CML-WSN: A Configurable Multi-layer Wireless Sensor Network Simulator

Carolina Del-Valle-Soto[1(✉)], Fernando Lezama[2], Jafet Rodriguez[1], Carlos Mex-Perera[3], and Enrique Munoz de Cote[2]

[1] Campus Guadalajara, Facultad de Ingeniería, Universidad Panamericana, Prolongación Calzada Circunvalación Poniente 49, 45010 Zapopan, Jalisco, Mexico
{cvalle,arodrig}@up.edu.mx
[2] Instituto Nacional de Astrofísica, Óptica y Electrónica, 72840 Puebla, Mexico
{f.lezama,jemc}@inaoep.mx
[3] Telemática Telemetría y Radiofrecuencia, 44190 Guadalajara, Jalisco, Mexico
carlosmex@ttr.com.mx

Abstract. Wireless Sensor Networks (WSNs) have large applications in environments where access to human cannot be constant or where reliable and timely information is required to support decisions. WSNs must show high reliability, robustness, availability of information, monitoring capabilities, self-organization, among other aspects. Also, engineering requirements, such as low-cost implementation, operation, and maintenance are necessary. In this context, a simulator is a powerful tool for analyzing and improving network technologies used as a first step to investigate protocol design and performance test on large-scale systems without the need of real implementation. In this paper, we present a Configurable Multi-Layer WSN (CML-WSN) simulator. The CML-WSN simulator incorporates a configurable energy model to support any sensor specification as a one of its main features. The CML-WSN simulator is useful because it allows exploring prototypes with much less cost and time compared to the requirements needed in real networks implementations.

Keywords: Wireless Sensor Networks · Network simulator · C++ · Object-oriented programming

1 Introduction

Nowadays, from the most fundamental electrical appliances like toasters to the most advanced devices like industrial machines, devices have the capability of communicating with each other due to the advances in technology. This communication is enabled by a network that connects each device so they can work together towards a common goal such as reducing the risk in a manufacturing process. In order to control all the devices, sensors are required to determine when is safe to operate and which one should be selected. A wireless sensor network (WSN) is the compound of nodes which collaborate in a common task.

E. Sucar et al. (Eds.): AFI 2016, LNICST 179, pp. 91–102, 2017.
DOI: 10.1007/978-3-319-49622-1_11

These nodes have certain sensory capabilities and wireless communication that allow building ad-hoc networks (i.e., without pre-established physical structure or central administration) [1].

One of the main problems about ad-hoc systems is that there is no infrastructure, so the routes change dynamically. This dynamic change is due to fading, interference, disconnection of nodes, obstacles, node movements, among others. In consequence, problems such as quality of services, mobile battery-perishable, security, reliability of routes and further more appear and cause packet loss [2].

The majoring of this work focuses on engineering and telecommunications, specifically in the WSN field. The use of network simulators provides a more accurate perception of topologies, technologies, transmission media, channel design and pieces of equipment that best suit to the application that is being designed. Moreover, network simulation plays a significant role in areas such as engineering and research.

In this paper, we present a Configurable Multi-Layer WSN (CML-WSN) simulator. The CML-WSN is based on discrete events and implemented in C++. The simulator was intended to overcome some of the limitations presented in other simulators. This new approach allows the user to operate in more than a single layer, specifically in the Physical, MAC, and Network layers. Furthermore, its design permits the inclusion of different routing protocols and functionalities which translates into a scalable software.

The goal of creating CML-WSN simulator is to have the ability to manipulate parameters as needed and observe different network behaviors. Implementing a new simulator from the ground up allows a better view of the nodes, change features and models, and analyze several kinds of statistics needed to study the performance of routing protocols and its influence on the network. The only requirement is to have the data sheet of the wireless sensor chip which the user wants to use. The simulator will provide the energy consumption from the primary task of a node in the network, some of the usual tasks are starting, shutting down, receiving, transmitting, switching, CSMA (Carrier Sense Multiple Access/Collision Avoidance) algorithm and microcontroller.

