TEPAWSN - A Tool Environment for Wireless Sensor Networks

K.L. Man, T. Vallee and H.L. Leung
Solari, Hong Kong
Emails: kalok2006@gmail.com, tv1@cs.ucc.ie and sales@solari-hk.com

M. Mercaldi
M.O.S.T., Italy
Email: michele.mercaldi@gmail.com

J. van der Wulp
Technische Universiteit Eindhoven, The Netherlands
Email: j.v.d.Wulp@tue.nl

M. Donno
Minteos, Italy
Email: donno@minteos.com

M. Pastrnak
Siemens, Slovakia
Email: m.pastrnak@gmail.com

***Abstract*—We strongly believe that Wireless Sensor Network (WSN) development must be supported from the design phase by formal methods to achieve strong results on correctness, performance, cost and efficiency.**

In this paper, we propose a tool environment called TEPAWSN for WSNs which will make a relevant contribution to the WSN research by facilitating the design and analysis of power aware WSNs.

***Index Terms*—Wireless Sensor Networks (WSNs), formal methods, process algebras, modelling, simulation and verification, low power**

I. Introduction and Background

A Wireless Sensor Network (WSN) is composed of a potentially large number of small sensor nodes communicating among themselves wirelessly. Their application areas range from defence and battlefield surveillance to health or home applications. Modern WSNs are complex and costly. They must be released within shorter time frames and fault free while their designers/developers face a constant increase in market competition. It is therefore very important that errors are found and removed as early as possible, preferably in the design phase. To detect these errors, designers/engineers often turn to *formal methods* [1].

Formal methods provide a set of notations and techniques for construction of mathematical models of systems/WSNs and for (automatic) verification of these models against some requirements. The requirements are usually represented in terms of a set of properties that a system/WSN should satisfy. A property can be either *qualitative*, e.g. "a system never deadlocks", if we are interested in the functional behaviour of a system/WSN, or it can be *quantitative*, e.g. "throughput is as desired", if we are interested in a system performance.

On the other hand, due to recent advances in applications such as mobile communications, multimedia, embedded systems and sensor networks, power-efficient design has become an essential task in developing WSNs. Efficient power estimation is paramount to energy saving. The early power estimation and accurate predictions for timing and power have a great impact on the overall cost of the fabricated circuits and WSNs as well as on shortening the time to market for such circuits and WSNs.

A wide range of power optimisation techniques (e.g. DPM and DVS [2]) has appeared in the literature/market which can dramatically reduce power consumption in many types of circuits and WSNs. Several of these techniques have already been incorporated into popular commercial tools and have become widely adopted as a result.

A. Formal Modelling and Analysis of WSNs

We strongly believe that WSN development must be supported from the design phase by formal methods to achieve strong results on correctness, performance, cost and efficiency. Moreover, advanced WSN algorithms pose challenges to their formal modelling and analysis, such as probabilistic and real-time behaviours, novel forms of communication and analysing both correctness and performance.

In the last few years, various formalisms and verification tools [3], [4], [5], [6] able to deal with timed, non-deterministic and probabilistic behaviour have been used for the formal specification and analysis of WSNs.

However, to the best of our knowledge, none of such formalisms can deal with power issues while saving power is a very critical issue in WSNs. Indeed sensor nodes are typically powered by batteries with a limited capacity. Power analysis of WSNs can be done at several stages during the development and deployment, e.g. at the instruction level [7], control algorithm level [8], hardware level [9], etc. Nevertheless, power analysis of WSNs has been mainly and currently addressed in a simulation context [10], [11].

B. WSN Application Development

TinyOS [12] is an operating system and platform targeting WSNs. It is an open source component-based embedded operating system written in the nesC programming language (network embedded systems C which is an extension to C) [13] as a set of cooperating tasks and processes. It is intended to be incorporated into *smartdust*. Common simulators for TinyOS are ns-2 [14], TOSSIM [15], PowerTOSSIM [11] and VMNet [10].

To date, WSN applications have been mainly and widely developed on TinyOS operating system using the nesC programming language. It is also worth mentioning that most of the current WSN hardware systems are TinyOS compatible.

