IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
e-ISSN: 2278-2834,p- ISSN: 2278-8735. Volume 5, Issue 2 (Mar. - Apr. 2013), PP 54-60
www.iosrjournals.org

Comparison between Different Wireless Sensor Simulation Tools

Mrs. Poonam Chhimwal[1], Dhajvir Singh Rai[2], Deepesh Rawat[3]
[1]*Asst. Prof.(Department of Computer Science),BTKIT Dwarahat, Uttarakhand,India*
[2]*M.Tech(Computer Science & Engineering),BTKIT Dwarahat Uttarakhand,India*
[3]*M.Tech(Digital Communication) BTKIT Dwarahat Uttarakhand,India*

Abstract: *Wireless sensor network have recently come into prominence because they hold the potential to revolutionize segment of our economy and life with its wide applications. Unlike a centralized system, a sensor network is formed by large numbers of networked sensing nodes. They are having unique set of resource constraints such as finite onboard battery power and limited network communication bandwidth. Since running real experiments is costly and time consuming, simulation is essential to study Wireless Sensor Networks (WSN). It is rather complex, or even unfeasible, to model analytically a WSN and it usually leads to oversimplified analysis with limited confidence. Therefore, simulation is essential to study WSN. The word "wise" in the title indicate the meaning most appropriate according to their usage and testing*

Keywords: *Sensor network, simulation, tools*.

I. INTRODUCTION

The sensors 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 have become representative of Wireless sensor networks applications [1].

The sensor chip sprayed on roads, walls or machine creates a digital skin that senses the physical phenomena of interest:-

- It can monitor the vehicular traffic, Pedestrian, intelligent transportation grid.
- It can report wildlife habitat condition for environment conservation.
- It helps in detecting forest fire.
- It can track job flow and supply chain in industries..

II. SENSOR NETWORK PROTOCOL

The Wireless sensor network consists of various sensor nodes as shown in fig 1. They are having unique set of resource constraints such as finite onboard battery power and limited bandwidth.

The protocol stack consists of the application layer, transport layer, network layer, data link layer, physical layer, power management plane, mobility management plane, and task management plane as shown in Fig 2. Depending on the sensing tasks, different types of application software can be built and used on the application layer.

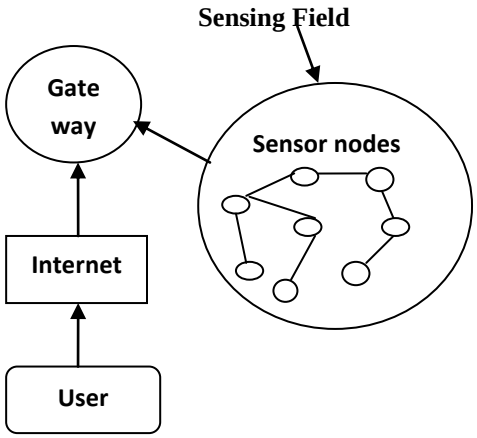

Fig 1: A Wireless Sensor Network

Simulation is essential to study WSN. It requires suitable model based on solid assumption and appropriate framework to ease implementation. Simulation is the process of designing a model of a real system and conducting experiments with this model for the purpose of either understanding the behaviour of the system and/or evaluating various strategies for the operation of the system. A detailed discussion of simulation methodology, in general, can be found in [11, 12].

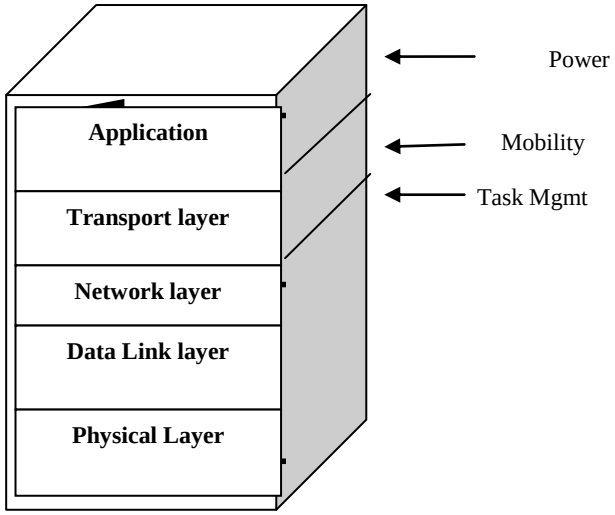

Fig 2:-The sensor networks protocol stack

The simulation result depends upon on the environment and physical layer assumption which may not be accurate to predict the real behavior of wireless sensor network. Simulation is necessary to test the application and protocols in this field. The correctness of the model and Suitability of model for the implementation are necessary factors of WSN simulations.

