Energy-Efficient Monitoring of Distributed System Resources for Self-Organizing Sensor Networks

Falko Dressler and Dominik Neuner

Computer and Communication Systems, University of Innsbruck, 6020 Innsbruck, Austria

***Abstract*— We study the need for integrated monitoring of system resources for supporting optimizations in self-organizing sensor networks. We focus on energy constrained sensor networks that, in order to improve the network lifetime, may need to reconfigure parts of the network. In general, the quality of the reconfiguration process strongly depends on the accuracy and freshness of information about available system resources such as the remaining energy of the nodes. Classical approaches to network management require a parallel monitoring infrastructure or at least continuous transfer of state information. Piggybacking of monitoring seems to be candidate solution, yet, it turned out that timeliness and overhead represent critical challenges. We present an adaptive and integrated monitoring solution that, substantiated by our simulation results, provide means for efficient transfer of monitoring data piggybacked to application or other monitoring packets with predefined upper latency bounds.**

***Index Terms*— Sensor networks, resource monitoring, self-organization, energy efficiency.**

I. Introduction

Typical sensor networks consist of a large number of spatially distributed sensors to monitor environmental conditions such as temperature, light, encounters, or vibrations [1]. Sensor nodes are thought to form an ad hoc network and to cooperatively transmit measured sensor data through the network to designated sink nodes. In such a network, energy is one of the most critical system resources. If the node are battery powered, energy efficiency is even more important as batteries might be difficult to replace and may constitute more than 50% of the devices' weight and volume [2]. The energy consumed at each node effectively reduces its lifetime and determines also an upper bound on the lifetime of the entire network [3]. Thus, evaluating the energy consumption of applications and protocols is an important task since erroneous lifetime predictions may cause high costs and may even render a sensor network useless before its purpose is fulfilled.

In order to prolong the network lifetime, it is not sufficient to save energy at a local context. Instead optimizations in a larger context are required. Most recently, the use of Data Stream Management System (DSMS) has been investigated to optimize the energy balance (and, of course, the data processing latencies) in the entire network [4], [5]. Energy models are required for these optimizations [6], and, of course, continuous status monitoring.

Classical network monitoring is done, for example via SNMP 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 [7], [8]. Many of the reported concepts are more debugging tools than efficient status monitoring concepts [9]. 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 [10].

In this paper, we present and discuss an integrated and energy-efficient resource monitoring technique for self-organizing sensor networks. Our monitoring protocol uses piggybacking, i.e., monitoring information are appended to already existing application or monitoring packets. Although the transmission and reception of packets with increased length due to the piggybacked monitoring data marginally increases the energy consumption, it is way more energy efficient compared to transmitting monitoring data in separate packets [11]. The general drawback of piggybacking 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.

II. Energy Model

The Contiki OS, which we also used for implementing and evaluating our architecture, provides a software-based online energy estimation mechanism [12]. It uses a simple linear model to empirically evaluate the energy efficiency of sensor network applications and protocols.

To save energy, sensor nodes may switch their components on and off – the online energy estimation mechanism is invoked as soon as a such a state change happens. For each change, a time stamp is recorded and the time difference is calculated and added to the total operating time of the component. For estimating the lifetime, the sensor node summarizes the consumed capacity of each component ($C_{new} = \sum I_x t_x$) between the current time t_1 and the last measurement t_0. The average electric current

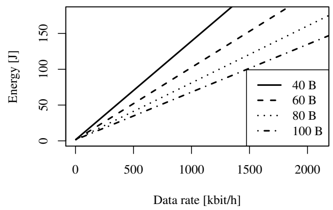


Fig. 1. Energy consumption for different payloads and data rates

I_{new} of the sensor node between the two time stamps can be calculated as:

$$I_{new} = \frac{C_{new}}{t_1 - t_0} \quad (1)$$

Based on the consumed capacity and the total available capacity C_{total}, the remaining lifetime t_{rem} of a node can be estimated:

