Simulators for Wireless Sensor Networks: A Review

Rakesh Sharma
Research Scolar
PTU Jalandhar

Pinki Sharma
Department of Computer Science & Engineering,
HCTM Technical Campus Kaithal, Haryana, India

V. A. Athavale,Ph.D
Department of Computer Science & Engineering,
Devraj Group of Institutions Ferozpur, Punjab,India

Sunil Kaushik
Department of Computer Science & Engineering,
HCTM Technical Campus Kaithal, Haryana, India

ABSTRACT
Wireless Sensor Networks (WSN) is made by an oversized variety of networked sensing nodes. It’s rather advanced, or perhaps unworkable, to model analytically a WSN and it always results in simple analysis with restricted confidence. Besides, deploying test-beds supposes a large effort. Therefore, simulation is important to check WSN. However, it needs acceptable model supported solid assumptions and an appropriate framework to ease implementation. Additionally, simulation results admit the actual state of affairs below study (environment), hardware and physical layer assumptions, that aren't typically correct enough to capture the behavior of a WSN, thus, make vulnerable the quality of results. However, a careful model yields to measurability and performance problems, attributable to the massive variety of nodes, that betting on application, got to be simulated. Therefore, the exchange between measurability and accuracy becomes a serious issue once simulating WSN. During this survey an acceptable model for WSN simulation is introduced, at the side of tips for choosing an acceptable framework. Additionally, a comparative description of obtainable tools is provided

Keywords
wireless sensor networks, simulators, ns2, omnet, j-sim

1. INTRODUCTION
Wireless sensing element Networks (WSN) is thought of a selected kind of Mobile Ad-hoc NETwork (MANET), shaped by lots of or thousands of sensing devices communication by suggests that of wireless transmission. Analysis on WSNs and MANETs share the similar technical issues. However in WSNs, two specific factors arise:

• The pictured applications and also the operation of the protocol layers are typically driven by the physical variables measured by the sensors. Therefore, the dynamics of the physical parameters perceived by the network govern the network traffic, and even the topology.
• The energy could be a primary concern in WSN. Usually, nodes run on non-rechargeable batteries. Therefore, the expected node period of time could be a basic component that has got to be taken into consideration. On the contrary, in MANETs, energy is a vital issue that ought to be optimized, though it's usually assumed that a node will recharge or replace its battery.

These constraints create impossible to analytically model a WSN and predict the particular performance of high-level protocols and network operation, which regularly ends up in simplistic analysis with restricted confidence. Currently, the primary real WSN applications are being explored and a few of them are nevertheless to come back. Meanwhile, deploying and operational a test-bed to check the particular behavior of protocols and network performance supposes a good effort [1], [2]. Consequently, simulation is crucial to check WSN, being the common thanks to take a look at new applications and protocols within the field. This truth has brought a recent boom of simulation tools out there to model WSN. However, getting reliable conclusions from analysis supported simulation is not a trivial task. There are two key aspects that ought to be evaluated before conducting experiments: (1) The correctness of the model and (2) the suitableness of a selected tool to implement the model. On one hand, there exists associate in nursing increasing concern regarding the methodology and assumptions of simulations [3], [4]: perfect hardware, protocols and non-realistic radio models will cause mistaken results. A “good” model supported solid assumptions is necessary to derive unsuspecting results. But, as well as the specified degree of detail adds sturdy machine needs. Large numbers of nodes in WSN, which will impersonate the additional stress on the matter. The elemental exchange is: accuracy and necessity of detail versus performance and measurability.

On the opposite hand, implementing an entire model needs a substantial effort. A tool that helps to make a model is required, and also the user faces the task of choosing the suitable one. Simulation software package normally provides a framework to model and reproduce the behavior of real systems. However, actual implementation and “secondary goals” of every tool disagree significantly, that is, some is also designed to attain smart performance et al to produce a straightforward and friendly graphical interface or emulation capabilities. The aim of this paper is to produce some insight on the building blocks of a general simulation model for WSN, introducing its specific problems.

2. A MODEL FOR WSN SIMULATION

Together with the event of simulation tools for WSN, their corresponding models are introduced. The models embrace new elements, not gift in classical network simulators, as elaborated power and energy consumption models or atmosphere models. This section describes a general part model, derived from [5], [6], for WSN simulation tools. This model is appropriate for many of the analysis tools utilized in on-going analysis on WSN.

