TEPAWSN: A Formal Analysis Tool for Wireless Sensor Networks

K.L. Man[1], T. Krilavičius[2], Th. Vallee[3] and H.L Leung[3]

[1]*Xi'an Jiaotong-Liverpool University (XJTLU), ka.man@xjtlu.edu.cn, China*
[2]*Vytautas Magnus University, t.krilavicius@if.vdu.lt, Lithuania*
[3]*Solari, vallee_th@yahoo.fr and sales@solari-hk.com, Honk Kong*

Correspondence should be addressed to K.L. Man (ka.man@xjtlu.edu.cn)

Abstract

Growing safety, correctness, reliability and performance requirements for Wireless Sensor Networks (WSN) have increased demand for advanced design and development techniques. Formal methods provide basic means to achieve these goals. We propose a formal language PAWSN and an umbrella tool environment TEPAWSN that combines different formal techniques for modeling, analysis and development of power aware WSNs.

1. Introduction

A wireless sensor network (WSN) consists of spatially distributed devices that monitor their environment and communicate with each other wirelessly. Application areas of WSNs range from battlefield surveillance, industrial process monitoring and control, machine health monitoring, environment and habitat monitoring, healthcare applications, home automation and traffic control. Such working conditions set high requirements on the reliability, correctness and, especially, power consumption of the devices, as well as of the whole network. Formal methods can be used as methodological means for the development of such systems in an efficient way.

Indeed, formal methods provide languages with strict semantics and syntax, corresponding techniques for the construction of models of systems under development, and verification (automatic or semi-automatic) of these models against selected requirements. As a consequence, quantitative and qualitative properties, such as required throughput or absence of deadlocks, can be checked.

Different formal methods and tools were recently applied in modeling and analysis of WSNs [1], [2]. These examples show how to deal with non determinism, timed and probabilistic aspects of WSNs. However, these approaches only deal with selected aspects, while neglecting power consumption issues.

Power consumption can be analyzed at several abstraction levels: instruction level [3], control algorithm level [4], hardware level [5], etc. However, currently, only simulation is applied for power consumption analysis [6], [7].

TinyOS (www.tinyos.net) and nesC (nescc.sourceforge.net) are, respectively, a well-known operating system and a well-known programming language for WSN development. A software is developed as a highly concurrent collection of processes and tasks. Simulation of the TinyOS application can be performed using ns-2 (see www.isi.edu/nsnam/ns), TOSSIM, PowerTOSSIM [7] and VMNet [6].

We propose a methodology for modeling, analysis and development of WSNs: a formal language PAWSN (Process Algebra for WSNs) and a corresponding tool environment TEPAWSN. PAWSN is the classical process algebra extended with time, probabilistic and, specifically, power consumption aspects. TEPAWSN is the related tool environment which facilitates the design, analysis and transformation of PAWSN specifications. It allows both qualitative and quantitative analysis by translating PAWSN specifications to other (formal) languages with tool support.

2. Process Algebra for WSN and Tool Environment

2.1 Process Algebra for Wireless Sensor Network

We introduce process algebra for wireless sensor network modeling (PAWSN). It combines classical process algebra (e.g., CCS) features, such as parallel and sequential composition, with time, probabilistic and power consumption behaviors. Its semantics allows formal analysis and provides a solid basis for the tool development. The relevant rationales behind the development of PAWSN are as follow:

- *Orthogonality:* timing, non deterministic, probabilistic and power aspects can easily be added or omitted from a specification when such aspects are unimportant.

- *Usability:* the syntax and language constructs of PAWSN have to be close to common languages used for WSNs (e.g. nesC), making PAWSN intuitive for the engineers.
- *Mapping to automata:* Different extensions of automata are widely used for formal modeling, including the analysis of WSNs with power issues [5]. We aim at transforming PAWSN specifications to the equivalent Power Probabilistic Timed Automata (PPTA) (a type of timed automata embedded with probabilistic and power issues). That will allow us to describe a very large spectrum such as timed, stochastic, probabilistic and power features. PAWSN semantics should allow relevant properties of any PAWSN specification to be preserved through the translation/mapping to the corresponding PPTA.

