Runtime Verification and Reflection for Wireless Sensor Networks

Stefan Fischer
Telematics Institute
University of Lübeck, Germany
fischer@itm.uni-luebeck.de

Martin Leucker
Institute for Software Engineering and Programming Languages
University of Lübeck, Germany
leucker@isp.uni-luebeck.de

***Abstract*—The paper proposes to re-visit a light-weight verification technique called runtime verification in the context of wireless sensor networks. The authors believe that especially an extension of runtime verification which is called runtime reflection and which is not only able to detect faults, but diagnose and even repair them, can be an important step towards robust, self-organizing and self-healing WSNs. They present the basic idea of runtime reflection and possible applications.**

I. Motivation

One of the main characteristics of wireless sensor networks (WSNs) is that they often operate in areas and/or in situations where they cannot be supervised by human administrators. It is thus of major importance that a WSN is designed and implemented very carefully in order to avoid system failures – which can then not be repaired, potentially resulting in a complete loss of the system. Traditionally, verification techniques such as theorem proving, model checking, and testing are used to prove or at least increase the trust in the correctness of software. Since these techniques often have very strong requirements like the existence of formal models or do not cover all potential errors, a technique called *runtime verification* was developed about 15 years ago [1], which complements techniques like model checking and testing and is rather lightweight compared to them. The distinguishing feature of runtime verification is that it operates at runtime, which makes it possible to re-act whenever a software system behaves incorrectly. We believe that this is a very interesting feature to be applied to autonomous and self-organized systems such as WSNs, since it would allow to detect faults at runtime and even realize a self-healing behavior of the overall system. With this position paper, we point out the potential of runtime verification and more specifically of runtime reflection [2], a technique based on runtime verification which is specifically focused on not only detecting failures but also finding their reason, and outline research directions.

The rest of this paper is structured as follows: in Section II, we first give a brief introduction into the general idea of runtime verification and runtime reflection. In Section III, we then present our ideas how these technologies can be applied to operating WSNs in a way which allows detecting faults and healing failures during runtime.

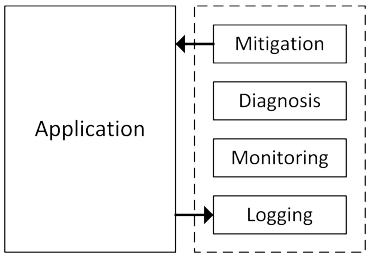

Figure 1. Architecture of the runtime reflection framework

II. Runtime Verification and Reflection

In [3], runtime verification is defined to be the discipline of computer science that deals with the study, development, and application of those verification techniques that allow checking whether a run of a system under scrutiny satisfies or violates a given correctness property.

A run in the above sense is a possibly infinite sequence of the system's states. An *execution* of a system is defined to be a finite prefix of a run. During runtime, it is only possible to observe executions of the system. In contrast to traditional verification techniques which deal with a run or rather all possible runs of a system, runtime verification only deals with executions which is why it can be performed during runtime and also explains why we call it a lightweight verification technique.

The most important tool in runtime verification is the *monitor* whose job is to decide whether an execution meets a correctness property. A monitor can be both an online monitor and then checks, in an incremental way, the current execution of a system, and it can also work offline and then checks recorded executions of a system. Typically, monitors installed on a system should be automatically generated from some higher-level description of the system specification, for instance in Linear Temporal Logic or variants of it.

Having said the above, it is clear that runtime verification itself only deals with the *detection of violations* of correctness properties. In order to also react to these violations, for instance in the sense of "repairing" them, further techniques are necessary. One such technique, which is based on runtime verification, is called runtime reflection (also see [2]). Basically, runtime reflection is an architecture pattern for the

978-1-4673-6269-6/13 © 2013 IEEE

SESENA 2013, San Francisco, CA, USA

development of reliable systems (see Figure 1). In addition to the monitoring (and logging) layer already described conceptually above, it provides further components for diagnosis and mitigation. The diagnosis layer collects the information from the distributed monitors and derives explanations for the current system state. The mitigation layer uses the diagnoses for reconfiguration of the system in order to try to bring it back into a correct state (which obviously may not always be possible).

