International Journal of Grid and Distributed Computing
Vol. 5, No. 3, September, 2012

Survey of Simulators for Wireless Sensor Networks

Bartosz Musznicki and Piotr Zwierzykowski
Poznań University of Technology, Faculty of Electronics and Telecommunications, Chair of Communications and Computer Networks, Poznań, Poland
bartosz@musznicki.com, piotr.zwierzykowski@et.put.poznan.pl

Abstract

Wireless Sensor Networks (WSNs) have been gaining growing interest in the past years, which has resulted in many proposals of their new applications. The main tasks of a sensor network include not only monitoring and measuring certain phenomena, but also delivering gathered data. Hence, every single node needs to be a wireless router aside from being only a simple sensing device. The idea and the attributes of a Wireless Sensor Network require designers to apply different techniques than those hitherto used in wired as well as in wireless networks. The dynamic progress in the field of WSNs entails the need of creating simulators that have more specific capabilities. Such simulation tools should allow researchers to verify new ideas and compare the proposed solutions in a virtual environment helping to avoid unnecessary, time-consuming or expensive hardware implementations. When beginning to work with Wireless Sensor Networks, it is important to choose a simulation environment that will be up to requirements and will allow the researcher to conduct experiments in a given area. Appropriately, the researcher faces a necessity of finding and getting oneself familiar with many simulators, often designed for radically different applications. This article presents a classification and a wide review the main features of various simulation platforms.

Keywords: *Wireless Sensor Networks, WSN, simulators, emulators, survey, review*

1 Introduction

When a new idea starts to grow in the researcher's mind, it is just the beginning of a long road leading to results. Novel concepts in most cases require a specific set of tools, other than just a pen and a sheet of paper, to develop into fully grown solutions. It is undeniable that the use of a simulation software package can often be the only way of reaching the goal, or at least can make the whole process easier and faster.

At the first glance, it may seem easy that everything we need to do is to find an appropriate program and get to work. Unfortunately, this is a very misleading assumption. As our experience shows, looking for the right environment for Wireless Sensor Networks simulations is a very arduous task. There are several surveys of sensor networks simulators [1–4], covering either only a number of currently available simulators, or including commercial products. Some bring useful contribution, such as the presentation of main

simulation methodologies [5], a comparison of case studies and simulation results [6], simulation scalability [7], as well as an introduction of testbeds [8]. New programs have been developed, other frameworks have disappeared or evolved causing some of the information to be outdated. Therefore, the aim of this unique article is to provide a review of 36 available WSNs simulation tools. We hope that the introduced classification will speed up the decision process and help researchers to focus their attention on the software that meets specific requirements.

The article is divided into eleven sections. Section 2 presents the classification criteria proposed by the authors. Sections from 3 to 10 describe the introduced categories and review available simulation environments. Section 11 sums up the discussion.

2 Classification criteria

This survey is directed at presenting as many publicly available WSNs open source simulators as we were able to find. Therefore, the initial condition is that each described environment can be downloaded from the Internet and used free of charge, at least for academic purposes.

Particular simulation platforms vary in many aspects, so a choice of the right one will become much easier after limiting the number of potential candidates. Hence, the simulators have been divided into categories according to their features and main applications. Each category presents the programs in the alphabetical order. When available, the screenshots of Graphical User Interfaces (GUIs) have been included. Simulators based on NS-2, OMNeT++ and Ptolemy II frameworks should be treated, in general, as the sub-groups of the Cross level simulators. The division has been introduced for convenience and to provide clarity.

3 Emulators and code level simulators

Emulators and code level simulators described in the next sections emulate the sensor hardware or process the provided program code in a manner it would be executed on a real device. Table 1 presents the basic information concerning this group of tools.

3.1 ATEMU

The main features of ATEMU (ATmel EMUlator) are related to the low-level emulation of MEMSIC Mica2 sensor platform (formerly known as Crossbow Mica2). Nevertheless, according to the framework authors, the program is customizable enough to be extended to support different hardware platforms used in heterogeneous network simulations [9]. The user is allowed to set different system parameters for each single node. Out of the box, the simulator is binary compatible with MICA2 sensor node, emulating the processor, radio interface, timers, LEDs and other devices, making the platform able to run TinyOS [10] operating system. CPU instructions are decoded and executed according to the Atmel ATmega 128L microcontroller specification. ATEMU comes with XATDB, a graphical debugger presented in Figure 1, offering setting breakpoints, showing values of variables, statuses of peripherals, etc. The platform supports a configuration specification based

on XML files, defining the hardware and software configuration along with the physical location of each node.

3.2 Avrora

Avrora is a command-line framework capable of simulating and analyzing programs developed for MEMSIC Mica2 and MicaZ sensor platforms. In the simulation each node has its own separated thread [11]. Avrora supports applications written using the Atmel and GNU assembler syntaxes. It likewise recognizes the avr-objdump utility output format. The emulator comes with a set of monitoring and profiling utilities [12] providing the ability of setting breakpoints and timeouts. Control flow graphs, the graphical representation of executed instructions, may prove to be useful as well.

3.3 EmSim

EmSim is a program capable of simulating networks of 32-bit nodes running the Linux-based EmStar software environment [13]. In the simulation, a separate EmStar process tree corresponds to each node in the web, thus developed and tested applications may be directly implemented in the production environment. Apart from maintaining communication with virtual nodes, sensors are able to interact with real devices using identical interfaces in both cases. The latter approach, called "emulation mode" provides a noteworthy possibility of expanding and controlling already deployed testbeds. What is more, extensions offered by EmStar cooperate with EmSim as well [14]. In this way, EmTOS enables EmSim to use nesC [15] applications as modules of the simulation. Additionally, EmView visualization and analysis tool presents the topology and current status of the network.

3.4 Freemote Emulator

The Freemote Emulator is a tool for developing software for nodes based on the Java and IEEE 802.15.4 standard [16]. The software is a part of the Freemote Environment that also includes the Freemote Testbed [17]. System architecture defines the following layers: Physical, Data Link (MAC), Routing and Application. There are three node types introduced in the project: Emulated Nodes run by the simulator in separated threads, Bridge Nodes acting as gateways to the real world and Real Nodes that compose the testbed. The framework executes the same code on emulated as well as Java-based devices and

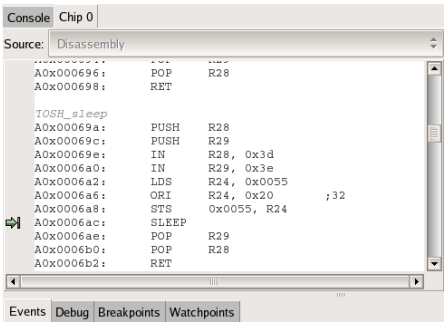

Figure 1. XATDB – the graphical debugger front-end to ATEMU
Source: http://www.hynet.umd.edu/research/atemu/userdoc

supports communication with MEMSIC TelosB and MICAz hardware platforms running TinyOS [18]. The Emulator may be installed either as a standalone application or as a web service, providing remote access to the environment. The simulation can be started in two ways, from the command line loading a prepared configuration XML file, or using a Graphical User Interface allowing to set-up parameters and visualize the experiment.

3.5 MSPsim

MSPsim is an emulator for WSN nodes based on Texas Instruments MSP430 microcontroller. Therefore, the program simulates and displays a visual representation of the whole sensor board, depicted in Figure 2, equipped with elements such as sensors, interfaces and LEDs. The developers state that along with the proper design and implementation in Java, the number of supported node platforms may be easily increased [19]. MSPsim is a part of the Contiki Operating System [20] and can be used in cross-level simulations conducted with the COOJA platform [21]. The instruction level simulator is capable of loading unmodified target platform ELF and IHEX firmware files, supporting monitoring and profiling of the executed code.

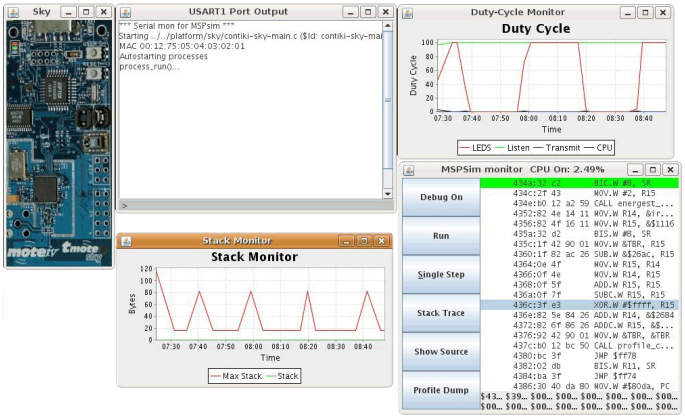

Figure 2. MSPsim – the GUI
Source: http://docs.tinyos.net/tinywiki/index.php/Mspsim

3.6 TOSSIM

TOSSIM is an emulator for WSNs operating under the control of TinyOS operating system. The environment simulates networks at the bit level, so, for instance, hundreds of simulated nodes may communicate with a number of actual nodes and create a common topology running exactly the same TinyOS applications [22]. TOSSIM is included in the TinyOS and currently supports the MICAz node platform, emulating radio interface, Analog-to-Digital Converters (ADCs) and EEPROM. The simulator does not model the real environment, yet provides a probabilistic representation of transmission errors occurring between two nodes. TinyViz is a GUI tool allowing user to visualize, monitor, control and debug running simulations [23]. One of its properties is the possibility to capture and inject radio messages. TinyViz has the AutoRun feature, a special script allowing it to run multiple simulations, set breakpoints, and define actions taken before and after each simulation such as collecting statistics. To empower an extensive analysis of the WSN hardware

design, a project named SUNSHINE [24] has been started. It focuses on the integration of TOSSIM, an Atmel AVR family microcontroller simulator SimulAVR, and a hardware simulator GEZEL. Tython [25] is a scripting environment that extends TOSSIM to repeatedly execute complex simulations in different scenarios. Another add-on tool called SimX [26] complements TOSSIM with simulation speed control, topology manipulation and variable watching. When the energy constraints are crucial, one can use PowerTOSSIM [27], a power modelling extension to TOSSIM.

