Wireless Sensor Network Simulation Frameworks: A Tutorial Review

MATLAB/Simulink bests the rest.

By Madhupreetha L. Rajaram, Elias Kougianos, Saraju P. Mohanty, and Uma Choppali

A Wireless Sensor Network (WSN) is a distributed set of sensors deployed to work together for collective sensing and possible data processing. A WSN can be used to monitor environmental behavior and structural integrity in a variety of application fields, thus becoming an integral part of consumer electronics of smart buildings in smart cities. Due to ever increasing population growth with limited natural resources smart cities are expected to be the wave of the future. For instance, wireless sensor networks are widely used in industrial settings with machine monitoring and play an important role for monitoring the structural integrity of large buildings and bridges. This review paper focuses on existing WSN simulation frameworks that could be integrated with real-time hardware prototypes. Various such simulation frameworks are analyzed and compared, and a suitable simulation environment that supports specific software packages is determined.

I. Introduction

A typical sensor is used to sense environmental properties such as temperature, pressure, stress and vibration in the form of electrical signals which are then calibrated to measure the corresponding physical properties [1]. Wireless sensor networks (WSNs) are collections of such sensors deployed to sense variations in, and transmit data through, wireless networks as depicted in Figure 1 [2]. These sensors and sensor networks are an integral part of consumer electronics used for the development of smart cities, smart structures, smart transportation systems, and smart health care.

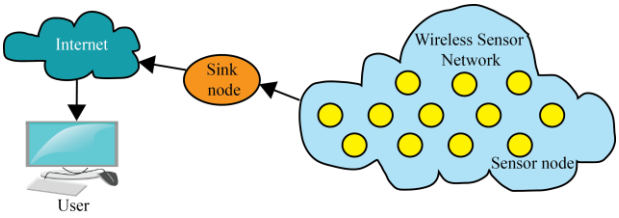


FIGURE 1. Illustration of a wireless sensor network (WSN).

A specific example of usage of a WSN for structural health monitoring is depicted in Figure 2. The collected data provide better understanding of the structural materials, capacity of the structure and can also be used to generate alert warnings during natural calamities. In order to obtain complete and accurate information, a large number of sensor nodes must be deployed in the areas of interest. During the process of sensing and transmitting data, issues such as power management, data collection, time synchronization, communication protocols used and congestion need to be addressed. Various algorithms are proposed by researchers to overcome these issues. These algorithms can be verified with a simulation modeling framework or an experimental setup.

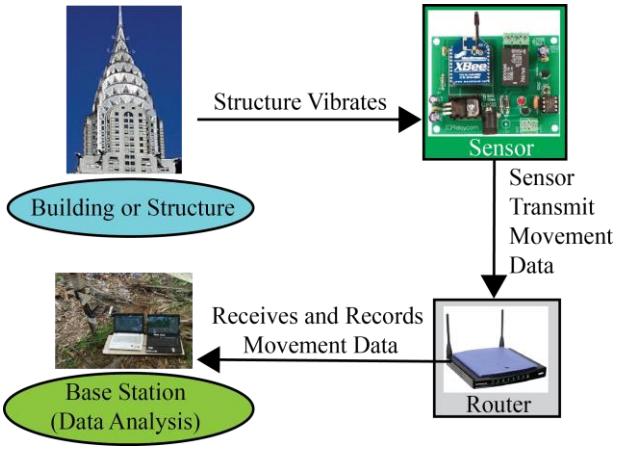


FIGURE 2. Illustration of a WSN for structural health monitoring.

One approach to test the design is through an experimental setup using actual components but it is expensive and rigid for explorative research. On the other hand, a simulation framework can reduce time and the risks associated with high cost. However, it is important to select a suitable simulation environment since there are numerous such platforms available for wireless sensor network (WSN) simulation. In this review paper, a comprehensive selection of WSN simulation frameworks is analyzed paying particular attention to those packages that can be interfaced with actual hardware.

This article is organized as follows. Section II reviews related works in WSN, and section III introduces various wireless sensor frameworks. Section IV and Section V provide analysis and comparison of these simulation frameworks. Finally, Section VI concludes with final remarks.

II. Wireless Sensor Networks: State-of-Art

The advancement of WSNs has led to more accurate monitoring of structural integrity, data collection and analysis of observed data. However, the data collection process is affected by various factors. Different solutions and algorithms have been proposed by researchers to improve the performance of the WSN.