The key properties of good Simulator:-

- Reusability and availability
- Performance and scalability.
- Support for rich-semantics scripting languages to define experiments and process results.
- Graphical, debug and trace support.

III. A MODEL FOR WSN SIMULATION

A. 3.1 Network model

The following components are considered as shown in Fig 3 [4][5]:

1. *Nodes:* Each node is a physical device monitoring a set of physical variables. Nodes communicate with each other via a common radio channel.
2. *Environment:* The main difference between classical and WSN models are the additional "environment" component. This component models the generation and propagation of events that are sensed by the nodes, and trigger sensor actions, i.e. communication among nodes in the network. The events of interest are generally a physical magnitude as sound or seismic waves or temperature.

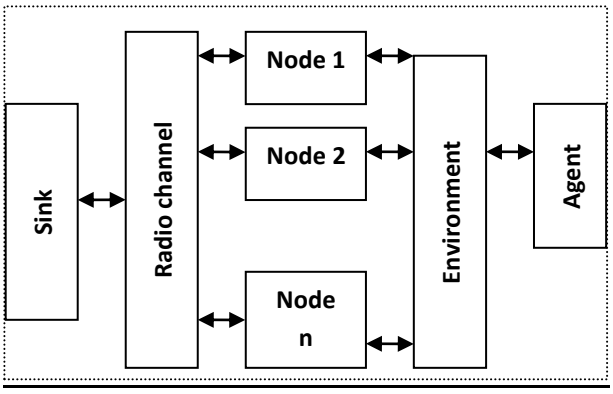

Fig3: Wireless Sensor Network Model

3. *Radio channel:* It characterizes the propagation of radio signals among the nodes in the network. Very detailed models use a "terrain" component, connected to the environment and radio channel components. The terrain component is taken into consideration to compute the propagation as part of the radio channel, and also influences the physical magnitude.
4. *Sink nodes:* These are special nodes that, if present, receive data from the net, and process it. They may interrogate sensors about an event of interest. The use of sinks depends on the application and the tests performed by the simulator.
5. *Agents:* The agent may cause a variation in a physical magnitude, which propagates through the environment and stimulates the sensor. This component is useful when its behavior can be implemented independently from the environment, e.g., a mobile vehicle. Otherwise, the environment itself can generate events.

B. 3.2 Node model

This model divides a node into abstract tiers as shown in Fig 4 [4] [5]:-

The protocol tier comprises all the communication protocols. Typically, three layers coexist at this tier: A MAC layer, a routing layer and a specific application layer.

The Physical node tier represents the hardware platform and its effects on the performance of the equipment.

The media tier is the link of the node with the "real world". A node is connected with the environment through:

(1) A radio channel, and (2) Through one or more physical channels

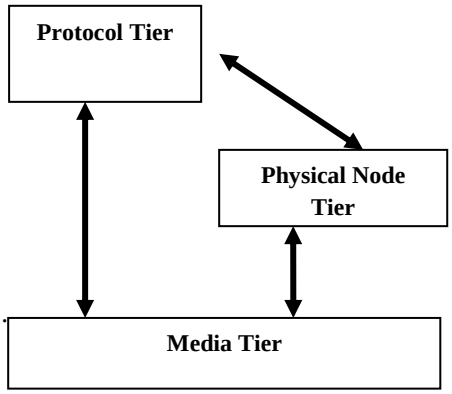

Fig 4: Tier-based Node Model

IV. WSN SIMULATION SOFTWARE

To get an insight into the simulation tools used in the selected IEEE Journal and Conference published papers, Nurul I. Sarkar *et. al.* [10] surveyed all papers published in the IEEE Transactions on Communications (1071 papers), IEEE/ACM Transactions on Networking (377 papers), and in proceedings of IEEE GLOBECOM (2991papers), INFOCOM (817 papers), and ICC (3114 papers) between 2007 and 2009. A total of 8370 papers were surveyed. The survey results were summarized about 42.8% of 8370 papers surveyed have mentioned that they use ns-2 for network modeling and simulation tasks. About 36.8% of the total papers surveyed have used MATLAB whereas 7.6% used OPNET. The remaining 4.2%, 1.6% and 0.8% of the total papers surveyed have used QualNet, GlomoSim, and OMNet++, respectively. And about 6.2% of the papers surveyed did not bother to mention the name of the simulators that they had used. That's why I have chosen the five below mentioned tools on the basis of their widely accepted usage.

4.1 NS-2

NS-2 is the abbreviation of Network simulator version two, which first been developed by 1989 using as the REAL network simulator [2][3]. 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++. People can run NS-2 simulator on Linux Operating Systems or on Cygwin, which is a Unix-like environment and command-line interface running on Windows. NS-2 is a popular non-specific network simulator can used in both wire and wireless area. This simulator is open source and provides online document.

