora.

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

Table of Contents

- 1. Introduction
  - 1.1 What is WSN
  - 1.2 Comparison of wired and wireless network
  - 1.3 Why use simulation in WSNs
- 2. Basic Concepts
  - 2.1 Discrete-Event Simulations and Trace-Driven Simulation
  - 2.2 Simulator and Emulator
- 3. Simulation Tools
  - 3.1 NS-2
    - 3.1.1 Overview
    - 3.1.2 Merits and Limitations
  - 3.2 TOSSIM
    - 3.2.1 Overview
    - 3.2.2 Merits and Limitations
  - 3.3 EmStar
    - 3.3.1 Overview
    - 3.3.2 Merits and Limitations
  - 3.4 OMNeT++
    - 3.4.1 Overview
    - 3.4.2 Merits and Limitations
  - 3.5 J-Sim
    - 3.5.1 Overview
    - 3.5.2 Merits and Limitations
  - 3.6 ATEMU
    - 3.6.1 Overview
    - 3.6.2 Merits and Limitations
  - 3.7 Avrora
    - 3.7.1 Overview
    - 3.7.2 Merits and Limitations
- 4. Summary
- 5. Reference
- 6. List of Acronyms

1. Introduction

1.1 What is WSN

With the development of embedded system and network technology, there has been growing interest in providing fine-grained metering and controlling of living environments using low power devices. Wireless Sensor Networks (WSNs), which consist of spatially distributed self-configurable sensors, perfectly meet the requirement. The sensors provide the ability to monitor physical or environmental conditions, such as temperature, humidity, vibration, pressure, sound, motion and etc, with very low energy consumption.

The sensors also have the ability to transmit and forward sensing data to the base station. Most modern WSNs are bi-directional, enabling two-way communication, which could collect sensing data from sensors to the base station as well as disseminate commands from base station to end sensors. The development of WSNs was motivated by military applications such as battlefield surveillance; WSNs are widely used in industrial environments, residential environments and wildlife environments. Structure health monitoring, healthcare applications, home automation, and animal tracking become representative WSNs applications.

A typical Wireless Sensor Network (WSN) is built of several hundreds or even thousands of "sensor nodes". The topology of WSNs can vary among star network, tree network, and mesh network. Each node has the ability to communication 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, several kinds of sensors sensing the environment data, and batteries providing energy supply.

A sensor node might vary in size. The "Smart dust" [SmartDust] sensor node, shown in Figure 1, from Electrical Engineering and Computer Science department at University of California Berkeley, is smaller than a coin. The cost of sensor nodes is similarly variable depending on the quality of onboard chips. One of the main challenges in WSNs is to decrease the cost and size. There are an increasing number of small companies producing WSN hardwires. The most popular two are TelosB sensor node, shown in Figure 2, from Crossbow Technology [TelosB] and Tmote Sky sensor node, shown in Figure 3, from Sentilla [TmoteSky].

Operating systems for WSN nodes are much less complex than general-purpose operating systems. This is because WSN usually deployed with a particular purpose and low power microcontrollers cannot afford complicated computing. TinyOS is the most popular operating system specifically designed for wireless sensor networks. TinyOS is based on an event-driven architecture using NesC programming language.

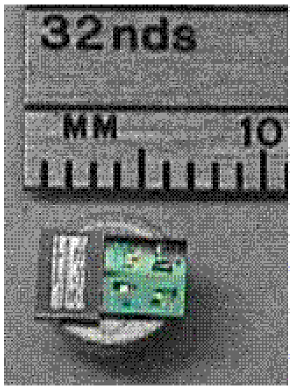


Figure 1: Smart dust [SmartDust]

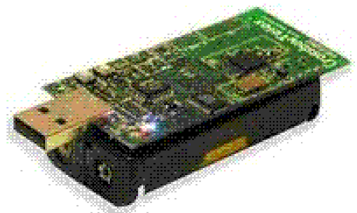


Figure 2: TelosB [TelosB]

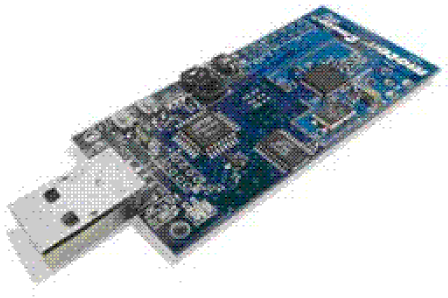


Figure 3: Tmote Sky [TmoteSky]

