Self-Adaptive Wireless Sensor Networks

Michael Frey
Department of Computer Science
Humboldt-Universität zu Berlin
frey@informatik.hu-berlin.de

Abstract

Wireless sensor networks (WSNs) employ complex software systems where environmental changes are not only monitored and propagated, but also often decisions are taken and appropriate actions upon the environment are performed. Hence versatile, flexible, resilient and dependable software systems are required which are capable of handling changing operational contexts, environments or system characteristics. Self-adaptive software systems are capable of handling these requirements and adjust their behavior by monitoring the environment and the underlying system itself. We aim at providing abstractions for the design and development of self-adaptive software systems for WSNs where the specialties of these networks are taken into account. Furthermore concepts of control theory, such as hierarchical control loops, will be adapted and used in order to build flexible self-adaptive WSNs which are capable of handling dynamic changes.

Keywords

self-adaptive systems, wireless sensor networks, run-time phenomena

1 Problem Domain

WSNs have been an active research field over the last decade. Challenges concerning design and application have been addressed and significant advances in development and deployment have been made. In order to simulate the behavior of complex algorithms on top of these networks, often simple wireless radio propagation models are used. Typically, these simplified models do not cope well with the complexity of wireless wave propagation where phenomena such as multi-path or fading have to be taken into account [9]. Additionally, wireless communications is often affected by environmental conditions and for most cases it is difficult to assess the impact of the environment prior to deployment. Examples for these conditions are interferences caused by concurrent transmissions within a wireless network or between cohabiting wireless networks and other electromagnetic sources or radio tranceivers which distort received or send signals due to internal noise. In many applications software on sensor nodes is also required to adapt in terms of non-functional constraints such as energy consumption, memory or computational power. These effects are embraced by the term *run-time phenomena*. Because run-time phenomena are often temporary and spatially limited, self-adaption mechanisms are required to handle these dynamic changes.

High-level programming models for WSNs have been proposed [16, 13] providing simple and versatile abstractions for low-level programming issues such as memory management and communication or data aggregation and dissemination. While programming models ease the development of applications for WSNs for domain experts, little or no methods and abstractions for handling of run-time phenomena exist [14]. In addition, with a new generation of sensor nodes [11] with higher computational power, more memory and less energy consumption, WSNs are capable of adapting to new complex tasks. Disaster management is an example where a WSNs adapts to new tasks. For example, such a system warns inhabitants in case of a natural disaster, disables gas and power pipelines and becomes a communication fallback system for locals and rescue teams. At present, domain experts are forced to implement dedicated self-adaption mechanisms on a per-application basis. Typically, this includes a feedback process with four key activities: collecting and analyzing data, deciding and acting [4]. In many cases concepts from control theory [3] are applied to self-adaptive software systems where embedded control loops adapt the behavior of a software system. However, little has been done to make control loops explicit and to expose self-adaptive properties to domain experts [4].

1.1 Contribution of the Thesis

The main goal for the thesis is the development of programming abstractions for run-time phenomena as well as methods and concepts for handling these dynamic effects. For this purpose a domain-specific language (DSL) and a run-time environment will be developed and evaluated. Furthermore, methods and concepts for the integration of control loops in programming abstractions and how to expose self-adaptive properties to domain experts will be investigated.

With regard to the feedback process, questions arising from the key activities need to be answered. This includes how reliable is the collected data, how can the current state of a system be inferred, how much past state is required for future decisions or how to adapt in order to reach a specific state? The proposed framework will support a developer regarding this questions during development by providing building blocks for this issues. The work will be evaluated at the Humboldt Wireless Lab (HWL) testbed at Humboldt-Universität zu Berlin [7]. The testbed consists of up to 160 heterogeneous indoor and outdoor sensor nodes. The sensor nodes are off-the shelf Linux based routers which are equipped with sensors. However, observations of run-time phenomena in wireless communication in the testbed are also applicable on WSNs. By tightening the requirements on runtime-environment and heuristics the approach will also be applicable on strong resource-constrained sensor nodes found in many test-beds. Hence, the results of the thesis might be applicable to other areas of self-adaptive software systems.

2 Approach

Research within this thesis focuses on two major topics. First, a DSL for high-level programming abstractions and heuristics for run-time phenomena will be provided. Since run-time phenomena affect a system on many different levels ranging from low-level communication issues up to high-level application specific topics, abstractions are required which allow to cover different level of details. The concept of platform-independent (PIM) and platform-specific models (PSM) introduced by model-driven software development (MDSD) [18] will allow to close the gap. In MDSD executable applications are generated from formal software models. The transition between these two types of models is automated by using a set of formal transformation rules. Using MDSD will first allow to abstract from the underlying system and second providing the DSL for different existing programming abstractions and systems. Furthermore, the usage of models for the run-time will be investigated where a model of an application reflects the state of the application. Thus, the application and a model representation of the application needs to be synchronized. Here, self-adaption is realized as a set of transformations on the run-time model which itself effect the application.