III. Application to WSNs

We believe that, following the descriptions in Section II, runtime verification and reflection can be very useful tools for supporting the self-organized and widely un-supervised operation of WSNs, up to allowing for self-healing processes of faulty system behavior. We envision the usage of runtime verification techniques during all software development phases, from requirement analysis to design (possibly in some high-level logic language) and then further to simulation, testing, implementation, and maintenance. Runtime monitors can be automatically generated from higher-level descriptions and then deployed on sensor nodes. Diagnosis and mitigation layers will be designed and implemented in order to steadily keep track of the system state, the reasons for faults and the possibility to repair them.

We further believe that due to the nature of WSNs, realizing this vision will be a very interesting challenge. Just to name a few, we see the following interesting questions:

- Can monitors be created in a resource-efficient way, to be installed on resource-constrained sensor nodes?
- How to handle situations in which a sensor node crashes and with it its monitor? We believe that an intelligent placement of monitors and a vice-versa surveillance could be an interesting solution. This triggers a set of new questions such as where to place the monitors, how to associate them to each other, how to make as little as possible use of bandwidth etc.
- How to and where to implement the diagnosis and mitigation layers. Do we need central components for these or could they also be implemented on the sensor nodes.
- How well are logic-based languages suited for WSN system and application specification if done by domain experts and not computer scientists? How could they be supported?
- How to mitigate a failure in the WSN?

IV. Related Work

While runtime verification itself has been around for roughly 15 years as of today, its application to wireless sensor networks has been looked at about only six years ago. Well-known approaches have been presented in [4], [5] and [6].

Herbert et al. [4] develop a method and a prototype for invariant checking during runtime. The approach mainly concentrates on checking application behavior and not so much the correctness of the software itself. Also, it has no focus on the repair of faults itself. Sokolsky et al. [5] provide a much more formal approach based on temporal logics, i.e., they develop a formal runtime verification concept. Still, they just analyze simulator runs and do not work on real-world deployments which, as they say themselves, might behave much different. Wu et al. [6] also employ runtime verification, but use a Petri-Net-based mechanism for specification and also do not further work on solving problems automatically.

While the above are all promising approaches, we believe that they stopped short of practically applicable solutions for WSNs, mainly due to the following issues:

- We need more work on real-world deployments which are much more complex than simulations.
- The next and important step is not only detecting faults, but diagnosing their reason and, if possible, healing them automatically.
- It seems that most approaches have not been developed any further, which is a pity, since thze approach of runtime verification looks very promising.

We believe that putting substantial efforts into developping these methods further could be an important milestone on the road to self-organizing and self-healing sensor networks. We also believe that a transfer of these ideas into industry is of major importance to support these developments.

References

[1] M. Kim, M. Viswanathan, H. Ben-Abdallah, S. Kannan, I. Lee, and O. Sokolsky, "Formally specified monitoring of temporal properties," in *Real-Time Systems, 1999. Proceedings of the 11th Euromicro Conference on*, 1999, pp. 114–122.

[2] M. Leucker and C. Schallhart, "A brief account of runtime verification," *The Journal of Logic and Algebraic Programming*, vol. 78, no. 5, pp. 293 – 303, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1567832608000775

[3] A. Bauer, M. Leucker, and C. Schallhart, "Model-based runtime analysis of distributed reactive systems," in *Software Engineering Conference, 2006. Australian*, Apr. 2006, p. 10 pp.

[4] D. Herbert, V. Sundaram, Y.-H. Lu, S. Bagchi, and Z. Li, "Adaptive correctness monitoring for wireless sensor networks using hierarchical distributed run-time invariant checking," *ACM Trans. Auton. Adapt. Syst.*, vol. 2, no. 3, Sep. 2007. [Online]. Available: http://doi.acm.org/10.1145/1278460.1278462

[5] O. Sokolsky, U. Sammapun, J. Regehr, and I. Lee, "Runtime verification for wireless sensor network applications," in *Runtime Verification*, ser. Dagstuhl Seminar Proceedings, B. Finkbeiner, K. Havelund, G. Rosu, and O. Sokolsky, Eds., no. 07011. Dagstuhl, Germany: Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2008. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2008/1371

[6] Y. Wu, K. Kapitanova, J. Li, J. A. Stankovic, S. H. Son, and K. Whitehouse, "Run time assurance of application-level requirements in wireless sensor networks," in *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, ser. IPSN '10. New York, NY, USA: ACM, 2010, pp. 197–208. [Online]. Available: http://doi.acm.org/10.1145/1791212.1791236