Simulation Environments for Wireless Sensors Networks

Basavaraj.S.M[1], V.D.Mytri[2] and Siddarama.R.Patil[3]

[1] Appa Institute of Engineering and Technology, Gulbarga,Karnataka,India
[2]School of Computer Science and Engineering, XYZ University,
3 P.D.A College of Engineering GUlbarga ,Karnataka
Corresponding Adresses
{first author, second author, third author}@email.com

***Abstract*:** The emergence of wireless sensor networks brought many open issues to network designers. Traditionally, the three main techniques for analyzing the performance of wired and wireless networks are analytical methods, computer simulation, and physical measurement. However, because of many constraints imposed on sensor networks , such as energy limitation, decentralized collaboration and fault tolerance, algorithms for sensor networks tend to be quite complex and usually defy analytical methods that have been proved to be fairly effective for traditional networks. Furthermore, few sensor networks have come into existence, for there are still many unsolved research problems, so measurement is virtually impossible. It appears that simulation is the only feasible approach to the quantitative analysis of sensor networks. The goal of this paper is to aid developers in the selection of an appropriate simulation tool.

.

1. Introduction

The goal for any simulator is to accurately model and predict the behavior of a real world environment. Developers are provided with information on feasibility and reflectivity crucial to the implementation of the system prior to investing significant time and money. 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 these time and monetary investments are wise. Simulation is, therefore, the most common approach to developing and testing new protocol for a sensor networks. Many published papers contain results based only on experimental simulation. There are a number of advantages to this approach: lower cost, ease of implementation, and practicality of testing large scale networks. In order to effectively develop any protocol with the help of simulation, it is important to know the different tools available and the benefits and drawbacks therein associated. Section 2 of this paper presents the problems inherent in the use of simulation for testing, specifically applied to sensor networks. Section 3 presents a number of sensor network simulators. Section 4 provides analysis, comparing the simulators in situation-specific circumstances and making recommendations to the developers of future sensor simulators.

2. Problem Formation

NS-2 perhaps the most widely used Network Simulator, has been extended to include some basic facilities to simulate sensor Networks. However, one of the problems of ns2 is its object-oriented Design that introduces much unnecessary interdependency between modules. Such interdependency sometimes makes the addition of new protocol models extremely difficult, only mastered by those who have intimate familiarity with the simulator. Being difficult to extend is not a major problem for simulators targeted at traditional networks, for there the set of popular protocols is relatively small. For example, Ethernet is widely used for wired LAN, IEEE 802.11 for wireless LAN, TCP for reliable transmission over unreliable media. For sensor networks, however, the situation is quite different. There are no such dominant protocols or algorithms and there will unlikely be any, because a sensor network is often tailored for a particular application with specific features, and it is unlikely that a single algorithm can always be the optimal one under various circumstances.

Various network simulation environments exist in which sensor networks can be tested, including GloMoSim, OPNET, EmStar, SensorSim, ns-2, and many others. However, because of the unique aspects and limitations of sensor networks, the existing network models may not lead to a complete demonstration of all that is happening [1]. In fact, the developers in charge of ns-2 provide a warning at the top of their website indicating that their system is not perfect and that their research and development is always on-going [2]. Various problems found in different simulators include oversimplified models, lack of customization, difficulty in obtaining already existing relevant protocols, and financial cost [3]. Given the facts that simulation is not perfect and that there are a number of popular sensor simulators available, one can conclude that different simulators are appropriate and most effective in different situations. It is important for a developer to choose a simulator that fits their project, but without a working knowledge of the available simulators, this is a difficult task. Additionally, simulator developers would benefit by seeing the weaknesses of available simulators as well as the weaknesses of their own models when compared with these simulators, providing for an opportunity for improvement. For these reasons, it is beneficial to maintain a detailed description of a number of more prominent simulators available

3. Simulators

This paper will present different simulators framework. These simulators were selected based on a number of criteria including popularity, published results, and interesting characteristics and features

3.1 NS -2

NS-2 [2, 4, and 5] is the most popular simulation tool for sensor networks. It began as ns (Network Simulator) in 1989 with the purpose of general network simulation. Ns-2 is an object-oriented discrete event simulator; its modular approach has effectively made it extensible. Simulations are based on a combination of C++ and OTcl. In general, C++ is used for implementing protocols and extending the ns-2 library. OTcl is used to create and control the simulation environment itself, including the selection of output data. Simulation is run at the packet level, allowing for detailed results.