Another purpose of CML-WSN is to deploy large networks with low costs considering real parameters. The results of a simulation can provide good scalability projections. Besides, our CML-WSN employs an energy model as close to the activity of a common node to optimize future decisions.

Also, with the simulation process, the time can be compressed or expanded allowing an increase or decrease of the speed of the research phenomena. In other words, the design allows the user to easily create and modify highly complex and different scenarios based on large amounts of various input parameters like power of reception and transmission locally and globally, size of the packet, packet's transmission rate, buffer storage size, routing protocol, sampling time, energy model, states of each node and others.

The chosen programming language was C++ due to its fast performance and ease to compile in multiple platforms making it a tool that students, network technicians, and researchers can use to analyze and design better WNS networks.

2 Related Work

WSNs are extensively studied with several network simulators that analyze various performance and power consumption parameters. However, the use of these simulators is intended to study certain topologies or already defined and parameterized environments. The comparison between WSN simulators is out of the scope of this paper. However, in this section we give a brief overview of some of the main-stream WSN simulators highlighting their main characteristics to put in context our CML-WSN. For an extensive comparison of WSN simulators, the reader can refer to [3–6].

NS-2 [7] is a discrete-time simulator that can support multiple protocols for wired, wireless and satellite networks in all layers. This software contains modules that cover a broad group of applications, protocols, routing, transport, different types of links and routing strategies and mechanisms. However, the simulator is not easy to understand and operate, and it was not specifically designed for WSNs simulation.

OPNET (OPtimized Network Engineering Tool) [8,9] Modeler permits to model and simulate communication systems. It allows to design and study networks, devices, protocols, and applications, providing flexibility and scalability. In addition, it simulates diverse networks where it can involve a large number of protocols and specific variables that the user can modify and study. Although it offers a C++ simulation class library and GUI support, it is too slow for MAC protocol simulations and does not have a good wireless mobility model.

Among other network simulators, one of them is TOSSIM [10], which estimates the energy consumption while considering the batteries lifetime of the devices and set realistic scenarios with common platforms. TOSSIM [10] is a TinyOS based interrupt-level discrete event simulator. It simulates only sensor applications ported to the i386 architecture, so it is only compatible with TinyOS. It does not capture CPU time neither energy consumption.

Consequently, one of the best ways to calculate a more precise energy saving model for WSNs is having extensive knowledge of the variables that affect the energy consumption of sensors and in what proportion. In a network, there are many variables that directly or indirectly affect energy consumption, eg., retransmissions of packets, collisions, and so on. Besides, the use of the channel is a variable that directly affects the MAC layer algorithm since nodes have to wait a lot of time if the channel is continuously busy. Moreover, control packets used by protocols significantly increase the amount of packets flowing in the network which causes collisions and, although nodes in WSN are designed to handle low processing, storing large routing tables usually translates into the sensors spending more energy in the development and maintenance. Another important factor is the establishment of links among nodes, which is proposed by a routing protocol. If the links are few, is more likely that some fail and cause others to become bottlenecks for traffic [11].

This overview enables us to propose a network simulator based on an event-driven system where we have an approach to the Physical, MAC and Network layers. The proposed CML-WSN simulator allows the implementation of any

topology, several routing protocols, observe and count collisions and retransmissions and establish a model to evaluate energy consumption power techniques.

3 Simulator Overview

In this section, we give a brief overview of our CML-WSN simulator architecture. Our proposed tool is a discrete event simulator implemented in C++ for WSNs. The CML-WSN simulator allows users to create network topologies, configure devices, inject packets and change network settings.

Figure 1 shows the general overview of the CML-WSN simulator. It consists of three blocks: inputs, the discrete event simulation of the WSN through a scheduler, and outputs. Despite the simplicity of the general structure, the simulator provides the ability to debug, test and analyze algorithms in a controlled environment for networking research. Some of the features of the CML-WSN simulator are:

- Viewing code: Window displays the machine code at runtime.
- Debug On: Ability to enable or disable the debugger.
- Step by step execution mode.
- Run/Stop program execution.
- Profile: Displays usage statistics system.
- Traffic monitoring capabilities.
- Energy model required for adequate analysis of the entire platform.
- Physical Layer: 802.15.4 physical layer is split into two sub-layers: PHY data service and PHY management which are responsible for transmitting and receiving messages through the physical environment.
- MAC Layer: Here CSMA/CA algorithm and detection carrier were implemented.
- Network Layer: It is composed for the routing protocol.

