Towards a Taxonomy of Simulation Tools for Wireless Sensor Networks

Wan Du, David Navarro, Fabien Mieyeville and Frédéric Gaffiot
Lyon Institute of Nanotechnology (INL), University of Lyon
UMR5270 - CNRS, Ecole Centrale de Lyon, Ecully, F-69134, France
(33) 04 72 18 63 98
{wan.du, david.navarro, fabien.mieyeville, frederic.gaffiot}@ec-lyon.fr

ABSTRACT

Many limitations (e.g. complexity, cost, scalability and capability) make the analytical methods and physical testbeds improper to evaluate the performances of wireless sensor networks (WSNs). Simulations can provide a good approximation at lower cost and in less time. Hence, a number of simulation tools for WSN have been developed in the past few years. However, different tools may emphasize on different features. For example, besides the general network simulators, some SystemC-based simulators have been developed recently in order to realize the hardware/software (HW/SW) co-design of the node at the system-level that also takes into account its network performances. So it is necessary to study the existing WSN simulators and to distinguish their different features. In this paper, we propose a taxonomy that categorizes the existing simulation tools into four classes according to their modeling methodologies and their target applications. In order to prove that the proposed taxonomy is reasonable and comprehensive, we use it to make a survey of the existing simulation tools. This study is intended to be broad enough to cover all the important existing simulation tools. The goal of this paper is to analyze the WSN simulation tools and help the WSN designers find an appropriate simulator.

Categories and Subject Descriptors

I.6.5 [**Simulation and Modeling**]: Model Development – *Modeling Methodologies*

General Terms

Performance, Design, Experimentation

Keywords

Taxonomy, Wireless Sensor Networks, Simulation, Modeling

1. INTRODUCTION

Wireless sensor networks (WSNs) are normally ad-hoc networks composed of resource-constrained sensor nodes that can cooperatively monitor physical or environmental conditions, such as temperature, sound and acceleration. WSNs have been employed in a wide range of application domains, such as health-related deployments, environment monitoring, industry and military applications [4]. Three techniques have been used to evaluate the performances of WSN systems: analytical methods [26], physical testbeds [15] and simulation. However, many constraints imposed on sensor networks, such as limited resources, decentralized collaboration and fault tolerance, necessitate the use of complex algorithms that usually make analytical methods be impossible [3]. Additionally, although using the physical testbeds is direct, such studies also suffer some significant limitations, such as cost and scalability. It is costly and troublesome to establish a testbed for a network with thousands of nodes. And some tests may last too long to be repeated many times. Moreover, the testbed is limited by the experiment environment, and sometimes it is incapable of presenting a diverse set of operational scenarios. However, simulation can overcome these limitations mentioned above, and provide a good approximation at lower cost and in less time. It also generally provides an easy-to-use debugging environment and graphic user interfaces. So, simulation has become a common way to evaluate performances of WSN systems.

Lots of simulators for WSNs have been developed in the past few years. But different simulators may be designed to accomplish different target applications. For example, some are intended to simulate the performance of communication protocols and some may be designed to emulate the execution of the binary code. So it is important to find out their similarities and differences. Based on an elaborate study of WSN simulations and the existing simulation tools, we proposed a classification scheme that categorizes the existing simulation tools into four classes. According to the taxonomy, a comprehensive study of the existing simulation tools for WSN is made.

The rest of the paper is organized as follows. In the next section, we study the requirements of the simulation for WSN. They are used as metrics to evaluate the existing simulation tools. Section 3 presents a typical model of WSN system. Based on the analysis of the former two sections, a taxonomy is proposed in Section 4. The existing simulation tools are classified by using this taxonomy and an elaborate analysis of them is made in section 5. Section 6 concludes this paper.

2. Requirements of WSN Simulation

WSN is a unique network in the following aspects: restricted resources (memory, power, and processing ability), big quantity of nodes, decentralized collaboration, multitasking, heterogeneity and so on. These features make the development of the simulation tools for WSN more challenging. By taking into account of the special characteristics of WSN and the requirements of different WSN design fields (e.g. communication protocol design, application design and node system design), we summarized the following six key requirements that are important to a WSN simulation framework:

