We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900
Open access books available

185,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



Chapter 12

Simulation Framework of Wireless Sensor Network (WSN) Using MATLAB/SIMULINK Software

Qutaiba I. Ali

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/46467

1. Introduction

A wireless sensor network consists of spatially distributed autonomous sensors to cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants. The development of wireless sensor networks was motivated by military applications such as battlefield surveillance. They are now used in many industrial and civilian application areas, including industrial process monitoring and control, machine health monitoring, environment and habitat monitoring, healthcare applications, home automation, and traffic control [1-2].

A smart sensor node is a combination of sensing, processing and communication technologies. Figure 1 shows the basic architectural components of a sensor node. The sensing unit senses the change of parameters, signal conditioning circuitry prepares the electrical signals to convert to the digital domain, the sensed analog signal is converted and is used as the input to the application algorithms or processing unit, the memory helps processing of tasks and the transceiver is used for communicating with other sensors or the base stations or sinks in WSN[3], see figure 1.

Sensors can monitor temperature, pressure, humidity, soil makeup, vehicular movement, noise levels, lighting conditions, the presence or absence of certain kinds of objects or substances, mechanical stress levels on attached objects, and other properties. Their mechanism may be seismic, magnetic, thermal, visual, infrared, acoustic, or radar. A smart sensor is also capable of self-identification and self-diagnosis. The mechanisms of smart sensors work in one of three ways: by a line of sight to the target (such as visual sensors), by proximity to target (such as seismic sensors), and by propagation like a wave with possible bending (such as acoustic sensors)[4,5].

INTECH
open science | open minds

© 2012 Ali, licensee InTech. This is an open access chapter distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

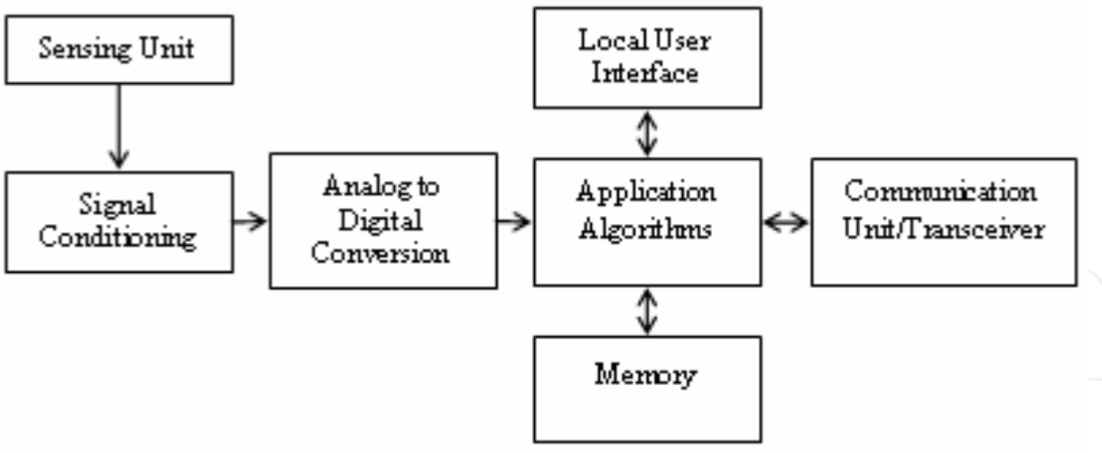


Figure 1. Basic architectural components of a smart sensor

2. Review of existing simulation environments for WSNs

In this section, a selection of existing simulation environments for WSNs is discussed. Basically, the investigated simulation environments can be divided into two major types: adaptive development and new development. The adaptive development covers simulation environments that already existed before the idea of WSNs emerged. These simulation environments were then extended to support wireless functionality and were then adapted for the use with WSNs. In contrast, new developments cover new simulators, which were created solely for simulating WSNs, considering sensor specific characteristics from the beginning. Both types have advantages and disadvantages, but basically it can be stated that while the evolutionary adaptation has some advantages in reusing well-tested ideas and source code as well as the bigger user and developer basis, the new developments have their advantages in focusing on the special characteristics and the functioning of sensor nodes.

a. GloMoSim/QualNet

GloMoSim [6] is a scalable simulation environment for wireless and wired network systems, which uses the parallel discrete-event simulation capability provided by Parsec [7], a c-based simulation language for sequential and parallel execution of discrete-event simulation models. Both, GloMoSim as well as Parsec, were developed by the Parallel Computing Lab. at UCLA. GloMoSim offers basic functionality to simulate wireless networks, even for ad hoc networks (e.g. AODV, DSR). However, the current version of GloMoSim does not offer any sensor network specific features in the default package so that without any further efforts no WSNs can be simulated meaningfully. In 2000 QualNet [6], [7], a commercial derivate of GloMoSim, was created and with GloMoSim 2.0, the last version of GloMoSim, was released under an academic license. From this point in time, no further improvements to GloMoSim were made, whereas the development of QualNet expedited. In October 2009, version 5.0 of QualNet was released including enhancements such as a new sensor network library for ZigBee, new network security library, parallel updates, new models (e.g. battery

and energy), updates to current models as well as performance improvements. Furthermore, a new QT based GUI was added providing a scenario designer, a visualizer to view network scenarios (2D and 3D), a packet tracer for debugging, an analyzer for statistics and a file editor to edit the scenarios directly.

b. OPNET Modeler Wireless Suite

OPNET Modeler Wireless Suite [8]–[10] is a commercial modeling and simulation tool for various types of wireless networks. It is developed by developed by OPNET Technologies, Inc. and based on the well-known product OPNET Modeler. The simulation environment uses a fast discrete event simulation engine operating with a 32-bit/ 64-bit fully parallel simulation kernel, which is available for Windows and Linux. The OPNET Modeler provides an object-oriented modeling approach and a hierarchical modeling environment. Although there are no special routing protocols for wireless sensor network available, at least different propagation and modulation techniques as well as a ZigBee (802.15.4) MAC layer are provided. Additional modules have to be customized or developed from the scratch. The simulations of wireless networks can be run as discrete event, hybrid or analytical, encompassing terrain, mobility and path-loss models. Due to the open interface external object files, libraries as well as other simulators can be integrated to the OPNET Modeler. Optional a System-in-the-Loop is available to interface simulations with live systems. Furthermore, the OPNET Modeler Wireless Suite provides grid computing support so that simulations can be executed in a distributed manner[11].

c. TOSSIM

TOSSIM (TinyOS mote simulator) [12]–[15] is a discrete event simulator for TinyOS sensor networks that is part of the official TinyOS package. TOSSIM takes advantage of the component based architecture of TinyOS by integrating it transparently by providing a new hardware resource abstraction layer that simulates the TinyOS network stack at the bit level for normal PCs. Due to this approach low-level protocols up to top-level applications can be simulated with TOSSIM. TOSSIM has an external communication system so that transmitted packets can be monitored and even new packets can be injected to network. Furthermore, the configuration of the debug options is fine grained providing the desired debug output at runtime. TOSSIM offers three network connectivity models: simple connectivity, static connectivity and space connectivity. The running simulations can be visualized and controlled by the Java-based GUI TinyViz[16].

d. OMNeT++

OMNeT++ [17]–[20] is an object-oriented discrete network simulation framework. The architecture is rather generic so that various problem domains can be simulated such as protocol modeling, validation of hardware architectures and modeling of wired and wireless communication networks. OMNeT++ is not a simulator, but it rather provides a framework and tools to write simulations. It is highly portable so that it can be run on the most common operating systems such as Windows, Linux and Mac OSX. There are a couple

of simulation frameworks that enable OMNeT++ to be used for wireless sensor networks[21]. The most common of these frameworks are discussed in the following subsections.

