Demo: Energy-Efficient Status Monitoring in Sensor Networks Using Adaptive Piggybacking

Dominik Neuner, Margit Mutschlechner and Falko Dressler
Computer and Communication Systems, Institute of Computer Science, University of Innsbruck, Austria
{dominik.neuner,mutschlechner,dressler}@ccs-labs.org

***Abstract*—In this demo, we show the capabilities of adaptive status monitoring in sensor networks. We developed an architecture for integrated and adaptive status monitoring that works as a sublayer directly integrated into the network layer. The main advantages of this system, which extends well-known piggybacking concepts, are to make monitoring as energy efficient as possible while maintaining certain real-time requirements for the logged information. Such status monitoring is needed in many application domains that require global status for optimizing the performance of the network. A very prominent example is data stream processing in sensor networks. We demonstrate the main functionality but also the performance improvements compared to related approaches.**

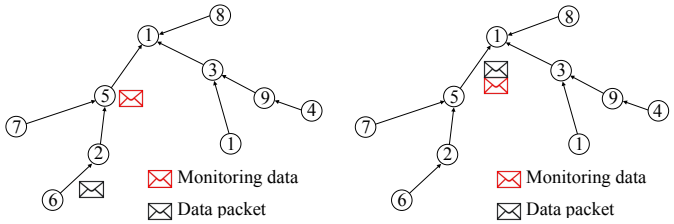
I. Introduction

Evaluating the energy consumption of applications and protocols is an important task to prolong the network lifetime in sensor networks [1], [2]. Erroneous lifetime predictions may cause high costs and may even render a sensor network useless before its purpose is fulfilled. Recent research strongly focuses on power conservation, energy management, and energy-aware applications and protocols. Application specific protocols can be designed by appropriately trading off other performance metrics such as delay and throughput with energy efficiency.

In order to prolong the network lifetime, it is not sufficient to save energy at a local context. For example, the use of data stream processing has been investigated to optimize the energy balance (and, of course, the data processing latencies) in the entire network [3], [4]. Energy models are required for these optimizations [5], and, thus, a *continuous status monitoring*.

Classical network monitoring is performed either via push or pull based mechanisms. Yet, the resource-critical environment in sensor networks prohibits the usage of such complex protocols. Monitoring is generally assumed to be periodic [6]. Many of the reported concepts are more debugging tools than efficient status monitoring concepts [7], [8]. However, transmitting monitoring information in additional separate packets will drastically increase the energy consumption and reduce the network lifetime. Thus, aggregation techniques have been proposed [9].

Piggybacking is a technique often used for acknowledging data in bi-directional communication protocols to reduce the number of transmissions. Packing multiple pieces of information into the same physical packet has the huge benefit of reducing the performance costs in sensor networks [10]. Dunkels et al. introduced the concept of an announcement layer [11], which piggybacks beacons and coordinates their transmission to reduce the total number of transmissions. Status monitoring became even more relevant with the advance of data stream processing in sensor networks [3], [4].



(a) Transmission of a data packet from node 6 to node 2; node 5 has monitoring information available.

(b) Piggybacked packet containing data from node 6 and monitoring information from node 5.

Figure 1. Piggybacking of monitoring information to a received data packet.

In this demo, we present and discuss an *integrated and energy-efficient resource monitoring technique* for self-organizing sensor networks, extending our previous work in [12]. The main advantage of the proposed solution is that it supports both highly energy-efficient transmission of monitoring messages using the well-known piggybacking scheme as well as a timely delivery of status based on a per message aging scheme. Although the transmission and reception of packets with increased length due to the piggybacked monitoring data marginally increases the energy consumption, it is more energy efficient compared to transmitting monitoring data in separate packets [10], [11]. The general drawback is the increased delay (age) for monitoring information, because each node has to wait for the next packet transmission to piggyback its monitoring data. We solved this problem by limiting the maximum (tolerated) age of monitoring data. This can be seen as a *compromise between timeliness and increased energy consumption* due to additional packet transmissions. Our experiments confirm that, for all the investigated network topologies and for all traffic pattern, the energy overhead of our protocol is extremely low.