1. Fidelity: The main purpose of the simulation is to model the real-world system faithfully and predict the system's behavior. For WSN, it requires accurate models of the radio channels, physical environment and node system. Inaccurate simulation may lead to erroneous conclusions. For example, an ideal battery model usually treats the battery as a reservoir of energy from which the energy consumption can be subtracted. However, this is not accurate as a real battery that shows non-linear discharge behavior and recovery effects. It is proved that the accuracy of battery models can affect the route fluctuations and routing overheads [38].
2. Scalability: Because the nodes are often deployed in large quantities in many WSN applications, the simulator shall well support the scalability. The simulation time and memory usage shall not augment too much as the number of nodes increases.
3. Energy aware: Power consumption is especially critical for sensor networks operating on limited power supply, such as batteries or solar cells. Sensor network designers need to obtain accurate power consumption and timing figures to tune their applications before the deployment in real environments [33]. Therefore, the simulator shall be able to accurately capture the energy consumption and timing information of the embedded software and radio communication at the network level.
4. Extensibility: It shall be easy to modify the existing modules or integrate some new ones. A careful structure with clean interfaces and high modularity allows the users to easily add or change functionality.
5. Heterogeneity Support: Many recently deployed WSN systems are heterogeneous systems, incorporating a mixture of elements with widely varying capabilities [14]. So, modeling different kinds of nodes and managing the interconnections among them are necessary in WSN simulations.
6. Graphical user interface (GUI): A good GUI can facilitate and speed the establishment of the network topology and the composition of basic modules. It can also allow the quick visualization of the simulation results. In addition, it supports to trace and debug the simulation at real time. Non-specialist users can get an easier control of the simulation by using GUI.

There is always a tradeoff between fidelity and scalability [8]. Better fidelity involves more complex and detailed modeling. However, the simulators need more time to deal with the additive detail. The simulation time may become intolerable if the number of nodes is very large in some WSN applications. So, the high level abstraction is sometimes more suitable for implementing the simulation with proper complexity and little running time, and their results are detailed enough to answer the design questions at the early stages of the design flow. For example, at the beginning of a system design, the need to quickly explore a variety of alternatives is more important than a detailed result for a specific scenario. The challenge is to identify which level of detail does not affect answers to the design questions at hand. In [38] [16], the authors have explored this question. Many cases have been studied.

In addition, many methods have been used to deal with the scalability problem, e.g. component-based design and parallel simulation. In a parallel simulator, the simulated components are dispatched over several CPUs, where the sub-programs are concurrently executed, and the simulation algorithm is responsible for the synchronization.

3. A Typical Model of WSN System

WSN mainly involves three parts: node system, network and physical environment. A typical model of WSN system is presented in Figure 1. In this model, the node system is composed by two parts: hardware and software. The hardware platform consists of processing unit, radiofrequency transceiver, sensor and battery. The software model includes operating system, middleware, protocol stack, application software implementation and so on. Nodes are connected with each other by the wireless network model that holds the network topology and transfers packets among nodes. It also implements many radiofrequency channel models. The environment model specifies how the physical parameters in the environment vary both in spatial and temporal sense. Environment modeling of WSNs is still at the beginning of development. A more detailed description of environment modeling can be found in [23].

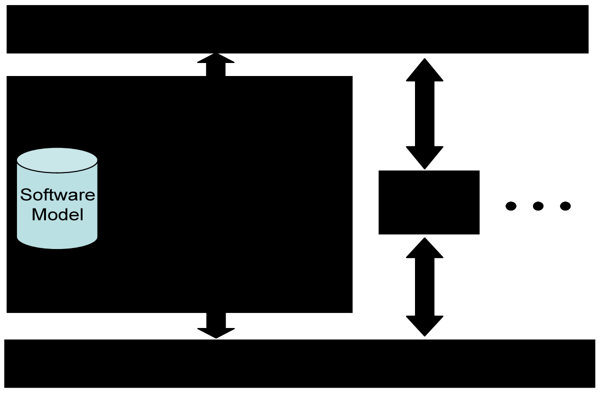

Figure 1. A Typical Model of WSN System

Since only few simulators have addressed environment modeling well, our taxonomy will not treat it as a determinant. We mainly focus on the node system and network modeling. Simulation has been used in both node system and protocol designs to help the designer easily evaluate their new designs. At the beginning, these two aspects are addressed by different people with different knowledge and tools. In the context of node system design, the aim is to design the nodes' hardware, to implement the software running on the hardware and to co-simulate the hardware and software (HW/SW) [13] [34]. In the context of protocol design, the tools model the protocols, manage the concurrency among different nodes, and simulate the throughput of the network. The protocol designers often make simple assumptions to the behavior of hardware and software, but this may be not detailed enough for some applications. For example, timing information in instruction granularity shall be considered to the fast routing lookups [6]. In addition, it is better to compress the data by processing them in local CPU rather than transmitting the raw data to the destination node in some applications, since wireless communication is a major energy consumer during the system operation [32].