Language:-

Object- Oriented extension of Tool Command Language and C++

Key feature:-

- NS-2 extensibility features.
- Object oriented design allow creating and using of new protocol.
- It provide visualization tool-NAM (Network Animator)

Limitation:-
However, this simulator has some limitations [2][3].

- People who want to use this simulator needs to be familiar with writing scripting language and modeling technique; the Tool Command Language is somewhat difficult to understand and write.
- Sometimes using NS-2 is more complex and time-consuming than other simulators to model a desired job.
- NS-2 provides a poor graphical support, no Graphical User Interface (GUI); the users have to directly face to text commands of the electronic devices.
- Due to the continuing changing the code base, the result may not be consistent, or contains bugs.

*4.2 **NS-3***
The NS-3 simulator is an open-source and discrete-event network simulator specially meant for research and educational use. It was first developed in 2006.Many version have been so far released after its first development. The recent version NS-3.13 was released on 23 December 2011.

Language:-
C++, Python

Key feature:-

- NS-3 is not an extension of NS-2; it is a new simulator. The two simulators are both written in C++ but NS-3 is a new simulator that does not support the NS-2 APIs. Some models from ns-2 have already been ported from NS-2 to NS-3. The project will continue to maintain NS-2 while NS-3 is being built, and will study transition and integration mechanisms.
- NS-3 is open-source, and the project strives to maintain an open environment for researchers to contribute and share their software.

Limitation:-

- Python bindings do not work on Cygwin [9].
- Only IPv4 is supported [9].

4.3 TOSSIM

It is discrete event simulator for TinyOS Wireless Sensor Network, which is open source operating system targeting embedded operating system [3]. It was first developed at UC Berkeley. TOSSIM is a bit-level discrete event network emulator built in 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.
Environment:
It runs on custom mote hardware. It chooses the accuracy and complexity of model necessary for their simulation.
Language:-
Python, NesC, C++
Key Features:
It provides interaction with the networks due to its graphical support. Packet can be dynamically injected into the network. Packet traffic can be easily monitored in his way.
Advantages:

- Open Source and online documentation
- Graphical User Support (Tiny ViZ)
- Simple and powerful emulator for Wireless sensor Network [3]
- Support thousand of Nodes

Limitation:
It is specially designed for tinyOS, not designed for simulation performance metrics of other new protocol [3].Therefore, TOSSIM cannot correctly simulate issues of the energy consumption in WSN; people can use 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.

4.4 OMNeT++

Modular discrete event simulator implemented in C++. Getting started with it is quite simple, due to its clean design. OMNET++ also provides a powerful GUI library for animation and tracing and debugging support. Its major drawback is the lack of available protocols in its library, compared to other simulators [3].
However, OMNET++ is becoming a popular tool and its lack of models is being cut down by recent contributions. For instance, a mobility framework has recently been released for OMNET++ , and it can be used as a starting point for WSN modeling. Additionally, several new proposals for localization and MAC protocols for WSN have been developed with OMNET++, under the Consensus project, and the software is publicly available. Nevertheless, most of the available models have been developed by independent research groups and do not share a common interface, what makes difficult to combine them. As an example, not even the localization and MAC protocols developed in the Consensus project are compatible.
Advantages:

- Powerful Graphical User Interface (making tracing and bugging easier)
- Simulate power Consumption problem

Limitation:

- Number of protocol is not large enough.
- Compatibility problem (not portable)

We can say that both advantages and disadvantages are included in the OMNeT++ design.

4.5 J-Sim

A component-based simulation environment developed entirely in Java. It provides real-time process based simulation. The main benefit of J-Sim is its considerable list of supported protocols, including a WSN simulation framework with a very detailed model of WSNs, and a implementation of localization, routing and data diffusion WSN algorithms [2]. J-Sim models are easily reusable and interchangeable offering the maximum flexibility. Additionally, it provides a GUI library for animation, tracing and debugging support.
Advantages:

- Models in J-Sim have good reusability and interchangeability, which facilities easily simulation [2].
- 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.
- 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.

Limitation:

- J-Sim is relatively complicated to use.
    - The execution time is much longer than that of NS-2.

4.6 Castalia

Castalia is a simulator for Wireless Sensor Networks (WSN), Body Area Networks (BAN) and generally networks of low-power embedded devices. It is based on the OMNeT++ platform and can be used by researchers and developers who want to test their distributed algorithms and/or protocols in realistic wireless channel and radio models, with a realistic node behavior especially relating to access of the radio. Castalia can also be used to evaluate different platform characteristics for specific applications, since it is highly parametric, and can simulate a wide range of platforms [7].
Key Features:-

- Advanced channel model based on empirically measured data.
- Advanced radio model based on real radios for low-power communication.
- Extended sensing modeling provisions
- Node clock drift
- MAC and routing protocols available.
- Designed for adaptation and expansion.