C. Introducing PAWSN and TEPAWSN

In this paper, we outline the benefits of using formal methods to specify and analyse WSNs including power issues. We introduce a formal language PAWSN (Process Algebra for WSNs) able to describe the timed, non-deterministic and/or probabilistic behaviour of WSNs including the power issues. Then we present an efficient tool environment, TEPAWSN, for both qualitative and quantitative performance evaluation as well as effective power estimation of WSNs. The TEPAWSN tool environment facilitates the design, analysis and transformation of PAWSN specifications. It implements the formal semantics of PAWSN and it is also designed to transform and abstract PAWSN specifications such that analysis can be carried out by third-party tools for different analysis purposes. Furthermore, TEPAWSN provides a flexible programming environment in which new techniques for WSNs can be developed (see Section II-C also); and possibly compared with similar techniques within the TEPAWSN tool environment.

II. PAWSN AND TEPAWSN: DESCRIPTION

A. Formalisms: PAWSN and PPTA

We first propose the description language PAWSN which is intended to have a rigid formal semantics that incorporates timed, non-deterministic and/or probabilistic behaviour as well as power issues for WSNs. PAWSN is a variant of classical process algebras like ACP [16] and ATP [17] and shares their compositional structure (e.g. sequential and parallel composition). Its semantics allows formal analysis and gives a solid basis for the tool development. The relevant rationales behind the development of PAWSN are as follows:

- Orthogonality: timing, non-deterministic, probabilistic and power aspects can easily be added or omitted from a specification if such aspects are unimportant.
- Usability: formal specification languages have also disadvantages because WSN designers are often uncomfortable with mathematical notations and the syntax of formal specification languages is not intuitive to them. Hence, the syntax and language constructs of PAWSN have to be close to common languages used for WSNs (e.g. nesC).
- Mapping on automata: automaton theory and different extensions have been widely used for the analysis of WSNs including power issues (see [4], [9]). We intend to formalise and develop the common automaton models used in the literature (e.g. [9]) for the analysis of WSNs to a new extension namely Power Probabilistic Timed Automata (PPTA) which allows us to describe a very large spectrum of features such as timed, stochastic, probabilistic and power. Furthermore, the semantics of PPTA allows the mapping from each PAWSN specification on some PPTA. This means in particular that relevant properties of any PAWSN specification can be preserved after the translation/mapping to the corresponding PPTA.

B. TEPAWSN: a single formalism and multi-solution approach

The key idea behind TEPAWSN is to specify the behaviour of WSNs including power issues using PAWSN and to translate or adapt the specifications described by PAWSN in such a way that the actual analysis can be carried out by third-party tools such as PowerTOSSIM, VMNet, PRISM [18], [19], UPPAAL [5] or CADP [20] for different purposes (e.g. simulation, verification and power analysis). This is the so-called *single formalism and multi-solution* approach as in [21], [22].

It is worth mentioning that this approach is contrary to the common approach to build different models to describe different aspects of a system/WSN and then analyse these models. Such a common approach does not ensure any property consistency between these models because it lacks of a formal semantics to relate different models together (at the semantical level). Hence, it may not be possible to relate back the validation and verification results to the original system/network under analysis.

C. Software development in TEPAWSN

1) Simulation, verification and implementation of WSNs: 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 is 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 of the system under scrutiny. The model typically abstracts from everything that is not relevant for proving its correctness. Verifying correctness of the model, and thereby the system, is reduced to a set of requirements that are translated to system properties. These properties are then formalised by expressing them in a property specification language e.g. temporal logics. Model checking is the (often automated) analysis whether the constructed model satisfies all formalised properties.

Probabilistic model checking [23] concerns systems with behaviour that is subject to chance. All events or actions in occur with a certain probability which is formalised by constructing a probabilistic model. Properties can refer to the probabilities and are formalised 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 a tool for probabilistic modelling and analysis. Models are formulated in the PRISM language that contains thee types of models based on: discrete Markov chains (DTMC), Markov decision processes (MDP) and continuous-time Markov chains (CTMC) (see [24]

for details). Properties are formulated in the PRISM property specification language containing: probabilistic computation tree logic (for DTMCs and MDPs) and continuous stochastic logic (for CTMCs) [23]. 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 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 analyse 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 with respect to power consumption.

Similarly, for verification, PAWSN2PRISM translates PAWSN specifications to 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.

2) Power visualisation tool: 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 full environment is a visualisation tool for power analysis. The main goal of involving a visualisation 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 inherently ask 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 uncover more opportunities of existing low-power design. Last, but not least, it offers an elegant way to compare individual designs among them.