NS-2 sensor simulation is a modification of their mobile ad hoc simulation tools, with a small number of add-ons. Support is included for many of the things that make sensor networks unique, including limited hardware and power. An extension developed in 2004[4] allows for external phenomena to trigger events. Ns-2 extensibility is perhaps what has made it so popular for sensor networks. In addition to the various extensions to the simulation model, the object-oriented design of ns-2 allows for straightforward creation and use of new protocols. The combination of easy in protocol development and popularity has ensured that a high number of different protocols are publicly available, despite not be included as part of the simulator's release. Its status as the most used sensor network simulator has also encouraged further popularity, as developers would prefer to compare their work to results from the same simulator.

NS-2 does not scale well for sensor networks. This is in part due to its object-oriented design. While this is beneficial in terms of extensibility and organization, it is a hindrance on performance in environments with large numbers of nodes. Every node is its own object and can interact with every other node in the simulation, creating a large number of dependencies to be checked at every simulation interval, leading to an n^2 relationship. Another drawback to ns-2 is the lack of customization available. Packet formats, energy models, MAC protocols, and the sensing hardware models all differ from those found in most sensors. One last drawback for NS-2 is the lack of an application model. In many network environments this is not a problem, but sensor networks often contain interactions between the application level and the network protocol level.

3.2 SensorSim

SensorSim is a simulation framework for modeling sensor networks. It is build upon on the NS-2 simulator and provides additional features for modeling sensor networks . SensorSim [6] uses ns-2 as a base, and extends it in three important ways. First, it includes an advanced power model. The model takes into account each of the hardware components that would need battery power in order to operate. The developers researched the affects of each of these different components on energy consumption in order to create their power model. It is included as part of the sensor node model (Figure 1).

Secondly, SensorSim includes a sensor channel. This was a precursor to the phenomena introduced to ns-2 in 2004. Both function in approximately the same way. SensorSim's model is slightly more complicated and includes sensing through both a geophone and a microphone. However, the model is still simplistic, and the developers felt that another means of including more realistic events was needed.

This led to the third extension to ns-2: an interaction mechanism with external applications. The main purpose is to interact with actual sensor node networks. This allows for real sensed events to trigger reactions within the simulated environment. In order to accomplish this, each real node is given a stack in the simulation environment. The real node is then connected to the simulator via a proxy, which provides the necessary mechanism for interaction.

One further extension to ns-2 is the use of a piece of middleware called Sensor Ware. This middleware makes it possible to dynamically manage nodes in simulation. This provides the user with the ability to provide the network with small application scripts than can be dynamically moved throughout the network. This ensures that it is not necessary to preinstall all possible applications needed by each node, and provides a mechanism for distributed computation. Because of the battery model and sensor channel, improvements were made in the associated hardware models when compared with ns-2. However, especially in the case of the sensing hardware, the models are still very simple and do not accurately reflect what is found on most sensors. Like ns-2, SensorSim faces a scalability problem. Additionally, SensorSim is not being maintained and is not currently available to the public.

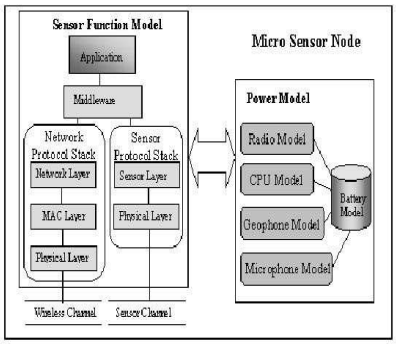

Figure 1. Micro sensor node model in SensorSim

3.3 SENSE

The SENSE is designed to be an efficient and powerful sensor network simulator that is also easy of use. The SENSE [7] simulator is influenced by three other models. It attempts to implement the same functionality as ns-2. However, it breaks away from the object-oriented approach, using component based architecture. It also includes support for parallelization. Through its component-based model and support for parallelization, the developers attempt to address what they consider to be the three most critical factors in simulation: extensibility, reusability, and scalability. SENSE was developed in C++, on top of COST, a general purpose

discrete event simulator. It implements sensors as a collection of static components. Connections between each component are in the format of in ports and out ports (Figure 2). This allows for independence between components and enables straightforward extensibility and reusability. Traversing the ports are packets. Each packet is composed of different layers for each layer in the sensor. The designers try to improve scalability by having all sensors use the same packet in memory, assuming that the packet should not have to be modified. SENSE's packet sharing model is an improvement on ns-2 and other object-oriented models that can not do this, helping improve scalability by reducing memory use. However, the model is simplistic and places some communication limitations on the user. While SENSE implements the same basic functionality as ns-2, it can not match the extensions to the ns-2 model. Whether because it is a new simulator, or because it has not achieved the popularity of ns- 2, there has not been research into adding a sensing model, eliminating physical phenomena and environmental effects.