2.1 Network model

Figure one depicts the overall model at a network-wide scale. The subsequent elements are considered:

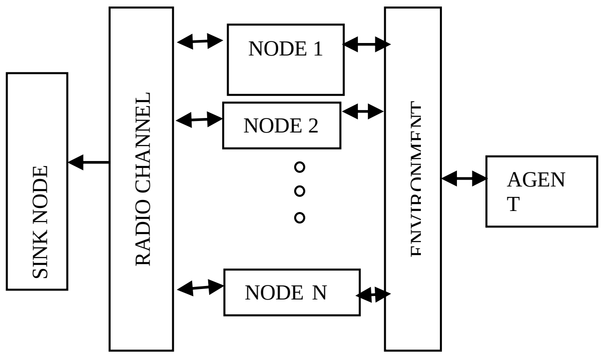

Figure 1 network model

1) Nodes: every node could be a physical device observance a collection of physical variables. Nodes communicate with one another via a typical radio channel. Internally, a protocol stack controls communications. In contrast to classical network models, detector modes embrace a second cluster of components: The physical node tier that is connected to the atmosphere. Nodes are sometimes positioned in a very two or three dimensional world. An extra “topology” part, not showed in figure one might management node coordinates.
2) Environment: the most distinction between classical and a WSN model is that the extra “environment” part. This part models the generation and propagation of events that are perceived by the nodes, and trigger detector actions, i.e. communication among nodes within the network. The events of interest are usually a physical magnitude as sound or seismic waves or temperature.
3) Radio channel: It characterizes the propagation of radio signals among the nodes within the network. Terribly elaborated models use a “terrain” part, connected to the atmosphere and radio channel elements. The tract part is taken into thought to reckon the propagation as a part of the radio channel, and conjointly influences the physical magnitude.
4) Sink nodes: These are special nodes that, if present, receive knowledge from information superhighway, and method it. They will interrogate sensors regarding an incident of interest. The utilization of sinks depends on the applying and also the tests performed by the machine.
5) Agents: A generator of events of interest for the nodes. The agent might cause a variation in a very physical magnitude, which propagates through the atmosphere and stimulates the detector. This part is helpful once its behavior is enforced severally from the atmosphere, e.g., a mobile vehicle. Otherwise, the atmosphere itself will generate events.

2.2 Node model

Node behavior depends on interacting factors that cause cross-layer interdependencies. Convenient thanks to describe it have to divide a node into abstract tiers, as diagrammatical in Figure two.

• The Protocol-tier includes all the communication protocols. Typically, three layers are at this tier: A mackintosh layer, a routing layer and a particular application layer. Note that the operation of the protocol tier sometimes depends on the state of the physical tier delineate below, e.g. a routing layer will take into account battery constraints to come to a decision on packet route. Hence, associate degree economical technique to interchange tier info should be developed.

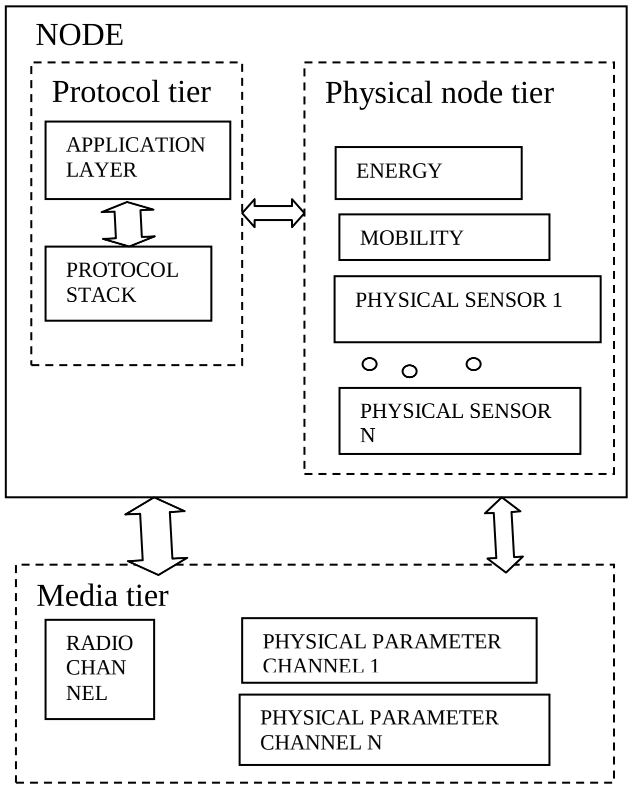

Figure 2 Tier based node model

• The physical-node tier represents the hardware platform and its effects on the performance of the instrumentality. Actual composition of this tier might modification counting on the precise application. The common parts of this tier are the set of physical sensors, the energy module and also the quality module. Physical Sensors describe the behavior of the observance hardware. Energy module simulates power consumption within the part hardware, a crucial issue in WSN analysis. Quality module controls detector position.