A. Power Management

The sensor nodes are powered by a battery source, and the lifetime of the sensor node is determined by the energy stored in the battery. Hence, the effective use of the available power is a main challenge faced in sensor data collection. An algorithm for selecting the cluster heads for a group of sensors in order to reduce the power consumption is proposed in [3]. The algorithm was based on random head selection where minimum distance between the nodes was tabulated. Then the node with the minimum hopping distance was determined and assigned as the cluster head, which reduces the energy required to hop the data over long distances. An optimizing algorithm for limited buffering and controlled mobile sink is proposed in [4]. The mobile sink that collects the data by moving from one point to another increases the wait time, and therefore, can lead to buffer overflows. So, an optimization algorithm to reduce wait time is discussed, and the algorithm is implemented in the OMNeT++ simulator.

B. Data Collection

One of the most important operations of the sensor nodes is the data collection. Different data aggregation techniques have been proposed for efficient data collection. A complete information collection mechanism by deploying an agent in the WSN is proposed in [5]. This agent collaboration provides a means to coordinate with multiple sensor nodes to complete data collection, analysis and distributed fault diagnosis. Different agents are assigned different operations, and they coordinate with each other to complete the entire task, as shown in Figure 3. The model is simulated in a network simulator to verify operation.

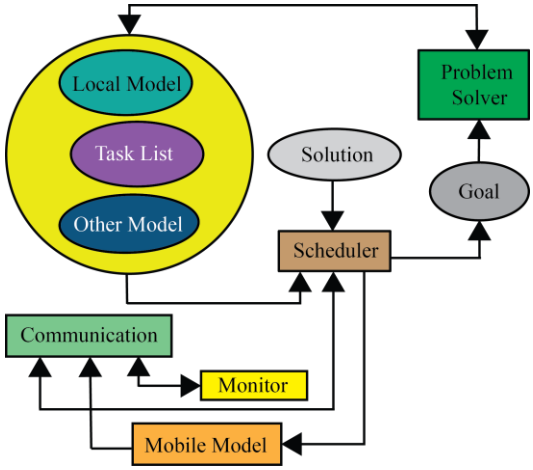


FiGURE 3. A Typical agent structure.

C. Communication Protocols

In order to obtain complete information of the area of interest, a large number of sensor nodes are deployed. When all these sensor nodes try to communicate with the base station or sink, data congestion can occur. A new congestion control mechanism is proposed in [6]. In this mechanism, the buffer in each node is adjusted based on downstream data transmission in order to minimize packet loss. The performance of the algorithm was verified by simulating the model in MATLAB®.

Data collected at sensor nodes needs to be transmitted to the base station or sink. This requires an efficient communication protocol. For this, a sleep scheduling algorithm is proposed in order to turn on and off the radios in [7]. The switching on and off of the radio is defined as a contiguous link scheduling problem and simulated in a C++ simulator. A hierarchical routing protocol, which is an optimized Low Energy Adaptive Clustering (LEACH) protocol, has been proposed in [8]. LEACH is a clustering-based protocol that aims at reducing the energy based on the assumption that all nodes have the same amount of energy. In actual practice all nodes do not consume the same amount of energy hence, in this algorithm an optimized LEACH is proposed. The algorithm allocates time slots for all the nodes. When the node does not transmit data during its time slot, then some other node utilizes the vacant slot for transmission. This reduces the wait time for other nodes, which results in efficient utilization of the available resource. The performance of the algorithm was tested in OMNeT++. A data gathering tree is constructed, and an energy efficient scheduling algorithm is proposed in [9]. The algorithm aims at reducing the state transitions, and hence the energy consumption. This algorithm uses a Time

division multiple access (TDMA) technique for scheduling. The activities of subsets of the sensors are divided into different groups, and successive time slots are scheduled. The entire network is divided into different groups, as shown in Figure 4. In the network, each group consists of a parent and children nodes. The children nodes will send data to the parent, and the parent will send information to the sink later within the allotted time slot. The model was simulated in NS2.34 to verify the performance.

