Distributed Passive Monitoring in Sensor Networks

Falko Dressler, Rodrigo Nebel and Abdalkarim Awad
Autonomic Networking Group, Dept. of Computer Science 7, University of Erlangen-Nuremberg, Germany
Email: dressler@informatik.uni-erlangen.de

***Abstract*—Operation and control in wireless sensor networks (WSN) demands for new concepts and strategies such as distributed behavior control and self-organization. During the development and operation, the verification of the implemented algorithms is usually hard to discover. Monitoring techniques are required for this purpose. We present a concept for passive monitoring of WSN. Our hierarchical architecture allows a distributed monitoring and a subsequent analysis of the network traffic in the sensor networks. Basically, we employ sensor nodes with two radio interfaces. The first one is used to passively intercept radio packets in order to prevent any impact on the observed network behavior. The other one sends received information to the next level in the monitoring hierarchy towards a central analysis station.**

I. Introduction

The communication in wireless sensor networks (WSN) is often complex and in some cases difficult to predict. Especially during the development of WSNs, methods for analyzing and debugging communication methods are strongly demanded. Usually, only the behavior of single sensor nodes can be supervised using directly attached debugging interfaces. Therefore, the communication between nodes can only be estimated if all participating nodes can be analyzed simultaneously. A second problem lies in the operation and control of already deployed sensor networks. In failure situations, tools are needed to analyze the behavior of the network as a whole.

In this work, we present an architecture and a tool to passively monitor sensor networks. We are able to intercept all radio packets in the network, and to store and transmit them to a central computer for further analysis. The transport (and the coordination of the monitoring environment) is accomplished in a hierarchical way. For simplified analysis, we developed a plugin for Wireshark[1] for graphical analysis.

In network monitoring, *passive* and *active* monitoring techniques are distinguished. In general, active monitoring refers to the active interaction with the system under observation. Thus, the system behavior is being influenced by the monitoring actions. In contrast, passive monitoring means the access to the network traffic without interfering the communicating systems. Nevertheless, additional hardware is required for this purpose. The obtained data can be used for several aims. We are focusing on passive management and control of sensor networks and on enhanced debugging during protocol development.

The following approaches have been described in the literature. The nucleus network management system (NMS) [1] represents a set of TinyOS[2] components that can be integrated into developed applications. These components frequently deliver information to a control unit, which can determine the behavior of the sensor nodes. The NMS allows to configure the frequency and event types. Sympathy [2] is an active monitoring system in which the sensor nodes are supplied with an additional piece of software. All sensor nodes periodically send local information to a dedicated sink node. Using this information, the Sympathy sink can analyze the behavior of the network, detect failures, and localize these failures. The raising demand for sensor network monitoring encouraged also the ScatterWeb project [3] to include monitoring techniques. The integrated tool ScatterViewer allows to manage sensor nodes and to collect status information. It also represents an active monitoring application that may influence the network behavior. A passive monitoring environment has been developed, which was named TWIST [4]. It collects information from the participating sensor nodes using an USB-based cable-network. This architecture allows to specifically debug all connected nodes while it does not support the collection of radio messages. Additionally, an expensive infrastructure (USB cables) must be provided.

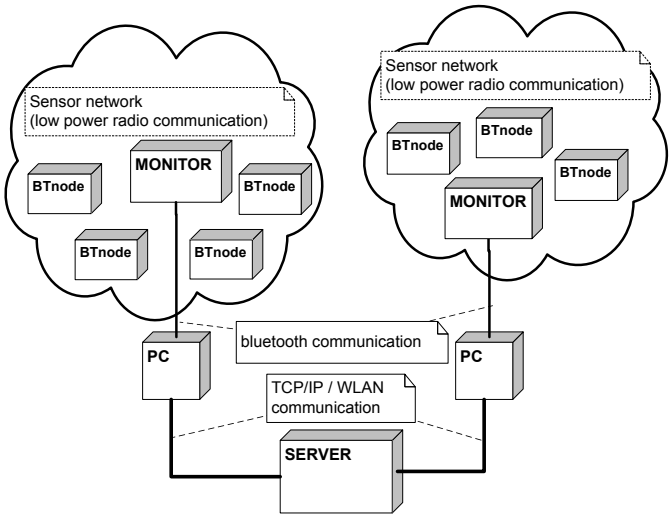

Fig. 1. System architecture for distributed passive monitoring in WSN

II. pimoto – Passive Island Monitoring Tool

In order to enable a comprehensive monitoring in WSNs, we developed *pimoto*, a distributed monitoring environment for passive monitoring in sensor networks. In the following, we describe its architecture and characteristics.

[1] http://www.wireshark.org/
[2] http://www.tinyos.net

A. Architecture

Two aspects should be supported by our passive monitoring system. First, we want to allow hierarchical monitoring, i.e. multiple deployed monitoring nodes have to be interconnected by a network separate from the WSN. Secondly, the collected packet data should be visualized at the central server in real time, i.e. the latency from monitoring, transmission, and analyzing should be minimized.