The media-tier is that the link of the node with the “real world”. A node is connected with the atmosphere through: (1) A radio channel, and (2) through one or additional physical channels. Physical channels receive environmental events as delineate in section two.

3. FRAMEWORK CHOICE

Widespread researches on WSN have raised a race involving several simulation tools and frameworks. The election of a simulation framework for any sort of network could be a task that's value to pay enough time. Indeed, this is often notably true for wireless sensors networks, due to the variety and quality of the simulation situations, protocols, and parts concerned. In such a heterogeneous scope, totally different analysis tools reach different goals. This section identifies and discusses the most options to be thought-about within the choice of a WSN simulation framework. A comparative description of representative simulators follows in section four. In a very opening, existing WNS frameworks is classified in: (a) Specific add-ons to general purpose communication networks (section 4.1) and (b) WSN frameworks designed from scratch (section 4.2).

3.1 The long-way road to simulation

Simulation style starts with an appropriate description of the important system. Such description constitutes the simulation model, designed up with the help of common-simulation ideas like entities, attributes, events, channels, etc. Therefore, the modeler declares the structure of the simulation in terms of entities and their relations and implements the behavior of these entities and their response to events. Common simulation packages clearly separate implementation from model description and instantiation:

• The simulation engine and also the basic model objects are provided as a collection of software package libraries in a very high level artificial language, sometimes Java or C++. This is often the simulation API.
• Some reasonably scripting (Tcl, e.g.) or mark-up language (XML, e.g.) is often utilized to support model description, that is, to determine (declare) relations between entities. Scripts enable a standardized and economical approach to model description and configuration, model mental representation of simulation runs and runtime review.
• Additionally, different utility libraries are usually enclosed like graphical illustration support or applied mathematics knowledge gathering and analysis. Therefore, a simulation framework sometimes consists of a basic simulation library, a utility library, and a few scripting support. The particular kind the package is deployed depends on the implementation. Some packages give tools that translate model scripts into objects within the implementation language to be compiled afterward. Different packages bind library and scripting so simulation objects is instantiated from a script. Others give a visible interface.

3.2 What might we have a tendency to expect from an honest WSN simulator?

Usually, the key properties to pick out appropriate simulation atmosphere are:

1) Reusability and availableness.

2) Performance and measurability.

3) Support for rich-semantics scripting languages to outline experiments and method results.

4) Graphical, correct and trace support.

In this section, we have a tendency to specialize in the impact of every feature within the context of the WSN.

3.3 Reusability and availableness

Simulation is employed to check novel techniques in realistic and controlled situations. Researchers are sometimes inquisitive about comparison the performance of a brand new technique against existing proposals. Therefore, two key aspects are: will the simulation tool embrace implementations of common models? However simple is to switch or integrate a brand new model with the prevailing ones? The primary question primarily depends on however long a framework has been used for, and the way many of us use it. Early and wide adopted frameworks have several offered models and it is terribly seemingly that the new flourishing proposals are going to be another to next releases. The second facet is closely associated with the look of the package. A careful structure with clean interfaces and high modularity permits the user to simply add or modification practicality. Ready-to-use models enable users to quickly build a practical simulation situation and specialize in modeling additional specific details of WSN. All the overall purpose packages embrace an additional or less complete TCP/IP suite, which may be thought-about the minimum customary support. Additionally, typical necessities for WSN simulators are:

Ad-hoc routing support and wireless mackintosh protocols, and propagation and quality models to synthesize the physical node distribution. For instance, these entities are unremarkably implemented: The AODV [7] for routing, the IEEE 802.11 [8] wireless mackintosh protocol, a path loss model [9] for propagation, and also the random-waypoint-based quality. For specific tools the question is subtlety different: All the precise frameworks are ready to execute native detector code. Hence, each application, protocol or part developed for the particular detector platform is simulated or emulated. Just some specific elements are strictly simulated, e.g. the radio channel or the physical media atmosphere. Summing up, during this case protocols availableness depends on the important availableness of them for the target platform, and vice versa.

3.4 Performance and measurability

Performance and measurability could be a major concern once facing WSN simulation. The previous is sometimes finite to the artificial language effectiveness. The latter is strained to the memory, processor and logs storage size necessities. To boot, the sort of simulation implies some limits: Emulation mode and time-driven simulations operate in real time in order that they can't be indiscriminately long. Wireless simulations stress performance and measurability problems as a result of the enhanced quality another by the interaction with the atmosphere, radio propagation, quality and power consumption. Simulation of many hundreds of thousands of nodes remains a difficult downside.

