Monitoring and Debugging of Deployed Sensor Networks

Matthias Ringwald, Kay Römer

Dept. of Computer Science
ETH Zurich, Switzerland
{mringwal,roemer}@inf.ethz.ch

1 Introduction

As researchers actually start to deploy Wireless Sensor Networks (WSNs) in the real-world, they are confronted with misbehaviors and problems not detected or seen during previous simulations or lab tests. Monitoring and debugging a deployed sensor network is often more difficult and more expensive than earlier in-house tests as the network might be hard to reach or located in an human-unfriendly environment. Furthermore, as these networks are optimized to fulfill their task with minimal energy consumption, it is hard to collect enough information to understand and solve such problems. The gathering of exhaustive information for debugging purposes would require changes to the distributed application thus modifying its behavior and might lead to additional bugs. In this extended abstract, we present our preliminary work on non-intrusive debugging of WSNs, which, as we envision, will shorten and simplify the deployment phase of future WSN applications.

2 Related Work

Although there is a multitude of tools available to simulate or emulate sensor nodes and sensor networks on various levels, not much is available to aid in monitoring and debugging of deployed sensor networks.

The Nucleus network management system (NMS) [3] facilitates monitoring of running WSN applications by providing access to the internal data structures of TinyOS nesC components over the network. To reduce interference of the Nucleus system and the WSN application, data is only sent over the network in response to a user query. In addition to this query approach, unexpected events are logged to persistent local storage and can be retrieved later from the node on demand.

In contrast to this single node debugging approach, Sympathy [2] focuses on the collection of various metrics on the sensor node to detect and identify failures in the network. The collected metrics are routed by the sensor network itself to a centralized sink where they are evaluated. Sympathy's failure detection and localization is based on the assumption that a continuous data flow exits in a working network. If less traffic is monitored than expected, a failure is reported.

Sympathy uses the collected metrics to localize the failure following a decision tree based on previous experience and heuristics.

Both methods require to run additional program code on the sensor node and create additional network traffic. They are implemented for a specific system software platform. To minimize the changes and the impact to the sensor nodes, the concept of a deployment-support network (DSN) [1] has been proposed to provide the benefits of a wired testbed to a deployed wireless sensor network. An additional DNS node can be attached to some or all WSN nodes. It then allows to reprogram and restart the node, measure its battery level and provides access to a local communication port for single node debugging or network-wide log message collection. Our approach makes use of an additional DSN that is deployed in the same area as the deployed WSN, but instead of a direct connection to a node, we use the radio receiver of the DSN nodes to realize a network-wide distributed sniffer.

3 Non-Intrusive Monitoring and Debugging of WSNs

In order to gain insight into deployed WSN, we investigate the usefulness of network-wide observation of exchanged radio communication. We want to provide hardware and software tools which make use of this observation to allow for non-intrusive monitoring and debugging of sensor networks. In the following sections, we show how sensor network communication can be observed, decoded and evaluated without changes to the deployed network.

3.1 Distributed Sniffer

The inherent broadcast characteristic of wireless communication makes it easy to eavesdrop on exchanged messages in the neighborhood given that a compatible radio transceiver is available and some basic knowledge about the radio communication such as bit rate, frequency and the packet format is provided. For our own sensor network, we have this information available. As we do not want to participate actively in the message exchange, it is sufficient for a sniffer node to continuously listen to incoming radio messages regardless of the implemented Medium Access Protocol of the sensor network. If a message is transmitted without collision, we can safely assume that we will receive it as good as a sensor node at the same location would.

We assume that the DSN nodes are more powerful and can also use more energy then the sensor node to fulfill their observation task. For example, we use our BTnode rev3 [4] as a DSN node. Its Chipcon CC1000 radio module is also used on the second generation of the Berkeley Motes (mica2 and mica2dot nodes). In addition, the BTnode contains a Bluetooth module and 256 KB RAM. The Bluetooth module provides fast and reliable backbone communication, whereas the extra memory allows to develop more complex applications and to queue received packets.

To observe parts or a complete WSN such as shown in Fig. 1, we add our deployment-support network (Fig. 2).