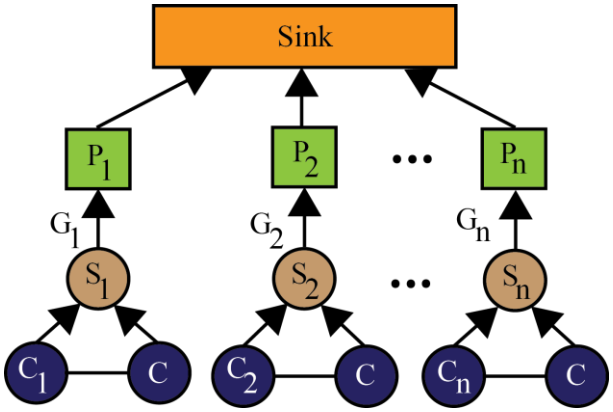
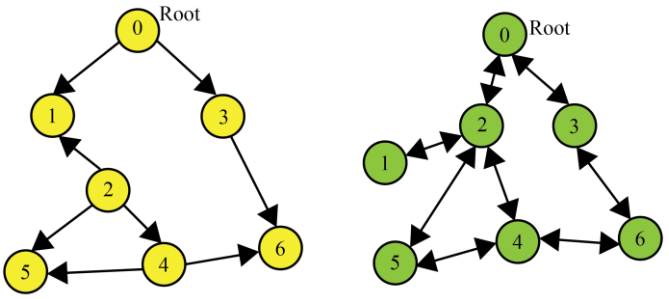


FIGURE 4. Illustration of a network model.

D. *Time Synchronization*

The data transmitted to the base station is stamped with time information to provide details on when the data was sensed. If each node operates at a different clock time, the time stamped with data transmission will not be the same for all nodes. Hence, it is important to synchronize the time for all nodes. In [10], an algorithm for time synchronization of neighboring nodes is proposed based on a Gradient Time Synchronization Protocol (GTSP), as shown in Figure 5, designed for accurate clock synchronization of neighboring nodes in a network simulator.



Time-Based Synchronization Protocol Gradient Synchronization Protocol

FIGURE 5. The Gradient Time Synchronization Protocol (GTSP).

III. Wireless Sensor NetworksSimulation: Selected Frameworks

The behavior of a system can be analyzed analytically, experimentally with a simulation mode, or a combination of these approaches. Analytical methods, however, cannot provide complete details on the impact of power consumption and other issues. On the other hand, experimental analysis can provide more accurate information but this is achieved at a higher cost. Simulation models serve as the best alternative to understand the behavior of a system at low cost and in short time. A simulation model can be designed based on different algorithms. It is important to determine an algorithm that best fits the requirements of the WSN.

A wide variety of simulation platforms are available, but only a few simulators might be applicable for certain operations. Various software tools are available such as NS-2, OMNeT++, PAWiS, GloMoSim/QualNet, OPNET, SENSE, J-Sim, Ptolemy II, Cell-DEVS, NesCT, GTnets, System C, Prowler, NCTUns2.0, Jist/SWANS, SSFNet, TOSSIM, Atarraya, Avrora, ATEMU, EmStar, SENS, Shawn, PiccSIM, TrueTime 2.0 in MATLAB®/Simulink® and native MATLAB®/Simulink® [11, 12]. Each one of them could be used for different kinds of applications.

IV. Details of Selected Simulation Frameworks

There are more than 25 WSN frameworks available. Out of all the simulation environments, a few frameworks were short listed and analyzed. The key features considered for the selection of a simulation platform were the following: co-simulation with MATLAB®, operating system support, programming language implementation and API, number of nodes that could be simultaneously simulated, documentation, latest version availability, ZigBee® support and potential issues during installation. Since the objective is focused on integrating the software model and hardware prototype, MATLAB® plays an important role in the design. So, the simulation environment must have capabilities to communicate with MATLAB®/Simulink® during real time operation.

A. *Network simulator NS-2*

Network Simulator (NS-2) is a discrete event, object oriented, and general purpose network simulator based on C++ language that can be used to simulate local and wide

area networks [11, 12, 13]. It was primarily developed to operate in Linux-based Operating Systema such as Ubuntu. However, it can also be installed in Windows® OS with Cygwin support. For analysis purposes, NS-2 was installed in Ubuntu. There are three commands that need to be typed in the terminal to install the latest version: (1) `sudo apt-get install ns2`; (2) `sudo apt-get install nam`; and (3) `sudo apt-get install xgraph`. When installation is complete, "`ns`" can be typed in the terminal, and a percentage symbol is returned, which confirms successful installation of NS-2. A network animator (NAM) editor window, as shown in Figure 6, can be opened by typing "`nam`" in the terminal. In order to design a model, a program can be written in the "Tool Command Language" (TCL) and visualized through the NAM editor. The performance is reliable for node sizes up to 100 nodes and degrades with increased node size. The disadvantages of NS-2 are the interdependency between the modules and that co-simulation with MATLAB® requires special framework definitions [14]. Also, in order to implement the ZigBee® communication protocol, separate patch files must be installed. Even though this network simulation can be used for verifying different algorithms, NS-2 cannot be used easily for hardware-software co-simulation in MATLAB® as it requires separate framework definitions.