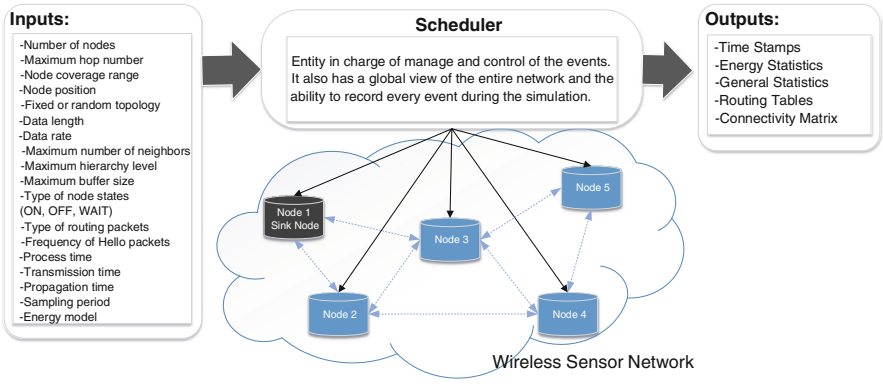

Fig. 1. General overview of CML-WSN simulator.

Besides, the core of the network simulator is composed of two main objects: the node and the scheduler. In the next subsections, we explain the structures of these two objects in the CML-WSN simulator.

3.1 Node Structure

The Node is an autonomous entity (object) that has properties and functions such as transmit, receive, turn on and off, listen, switch and so on. In the CML-WSN simulator, the node is an independent entity that performs many tasks during the simulation period. Figure 2 shows the node behavior during a simulation. Nodes are turned on and off, consume energy, listen to the channel, send and receive packets, have input and output packet queues and manage routing tables. All these tasks are performed by functions that accept input parameters and have output variables.

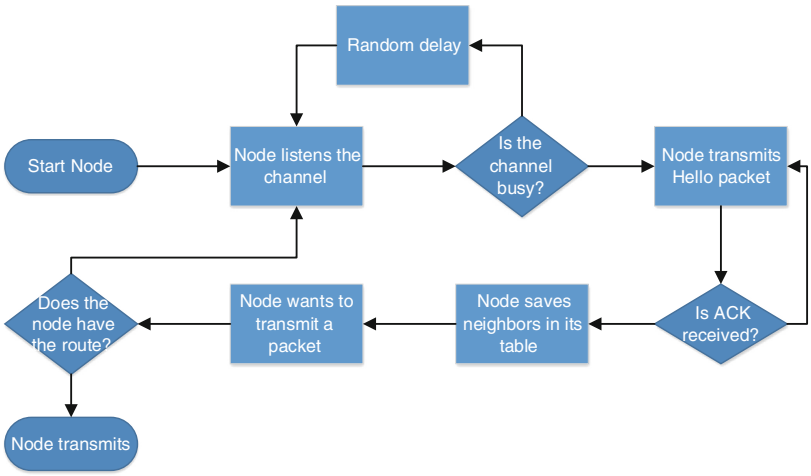

Fig. 2. Basic structure of a node in the CML-WSN simulator.

All the node tasks and interactions are organized and supervised by an external entity called network scheduler. The scheduler is described in next subsection.