3.7 VMNet

VMNet is an emulator oriented at providing performance evaluation for WSN applications. A real network is emulated as a Virtual Mote Network (VMN), whereas a single node is modelled as a Virtual Mote (VM) [28]. The Virtual CPU (VCPU) module is based on a modified ATEMU emulator, hence it supports a low-level simulation of MEMSIC MICA2 devices. The transmission medium is simulated with respect to the effects of noise and signal fading [29]. VMNet offers two performance metrics: power consumption and response time, calculated from logs gathered during the simulation.

3.8 WSim

A sensor node platform emulator called WSim supports running, analysis and debugging of applications that use the exact firmware the target platform operates on. WSim reflects hardware behaviour of a number of Texas Instruments MSP430 microcontroller-based sensor platforms along with flash memories, radio communication chipsets, and peripherals. The emulator does not depend on any embedded operating system, so one of popular WSN software platforms such as ContikiOS, TinyOS, FreeRTOS [30] and MANTIS OS [31] can be used. WSim can run the target platform code without modification or recompilation. Timing and interrupt data, as well as memory and power consumption, can be estimated during simulation allowing developers to perform thorough analyses and evaluations. Together with WSNet, WSim is a part of Worldsens development environment designed for prototyping WSN applications [32]. WSim can cooperate with WSNet through radio interface models, and supports cross-level simulations presented in Figure 3. Large simulations can be carried out on distributed computers exchanging information by means of IP multicast.

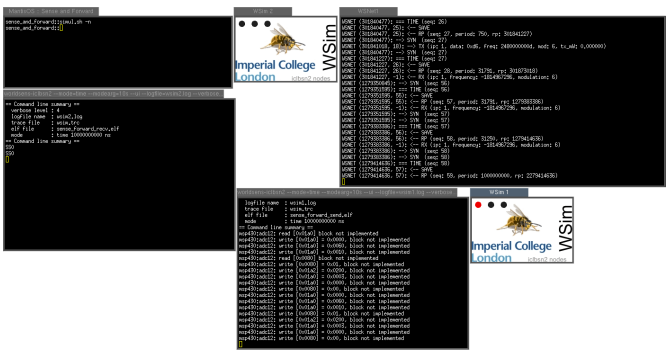

Figure 3. WSim – MANTIS OS sense and forward demo application
Source: http://wsim.gforge.inria.fr/tutorials/wasp/distributed.html

Table 1. Emulators and code level simulators

No.	*Simulator*	*Prog. language*	*Version*	*Release date*
1.	**ATEMU**	C	0.4	31 March 2004
		http://www.hynet.umd.edu/research/atemu/		
2.	**Avrora**	Java	1.7.106	9 August 2008
		http://compilers.cs.ucla.edu/avrora/		
3.	**EmSim**	C	2.5	5 October 2005
		http://www.cvs.cens.ucla.edu/emstar/		
4.	**Freemote Emulator**	Java	1.0	11 January 2008
		http://www.assembla.com/wiki/show/freemote/		
5.	**MSPSim**	Java	0.97	30 April 2009
		http://www.sics.se/project/mspsim/		
6.	**TOSSIM**	C/C++, Python	2.1.1	6 April 2010
		http://www.tinyos.net		
7.	**VMNet**	C	1.0.2	30 October 2005
		http://www.cse.ust.hk/vmnet/		
8.	**WSim**	C	-	4 January 2012
		http://wsim.gforge.inria.fr		

4 Topology control simulator – Atarraya

Atarraya is the only simulator aimed at topology control in WSNs the present researchers have succeeded to find (Tab. 2). The framework places emphasis on two consecutive processes: topology construction and topology maintenance [33], hence appropriate algorithms and protocols can be designed or tested. Users get opportunity to examine various solutions by comparing performance and efficiency in an environment providing a pre-defined energy consumption and communication models. The simulator comes with a tool for topology generation based on different distributions and a visualization interface, presented in Figure 4 that shows changes taking place during the experiment.

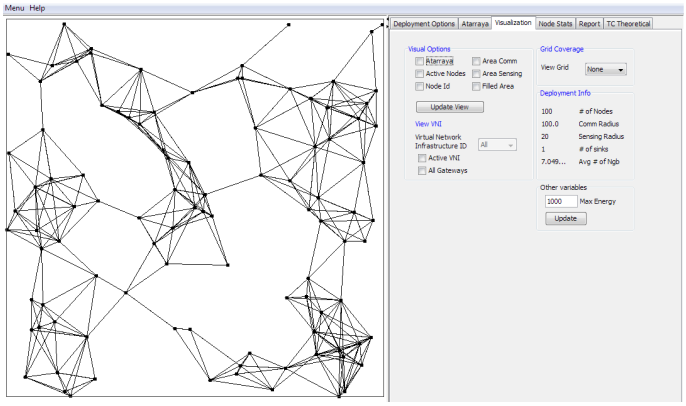

Figure 4. Attaraya – a sample topology visualization

Table 2. Topology control simulator

No.	*Simulator*	*Prog. language*	*Version*	*Release date*
1.	**Atarraya**	Java	1.2.3	6 April 2010
		`http://www.cse.usf.edu/~labrador/Atarraya/`		

5 Environment and wireless medium simulators

The platforms simulating the physical environment of a node and the wireless medium used for the transmission are briefly described in the following sections (Tab. 3).

5.1 Prowler

Prowler is a probabilistic WSN simulator running under the MATLAB environment [34]. The tool supports the Gaussian and Rayleigh radio propagation models, complemented with two collision detection schemes and the following MICA2 MAC-layer communication protocol. Aside from the deterministic approach, Prowler can be set to operate in a probabilistic mode, simulating the random aspects of the radio channel (fading, signal strength variations, etc.). At the application level, a set of basic events and actions related to the simulation is defined. Figure 5 depicts the GUI of Prowler. JProwler [35] is an equivalent to Prowler, but implemented in Java and available from the TinyOS CVS repository.

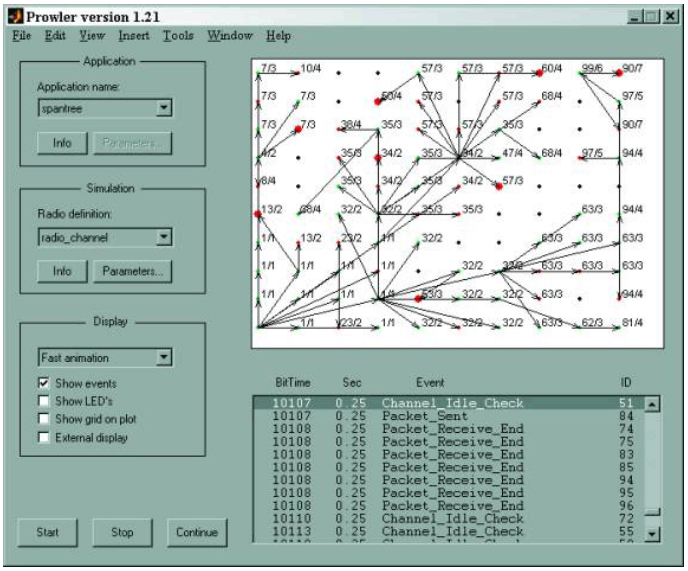

Figure 5. Prowler – the GUI
Source: http://w3.isis.vanderbilt.edu/projects/nest/prowler/pics.html

5.2 Wireless Sensor Network Localization Simulator

A determination of the location of sensor nodes is a simulation task that Wireless Sensor Network Localization Simulator is designed for. The program comes with eight localization algorithms, while other required procedures may be implemented. Numerous parameters

that define network topology include: network size, locators deployment strategy and antenna type, as well as the path loss and node mobility. The graphical overview of the network is presented in the GUI shown in Figure 6.

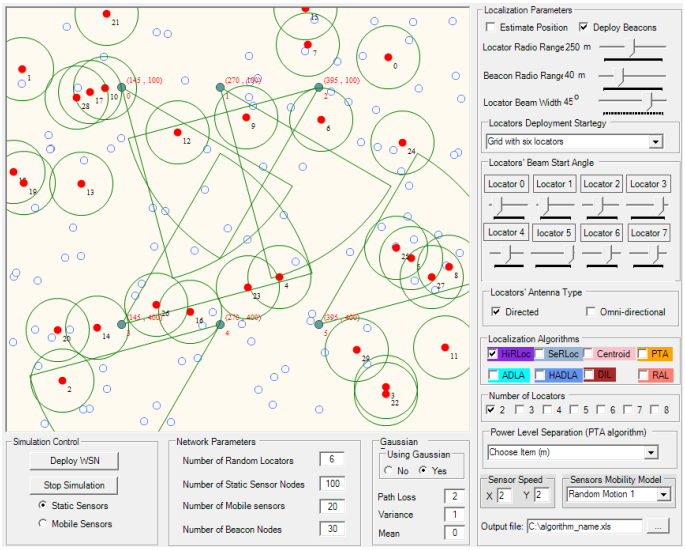

Figure 6. Wireless Sensor Network Localization Simulator – the GUI

5.3 WSNet

WSNet allows researchers to simulate environment focusing on physical measures and phenomena [36] (e.g. spreading fire damaging nodes, see Figure 7). It also offers means to analyze energy consumption and provides mobility. Radio communication models involve a simple ideal physical layer, as well as more complex representations of fading, interference, delays, transmission errors, modulations and channel access methods. Users can extend simulator scope of operation in upper network layers by developing custom models. WSNet attached to WSim emulator is targeted to be a tool supporting development and in-depth

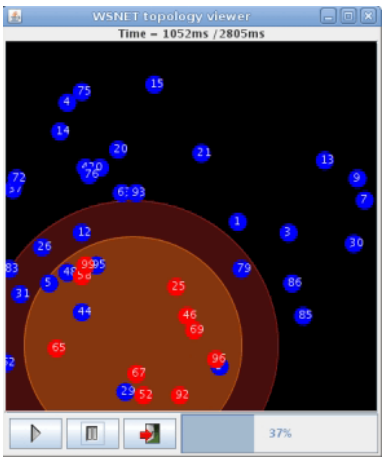

Figure 7. WSNet – topology viewer
Source: http://wsnet.gforge.inria.fr/tools.html

investigation of WSN applications.