For the second part of the thesis, the creation and integration of control loops for WSNs will be analyzed. In particular, methods and concepts for the design and implementation of self-adaptive WSNs based on control loops must be identified and specified. In [10], IBM presented an architectural blueprint for self-adaptive software systems. However, the approach is not directly applicable to WSNs due to resource constraints. Still, re-using building blocks such as services and components on a conceptual level of the architectural blueprint could be feasible. Typically, communication protocols in WSNs are application specific and tied to a concrete WSN testbed or WSN deployment. Therefore, multiple approaches for a service-oriented communication paradigm in terms of remote procedure calls have been proposed [5, 12] in order to re-use a standard set of services for new deployments and application scenarios. Re-using the service-oriented paradigm will allow using services and components as main entities for self-adaptive WSNs. Here, applications for WSNs are bundled as components which are offering and consuming services of other applications. In addition, system properties are bundled as well in order to ease and unify access to the underlying operating system. Components and services compose building blocks for applying control loops to WSNs. In order to ensure certain properties of control loops such as stability, controllability and observability, the system needs to be instrumented and measurements need to be done. Applying Quality-of-Service (QoS) requirements to services will allow to quantify and qualify control loop properties. A control loop can be specified as a set of components which interact by means of services. This approach allows to tie multiple control loops together without modeling them explicitly.

The presented approach lead to a number of interesting research questions. Control engineering has developed multiple approaches for modeling and reasoning about feedback such as the Model Identification Adaptive Control (MIAC) [6] model. Typically, these approaches introduce well-defined elements including controllers, processes, adjustment mechanisms and model references along with prescribed dependencies among these elements. From a modeling perspective questions within respect to the integration in the presented MDSD approach need to be answered. The transformation of the PIM to a control loop model as immediate representation of the PSM might be a promising approach. In addition, suitable stateful model representations for applications as well as synchronization and instrumentation mechanisms for these run-time models need to be defined. Applying control loops to WSNs in order to enable self-adaption leads to a number of interesting research questions. The interaction of multiple control loops can be represented in different structural arrangements including sequential, parallel or hierarchical. What are the effects of these different structural arrangements to the WSNs? What are the trade-offs of applying a control loop approach to WSNs? Furthermore, oscillation effects need to be analyzed since they might lead to desired or undesired behavior on sensor nodes of a network which are not affected by run-time phenomena. What are the metrics to define these oscillation effects and how to attenuate or circumvent them? In addition, suitable evaluation metrics need to be defined in order to evaluate the presented approach.

3 Related Work

Discussion about related work is divided in two parts. First similar approaches in programming abstractions and second self-adaptive run-time environments for WSNs are presented and discussed.

At present, the scientific model-driven community focuses on run-time models, where the applicability of models and abstractions is extended to the run-time. In contrast to development models, run-time models are used to reason about the operating environment and run-time behavior [2]. It is envisioned that run-time models will play a integral role in the management of self-adaptive systems. Since research on run-time models is quite new, most of the related work in model-driven software development for WSNs focuses on the modeling of applications with a subsequent code generation. In [1], a toolchain for the development and test of WSN applications is proposed. Test cases are installed and executed on the respective nodes and visualized after test execution in the corresponding model. In [19], run-time models are used for the monitoring and adapting of software on mechatronic devices. Each device is considered as an agent in an decentralized system where each agent maintains a model holding a representation of itself and its environment. Run-time models are used for the detection of inconsistencies and to predict possible future states. Mechatronic devices and multi-agent systems are not directly related to WSNs, but are comparable in terms of resource constraints. To the authors best knowledge no approaches for the application of run-time models to WSNs and its applications are known. However, approaches on resource constrained devices might be a good starting point for further investigations.

Muller et.al proposed in [15] a approach for the specification of self-adaptive model-driven software by using methods and concepts from control theory. Here, self-adaptive model-driven software is represented as a closed loop system where the system under control is the running application and the corresponding reference is the goal of the system. The controller itself forms the development approach where the software application is built, deployed and evolves over time. Run-time information is monitored and applied to run-time models which are manipulated and transformed. The publication itself proposes a general idea instead of a architectural blueprint which defines where models are applied to. However, it points out a few research directions and maps concepts from control theory to software-engineering. The usage of services and components to self-adaptive software is not new. Peper et. al [17] presented a component-oriented approach for the development of dynamic-adaptive network embedded systems. Components provide and consume services and are controlled by channels which allow among others to configure or monitor a component and its services. In contrast to many other approaches the framework does not rely on centralized entities. A service unit serves exactly only one client since resources are bound by requests. Thus access conflicts to hardware resources and side-effects are avoided, but leads to limitation in communication. Agilla [8] is a programming model and a middleware which implements self-adaptive applications for WSNs by means of mobile agents. Mobile agents are processes which can migrate or clone from node to node while maintaining their state. In the presented approaches the programming model or the run-time environment are independent from each other. The integration of programming abstractions with the run-time environment by means of run-time models and control loops lead to a new approach for self-adaptive software systems.

4 Current Work and Outlook

At present, different run-time phenomena at the HWL are being analyzed. In particular, fairness issues in wireless communication where nodes are communicating rarely or not at all due to run-time phenomena are being investigated. Following from the analysis, heuristics for this run-time phenomena which can applied to the nodes need to be developed. Since the evaluation of the approach in a testbed is limited with regard to real-world issues, an evaluation is planned for an already deployed scheduled real-world WSN.