Mobility Framework: The Mobility Framework [22]–[24], developed in the Telecommunication Networks Group (TKN) at the Technical University of Berlin, provides only basic support for mobile and wireless networks. It includes some basic layers such as MAC layers (Aloha, CSMA) and network layers (flooding) as well as some basic mobility functionality and some basic application layer.

MiXiM: MiXiM [25], [26] is a merger of several OMNeT++ frameworks to support mobile and wireless simulations. It uses the mobility support, the connection management, and the general structure from the Mobility Framework (MF); the radio propagation models from the CHannel SIMulator (ChSim); and the protocol library from the MAC simulator, the Positif frame- work [27], and the Mobility Framework.

Castalia: Castalia [28], [29] is a simulator for WSNs (WSN), body area networks (BAN) and generally networks of low-power embedded devices that is based on OMNeT++. It is developed at the National ICT Australia since 2006 and made public as open source under the Academic Public License in 2007.

INET Framework: The INET Framework [30], [31] is a framework for OMNeT++ that contains various implementations of common protocols, such as IPv4, IPv6, TCP, UDP etc., as well as several application models. The INET Framework is not specialized on mobile and wireless networks, but has some support for it.

NesCT: NesCT [32] is not a real framework, but rather a translator from the programming language NesC to C++ classes for OMNet++.

e. NS-2

NS (the Network Simulator) [33], [34] is an object-oriented discrete event simulator targeting at networking research. NS-2 is written in C++ and OTcl, an object-oriented version of Tcl. A huge amount of contributed protocol source codes can be found on the website http://nsnam.isi.edu/nsnam/index.php/Contributed Code. among them there are also some for WSNs interesting wireless protocols such as different variations of 802.11, 802.16, IR-UWB, BlueTooth and 802.15.4. Despite the great number of contributing researchers the support for wireless sensor network specific protocols is rather low. As special wireless sensor network framework the Mannasim Framework [36] should be highlighted that provides sensor network specific protocols such as LEACH and Directed Diffusion. Also the extension NS2-MIUN [37] provides some wireless sensor network specific contributions with the focus on intrusion detection. SensorSim: SensorSim [38]–[40] is a simulation framework for modeling sensor networks that built up on NS-2. It provides additional features for modeling sensor networks such as sensor channel models, power models (battery and radio), lightweight protocol stacks for wireless micro-sensors, scenario generation and hybrid simulation.

f. Avrora

Avrora [41]–[43] is a set of simulation and analysis tools for programs written for AVR micro-controllers. It has support for different sensor platforms, such as Mica2 and MicaZ, allowing wireless network simulation, dynamic instrumentation and static analysis. Since 2004, Avrora is developed in a research project of the UCLA compiler group. The special characteristic of Avrora is that it operates on the instruction-level, i.e. actual microcontroller programs can be run in the simulator, instead of just simulating software models.

g. J-Sim

J-Sim [44]–[46] is a component-based compositional simulation environment based on the autonomous component architecture (ACA). The basic entities of ACA are components, which communicated with each other by sending and receiving data using their ports. Application specific models can be defined by sub-classing the specified classes of the WSN simulation framework and adapting them to the desired behavior. At the moment, 802.11 is used as MAC Layer and AODV is provided as routing protocol.

h. ATEMU

ATEMU [47], [48] is one of the first instruction-level software emulators for AVR based systems. Additionally peripheral devices of the MICA2 sensor node platform such as radio is supported. Although at the moment only the MICA2 hardware is supported, ATEMU can be easily extended to support other sensor node platforms. Although ATEMU is the most accurate instruction-level emulator for wireless sensor network research, it lacks from simulation speed, being 30 times slower than TOSSIM, for example.

i. EmStar

EmStar [49]–[51] is an environment for WSNs built from Linux-class devices, so called micro servers. In comparison to motes, micro servers are much less constrained in computational power and data storage size so that they can handle more complex tasks such as image and audio processing. EmStar consists of simulation and emulation tools, which utilize a modular, but not strictly layered, architecture. EmStar provides different simulation modes: a pure simulation mode, an emulation mode, a real mode and a hybrid mode. EmStar provides various services that are used and combined to provide network functionality for wireless embedded systems. This includes link drivers for the lowest-layer interfaces to network resources, pass-through modules that implement various types of filter and passive processing, and routing modules such as Flooding, DSR, Sink, StateSync and Centroute.

j. SENS

SENS [52], [53] is an application-oriented simulator for WSNs. It has a modular, layered architecture so that components for applications, network communication and the physical environment can be easily interchanged and extended. Due to different component

implementations, which varies in the degree of realism, application-specific environments can be created and simulated. Due to the chosen approach, SENS enables application portability because the same source code can be run with in a simulation or deployed on actual sensor nodes.

k. SENSE

SENSE (Sensor Network Simulator and Emulator) [54], [55] is a simulator for WSNs that is based on a novel component-oriented simulation methodology, which promotes extensibility and reusability. At the same time, the simulation efficiency and the scalability was considered. In the component repository of SENSE there are already different components available from the application to the physical layer including IEEE 802.11, AODV, DSR, SSR, SHR as well as Battery Models and a Power Model. At the moment, there does not seem to be any further tools included in SENSE so that, for example, a visualization tool to analyze the network behavior graphically is missing.

l. Shawn

Shawn [56]–[58] is customizable sensor network simulator based on an algorithmic approach. The primary design goals of Shawn are: simulate the effect caused by a phenomenon, scalability and support for extremely large networks and free choice of the implementation model. Models are the interfaces that are used by Shawn to control the simulation without knowing the exact implementation. Each implementation of a model specifies the actual behavior. Currently there are several implementations for the transmission model provided such as Pure CSMA & CSMA/CA, (Slotted) Aloha, Random Drop etc. Additionally to the three core models, there are several other models provided by Shawn for random variables, distance estimation and mobility. The simulation environment provides a sort of virtual world in which the different parts of the simulation are located. The simulated nodes are located in a single world instance and the nodes themselves are containers for processors. The application logic is implemented as instances of processors.

In this chapter, Simulink MATLAB was adopted to be the simulation tool of wireless sensor network (WSN). The main advantage of the suggested method is to determine the effect of the different channel parameters (i.e., Signal to Noise ratio, Attenuation and Interference) on the system behavior.

3. The proposed WSN simulation methodology

The environment in which we build our simulation model was MATLAB. The name MATLAB stands for matrix laboratory. *MATLAB*, developed by MathWorks Inc., is a software package for high performance numerical computation and visualization. The combination of analysis capabilities, flexibility, reliability, and powerful graphics makes *MATLAB* the premier software package for scientific researchers. *MATLAB* provides an interactive environment with hundreds of reliable and accurate built-in mathematical functions. These functions provide solutions to a broad range of mathematical problems

including matrix algebra, complex arithmetic, linear systems, differential equations, signal processing, optimization, nonlinear systems, and many other types of scientific computations. The most important feature of *MATLAB* is its programming capability, which is very easy to learn and to use, and which allows user-developed functions. It also allows access to Fortran algorithms and C codes by means of external interfaces. There are several optional toolboxes written for special applications such as signal processing, control systems design, system identification, statistics, neural networks, fuzzy logic, symbolic computations, and others. *MATLAB* has been enhanced by the very powerful Simulink program[59].