Table 3. Environment and wireless medium simulators

No.	*Simulator*	*Prog. language*	*Version*	*Release date*
1.	**Prowler**	MATLAB/Java	1.25	28 January 2004
		http://w3.isis.vanderbilt.edu/projects/nest/prowler/		
2.	**Wireless Sensor Network Localization Simulator**	C#	1.1	14 July 2011
		http://www.codeproject.com/Articles/225536/Wireless-Sensor-Network-Localization		
3.	**WSNet**	C	9.07	16 July 2009
		http://wsnet.gforge.inria.fr		

6 Network and application level simulators

The software presented in the next sections is mainly directed at the simulation of the network as a structure transporting and processing gathered data. The basic information about this group of simulators is gathered in Table 4.

6.1 AlgoSenSim

AlgoSenSim is directed precisely at the simulation of distributed routing algorithms for WSNs. The framework is customizable through XML configuration files. The simulator offers several algorithms and supports implementation of new ones. Sensor nodes consist of three logical components defining the position information, message delivery method and the routing algorithm [37]. AlgoSenSim provides a graphical user interface facilitating the comprehension and analysis of the routing process, see Figure 8. Connections may be visualized and paths of the messages can be checked. The researcher can also use tools creating and showing saved animations, collecting diverse statistics, etc.

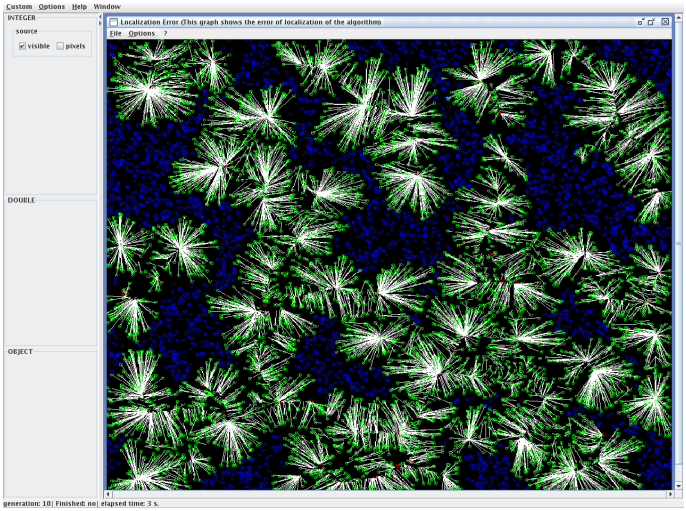

Figure 8. AlgoSenSim – the GUI
Source: http://tcs.unige.ch/doku.php/code/algosensim/using.the.gui

6.2 NetTopo

NetTopo comprises simulation and visualisation frameworks that share common elements for high level operations [38]. The GUI, shown in Figure 9, allows the user to view the network topology and control the properties of selected nodes. The simulator is designed to simplify studies of various WSNs algorithms. Consequently, modules are used to represent built-in and user-defined controlling, scheduling, routing or clustering algorithms. Two-Phase geographic Greedy Forwarding (TPGF) and Greedy Perimeter Stateless Routing (GPSR) are already implemented as the examples of routing algorithms [39]. Sensors can be deployed forming various shapes, e.g., circle, triangle, tree and grid. Apart from simulation, the environment performs actions such as logging changes and recording statistics. The Visualizer is a tool for presenting data gathered by the sink of an actual WSN testbed. Information sensed in the real environment is converted into a XML data stream and sent to NetTopo through a TCP/IP connection.

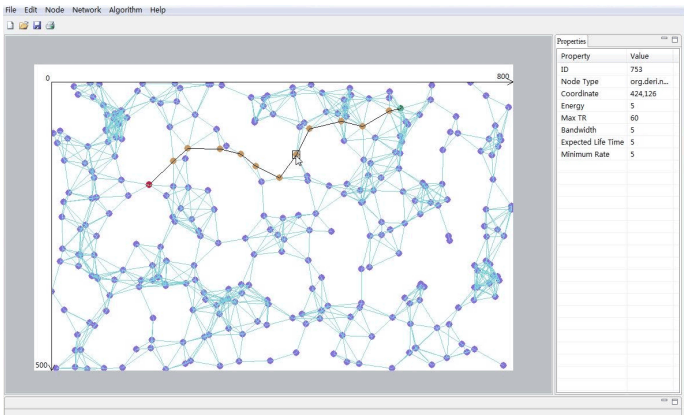

Figure 9. NetTopo – The GUI
Source: http://www.semanticreality.org/nettopo/index.htm

6.3 SENSE

In SENSE, elements of the network are represented in a component-port architecture provided by CompC++ [40], an extension to C++. Available modules include battery models and MAC layer protocols (idealistic NullMAC and IEEE 802.11 with DCF). The implemented network layer protocols include Self Selective Routing (SSR) and Self Healing Routing (SHR). iNSpect [41] visualization tool can be used to analyze the simulation output and perform an animated playback showing message flows. G-Sense, presented in Figure 10, is an additional GUI tool designed for SENSE [42].

6.4 Sensor Security Simulator (S3)

Sensor Security Simulator (S3) is designed for the evaluation of selected WSN security aspects in networks that consists of thousands of sensors [43]. Secrecy amplification protocols may be modeled and studied in relation to several routing algorithms. S3 supports the analysis of the impact of selected nodes and encryption keys getting compromised on

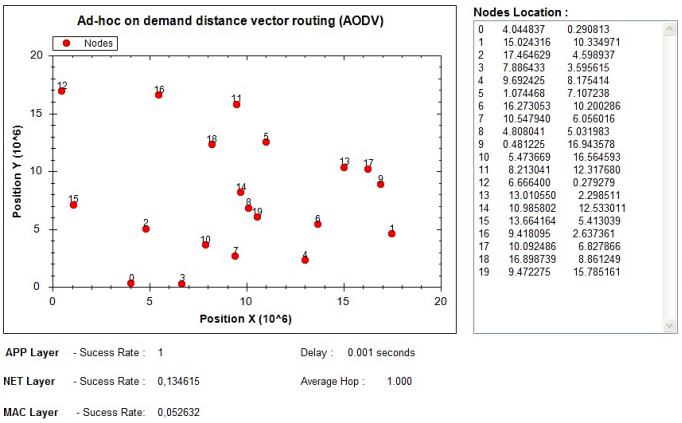

Figure 10. G-Sense – Simulation
Source: http://netgna.it.ubi.pt/Downloads/G-SENSE.php

the network operation and security. Every node is modeled as a memory-constrained computing device. Sensors are deployed in a rectangular area either randomly or following a user-defined topology. Simulation parameters may be defined in the GUI, presented in Figure 11, or in a configuration file. Experiments can be performed either locally or on multiple remote machines communicating with a central management application. Simulations are based on a pseudo-random number generator and a specified seed can be defined for each experiment. Therefore, simulation steps are fully deterministic, yet being complemented by random operations when required. Batch sequential execution of several simulations is supported and parameters of network topologies can be iterated over destined ranges. Data gathered during each simulation is stored in files formatted for further processing in MATLAB environment.

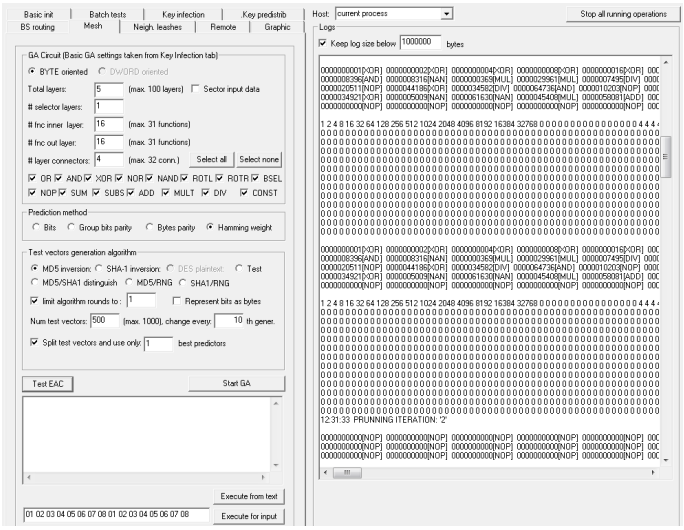

Figure 11. Sensor Security Simulator (S3) – the GUI

6.5 Shawn

Shawn has been developed to support the gradual evolution of an initial idea from a centralized algorithm to a decentralized protocol and, eventually, a fully distributed network protocol. Nevertheless, the researcher does not need to follow the complete development path and can perform only some of the analyses. Shawn is based on three elementary types of models [44]. Communication models determine if two nodes can communicate by means of exchanging messages. Implemented edge models are responsible for different ways of calculating and maintaining graph representations (connection maps) of the network. When a message is being prepared to be sent, one of the transmission models may be used to decide whether the message will get delayed, corrupted or dropped. The simulator provides components for distance estimations and nodes mobility. It is possible to run a separate application on each node in the network. Shawn is ready to execute software prepared for the iSense platform [45]. Simulation scenarios can be saved to and loaded from XML files. The runtime parameters can be typed in manually or loaded from a configuration file. A special extension entitled JShawn [46] is capable of interpreting simple Java-based scripts defined in the start-up files. Shawn's visualization capabilities are limited to generating Postscript files.