3.5 Support for rich-semantics scripting languages to outline experiments and method results

The configuration of a WSN typical trial needs to answer (at least) queries like: what number nodes are there within the test? wherever is every node placed?, do nodes move?, all of them?, how?, that energy model is used?, what number physical environments are?, however they generate events?, that physical magnitudes ought to live every node?, that statistics should be measured within the experiment?, that are the parameters of the radio model? The immense quantity of variables concerned within the definition of a WSN experiment needs the utilization of specific input scripting languages, with high-level linguistics. To boot, it's seemingly that enormous quantities of output knowledge will be generated through several replicas of the experiments. Therefore, an appropriate output scripting language that helps to get the results from the experiments quickly and exactly is fascinating.

3.6 Graphical, correct and trace support

Graphical support for simulations is attention-grabbing in three aspects: (1) as a debugging aid. The first and additional sensible thanks to quickly discover a nasty behavior is to “watch” and follow the execution of a simulation. The key options that a graphical interface ought to support are: Capability of review of modules, variables and event queues at real time, beside “step-by-step” and “run-until” execution potentialities. These options create graphical interfaces terribly powerful debugging tools. Note that the secret is the flexibility to move with the simulation. (2) As a visible modeling and composition tool. This feature sometimes facilitates and speeds the look of little experiments or the composition of basic modules. However, for big scale simulations, it is not terribly sensible. (3) Finally, as result plotters, this enables fast visualization of results while not a post-processing application.

4. WSN SIMULATION software package

In this section the foremost relevant simulation environments won’t to study WSN are introduced, and their main options and implementation problems delineate and mentioned. We have a tendency to essentially specialize in free, ASCII text file, simulation tools.

4.1 General simulation packages

• NS-2 [10]. Distinct event machine developed in C++. NS-2 is one in all the foremost standard non-specific network simulators, and supports a large vary of protocols altogether layers. It uses OTcl [11] as configuration and script interface. NS-2 is that the paradigm of reusability. It provides the foremost complete support of communication protocol models, among non-commercial packages. Relating to WSN, NS-2 includes ad-hoc and WSN specific protocols like directed diffusion [12] or SMAC [13]. Also, many comes will give WSN support to NS-2 like SensorSim [5] and office [14]. Each is extensions of NS- two to support WSN modeling. However, SensorSim appears to be now not offered at [15]. NS-2 will well model wired network topologies up to 1000 nodes or higher than with some optimizations. This experiment size is unbroken for wireless topologies mistreatment some new optimizations [16]. A drawback of NS-2 is that it provides meager graphical support, via Nam. This application simply reproduces a NS-2 trace. NS-2 has been an important testing tool for network analysis and, so, one may expect that the new standard protocols are going to be another to future releases. However, new proposals for WSN are progressively being tested in specific tools, e.g. TOSSIM or EmTOS (see section 4.2 for an outline of both), due to the advantage of native detector code simulation and also the specific style of those tools for WSN. Therefore, it is undecided the provision of latest WSN proposals for next releases of NS-2. This downside is also even worse for fewer used frameworks.

• OMNET++ [17] standard distinct event machine enforced in C++. Obtaining started with it's quite easy, as a result of its clean style. OMNET++ conjointly provides a strong GUI library for animation and tracing and debugging support. Its major disadvantage is that the lack of obtainable protocols in its library, compared to different simulators. However, OMNET++ is turning into a preferred tool and its lack of models is being prevented by recent contributions. As an example, a quality framework has recently been discharged for OMNET++ [18], and it is used as a place to begin for WSN modeling. To boot, many new proposals for localization and mackintosh protocols for WSN are developed with OMNET++, underneath the agreement project [19], and also the software package is publicly offered. Notwithstanding, most of the offered models are developed by freelance analysis teams and don't share a typical interface, what makes troublesome to mix them. As associate degree example, not even the localization and mackintosh protocols developed within the agreement project are compatible.

• J-Sim [20]. A component-based simulation atmosphere developed entirely in Java. It provides period of time method based mostly simulation. The most advantage of J-sim is its goodly list of supported protocols, together with a WSN simulation framework with a really elaborated model of WSNs, and an implementation of localization, routing and knowledge diffusion WSN algorithms [6]. J-sim models are simply reusable and interchangeable providing the utmost flexibility. To boot, it provides a GUI library for animation, tracing and debugging support and a script interface, named Jacl [21]. J-Sim claims to scale to an analogous variety of wireless nodes than NS-2 (around 500) with 2 orders of magnitude higher memory consumption however a forty first worse execution time [6].