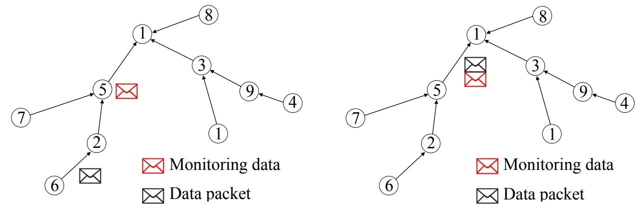
$$t_{rem} = \frac{C_{total} - C_{new}}{I_{new}} \quad (2)$$

Before introducing our monitoring architecture, we show that piggybacking indeed has major advantages. We rely on a simple network with only two sensor nodes with one node acting as packet generator broadcasting packets to the other node. By varying the payload size of each packet and the sending packet rate, the impact of these parameters on the energy consumption of the nodes can be studied.

Figure 1 illustrates the total energy consumption of the sending node for different parameter combinations. The energy values correspond to a simulation time of 1 h. The results underline that a low packet rate in combination with a large packet size compared to a high packet rate and small packet size, reduces the energy consumption drastically. This expected behavior is a result of the overhead when transmitting a single packet.

III. Monitoring Architecture

Our protocol is based on two concepts: Energy-efficient operation by utilizing the full available packet size and timely transmission of monitoring data based on distributed timer management. The monitoring protocol has been integrated into the network layer in order to make it aware of all data transmitted towards the sink node. In order to distinguish application layer data from monitoring information, the protocol defines multiple packet types, i.e., Data, Monitoring, and Combined. A new packet header is used to carry the packet type as well as the number of piggybacked monitoring information messages. A monitoring timer is used for indicating the availability of new monitoring information to the monitoring sublayer. Here, an additional timer is used for observing the age of the monitoring information. On packet reception, the network layer either forwards the content to the application layer (at the sink node) or tries to piggyback available monitoring data and forwards the packet towards the sink node. If a node has no packets to transmit for a longer period of time, the timeout of the transmit timer will generate and forward an explicit packet containing only the node's monitoring data. Although this mechanisms slightly increases the number of necessary CPU cycles, it allows for substantial energy savings due to the reduced number of data packets in the network.



(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.

Fig. 2. Piggybacking of monitoring information to a received data packet

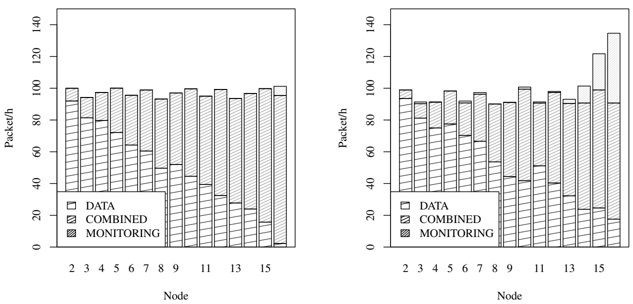
Figure 2 outlines the basic operation of our status monitoring concept. In this example, node 6 is transmitting a data packet towards node 1. At the same time node 5 has some monitoring information available (cf. Figure 2a). Instead of transmitting an additional packet with this status information, node 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 2b). The sink node separates the monitoring information from the application layer data and continues processing both separately.

IV. Selected Performance Results

For evaluating both the functionality and the performance of our monitoring protocol, we integrated the functionality in the sensor operating system Contiki. A simple application running on each node except of the sink node periodically (we used both a constant and a bursty packet rate) creates data packets destined to the base station. For the presented data, we used a simple linear setup where 16 nodes have been placed such that each node can only communicate with its direct neighbors on each side. For comparison, we also used random and grid topologies. Each application layer data packet has a size of 40 B. During the whole simulation duration of 1 h, the average packet rate was 100 packet/h. For the bursty traffic, we switched between 540 packet/h and 12 packet/h. Each simulation was repeated at least 100 times with different random seeds.