6.6 SIDnet-SWANS

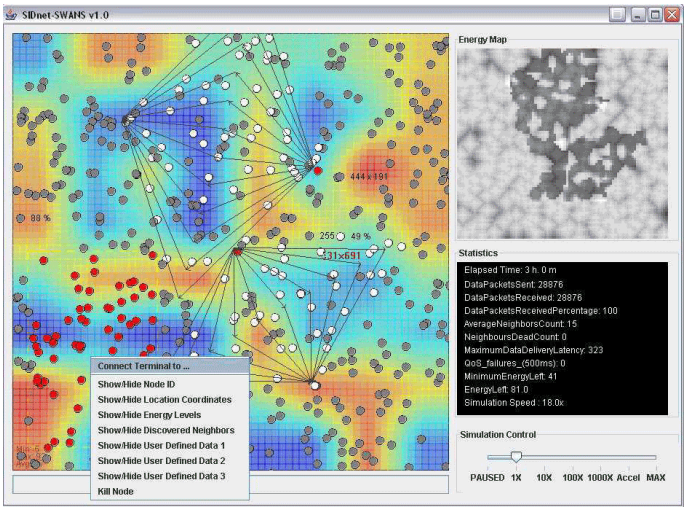

Figure 12. SIDnet-SWANS – the GUI
Source: http://users.eecs.northwestern.edu/∼ocg474/SIDnet.html

The architecture of SIDnet-SWANS is based on the JiST/SWANS platform [47] adapting it to meet the WSN characteristics. The program allows the user to model and to interact with a graphical representation of the network, see Figure 12, at different levels of granularity, i.e., a single node, a region, or a routing structure. The simulator supports energy consumption modelling and allows the researcher to define various phenomena, such as temperature, humidity and objects movement. Application layer procedures can be implemented and routing algorithms may be defined. Additionally, series of batch mode simulations can be performed with the use of a dedicated module [48]. The framework

gives also the user a component collecting and performing initial processing of simulation statistics.

6.7 Sinalgo

Sinalgo has a GUI supporting 2D and 3D animated visualisations, see Figure 13. The user is allowed not only to zoom in and out or rotate the area, but also to interact with sensors [49]. Snapshots of network graphs can be saved as a vector graphic in the EPS format. Sinalgo operates in synchronous or asynchronous mode, differing the way events occur. When the simulation is stopped, each node can be moved or deleted. In addition, general and more precise information can be displayed. When needed, simple statistics are collected during the execution. Mobility, connectivity, transmission reliability and interference models may be written by the user, increasing the number of accessible components. Nodes distribution and movement is possible both in two- and three-dimensional spaces. Simulations of different network algorithms may be stored into separate projects. Moreover, the GUI can be extended with project specific buttons and menus. Regrettably, to conduct an experiment as a series of related simulations an external scripting language must be used.

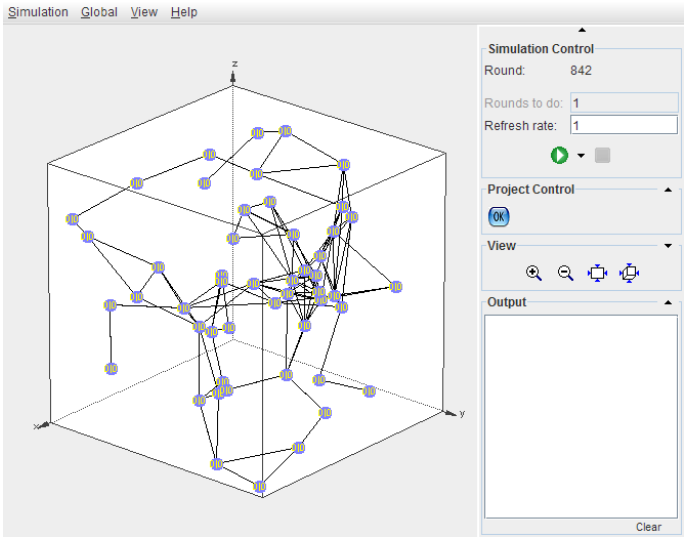

Figure 13. Sinalgo – a sample 3D visualization

6.8 TRMSim-WSN

TRMSim-WSN (Trust and Reputation Models Simulator for Wireless Sensor Networks) is a software designed to help researchers study and compare trust and reputation models. The simulation can be run over a single randomly generated WSN or over a set of networks. The user is able to define parameters of the network, such as the percentage of clients and that of malicious nodes, as it is shown in Figure 14. Network topologies may also be loaded from and saved to XML files. Sample trust and reputation models have been included and additional ideas can be implemented [50]. Nodes support switching over to idle state, as well as, oscillating their behaviour and acting in collusion with other malicious sensors [51].

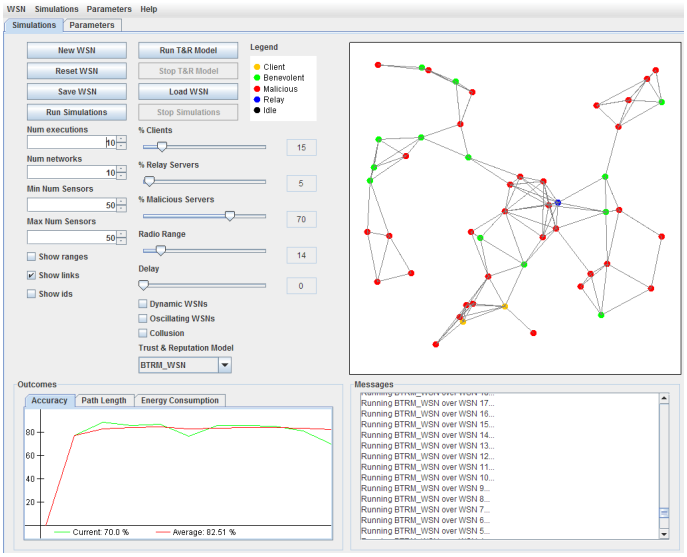

Figure 14. TRMSim-WSN – the GUI

6.9 Wireless Sensor Network Simulator

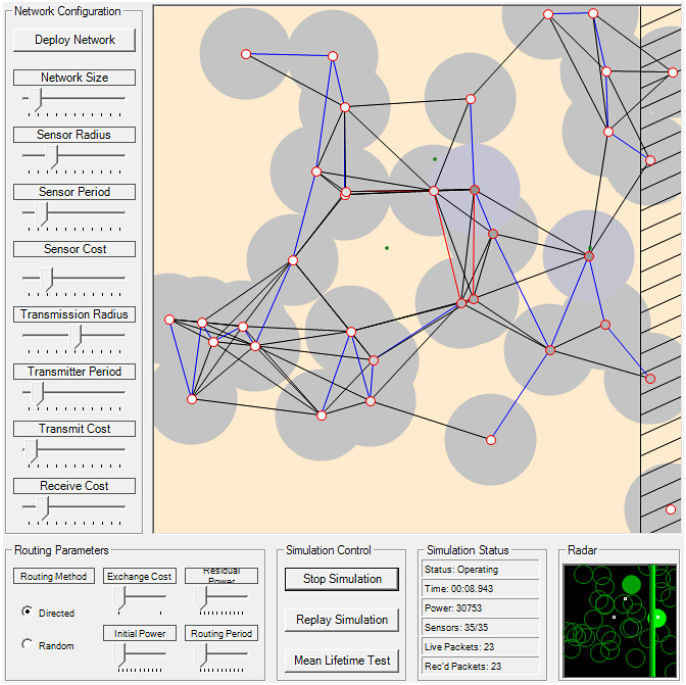

Figure 15. Wireless Sensor Network Simulator – the GUI

The software can be mainly recognized as an interesting compact demonstration of the concept of routing in WSNs. The user is allowed to tune different parameters including sensor radius, energy reserves and network size. Then, sensors powered by batteries are randomly deployed across the simulation area. During the simulation, each element moving within the sensing radius triggers the sensor. As a result, routing of an event notification

message towards the uplink zone begins. With every transmission, the energy stored in network nodes decreases. Figure 15 presents the GUI with an animated preview of the simulation. The average life-time test is available as well.

6.10 WsnSimPy

Trace based simulator called WsnSimPy relies on connectivity data gathered from an actual network using the WSN Profiler tool [52]. During the process, each node in the network broadcasts a specific number of packets. Every received packet is recorded and analyzed in a central profile server to create a connectivity trace used for comparative simulations. Alternatively, WsnSimPy can use a synthetic model to simulate radio communication. The simulator is developed from the discrete event simulator SimPy [53], extending it with generic models of the radio layer, link layer, network layer, and the application layer supporting performance analysis of WSN protocols. An implementation of Collection Tree Protocol (CTP) [54] is included.