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 Why use simulation in WSNs

Nowadays, the WSN is a hot research topic. Many network details in WSNs are not finalized and standardized. Building a WSNs testbed is very costly. Running real experiments on a testbed is costly and difficulty. Besides, repeatability is largely compromised since many factors affect the experimental results at the same time. It is hard to isolate a single aspect. Moreover, running real experiments are always time consuming. Therefore, WSNs simulation is important for WSNs development. Protocols, schemes, even new ideas can be evaluated in a very large scale. WSNs simulators allow users to isolate different factors by tuning configurable parameters.

Consequently, simulation is essential to study WSNs, being the common way to test new applications and protocols in the field. This leads to the recent boom of simulator development.

However, obtaining solid conclusions from a simulation study is not a trivial task. There are two key aspects in WSNs simulators: (1) The correctness of the simulation models and (2) the suitability of a particular tool to implement the model. A "correct" model based on solid assumption is mandatory to derive trustful results. The fundamental tradeoff is: precision and necessity of details versus performance and scalability. In the rest of this survey, several main-stream WSNs simulators are described and compared in more detail.

2. Basic Concepts

There are three types of simulation: Monte Carlo Simulation, Trace-Driven Simulation and Discrete-Event Simulations [Jain91]. The last two simulations are used commonly in WSN. The first subsection will talk about the concepts of Trace-Driven Simulation and Discrete-Event Simulations. The second subsection will illustrate the concepts of simulator and emulator.

2.1 Discrete-Event Simulations and Trace-Driven Simulation

Discrete-event simulation [Jain91, DiscEvent_wiki] 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.

However, Trace-Driven Simulation [Jain91] provides different services. This kind of simulation is commonly used in real system. The simulation results have more credibility. It provides more accurate workload; these detail information allow users to deeply study the simulation model. Usually, input values in this simulation constant unchanged. However, this simulation also contains some drawbacks. For example, the high-level detail information increases the complexity of the simulation; workloads may change, and thus the representativeness of the simulation needs to be suspicious. In this survey, seven main-stream simulation tools are categorize into this two types, the detail information are described in section 3.

2.2 Simulator and Emulator

Simulator [Imran10] 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 [Imran10]. 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

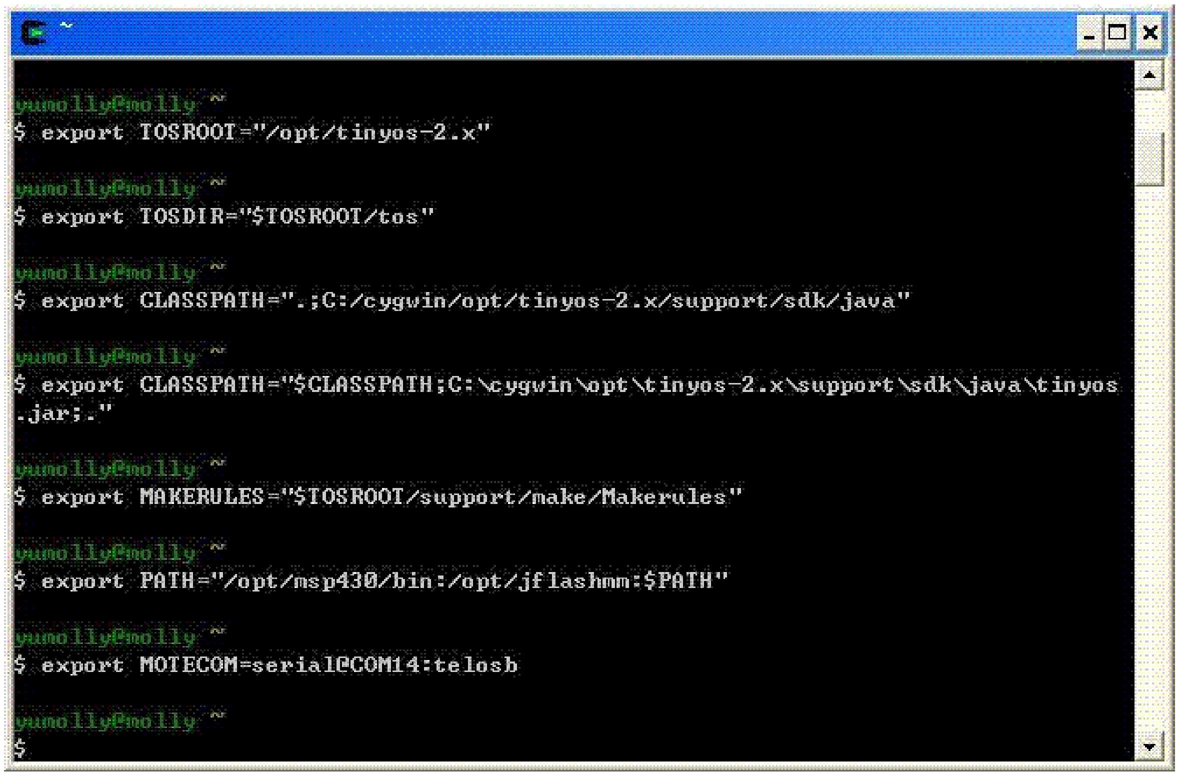
This section illustrates seven main-stream simulation tools used in WSNs: NS-2, TOSSIM, EmStar, OMNeT++, J-Sim, ATEMU, and Avrora, and analyzes the advantage and disadvantage of each simulation tool.