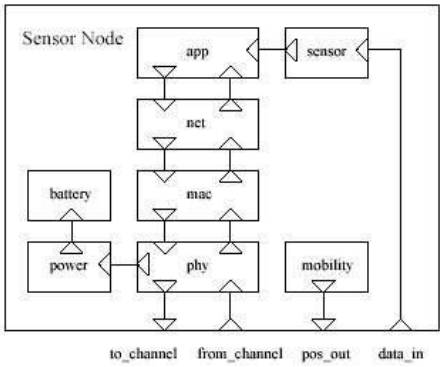

Figure 2. SENSE's sensor node structure with ports

3.4 Mannasim

Mannasim[9] goal is to develop a detailed simulation framework, which can accurately model different sensor nodes and applications while providing a versatile test bed for algorithms and protocols . Numerous challenges make the study of real deployed sensor networks very difficult and financially infeasible. At the current stage of the technology, a practical way to study WSNs is through simulations that can provide a meaningful perspective of the behavior and performance of various algorithm This framework is free software and it can be redistributed under the GNU Public License

Mannasim is a Wireless Sensor Networks simulation environment comprised of two solutions: The Mannasim Framework, The Script Generator Tool

The Mannasim Framework is a module for WSN simulation based on the Network Simulator (NS-2). Mannasim extends NS-2 introducing new modules for design, development and analysis, development and analysis of different WSN applications. The Script Generator Tool (SGT) is a front-end for TCL simulation scripts easy creation. SGT comes blunded with Mannasim Framework and it's written in pure Java making it platform independent

3.5 EYES WSN Simulation Framework

At the start of the EYES WSN[10] project the template was needed to be built because the OMNeT + + simulator did not include support for mobile networks that communicate using radios. Although the existing ones are quite complicated to use, we tried to build a simple simulation framework and we have recently extended it with a language translator tool named NesCT. With the benefit of this tool we are able to run most of the code written in TinyOS[11] using Omnet + +[12] and our simulation framework. This tool is a general purpose language translator and with some trivial customization it's also possible to make it work with other environments too. The framework was designed in such a way that allows easy modifications of the main parameters and, at the same time, the implementation details are transparent to the user.

Mobility is implemented (Random Way Point algorithm by default). Each node is responsible for defining its own trajectory and announcing it to the simulator. Nodes exchange messages using wireless communication. A message will be heard by all the neighbours situated within the transmission range (the modules within transmission range are connected automatically to each-other).The user can specify if unidirectional or bidirectional links have to be used. Each node can specify and update its transmission range independently.The nodes have different kinds of failing probabilities (individual failures, failures that affect regions of the map, etc.) Maps for area failures can be specified and used. Other maps can easily used for obstacles, fading, etc. In order to perform all of this features we have chosen to use.

3.6 NS-2 MIUN

Ns-2 is a popular Open source Network Simulator. A lot of researchers in the community of Wireless Sensor Networks have used ns2 to verify their research results. However, ns2 is not an Easy tool for the simulation of Sensor Networks, partially because of its high difficulty in understanding the ns2 itself, and also because there is currently lack of support for sensor network simulation.

Ns modified to support wireless sensor network simulation, with a specialty on intrusion detection simulation. This enhanced parts is named NS2-MIUN[13]
The enhanced features include.

The Integration of **NRL's phenomenon node,** which enables the ability of simulating an environmental phenomenon.

The Integration of **AODVUU,** which is an AODV Routing Protocol implementation that follows AODV specification better than the one included in the Standard ns2 Release.

The definition of a new packet type PT_SensorApp, which is used to simulate the type of packets used by sensor application.

The support of dynamic packet destination configuration. In the standard ns2 release, the <src, dst> pair is configured by statically binding an agent in the source node with an agent in the destination node in the TCL scenario file. This means a source node needs to configure multiple source agents when there are multiple potential recipients and bind

each potential <src, dst> pair manually at the configuration file. This doesn't scale well in a dynamic wireless sensor network, where the destination node can vary over time. This drawback is fixed by allowing run-time <src, dst> binding.

The integration of an intrusion detection module. It is a module inserted between the MAC layer and the network layer that captures all packets and impose intrusion detection analysis.The imitation of different attacks. The attacks implemented include wormhole, symbil / ID spoofing, DOS / DDOS, sinkhole, etc. An Extension for Simulating multi-homed nodes [14]ns-2 also is provided.