Table 4. Network and application level simulators

No.	*Simulator*	*Prog. language*	*Version*	*Release date*
1.	**AlgoSenSim**	Java	0.9.2.2	21 September 2006
		http://tcs.unige.ch/doku.php/code/algosensim/overview		
2.	**NetTopo**	Java	1.3	1 August 2008
		http://nettopo.cvs.sourceforge.net/viewvc/nettopo/		
3.	**SENSE**	C++	3.1	19 November 2008
		http://www.ita.cs.rpi.edu/sense/		
4.	**Sensor Security Simulator (S3)**	C++	2.0.0.22	2008
		http://www.fi.muni.cz/~xsvenda/s3.html		
5.	**Shawn**	C++	-	-
		http://shawn.sourceforge.net		
6.	**SIDnet-SWANS**	Java	1.5.6	6 January 2011
		http://users.eecs.northwestern.edu/~ocg474/SIDnet.html		
7.	**Sinalgo**	Java	0.75.3	8 April 2008
		http://disco.ethz.ch/projects/sinalgo/		
8.	**TRMSim-WSN**	Java	0.5	25 March 2012
		http://ants.dif.um.es/~felixgm/research/trmsim-wsn/		
9.	**Wireless Sensor Network Simulator**	C#	1.1	1 December 2006
		http://www.djstein.com/projects/WirelessSensorNetworkSimulator.html		
10.	**WsnSimPy**	Python	1.0	3 July 2010
		http://pecs.mines.edu/trac/wiki/Release/WsnSimPy		

7 Cross level simulators

The next subsections contain descriptions of simulators capable of simulating nodes and networks at various levels of abstraction. Table 5 shows basic information for this group of simulation tools.

7.1 COOJA

COOJA is a simulator included in the Contiki Operating System [55] and mainly developed for wireless networks made up of sensors running this particular OS. COOJA is also available as a part of Instant Contiki [56], a virtual machine containing ready-to-go Contiki development environment. The simulator operates at three abstraction levels. The highest one is called the networking level, at which users can implement various applications and routing protocols. The program code developed and tested on COOJA at the operating system level may be run without modifications directly on Contiki OS. The machine code instruction level makes the emulation at a bit level possible due to the use of MSPSim, therefore the platform is able to emulate ESB nodes with TI MSP430 microcontrollers [57]. Simulated nodes can be of a different type, both in the hardware and software aspects, including sensors that do not use Contiki. The only limitation is that every particular node is simulated solely at one of the supported levels. It may seem as a disadvantage, but then a single network of sensors simulated at distinct levels of detail can give a wider look on the structure at an acceptable amount of time. The progress of the simulation can be observed using timeline visualizer depicted in Figure 16.

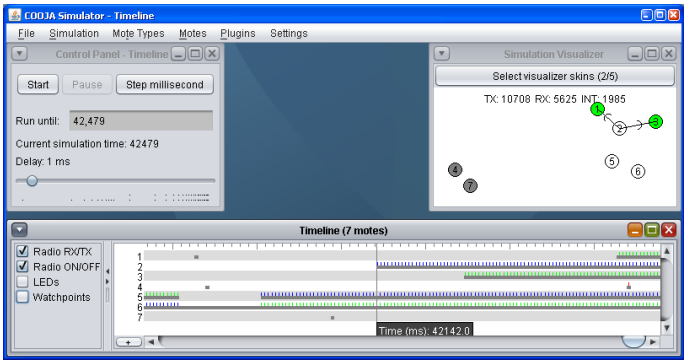

Figure 16. COOJA – the TimeLine visualizer
Source: http://www.sics.se/contiki/changelog.html

7.2 J-Sim and Sensor Network Package

A framework for modelling and simulating of WSNs has been developed in Java upon J-Sim network simulator. It provides models of communication channels, mobility, power consumption and nodes (battery, CPU, radio, protocol stack, etc.). To demonstrate applications of the simulator, A. Sobeih et al. [58] have implemented and described three algorithms: APS/DV-hop for distributed positioning, GPSR and Directed Diffusion. Authors have also defined forms of network emulation based on extracting, processing and transmitting data with the use of real devices.

7.3 SENS

SENS is a simulation tool consisting of components for physical environment, network communication and applications [59]. Module implementations are based on data and characteristics obtained from real sensor networks. Although power utilization analysis is

supported, phenomena detection capabilities are limited only to sound. The environment component yields models of various types of surfaces that influence the radio and sound propagation parameters. The simulator provides network models, responsible for packet losses, collisions and propagation delays.

7.4 WSN-Sim

A WirelessHART [60] technology focused simulator called WSN-Sim is shown in Figure 17. The software addresses deployment of sensor nodes and access points, connectivity, energy consumption and routing simulation. Latency and reliability monitoring is available as well.

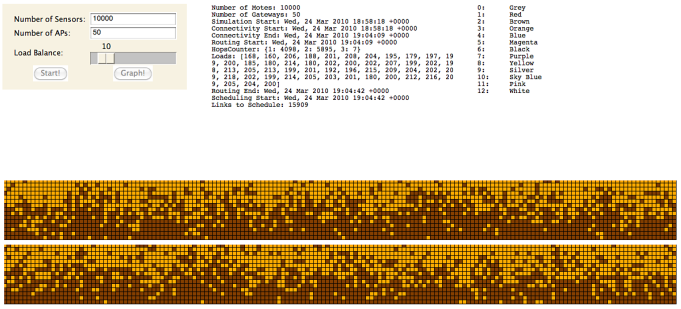

Figure 17. WirelessHART Large-Scale Deployment Simulation
Source: http://wsn.eecs.berkeley.edu/trac/simulation/browser/demo/SimDemo.swf

Table 5. Cross level simulators

No.	*Simulator*	*Prog. language*	*Version*	*Release date*
1.	**COOJA**	C	2.5	8 September 2011
		`http://www.contiki-os.org`		
2.	**J-Sim and Sensor Network Package**	Java	1.3 + patch4	5 July 2006
		`http://sites.google.com/site/jsimofficial/`		
3.	**SENS**	C++	-	31 January 2005
		`http://osl.cs.uiuc.edu/sens/`		
4.	**WSN-Sim**	Python	-	3 May 2010
		`http://wsn.eecs.berkeley.edu/trac/simulation/`		

8 NS-2 based simulators

NS-2 [61] is a popular general purpose network simulator mainly used in the studies of TCP, routing and multicast protocols. Although many Ad-Hoc protocols are implemented, the framework lacks support for WSNs and only the module for Directed Diffusion [62] is included. To satisfy some of the researchers' requirements, the following extensions have been developed (Tab. 6).

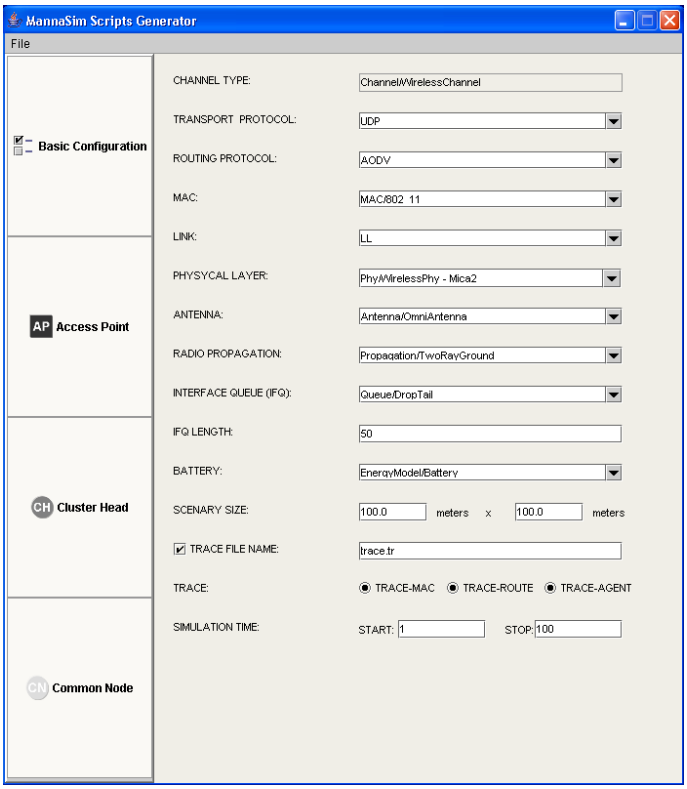

Figure 18. Mannasim Script Generator
Source: http://www.mannasim.dcc.ufmg.br

8.1 Mannasim

Mannasim offers modules for development and analysis of WSNs applications. Included generator of TCL simulation scripts, shown in Figure 18, allows users to configure simulation parameters in a graphical environment, instead of editing a text file. Two physical layer components are provided: MEMSIC Mica2 node and a 914MHz Lucent WaveLAN DSSS radio interface. In the simplest scenario, the sensor nodes of a single type constitute the network. Only when a hierarchical topology is being created, special cluster head nodes appear [63]. In Mannasim, routing protocols such as Ad hoc On-Demand Distance Vector (AODV), Directed Diffusion, Destination-Sequenced Distance-Vector Routing (DSDV), Dynamically Controlled Routing (DSR), Low-Energy Adaptive Clustering Hierarchy (LEACH) and Temporally Ordered Routing Algorithm (TORA) can be simulated.

8.2 NRL Sensorsim

NRL Sensorsim introduces the support of sensor networks in Ns-2. The tool provides the facilities to simulate and detect parameters of carbon monoxide concentration, seismic activity or audible sound [64]. Not only wireless sensors, but also phenomena sources and gateways could constitute the network. Power utilization parameters may be described with the use of energy consumption model. Nam [65], the Ns-2's network animator, can be used to visualize the simulation, see Figure 19.

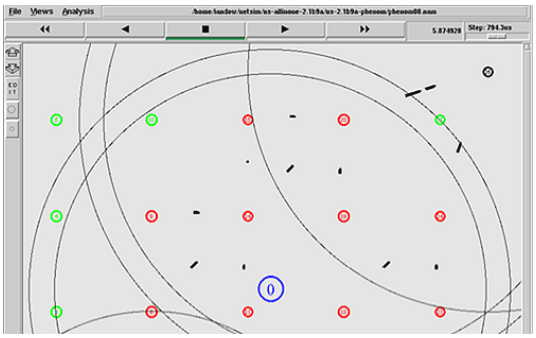

Figure 19. NRL Sensorsim – the GUI
Source: http://cs.itd.nrl.navy.mil/work/sensorsim/

8.3 RTNS

The Real Time Network Simulator (RTNS) is a software simulating operating system mechanisms for distributed networked applications. The tool is aimed at providing real time CPU simulations by the integration of the Ns-2 environment and the Real Time Operating System SIMulator (RTSim) [66]. Sample simulation scripts defining a tree topology and a simple cluster of nodes communicating with the sink are provided [67]. The program also offers basic mechanisms for distributed image processing.