We propose to use a WSN Simulator and Visualiser NetTopo [25] that aims to visualise 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 (see also [26] for details). Figure 1 provides an example of NetTopo user interface that presents the WSN topology and visualise the subset of nodes those discriminate certain criteria.

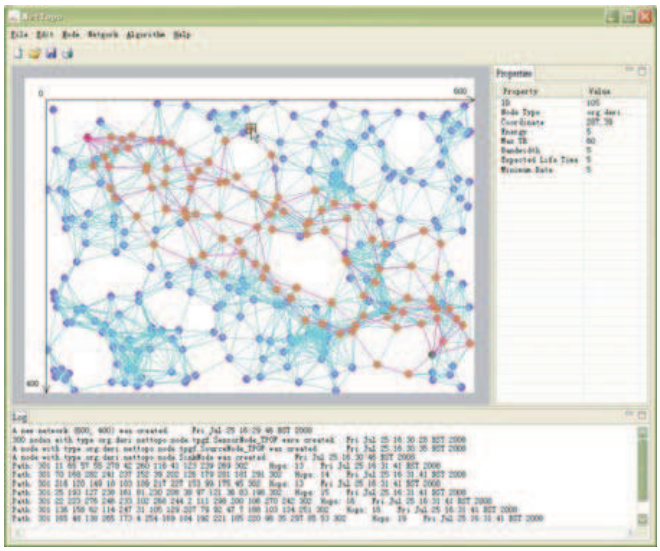

Fig. 1. NetTopo main GUI (taken from [25]).

Figure 2 depicts the TEPAWSN architecture. Also, it is not hard to see that existing tools (e.g. Approximate Probabilistic Model Checker (APMC) [27]) can be easily integrated into the TEPAWSN tool environment.

III. Summary and Current Status

The TEPAWSN tool environment for WSNs has been proposed. We believe that TEPAWSN will make a relevant contribution to the WSN research by facilitating the design and analysis of power aware WSNs.

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 March 2009; and in cooperation with engineers from industrial entities as well as researchers from academic research institutes. We hope to complete the development of PAWSN and PPTA by the end of 2009.

Acknowledgement

Many thanks go to the industrial partner Minteos, Italy (http://www.minteos.com) for the research work presented in this paper.

References

[1] FMEurope, "Formal Methods Europe," http://www.fmeurope.org.

[2] M. Keating, D. Flynn, R. Aitken, A. Gibbons, and K. Shi, Eds., *Low-Power Methodology Manual*. Springer-Verlag, 2007.

[3] 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.

[4] A. Fehnker, L. van Hoesel, and A. Mader, "Modelling and verification of the lmac protocol for wireless sensor networks," in *Integrated Formal Methods IFM*, 2007.