Simulations shall be able to help the designer find a balance between the wireless communication and the local processing. Therefore, WSN simulations require designers to integrate the node system and the network simulation together.

4. Taxonomy

A common way to evaluate the WSN system is to add node models to the network simulators (e.g. NS-2 [22] and OMNeT++ [39]). There are two kinds of node model: node models implemented by the network simulators and node emulators. Node emulators refer to the instruction level simulators of the nodes' microcontrollers with extensions of sensors, transceivers and other peripherals models.

Besides adding node models to the network simulators, we can also model the network in the node system design tools (e.g. SCNSL [11]) or in the node emulator (e.g. Avrora [35]). Therefore, the existing efforts in WSN evaluation can be divided into four categories: network simulators with node models (NSNM), network simulators with node emulators (NSNE), node system simulator with network models (NSSNM) and node emulators with network models (NENM).

NSNM emphasizes more on discrete event scheduling, the radio medium, network modeling and perhaps the sleep duty cycles of the sensor node. Network modeling is the predominate object. Many node models implemented by NSNM are simple power and estimated timing profiles.

NSNE integrates the advantages of both the network simulators and node emulators. The network simulator provides the detailed network model. The node emulator gives accurate timing information of the software execution because they simulate the system performance with instruction cycle granularity. But the interconnection between the network simulator and node emulator may takes much time.

In NSSNM, the node system is often modeled by the hardware description languages at system level, such as SystemC [19]. NSSNM has a simulation kernel which supports modeling the concurrency and synchronization among different hardware components. The system level description language can also model the software, which allows the HW/SW co-design and co-simulation. It models the node hardware in different abstraction level with different degrees of detail (e.g. system level, transaction level and register transfer level). The level at which the simulation is performed affects the level at which the software development can occur and the execution efficiency of the simulator. A simulator that simulates a particular sensor node platform at the very low level enables the development of low level software such as device drivers but at the price of longer simulation time.

The node emulators of NENM can be divided into two different sets: instruction set simulators (ISS) for special microcontrollers and emulators designed to emulate the execution of the application code of an operating system (e.g. TinyOS [17], SOS [18] and Contiki [5]). They can execute the application code directly (or with minor modifications). Generally, the network models in NSSNM or NENM provide less detail than the network simulator, since the latter always includes lots of protocol implementations and channel models.

Our taxonomy can be used by WSN designers to find a proper simulation tool. To do that, first, WSN designers must decide whether they emphasize particularly on communication protocol design or node system design. The main features of these two different designs have been presented in Section 3. Then, the application or communication protocol must be carefully analyzed to limit the level of detail in an appropriate rank. It is a challenging work especially when entering a relatively unexplored area. The results of existing simulation validations can help the users to build an understanding of what details are important, such as the case studies in [38] [16]. Based on the level of detail, the developers must decide whether they need a system-level simulator or an instruction-level emulator of nodes. Generally, node simulators are always at the high level of abstraction with limited accuracy and less simulation time. Emulators can emulate the execution of the application software, which provides more accurate timing information. Until now, the category of the simulator has been chosen. The simulators within the same category shall be evaluated according the requirements of WSN simulation listed in section 2. They are compared in the aspect of scalability, heterogeneity support, extensibility, radio propagation models, power models, easy to use and others.

5. Survey

In this section, we will analyze the existing simulation tools according to the taxonomy. The existing simulation tools are divided to 4 classes. In each category, many simulators will be studied to demonstrate their common features and differences.