• NCTUns2.0 [22]. Distinct event machine whose engines are embedded within the kernel of a UNIX system machine. The particular network layer packets are tunneled through virtual interfaces that simulate lower layers and physical devices. This notable feature permits simulations to be fed with real program knowledge sources. A helpful GUI is out there additionally to a high variety of protocols and network devices, together with wireless local area network. Sadly, no specific styles for WSN are enclosed. On one hand, the shut relationship between the simulation engine of NCTUns2.0 and also the Linux kernel machine appears a problem (adding WSN simulation modules to the present design isn't a simple task). But, on the opposite hand, real detector knowledge is simply blocked into simulated devices, protocols and actual applications, simply by putting in these sensors within the machine. NCTUns2.0 conjointly has worthy graphical edition capabilities.

• JiST/SWANS [23]. Distinct event simulation frameworks that are embed the simulation engine within the Java bytecode. Models are enforced in Java and compiled. Then, bytecodes are rewritten to introduce simulation linguistics. Afterwards, they're dead on a customary JVM. This implementation permits the utilization of un-adapted existing Java software package within the simulation, as happens with NCTUns2.0 and UNIX system programs. The most disadvantage of JiST tool is that the lack of enough protocol models. At the instant it solely provides associate degree ad-hoc network machine referred to as SWANS, designed atop JiST engine, and with a reduced protocol support. The sole graphical aid is an incident lumberjack. Jython [24] is employed as a scripting engine. JiST claims to scale to networks of 106 wireless nodes with 2 and one order of magnitude higher performance (execution time) than NS-2 and GloMoSim severally [23]. It’s been conjointly shown that it outperforms Glo- MoSim and NS-2 in event out turn and memory consumption, despite being designed with Java.

• GloMoSim [25]. Simulation atmosphere for wireless networks designed with secpar. Secpar [26] could be a simulation language derived from C that adds linguistics for making simulation entities and message communication on a spread of parallel architectures. Taking advantage of parallelization, it's been shown to scale to 10000 nodes [27]. Many proposals for WSN protocols are tested with it.

Recently, a development kit for WSN has been discharged, sQualnet [28].

• SSFNet [29]. Set of Java network models designed over the ascendible Simulation Framework (SSF). SSF could be a specification of a typical API for simulation that assures immovability between compliant simulators. There are multiple Java and C++ implementations of SSF. DartmouthSSF (DaSSF) [30], as an example, could be a C++ implementation of SSF familiarized to (parallel) simulation of terribly giant scale communication networks. Besides, specific extensions familiarized towards ad-hoc networking exists, e.g., SWAN2. SWAN is being extended to be ready to execute TinyOS code (see section four.2), in a very new framework referred to as TOSSF [31].

• Ptolemy II [32]. Java packages that support totally different models of simulation paradigms (e.g. continues time, Data flow, discrete-event). It conjointly addresses the modeling, simulation and style of coinciding, real-time, embedded systems. Ptolemy models are made in associate degree actor oriented method, terribly kind of like the component-based style of J-Sim. VisualSense [33] could be a modeling and simulations framework for WSN designed on Ptolemy II. Models are developed by sub-classing base categories of the framework or by combining existing Ptolemy models. Ptolemy visual edition assures a straightforward and intuitive graphical composition of models and result plotting.

4.2 Specific WSN frameworks

This section describes the foremost relevant tools specifically aimed to emulate and simulate the WSN hardware and software package (unlike the WSN extensions of the overall network simulators delineate within the previous section). WSN situations are sometimes extremely application-dependent, and subjected to laborious constraints that cause, in turn, a decent coupling between layers. Therefore, dedicated tools might facilitate to higher capture these dependencies. This approach conjointly permits simulating "real" application code, dashing up the migration from simulation to implementation, and facilitates testing and debugging of real applications. Emulation makes attainable real time correct and analysis of knowledge. The sole disadvantage is that the user is tied to one platform either software package or hardware (typically transparent substance Motes [34]), and to one artificial language (typically TinyOS/NesC [35]). However, TinyOS and transparent substance motes have become the de facto platform for WSN, reassuring somehow the "utility" of these tools.

