Simulation Domains for Networked Embedded Systems

Hugo V. Bitencourt, Adriano B. da Cunha,Diogenes C. da Silva
Federal University of Minas Gerais
Av. Antônio Carlos 6627, 31270-901, Belo Horizonte, MG, Brazil
hugov@cpdee.ufmg.br, adborges@ufmg.br, diogenes@cpdee.ufmg.br

ABSTRACT

Networked embedded systems are a fast growing application for embedded systems. Choices taken during the system design can influence the network configuration and vice versa. It poses new challenges in design and simulation domains. It is very important at the early stages of the design of networked embedded systems to simulate both embedded systems and networked environment in which they operate. Besides, due to limited available energy, simulate the energy consumption in networked embedded systems is also almost imperative. In this paper, we propose a new view for design and analysis of simulator for networked embedded systems defining four domains to capture and describe networked embedded systems with purpose of simulation. It is also presented an elaborate comparison and analysis of four simulators designed to networked embedded systems.

Categories and Subject Descriptors

I.6.1 [**Simulation and Modeling**]: Simulation Theory—*Model classification*; C.3 [**Special-Purpose and Application-Based Systems**]: Real-time and embedded systems

General Terms

Design, Theory

Keywords

Embedded systems, network, environment, energy, simulators, domains

1. INTRODUCTION

With the goal to instrument the physical world with pervasive networks that are able to sense, process and interact with their surrounding environment, networked embedded systems (NES) have been emerged. However, this poses a set of challenges that should be address. First, the lifetime of the application. Second, the limited available energy. Third, a network formed by embedded systems need to cover larger areas, but the communication is provided via short distance radios, as a result, the communication is the most expensive operation [10].

Choices taken into during the system design can influence the network configuration and vice versa. It is very important at the early stages of the design of NES to simulate not only the embedded system, but also the communication network and the surrounding environment in which it operates [11]. Besides, due to limited available energy, simulate the energy consumption in NES is also almost mandatory.

Simulators for NES should address the computing platform, the communication network, the surrounding environment, and energy available and necessary to make the NES operational. Figure 1 illustrates these different domains.

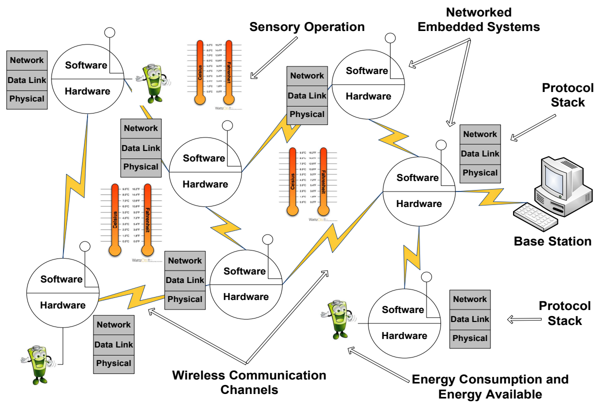

Figure 1: Different domains that simulators for NES should address

In this paper, we propose a new view for design and analysis of simulators for NES. We define four domains to capture and describe NES with purpose of simulation. The proposed domains are the following: embedded system, network, environment and energy. In order to show that the proposed domains are useful and reasonable, we conduct an elaborate comparison and analysis of four simulators designed for NES in each proposed domain.

The rest of this paper is organized as follows. Section 2 presents the four proposed domains. Section 3 presents an elaborate analysis and comparison of four simulators for NES. Section 4 concludes this paper.

2. PROPOSED DOMAINS

This paper proposes a new view for design and analysis of simulators for NES: the computing platform, the communication network, the surrounding environment, and energy available and necessary to make the networked embedded system operational.

2.1 Embedded System Domain

The embedded system domain models the computing platform and is divided into two separated dimensions: hardware and software. Hardware can be modeled by the following approaches: HA (Hardware Abstraction), PMS (Process/Memory/Switch), ISS (Instruction Set Simulator), and RTL (Register Transfer Level). Software can be modeled by the approaches: APP (application) and OS (operating system).

HA is the hardware abstraction level where modules and subsystems are described as an executable code or simulatable code. At this level no specific hardware description is used, and no mapping to existing actual hardware is enforced. PMS is a notation where the embedded system architecture is described as a composition of modules like processors, memories, buses, and peripherals [21]. At PMS level a real hardware can be described and more details related to real implementations can be used. ISS is the level equivalent to an instruction set simulator of a processor. At this level, the hardware behavior can be described as a sequence of machine instructions. RTL is the description level of a digital system composed by registers, functional units and their interconnections. RTL is the lowest and more detailed level of abstraction that is useful for networked embedded systems designers.