II. WSN Monitoring Architecture

The monitoring protocol has been integrated into the network layer in order to make it aware of all data transmitted towards the sink node. Without loss of generality, we assume a single sink node for the presentation of the protocol. This can, however, easily be extended by marking both data and monitoring packets as to which sink they need to be transmitted.

Figure 1 outlines the basic operation of our status monitoring concept. In this example, node 6 is transmitting a data packet towards node 1. At the same time node 5 has some monitoring information available (cf. Figure 1a). Instead of transmitting an additional packet with this status information, node 5 is waiting for another packet targeted to node 1 (at least for a threshold time interval). In our example, node 5 piggybacked this monitoring information to the packet from node 6 and transmits the resulting packet further towards node 1 (cf. Figure 1b). The sink separates monitoring information from application layer data and continues processing both separately.

In order to distinguish application layer data from monitoring information, the protocol defines three packet types (a new packet header is used to describe the packet type as well as the number of piggybacked monitoring information messages): DATA packets contain only application layer data. The origin of the application data is defined by the network protocol header. COMBINED packets contain application layer data and monitoring information. Here, a `count` parameter indicates the number of piggybacked monitoring messages. The maximum number of piggybacked monitoring messages depends on the size of the application data in the packet. The origin of each monitoring message is indicated in the piggybacking header. MONITORING packets are used to limit the maximum age of monitoring information, i.e., to ensure timely delivery to the sink node. This packet contains only monitoring messages and `count` indicates their number.

The monitoring protocol uses two timers for the generation of monitoring data messages and for limiting their age: MONITORINGINTERVAL defines the monitoring interval in seconds. Please note that the timer is not responsible for the actual transmission of monitoring information; it only sets a flag indicating its availability to the monitoring sublayer. After successful piggybacking the monitoring data, the flag is cleared. MONITORINGMAXAGE limits the maximum age of monitoring information. If monitoring data is available for this time, the monitoring sublayer sends new packet of type MONITORING towards the sink node.

III. DEMO AND SIMULATION SETUP

For evaluating both the functionality and the performance of our monitoring protocol, we integrated the functionality in the sensor operating system Contiki. This system features the light-weight multihop routing protocol Rime [13], which we used to integrate the monitoring sublayer. We experimentally validated the functionality using TelosB sensor nodes. For larger scale performance evaluation, we used Contiki's cross-level network simulator Cooja [14]. We estimated the energy consumption and the remaining lifetime using the Contiki Energest library. Considering the frequently used TelosB nodes, we assume a supply voltage of 3 V and an initial capacity of 2100 mAh. However, more important is the behavior of the monitoring protocol and the relative values.

A simple application running on each node except of the sink node periodically (we used both constant and bursty packet rates) creates data packets destined to the base station. Each application layer data packet has a size of 40 B, i.e., roughly half of the available payload, and the average packet rate was 100 packet/h. For the bursty traffic, we switched between 540 packet/h and 12 packet/h. For statistical confidence, each simulation was repeated at least 100 times with different random seeds. We also experimented with different values for the MONITORINGINTERVAL of 12 s, 36 s, and 108 s. The 36 s interval corresponds with the data generation rate. The MONITORINGMAXAGE was set to 36 s.

Three different network scenarios (linear, random, and grid) were simulated. The communication range allows only direct neighbors to communicate with each other. We configured static routing to exclude dynamics of rerouting during the simulation. Please note that the longest possible path varies between 15 (linear), 9 (random), and 5 (grid).