4.2.1 Following environments are specifically designed for WSN research:

• TOSSIM [36]. Bit-level distinct event machine and person of TinyOS, i.e. for every transmitted or received bit an event is generated rather than one per packet. This is often attainable due to the reduced rate (around forty kbps) of the wireless interface. TOSSIM simulates the execution of nesC code on a TinyOS/MICA, permitting emulation of actual hardware by mapping hardware interruptions to distinct events. A simulated radio model is additionally provided. Emulated hardware elements are compiled beside real TinyOS elements mistreatment the nesC compiler. Thus, associate degree possible with real TinyOS applications over a simulated physical layer is obtained. To boot, there are many communication services that give how to feed knowledge from external sources. The result's a hi-fi machine and person of a network of TinyOS/MICA nodes. The goal of TOSSIM is to review the behavior of TinyOS and its applications instead of performance metrics of some new protocol. Hence, it's some limitations, as an example, it doesn't capture energy consumption. Another disadvantage of this framework is that each node should run an equivalent code. Therefore, TOSSIM cannot be wont to appraise some kinds of heterogeneous applications. TOSSIM will handle simulations around thousand of Motes. It's restricted by its bit-level granularity: Performance degrades as traffic will increase. Channel sampling is additionally simulated at bit level and consequently the utilization of a CSMA protocol causes additional overhead than would do a TDMA one.

• EmStar/EmSim/EmTOS [37] [38]. EmStar could be a software package framework to develop WSN applications on special platforms referred to as micro servers: Ad-hoc systems with higher hardware than a traditional detector. The EmStar atmosphere contains a Linux microkernel extension, libraries, services and tools. The foremost necessary tools are:

– EmSim: A machine of the micro-servers atmosphere. In EmSim each simulated node runs associate degree Em- Star stack, and is connected through a simulated radio channel model. It's not a distinct event however a time-driven machine, that is, there is no virtual clock.

– EmCee: associate degree interface to real low-power radios, rather than a simulated radio model, getting radio emulation. EmStar ASCII text file (note that this code is in any language) is employed within the simulations. To boot, the UCLA employees have developed EmTOS: associate degree extension of EmStar that permits nesC/TinyOS applications to run in associate degree EmStar framework. Thus, it opens the thanks to heterogeneous systems of detector and micro servers. Simulation of micro server and detector networks is additionally supported. Additionally, EmTOS provides 3 modes of emulation: Pure emulation, wherever all the motes are emulated by software package, "real mode", wherever all the motes are real, and "hybrid mode", wherever some motes are real et al are emulated. EmTOS reaches up to two hundred modes and it's claimed that for over five hundred nodes it'd be necessary to distribute the simulation on many processors. • ATEMU [39]. Associate degree person of the AVR processor (this processor is employed within the transparent substance platform). Whereas the operation of the speck is emulated instruction by instruction, the radio model is simulated. ATEMU conjointly provides a library of different hardware devices, e.g., timers or transceivers. Therefore, a whole hardware platform is emulated, getting two advantages: (1) the aptitude of testing OS and applications aside from TinyOS and (2) the aptitude of simulating heterogeneous networks with totally different sensors. They're achieved at the price of high process necessities and poor measurability.

• SENS [40]. Distinct event machine are enforced in C++. SENS utilizes a simplified detector model with three layers (application, network and physical) and an extra combined atmosphere and radio layer. NesC code is used directly thereon.

Table1 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 Cannot 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

• Prowler/JProwler [41]. A distinct event machine running underneath MATLAB meant to optimize network parameters. JProwler could be a version of interloper developed in Java.

• SNAP [42]. Totally different approach is used. SNAP is outlined as associate degree integrated hardware simulation- and preparation platform. It's a chip which will be utilized in two ways: (1) because the core of a deployed detector or (2) as a part of associate degree array of processors that performs parallel simulation. Again, "real" code for sensors is simulated. By combining arrays of SNAPs (called Network on a Chip), it's claimed to be ready to simulate networks on the order of 100,000 nodes.