APP is the description level of an application program to be executed on the embedded system, generally using NesC, C, C++, SystemC, or Java languages. This application program can be executed either on the development environment or on the embedded system. SO is the description that includes the application program and an operation system or microkernel.

2.2 Network Domain

The network domain models the network communication. It is usually modeled after the OSI model from ITU-T and is commonly known as layered protocol stack. However, protocols for traditional wireless networks cannot be applied directly to networked embedded systems without modification, since they do not take into account energy, computation and storage constraints as primary concerns.

2.3 Environment Domain

The environment domain models the ambient where the networked embedded systems are deployed and involves the communication medium and the sensory operation.

Networked embedded systems communicate through a wireless communication channels. However, this communication presents imperfections due to collisions, noise, fading, path loss, interferences, obstacles and so on. Sensory operation includes the sensed-data generation (i.e. sensing) and sensing ability (i.e. how well an area of interest is being monitored by the deployed network).

2.4 Energy Domain

Networked embedded systems are commonly battery-powered. Each mote must manage and balance its energy resources intelligently and autonomously since network lifetime depends on the amount of available energy. It is necessary that simulators for networked embedded systems provide the energy consumption and the amount of available energy for each mote and the whole network at any time [3]. The energy domain models how the embedded systems and the network perceives the energy used for their operation.

3. SELECTED SIMULATORS

The simulators selected for this paper are the following: TOSSIM 2.x, COOJA 2.5, IDEA1 and ATLeS-SN 2.0, commolly used for simulation of wireless sensor network (WSN). TOOSIM [14] and COOJA [18] were developed to simulate the behavior of an operating system for NES, TinyOS [23] and Contiki [6], respectively. IDEA1 [5] and ATLeS-SN [12] belong the new class of simulators for WSN written in SystemC language.

Recently, researchers began to design simulators using the SystemC language, since SystemC enables the simulation of embedded systems with different functionality at different abstraction levels, the communication network and the surrounding environment. The use of a single tool for system design could be an advantage for the design of NES [11]. SystemC simulators can model sensor nodes at the RTL level and can be integrated to an ISS. An ISS-SystemC integration enables the simulation of the same application code used in the real sensor node. SystemC language supports the hardware and software co-simulation of sensor nodes closer to the real implementation.

This work conducts an elaborate analysis aiming at evaluating the potentiality of SytemC and operating system simulators for NES in each simulation domain. It also presents the differences and similarities among simulators.

3.1 TOSSIM

TOSSIM models the sensor node using the OS and HA approaches. It maps directly to TinyOS code, and works by replacing hardware components by simulation wrappers using a Hardware Abstraction Layer [19][25]. However, TOSSIM only supports the MICAz motes [15].

TOSSIM simulates the behavior of the physical layer through a radio object that is based on experimental results using the TI CC2420 [2] radio. The MAC object simulates the data link layer, and can control several properties important for the CSMA protocol. Besides, TOSSIM supports routing protocols available in TinyOS [25].

Designers can provide a set of data to TOSSIM which describes the propagation strengths, and can also specify noise floor and receiver sensitivity. TOSSIM simulates the radio frequency noise and interference using the Closest Pattern Matching (CPM) algorithm [25]. However, this model is not perfect, since it does not handle correlated interference at sensor nodes that are close to one another.

TOSSIM has an extension called PowerTOSSIMz [19] that calculates the energy consumption per-node and per-component, and has a battery model that provides the amount of available energy for each sensor node.

3.2 COOJA

COOJA models the sensor nodes using the APP, OS, HA and ISS (i.e. using either MSPsim [8] or Avrora [24], since both were integrated into COOJA) approaches. COOJA can simulate Contiki applications in two ways, either by running the application as a compiled native code directly on the host CPU, or by running compiled application code in MSPSim. COOJA can also simulate a WSN without a specific sensor node or Contiki application focusing only on the network behaviour. Using MSPsim, COOJA can also simulate TinyOS applications for MSP430 motes (e.g. Telosb [20]) [9]. COOJA also uses the Avrora simulator to simulate Contiki application of AVR-based motes (e.g. MICAz).

The set of protocols supported by COOJA is equivalent to Contiki that has the following main features: TI CC2420 and TR1100 [26] radios; IEEE 802.15.4 standard [13]; uIPv4, uIPv6 and RPL [17] routing protocol.

COOJA provides four radio propagation models: Unit Disk Graph Medium (UDGM) Constant Loss; Unit Disk Graph Medium (UDGM) Distance Loss; Directed Graph Radio Medium (DGRM); Multi-path Ray-tracer Medium (MRM) [22].