3.1 NS-2

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

3.1.1 Overview

NS-2 [NS-2_wiki,NS-2_isi,Egea05,Sinha09,Yi08,Stevens09,Xue07] 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++ [C++]. People can run NS-2 simulator on Linux Operating Systems or on Cygwin, shown in Figure 3, which is a Unix-like environment and command-line interface running on Windows. NS-2 is a popular non-specific network simulator can used in both wire and wireless area. This simulator is open source and provides online document.



```
yumolly@molly ~
$ export TOSROOT="/opt/tinyos-2.x"

yumolly@molly ~
$ export TOSDIR="$TOSROOT/tos"

yumolly@molly ~
$ export CLASSPATH=".;C:/cygwin/opt/tinyos-2.x/support/sdk/java"

yumolly@molly ~
$ export CLASSPATH="$CLASSPATH;C:\cygwin\opt\tinyos-2.x\support\sdk\java\tinyos
.jar;."

yumolly@molly ~
$ export MAKERULES="$TOSROOT/support/make/Makerules"

yumolly@molly ~
$ export PATH="/opt/msp430/bin:/opt/jflashmm:$PATH"

yumolly@molly ~
$ export MOTECOM=serial@COM14:telosb

yumolly@molly ~
$
```

Figure 4: Cygwin

3.1.2 Merits and Limitations

NS-2[NS-2_wiki,NS-2_isi,Egea05,Sinha09,Yi08,Stevens09,Xue07] 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 difficulty 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) [GUI]; 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 but WSNs, there are some limitations when apply it to simulate WSNs. Firstly, NS-2 can simulate the layered protocols but 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 and the comparison with other simulation tools will be discussed in this subsection.

3.2.1 Overview

TOSSIM [Imran10,TOSSIM,Polley04,Egea05,Shu08,Levis03,Yi08,Stevens09] 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[Python], 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 [Imran10,TOSSIM,Polley04,Egea05,Shu08,Levis03,Yi08,Stevens09] 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 [PowerTOSSIM], 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 EmStar

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

3.3.1 Overview

EmStar [Imran10,Elson03,Girod04,Polley04,Egea05,Yi08] is an emulator specifically designed for WSN built in C, and it was first developed by University of California, Los Angeles. EmStar is a trace-driven emulator [Girod04] running in real-time. People can run this emulator on Linux operating system. This emulator supports to develop WSN application on better hardware sensors. Besides libraries, tools and services, an extension of Linux microkernel is included in EmStar emulator.