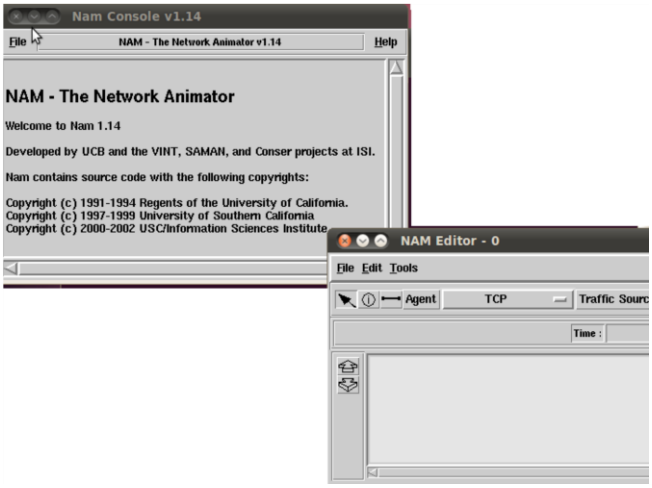


FIGURE 6. NS- 2 NAM editor window.

B. *OMNeT++*

OMNeT++ [11, 12, 15] is a modular simulation framework written in C++ that can be used for simulating ad-hoc networks. Frameworks developed in OMNeT++ that can be used for WSN analysis include: MiXiM, Mobility Framework, media access control (MAC) layers, Castalia, INET framework and NesCT. NesCT can be used for simulating TinyOS sensor based networks. PAWiS is also an OMNeT++ based simulator that captures a wide array of modules, and provides support for mobility and environmental dynamics [9]. OMNeT++ is a commercial software that can be used for educational and research purposes. The modules are written in the Network Description (NED) programming language. Co-simulation of MATLAB® and OMNeT++ can be achieved by converting C/C++ code into objects and compiling these objects in OMNeT++. However, it does not support communication during real time simulation.

C. *Prowler*

Prowler is an event driven wireless sensor network simulator designed to run on MATLAB® [11]. Simulation codes that implement routing protocols and other applications can be written in the MATLAB® language. This framework can be used for optimizing communication protocols [16]. Prowler can be installed in MATLAB® by simply adding the directory to the MATLAB® startup path. The editor window can be opened by typing "prowler" in the MATLAB® command window. A window, as shown in Figure 7, appears where the data transferred between the nodes can be visualized. Some routing protocols such as flood 1D, flood 2D, span tree and collision demo can be verified through Prowler. However, it cannot be used for developing a customized WSN framework. There is no clear documentation provided for developing user defined models, and no new versions have been released recently. Also, Prowler does not support the ZigBee® protocol.

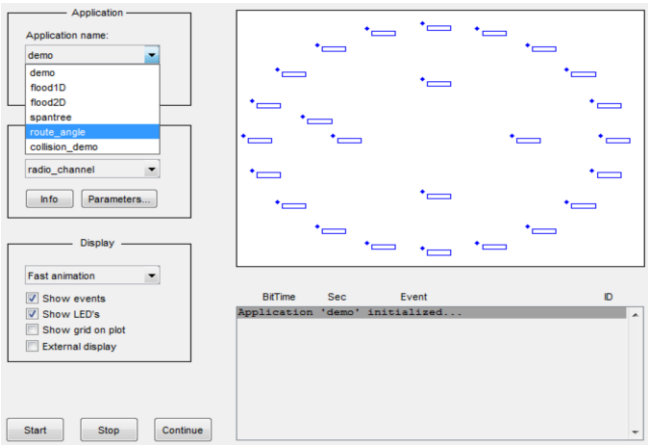


FIGURE 7. User interface of Prowler.