5 References

[1] M. Al Saad, E. Fehr, N. Kamenzky, and J. Schiller. Scatterclipse: A model-driven tool-chain for developing, testing, and prototyping wireless sensor networks. In *Parallel and Distributed Processing with Applications, 2008. ISPA'08. International Symposium on*, pages 871–885. IEEE, 2008.

[2] G. Blair, N. Bencomo, and R. France. Models@run.time. *Computer*, 42(10):22–27, 2009.

[3] W. L. Brogan. *Modern Control Theory*. Prentice-Hall, Englewood Cliffs, NJ, 1990.

[4] B. Cheng, R. de Lemos, H. Giese, P. Inverardi, J. Magee, J. Andersson, B. Becker, N. Bencomo, Y. Brun, B. Cukic, and Others. Software engineering for self-adaptive systems: A research roadmap. In *Software Engineering for Self-Adaptive Systems*, pages 1–26. Springer, 2009.

[5] P. Corke, T. Wark, R. Jurdak, W. Hu, P. Valencia, and D. Moore. Environmental Wireless Sensor Networks. *Proceedings of the IEEE*, 98(11):1903–1917, 2010.

[6] G. Dumont and M. Huzmezan. Concepts, methods and techniques in adaptive control. In *American Control Conference, 2002. Proceedings of the 2002*, volume 2, pages 1137–1150. IEEE, 2002.

[7] J. Fischer, J.-P. Redlich, J. Zschau, C. Milkereit, M. Picozzi, K. Fleming, M. Brimbuli, B. Lichtblau, and I. Eveslage. A Wireless Mesh Sensing Network for Early Warning. *Journal of Network and Computer Applications*, to appear.

[8] C. Fok, G. Roman, and C. Lu. Agilla: A mobile agent middleware for self-adaptive wireless sensor networks. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 4(3):1–26, 2009.

[9] E. Gaura, L. Girod, J. Brusey, M. Allen, and G. Challen, editors. *Wireless Sensor Networks: Deployments and Design Frameworks*. Springer Verlag, 2010.

[10] IBM. An architectural blueprint for autonomic computing. *Autonomic Computing White Paper*, 2004.

[11] J. Ko, Q. Wang, T. Schmid, W. Hofer, P. Dutta, and A. T erzis. Egs: A Cortex M3-based Mote Platform. In *Sensor Mesh and Ad Hoc Communications and Networks (SECON), 2010 7t h Annual IEEE Communications Society Conference on*, pages 1–3.

[12] T. May, S. Dunning, G. Dowding, and J. Hallstrom. An rpc design for wireless sensor networks. *International Journal of Pervasive Computing and Communications*, 2(4):384–397, 2007.

[13] L. Mottola and G. Picco. Logical neighborhoods: A programming abstraction for wireless sensor networks. *Distributed Computing in Sensor Systems*, pages 150–168, 2006.

[14] L. Mottola and G. Picco. Programming wireless sensor networks: Fundamental concepts and state of the art. *ACM Computing Surveys (CSUR)*, 43(3):19, 2011.

[15] P. Muller, O. Barais, and F. Fleurey. Control-theory and models at runtime. In *Proceeding of 2nd International Workshop on Models@ run. time.(Sept 2007)*, pages 1–6, 2007.

[16] R. Newton, A. Giridhar, and M. Welsh. Building up to Macroprogramming: An intermediate language for sensor networks. *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005.*, pages 37–44, 2005.

[17] C. Peper and D. Schneider. Component engineering for adaptive ad-hoc systems. In *Proceedings of the 2008 interna-*

tional workshop on Software engineering for adaptive and self-managing systems, pages 49–56. ACM, 2008.

[18] T. Stahl and M. Völter. *Model-Driven Software Development: Technology, Engineering, Management*. John Wiley & Sons, 2006.

[19] M. Vallée, M. Merdan, and T. Moser. Using models at runtime for monitoring and adaptation of networked physical devices: Example of a flexible manufacturing system. In *5th International Workshop on Modelsruntime*, pages 84–95, 2010.

Acknowledgments

This work was supported by a grant from the DFG (German Research Foundation, research training group METRIK).

Biography

Michael Frey is a Ph.D. student in the German Research Foundation (DFG) funded research training group ”Model-Based Development of Technologies for Self-Organizing Decentralized Information Systems in Disaster Management” (METRIK) at Humboldt-Universität zu Berlin. He received his diploma degree (equivalent to bachelor of science) in computer science in February 2009 from Wiesbaden University of Applied Sciences. While he was an undergraduate student he worked as research assistant at the Communication Infrastructure Management group at the Telecommunications Software and Systems Group in Waterford, Ireland. In October 2010 he received his Master of Science at the same university. During his studies he worked as research assistant at the Distributed Systems Lab at Wiesbaden University of Applied Sciences. His thesis advisors are Prof. Dr. Joachim Fischer and Prof. Dr. Jens-Peter Redlich. Submission of the Ph.D. thesis is scheduled for the beginning of the year 2014.