Table 6. NS-2 based simulators

No.	*Simulator*	*Prog. language*	*Version*	*Release date*
1.	**Mannasim**	C++/Java	ns-2.29	16 August 2006
		`http://www.mannasim.dcc.ufmg.br`		
2.	**NRL Sensorsim**	C++/Tcl	ns-2.27	1 April 2004
		`http://cs.itd.nrl.navy.mil/work/sensorsim/`		
3.	**RTNS**	C++/Tcl	1.0	25 August 2008
		`http://rtns.sssup.it/RTNSWebSite/RTNS.html`		

9 OMNeT++ based simulators

OMNet++ [68] is a software environment directed at the simulation of communication networks. It is also widely used in other research areas, such as queuing systems or hardware emulation. There are several OMNeT++-based WSNs simulation frameworks. The related information is presented in Table 7.

9.1 Castalia

Castalia is meant to ease the cross-level analysis and the development of distributed algorithms or protocols. Framework modules are implemented with the use of the OMNeT++ NED (NEtwork Description) language. The simulator supports node clock drift, which can make some simulations more realistic and accurate. In Castalia, a module specifying the movement of the nodes through space is provided. Communication channel characteristics are based on empirically measured data and temporal variations of the signal strength

are implemented [69]. Realistic wireless medium and communication interface models are available. Radio model determines distinct power levels related to the node state. FSK and PSK are available, but the developer can also define additional modulations. Custom MAC, routing and application layer modules can be prepared.

9.2 MiXiM

MiXiM (mixed simulator) has emerged as a cross-level platform integrating four preceding frameworks: CHannel SIMulator, Mac Emulator, Mobility Framework and Positif Framework. The system comprises five base types of components: environment, connectivity and mobility, reception and collision, experiment support and protocol library [70]. Each logical network layer can be implemented by the user. For example, attenuation, noise and corruption effects may be defined and affect the signal. Many models are, however, provided in the package. The framework supports node mobility, models for walls or other obstacles, various network layers and protocols including IEEE 802.15.4. MiXiM implements also the OFDM and MIMO transmission techniques. Created modules can be connected or stacked in layers using the GUI. Additionally, the simulator provides means to visualize, monitor and debug the simulation.

9.3 NesCT

Thanks to NesCT [71] users are able to run TinyOS applications in the OMNet++ environment. Original NesC code is translated into C++ classes. Currently, NesCT is compatible with OMNeT++ 4.2 and 1.1.x releases of TinyOS.

9.4 PAWiS

PAWiS (Power Aware Wireless Sensors) framework is capable of simulating different types of nodes and networks due to the modular design, see Figure 20. Concurrent simulation of the power consumption of every sensor is its most crucial functionality [72].

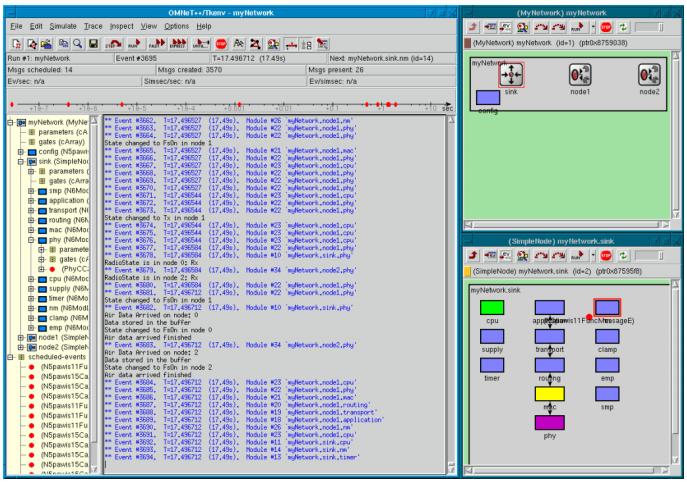

Figure 20. PAWiS – the GUI
Source: http://pawis.sourceforge.net

Generated simulation log files can be analyzed by a special application that visualizes timings, interactions and power requirements for each node. There is a module responsible for defining processor characteristics such as interrupt handling or power management [73]. To provide nodes with variable sensed data, environmental dynamics are simulated. Besides energy and hardware related issues, PAWiS simulates the operation of higher network levels. Users can define dynamic behaviors, such as mobility, with the use of the LUA [74] scripting language. The radio communication is based on a realistic model with noise, signal fading and interference. TDMA, FDMA and CDMA access methods are included. The simulator supports development of MAC and network protocols. Basic sensor specific tasks, such as environmental measurements, can be described in the application layer modules.

9.5 SENSIM (SensorSimulator)

The platform allows researchers to develop new sensor networks protocols and investigate networking and scalability. The hardware model of a node comprises battery, radio and CPU modules [75]. The battery model is used to control the amount of the remaining energy. Radio model defines the antenna power requirements. The processor model provides different levels of energy consumption in various states of operation. A wireless channel module controls and establishes connections between nodes. Free space and two-ray ground reflection propagation models can be used. Among others, Directed Diffusion with Geographical and Energy Aware Routing (GEAR) and Rumor Routing protocols have been implemented.

Table 7. OMNeT++ based simulators

No.	*Simulator*	*Prog. language*	*Version*	*Release date*
1.	**Castalia**	C++/NED	3.2	3 March 2011
		`http://castalia.npc.nicta.com.au`		
2.	**MiXiM**	C++/NED	2.2.1	15 December 2011
		`http://mixim.sourceforge.net`		
3.	**NesCT**	C++	-	4 August 2011
		`http://nesct.sourceforge.net`		
4.	**PAWiS**	C++/NED/Lua	2.0	1 July 2008
		`http://pawis.sourceforge.net`		
5.	**SENSIM (SensorSimulator)**	C++/NED	3.1	28 October 2005
		`http://csc.lsu.edu/sensor_web/simulator.html`		

10 Ptolemy II based simulators

Ptolemy II is a simulation framework providing experimentation with the actor-oriented design [76]. Software components, called actors, execute simultaneously, exchanging messages through interconnected ports and thus form hierarchical structures of models (Tab. 8).

10.1 Viptos

Viptos (Visual Ptolemy and TinyOS), integrated graphical development framework allows the researcher to simulate TinyOS-based WSNs in Ptolemy II environment (Fig-

ure 21) [77]. The idea is quite similar to that of NesCT authors. NesC files are transformed into MoML [78] files containing adequate Ptolemy II models. Viptos provides users with means to easily create TinyOS applications by drawing block and arrow diagrams. Later, the diagram will be automatically converted into a nesC 1.2 program file. Such an application can be compiled and directly transferred to a device running TinyOS [79]. The operation is feasible, because Viptos is based on TOSSIM and Ptolemy II (that includes VisualSense). TOSSIM provides a low-level simulation of TinyOS programs, while VisualSense offers other network capabilities. Hence, Viptos supports experiments concerning heterogeneous networks where distinct nodes may use different models at each level of simulation. Nodes that are not based on TinyOS can be also included in the network. The interaction with the node hardware is performed by Ptolemy II and then passed on to VisualSense to simulate wireless channel parameters such as delay and packet loss, etc.

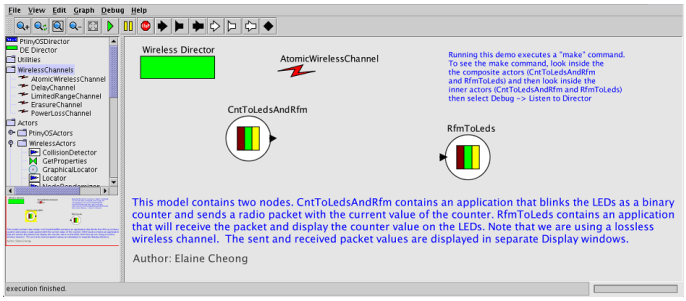

Figure 21. Viptos – the SendAndReceive application [79]

10.2 VisualSense

It is worthwhile to get familiar with VisualSense because of its quite unique approach to the simulation. As a part of Ptolemy II framework, the environment offers a user-friendly GUI for modelling and visualization. Thus, VisualSense can be used as a teaching aid allowing students to understand the principles of WSNs [80]. The framework consists of models representing communication channel and nodes (mobility, power management, packet losses, collisions, etc.). Hierarchical, heterogeneous systems are supported. To simulate anything different than a pre-built model introducing an idea of sound detection on a sensor field, presented in Figure 22, the user will have to create customized network models.

Table 8. Ptolemy II based simulators

No.	*Simulator*	*Prog. language*	*Version*	*Release date*
1.	**Viptos**	C++/Java	1.0.2	9 February 2007
		`http://ptolemy.berkeley.edu/viptos/`		
2.	**VisualSense**	Java	8.0.1	28 October 2010
		`http://ptolemy.berkeley.edu/visualsense/`		

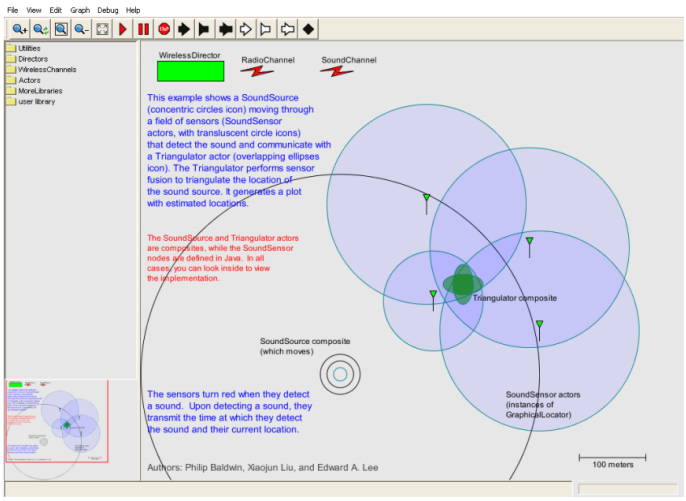

Figure 22. VisualSense – the representation of a wireless sound detection model [80]

11 Conclusions

Before we actually began to work on the article, we acquainted ourselves with the relevant professional literature presenting reviews and comparisons of WSN simulation environments [1–8]. Unfortunately, both their topicality and scope raised many reservations. Thus, our main motivation was to compose a survey that would put together and compile the most useful information.