5.1 NSNM

Lots of simulators have been designed in this category. We mainly study some typical ones to illustrate the main characteristics of this kind of simulators

NS-2 [22] is a discrete event, object-oriented, general purpose network simulator. Simulations are written by C++ and OTcl (Object-oriented Tcl) languages. In general, C++ is used for implementing protocols and extending the NS-2 library. OTcl is used to create and control the simulation environment. Its extensibility has been a major contributor to its success, with protocol implementations being widely produced and developed by the research community. According to [20], it is the most used simulator in Mobile Ad hoc NETwork (MANET) research. Regarding WSN, it includes many ad-hoc and WSN specific protocols [8]. An IEEE 802.15.4 model is developed in [45]. However, NS-2 does not scale well in terms of memory usage and simulation time [24]. It also lacks detailed support to measure the energy utilization of different hardware, software, and firmware components of a WSN node [43]. SensorSim [27] is built on top of an NS-2 802.11 network model. It models the sensor node in two parts: software model (Function Model) and hardware model. The power models of different hardware components have been implemented. The state of the hardware model is changed based on the function that is carried out by the software model. So, the power consumption of the whole network can be simulated. In addition, SensorSim can be interacted with real nodes. The use of real sensor nodes can provide accurate and valid inputs to the simulation instead of modeling the sensor channel that is not yet well defined and understood. SensorSim also provide the underlying network on which we can develop, test and evaluate the SensorWare [1]. SensorWare is a middleware that allows easy, efficient dynamic programmability for sensor network. The execution environment in a sensor node is closely modeled in SensorSim so that the SensorWare scripts can run on both the real nodes as well as the simulated nodes. However, the CPU and sensor device models have not been implemented and the simulator is no longer in active development. Furthermore, IEEE 802.11 is designed for high speed connectivity and not optimized for WSN.

OMNeT++ [39] is a component-based network simulator, with an Eclipse-based IDE and a graphical runtime environment. The IDE supports all stages of a simulation project: developing, building, configuring, running simulation models and analysing results. OMNeT++ consists of modules that communicate with message passing. Simple modules implement the atomic behavior of a model, e.g. a particular protocol. Multiple simple modules can be linked together and form a compound module. OMNeT++ provides the infrastructure to assemble simulations from these modules and configure them (*NED language*). OMNeT++ can be extended easily by interfaces for real-time simulation, emulation, parallel distributed simulation, SystemC integration and so on. As OMNeT++ is becoming more popular, many contributions have been added to it. The Mobility Framework (MF) [7] supports simulations of wireless and mobile networks within OMNeT++. MF includes an 802.11 model. It can be seen as the first start point of the WSN modeling by OMNeT++. An IEEE 802.15.4 implementation by OMNeT++ can be found in [2]. PAWiS [43] is an OMNeT++ based WSN simulator. Its architecture is similar to SensorSim. It can evaluate the power consumption of WSN systems with many levels of accuracy which can still be balanced with complexity. The model programmer has to insert special framework requests to the CPU module to simulate the execution time and power consumption. These requests include the estimated execution time of the firmware code on the CPU.

SENSE [3] is another component-based simulator developed by C++. It models various network devices as a collection of static components. Connections between each component are in the format of in and out ports. Dynamic packets are created, transmitted and received by components through the ports. Through its component-based model, SENSE can be extended easily. A new component can replace an old one if they have compatible interfaces; inheritance is not required. SENSE also supports the parallel simulation, which is provided as an option to the users.

GloMoSim [44] is a parallel simulator for WSNs. GloMoSim allows the users to select sequential or one of the 3 available parallel synchronization algorithms (null message protocol, conditional event protocol and accelerated null message protocol). Once a parallel algorithm is selected, the analyst can additionally indicate the mapping strategy and number of processors. Taking advantage of parallel simulation, GloMoSim has been shown to scale to 10 000 nodes [36]. QualNet [31] is a commercial derivative of GloMoSim. It has extended GloMoSim to other networks, such as satellite, cellular and sensor networks. ZigBee protocol model is provided too.

Prowler [28] is an event-driven network simulator running in Matlab environment. Benefits gained from Matlab environment are easy prototyping of applications and GUI interface. Prowler is capable of simulating the radio transmission, propagation and the MAC-layer operation in ad hoc networks. The radio models are based on specific signal strength calculations combined with random errors. Prowler is well suited for protocol and algorithm development. However, it does not have sensor node energy modeling.

The main advantages of these simulators are that they usually have a rich library of the radio modules and protocol implementations. Many contributions to these tools are carried out ceaselessly. For example, NS-2 is currently undergoing a major redesign [42]. The performances in the aspects of scalability and extensibility will be improved by its successor, NS-3. However, the network simulator is dedicated to model the network. It may be not the best way to model the node system since they are normally incapable to model the concurrency within the node and provide a direct path to HW/SW synthesis [11]. The energy consumption is usually based on some assumptions or estimations of the software execution, which is not as accurate as the node emulator.

