Poster Abstract: Modular Communication Protocols for Sensor Networks

Olaf Landsiedel Jó Ágila Bitsch Katharina Denkinger Klaus Wehrle

Protocol Engineering and Distributed Systems Group
University of Tübingen, Germany

firstname.lastname@uni-tuebingen.de

ABSTRACT

In this paper we present our ongoing work on modular communication protocols for sensor networks. Their modularity allows recomposing a protocol dynamically at runtime and adapting it to the changing needs of a sensor network. Compared to existing work, our componentization is fine grained and protocol independent, enabling a high degree of component reusability.

1. INTRODUCTION

Wireless sensor networks – consisting of small, often battery powered embedded nodes – are able to sense the environment in a distributed fashion and so accomplish tasks that previously were too complex or expensive. Sensor nodes can be deployed without network infrastructure and far away from human access: anywhere from the forest canopy [6] to the backs of zebras [5].

As it is difficult and expensive to maintain such distant deployments, a sensor network is expected to be autonomous and long-lived. Thus, the network is required to adapt to environmental changes as well as changes in the network topology, when nodes fail or cannot reach each other anymore, by reconfiguring the communications protocols and sometimes even the applications. Furthermore, during deployment the sensor network's needs can change, too, as data sampled by the sensor network influences follow up experiments.

In this paper we discuss the case of reconfigurable communication protocols for sensor networks and our ongoing work in this area. Furthermore, we argue that other components of the sensor network operating system do not require such an amount of flexibility. Thus, compared to the SOS [3] operating system we do not propose a modular scheduler or memory management.

The remainder is structured as follows: Section 2 discusses the case of modular communication protocols. Next, section 3 introduces our fine grained modules for protocol building. Section 4 discusses related work and section 5 concludes.

2. THE CASE FOR MODULAR PROTOCOLS IN SENSOR NETWORKS

In this section we motivate the use of modular and reconfigurable communication protocols in sensor networks, see fig. 1. Commonly sensor network deployment and maintenance consists of several steps: (1) The sensor network is deployed: Via flooding a sensor node determines its position in the network and announces its existence to the surrounding nodes. (2) Based on its position in the network, various tasks are assigned to a node: while data collection and forwarding are commonly assigned to a huge number of nodes, selected nodes take care of data aggregation [4, 8] or act as routing beacons [1, 2]. (3) During deployment the conditions change. Due to node failure or other environmental influences – such as changing radio propagation – nodes take over tasks from other nodes. Furthermore, nodes are retasked based on data sampled in previous measurements to adapt their functionality to new upcoming needs.

Today's sensor node operating systems [7] and their applications are statically linked at compile time. This approach allows to use code optimization and resource facilitation analysis. However, all functionality that might be used during deployment needs to be compiled into the binary at compile time. Furthermore, updates while a sensor network is deployed become very costly, as a whole binary needs to be redistributed. Thus, modular communication protocols can be of high benefit for sensor networks.

3. THE MODULES

In this section we discuss the modules that are used to compose a communication protocol. When analyzing various communication protocols we identified the following key properties: (1) A module shall present protocol independent functionality, for example to set certain bytes in a packet. (2) A configuration string at runtime or compile time specifies the exact functionality, making a component protocol dependent. (3) All component interfaces (in- and out-ports) are standardized to ensure that components can be combined arbitrarily.

Based on these above described properties, we designed our modular communication protocols. The components can be grouped into five main groups: Source, sink, operational, validation and de-multiplex. In the source group are all components that emit packets into the protocol, i.e. the incoming network interface, the application, and timers which omit packets at certain intervals. Similar, the sink class represents outgoing network interfaces, the application and

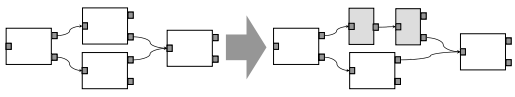

Figure 1: Modular and reconfigurable protocols allow for dynamic changes.

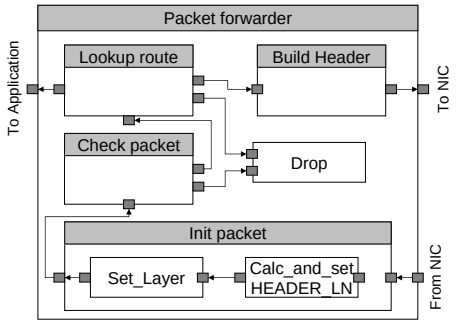

Figure 2: A packet forwarder build from compound and simple components

packet droppers. The operational components change the packet's header, payload or additional options like the outgoing device. Validation components check certain parts of a packet, based on the result they emit it to one of their outgoing ports. This class of components splits a packet flow into multiple flows. The de-multiplex component merges flows.

