RealSim and DryRun: Tools for a deployment-targeted development of WSNs

Moritz Strübe
Dept. of Computer Science, University of Erlangen, Germany
struebe@cs.fau.de

Abstract

Often large WSN deployments are reported to exhibit unexpected behavior, bad performance or even severe failures due to software problems. One way of mitigating these problems is thorough testing using simulations. However, the simulator often does not resemble the real deployment, even for simple configurations, and thus the simulation results often do not resemble the deployment at all. We therefore suggest to do more deployment-targeted development. Our suggested way of doing this is to use measurements acquired on the actual network to map the network to the simulator and thus conduct more precise simulations. Together with a set of tools to analyze the more accurate simulation results, the knowledge won can be used to improve the quality of the deployed network.

1 Introduction

Wireless sensor networks (WSNs) are actively researched for more than a decade and are said to emerge in huge quantities any time soon. This has not happened yet, though. One of the reasons for this surely is that deploying sensor nodes is not as easy as expected. In fact, most larger deployments in the context of research suffered unpredicted and severe problems [1, 2]. The trouble starts with the distribution of nodes in the wild, continues due to environmental condition changes and ends in bricking the node with the firmware update that was supposed to improve the situation.

A good way of reducing the chance of such troubles is simulating and testing the network before deploying it. One of the most reliable ways of testing is to set up a test-network in the lab. Unfortunately toggling LEDs can give only very limited information on what is going on in the network and dumping information using a side-channel like the serial port can have major probe effects, that is, by exporting the data the node's behavior is changed. Using debug-hardware provides a better insight into what is going on on a node, but has the disadvantage that the other nodes are not halted and might time out waiting for an answer. Especially when debugging, a WSN-simulator can mitigate these problems. It is possible to debug not only a single node, but the network as a whole. Setting up arbitrary network topologies and scripting different scenarios or test cases can be done with minimal effort.

SenSys'11, November 1–4, 2011, Seattle, WA, USA.

2 Problem Statement

When deploying WSNs, it is best practice to test and simulate your system before deploying it [1]. None the less, in the deployment the WSN software seldom shows the same behavior as in the tests. A main issue is that neither simulation nor testbed resemble the real deployment. This is the case for the topology, as well as environmental sensor readings. As a consequence algorithms working fine during testing behave odd once executed in the real deployment and finding the cause is likely to become tedious.

We therefore suggest to use *deployment-tailored simulation* to allow a *deployment-targeted development*, where adjusting parameters and code for each deployment is the rule and not the last resort. If configured correctly, a simulator can give a reasonable good resemblance of a real deployment [6]. Using such a simulator can not only help to understand why a certain algorithm behaves the way it does, but also allows to detect routing issues, network congestion, hot-spots causing reduced lifetime or insufficient resources for in-network-processing. Even automated parameter optimization on a per-node basis is possible. But, although most WSN simulators provide the means to configure complex network configurations this is seldom used outside the domain of network protocols.

To apply this approach detailed knowledge about the deployed network is needed. Unfortunately, it is almost impossible to make accurate a-priory assumptions about the network topology. Physical obstacles, a change in temperature or humidity may influence the connectivity of the nodes [3]. Even when deploying nodes in a hall, with an almost ideal environment, the signal strength had to be adjusted to get reasonable predictions [10].

It has been shown that a simulator-supported deployment-targeted development is possible by configuring the simulator accordingly, as well as extracting the information to improve the behavior of the deployed network. None the less, both steps are often laborious. This opens three fields for research that work hand in hand: How can the process of mapping the real network to the simulator be automated and improved? And, how can the simulation results put to best use in the analyzed network? The third field, connecting those two, is the relation between the accuracy of the acquired data and the quality of the conclusions drawn from the simulation results.