2.2 Tool Environment for PAWSN

Instead of developing a new tool, our intention is to provide an umbrella tool that allows to specify the behavior of WSNs, including power issues, using PAWSN, and then to translate or to adapt the specification in such a manner that the analysis can be carried out by third party tools, e.g. PowerTOSSIM, VMNet, Prism (see www.prism modelchecker.org), MRMC, Bhave simulator [9], Uppaal (see www.uppaal.org) or CADP (www.inrialpes.fr/vasy/cadp/) according to the different purposes, e.g. simulation, verification or power analysis. This approach was inspired by MOTOR [10] that advocates the so-called *single formalism and multi-solution approach*. The usual practice is to build multiple models, one for each of the different aspect or group of aspects of the system, and then to analyze them. However, this usual approach does not guarantee any property consistency among the models. Indeed, it misses a formal semantics relating the models. Consequently, analysis results hold only for particular models, models which are not completely equivalent to the whole system. Our approach allows the analyze of certain aspects/properties of the same model in such a way that the consistency of analysis/verification is guaranteed. The reference TESPAWN architecture is depicted in Figure 1.

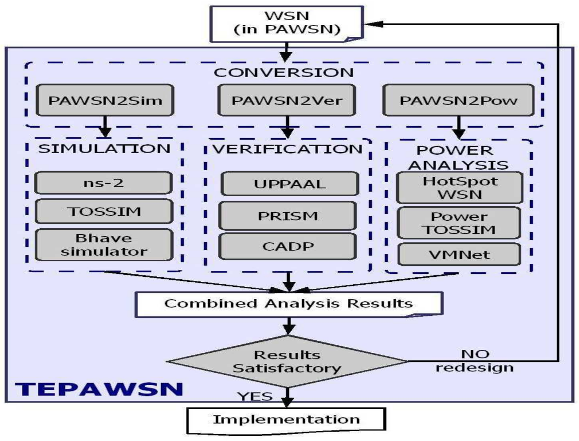

Figure 1. TEPAWSN architecture

3 Application of TEPAWSN

System development using TEPAWSN consists of several components, namely: visualization, simulation, verification and implementation. The procedure is clearly visible from the reference architecture depicted in Figure 1. Three groups of conversion tools are defined for the different purposes.

3.1 PAWSN2Sim Converters and Simulation

PAWSN2Sim converters translate PAWSN specifications to the corresponding models specified by several languages. These models allow simulating PAWSN specifications using different oriented simulators. For instance, PAWSN2nesC converts PAWSN specifications to nesC models in a way allowing simulation and implementation of such models within the TinyOS environment. Also, PAWSN2BHPC allows to employ the toolset BHave (bhpc-simulator.sourceforge.net) for simulation and certain analysis of power consumptions aspects (as a part of simulation).

3.2 PAWSN2Ver Converters and Verification

For the purpose of simulation, verification and implementation of WSNs described in PAWSN, the PAWSN2PPTA, PAWSN2nesC and PAWSN2PRISM tools will be developed. Any PAWSN specification will be mapped to the corresponding PPTA by means of PAWSN2PPTA. The translator PAWSN2nesC converts PAWSN specifications into the corresponding models in nesC for simulation and implementation in TinyOS.

For verification of WSNs, we focus on Model Checking, which is a formal verification technique. The basic idea is to create a mathematical model for the system under scrutiny. The model typically abstracts from everything that is not relevant for the proof of its correctness. Verifying correctness of the model, and thereby of the system, is reduced to a set of requirements that are translated into system properties. These properties are then formalized by expressing them in a property specification language, e.g. temporal logics. Model checking is the (often automated) analysis of whether the constructed model satisfies all formalized properties.

Probabilistic model checking concerns systems with behavior is subject to chance. All events or actions occur with a certain probability. This feature is formalized by constructing a probabilistic model. Properties can refer to the probabilities and are formalized in probabilistic property specification languages. The added value of probabilistic model checking is that it can be used to do quantitative analysis of systems.

PRISM is a probabilistic model checker, that is, a tool for probabilistic modeling and analysis. Models are formulated in the PRISM language that contains three types of models based on discrete Markov chains (DTMC), Markov decision processes (MDP), and continuous-time Markov chains (CTMC). Properties are formulated in the PRISM property specification language containing: probabilistic computation tree logic (for DTMCs and MDPs) and continuous stochastic logic (for CTMCs). The tool itself can be used for automated analysis by discrete event simulation or formal verification based on numeric computation.

Extreme resource constraints and unreliability are inherent to WSNs, leading to different trade-offs. For instance, power consumption versus sensor detection reliability. We typically want to explore different solutions which may have very different power consumption characteristics and different probabilities of detecting a new sensor. Comparing solutions taking into account both the detection reliability and power consumption requires quantitative analysis. Probabilistic model checking allows us to perform this analysis. To this end we create a requirement on the detection probability of a solution. We are interested in all solutions with a detection probability of at least 50%. We construct probabilistic models of WSNs for each solution and analyze whether each model satisfies this property. The models that satisfy the property can be ranked according to power consumption; from which we can choose the best solution.