3.2 Scheduler Structure

The scheduler is the entity responsible for control and management of all events during simulations. Moreover, it has a global vision of the entire network. Figure 3 shows the scheduler events and actions. The scheduler organizes the events as a queue and performs actions by times. The scheduler is responsible for handling these events regardless of category. Each event is distributed and carried to the node. Then, the node processes the event depending on the type

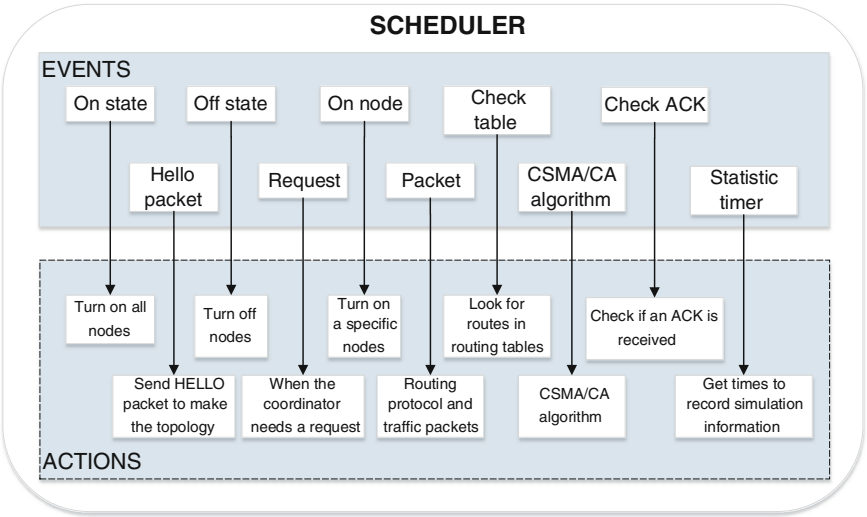

Fig. 3. Scheduler main functions in the CML-WSN simulator.

(e.g., if the event type is packet, the node opens it, processes it and sends or receives it). Thus, nodes are related to each other through the exchange of events that are controlled by the scheduler.

3.3 Input and Output Variables

A simulator is a program that performs experiments with a model mimicking the operation of a complex system. This process involves several stages such as: system definition, model formulation, data collection, implementation, validation, experimentation, and documentation.

Due to these stages, there are variables that in and out of the system. Left side of Fig. 1 lists some of the input parameters used in our simulator. The inputs can be read from a .txt file. Notice that the input parameter list includes different data allowing the user to modify the conditions of the simulation. Right side of Fig. 1 shows the outputs after simulation ends. The output consists in 5 .txt files with general information of the simulation process, and also with useful stats for analysis.

The first output file contains the data about individual per node and network. Also, it provides the stats for global consumption of each kind of energy used on the main tasks of the nodes. The second file includes the number of delivered packets, received packets, types of packets, packets loss due to a collision or interference in the channel and much more by each node and the whole network. The third file has the routing tables used by each node according to the rules of the selected routing protocol. The fourth file consists of the connection matrix between the nodes which will help verify the information about the neighbors of each node in the network. The fifth and last file includes the

general resulting parameters from the processes of each node and the time data when the operations were executed.

The resulting collection of stats provides insights into the working procedure of each routing protocol such as the redundancy in the tables in case a path is inaccessible. This output from the simulator is a set of .txt files formatted to be easy to read by the user. The main reason to select this type of file was to allow a simple interaction with the data and to be able to create graphics with ease. Moreover, the data collected from the files allows the gathering of routing metrics such as the number of attempts to listen the channel, retransmissions of packets, delay times, overhead, special packets from the routing protocol, among others.

4 Framework and Functionality

One of the most important features in WSNs is the ability of a node to process and manage the network traffic and make decisions about processing received data.

Likewise, a node can make decisions as a function of collected data, and if they have low relevance, the node will not generate unnecessary traffic thus saving battery and avoiding possible congestion on the network. These decisions are made based on a routing protocol and can be improved or adapted to the network conditions through a network simulator that optimizes certain parameters before the implementation. Current available simulators were not suitable for high investigation level because of the lack of flexibility for modifications or impossibility to incorporate new protocols.

Therefore, we designed and implemented a network simulator based on events using the C++ language. The simulator was conceived with the paradigm of "object-oriented programming", where nodes are autonomous entities (objects) that have properties and functions. Simulation events are managed by a planner (i.e., the scheduler) who serves as "tasks organize" for objects involved in the simulation. Some advantages of having made the simulator in C++ was the speed and the capability of managing various classes as separate entities.