In the process of preparing this article we were persistently looking for every WSNs simulator that could be found on the Internet. Although it cannot be ruled out that there are still programs yet to be brought to wider audience. What is even more likely, new software will emerge, so researchers should keep themselves up to date.

Besides using frameworks designed especially for WSNs simulations, some scientists have succeeded in making use of general purpose simulators in their sensor networks studies. They include NS-2 and OMNeT++, as well as other platforms. For instance, a WSN topology control algorithm has been implemented in CD++ using the Cell-DEVS formalism [81]. Following the examples, when none of WSN specific tools is suitable for the task, other environments may prove useful. In the most demanding situation, researchers will have to develop appropriate simulators of their own.

References

[1] E. Egea-López, J. Vales-Alonso, A. S. Martínez-Sala, P. Pavón-Marino, and J. García-Haro, "Simulation tools for wireless sensor networks," in *Proceedings of SPECTS 2005, Summer Simulation Multiconference*, (Philadelphia, USA), 24–28 July 2005.

[2] C. P. Singh, O. P. Vyas, and M. K. Tiwari, "A Survey of Simulation in Sensor Networks," in *Proceedings of CIMCA'08, 2008 International Conference on Computational Intelligence for Modeling Control and Automation*, (Vienna, Austria), 10–12 December 2008.

[3] M. Jevtić, N. Zogović, and G. Dimić, "Evaluation of Wireless Sensor Network Simulators," in *Proceedings of TELFOR 2009, 17th Telecommunications forum*, (Belgrade, Serbia), 24–16 November 2009.

[4] A. Kellner, K. Behrends, and D. Hogrefe, "Simulation Environments for Wireless Sensor Networks," tech. rep., Institute of Computer Science, Georg-August-Universität Göttingen, June 2010.

[5] S. Mahlknecht, S. A. Madani, and J. Kazm, "Wireless Sensor Networks: Modelling and Simulation, Discrete Event Simulations," *InTech*, 2010.

[6] K. Garg, A. Förster, D. Puccinelli, and S. Giordano, "Towards Realistic and Credible Wireless Sensor Network Evaluation," in *Proceedings of ADHOCNETS 2011, 3rd International ICST Ad Hoc Networks Conference*, (Paris, France), 21–23 September 2011.

[7] H. Sundani, H. Li, V. K. Devabhaktuni, M. Alam, and P. Bhattacharya, "Wireless Sensor Network Simulators A Survey and Comparisons," *International Journal of Computer Networks*, vol. 2, pp. 249–256, February 2011.

[8] M. Imran, A. M. Said, and H. Hasbullah, "A Survey of Simulators, Emulators and Testbeds for Wireless Sensor Networks," in *Proceedings of ITSim 2010, 4th International Symposium on Information Technology*, vol. 2, (Kuala Lumpur, Malaysia), pp. 897–902, 15–17 June 2010.

[9] J. Polley, D. Blazakis, J. McGee, D. Rusk, and J. S. Baras, "ATEMU: A Fine-Grained Sensor Network Simulator," in *Proceedings of SECON'04, First IEEE International Conference on Sensor and Ad Hoc Communications and Networks*, (Santa Clara, USA), 24–16 November 2004.

[10] "TinyOS operating system." `http://www.tinyos.net`. accessed June 2012.

[11] B. Titzer, D. Lee, and J. Palsberg, "Avrora: Scalable Sensor Network Simulation with Precise Timing," in *Proceedings of IPSN'05, Fourth International Conference on Information Processing in Sensor Networks*, (Los Angeles, USA), 25–27 April 2005.

[12] B. Titzer and J. Palsberg, "Nonintrusive precision instrumentation of microcontroller software," in *Proceedings of LCTES'05, Conference on Languages, Compilers and Tools for Embedded Systems*, (Chicago, USA), 15–17 June 2005.

[13] L. Girod, J. Elson, A. Cerpa, T. Stathopoulos, N. Ramanathan, and D. Estrin, "EmStar: a Software Environment for Developing and Deploying Wireless Sensor Networks," in *Proceedings of General Track: 2004 USENIX Annual Technical Conference*, (Boston, USA), 27 June – 2 July 2004.

[14] L. Girod, T. Stathopoulos, N. Ramanathan, J. Elson, D. Estrin, E. Osterweil, and T. Schoellhammer, "A System for Simulation, Emulation, and Deployment of Heterogeneous Sensor Networks," in *Proceedings of SenSys 2004, Second ACM Conference on Embedded Networked Sensor Systems*, (Baltimore, USA), 3–5 November 2004.

[15] "nesC: A Programming Language for Deeply Networked Systems." `http://nescc.sourceforge.net`. accessed June 2012.

[16] IEEE Computer Society, "Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)," in *IEEE Standard 802.15.4-2006, Part 15.4*, (New York, USA), 8 September 2006.

[17] T. Maret, "Overview of the Freemote Project." `http://www.assembla.com/wiki/show/freemote/Documentation`. accessed June 2012.

[18] R. Kummer, T. Maret, P. Kropf, and J. F. Wagen, "FREEMOTE: A Wireless Sensor Networks Emulation System," in *Proceedings of 7th MINEMA workshop*, (Lappeenranta, Finland), 21 August 2008.

[19] J. Eriksson, A. Dunkels, N. Finne, F. Österlind, and T. Voigt, "Mspsim – an extensible simulator for msp430-equipped sensor boards," in *Proceedings of EWSN, European Conference on Wireless Sensor Networks, Poster/Demo session*, (Delft, The Netherlands), 29–31 January 2007.

[20] "The Contiki OS." `http://www.contiki-os.org`. accessed June 2012.

[21] J. Eriksson, F. Österlind, N. Finne, A. Dunkels, and T. Voigt, "Accurate Power Profiling for Sensor Network Simulators," in *Proceedings of 8th Scandinavian Workshop on Wireless Ad-hoc and Sensor Networks*, (Stockholm, Sweden), 7–8 May 2008.

[22] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications," in *Proceedings of SenSys 2003, First ACM Conference on Embedded Networked Sensor Systems*, (Los Angeles, USA), 5–7 November 2003.

[23] P. Lewis and N. Lee, "TOSSIM: A Simulator for TinyOS Networks," *TinyOS documentation*, 17 September 2003.

[24] "SUNSHINE: A Software-Hardware Emulator for Sensor Networks." `http://rijndael.ece.vt.edu/sunshine/`. accessed June 2012.

[25] "Tython: A Dynamic Simulation Environment for Sensor Networks." `http://www.tinyos.net/tinyos-1.x/doc/tython/tython.html`. accessed June 2012.

[26] "SimX: an Integrated Sensor Network Simulation and Evaluation Environment." `http://sensorweb.cs.gsu.edu/?q=SimX`. accessed June 2012.

[27] V. Shnayder, M. Hempstead, B. Chen, G. W. Allen, and M. Welsh, "Simulating the Power Consumption

of Large-Scale Sensor Network Applications," in *Proceedings of SenSys 2004, Second ACM Conference on Embedded Networked Sensor Systems*, (Baltimore, USA), 3–5 November 2004.

[28] H. Wu, Q. Luo, P. Zheng, B. He, and L. M. Ni, "Accurate Emulation of Wireless Sensor Networks," in *NPC 2004, Network and Parallel Computing, IFIP International Conference*, (Wuhan, China), 18–20 October 2004.

[29] H. Wu, Q. Luo, P. Zheng, B. He, and L. M. Ni, "VMNet: Realistic Emulation of Wireless Sensor Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 2, pp. 277–288, 2007.

[30] "About FreeRTOS." `http://www.freertos.org/RTOS.html`. accessed June 2012.

[31] S. Bhatti, J. Carlson, H. Dai, J. Deng, J. Rose, A. Sheth, B. Shucker, C. Gruenwald, A. Torgerson, and R. Han, "MANTIS OS: An Embedded Multithreaded Operating System for Wireless Micro Sensor Platforms," in *Mobile Networks and Applications (MONET), Special Issue on Wireless Sensor Networks*, vol. 10, (Hingham, MA, USA), pp. 563–579, Kluwer Academic Publishers, August 2005.

[32] A. Fraboulet, G. Chelius, and E. Fleury, "Worldsens: Development and Prototyping Tools for Application Specific Wireless Sensors Networks," in *Proceedings of IPSN, International Conference on Information Processing in Sensor Networks*, (Boston, USA), April 2007.

[33] P. Wightman and M. A. Labrador, "Atarraya: A Simulation Tool to Teach and Research Topology Control Algorithms for Wireless Sensor Networks," in *Proceedings of SIMUTools 2009, Second International Conference on Simulation Tools and Techniques*, (Rome, Italy), 2–6 March 2009.

[34] G. Simon, P. Volgyesi, M. Maroti, and A. Ledeczi, "Simulation-based optimization of communication protocols for large-scale wireless sensor networks," in *Proceedings of IEEE International Aerospace Conference*, vol. 3, (Big Sky, USA), pp. 1339–1346, March 2003.

[35] "JProwler." `http://w3.isis.vanderbilt.edu/projects/nest/jprowler/`. accessed June 2012.