5.2 NSNE

Two main simulators have been developed in this category. Heemin Park et al. [29] have developed a unified network and node level simulation framework. They developed the Embedded Systems Power Simulator (ESyPS) by integrating sensor and radio modules into EMSIM [37]. EMSIM is an energy simulation framework for embedded systems featuring in StrongARM microprocessor and Linux OS. Then, they integrated the ESyPS with SensorSim [27]. The framework can explore the interactions between network level and node level.

Another example is sQualNet [40], which is a scalable and extensible sensor network simulation framework built on top of QualNet [31]. It uses QualNet as the network simulator and provides the emulation of the SOS operating system [18]. sQualNet allows using the QualNet's detailed models of channel, propagation, mobility, etc. The user also can use the rich protocol suite for other kinds of networks to model heterogeneous sensor networks. sQualNet introduces a sensor stack parallel to the networking stack and provides accurate simulation models for various layers in the sensor and networking stack.

These two simulators integrate the advantages of both the network simulators and node emulators. They provide accurate results about the energy consumption of the whole network. However, they are both constrained to particular hardware and operating system. Moreover, interactions between the network simulator and the node emulator have to be well maintained, which increases the simulation time and impacts the scalability.

5.3 NSSNM

There are mainly two simulators in this kind. Kashif Virk et al. [41] have developed a SystemC-based modeling framework for WSN. It enables system-level modeling of sensor network behaviors by modeling the applications, real-time operating system, sensors, processor, and radiofrequency transceiver at the node level and the signal propagation at the network level. But the simulation result is simple. Only a MAC behavior (states of the sending and receiving tasks) simulation result has been presented.

The SystemC Network Simulation Library (SCNSL) [11] is a Networked Embedded Systems simulator, written in SystemC and C++. Because SystemC is a C++ class library, it has the advantage that the hardware, software, and network can be modeled with a same language. SCNSL contains three modules: node, node-proxy and network. The network module is written in C++. The node-proxy and the node modules are written in SystemC. The node-proxy can access the network instance. It is the interface between the nodes and the network. By using Node-Proxy, nodes can be designed as pure SystemC modules without object references to other non-SystemC classes; this approach enables the use of all the advantages of SystemC in HW/SW co-design, verification and the design flow. There are two limitations of that library. One is that only an IEEE 802.15.4 non-beacon model communication protocol has been implemented. The other is that important network simulation results are not given in [11].

These simulators usually model WSN at the system level, so they scale well. New hardware and software modules can be easily added to the existing library. However, the simulation results only can be used to the system level design, and no GUI has been provided by these two simulators.

5.4 NENM

Two kinds of node emulator, operating system emulator and instruction set simulator, are studied separately in this section. One special simulator providing the both features will be presented too.

TOSSIM [21] and PowerTOSSIM [33] are two emulators designed to emulate the execution of TinyOS [17]. Software development for WSN can be simplified by using these emulators. They permit developing algorithms, studying system behaviors and observing interactions among the nodes in a controlled environment. The application code of TinyOS can be compiled to the simulation framework by only replacing a few low-level TinyOS components that deal with hardware. TOSSIM can capture the behavior of the network of thousands of TinyOS nodes at bit granularity. TOSSIM allows developer to easily transition between running an application on motes and in the simulation environment. PowerTOSSIM is an extension to TOSSIM in evaluation of the power consumption. The main problem with such frameworks is that the user is tied to a single platform (typically MICA motes) and a single programming language (typically TinyOS/NesC) [43]. In addition, TOSSIM loses the fine-grained timing and interrupt properties of the code that can be important when the application runs on the hardware and interacts with other nodes [35].

ATEMU [30] is an instruction-level cycle-accurate emulator for WSN written in C. It simulates programs of each individual node with accuracy down to the clock cycle. Its core is an ISS. Along with support for the AVR processor, it also includes support for other peripheral devices on the MICA2 sensor node platform, such as the transceiver. ATEMU provides a GUI, called Xatdb, which provides users a complete system for debugging and monitoring the execution of their code. Avrora [35], written in Java, improves the performance of ATEMU in the scalability aspect. Avrora can scale to networks of up to 10000 nodes. Both ATEMU and Avrora provide the highest behavioral and timing accuracy of the WSN programs. Moreover, they are both language and operating system independent. The main disadvantage of such frameworks is that they only support systems based on components that have already existed, e.g. memories and processors, like MICA motes. Unfortunately they do not cover systems containing new hardware blocks. And they are not implemented by HW description languages, so they can not use the typical HW design flow and verification tools.