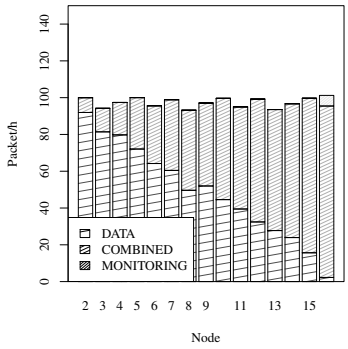
In a first step, we determined the necessary simulation time to obtain results with a high confidence level. Furthermore, we analyzed the initial transient time the protocol spends in its initialization phase. It took about 500 s for the network to get into a steady state. We therefore decided to skip an initialization phase of 600 s before collecting statistical information. To prevent synchronization effects, we added a random delay of 0 min to 5 min when starting up each node. Of course, the energy consumption during the initialization phase has not been considered in the evaluation.

IV. SELECTED SIMULATION RESULTS

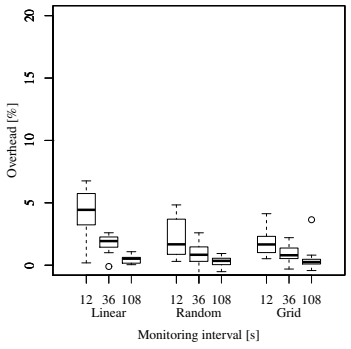
In the following, we presented selected results from our measurements that demonstrate the capabilities of our approach. In a first set of experiments, we evaluated the distribution of packet types received at the sink node. This is of course highly dependent on the scenario as well as on the monitoring interval and maximum message age parameters. In this experiment, each node generated on average 100 data packets during the simulation time of 1 h. Figure 2a shows the distribution of the packet types received at the sink for the linear network and using a constant packet rate and a monitoring interval of 36 s. Packets of type DATA originating at the edge nodes have a high probability that a node along path can piggyback monitoring information, changing the packet type to COMBINED.

The generation of MONITORING packets happens especially at the very edge of the network. The probability depends on the combination of the data rate from the application, the monitoring interval, and the maximum message age. As we allowed for some randomness in the data generation rate (again, to prevent global synchronization effects), some of the monitoring messages aged above the threshold, thus, resulting in the mentioned MONITORING packets. As expected, with a monitoring interval larger than the data generation rate, the number of MONITORING packets is almost zero (data not shown). We found similar patterns also in the random and grid network scenarios (data not shown).

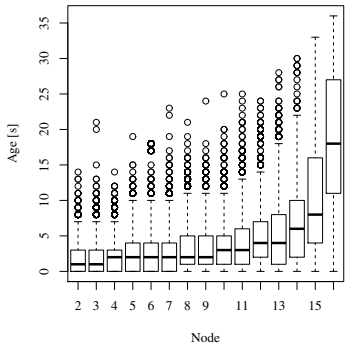
We further executed two sets of simulation experiments to evaluate the overhead of the monitoring protocol in terms of energy consumption. As a baseline, we configured all the nodes to transmit application data only, i.e., we disabled the



(a) Distribution of packet types (linear network, interval of 36 s).



(b) Relative energy overhead of the monitoring protocol



(c) Age of monitoring messages (linear network, interval of 108 s).

Figure 2. Simulation results for the constant traffic pattern

monitoring protocol. In the second set, the fully featured monitoring protocol was enabled. The protocol overhead is composed of creating and piggybacking monitoring messages as well as of transmitting, forwarding, and receiving COMBINED and MONITORING packets (together with the additional MAC layer acknowledgments). Obviously, with an increasing number of forwarded packets the energy consumption for transmitting (TX), receiving (RX), and also for processing (CPU) increases. We are primarily interested in the relative overhead caused by our new protocol.

In general, it holds that the total overhead decreases for larger monitoring intervals because of the fewer monitoring messages. With the bursty traffic pattern, the number of explicit MONITORING packets is higher and, therefore, also the transmission overhead is higher.

Figure 2b shows the observed relative overhead for the constant traffic rate in form of a boxplot. With a large monitoring interval (few monitoring information) the protocol overhead is about the same for all network scenarios. For a monitoring interval of 36 s, the median is at 1.92 % for the linear network and at 0.8 % for the grid network. Independently of the monitoring interval, the protocol performs best in the grid scenario. This is a result of the node distribution and the shorter path lengths.