3 The RealSim/DryRun architecture

To help solve these questions we suggest to develop a tool-set for each problem domain: RealSim, which will support the mapping of deployed networks to the simulator, and DryRun, which will help to analyze, visualize and optimize the deployment using the simulator (Fig. 1) . RealSim is split up into an application running on the WSN nodes that collects relevant information (e.g. network connectivity), and a toolchain running on a PC that is able to collect the data and process it, so it can be used to configure the simulator. DryRun is a set of tools that use the simulator configured by

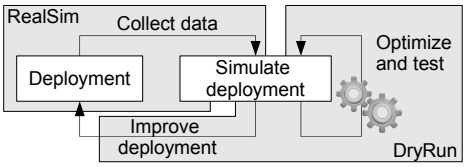

Figure 1. RealSim/Dryrun architecture

RealSim to help draw conclusions on how to improve the deployment. These tools cover a wide range of use-cases like testing on non-functional properties, parameter optimization or visualization.

As connectivity and the environmental sensors are the main factors that influence a node's behavior, these must be modeled in the simulation. Environmental sensors normally have a predictable range of values that can be, at least roughly, modeled without the need of actual sensors. Opposed to that, as already illustrated, it is very hard to make accurate predictions about the connectivity of a certain network without actual measurements, and can be considered impossible with high mobility. We will therefore concentrate on deployments with minimal mobility that change at low rate and thus allow to make reasonable predictions or to track the changes with a minimal load to nodes and network. Temporary disturbances in connectivity like elevators or persons passing by are considered representative if data is captured over a sufficient amount of time. Finally, generators for unexpected events like sudden failures, random reboots, or clock skew can be derived from previous deployments.

In the first phase of the project the link quality between the different nodes will be estimated sampling the Received Signal Strength Indicator (RSSI), Link Quality Indicator (LQI) and packet loss. The link quality can either be actively probed sending packets, or passively monitored using the data sent over the network by other applications. The goal is, of course, to have an hybrid solution that uses passive probing by default and can manually, or even automatically, be switched to active probing if a change in the topology is suspected. In a second step it must be evaluated whether the results can be improved by acquiring properties like clock skew or sensor readings. It is also likely that the amount of data that must be transferred to the sink can be reduced by finding certain characteristics that reduce the amount of data needed to describe the connection.

Even before using the gathered information to calibrate the simulator, DryRun can already use it to visualize the network. The topology can, for example, be visualized by distancing the nodes based on their connectivity using a spring graph layout (Fig. 2), or by placing the nodes manually on a map and visualizing the connection quality. Especially the latter can be a helpful tool during the deployment as the user can easily see that something is wrong if two nodes close together have a bad connection or that more nodes are needed if they are sparsely connected. Tools targeting routing or graphing as used in the bioinformatics domain can support this process as they already provide comprehensive interfaces for visualization.

Once the simulator is configured to resemble the real network, the software can be tested thoroughly. To support data analysis DryRun provides a workbench of several modules. Yet, before analyzing the data from the simulations the data must be acquired. The typical success/fail test provided by most simulators is not sufficient. On the other side, just saving everything is not acceptable either, as this will quickly produce an unmanageable amount of data and slow down the simulation needlessly. DryRun therefore provides a data acquisition module to select and preprocess the data to be saved to a database for analysis. The nature of the preprocessing strongly depends on the requirements. In one case it might of interest to save the maximum stack pointer for a given time slice, in another to save it for every function call. Besides accessing variables by name, using debugging information, dereferencing pointers and allowing access to data saved in structures, it also allows implementing simple programs to decode more complex data structures. Monitoring function calls or the execution of a certain instruction can also help to detect unwanted behavior.

Once the data is acquired and saved, DryRun's data analysis module provides the possibility to graph the data or test against certain rules (e.g. a certain function should only be called every ten seconds). Tools from the data-mining domain, like cross-correlation, can help to find the cause for an unexpected event.

A problem often faced when planning a deployment is finding optimal settings. Especially tuning the network layer can often only be done by a domain expert with an in-depth knowledge of the application running in the network, in particular because the nodes might need different configurations depending on their position or function within the network. DryRun therefore provides a module to explore a parameter space, and thus supports finding a good configuration for each node within the deployment.

Further more, even after the deployment DryRun supports the maintenance of the network. By contentiously monitory the network using RealSim, changes to the network topology can be detected automatically and using DryRun also their consequences. By comparing a simulation run of the current network topology with an older run, DryRun can not only detect problems one is explicitly looking for (e.g. energy usage and package loss), but also those not considered: For example, a function suddenly being called significantly more or less often might forebode that something is going wrong under the hood.