5. CONCLUSIONS

Simulation is an important tool to review Wireless Sensor Networks as a result of the impracticableness of research and also the difficulties of putting in real experiments. This survey provides tips to assist choosing an appropriate simulation model for a WSN and a comprehensive description of the foremost used offered tools. relating to availableness of models, OMNET++, JiST and SSFNet lack of obtainable protocol models compared to different simulators (specially, NS-2), that will increase development time. Reaching to the flexibility to compose models from basic items, the part or actor based mostly packages J-Sim or Ptolemy II supply the utmost flexibility. Tools like NCTUns2.0 or JiST enable any, Linux or Java severally, application to be utilized in a simulation. This feature greatly will increase their potentialities. Specific tools like TOSSIM, EMTOS or ATEMU are ready to simulate real detector code. Relating to performance, one will expect higher performance from C/C++ engines than from their Java counterparts. However, recent simulators like JiST/SWAN claim to perform higher than NS-2 and GloMoSim (in its serial version). Obviously, parallel simulations ought to perform and scale higher than serial ones. The exchange could be a larger quality of programming. Parallel simulators as GloMoSim (whose goal is performance instead of scalability) will simulate up to around 10000 wireless nodes. DaSSF parallel tool, whose main goal is measurability, supports network topologies as giant as 100000 wired parts. All the packages give graphical support. OMNET++, NCTUns2.0, J-Sim and Ptolemy give powerful GUI libraries for animation, tracing and debugging. All they embrace the said options like review, modification of parameters at execution time, etc. OMNET++ and Ptolemy stand gently up among them. On the contrary, JiST don't embrace different graphical interface than an incident lumberjack and viewer. Current support in NS-2 is that the in elaborate and easy trace copy Nam tool. Specific tools conjointly give amazingly wealthy GUIs. TinyViz is that the TOSSIM visualization tool, associate degree protrusive Java application that gives helpful correct info. Besides, it will management and drive the simulation parts.

6. REFERENCES

[1] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, J. Anderson, “Wireless Sensor Network for Habitat Monitoring.” In Proc. 1st ACM Int. Workshop on Wireless Sensor Networks and Applications, Atlanta, GE, pp. 88– 97, September 2002.

[2] D. Ganesan, D. Estrin, A. Woo, D. Culler, “Complex Behavior at Scale: An Experimental Study of Low-power Wireless Sensor Networks.” Technical Report UCLA/CSD-TR 02-0013, Center for Embedded Networked Sensing, University of California, Berkeley, February 2002.

[3] K. Pawlikowski, H. D. Joshua Jeong, J. S. Ruth Lee. “On Credibility of Simulation Studies of Telecommunication Networks.” IEEE Communications Magazine, vol. 40, no.1, pp. 132–139, January 2002.

[4] D. Kotz, C. Newport, B. Gray, J. Liu, Y. Yuan, C. Elliot. "Experimental Evaluation of Wireless Simulation Assumptions.” In Proc. of the 7th ACM/IEEE Int. Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM’04), Venice, Italy, pp. 78–82, October 2004.

[5] S. Park, A. Savvides, M. B. Srivastava. “SensorSim: A Simulation Framework for Sensor Networks.” In Proc. ACM Modeling, Analysis and Simulation of Wireles and Mobile Systems (MSWiM 2000), Boston, MA, pp. 104–111, August 2000

[6] A. Sobeih, W. Chen, J. C. Hou, L. Kung, N. Li, H. Lim, H. Tyan, H. Zhang, “J-Sim: A simulation and emulation environment for wireless sensor networks.” In Proc. Annual Simulation Symposium (ANSS 2005), San Diego, CA, pp. 175–187, April 2005.

[7] C Perkins, EM Royer, S Das, Ad-Hoc On Demand Distance Vector Routing (AODV), IETF draft, 2000. [8] Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications, ANSI/IEEE Std. 802.11, 1999.

[9] T. S. Rappaport, Wireless Communications, principles and practice, Second Edition, Prentice Hall, 2002.

[10] The Network Simulator, NS–2 [Online]. Available: http://www.isi.edu/nsnam/ns/ [11] MIT Object Tcl. [Online]. Available:http://bmrc.berkeley.edu/research/cmt/ cmtdoc/otcl

[12] C. Intanagonwiwat, R. Govidan, D. Estrin, J. Heidemann, F. Silva, “Directed diffusion for wireless sensor networking.” IEEE/ACM Transactions on Networking, vol. 11, issue 1, pp. 2–16, February 2003.

[13] W. Ye, J. Heidemann, D. Estrin, “Medium Access Control with Coordinated, Adaptive Sleeping for Wireless Sensor Networks.” ACM/IEEE Transactions on Networking, vol. 12, pp. 493–506, 2004.

[14] NRL’s Sensor Network Extension to NS-2 [Online]. Available: http://nrlsensorsim.pf.itd.nrl.navy.mil/

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

[16] V. Naoumov, T. Gross, “Simulation of Large Ad Hoc Networks.” In Proc. ACM Modeling, Analysis and Simulation of Wireles and Mobile Systems (MSWiM 2003), San Diego, CA, pp. 50–57, 2003.