4.1 Physical Layer

One of the main problems about ad-hoc systems is there is no infrastructure, so the routes change dynamically. This is due to fading, interference, disconnections, obstacles, node movements, etc. In consequence, issues such as quality of services, mobile battery-perishable, security, reliability of routes appear and cause packet loss. These factors (e.g., noise, fading, shading and modulation signal) generate interference. As a basic Physical layer, we propose a percentage of packet loss per link based on the conditions of the environment, where the channel is not ideal.

4.2 MAC Layer

MAC layer is designed under the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) mechanism [12], established by IEEE 802.15.4/Zigbee standard which takes into account a channel evaluation algorithm to prevent collisions, called clear channel assessment (CCA). Moreover, at this layer we can graduate the radio coverage for nodes and thus, variation of the transmission and receiving power.

4.3 Network Layer

In communication networks, there are routing protocols classified into two groups: proactive routing protocols and reactive routing protocols [13,14]. When nodes are under a reactive protocol they ask for a route only when it is needed. This involves high latency for the first packet and some independence among routes. The implementation of this layer also enables analysis of node spatial distribution in a uniform and non-uniform way and analysis of traffic load distribution of balancing links.

4.4 Functions in the Simulator

Figure 4 present a scheme showing relationships among the main functions of the simulator. First of all, the connectivity matrix of nodes describes nodes that are connected. Therefore, traffic packets begin. This is where each node receives a packet, opens it, processes it and takes a decision. This decision can be found in its routing table and send the packet or simply forward it. Meanwhile, nodes are consuming energy, and we can see statistics when needed.

The *startUp* function turns on generically all nodes or one that has been turned off. It can turn on them in a uniformly random way or in a particular distribution. The *channelMonitor* function gets the input time variable in which the node will listen to the channel to see if a delay is needed or not. The *CSMACA algorithm* function execute the CSMA/CA algorithm, with respect to the decision-making loop. The *eventSignal* function classifies the type of events to be processed in the network. This function has subfunctions processing to redirect each of the events. It gives flexibility and organization. The *Generator* function generate packets with respect to the type of packet needed by the node. The *receivePacket* is part of three separate functions which together receive and process a packet at a node: receiving, opening and processing. This function determines the time when the packet arrives and gets the source from which came the packet. The *openPacket* function takes the packet time and it opens the packet in the current node to establish what the source and destination of the packet are. The *process* function processes the packet according to the type. There are a number of functions associated with processing of every single type of packets on the network. The *updateTable* function updates the routing tables of the nodes. The function fills the fields to know all the possible routes to the

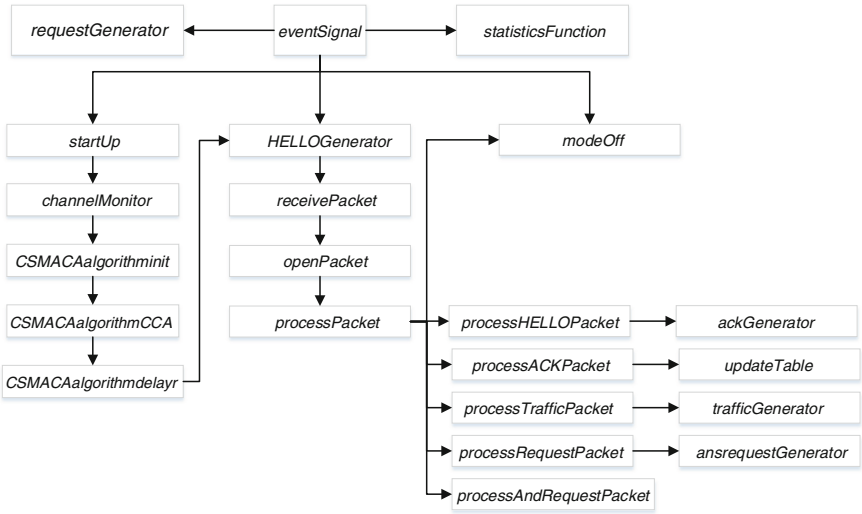

Fig. 4. Main functions in the CML-WSN simulator.