Simulink is a software package for modeling, simulating, and analyzing dynamical systems. It supports linear and nonlinear systems, modeled in continuous time, sampled time, or a hybrid of the two. Systems can also be multi-rate, i.e., have different parts that are sampled or updated at different rates. For modeling, Simulink provides a graphical user interface (GUI) for building models as block diagrams, using click-and-drag mouse operations. With this interface, you can draw the models just as you would with pencil and paper (or as most textbooks depict them). Simulink includes a comprehensive block library of sinks, sources, linear and nonlinear components, and connectors. You can also customize and create your own blocks Models are hierarchical. This approach provides insight into how a model is organized and how its parts interact. After you define a model, you can simulate it, using a choice of integration methods, either from the Simulink menus or by entering commands in MATLAB's command window. The menus are particularly convenient for interactive work, while the command-line approach is very useful for running a batch of simulations (for example, if you are doing Monte Carlo simulations or want to sweep a parameter across a range of values). Using scopes and other display blocks, you can see the simulation results while the simulation is running. In addition, you can change parameters and immediately see what happens, for "what if" exploration. The simulation results can be put in the MATLAB workspace for post processing and visualization. And because MATLAB and Simulink are integrated, you can simulate, analyze, and revise your models in either environment at any point. [59]**.**

3.1. Simulating a simple WSN in Simulink MATLAB

In order to demonstrate the concepts of the suggested simulation methodology, a simple WSN model was built as shown in figure 2[60]. This network consisted of three sensors (slaves) sending their measured data samples to a master node. In this chapter, MATLAB Simulink communication block set was used to build a complete WSN system. Simulation procedure includes building the hardware architecture of the transmitting nodes, modeling both the communication channel and the receiving master node architecture. Bluetooth was chosen to undertake the physical layer communication with respect to different channel parameters (i.e., Signal to Noise ratio, Attenuation and Interference). The simulation model was examined using different topologies under various conditions and numerous results were collected.

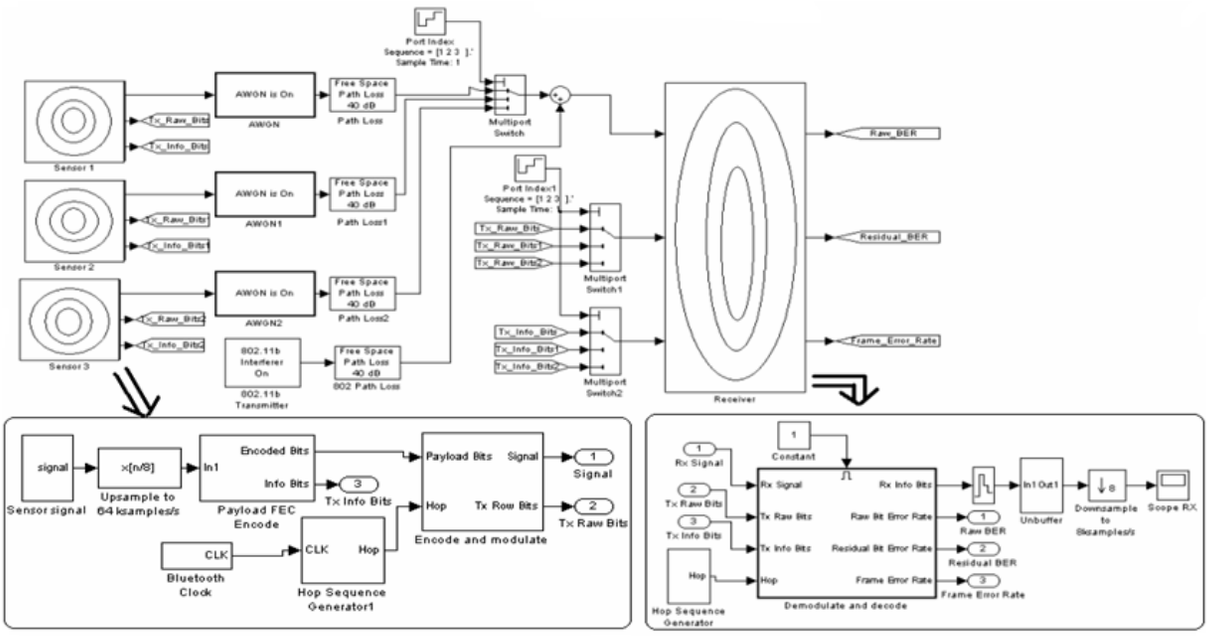


Figure 2. Simple WSN model

The architecture of the system could be explained as follows:

1. The transmitter

This system was based on Bluetooth technology that is considered as the backbone of transmission operation. Bluetooth is a short-range radio link technology that operates in the 2.4 GHz Industrial, Scientific, and Medical (ISM) band[60]. In this system we modulated the signal using Gaussian frequency shift keying (GFSK) over a radio channel with maximum capacity of 1 Mbps. The transmitter consists of the following blocks:

- Sensor signal stage: It is represented by a sensor to sense the physical signals such as temperature, pressure…etc, then transducing them into an electrical signal. In addition, this stage includes the A/D convertor which converts the signal from Analog to Digital using 256 quantization level.
- Up-sampling to 64ksamples/s: Up-samples the input to a higher rate by inserting zeros between samples.
- Payload FEC encode: Encodes the data to enable error correction(an FEC encoder may include a binary convolutional encoder followed by a puncturing device).
- Bluetooth Clock: Each Bluetooth device has a free-running 28-bit Bluetooth clock. The clock ticks 3,200 times per second or once every 312.5 μsec, representing a clock rate of 3.2 KHz.
- Hop Sequence Generator: For devices to communicate with each other, they must transmit and receive on the same frequency at the same time. The hop sequence generator generates a sequence of hop frequencies in the range 0 to 78. It can generate either the connection state hop sequence, a random white sequence, or be fixed.
- Encoder and modulator: The 366 data bits are transmitted at 1 Mbps and modulated using Gaussian frequency shift keying (GFSK). GFSK effectively transmits +150 kHz signal relative to the carrier for a 1bit, and a 150 kHz signal for a 0 bit. The carrier signal is

generated in the Simulink model by a baseband MFSK block set to 79 symbols and a separation of 1MHz. If a hop frequency value 0 is input, a -39MHz complex sinusoid is generated. If a 1 is entered, a -38 MHz complex sinusoid is generated and so on. In the model, the hop sequences are generated by a simple random number generator, not using the actual method specified in the standard. The transmitter is turned off after 366 bits using a Gain block to multiply the frame with a mask of 36600 ones and 26500 zeros.

2. The medium which consists of the following blocks

- AWGN Channel: The AWGN Channel block adds white Gaussian noise to a real or complex input signal. When the input signal is real, this block adds real Gaussian noise and produces a real output signal. When the input signal is complex, this block adds complex Gaussian noise and produces a complex output signal.
- Path Loss: This block reduces the amplitude of the input signal by an amount specified. The loss can be specified directly using the "Decibels" mode, or indirectly using the "Distance and Frequency" mode. The reciprocal of the loss is applied as a gain, e.g., a loss of +20 dB, which reduces the signal by a factor of 10 corresponds to a gain value of 0.1.
- 802.11b interferer: This block adds signals that have the same frequency of the data signal to make interference between the data signal and other signals(i.e. a Wireless Local Area Network (WLAN) transmission).
- Multiport Switch: In order to simulate the multiple access and multiplexing functions of the channel, this block was used. It chooses between a number of inputs. The first input is called the control input, while the rest of the inputs are called data inputs. The value of the control input determines which data input is passed through to the output port.