D. *Atarraya*

Atarraya is an event driven simulator that can be used for teaching and researching control topology algorithms and wireless sensor protocols [17]. It is designed to operate in the Windows® OS. The simulator can be installed by adding the directory path to the environmental variable. After installation, the Atarraya simulation panel, as shown in Figure 8, can be opened by double clicking the "Atarraya.jar" file. Atarraya is written in Java. The latest Oracle Java version must be installed for using Atarraya. The simulation panel consists of a deployment panel, a protocol selection panel, a visualization panel, a node stats and a report panel. In the deployment panel, the number of nodes, location, and size can be selected. The communication protocol can be selected from the protocol selection panel, and the nodes can be visualized in the visualization panel. The advantages of Atarraya are: it supports different topology construction and maintenance; it allows simulation of initial agent topology with available algorithm. In addition, Atarraya supports simple and walk-based mobility and energy models. On the other hand, the simulator does not support the ZigBee® protocol and cannot be integrated with MATLAB®/Simulink®.

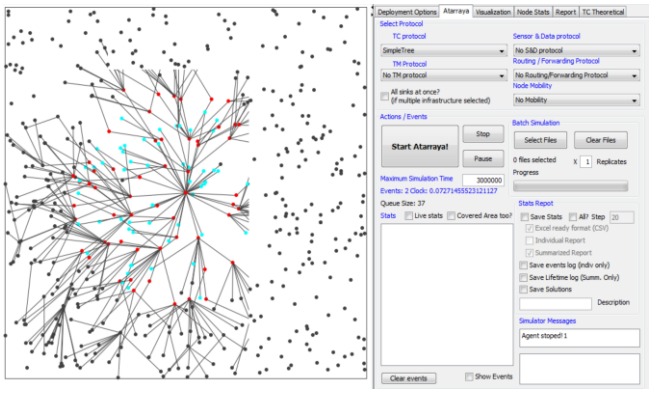


FIGURE 8. Atarraya simulation panel.

E. PiccSIM

PiccSIM [18] is a simulation platform for integrated communication, control design simulation, implementation and modeling. This tool kit can be used for co-simulation of networked control systems (NCS). The main advantage is that the simulator can be integrated with Simulink® and NS2.34. But two separate computers - one running NS-2.34 and the other running MATLAB® - are required. For analysis, NS-2.34 was installed in a Virtual machine, and MATLAB®/ Simulink® was installed in the host machine. Since NS-2.34 was an older version, some bugs had to be rectified before installation. After successfully installing NS-2.34, the PiccSIM-NS-2.34 patch was installed. But the patch files corrupted NS2.34 with an error, as shown in Figure 9. The simulator is no longer supported by the developer and the error in the patch could not be rectified.

```
[code omitted because of length]
: invalid command name "Agent/Tap"
    while executing
"Agent/Tap set superNetwork_ 0  "
```

FIGURE 9. PiccSIM- NS-2.34 patch error.

F. Truetime

TrueTime is a real-time control system based framework that operates in MATLAB®/Simulink®, as shown in Figure 10 [19]. TrueTime is written in the C++ and MEX languages. TrueTime can be installed in MATLAB® by adding its path to the PATH environmental variable and the MATLAB® startup path. In order to compile programs in Truetime, Microsoft Visual Studio must be installed. The Truetime package comprises of network blocks such as Ethernet, controller area network (CAN), time division multiple access (TDMA) and frequency division multiple access (FDMA), Round Robin, Switched Ethernet, FlexRay and PROFINET. It also consists of wireless network blocks such as WLAN 802.11b and 802.15.4 ZigBee® technology, as shown in Figure 10. Truetime supports battery power sources and can act as a stand-alone network interface block. It allows co-simulation of controller task execution in real-time, network transmission, and continuous plant dynamics. The disadvantage with true time is that clear documentation for programming the kernel blocks is not available, which made it difficult to use this framework.

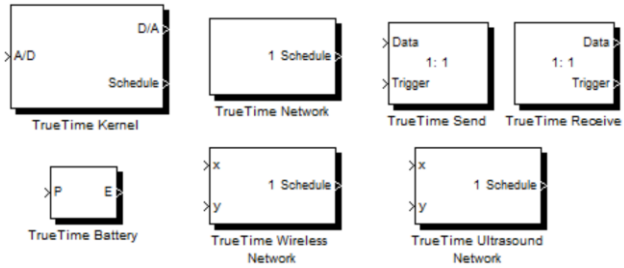


FIGURE 10. TrueTime Tool Box or Block Library.