Similarly, for verification purpose, PAWSN2PRISM translates PAWSN specifications into the equivalent reactive modules which are the input format of the probabilistic model checker PRISM. Also, it is evident that popular automaton based model checkers (e.g. UPPAAL) can be used, with some adaptation on PPTA, to verify properties of WSNs described in PAWSN via the translations to PPTA.

3.3 PAWSN2Pow and Power Consumption Visualization

In addition to simulation tools, we aim to establish a visual connection between a WSN design and its power consumption. This will be achieved by annotating power consumption information from simulation as TinyOS applications (e.g. using PowerTOSSIM/VMNet) onto a PPTA of the WSN design. This visual connection helps:

- to address power consumption and to identify possible design flaws at an earlier stage;
- to uncover more opportunities for application of existing low-power design techniques;
- to find such opportunities more quickly than in traditional manual/iterative approaches.

The completing part of the full environment is a visualization tool for power analysis. The main goal of involving a visualization tool is to provide a visual connection between a WSN design and its power consumption. The large amount of nodes in WSN and the invisible communication topology ask inherently for a graphical interpretation of the design algorithm. Such a tool provides quick overview of the simulation results and speeds up the evaluation phase. Further, a visual presentation allows designers to have more opportunities to uncover existing low-power designs. Last, but not least, it offers an elegant way to compare individual designs among them.

We propose to use a WSN Simulator and Visualize NetTopo in order to visualize power consumption. This will be achieved by annotating power consumption information from simulation as TinyOS applications (e.g. using PowerTOSSIM/VMNet) onto a PPTA of the WSN design.

4. Conclusions

The TEPAWSN tool environment for WSNs has been presented. We expect that TEPAWSN will make a relevant contribution to the WSN research and development by facilitating the design and analysis of power aware WSNs. The development will be performed step-wise, i.e. selected converters will be implemented and tested with simple case studies. Depending on the results further actions will be taken.

The development of TEPAWSN will be funded by Solari, Hong-Kong (official sales agent of Sanyo LCD camera modules - http://www.solari-hk.com) starting from September 2009; and in cooperation with engineers from industrial entities as well as researchers from academic research institutes.

References

[1] P. Olveczky and S. Thorvaldsen, "Formal Modeling and Analysis of Wireless Sensor Network Algorithms in Real-Time Maude," in the 20th IEEE International Parallel &Distributed Processing Symposium, 2006.

[2] A. Demaille, T. Herault, and S. Peyronnet, "Probabilistic verification of sensor networks," in International Conference on Research, Innovation and Vision for the Future, 2006.

[3] H. Joe, J. Park, C. L. Dukkyun Woo and H. Kim, "Instruction-level power estimator for sensor networks," ETRI Journal, vol. 30, no. 1, 2008, pp. 47–58.

[4] B. Z. Ares, P. Park, C. F. A. Speranzon, and K. H. Johansson, "On Power Control for Wireless Sensor Networks: System Model, Middleware Component and Experimental Evaluation," in IFAC European Control Conference (ECC'07), 2007.

[5] F. Maraninchi, L. Samper, K. Baradon, and A. Vasseur, "Lustre as a System Modeling Language: Lussensor, a Case-study with Sensor Networks," in ETAPS'07, Satellite Workshop on Model-driven High-level Programming of Embedded Systems, 2007.

[6] H. Wu, Q. Luo, P. Zheng, and L. M. Ni, "VMNet: Realistic emulation of wireless sensor networks," IEEE Trans. Parallel Distrib. Syst., vol. 18, no. 2, 2007, pp. 277–288.

[7] V. Shnayder, M. Hempstead, B. Chen and M. Welsh, "Power-TOSSIM: Efficient Power Simulation for TinyOS Applications," in the Second ACM Conference on Embedded Networked Sensor Systems (SenSys04), 2004.

[8] J.-P. Katoen, I. S. Zapreev, E. M. Hahn, H. Hermanns and D. N. Jansen, "The Ins and Outs of The Probabilistic Model Checker MRMC," QEST, www.mrmc-tool.org, 2007.

[9] T.Krilavičius, "Hybrid Techniques for Hybrid Systems", Enschede, 2006.

[10] H. Bohnenkamp, H. Hermanns, and J.-P. Katoen, "Motor: The MoDeST Tool Environment," in Proceedings of the 13th International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS'07), LNCS vol. 4424, 2007, pp. 500–504.