3.3.2 Merits and Limitations

EmStar [Imran10,Elson03,Girod04,Polley04,Egea05,Yi08] contains both merits and limitations when people use it to simulate WSNs. To the merits, firstly, the modular programming model in EmStar allows the users to run each module separately without sacrificing the reusability of the software. EmStar has a robustness feature that it can mitigate faults among the sensors, and it provides many modes make debug and evaluate much easier. There is a flexible environment in EmStar that users can freely change between deployment and simulation among sensors. Also with a standard interfaces, each service can easily be interconnected. EmStar has a GUI, which is very helpful for users to control electronic devices. When using EmStar, every execution platform is written by the same codes, which will decrease bugs when iterate the separate modes. In addition, EmStar provides many online documents to facilities the widely use of this emulator. However, this emulator contains some drawbacks. For example, it can not support large number of sensors simulation, and the limited scalability will decrease the reality of simulation, shown in Figure 5. In addition, EmStar is can only run in real time simulation. Moreover, this emulator can only apply to iPAQ-class sensor nodes and MICA2 motes. All these drawbacks limit the use of this emulator. In sum, both advantages and disadvantages are included in the EmStar design.

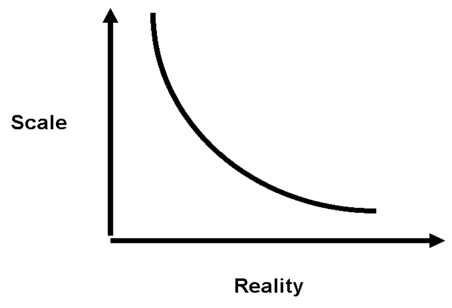


Figure 5: The relationship between Scale and Reality [Elson03]

3.4 OMNeT++

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

3.4.1 Overview

OMNeT++ [Omnet++_wiki,Omnet++,Egea05] 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.

3.4.2 Merits and Limitations

OMNeT++ [Omnet++_wiki,Omnet++,Egea05] 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.

3.5 J-Sim

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

3.5.1 Overview

J-Sim[J-sim,Egea05,Shu08,Sinha09] 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.

3.5.2 Merits and Limitations

J-Sim[J-sim,Egea05,Shu08,Sinha09] 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.

3.6 ATEMU

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

3.6.1 Overview

ATEMU [ATEMU,Polley04,Egea05,Shu08,Yi08] 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.

3.6.2 Merits and Limitations

ATEMU [ATEMU,Polley04,Egea05,Shu08,Yi08]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.

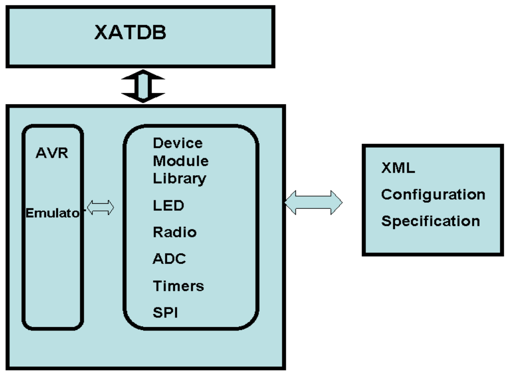


Figure 6: ATEMU Components Architecture [Polley04]

3.7 Avrora

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

3.7.1 Overview

Avrora [Avrora,Shu08,Yi08] 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.

3.7.2 Merits and Limitations

Avrora [Avrora,Shu08,Yi08]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.

4. Summary

The purpose of this survey is to give a general picture of main-stream 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 seven main-stream simulators: NS-2, TOSSIM, EmStar, OMNeT++, J-Sim, ATEMU, and Avrora, 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 Seven Main-Stream Simulation Tools

	Simulator or Emulator	Discrete-Event Simulations or Trace-Driven Simulation	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
EmStar	Emulator	Trace-Driven Simulation	Yes	Yes	specifically designed for WSNs	1.can not support large number of sensors simulation 2.only run in real time simulation and only apply to iPAQ-class sensor nodes and