The same strategy can be used during the continuous development of software. Testing each revision against the reference-run can even help to detect the exact change that caused a regression or improvement. Of course, the results only apply to the distinct WSN deployment tested against: The same change might have a totally different impact on the behavior of another deployment.

4 Related Work

To analyze the behavior of network protocols classic network simulators like NS-2 or OMNeT++ have been extended to the WSN domain [10, 13]. As the code must be ported, or at least be recompiled, to run in these simulators, no timing assumptions for the real node can be made, that is, delays or bottlenecks caused by the limited processing power of a sensor node cannot be detected. TOSSIM, which was developed especially for WSNs, can be used to produce quite accurate results when configured to match a real network [5], but does not work at instruction level, either.

Simulators like COOJA [7] or WSim/WSNet [4] also target cycle accurate WSN simulations. While COOJA is rather a meta-simulator that connects different simulators like MSPsim[1] or Avrora[2] and connects them with different network models, WSim implements different nodes and uses the WSNet network simulator to simulate the network traffic.

KleeNet [11] is a powerful extension to COOJA, which finds bugs using symbolic execution. Opposed to DryRun it targets pre-deployment testing.

NetTopo [12] is a simulator that has a strong emphasis on visualizing the network. Opposed to the ideas presented, its focus is on visualizing sensor readings of real networks and analyzing routing protocols in the simulator, rather than the quality of the network connections themselves. There have also been approaches to

[1] `http://www.sics.se/project/mspsim`

[2] `http://compilers.cs.ucla.edu/avrora/`

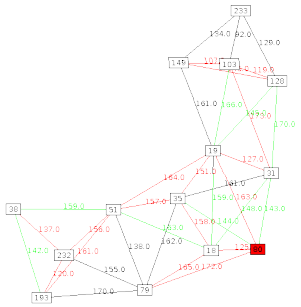

Figure 2. Spring layout of the network deployed at the department. Bad connections are spaced further apart.

support the developer by visualizing simulation events in a timeline [9]. These are programmed for a very specific use-case and can be deviated from the data collected by DryRun.

Österlind et al. [8] present an approach that is orthogonal to our concept of mapping a real deployment to a simulator. They implemented a check pointing mechanism that allows stopping the whole system, save its state and roll it back, thus allowing to move the state of real sensor nodes, but not the network topology, to and from the simulator.

5 Current State, Findings and Future Work

RealSim currently supports automatic detection of nodes and links by active probing. Detected nodes and links are automatically added to or removed from the COOJA simulator, which was chosen for its easy extendability, even while a simulation is running. The nodes are placed within the simulator using a spring layout (Fig 2). To represent the links, Cooja's Directed Graph Radio Medium (DGRM) is used, which allows to set RSSI, LQI and packet loss for each unidirectional connection between the nodes. While RealSim already produced first results, DryRun is still at an early stage. We added some additional interfaces to COOJA to access as much data as possible efficiently and save it to a database.

Using RealSim we were able to see that the topology of a testnetwork deployed in our department did not resemble our expectations at all: While we did expect some asymmetry, we did not expect it to be so pronounced. We witnessed nodes with extraordinary good reception and such with very bad reception, having trouble to receive data from nodes an arm's length away. We were able to relate the regular loss of routes to this, as the algorithm assumed that, if it can receive packets from a node, it can also send packets to that node directly.

While RealSim has proven to work, it must undergo thorough evaluation. It is likely that the probing mechanism, as well as the simulation model can be improved significantly. To allow running RealSim as a background service, methods to minimize the impact of acquiring the data must be investigated. For DryRun, the development will continue in two different directions: Firstly, using arbitrary test cases, we want to extract and visualize the evolution of non-functional properties (e.g. packet loss) using the history of a WSN operating system like Contiki. Secondly, we will try to prove that the idea as a whole – improving the quality of a deployed network using RealSim and DryRun – holds, by building a simple sensing application and optimizing the network parameters to fit the deployment.

In combination RealSim and DryRun will not only help to avoid failures of WSN deployments, but will also help to get a better understanding of how and why WSNs often do not behave as expected when used in real world scenarios.