3. The receiver consists of the following blocks:

- Hop Sequence Generator: same as mentioned earlier.
- Demodulation and decoding: This block is used to extract the original information-bearing signal from a modulated carrier wave, and to recover the information contends in it.
- Zero-Order Hold: This block samples and holds its input for the specified sample period. The block accepts one input and generates one output, both of which can be scalar or vector. If the input is a vector, all elements of the vector are held for the same sample period.
- Un-buffer: This block un-buffers an Mi-by-N frame-based input into a 1-by-N sample-based output. That is, inputs are un-buffered row-wise so that each matrix row becomes an independent time-sample in the output. The rate at which the block receives inputs is generally less than the rate at which the block produces outputs.
- Down-sampling to 8ksamples/s: This block down-samples the input to a lower rate by deleting the repeating samples.
- Scope RX**:** It was used to display the received signal and compare it with the original signal to discover the system behavior.

As known, a piconet can includes up to seven slaves and one master. In this example three signals were sent from three sensors (slaves) to the receiving component (master) representing one piconet , the information obtained by the sensors are used to estimate the

Bluetooth performance as well as to study the media effect. Noise and interference are added to the signals in order to simulate the channel effect and measure Bit Error Rate (BER) and Frame Error Rate (FER). The following figures shows the system performance under different working conditions.

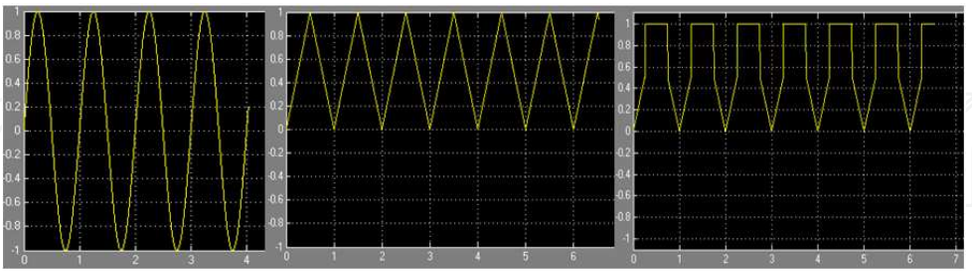
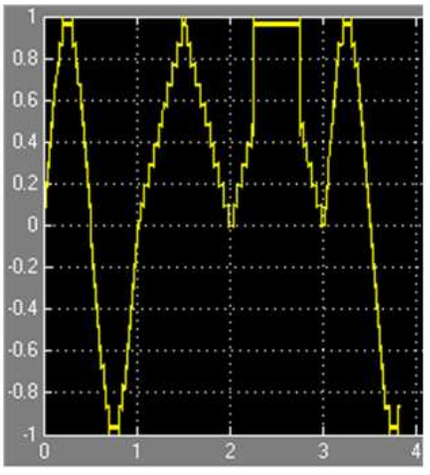
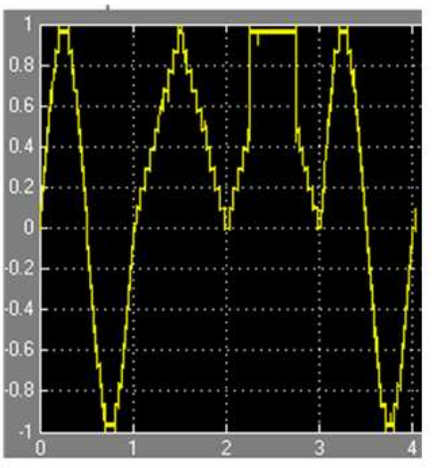


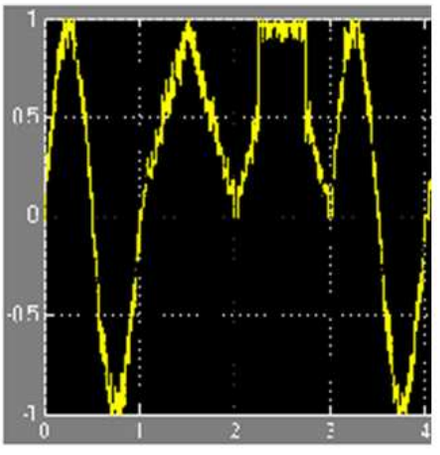
Figure 3. Signals sent from the three sensors