[36] "WSNet Overview." `http://wsnet.gforge.inria.fr/overview.html`. accessed June 2012.

[37] J. Fontignie and A. Marculescu, "AlgoSenSim: Developer's guide," 19 September 2006.

[38] L. Shu, C. Wu, and M. Hauswirth, "NetTopo: Beyond Simulator and Visualizor for Wireless Sensor Networks," tech. rep., Digital Enterprise Research Institute (DERI), 31 July 2008.

[39] L. Shu, Y. Zhang, L. T. Yang, Y. Wang, and M. Hauswirth, "Geographic Routing in Wireless Multimedia Sensor Networks," in *Proceedings of FGCN 2008, 2nd International Conference on Future Generation Communication and Networking*, (Sanya, China), 13–15 December 2008.

[40] B. K. Szymanski and G. G. Chen, "Sensor Network Component Based Simulator," in *Handbook of Dynamic System Modeling* (P. Fishwick, ed.), pp. 35–1 – 35–16, CRC/Taylor and Francis, 2007.

[41] C. Morrell, T. Babbitt, and B. K. Szymanski, "Visualization in Sensor Network Simulator, SENSE and Its Use in Protocol Verification," 2008.

[42] P. M. P. Rosa, P. A. C. S. Neves, B. Vaidya, and J. J. P. C. Rodrigues, "G-Sense – A Graphical Interface for SENSE Simulator," in *Proceedings of SIMUL 2009, The First International Conference on Advances in System Simulations*, (Porto, Portugal), 20–25 September 2009.

[43] P. Svenda, *The link key security in wireless sensor networks.* PhD thesis, Faculty of Informatics – Masaryk University, Brno, Czech Republic, September 2008.

[44] A. Kröller, D. Pfisterer, C. Buschmann, S. P. Fekete, and S. Fischer, "Shawn: A new approach to simulating wireless sensor networks," in *Proceedings of DASD'05, Design, Analysis, and Simulation of Distributed Systems*, (San Diego, USA), 5 April 2005.

[45] "iSense." `http://coalesenses.com/index.php?page=isense-hardware`. accessed June 2012.

[46] "JShawn." `http://shawnwiki.coalesenses.com/index.php?title=JShawn`. accessed June 2012.

[47] "JiST/SWANS – Java in Simulation Time / Scalable Wireless Ad hoc Network Simulator." `http://jist.ece.cornell.edu`. accessed June 2012.

[48] O. C. Ghica, "SIDnet-SWANS Manual," March 3 2010.

[49] "Sinalgo Documentation." `http://disco.ethz.ch/projects/sinalgo/tutorial/Documentation.html`. accessed June 2012.

[50] F. G. Marmol, "Implementing and Integrating a new Trust and/or Reputation Model in TRMSim-WSN," 8 June 2009.

[51] F. G. Marmol and G. M. Perez, "TRMSim-WSN, Trust and Reputation Models Simulator for Wireless Sensor Networks," in *Proceedings of IEEE ICC 2009, IEEE International Conference on Communications*, (Dresden, Germany), 14–18 June 2009.

[52] A. Marchiori, L. Guo, J. Thomas, and Q. Han, "Realistic Performance Analysis of WSN Protocols Through Trace Based Simulation," in *The Seventh ACM Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks Conference (PE-WASUN)*, (Bodrum, Turkey), 17–21 October 2010.

[53] "SimPy Overview." http://simpy.sourceforge.net/simpy_overview.htm. accessed June 2012.

[54] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection Tree Protocol," in *Proceedings of SenSys'09, 7th ACM Conference on Embedded Networked Sensor Systems*, (Berkeley, CA, USA), November 2009.

[55] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with COOJA," in *Proceedings of SenseApp 2006, First IEEE International Workshop on Practical Issues in Building Sensor Network Applications*, (Tampa, USA), 12–16 November 2006.

[56] "Instant Contiki." http://www.contiki-os.org/p/instant-contiki.html. accessed June 2012.

[57] T. Voigt, J. Eriksson, F. Österlind, R. Sauter, N. Aschenbruck, P. J. Marrón, V. Reynolds, O. V. L. Shu, A. Koubaa, and A. Köpke, "Towards Comparable Simulations of Cooperating Objects and Wireless Sensor Networks," in *Proceedings of WSNperf, 1st International Workshop on Performance Methodologies and Tools for Wireless Sensor Networks*, (Pisa, Italy), 23 October 2009.

[58] A. Sobeih, W.-P. Chen, J. C. Hou, L.-C. Kung, N. Li, H. Lim, H.-Y. Tyan, and H. Zhang, "J-Sim: A Simulation and Emulation Environment for Wireless Sensor Networks," *J-Sim documentation*. accessed June 2012.

[59] S. Sundresh, W. Kim, and G. Agha, "SENS: A Sensor, Environment and Network Simulator," in *Proceedings of ANSS37, 37th Annual Simulation Symposium*, (Arlington, USA), 21 April 2004.

[60] "WirelessHART Overview." http://www.hartcomm.org/protocol/wihart/wireless_overview.html. accessed June 2012.

[61] "NS-2 (The Network Simulator)." http://nsnam.isi.edu/nsnam/. accessed June 2012.

[62] "NS-2 Manual, 20. Directed Diffusion." http://www.isi.edu/nsnam/ns/doc/node227.html. accessed June 2012.

[63] "Using Mannasim Framework." http://www.mannasim.dcc.ufmg.br. accessed June 2012.

[64] I. Downard, "SIMULATING SENSOR NETWORKS IN NS-2," in *NRL/FR/5522–04-10073*, (Naval Research Laboratory, Washington, USA), May 2004.

[65] "Nam: Network Animator." http://www.isi.edu/nsnam/nam. accessed June 2012.

[66] "RTSIM – Real-Time system SIMulator." http://rtsim.sssup.it. accessed June 2012.

[67] P. Pagano, M. Chitnis, and G. Lipari, "Real-time applications in Wireless Sensor Networks," 1 December 2007.

[68] "OMNeT++ (Objective Modular Network Test-bed in C++)." http://www.omnetpp.org. accessed June 2012.

[69] A. Boulis, "Castalia – A simulator for Wireless Sensor Networks and Body Area Networks," *User's Manual*, October 2009. accessed June 2012.

[70] A. Köpke, M. Swigulski, K. Wessel, D. Willkomm, P. K. Haneveld, T. Parker, O. Visser, H. S. Lichte, and S. Valentin, "Simulating Wireless and Mobile Networks in OMNeT++: The MiXiM Vision," in *Proceedings of OMNeT++ 2009, 1st International Workshop on OMNeT++*, (Marseille, France), 7 March 2008.

[71] "NesCT." http://nesct.sourceforge.net. accessed June 2012.

[72] J. Glaser, D. Weber, S. A. Madani, and S. Mahlknecht, "Power Aware Simulation Framework for Wireless Sensor Networks and Nodes," *EURASIP Journal on Embedded Systems*.

[73] D. Weber, J. Glaser, and S. Mahlknecht, "Discrete Event Simulation Framework for Power Aware Wireless Sensor Networks," in *Proceedings of INDIN 2007, 5th International Conference on Industrial Informatics*, vol. 1, (Vienna, Austria), pp. 335–340, 23–26 July 2007.

[74] "The Programming Language Lua." http://www.lua.org. accessed June 2012.

[75] C. Mallanda, A. Suri, V. Kunchakarra, S. S. Iyengar, R. Kannan, and A. Durresi, "Simulating Wireless Sensor Networks with OMNeT++," 24 January 2005.

[76] "Ptolemy II." http://ptolemy.berkeley.edu/ptolemyII/. accessed June 2012.

[77] E. Cheong, E. A. Lee, and Y. Zhao, "Viptos: A Graphical Development and Simulation Environment for TinyOS-based Wireless Sensor Networks," in *Technical Report No. UCB/EECS-2006-15*, EECS Department, University of California, Berkeley, USA, 15 February 2006.

[78] E. A. Lee and S. Neuendorffer, "MoML – A Modeling Markup Language in XML – Version 0.4," in *Technical Memorandum ERL/UCB M 00/12*, 14 March 2000.

[79] E. Cheong, E. A. Lee, and Y. Zhao, "Joint Modeling and Design of Wireless Networks and Sensor Node Software," in *Technical Report No. UCB/EECS-2006-150*, EECS Department, University of California, Berkeley, USA, 17 November 2006.

[80] P. Baldwin, S. Kohli, E. A. Lee, X. Liu, and Y. Zhao, "VisualSense: Visual Modeling for Wireless and Sensor Network Systems," in *Technical Memorandum UCB/ERL M05/25*, University of California, Berkeley, USA, 15 July 2005.

[81] B. Qela, G. Wainer, and H. Mouftah, "Simulation of Large Wireless Sensor Networks Using Cell-DEVS," in *Proceedings of WSC'09, Winter Simulation Conference*, (Austin, USA), 13–16 December 2009.

Authors

Bartosz Musznicki
received the M.Sci degree in Telecommunications from Poznań University of Technology, Poland, in 2010. He currently holds the position of Internet Services Department Manager at INEA S.A. Bartosz Musznicki is the author of three articles in conference proceedings, two journal papers and of one book chapter. Since 2009, he has been engaged in research in the area of routing in Wireless Sensor Networks.

Piotr Zwierzykowski
received the M.Sci and Ph.D degrees in Telecommunications from Poznań University of Technology, Poland, in 1995 and 2002, respectively. Since 1995 he has been working in the Faculty of Electronics and Telecommunications, Poznań University of Technology. He is currently Assistant Professor in the Chair of Communications and Computer Networks. Piotr Zwierzykowski is the author or co-author of over 150 articles and 3 books. He is currently engaged in research in the area of routing protocols and algorithms.