For evaluating the timeliness of the monitoring information, we use additional age information included in each monitoring message. This time corresponds to the time that the message had to wait until transmission after its creation (limited by MONITORINGMAXAGE). The transmission time along the path to the sink was not measured because it strongly depends on the used MAC protocol.

Figure 2c illustrates the message age plotted for each node in the linear network for constant traffic and a monitoring interval of 108 s. As can be seen, at the edge of the network (node 16), the message age is uniformly distributed in the interval of 0 s to 36 s with its median at 18 s. This is a direct result of the application layer packet generation interval of 36 s. In theory, there should be an exponential decrease of the message age with each node towards the sink because in each step the number of application data packets doubles. This trend can also be observed in the presented graph. Of course, the maximum number of monitoring messages per COMBINED packet is limited, thus, the trend is smoother towards the sink.

ACKNOWLEDGMENTS

This project has been funded in part by DFG FOR 1508 (project BATS "Dynamically adaptive applications for bat localization using embedded communicating sensor systems").

REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Elsevier Computer Networks*, vol. 38, pp. 393–422, 2002.

[2] I. Dietrich and F. Dressler, "On the Lifetime of Wireless Sensor Networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 5, no. 1, 2009.

[3] K. Kalpakis and S. Tang, "Maximum lifetime continuous query processing in wireless sensor networks," *Ad Hoc Networks*, vol. 8, no. 7, 2010.

[4] N. Pollner, M. Daum, F. Dressler, and K. Meyer-Wegener, "An Overlay Network for Integration of WSNs in Federated Stream-Processing Environments," in *IFIP/IEEE Med-Hoc-Net 2011*, Favignana Island, Sicily, Italy, June 2011, pp. 157–164.

[5] F. J. Villanueva, M. Daum, M. Strübe, J. C. Lopez, R. Kapitza, and F. Dressler, "Deployment-aware Energy Model for Operator Placement in Sensor Networks," in *IEEE/ACM DCOSS 2011, IWSN Workshop*, Barcelona, Spain, June 2011, pp. 1–6.

[6] S. Rost and H. Balakrishnan, "Memento: A Health Monitoring System for Wireless Sensor Networks," in *IEEE SECON 2006*, Reston, VA, September 2006, pp. 575–584.

[7] Z. Chen and K. Shin, "Post-Deployment Performance Debugging in Wireless Sensor Networks," in *IEEE RTSS 2009*, December 2009.

[8] K. Römer and J. Ma, "PDA: Passive distributed assertions for sensor networks," in *ACM/IEEE IPSN 2009*, San Francisco, CA, April 2009.

[9] K. Liu, Q. Ma, X. Zhao, and Y. Liu, "Self-diagnosis for large scale wireless sensor networks," in *IEEE INFOCOM 2011*, Shanghai, China, April 2011, pp. 1539–1547.

[10] K. Lin and P. Levis, "Data Discovery and Dissemination with DIP," in *ACM/IEEE IPSN 2008*, St. Louis, MS, April 2008, pp. 433–444.

[11] A. Dunkels, L. Mottola, N. Tsiftes, F. Osterlind, J. Eriksson, and N. Finne, "The announcement layer: beacon coordination for the sensornet stack," in *EWSN 2011*, Bonn, Germany, February 2011, pp. 211–226.

[12] F. Dressler and D. Neuner, "Energy-efficient monitoring of distributed system resources for self-organizing sensor networks," in *IEEE RWW 2014, WiSNet Workshop*. Austin, TX: IEEE, January 2013, pp. 145–147.

[13] A. Dunkels, "Rime - A Lightweight Layered Communication Stack for Sensor Networks," in *EWSN 2007, Poster/Demo Session*, Delft, The Netherlands, January 2007.

[14] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-Level Sensor Network Simulation with COOJA," in *IEEE LCN 2006*, Tampa, FL, November 2006, pp. 641–648.