UDGM Constant Loss models the transmission range as an ideal disk in which all the sensor nodes outside that disk do not receive packets while the sensor nodes within that disk receive all the packets. UDGM Distance Loss is an extension of first radio propagation model in which interferences are considered and the packets can be transmitted with SUCCESS RATIO TX probability and received with SUCCESS RATIO RX probability. DGRM specifies the transmission success ratio in an asymmetric per-link base and can define propagation delays for the links. MRM uses ray tracing technique. The receiver power is calculated using Friis formula, and obstacles are considered as attenuators. It also calculates refractions, reflections and diffractions. Furthermore, others radio propagation models can be added [18].

In COOJA, developers can use either the duty command of MSPsim or the energy profiling used in Contiki [7], but both only print the duty cycle of components of a sensor node in different activity states.

3.3 IDEA1

IDEA1 models the sensor node using the APP and PMS approaches. IDEA1 has implemented some available off-the-shell (COTS) processors and transceivers that are basic components of some COTS motes as finite state machines (FSM). IDEA1 simulates an application program written in SystemC language [5].

IDEA1 has implemented three transceivers as FSMs: TI CC2420 [2], TI CC1000 [1] and Microchip MRF24J40 [16]. There are two IEEE 802.15.4 media access algorithms implemented: unslotted CSMA-CA and slotted CSMA-CA [5].

IDEA uses SCNSL that models propagation delay, interferences, collisions and path loss, takes into account the spatial positions of sensor nodes and their on-going transmissions. Sensors are simulated as a stimuli generator [5].

IDEA1 models the energy consumption using FSMs that works as follows: during simulation, the state transition of microcontroller and transceiver is recorded and each state is associated with a given electric current consumption. Then, based on this information, IDEA1 calculates the energy consumption of each component [4].

3.4 ATLeS-SN

ATLeS-SN models the sensor node using the APP and PMS approaches. Each sensor node is composed by an App component to implement the sensor node functionality, a Sensor component for modeling the sensing unit and a NetStack component for designing the communication unit. ATLeS-SN also simulates an application program written in SystemC. However, ATLeS-SN does not have modeled COTS processors and transceivers [12].

ATLeS-SN has a NetStack component that enables designers to implement their own protocols such as MAC and routing protocols [12]. However, at this moment, ATLeSN-SN does not include standard protocols (e.g. IEEE 802.15.4).

ATLeS-SN provides a PhysChannel component that enables the modeling of various radio propagation models, from the simplest to the most complex. PhysChannel also simulates collisions. The Sensor component enables designers to model the sensors at various abstractions levels, from sensor that only reads values from an input file to specific sensors that incorporates the device driver code. However, up to now, ATLeSN-SN does not include advanced radio propagation models and advance sensed-data generation is not available [12].

ATLeSN-SN calculates the energy consumption per-node, per-component and per-state for each component of the sensor nodes. However, ATLeSN-SN does not have implemented COTS processors and transceivers.

3.5 Comparison and Results

TOSSIM and COOJA simulate the same application code used in the real embedded system, as a result, designers do not need to spend effort in rewriting the application code already developed. In contrast, IDEA1 and ATLeS-SN do not simulate the same application code used in the real embedded system.

COOJA and SystemC simulators enable the simulation of heterogeneous networks, composed by different sensor nodes and applications. SystemC simulators also allow designers to simulate a network composed by their own NES platform with different functionality or description at different abstraction levels. In contrast, TOSSIM do not allow simulation of heterogeneous networks. SystemC simulators and TOSSIM are scalable simulators while COOJA scalability is poorer than other simulators.

COOJA has implemented different radio propagation models from the simplest to the most complex, as a result, developers can test the behavior of their TinyOS and Contiki applications on different radio propagation models. In contrast, SystemC simulators only provide a very simple radio propagation model and TOSSIM wireless channel model presents imperfections.

TOSSIM is the only simulator that has implemented a battery model that provides the amount of energy available for each mote. However, it only calculates the consumption and available energy of a network composed by MICAz motes and a single TinyOS application. Besides, TOSSIM loses the fine-grained timing and interrupt properties of the code that can be important when the application runs on the hardware [24]. It is important to estimate the energy consumption. SystemC enables the modeling of embedded systems at RTL level that is closes to the real implementation. As a result, designers can test the energy consumption of the networked embedded systems in an accurate way. COOJA does not provide the consumption and available energy.

4. CONCLUSIONS

In this paper, four domains to capture and describe NES with purpose of simulation were defined. The proposed domains serves as the basis for the design of a new simulator and in choosing the best simulator for a given application. For a detailed design and simulation of NES, the simulators should address the four proposed domains.