Fummi, F. et al. [12] have developed an energy-aware simulator by integrating an ISS of node's microcontroller and a functional SystemC model of the network module on SCNSL [11]. They used the µCsim as the ISS for the Intel 8051 microcontroller of the Texas Instruments CC2430–F128 chip. Using ISS makes it possible to run the exact binary embedded software on the simulated hardware platform. The SystemC kernel is modified to communicate with the ISS through inter-process communication primitives (e.g. a socket or shared memory). Instruction cycle level node emulator can provide more accurate information of the software than the node model does. But it is slow because it deals with much detail at the cycle-accurate level, and the inter-process communication between ISS and the SystemC kernel takes much time.

COOJA [25] is a Java-based simulator that provides both the operating system emulation and the instruction set emulation in a single framework. The Contiki operating system [5] can be compiled to the simulation framework. It executes native code by making Java Native Interface (JNI) calls from the Java environment to a compiled Contiki system. MSPSim [9] is used as the instruction set simulator in the COOJA. MSPSim is also written in Java. It supports the Texas Instruments MSP430 microcontroller and includes some hardware peripherals such as sensor, communication ports, LEDs, and sound devices. Recently, the COOJA/MSPSim platform [10] has been extended to support the TinyOS, so the interoperability testing of nodes with different operating systems is realized.

The main advantage of using such tools is that the code used for emulation can also run on the real node with no or minor modifications, which reduces the effort to rewrite the code. And they often provide detailed information about resource utilization. The main problems are that they are always constrained to certain hardware platforms or operating systems, and it is difficult for them to support heterogeneous networks. They can not scale as well as the node system models at system level.

6. Conclusion

In this paper, the particular requirements of the WSN simulation were studied. A typical WSN system model was presented. Based on these, a taxonomy of WSN simulations was proposed, and a survey of the existing simulation tools for WSN was made according to the taxonomy. Most of the significant existing simulation tools with relatively widespread uses have been studied. We believe that the survey is broad enough to prove that almost all the simulation tools for WSN can be divided into one of the four categories in our classification scheme. This paper can be used by WSN designers to find an appropriate simulator to their special requirements.

7. REFERENCES

[1] Boulis, A., Han, C. C., Shea, R., and Srivastava, M. B. 2007. SensorWare: Programming sensor networks beyond code update and querying. Pervasive and Mobile Computing (Aug. 2007), 386-412.

[2] Chen, F., Wang, N., German, R., and Dressler, F. 2008. Performance Evaluation of IEEE 802.15.4 LR-WPAN for Industrial Applications. In Proceedings of the 5th IEEE/IFIP Conference on Wireless On demand Network Systems and Services (Garmisch-Partenkirchen, Germany, January 2008). 89-96.

[3] Chen, G., Branch, J., Pflug, M. J., Zhu, L., and Szymanski, B. 2004. SENSE: A Wireless Sensor Network Simulator. Advances in Pervasive Computing & Networking (2004), 249-267.

[4] Diana B., Miguel G., Jaime L., and Petre D. 2009. Real Deployments of Wireless Sensor Networks. In Proceedings of the 3rd International Conference on Sensor Technologies and Applications (Athens/Glyfada, Greece, June 18-23, 2009), 415-423.

[5] Dunkels, A., Gronvall, B., and Voigt, T. 2007. Contiki – a lightweight and flexible operating system for tiny networked sensors. In Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (Washington, DC, USA, 2007), 455-462.

[6] Degermark, M., Brodnik, A., Carlsson, S., and Pink, S. 1997. Small forwarding tables for fast routing lookups. In Proceedings of the ACM SIGCOMM Conference (Cannes, France, September 1997), 3–14.

[7] Drytkiewicz, W., Sroka, S., Handziski, V., Koepke, A., and Karl, H. 2003. A mobility framework for OMNeT++. In Proceedings of the 3rd International OMNeT++ Workshop (Budapest, Hungary, 2003).

[8] Egea-López, E., Vales-Alonso, J., Martínez-Sala, A., Pavón-Marñio, P., and García-Haro, J. 2005. Simulation tools for wireless sensor networks. In Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems (Philadelphia, Pa, USA, 2005).