4. Analysis

This paper by no means presents an exhaustive list of sensor simulators. But the most of the issues facing the developers of sensor networks can be seen in this paper. Of course, many decisions must be made for specific situations rather than following all encompassing guidelines.

The developers must decide whether they want a simulator or an emulator. Each has advantages and disadvantages, and each is appropriate in different situations. Generally, a simulator is more useful when looking at things from a high view. The effect of routing protocols, topology, and data aggregation can be see best at a top level and would be more appropriate for simulation. Emulation is more useful for fine-tuning and looking at low-level results. Emulators are effective for timing interactions between nodes and for fine tuning network level and sensor algorithms.

If the developers decide to build a simulator, another design level decision that must be made is whether to build their simulator on top of an existing general simulator or to create their own model. If development time is limited or there is one very specific feature that the developers would like to use that is not available, then it may be best to build on top of an existing simulator. However, if there is available development time and the developers feel that they have a design that would be more effective in terms of scalability, execution speed, features, or another idea, then building a simulator from the base to the top would be most effective.

In building a simulator from the bottom up, many choices need to be made. Developers must consider the pros and cons of different programming languages, the means in which simulation is driven (event vs. time based), component-based or object oriented architecture, the level of complexity of the simulator, features to include and not include, use of parallel execution, ability to interact with real nodes, and other design choices. While design language choices are outside of the scope of this paper, there are some guidelines that appear upon looking at a number of already existing simulators.

Most simulators use a discrete event engine for efficiency. Component-based architectures scale significantly better than object-oriented architectures, but may be more difficult to implement in a modularized way. Defining each sensor as its own object ensures independence amongst the nodes. The ease of swapping in new algorithms for different protocols also appears to be easier in object-oriented designs. However, with careful programming, component based architectures perform better and are more effective. Generally, the level of complexity built into the simulator has a lot to do with the goals of the developers and the time constraints imposed. Using a simple MAC protocol may suffice in most instances, and only providing one saves significant amounts of time. Other design choices are dependent on intended situation, programmer ability, and available design time.

5. Conclusion

The goals of this paper were to provide background on a number of different sensor network simulators and present the best and worst features of each. The purpose was three-fold. First, knowing the strengths and weaknesses of a number of different simulators is valuable because it allows users to select the one most appropriate for their testing. Second, the developers of new simulators are well served knowing what has worked in previous simulators and what has not. It also allows user to know how to scale NS-2 to suite their problem for simulating sensor networks

References

[1] S. Park, A. Savvides and M. B. Srivastava. Simulating Networks of Wireless Sensors. *Proceedings of the 2001 Winter Simulation Conference*, 2001.

[2] The Network Simulator – ns-2. http://www.isi.edu/nsnam/ns.

[3] Vlado Handziski, Andreas Köpke, Holger Karl, and Adam Wolisz. "A Common Wireless Sensor Network Architecture?". *Sentzornetze*, July 2003.

[4] Ian Downard. Simulating Sensor Networks in ns-2. *NRL Formal Report* 5522, April, 2004.

[5] Valeri Naoumov and Thomas Gross. Simulation of Large Ad Hoc Networks. *ACM MSWiM*, 2003.

[6] Sung Park, Andreas Savvides, and Mani B.Srivastava. SensorSim: A Simulation Framework for Sensor Networks. *ACM MSWiM*, August, 2000.

[7] Gilbert Chen, Joel Branch, Michael Pflug, Lijuan Zhu, and Boleslaw Szymanski.SENSE: A Sensor Network Simulator. *Advances in Pervasive Computing and Networking*, 2004.

[8] Jonathan Pollet, et al. ATEMU: A Fine-Grained Sensor Network Simulator. *Proceedings of SECON'04, First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, 2004.

[9] The Mannasim Framework
http://www.mannasim.dcc.ufmg.br/

[10] Eye WSN Framework
http://wwwes.cs.utwente.nl/ewsnsim/

[11] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. *Proceedings of SenSys'03, First ACM Conference on Embedded Networked Sensor Systems*, 2003.

[12] C. Mallanda, A. Suri, V. Kunchakarra, S.S. Iyengar, R. Kannan, and A. Durresi. Simulating Wireless Sensor Networks with OMNeT++.

[13]NI-MIUM
http://apachepersonal.miun.se/~qinwan/resources.htm.

[14] Simulating Wireless Multihomed Node in NS-2 ,Qinghua Wang, Tingting Zhang, Department of Information Technology and Media, Mid Sweden University, Sundsvall, SE 85170, Sweden