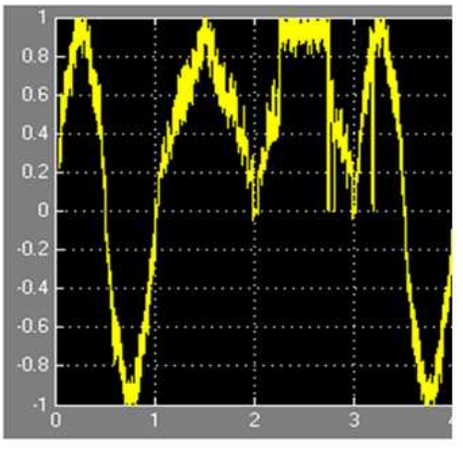
(a) SNR=20dB



(b) SNR=15dB

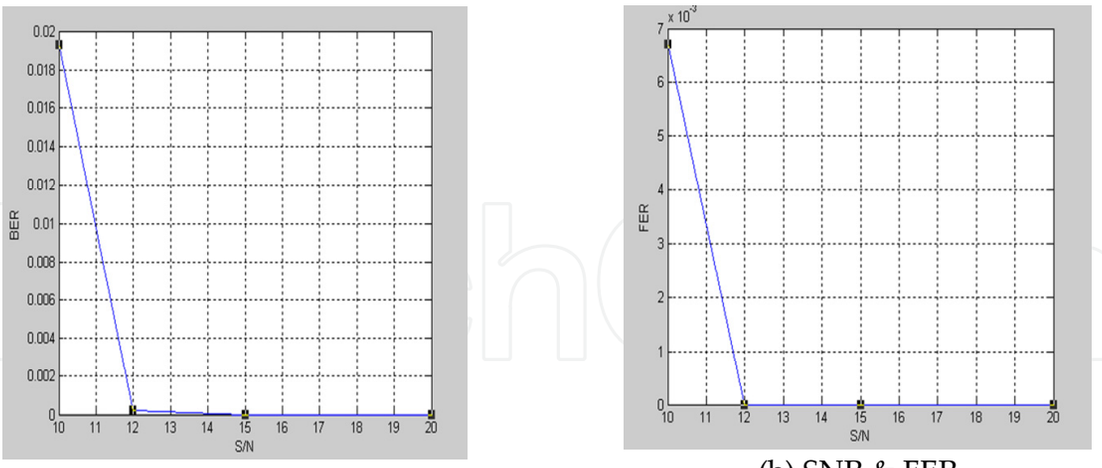


(c) SNR=12dB



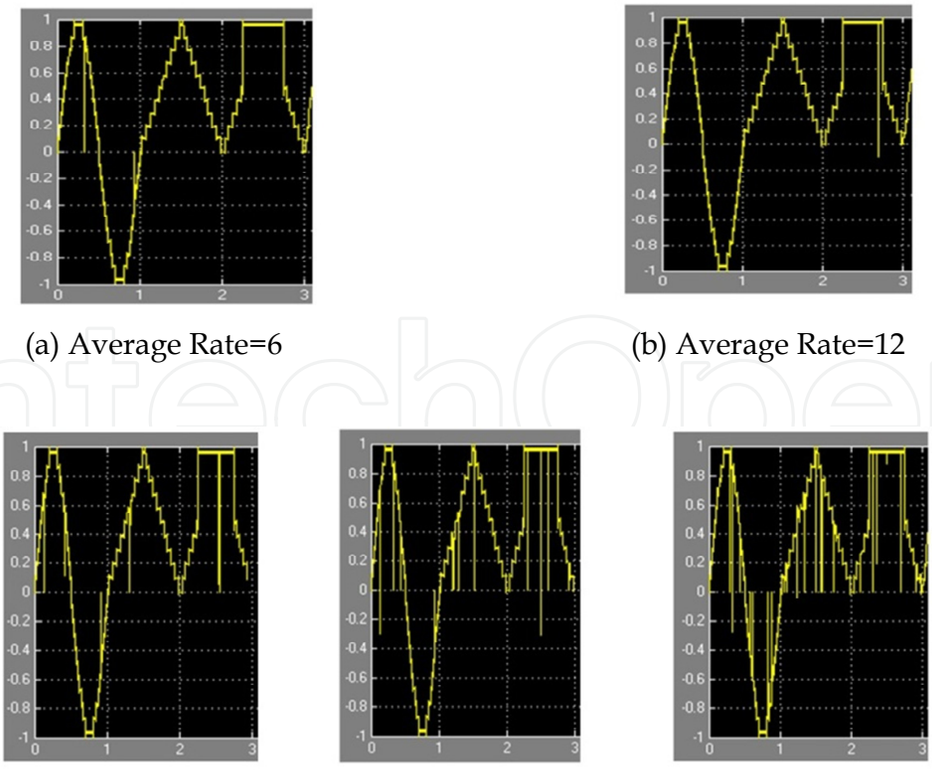
(d) SNR=10dB

Figure 4. Received signals with different SNR



(a) SNR & BER (b) SNR & FER

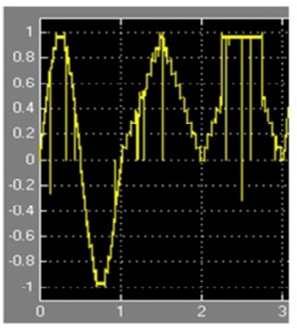
Figure 5. Relationship between SNR & (BER, FER)



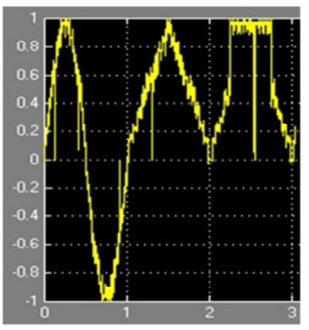
(a) Average Rate=6 (b) Average Rate=12

(c) Average Rate=25 (d) Average Rate=50 (e)Average Rate=100

Figure 6. Received signals with different rate of interference

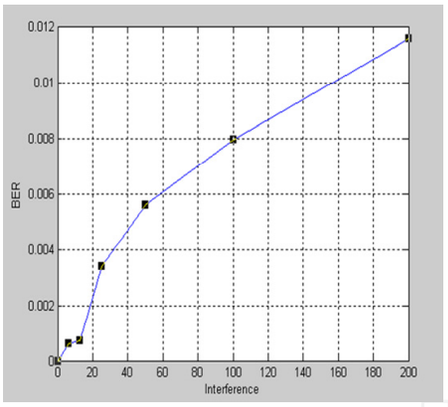


(a) SNR=15dB, Average Rate=6

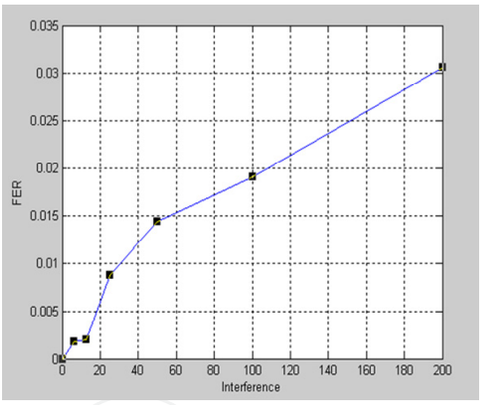


(b) SNR=12dB, Average Rate=25

Figure 7. Received signals with different rate of interference & different SNR



(a) Interference & BER



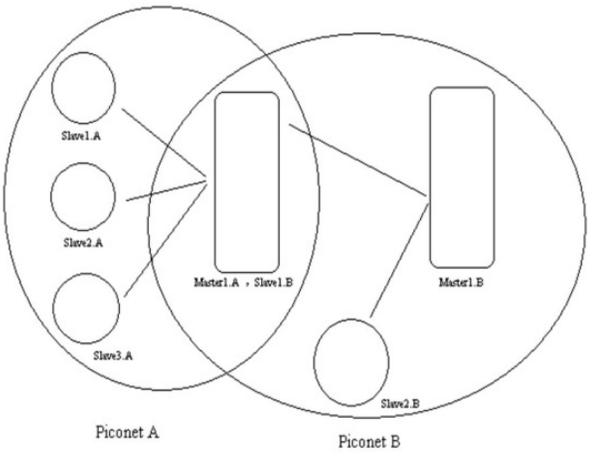
(b) Interference & FER

Figure 8. Relationship between interference & (BER, FER)

3.2. More complex example

As known from Blutooth operation, each piconet consists of one master and seven slaves and each master of a specific piconet may acts as a slave for another piconet which means the ability to expand the network to respond to more than seven sensors.

In this example two piconets are connected, so that the first piconet consists of three sensors connected to the master, and the later is connected as a slave to the second piconet. The second piconet consists of two slaves and one master as shown in Figure 9 below:



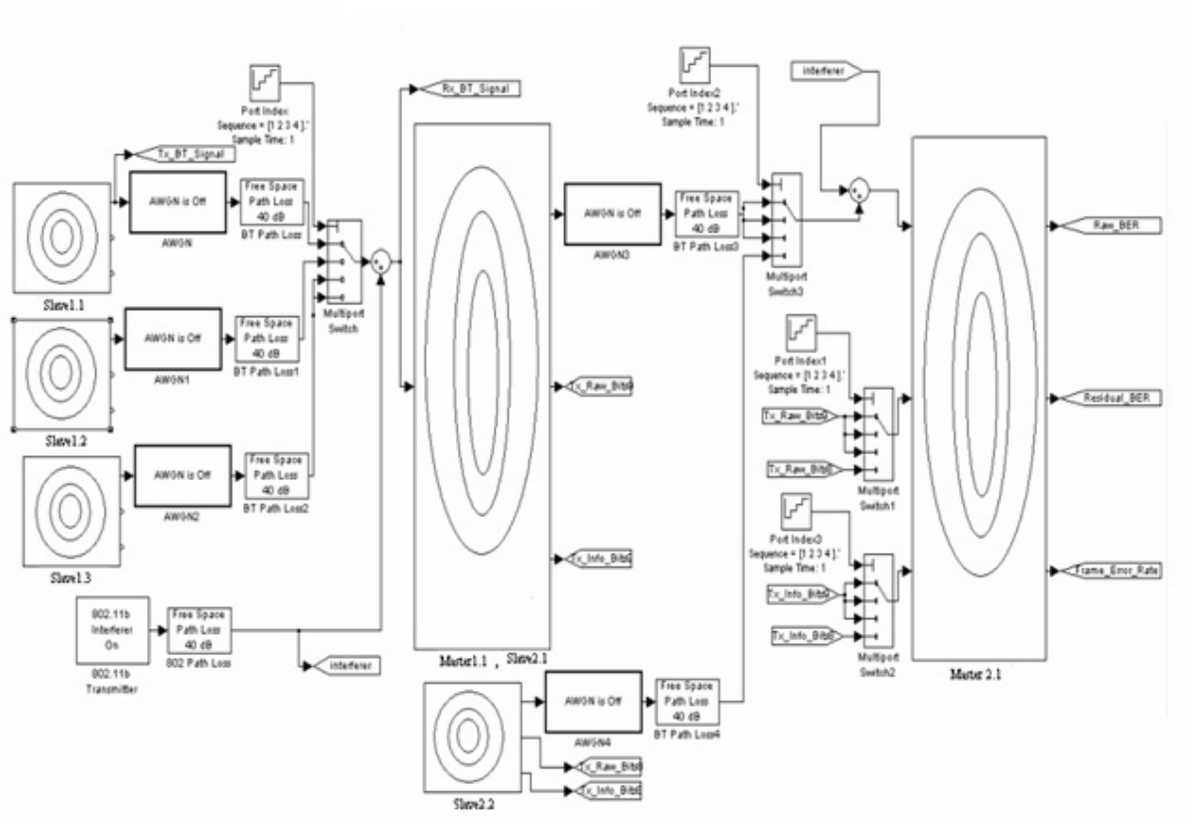
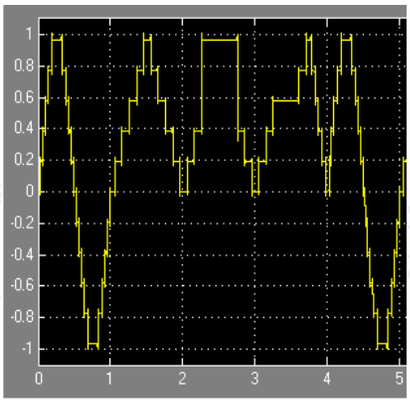
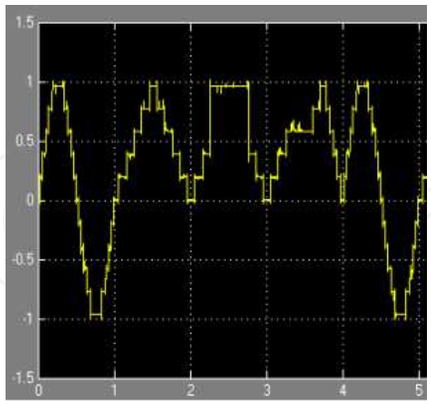


Figure 9. WSN model with masters & slaves

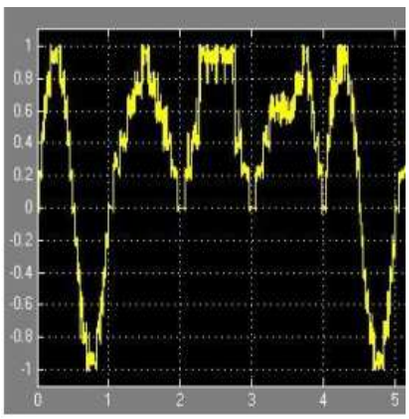
The following figures shows the system performance under different working conditions.



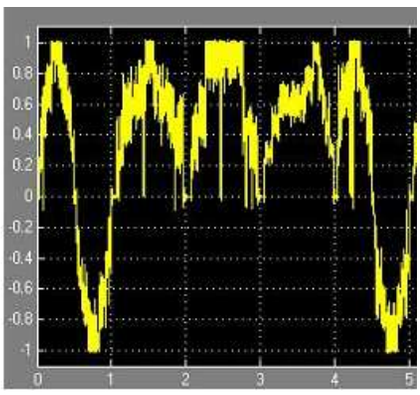
(a) SNR=20dB



(a) SNR=20dB

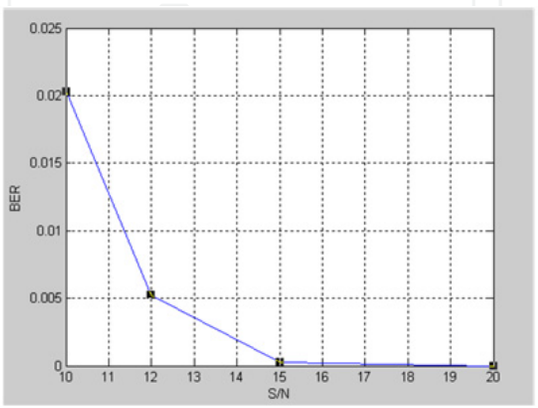


(a) SNR=20dB

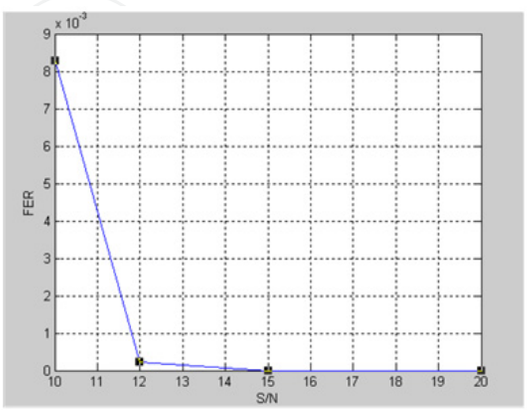


(a) SNR=20dB

Figure 10. Received signals with different Signal to Noise Ratio (SNR)

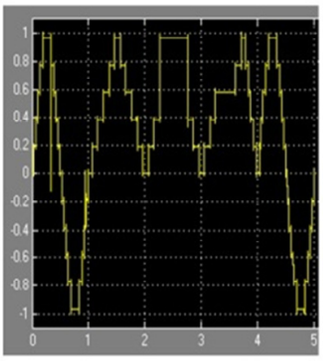


(a) SNR & BER

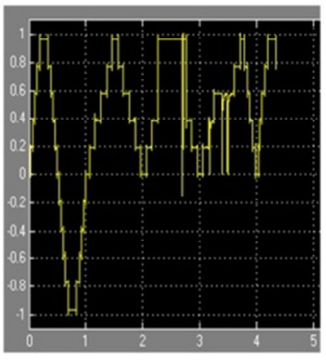


(b) SNR & FER

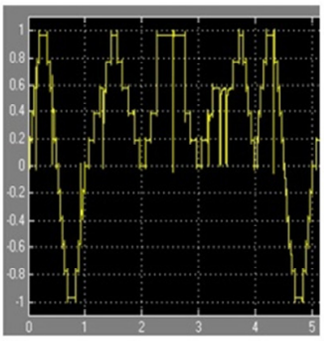
Figure 11. Relationship between SNR & (BER, FER)



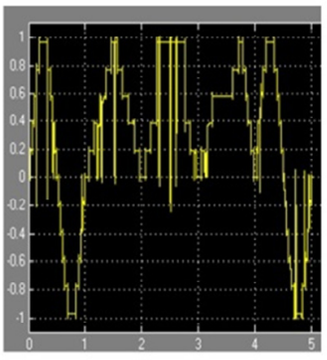
(a) Average Rate=6



(b) Average Rate=12

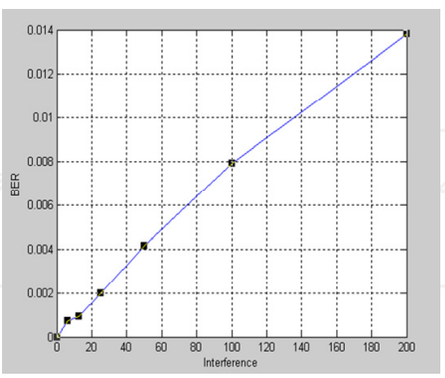


(c) Average Rate=25

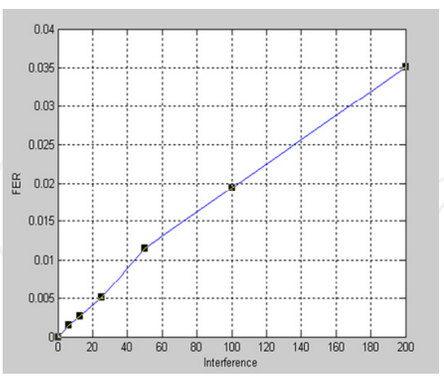


(d) Average Rate=50

Figure 12. Received signals with different rate of interference



(a) Interference & BER



(b) Interference & FER

Figure 13. Relationship between interference & (BER, FER)

From the above results, it is obvious that the behavior of the system was successfully described using the suggested simulation methodology. It is also important to mention that this simulation method provides the ability to change the different system parameters to create new environment and hence, new simulation scenarios. This new simulation

methodology proves the ability of the Simulink MATLAB to be a useful and flexible approach to study the effect of different physical layer parameters on the performance of wireless sensor networks.