destination that were learned. Here there are parameters for each routing protocol such as flags for valid routes, complete routes or quality of each path. The *modeOff* function allows a node turns off and it cannot send or receive packets and immediately its routing table is cleaned. Finally, *statisticsFunction* function can be called at any time of simulation time. Here there are the statistics of network processes. It can have local statistics (by node) or global statistics (over the entire network). Packet statistics, collisions, energy pattern, and so on are obtained.

5 Results and Discussion

A simple WSN of 5 nodes was implemented to demonstrate how the network simulator operates. The physical modeling of the channel was done obtaining a certain percentage of the packets loss on each link. In the beginning, each node is turned on randomly. Each node starts discovering the network by sending HELLO packets. Then, when a node listens to a HELLO packet, it replies with an ACK packet. When a node receives an ACK packet, it records the node that sent it into the neighbors table which causes the establishment of a link and a possible active route. Thus, when the exchange of HELLO and ACK packets ends, the nodes can start sending regular traffic packets to the required destinations.

For this particular example, the network was designed to follow the hierarchical routing protocol [15] where there is a sink node or base station which becomes the unique destination for all the nodes in the network. Then, the nodes establish parent-child relationships according to the rules of the routing protocol

Table 1. Node output statistics.

	Traffic	Overhead	Retransmission	Energy (J)	Neighbors
Node 1	100	40	30	0.137689	[2, 3, 4]
Node 2	100	49	39	0.270185	[1, 3, 4, 5]
Node 3	100	48	38	0.213067	[1, 2, 4, 5]
Node 4	100	47	37	0.213059	[1, 2, 3, 5]
Node 5	100	37	27	0.200958	[2, 3, 4]

and hierarchies are formed between the nodes which are saved into the neighbor table, so each node knows the relationship that it has with each neighbor. After a while, the network operates long enough to have exchanged all kind of packets (i.e., overhead and traffic). Meanwhile, since each node is on it starts consuming energy for every activity that it performs individually as a part of the network. Which means that each node spends energy in turning on and off, switching, listening to the channel using the algorithm CSMA/CA, transmitting and receiving packets and using the microprocessor.

After completing the simulation, the program outputs a series of files which contain relevant statistics about the process. Table 1 present the Node information at the end of the simulation. It consists of information of traffic packets, overhead (i.e., data packets used by protocols), retransmission and energy generated by each node. Also, the neighbors of nodes are provided as information of the connectivity of the network.

The general stats are presented in Table 2. It includes a global information of the packets (e.g., traffic, overhead, packet loss, retransmission) and also global information about the energy utilized for the network.

For more details on the application of the CML-WSN simulator, the reader can be referred to [16], where the CML-WSN was compared to a real 100-Nodes WSN, with satisfactory results.

Table 2. General output statistics

Packets		Energy (J)	
Traffic	500	Start	0.000001
Overhead	221	MCU	0.28125
Lost by collision	114	Switching	0.002921
Lost by interference	64	Transmission	0.6022275
Retransmission	171	CSMA/CA	0.008829
Dropped	6	Shutdown	0.000007
		Receiving	0.139677
		Total	1.03496

6 Conclusions and Future Work

The Internet of Things comes from the evolution of telecommunications and information technologies. Therefore, to push its potential it is important to improve and understand better the infrastructure that makes it possible. A fundamental part of it are the WSNs, which allow the communication of sensors to gather intel of their surroundings. They collaborate as a team not only to collect data but to process it as information. To comprehend and design better networks, different simulators have been created to date (e.g., NS-2 [7], TOSSIM [10] and OPNET [8]).