6 References

[1] G. Barrenetxea, F. Ingelrest, G. Schaefer, and M. Vetterli. The Hitchhiker's Guide to Successful Wireless Sensor Network Deployments. *Proceedings of the 6th ACM conference on Embedded network sensor systems (SenSys 2008)*, pages 43–56. 2008.

[2] J. Beutel, K. Römer, M. Ringwald, and M. Woehrle. *Sensor Networks*, chapter Deployment Techniques for Sensor Networks, pages 219–248. Signals and Communication Technology. 2009.

[3] C. A. Boano, J. Brown, Z. He, U. Roedig, and T. Voigt. Low-Power Radio Communication in Industrial Outdoor Deployments: The Impact of Weather Conditions and ATEX-compliance. *Proceedings of the 1st International Conference on Sensor Networks Applications,Experimentation and Logistics (Sensappeal 2009)*, pages 159–176. 2009.

[4] A. Fraboulet, G. Chelius, and E. Fleury. Worldsens: development and prototyping tools for application specific wireless sensors networks. *Proceedings of the 6th international conference on Information processing in sensor networks (IPSN 2007)*, pages 176–185. 2007.

[5] G. P. Halkes and K. G. Langendoen. Experimental evaluation of simulation abstractions for wireless sensor network MAC protocols. *EURASIP Journal on Wireless Communications and Networking*, 2010:24:1–24:2, 2010.

[6] D. Kotz, C. Newport, R. S. Gray, J. Liu, Y. Yuan, and C. Elliott. Experimental evaluation of wireless simulation assumptions. *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems (MSWiM 2004)*, pages 78–82. 2004.

[7] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-level Simulation in COOJA. *European Conference on Wireless Sensor Networks (EWSN 2007), Poster/Demo session*. 2007.

[8] F. Österlind, A. Dunkels, T. Voigt, N. Tsiftes, J. Eriksson, and N. Finne. Sensornet Checkpointing: Enabling Repeatability in Testbeds and Realism in Simulators. *6th European Conference on Wireless Sensor Networks (EWSN 2009)*. 2009.

[9] F. Österlind, J. Eriksson, and A. Dunkels. Cooja TimeLine: a power visualizer for sensor network simulation. *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems (SenSys 2010)*, pages 385–386. 2010.

[10] H. N. Pham, D. Pediaditakis, and A. Boulis. From Simulation to Real Deployments in WSN and Back. *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2007)*, pages 1–6. 2007.

[11] R. Sasnauskas, O. Landsiedel, M. H. Alizai, C. Weise, S. Kowalewski, and K. Wehrle. KleeNet: discovering insidious interaction bugs in wireless sensor networks before deployment. *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN 2010)*, pages 186–196. 2010.

[12] L. Shu, M. Hauswirth, H.-C. Chao, M. Chen, and Y. Zhang. NetTopo: A framework of simulation and visualization for wireless sensor networks. *Ad Hoc Networks*, 9(5):799 – 820, 2011.

[13] I. Stojmenovic, A. Nayak, J. Kuruvila, F. Ovalle-Martinez, and E. Villanueva-Pena. Physical layer impact on the design and performance of routing and broadcasting protocols in ad hoc and sensor networks. *Elsevier Computer Communications*, 28(10):1138–1151, 2005.

Biography

Moritz Strübe studied Mechatronical Engineering at the University of Erlangen-Nuremberg and graduated in 2008 (Diploma Degree). During his studies he spent half a year as intern at the Shadow Robot Company, London, participating in the development of the Smart Motor Hand. In his diploma theses he investigated possibilities of extending Nut/OS on BTnodes. This was accomplished using over-the-air programming of pre-linked modules at run time. In this context he also implemented a mechanism to recover data saved in heap memory after a hardware reset, thus allowing to tolerate watch-dog resets or updates using a boot loader, without data loss.

After graduating he started working as embedded software developer for a small company. Due to the global economy crises in 2009 the employment lasted only half a year.

He accepted a research position at the Department of Computer Science 4, Distributed Systems and Operating Systems at the University of Erlangen-Nuremberg. He is supervised by Dr. Rüdiger Kapitza and Prof. Dr. Wolfgang Schröder-Preikschat. He expects to finish his theses in 2014.