In order to show that the proposed domains are useful, we conducted an evaluation of the potentiality of four simulators in each domain. We also presented some conclusions based in this analysis.

5. ACKNOWLEDGMENTS

This work was supported by grants from the following sponsor agencies: FAPEMIG grant TEC-PPM-00630-11, CNPq and Capes master degree scholarship.

6. REFERENCES

[1] CC1000-datasheet. Single chip very low power rf transceiver, June 2012.

[2] CC2420-datasheet. 2.4 ghz ieee 802.15.4 zigbee-ready rf transceiver, June 2012.

[3] A. B. da Cunha. *In Portuguese: Uma Abordagem para a Modelagem da Consciência de Disponibilidade Energética em Nós Sensores de Rede de Sensores Sem Fio.* PhD thesis, Universidade Federal de Minas Gerais, 2010.

[4] W. Du, F. Mieyeville, and D. Navarro. Modeling energy consumption of wireless sensor networks by systemc. In *ICSNC, 2010*, pages 94 –98, aug. 2010.

[5] W. Du, D. Navarro, F. Mieyeville, and I. O'Connor. Idea1: A validated system c-based simulator for wireless sensor networks. In *MASS, 2011*, pages 825 –830, oct. 2011.

[6] A. Dunkels, B. Gronvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *LCN, 2004*, pages 455–462, Washington, DC, USA, 2004. IEEE Computer Society.

[7] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He. Software-based on-line energy estimation for sensor nodes. In *EmNets, 2007*, pages 28–32, New York, NY, USA, 2007. ACM.

[8] J. Eriksson, A. Dunkels, F. Finne, F. Österlind, and T. Voigt. MSPSim - an extensible simulator for msp430-equipped sensor boards. In *EWSN, 2007*, jan. 2007.

[9] J. Eriksson, F. Österlind, N. Finne, N. Tsiftes, A. Dunkels, T. Voigt, R. Sauter, and P. J. Marrón. Cooja/mspsim: interoperability testing for wireless sensor networks. In *SIMUTools, 2009*, pages 27:1–27:7, ICST, Brussels, Belgium, Belgium, 2009.

[10] J. Fonseca. *Towards a Test Framework for Networked Embedded Systems.* PhD thesis, University of Copenhagen, 2009.

[11] F. Fummi, D. Quaglia, and F. Stefanni. A systemc-based framework for modeling and simulation of networked embedded systems. In *FDL, 2008*, pages 49 –54, sept. 2008.

[12] J. Hiner, A. Shenoy, R. Lysecky, S. Lysecky, and A. G. Ross. Transaction-level modeling for sensor networks using systemc. In *SUTC, 2010*, pages 197 –204, june 2010.

[13] IEEE. *Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs).* IEEE Std 802.15.4, 2006.

[14] P. Levis, N. Lee, M. Welsh, and D. Culler. Tossim: accurate and scalable simulation of entire tinyos applications. In *SenSys, 2003*, pages 126–137, New York, NY, USA, 2003. ACM.

[15] MICAz. Micaz datasheet, June 2012.

[16] MRF24J40-datasheet. Ieee 802.15.4 2.4 ghz rf transceiver, June 2012.

[17] R. F. K. J. Omprakash Gnawali, Philip Levis and D. Moss. Ctp: Collection tree protocol, June 2012.

[18] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-level sensor network simulation with cooja. In *LCN, 2006*, pages 641 –648, nov. 2006.

[19] E. Perla, A. O. Catháin, R. S. Carbajo, M. Huggard, and C. Mc Goldrick. Powertossim z: realistic energy modelling for wireless sensor network environments. In *PM2HW2N 2008*, pages 35–42, New York, NY, USA, 2008. ACM.

[20] J. Polastre, R. Szewczyk, and D. Culler. Telos: enabling ultra-low power wireless research. In *IPSN, 2005*, Piscataway, NJ, USA, 2005. IEEE Press.

[21] D. P. Siewiorek, G. Bell, and A. C. Newell. *Computer Structures: Principles and Examples.* McGraw-Hill, Inc., New York, NY, USA, 1982.

[22] M. Stehlik. *Comparison of Simulators for Wireless Sensor Networks.* PhD thesis, Masaryk University, 2011.

[23] TinyOS. Tinyos documentation wiki, June 2012.

[24] B. L. Titzer, D. K. Lee, and J. Palsberg. Avrora: scalable sensor network simulation with precise timing. In *IPSN, 2005*, Piscataway, NJ, USA, 2005. IEEE Press.

[25] TOSSIM. Tinyos documentation wiki, June 2012.

[26] TR1100-datasheet. 916.50 mhz hybrid transceiver, June 2012.