[9] Eriksson, J., Dunkels, A., Finne, N., Österlind, F., Voigt, T. and Tsiftes, N. 2008. Demo abstract: MSPsim - an extensible simulator for MSP430-equipped sensor boards. In Proceedings of the 5th European Conference on Wireless Sensor Networks (Bologna, Italy, 2008).

[10] Eriksson, J., Österlind, F., Finne, N., Tsiftes, N., Dunkels, A., Voigt, T., Sauter, R., and Marrón, P. J. COOJA/MSPSim: Interoperability Testing for Wireless Sensor Networks. In Proceedings of the 2nd International Conference on Simulation Tools and Techniques (Rome, Italy, 2009).

[11] Fummi, F., Quaglia, D., and Stefanni, F. 2008. A SystemC-based Framework for Modeling and Simulation of Networked Embedded Systems, In Proceedings of the Forum on Specification and Design Languages (Stuttgart, Germany, 2008), 49-54.

[12] Fummi, F., Perbellini, G., Quaglia, D., and Acquaviva. A. 2009. Flexible energy-aware simulation of heterogeneous wireless sensor networks. Design, Automation & Test in Europe (2009), 1638–1643.

[13] Fummi, F., Loghi, M., Perbellini, G., and Poncino, M., SystemC co-simulation for core-based embedded systems, Design Automation For Embedded Systems (2007), vol. 11, 141-166.

[14] Girod, L., Stathopoulos, T., Ramanathan, N., Elson, J., Osterweil, E., Schoellhammer, T., and Estrin, D. 2004. A system for simulation, emulation, and deployment of heterogeneous sensor networks. In Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (Baltimore, USA, 2004), 201–213.

[15] Handziski, V., Köpke, A., Willig, A., and Wolisz, A. 2006. TWIST: A Scalable and Reconfigurable Testbed for Wireless Indoor Experiments with Sensor Networks. In Proceedings of the 2nd International Symposium on Mobile Ad Hoc Networking and Computing (Florence, Italy, 2006), 63-70.

[16] Heidemann, J., Bulusu, N., Elson, J., Intanagonwiwat, C., Lan, K., Xu, Y., Ye, W., Estrin, D., and Govindan, R. 2001. Effects of detail in wireless network simulation. In Proceedings of the SCS Multiconference on Distributed Simulation (Jan. 2001). 3-11

[17] Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D. and Pister, K. 2000. System architecture directions for networked sensors. ACM SIGPLAN Notices, v.35 n.11, p.93-104, Nov. 2000. [doi>10.1145/356989.356998]

[18] Han, C. C., Kumar, R., Shea, R., Kohler, E., and Srivastava, M. 2005. A dynamic operating system for sensor nodes. In Proceedings of the 3rd international conference on Mobile systems, applications, and services (New York, NY, USA, 2005). 163-176.

[19] IEEE Std 1666 - 2005 IEEE Standard SystemC Language Reference Manual. IEEE Std 1666-2005, pages 1–423, 2006.

[20] Kurkowski, S., Camp, T., and Colagrosso, M. 2005. MANET simulation studies: the incredibles. Mobile Computing and Communications Review (2005), vol. 9, no. 4, 50–61.

[21] Levis, P., Lee, N., Welsh, M., and Culler, D. 2003. Tossim: Accurate and scalable simulation of entire tinyos applications. In Proceedings of the ACM Conference on Embedded Networked Sensor Systems (New York, NY, USA, 2003), 126-137.

[22] McCanne, S., and Floyd, S., “Network simulator NS-2”, http://ww.isi.edu/nsnam/ns.

[23] Merrett, G. V., White, N., Harris, N., and Al-Hashimi, B. 2009. Energy-Aware Simulation for Wireless Sensor Networks. In Proceedings of the 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (Rome, Italy, June 2009).

[24] Naoumov, V., and Gross, T. 2003. Simulation of large ad hoc networks. In Proceedings of the 6th ACM International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems (San Diego, Calif, USA, September 2003), 50–57.

[25] Österlind, F., Dunkels, A., Eriksson, J., Finne, N. and Voigt, T. 2006. Cross-level sensor network simulation with COOJA. In Proceedings of the 1st IEEE International Workshop on Practical Issues in Building Sensor Network Applications (Tampa, Florida, USA, 2006).