Unfortunately, these simulators usually have a restricted number of models for each layer, and in many cases they are not realistic models, i.e. the physical layer using free space model which is certainly a good approach for some basic analysis but it may differ from real data collected from deployed networks. Besides, such simulators do not allow the user to add functionality to them. These constraints have as a consequence that the information and statistics only focus on particular areas which make it harder to acquire a global overview of the results.

For that reason, a network simulator, called Configurable Multi-Layer WSN (CML-WSN) simulator, was designed and developed to be able to implement different routing protocols, propose improvements, analyze performance parameters and apply optimized parameters related to WSNs. The CML-WSN simulator is capable of working on the Physical, MAC and Network layers, which allows the user to collect more information about each node and a comprehensive overview of the network. The simulator outputs a set of .txt files which enables the user to read quickly the results and to export it to other software to generate charts. The next step is to add a user-centered designed Graphic User Interface to the simulator that improves the interaction during the setup and results. Furthermore, adding the possibility of designing the network graphically and obtaining the information either raw or as a set of predefined charts. Lastly, aiding the user to make smarter decisions on how to build a WSN by understanding how the design and each sensor affects it.

References

1. Luo, H., Liu, Y., Das, S.K.: Routing correlated data in wireless sensor networks: a survey. IEEE Netw. **21**(6), 40–47 (2007)
2. Kulkarni, N., Prasad, R., H.C.N.G.: Performance evaluation of AODV, DSDV & DSR for quasi random deployment of sensor nodes in wireless sensor networks. In: International Conference on Devices and Communications (2011)
3. Korkalainen, M., Sallinen, M., Krkkinen, N., Tukeva, P.: Survey of wireless sensor networks simulation tools for demanding applications. In: International Conference on Networking and Services, pp. 102–106, April 2009
4. Yu, F., Jain, R.: A survey of wireless sensor network simulation tools. Washington University in St. Louis, Department of Science and Engineering (2011)
5. Musznicki, B., Zwierzykowski, P.: Survey of simulators for wireless sensor networks. Int. J. Grid Distrib. Comput. **5**(3), 23–50 (2012)

6. Nayyar, A., Singh, R.: A comprehensive review of simulation tools for wireless sensor networks (WSNs). J. Wirel. Netw. Commun. **5**(1), 19–47 (2015)
7. McCanne, S., Floyd, S.: The LBNL network simulator (1997). http://www.isi.edu/nsnam
8. Korkalainen, M., Sallinen, M.: A survey of RF-propagation simulation tools for wireless sensor networks. In: International Conference on Sensor Technologies and Applications, pp. 342–347, July 2010
9. Cavin, D., Sasson, Y., Schiper, A.: On the accuracy of MANET simulators. In: ACM International Workshop on Principles of Mobile Computing, pp. 38–43, October 2002
10. Mora-Merchan, J., Larios, D., Barbancho, J., Molina, F., Sevillano, J., Leon, C.: mTOSSIM: a simulator that estimates battery lifetime in wireless sensor networks. Simul. Model. Pract. Theory **31**, 39–51 (2013)
11. Shakshukia, E., Malikb, H., Sheltamic, T.: A comparative study on simulation vs. real time deployment in wireless sensor networks. J. Syst. Softw. **84**, 45–54 (2011)
12. LAN/MAN Standards Committee: Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs). IEEE Computer Society (2006)
13. The IETF website (2003). http://www.ietf.org/rfc/rfc3561.txt
14. The IETF website (2007). http://www.ietf.org/rfc/rfc4728.txt
15. Del-Valle-Soto, C., Mex-Perera, C., Olmedo, O., Orozco-Lugo, A., Galván-Tejada, G., Lara, M.: An efficient multi-parent hierarchical routing protocol for WSNs. In: Wireless Telecommunications Symposium (WTS), 1–8 April 2014 (2014)
16. Del-Valle-Soto, C., Mex-Perera, C., Olmedo, O., Orozco-Lugo, A., Galván-Tejada, G., Lara, M.: On the MAC/Network/Energy performance evaluation of wireless sensor networks: contrasting MPH, AODV, DSR and ZTR routing protocols. Sensors J. **14**, 22811–22847 (2014)