From these component classes we derived the individual components. Thus, each component has dedicated functionality which ranges from setting bits or bytes and computing a checksum to storing the system state in a so called blackboard.

Compound modules are used to group components into functional and semantic groups. Compound modules are either protocol dependent or protocol independent. A protocol independent compound for example is a loop, which parses a packet for a certain bit or byte pattern. In the case of the IP protocol it can be used to parse for IP-options. Protocol dependent compounds are used to group functionality and make the protocol description more readable. A typical compound is a compound which builds a protocol header, see fig. 2.

To allow for easy component and protocol development, we have implemented a compiler and GUI which derive from a meta language the right modules to use and concatenate and configure these accordingly. Space limitations prevent us from discussing these features in more detail.

The work discussed in this paper is ongoing work. As today's sensor network protocols are kept quite simple, we derived our modules from the more complex Internet protocols, e.g. IP and TCP. Currently we use the modules derived from the Internet protocols to build sensor network protocols ranging from tree-based routing to data aggregation.

4. RELATED WORK

In this section we discuss the existing works on modular communication protocols and compare our work to them. Modular protocols have been previously presented for the use in the internet. Click [9] is a modular software router. However, most of the Click modules present IP specific functionality. In our approach modules are protocol independent, a configuration at run- or compile-time makes their behavior protocol specific. As result, our modules can be reused for various protocols. KIDS [10] provides a modular QoS system, similar to Click it focuses on a certain protocol type, in this case on QoS functionality.

The Sensor Operating System (SOS) [3] introduces a modular operating system for sensor nodes. It allows to change major parts of the OS dynamically at run-time. However, from our point of view, OS components like the scheduler do not need to be changed at runtime. Thus, we propose a more lightweight approach. Furthermore, as our approach focuses on protocols only, it provides a more fine grained approach.

5. CONCLUSION

In this paper we presented our ongoing work on modular communication protocols for sensor networks. We introduced our fine grained approach to protocol independent modules which makes it applicable to wide range of different communication protocols.

Next to the implementation of various sensor network protocols our ongoing implementation efforts focus on two topics. First, it might be interesting to implement the modules on various platforms and even add a platform abstraction layer. Thus, the modules can be run on various systems. As a result, one can test a protocol by using modules implemented for a simulator and then use the same already evaluated and tested configuration for a sensor network. Furthermore, we consider it highly interesting to evaluate how protocol verification techniques can be applied to the meta language describing component configuration and concatenation.

6. REFERENCES

[1] Q. Cao and T. Abdelzaher. A scalable logical coordinates framework for routing in wireless sensor networks. In *Proc. of IEEE RTSS*, 2004.

[2] R. Fonseca, S. Ratnasamy, D. Culler, S. Shenker, and I. Stoica. Beacon Vector Routing: Scalable Point-to-Point in Wireless Sensornets. In *Proc. of NSDI*, 2005.

[3] C.-C. Han, R. Kumar, R. Shea, E. Kohler, and M. Srivastava. A dynamic operating system for sensor nodes. In *Proc. of MobiSys*, June 2005.

[4] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proc. of ACM MobiCom*, 2000.

[5] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. In *Proc. of ASPLOS*, October 2002.

[6] W. Kaiser, G. Pottie, M. Srivastava, G. Sukhatme, J. Villasenor, and D. Estrin. Networked Infomechanical Systems (NIMS) for Ambient Intelligence. In *Invited contribution to Ambient Intelligence, Springer-Verlag*, 2004.

[7] P. Levis et al. The Emergence of Networking Abstractions and Techniques in TinyOS. In *Proc. of NSDI*, March 2004.

[8] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks. In *Proc. of OSDI*, Dec. 2002.

[9] R. Morris, E. Kohler, J. Jannotti, and M. F. Kaashoek. The Click modular router. In *Proc. of SOSP*, 1999.

[10] K. Wehrle. An Open Architecture for Evaluating Arbitrary Quality of Service Mechanisms in Software Routers. In *Proc. of ICN*, 2001.