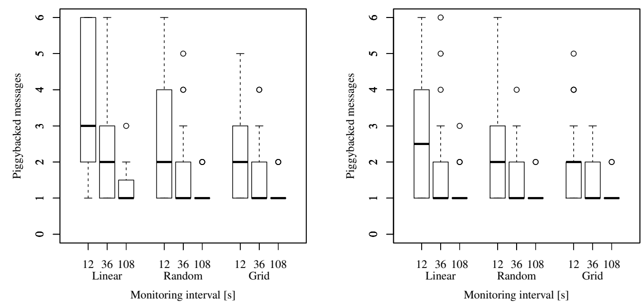
A. Distribution of Received and Forwarded Packets

We first evaluated the distribution of packet types received at the sink node. Figure 3a 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



(a) Constant traffic pattern (b) Bursty traffic pattern

Fig. 3. Distribution of packet types



(a) Constant traffic pattern (b) Bursty traffic pattern

Fig. 4. Piggybacked monitoring messages per packet

edge nodes have a high probability that a node along path can piggyback its monitoring information, thus, changing the packet type to COMBINED.

The generation of MONITORING packets happens especially at the very edge of the network. Only node 16 creates a substantial number of such packets, simply, because there is no chance to piggyback all monitoring data to its own data messages. 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).

Figure 3b illustrates the distribution of the packet types when using bursty traffic. The distribution of the DATA and COMBINED packets follows the same rules as discussed for constant traffic. However, but the probability of generating MONITORING packets, especially at the nodes close to the edge of the network, is much higher because of the small number of forwarded data packets during the low packet rate interval.

B. Piggybacking Capabilities

The number of piggybacked monitoring messages per packet, again, primarily depends on the monitoring interval. Figure 4a shows the statistical distribution of piggybacked messages per COMBINED packet for constant traffic. As can be seen, also the network topology, the number of nodes, and especially the path length influence this metric. Essentially, the probability of appending multiple messages increases with the number of hops. For bursty traffic, on average, the number of piggybacked messages is lower as seen in Figure 4b. This is due to the large intervals in which almost no piggybacking happens. Please note that the maximum number of monitoring data messages per COMBINED packet is limited to six because of the used application data size.

V. Conclusion

We studied the need for integrated monitoring of system resources for supporting self-organizing sensor networks. Our main focus was on energy constrained sensor networks that, in order to improve the network lifetime, may need to reconfigure parts of the network. The main advantage of our solution is that it supports both highly energy-efficient transmission of monitoring messages using piggybacking as well as a timely delivery of status based on a per message aging scheme. The presented simulation results clearly outline the benefit of combining multiple status monitoring messages instead of forwarding them separately.

References

[1] B. Zhang and G. Li, “Survey of Network Management Protocols in Wireless Sensor Network,” in *IEEE EBISS’09*, Wuhan, China, 2009.

[2] O. Landsiedel, K. Wehrle, and S. Götz, “Accurate Prediction of Power Consumption in Sensor Networks,” in *IEEE EmNetS-II*, Sydney, Australia, May 2005, pp. 37–44.

[3] I. Dietrich and F. Dressler, “On the Lifetime of Wireless Sensor Networks,” *ACM Transactions on Sensor Networks (TOSN)*, vol. 5, no. 1, pp. 1–39, February 2009.

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

[5] 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.

[6] 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.

[7] 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.

[8] A. Meier, M. Motani, H. Siquan, and S. Künzli, “DiMo: distributed node monitoring in wireless sensor networks,” in *ACM MSWiM 2008*, Vancouver, Canada, 2008, pp. 117–121.

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

[10] 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.

[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, 2011, pp. 211–226.

[12] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He, “Software-based On-line Energy Estimation for Sensor Nodes,” in *ACM EmNets 2007*, Cork, Ireland, June 2007, pp. 28–32.