The hierarchical structure is depicted in figure 1. As can be seen, monitoring nodes are placed inside a sensor network. These nodes collect all the radio traffic. Using a second radio interface (in our case, we employ Bluetooth), the monitoring data is delivered to a PC, which may control multiple monitoring nodes. The PC forwards the data using a TCP/IP network, e.g. WLAN-based, to a central server. At this place, the standard network monitoring application Wireshark is used to receive, decode, and visualize the packet information.

Two options can be used to transmit the received monitoring data from the monitoring nodes to the server: push and pull. We decided to use the push technique as the storage capabilities of the employed monitoring nodes (actually, these are typical embedded sensor nodes) is strongly limited. Thus, the intercepted radio packets must be forwareded as soon as possible to re-use the memory for further data. Additionally, the demanded real-time behavior of the analysis as well as the typically very low data rates in WSN contributed to our decision.

B. Implementation

For implementing pimoto, we used the BTnode sensor nodes[3]. This sensor node incorporates two radio interfaces, the Chipcon CC1000 for low power radio transmissions and a Bluetooth interface. We developed a sensor program that sets the radio interface in the *promiscuous mode*, i.e. in a mode that allows to capture any packet regardless of its destination address. As this capability is currently not supported by the used B-MAC protocol implementation, we extended the BTnut driver accordingly. For the Bluetooth communication, we employed the rfcomm protocol. It represents a simple serial connection between the sensor node and the PC. This PC forwards all received monitoring data to the server using TCP/IP.

Synchronization of the timestamps that are stored with each captured packet has been proven to be the most challenging issue. We need accurate timestamps in order to (re-)order the received packets in the analysis stage. Unfortunately, the clocks of the monitoring nodes cannot easily be synchronized. Therefore, we used a trick to synchronize the timestamps in the second hierarchy, i.e. at the controlling PCs. Each sensor node provides a 4-byte counter showing the milliseconds since its last reboot. Thus, the uptime of the sensor node can be exactly measured for 24 days (then, the counter overflows). We used this uptime measure as the recorded timestamp in each collected packet. Then, prior submission to the connected

[3]BTnode rev3, http://btnode.ethz.ch/

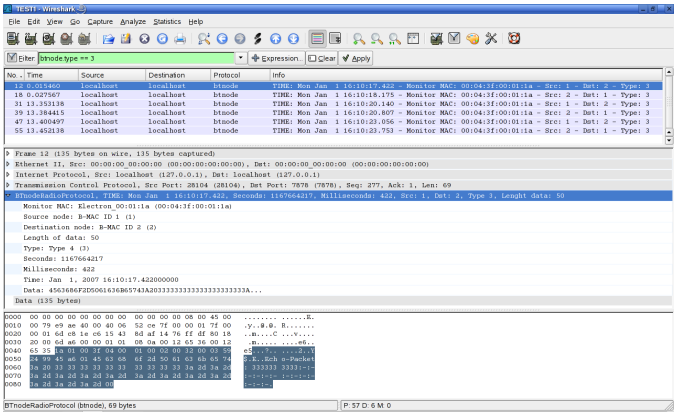

Fig. 2. Network analysis in WSN using Wireshark

PC, the timestamp is subtracted from the current submission time, i.e. depicting the time difference between reception and forwarding. The PC can then update the timestamp according to its, e.g. NTP-synchronized, local time.

At the server, we used the standard monitoring environment Wireshark for data analysis and filtering. Our plugin "BTnode Radio Protocol" decodes and interprets the received data packets. Currently, we support all features from the B-MAC protocol while other elements (and protocols) can easily be added. The graphical analysis is depicted in figure 2. Shown is a decoded B-MAC packet including source and destination address, length, type, timestamp, and payload data.

III. CONCLUSION

In this paper, we depicted pimoto, our passive monitoring tool for distributed monitoring and analysis of WSNs. The primary objectives were to intercept radio data packets in a completely passive way. Additionally, we wanted to support distributed monitoring in a hierarchical way. In conclusion, it can be said that the developed toolkit allows to perform the envisioned management and control tasks, which are usually required in WSN. Protocol development as well as failure detection are simplified by means of graphical protocol analysis.

REFERENCES

[1] G. Tolle and D. Culler, "Design of an Application-Cooperative Management System for Wireless Sensor Networks," in *Second European Workshop on Wireless Sensor Networks (EWSN)*, Istanbul, Turkey, January/February 2005, pp. 121–132.

[2] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin, "Sympathy for the Sensor Network Debugger," in *3rd ACM Conference on Embedded Networked Sensor Systems (SenSys 2005)*, San Diego, California, November 2005, pp. 255–267.

[3] H. Ritter, R. Winter, and J. Schiller, "A Partition Detection System for Mobile Ad-Hoc Networks," in *First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON 2004)*, Santa Clara, California, October 2004, pp. 489–497.

[4] V. Handziski, A. Köpke, A. Willig, and A. Wolisz, "TWIST: A Scalable and Reconfigurable Wireless Sensor Network Testbed for Indoor Deployments," Technical University Berlin, TKN Technical Report TKN-05-008, November 2005.