						MICA2 motes
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

5. Reference

- [Sinha09]Sourendra Sinha, Zenon Chaczko, Ryszard Klempous, “SNIPER: A Wireless Sensor Network Simulator”, Computer Aided Systems Theory- EUROCAST , 2009, Volume 5717/2009, pp. 913-920, URL: http://www.springerlink.com/content/g27621hku0712916/
- [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
- [Yi08]Sangho Yi, Hong Min, Yookun Cho, Jiman Hong, “SensorMaker: A Wireless Sensor Network Simulator for Scalable and Fine-Grained Instrumentation”, computational science and its application-ICCSA, 2008, Volume 5072/2008, pp. 800-810, URL: http://www.springerlink.com/content/135t337v633v6240/
- [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, URL: http://www.cs.virginia.edu/sigbed/archives/2008-10 /NetTopo_SIGBEDReview.pdf
- [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, URL: http://gdauto.gdut.edu.cn /lab1/lwfy_files/Performance%20Evaluation%20of%20NS-2%20Simulator%20for%20Wireless%20Sensor%20Networks.pdf
- [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, URL: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5561571
- [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. URL: http://www.ece.umd.edu/~baras/publications/papers /2004/PolleyBMRB_2004.htm
- [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, URL: http://portal.acm.org/citation.cfm?id=1671365&dl=ACM&coll=DL
- [Levis03]Philip Levis, Nelson Lee, Matt Welsh, 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
- [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. URL: http://rti.etf.rs/rti/prs/materijali/lektira/The_Art_of_Computer_Systems_Performance_Analysis.pdf

- [DiscEvent_wiki]"Discrete_event_simulation",URL:http://en.wikipedia.org/wiki/Discrete_event_simulation, Description: an introduction of discrete-event-simulation in wiki webpage.
- [NS-2_wiki]"NS-2", URL: http://en.wikipedia.org/wiki/Ns-2, Description: an introduction of NS-2 in wiki webpage.
- [NS-2_isi]"NS-2", URL: http://www.isi.edu/nsnam/ns/, Description: a webpage introduced NS-2.
- [Omnet++_wiki]"Omnet++", URL: http://en.wikipedia.org/wiki/Omnet%2B%2B, Description: an introduction of Omnet++ in wiki webpage.
- [J-sim]"J-sim" , URL: http://sites.google.com/site/jsimofficial/, Description: a webpage introduced J-sim.
- [TOSSIM]"TOSSIM", URL: http://docs.tinyos.net/index.php/TOSSIM, Description: a webpage introduced TOSSIM.
- [ATEMU]"ATEMU", URL: http://www.hynet.umd.edu/research/atemu/, Description: a webpage introduced ATEMU.
- [Avrora]"Avrora", URL: http://compilers.cs.ucla.edu/avrora/, Description: a webpage introduced Avrora.
- [PowerTOSSIM]"PowerTOSSIM", URL:http://www.eecs.harvard.edu/~shnayder/ptossim/, Description: a webpage introduced PowerTOSSIM.
- [Elson03]J. Elson, S. Bien, N. Busek, V. Bychkovskiy, A. Cerpa, D. Ganesan, L. Girod, B. Greenstein, T. Schoellhammer, T. Stathopoulos, D. Estrin, "EmStar: An Environment for Developing Wireless Embedded Systems Software", 2003. URL: http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.12.9771
- [Girod04]Lewis Girod, Jeremy Elson, Alberto Cerpa, Thanos Stathopoulos, Nithya Ramanathan, Deborah Estrin, " EmStar: a Software Environment for Developing and Deploying Wireless Sensor Networks" , USENIX Technical Conference, 2004. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.111.8123
- [Omnet++]"Omnet++", URL: http://www.omnetpp.org/home/what-is-omnet, Description: a webpage introduced Omnet++.
- [SmartDust]"Smart Dust", URL: http://robotics.eecs.berkeley.edu/~pister/SmartDust/, Description: An introduction website of Smart Dust.
- [TelosB]"TelosB", URL: http://www.willow.co.uk/TelosB_Datasheet.pdf, Description: an introduction of TelosB.
- [TmoteSky]"TmoteSky",URL: http://sentilla.com/files/pdf/eol/tmote-sky-datasheet.pdf. Description: an introduction of Tmote sky
- [C++]"C++", URL: http://en.wikipedia.org/wiki/C++, Description:an introduction of C++ in wiki webpage.
- [GUI]"GUI",URL: http://en.wikipedia.org/wiki/Graphical_user_interface, Description: an introduction of GUI in wiki webpage.
- [Python]"Python", URL: http://en.wikipedia.org/wiki/Python, Description: an introduction of Python in wiki webpage.

6. List of Acronyms

- WSN: Wireless Sensor Network
- NS-2: Network Simulator Version Two
- GUI: Graphical User Interface

Last modified on April 24, 2011
This and other papers on latest advances in performance analysis are available on line at http://www1.cse.wustl.edu/~jain/cse567-11/index.html
Back to Raj Jain's Home Page