Table (1) below, summarize the features of the different simulation methods of WSN including the suggested one in this chapter, MATLAB SIMULINK.

Simulation Environment	Programming Language	**WSN support**
GloMoSim/ QualNet	C and Parsec	GloMoSim: basic mobility and radio propagation models; 802.11; QualNet: additionally battery and energy model; ZigBee → GloMoSim seems to be outdated; QualNet seems to be more up-to-date, but commercial
OPNET Mod-eler Wireless Suite	configuration by GUI; internals C++	Different propagation models; 802.11, ZigBee; some MANET protocols, but no special WSN support → powerful tool with a nice GUI, but expensive
TOSSIM (part of TinyOS)	nesC	All TinyOS-based WSN protocols can be simulated with TOSSIM without modifications → good ap-proach especially if implementation should also be used with TinyOS-based nodes
OMNeT++	basic modules C++; larger structures NED	Several frameworks that add WSN functionality to OMNet++ such as MiXiM, Castalia, etc. → active project with a huge user base; Eclipse-based IDE for development
NS-2	C++; configu-ration OTcl	Huge amount of protocols available contributed by NS-2 users → complex configuration; unclear situa-tion due to large number of different user contributed implementations
Avrora	AVR micro-controller binaries	Particularly for programs written for AVR micro-controller with support for support for Mica2 and Mi-caZ → very special application area; project seems to be still active -still changes in CVS

J-Sim	Java; configuration Tcl/Java	Includes sensor network package containing models such as propagation, battery, radio model and sen-sor protocol stack including AODV and 802.11 → project seems to be abandoned
ATEMU	AVR micro-controller binaries	Complete emulation of the AVR instruction set with partial Mica2 support; TinyOS based code can be run → very special application area; slow simulation speed; project seems to be abandoned
EmStar	C	Provides network functionality for wireless embedded systems; EmTOS can be used to run TinyOS applications as EmStar module → project seems be abandoned (download links broken)
SENS	C++	Provides very basic network and physical layer sup-port. Source can be compiled for TinyOS. → project does not seem to be developed any further
SENSE	C++	Includes battery and power models, MAC layers (802.11) as well as network protocols (AODV, DSR, SSR, SHR) → does not seem to be developed any further
Shawn	C++	Algorithmic approach that concentrates on lower layers, no special WSN protocols → very active project -lot of recent changes in SVN
MATLAB SIMULINK	C , Java	Detailed simulation of the end nodes and their architecture, Physical layer parameters, different modulation & encoding techniques, communication channel modeling(SNR, effect of different Noise schemes, Interference, distance, etc..), various methods to monitor and record results, making use of the rich library of Matlab/Simulink.

Table 1. The features of the different simulation methods of WSN

4. Conclusions

In this chapter, a new simulation methodology of wireless sensor networks (WSN) was presented. MATLAB/Simulink was used as the tool to build the simulation environment. The strength of this simulation method falls in the ability to study the effect of different physical layer parameters (channel noise and interference, Signal to noise ratio…etc.) on the system behavior. The other advantage of this method is its flexibility in building the end nodes and sensors. This simulation methodology could be used to build different WSN types and opens the doors to use the MATLAB in this new field.

Author details

Qutaiba I. Ali
Mosul University, Computer Engineering Department, Iraq

Acknowledgement

Many thanks to my students Akram & Hussien for their assistance.

5. References

[1] Lewis, F "Wireless Sensors Networks, Smart Environments: Technologies, Protocols, and Applications', ed. Cook DJ, Das SK, John Wiley, New York, 2004; 1-18.
[2] Akyildiz, IF, Sankarasubramaniam, F, Cayirci, E, "A Survey on Sensor Networks", IEEE Commun Mag 2002; 102-114.
[3] Rabaey, J, Ammer, M, da Silva, J.L., D. Patel, and S. Roundy, "Picoradio supports ad hoc ultra-low power wireless networking," *Computer*, vol. 33, no. 7, pp. 42–48, July 2000.
[4] Gupta, G, Mukhopadhyay, SC, Sutherland, M., Demidenko, S., "Wireless Sensor Network for Selective Activity Monitoring in a home for the Elderly", Proceedings of 2007 IEEE IMTC conference. Poland, Warsaw 2007; 1(3): 1-6.
[5] Callaway, E., Gorday, P. , Hester, L., "Home Networking with IEEE 802.15.4: A Developing Standard for Low-Rate Wireless Personal Area Networks", IEEE Commun Mag 2002; 69-77.
[6] QualNet. [Online]. Available: http://www.scalable-networks.com/products/qualnet/,2012.
[7] S. Technologies, "Qualnet v. 3.9. 5 user's guide," 2006.
[8] OPNET Technologies, Inc. [Online]. Available: http://www.opnet.com/,2012.
[9] Jiang, H., Wang, P. ,Liu, H., "Research on OPNET simulation model in wireless sensor networks," Jisuanji Gongcheng/ Computer Engineering, vol. 33, no. 4, 2007.
[10] Jurˇc´ık, P., Koubˆaa,A., "The IEEE 802.15. 4 OPNET Simulation Model: Reference Guide v2. 0," 2007.