Limitation:-
Castalia is not sensor-platform specific.

4.7 QualNet

A Commercial derivation of GloMoSim based on C++. It was released in 2000 by Scalable Network Technology (SNT).GloMoSim is academic research version (an open source) but QualNet[8] is commercial version. It is GUI based model used for design, animation and analysis.

Key Features:-

- High fidelity commercial protocol and device models.
- Comparative performance evaluation of alternative protocol at each layer.
- Built in measurement on each layer.
- Modular layer stack design.
- Scalability via support for parallel execution.

V. COMPARISONS BETWEEN DIFFERENT SIMULATORS

The key features and limitations of each of these simulators are highlighted in Table 1.

No	Simulator	Programming language	Merits	Demerits
1	Ns-2	C++	-Easy to add new protocols. -A large number of protocols available publicly. -Availability of a visualization tool.	-Supports only two wireless MAC protocols, 802.11, and a single-hop TDMA protocol. -Need to familiar with writing scripting language
2	NS-3	C++	-NS-3 is not an extension of NS-2; it is a new simulator. -NS-3 is open-source	-Python bindings do not work on Cygwin. -Only IPv4 is supported.
3	TOSSIM	nesC	-High degree of accuracy or running the application source code unchanged. -Availability of a visualization tool.	-Compilation steps lose the fine grained timing and interrupt properties of the code.
4	J-Sim	Java	-Provides support for energy modeling, with the exception of radio energy consumption -Support mobile wireless networks and sensor networks. -Component-oriented architecture.	-Low efficiency of simulation. -The only MAC protocol provided for wireless networks is 802.11. -Unnecessary run-time overhead
5	OMNeT++	C++	-Powerful graphical User Interface (making tracing and bugging easier) -Simulate power Consumption problem	-Number of protocol is not large enough. -Compatibility problem (not portable)
6	Castalia	C++	-Physical process modeling, sensing device bias and noise, node clock drift, and several MAC and routing protocols implemented. -Highly tunable MAC protocol and a flexible parametric physical process model.	-Not a sensor specific platform. -Not useful if one would like to test code compiled for a specific sensor node platform.
7	QualNet	C++	-Easy-to-use and clear User Interface. -Sophisticated animation capabilities. -Support for multiprocessor systems and distributed computing.	-Difficult installation on Linux. -Slow Java-based UI. -Very expensive.

Table 1. Comparison between different wireless sensor simulation tools.

VI. CONCLUSION

Simulation is an essential tool to study Wireless Sensor Networks due to the unfeasibility of analysis and the difficulties of setting up real experiments. This study provides guidelines to help selecting a suitable simulation model for a WSN and a comprehensive description of the most used available tools. The goals of this paper have been to provide comprehensive study and background on a number of different sensor network simulators and present the pros and cons of each simulator. Knowledge of 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.

REFERENCES

[1] D. Estrin, R. Govindan, J. Heidemann, S. Kumar, "Next century challenges: scalable coordination in sensor networks", ACM MobiCom'99, Washington, USA, 1999, pp. 263–270.
[2] Sourendra Sinha, Zenon Chaczko, Ryszard Klempous, "SNIPER: A Wireless Sensor Network Simulator", Computer Aided Systems Theory- EUROCAST , 2009, Volume 5717/2009, pp. 913-920
[3] 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
[4] S. Park, A. Savvides, M. B. Srivastava. "SensorSim: A Simulation Framework for Sensor Networks." In Proc. ACM Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM 2000), Boston, MA, pp. 104–111, August 2000
[5] 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, April2005.
[6] Harsh Sundani, Haoyue Li, Vijay K. Devabhaktuni, Mansoor Alam, & Prabir Bhattacharya, "Wireless Sensor Network Simulators A Survey and Comparisons." International Journal of Computer Networks (IJCN), Volume2: Issue 5, 2010, pp 250
[7] http://castalia.npc.nicta.com.au/documentation.php
[8] Tobias Doerel, "Simulation of wireless ad-hoc sensor networks with QualNet", Chemnitz, 2009
[9] The NS-3 network simulator :http://www.nsnam.org/
[10] Nurul I. Sarkar and Syafnidar A. Halim," A Review of Simulation of Telecommunication Networks: Simulators, Classification, Comparison, Methodologies, and Recommendations", Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications (JSAT), March Edition, 2011
[11] A. M. Law and W. D. Kelton, "Simulation modelling and analysis", third Edition. New York: McGraw-Hill, 2000.
[12] J. S. Carson II, "Introduction to Modeling and Simulation," Winter Simulation Conference, December, 2004, pp. 1283-1289.