[26] Prasad, V., and Son, S. H. 2007. Classification of Analysis Techniques for Wireless Sensor Networks. In Proceedings of the 4th International Conference Networked Sensing Systems (Braunschweig , Germany, 2007), 93-97.

[27] Park, S., Savvides, A., and Srivastava, M. B. 2000. SensorSim: A Simulation Framework for Sensor Networks. In Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems (New York, USA, 2000), 104-111.

[28] Prowler network simulator, Available from: http://www.isis.vanderbilt.edu/Projects/nest/prowler, visited at September 20. 2009.

[29] Park, H., Liao, W., Tam, K. H., Srivastava, M. B., and He, L. 2003. A unified network and node level simulation framework for wireless sensor networks. Technical Report. University of California, Los Angeles

[30] Polley, J., Blazakis, D., McGee, J., Rusk, D., Baras, J. S., and Karir, M. 2004. Atemu: A fine-grained sensor network simulator. In Proceedings of the 1st Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (Santa Clara, CA, USA, 2004), 145-152.

[31] QualNet, http://www.scalable-networks.com/, visited at September 20. 2009.

[32] Raghunathan, V., Schurgers, C., Park, S., and Srivastava, M.B. 2002. Energy-Aware Wireless Microsensor Networks, IEEE Signal Processing Magazine (March 2002), vol.19, no.2, 40-50.

[33] Shnayder, V., Hempstead, M., Chen, B., Allen, G. W., and Welsh, M. 2004. Simulating the power consumption of large scale sensor network applications. In Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (Baltimore, USA, 2004), 188–200.

[34] Semeria, L., and Ghosh, A. 2000. Methodology for Hardware / Software Co-verification in C/C++. In Proceedings of the IEEE Asian and South Pacific Design Automation Conference (Yokohama, Japan, 2000), 405–408.

[35] Titzer, B. L., Lee, D. K., and Palsberg, J. 2005. Avrora: Scalable sensor network simulation with precise timing. In Proceedings of the 4th international symposium on Information processing in sensor networks (Los Angeles, California, USA, 2005), 477-482.

[36] Takai, M., Bagrodia, R., Tang, K., and Gerla, M. 2001. Efficient Wireless Networks Simulations with Detailed Propagations Models. *Wireless Networks* (2001), 297–305.

[37] Tan, T. K., Raghunathan, A., and Jha, N. K. 2002. EMSIM: an energy simulation framework for an embedded operating system. In Proceedings of the IEEE International Symposium on Circuits and Systems (Phoenix-Scottsdale, AZ, USA, 2002), volume 2, 464-467.

[38] Varshney, M., and Bagrodia, R. 2004. Detailed models for sensor network simulations and their impact on network performance. In Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems (Venice, Italy, 2004), 70-77.

[39] Varga, A. 2001. The OMNeT++ discrete event simulation system. In Proceedings of the 15th European Simulation Multiconference (Prague, Czech Republic, 2001). 319–324.

[40] Varshney, M., Xu, D. F., Srivastava, M. B., and Bagrodia, R. L. 2007. sQualNet: An Accurate and Scalable Evaluation Framework for Sensor Networks. In Proceedings of the International Conference on Information Processing in Sensor Networks (Cambridge, Massachusetts, April 25-27, 2007).

[41] Virk, K.   Hansen, K. and Madsen, J. 2005. System-level Modeling of Wireless Integrated Sensor Networks. In Proceedings of the International Symposium on System-on-Chip (Tampere, Finland, 2005), 179-182.

[42] Weingartner, E., vom Lehn, H., and Wehrle, K. 2009. A performance comparison of recent network simulators. In Proceedings of the IEEE International Conference on Communications (Dresden, Germany, 2009), 1-5.

[43] Weber, D., Glaser, J., and Mahlknecht, S. 2007. Discrete Event Simulation Framework for Power Aware Wireless Sensor Networks. In Proceedings of the 5th International Conference on Industrial Informatics (Vienna, Austria, 2007), 335-340.

[44] Zeng, X., Bagrodia, R., and Gerla, M. 1998. GloMoSim: a Library for Parallel Simulation of Large-scale Wireless Networks, In Proceedings of the Workshop on Parallel and Distributed Simulation (Banff, Alberta, Canada, 1998), 154-161.

[45] Zheng, J., and Lee, M. J. 2006. A comprehensive performance study of IEEE 802.15.4. Sensor Network Operations. IEEE Press, Wiley Interscience, Chapter 4, 218-237.