[11] OPNET Technologies, Inc. [Online]. Available: http://www.opnet.com/support/des model library/images/MANET scrnsht.jpg, 2012.
[12] Computer Science Division at UC Berkeley. [Online]. Available: http://www.cs.berkeley.edu/_pal/research/tossim.html, 2012
[13] Levis, P., Lee, N.,Welsh, M., Culler, M., "TOSSIM: Accurate and scalable simulation of entire TinyOS applications," in Proceedings of the 1st international conference on Embedded networked sensor systems. ACM New York, NY, USA, 2003, pp. 126–137.
[14] Levis, P., Lee, N., "Tossim: A simulator for tinyos networks," UC Berkeley, September, 2003.
[15] Notani, S., "Performance Simulation of Multihop Routing Algorithms for Ad-Hoc Wireless Sensor Networks Using TOSSIM," in Advanced Communication Technology, 2008. ICACT 2008. 10th International Conference on, vol. 1, 2008.
[16] Computer Science Division at UC Berkeley. Visualisation of a TOSSIM simulation with TinyViz. [Online]. Available: http://www.tinyos.net/tinyos-1.x/doc/tutorial/imgs/ tinyviz-screenshot1.gif, 2012.
[17] OMNeT++ Community. (2010, May) OMNeT++. [Online]. Available: http://www.omnetpp.org/
[18] Varga, A., et al., "The OMNeT++ discrete event simulation system," in Proceedings of the European Simulation Multiconference (ESM'2001), 2001, pp. 319–324.
[19] Varga, A., "OMNeT++ Discrete event simulation system. User Manual," Technical University of Budapest, Dept. of Telecommunications, 2006.
[20] Varga, A., Hornig, R., "An overview of the OMNeT++ simulation environment," in Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops table of contents. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering) ICST, Brussels, Belgium, Belgium, 2008.
[21] OMNeT++ Community. OMNeT++ 4.0 IDE.[Online]. Available: http://omnetpp.org/doc/omnetpp40/ide-overview/pictures/img1.png, 2012.
[22] Mobility Framework for OMNeT++ Community. [Online]. Available: http://mobility-fw.sourceforge.net, 2012.
[23] Drytkiewicz, W., Sroka, S., Handziski, V., Koepke, A., Karl, H., "A mobility framework for omnet++," in 3rd International OMNeT++ Workshop, 2003.
[24] L¨obbers, M., Willkomm, D., K¨opke, A., Karl, H., "Framework for Simulation of Mobility in OMNeT++(Mobility Framework)," 2004.
[25] MiXiM developers. MiXiM project. [Online]. Available: http://mixim.sourceforge.net/, 2012.
[26] K¨opke, A., Swigulski, M., Wessel, K., Willkomm, D., Haneveld, P., Parker, T., Visser, O., Lichte, H. Valentin, S., "Simulating wireless and mobile networks in OMNeT++ the MiXiM vision," in Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops table of contents. ICST (Institute for Computer Sciences, Social-

Informatics and Telecommunications Engineering) ICST, Brussels, Belgium, Belgium, 2008.

[27] University of Twente and TU Delft. Positif, MAC Simulator and T-MAC. [Online]. Available: http://www.consensus.tudelft.nl/software.html, 2012.

[28] National ICT Australia. Castalia. [Online]. Available: http://castalia.npc.nicta.com.au/, 2012.

[29] Boulis, A., "Castalia: revealing pitfalls in designing distributed algorithms in WSN," in Proceedings of the 5th international conference on Embedded networked sensor systems. ACM New York, NY, USA, 2007, pp. 407–408.

[30] OMNeT++ Community. INET framework for the OMNeT++. [Online]. Available: http://inet.omnetpp.org/, 2012.

[31] Ariza-Quintana, A., Casilari, E., Cabrera, A., "Implementation of MANET routing protocols on OMNeT++," in Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops table of contents. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering) ICST, Brussels, Belgium, Belgium, 2008.

[32] OMNeT++ Community. NesCT for the OMNeT++. [Online]. Available: http://nesct.sourceforge.net/, 2012.

[33] NS-2 developers. The Network Simulator – ns-2. [Online]. Available: http://www.isi.edu/nsnam/ns/, 2012.

[34] Downard I., DC, N., "Simulating sensor networks in ns-2,"2004.

[35] NS-2 developers. Visualisation of a ns-2 simulation with NAM. [Online]. Available: http://www.isi.edu/nsnam/nam/nambig.gif, 2012.

[36] Departamento de Ciˆencia da Computac¸ ˜ao, Universidade Federal deMinas Gerais. Mannasim Framework for ns-2. [Online]. Available: http://www.mannasim.dcc.ufmg.br/, 2012.

[37] Computer Science, Mid Sweden University , Sweden. NS2-MIUN. [Online]. Available: http://apachepersonal.miun.se/_qinwan/resources.htm, 2012.

[38] Networked and Embedded Systems Laboratory (NESL) at the University of California at Los Angeles (UCLA). SensorSim framework. [Online]. Available: http://nesl.ee.ucla.edu/projects/sensorsim/, 2012.

[39] Park, S., Savvides, A., Srivastava, M., "SensorSim: a simulation framework for sensor networks," in Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems. ACM New York, NY, USA, 2000, pp. 104–111.

[40] "Sensor Sim: A Simulation Framework for Networks Sensors", Electrical Engineering Department, University of California, Los Angeles, Retrieved October, vol. 16, 2006.

[41] UCLA Compilers Group). Avrora. [Online]. Available: http://compilers.cs.ucla.edu/avrora/, 2012.

[42] Titzer, B., "Avrora: The AVR simulation and analysis framework," Master's thesis, University of California, Los Angeles, 2004.
[43] Titzer, B., Lee, D. Palsberg, J., "Avrora: Scalable sensor network simulation with precise timing," in Proceedings of the 4th international symposium on Information processing in sensor networks. IEEE Press Piscataway, NJ, USA, 2005.
[44] Department of Computer Science at University of Illinois at Urbana-Champaign). J-Sim. [Online]. Available: http://sites.google.com/site/jsimofficial, 2012.
[45] Sobeih, A., Chen, W., Hou, J., Kung, L., "J-sim: A simulation environment for wireless sensor networks," in Proceedings of the 38th annual Symposium on Simulation. IEEE Computer Society Washington, DC, USA, 2005, pp. 175–187.
[46] Hou, J., Kung, L., "J-Sim: A Simulation and emulation environment for wireless sensor networks," IEEE Wireless Communications Magazine, vol. 13, no. 4, pp. 104–119, 2006.
[47] Center for Satellite and Hybrid Communication Networks (CSHCN) at University of Maryland. Atemu. [Online]. Available: http://www.cshcn.umd.edu/research/atemu/, 2012.
[48] Polley, J., Blazakis, D., "Atemu: A fine-grained sensor network simulator," in Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on, 2004, pp. 145–152.
[49] Laboratory for Embedded Collaborative Systems (LECS) at UCLA. [Online]. Available: http://www.lecs.cs.ucla.edu/emstar/, 2012.
[50] Elson, J., Bien, S., "Emstar: An environment for developing wireless embedded systems software," Center for Embedded Networked Sensing (CENS) Technical Report, vol. 9, 2003.
[51] Girod, L., Ramanathan, N.," A Software Environment for Developing and Deploying Heterogeneous Sensor Actuator Networks," Center for Embedded Network Sensing, p. 101, 2007.
[52] Open Systems Laboratory at University of Illinois at Urbana-Champaign. [Online]. Available: http://osl.cs.uiuc.edu/sens/, 2012.
[53] Sundresh, S., Kim, W., Agha, G., "SENS: A sensor, environment and network simulator," in Proceedings of the 37th annual symposium on Simulation. IEEE Computer Society Washington, DC, USA, 2004.
[54] Computer Science Department at Rensselaer Polytechnic Institute (RPI). [Online]. Available: http://www.ita.cs.rpi.edu/sense/, 2012.
[55] Chen, G., Branch, J., "Sense: A sensor network simulator," Advances in Pervasive Computing and Networking, pp. 249–267, 2004.
[56] SwarmNet project. [Online]. Available: http: //shawn.sourceforge.net/, 2012.
[57] Kroeller, A., Pfisterer, D., "Shawn: A new approach to simulating wireless sensor networks," Arxiv preprint cs/0502003, 2005.
[58] Fekete, S., Kroller, A., Fischer, S., Pfisterer, D., "Shawn: The fast, highly customizable sensor network simulator," in Networked Sensing Systems, 2007. INSS'07. Fourth International Conference on, 2007, pp.299–299.
[59] MATLAB Web Site: http://www.mathworks.com/

[60] Ali, Q., Abdulmaojod, A., Ahmed, H.," Simulation & Performance Study of Wireless Sensor Network (WSN) Using MATLAB, IJEEE Journal,2010.

IntechOpen

IntechOpen