Besides knowing hardware configuration parameters such as radio frequency and data rate, it is necessary to detect its start and end to correctly receive a packet. Most radio protocols use a start-of-packet symbol that is transmitted after the preamble. The length of the packet itself is either fixed for a given application or can be dynamic. For dynamic packet lengths, it is customary that a length byte is transmitted in the packet header which tells how many bytes will follow. Knowing the position of this length byte is therefore sufficient to correctly receive packets with variable lengths. All received packets are forwarded on the DSN to the monitoring sink. Improvements on this, like filtering, in-network processing etc, could be future work.

If a node's transmission can be received by multiple DSN nodes, it is necessary to distinguish this event from separate transmission of the same packet received by a single DSN node. We are confident that the time-synchronization of the DSN is sufficient for this.

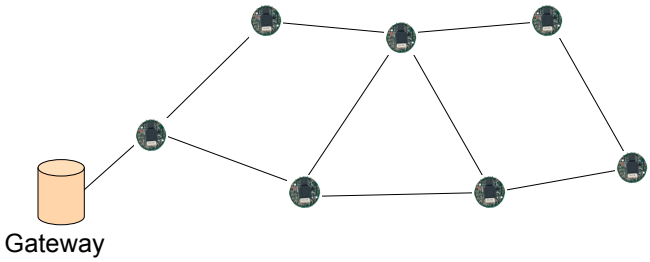

Fig. 1. Deployed Wireless Sensor Network

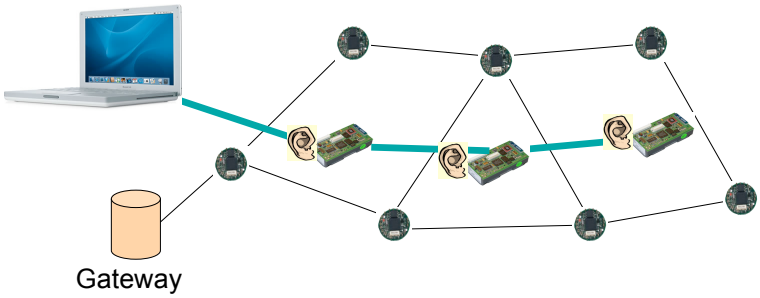

Fig. 2. WSN with additional deployment-support network

3.2 Packet Decoding

Mere reception of the raw data packets does not provide much insight into the WSN. The data packets need to be decoded to be useful for both humans or further automatic processing. Although there are many packet description languages available, most notably ASN.1, we could not find a simple way to decode binary packets into attribute-value pairs.

Our approach is based on the observation that packet layout is often implemented as nested structures data types (structs) in application developed in C or nesC. By augmenting these structured data types with additional meta data, we can extract packet layout from the application code. Using these packet descriptions, we can decode sniffed messages.

3.3 Data Analysis

After the packet decoding, all relevant information is available for further processing. Here we will start by using {source, destination} pairs in the lowest layer packets to reconstruct the network topology and present it to the user. If packet sequence numbers are used, we can count the number of times a packet is transmitted and spot regions where data is re-transmitted more often than in other areas. Taking routing packets into account, it should be possible to present routing paths and collect statistics on route changes due to variable link quality.

The proposed approach will facilitate further data data analysis methods. We plan to incorporate a rule-base engine to allow the specification of distributed assertions and tests on the spot.

4 Summary

We have given an overview of our proposed approach to support monitoring and debugging of deployed wireless sensor networks which differs from existing solutions by its applicability to an already deployed network and its non-intrusiveness. A deployment-support network provides the ability to collect messages of the deployed network itself. Augmented C or nesC code serves as a packet description to decode theses messages. The obtained data can then be uses to reconstruct the network topology, to collect packet transmission statistics an, and to implement distributed assertions.

5 Acknowledgments

The work presented in this paper was supported (in part) by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

References

1. J. Beutel, M. Dyer, L. Meier, and L. Thiele. Scalable topology control for deployment-sensor networks. In *Proc. 4th Int'l Conf. Information Processing in Sensor Networks (IPSN '05)*, pages 359–363, 2005.
2. Nithya Ramanathan, Kevin Chang, Rahul Kapur, Lewis Girod, Eddie Kohler, and Deborah Estrin. Sympathy for the sensor network debugger. In *Proc. 3rd ACM Conf. Embedded Networked Sensor Systems (SenSys 2005)*, November 2005.
3. Gillman Tolle and David Culler. Design of an application-cooperative management system for wireless sensor networks. In *EWSN 2005*, Istanbul, Turkeye, Jan 2005.
4. BTnodes - a distributed environment for prototyping ad hoc networking. http://btnode.ethz.ch.