G. MATLAB®/Simulink®

MATLAB®/Simulink® [20] is a software package for numeric computation and analysis that is developed and maintained by Mathworks® Inc., The software is flexible and reliable. Simulink® is a software package for modeling, simulating and analyzing dynamic systems. The software can be installed in the major operating systems such as Windows®, OS X, and Linux. Simulink® supports a wide

variety of toolkits such as digital signal processing toolkit, communication toolkit, control system, and embedded controller toolkit. As a result, it enables users to make customized designs with MATLAB®/Simulink® seamlessly. The key advantage of MATLAB®/Simulink® is that all these toolkits are documented with examples. Another benefit of MATLAB®/Simulink® is the automatic code generation feature, which promotes integration of real time processors with the simulation model. With Simulink®, customized node communication through the ZigBee protocol can be developed. With the code generation capabilities of MATLAB®, the simulation model can be integrated with hardware models easily. As a result, MATLAB®/Simulink® attractive for WSN framework simulations [1, 2].

Table 1. Comparative perspectives of simulation frameworks.

Framework	**OS**	**Compiler**	**Latest Update**	**Programming Language**	**Node size**	**MATLAB®/ Simulink® integration**	**Zigbee Support**
NS-2	Unix/ Windows with CYGWIN	C++, JDK 1.6	ns-2.35/2013	Tool Command Language (TCL)/Otcl	100 nodes Maximum	Yes	Yes
OMNeT++	Windows, OS X, Linux	C++11, JDK 1.7 or later	OMNeT 4.6/2014	NED (Network Description) Language	-	Yes	Yes
Prowler	OS that supports MATLAB®	Apple Xcode version 4.0 or higher Windows: C++, JDK	V1.25/2004	Graphical Programming tool (GUI)	Based on the type of application	Yes	No
Atarraya	Windows, requires GUI formatting for Linux	Java 6	1.3 beta/ 2011	GUI	Can simulate 1000 nodes	No	No
PiccSIM	Windows, OS X, Linux	Apple Xcode version 4.0 or higher Windows: C++, JDK	PiccSIM Simulink® Version 1.16 /2013	TCL/Otcl for network modelling	Similar to NS-2	Yes	Yes
TrueTime	Windows, OS X, Linux	Apple Xcode version 4.0 or higher Windows: C++, JDK, Microsoft visual studio	TrueTime 2.0 beta 7 / 2012	Graphical Programming tool	Limited	Yes	Yes
MATLAB®/ Simulink®	OS X, Windows, Linux	Apple Xcode version 4.0 or higher Windows: C++, JDK	R2015a	C, C++, Fortran	Code: > 100 nodes Simulation: Restricted	-	Yes

V. Summary and Conclusions

This article focuses on comparing WSN simulation frameworks that need to be integrated with an actual hardware platform. Various frameworks that can be used for simulating WSNs were analyzed. Specifically, software such as NS-2, OMNeT++, Prowler, Atarraya, PiccSIM, Truetime, and MATLAB®/Simulink® were analyzed in detail. These simulators were compared and studied thoroughly based on specific criteria. An important design

requirement is the co-simulation capability with the real-time processors and the support for the ZigBee network protocol. So, the simulators that did not meet these specifications were eliminated. Finally, it was found that MATLAB®/Simulink® met the most important design specifications such as customized node design, ZigBee, and co-simulation with hardware. Hence, MATLAB®/Simulink® can be effectively used for the framework design and simulation.

ABOUT THE AUTHORS

M. L. Rajaram (preeth.madhu05@gmail.com) obtained her Master's degree in Electrical Engineering Technology from the University of North Texas, USA in Dec. 2015.

Elias Kougianos (eliask@unt.edu) is currently an associate professor in Electrical Engineering Technology at the University of North Texas. He obtained his Ph.D. in electrical engineering from Louisiana State University in 1997. He is author or co-author of over 100 peer-reviewed journal and conference publications. He is a Senior Member of IEEE.

Saraju P. Mohanty (saraju.mohanty@unt.edu) is a Professor at the Department of Computer Science and Engineering, University of North Texas, and the director of the NanoSystem Design Laboratory. He obtained his Ph.D. in computer science and engineering from the University of South Florida in 2003, his master's degree in systems science and automation from the Indian Institute of Science, Bangalore, India, in 1999. He is the author of 200 peer-reviewed journal and conference publications and 3 books. He is an inventor of 4 US patents. He is a Senior Member of IEEE and ACM.