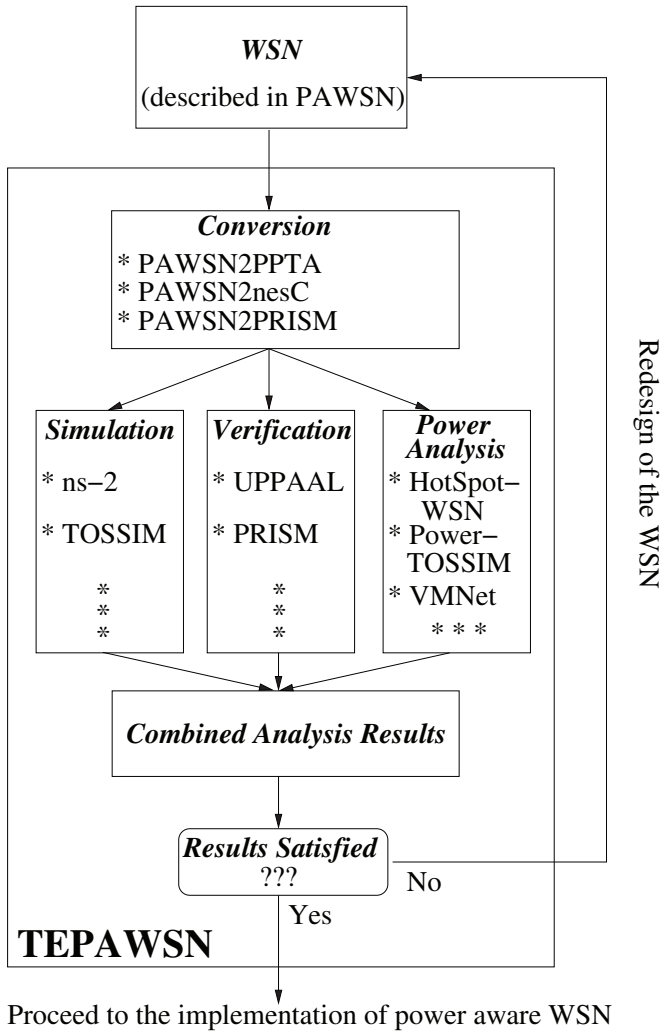

Fig. 2. TEPAWSN architecture

[5] K. G. Larsen, P. Pettersson, and W. Yi, "UPPAAL in a Nutshell," *Int. Journal on Software Tools for Technology Transfer*, vol. 1, no. 1–2, pp. 134–152, 1997.

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

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

[8] 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.

[9] 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.

[10] 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, pp. 277–288, 2007.

[11] V. Shnayder, M. Hempstead, B. rong Chen, and M. Welsh, "PowerTOSSIM: Efficient power simulation for tinyos applications," in *the Second ACM Conference on Embedded Networked Sensor Systems (SenSys04)*, 2004.

[12] TinyOS, "TinyOS: An open-source operating system designed for wireless embedded sensor networks," http://www.tinyos.net.

[13] nesC, "nesC: A programming language for deeply networked systems," http://nescc.sourceforge.net.

[14] ns-2, "The Network Simulator - ns-2," http://www.isi.edu/nsnam/ns.

[15] P. Levis, N. Lee, M. Welsh, and D. Culler, "Tossim: accurate and scalable simulation of entire tinyos applications," in *Proceedings of the 1st international conference on Embedded networked sensor systems*, 2003.

[16] J. C. M. Baeten and W. P. Weijland, *Process Algebra*, ser. Cambridge Tracts in Theoretical Computer Science. Cambridge, United Kingdom: Cambridge University Press, 1990, vol. 18.

[17] X. Nicollin and J. Sifakis, "The algebra of timed processes, ATP: Theory and application," *Information and Computation*, vol. 114, pp. 131–178, 1994.

[18] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker, "PRISM: A tool for automatic verification of probabilistic systems," in *the 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06)*, March 2006.

[19] PRISM, http://www.prismmodelchecker.org.

[20] J. C. Fernandez, H. Garavel, A. Kerbrat, L. Mounier, R. Mateescu, and M. Sighireanu, "CADP - a protocol validation and verification toolbox," in *Proceedings 8th Conference on Computer Aided Verification (CAV'96)*, ser. Lecture Notes in Computer Science, vol. 1102, 1996, pp. 437–440.

[21] E. Bortnik, N. Trčka, A. J. Wijs, B. Luttik, J. M. van de Mortel-Fronczak, J. C. M. Baeten, W. J. Fokkink, and J. E. Rooda, "Analyzing a χ model of a turntable system using Spin, CADP and Uppaal," *Journal of Logic and Algebraic Programming*, vol. 65, no. 2, pp. 51–104, 2005.

[22] 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)*, ser. Lecture Notes in Computer Science 4424. Springer-Verlag, 2003, pp. 500–504.

[23] L. D. Alfaro, "Model checking of probabilistic and nondeterministic systems," in *Foundations of Software Technology and Theoretical Computer Science*. Springer-Verlag, 1995, pp. 499–513.

[24] C. Baier, J.-P. Katoen, and H. Hermanns, "Approximate symbolic model checking of continuous-time markov chains," in *International Conference on Concurrency Theory*, 1999, pp. 146–161.

[25] L. Shu, C. Wu, Y. Zhang, J. Chen, L. Wang, and M. Hauswirth, "Nettopo: Beyond simulator and visualizer for wireless sensor networks," in *The Second International Conference on Future Generation Communication and Networking*. Hainan Island, China: IEEE, 2008.

[26] T. English, K. L. Man, E. Popovici, and M. Schellekens, "HotSpot : Visualising dynamic power consumption in rtl designs," in *the 6th East-West Design & Test Workshop (EWDTS)*. Lviv, Ukraine: IEEE, 2008.

[27] APMC, "Approximate Probabilistic Model Checker," apmc.berbiqui.org.