[17] OMNET++ discrete event simulator. [Online]. Available: http://www.omnetpp.org

[18] Mobility Framework for OMNET++. [Online]. Available: http://mobility-fw.sourceforge.net

[19] Consensus: Collaborative Sensor Networks [Online]. Available: http://www.consensus.tudelft.nl [20] J-Sim [Online]. Available: http://www.j-sim.org

[21] Jacl, Java implementation of Tcl8.x. [Online]. Available: http://www.tcl.tk/software/java

[22] NCTUns 2.0 Network Simulator and Emulator. [Online]. Available: http://nsl.csie.nctu.edu.tw/nctuns.html

[23] R. Barr, Z. J. Haas, R. van Renesse, “JiST: Embedding Simulation Time into a Virtual Machine.” In Proc. 5th EUROSIM Congress on Modeling and Simulation, Paris, France, September 2004

[24] Jython. [Online]. Available: http://www.jython.org

[25] Global Mobile Information Systems Simulation Library (GloMoSim). [Online]. Available: http://pcl.cs.ucla .edu/projects/glomosim/

[26] PARSEC: Parallel Simulation Environment for Complex Systems. [Online]. Available: http://pcl.cs.ucla.edu/ projects/parsec/

[27] M. Takai, R. Bagrodia, K. Tang, M. Gerla, “Efficient Wireless Networks Simulations with Detailed

Propagations Models.” Kluwer Wireless Networks, 7, pp. 297–305, 2001.

[28] sQualnet: A Scalable Simulation Framework for Sensor Networks. [Online]. Available: htpp://nesl.ee.ucla.edu/ projects/squalnet/

[29] Scalable Simulation Framework (SSF). [Online]. Available: http://www.ssfnet.org

[30] Dartmouth SSF (SSF). [Online]. Available: http://www.crhc.uiuc.edu/ jasonliu/projects/ssf/

[31] L. F. Perrone, D. M. Nicol, “A Scalable Simulator for TinyOS Applications.” In Proc. ACM 2002 Winter Simulation Conference, San Diego, CA, pp. 679–687, 2002.

[32] Ptolemy II. Heterogeneous model and design. [Online]. Available: http://ptolemy.eecs.berkeley.edu/ptolemyII

[33] P. Baldwin, S. Kohli, E. A. Lee, X. Liu, Y. Zhao. “Modeling of Sensor Nets in Ptolemy II.” In Proc. Information Processing in Sensor Networks (IPSN), Berkeley, pp.359–368, April 2004

[34] MICA Motes. [Online]. Available: http://www.xbow.com

[35] TinyOS: Open-source operating system for wireless embedded sensor networks. [Online]. Available: http://www.tinyos.net

[36] P. Levis, N. Lee, M. Welsg, D. Culler, “TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications.” In Proc. 1st ACM Int. Conf. Embedded Networked Sensor Systems (SenSys), Los Angeles, CA, pp. 126–137, 2003.

[37] L. Girod, J. Elson, A. Cerpa, T. Stathopoulos, N. Ramanathan, D. Estrin, “EmStar: A software Environment for Developing and Deploying Wireless Sensor Networks.” In Proc. USENIX 2004, Boston, MA, pp. 283–296, 2004.

[38] L. Girod, T. Stathopoulos, N. Ramanathan, J. Elson, D. Estrin, E. Osterweil, T. Schoellhammer, “A System for Simulation, Emulation and Deployment of Heterogeneous Sensor Networks.” In Proc. 2nd ACM Int. Conf. Embedded Networked Sensor Systems (SenSys), Baltimore, MD, pp. 201–213, 2004.

[39] J. Polley, D. Blazakis, J. McGee, D. Rusk, J. S. Baras, M. Karir, “ATEMU: A Fine-grained Sensor Network Simulator.” In Proc. 1st IEEE Int. Conf. Sensor and Adhoc Communication Networks (SECON’04), Santa Clara, CA, October 2004.

[40] S. Sundresh, W. Kim, G. Agha, “SENS: A Sensor, Environment and Network Simulator.” In Proc. 37th ACM Annual Symposium on Simulation, Washington, DC, pp.221, 2004.

[41] PROWLER: Probabilistic Wireless Network Simulator. [Online]. Available: http://www.isis.vanderbilt.edu/projects/nest/prowler

[42] C. Kelly, V. Ekanayake, R. Manohar, “SNAP: A Sensor-Network Asynchronous Processor.” In Proc. 9th ACM Int. Symposium on Asynchronous Circuits and Systems,Washington, DC, pp. 24, 2003.