Uma Choppali (umachoppali@gmail.com) is currently an adjunct faculty at the Department of Engineering Technology at the University of North Texas, Denton. She obtained a Ph.D. in Material Science and Engineering from the University of North Texas in 2006. She holds a master's degree from the Indian Institute of Technology Bombay, India. She has authored 10 peer-reviewed publications.

REFERENCES

[1] S. P. Mohanty, Nanoelectronic Mixed-Signal System Design, McGraw-Hill, 2015, ISBN: 978-0071825719.

[2] M. L. Rajaram, "Comparative Analysis and Implementation of High Data Rate Wireless Sensor Network Simulation Frameworks", Master's Thesis, Department of Engineering Technology, University of North Texas, Fall 2015.

[3] G. Zhang, G. Liu, W. Chen and C. Yang, "Quantitative Analysis of Cluster-Head Selection for Wireless Sensor Networks," in *Proceedings of the World Automation Congress (WAC)*, 2012, pp. 277-281.

[4] T. Rault, A. Bouabdallah, and Y. Challal, "WSN lifetime optimization through Controlled Sink Mobility and Packet Buffering," in *Proceedings of the Global Information Infrastructure Symposium*, 201, pp. 1-6.

[5] Y. Lei, "Research and Implementation of WSN-based Data Acquisition and Analysis System Using Agent Collaboration," in *Proceedings of the 2nd International Conference on Power Electronics and Intelligent Transportation System*, 2009, pp. 235-237.

[6] V. Michopoulos, L. Guan, and I. Phillips, "A New Congestion Control Mechanism for WSNs," in *Proceedings of the 10th IEEE International Conference on Computer and Information Technology*, 2010, pp. 709-714.

[7] J. Ma, W. Lou, and X.-Y. Li, "Contiguous Link Scheduling for Data Aggregation in Wireless Sensor Networks," *IEEE Transactions On Parallel And Distributed Systems*, Vol. 25, No. 7, July 2014, pp. 1691-1701.

[8] N. Fatima and S. Gambhir, "Op-Leach: An Optimized LEACH Method for busty Traffic in WSNs," in *Proceedings of the 4th International Conference on Advanced Computing & Communication Technologies*, 2014, pp. 222-229.

[9] P. Shrivastava and S. B. Pokle, "An Energy Efficient Scheduling Strategy for Data Collection in Wireless Sensor Networks," in *Proceedings of the International Conference on Electronic System, signal Processing and Computing Tchnologies*, 2014, pp. 170-173.

[10] P. Sommer and R. Wattenhofer, "Gradient Clock Synchronization in Wireless Sensor Networks," in *Proceedings of the International Conference on Information Processing in Sensor Networks*, 2009, pp. 37-48.

[11] S. A. Madani, J. Kazmi, S. Mahlknecht, "Wireless sensor networks: modeling and simulation," InTechopen, http://cdn.intechopen.com/pdfs-wm/11548.pdf, Last visited on 14 December 2015.

[12] Q. I. Ali, "Simulation Framework of Wireless Sensor Networks (WSN) using MATLAB®/Simulink® Software", http://cdn.intechopen.com/pdfs-wm/39337.pdf, Last visited on 14 December 2015.

[13] The Network Simulator - ns-2, http://www.isi.edu/nsnam/ns/, Last visited on 14 December 2015.

[14] O. Heimlich, R. Sailer, and L. Budzisz, "NMlab: Cosimulation Framework for MATLAB® and NS-2," in *Proceedings Second International Conference on Advances in System Simulation (SIMUL)*, 2010, pp. 152-157.

[15] OMNeT++, http://www.omnetpp.org, Last visited on 14 December 2015.

[16] G. Simon, P. Volgyesi, M. Marioti, and A. Ledeczi, "Simulation-based Optimization of Communication Protocols for Large-Scale Wireless Sensor Networks," in *Proceedings of the Aerospace Conference*, 2003, pp. 1339-1346.

[17] Attaraya, http://www.cse.usf.edu/~labrador/Atarraya/, Last visited on 14 December 2015.

[18] PiccSIM, http://wsn.aalto.fi/en/tools/piccsim, Last visited on 14 December 2015.

[19] TrueTime, http://www.control.lth.se/truetime, Last visited on 14 December 2015.

[20] MATALB®/Simulink®, http://www.mathworks.com, Last visited on 14 December 2015 2015.