Energy-efficient Management of Wireless Sensor Networks

Jochen Furthmüller, Stephan Kessler, and Oliver P. Waldhorst
Telematics Institute
Karlsruhe Institute of Technology
76128 Karlsruhe, Germany
Email: furthmueller|waldhorst@kit.edu, stephan.kessler@student.kit.edu

Abstract—**Managing wireless sensor networks in an energy-efficient manner is no mean feat. Management requests and responses create additional traffic in addition to the data issuing from the network's actual sensing application. Effective management therefore requires balancing the need for detailed oversight of the network against the energy consumption of the management system itself. This paper explores whether sending the management data and the sensing data together rather than separately can reduce the management system's energy footprint. From the results of our experiment using BMAC and DYMO on MICAz motes running on TinyOS, we find that our approach does indeed substantially reduce the communication costs of the management system. We discuss different models for cooperation between the management system and the sensing application and estimate the potential trade-off between the number of packet transmissions and the delay of management data. To put the theoretical results into practice, we present a management framework for monitoring wireless sensor networks that is independent of the actual sensing application. This framework allows fine-grained control over the latency tolerated for management requests for the sake of reduced energy consumption. Measurements based on a prototype implementation of the framework in an experimental setup show that up to 61% of the energy previously needed for management requests can be saved.**

I. Introduction

The topic of wireless sensor network management has been rising steadily on research agendas. Effective management is critical because even well-designed and properly implemented applications can fail at runtime due to the inherent characteristics of these networks. A serious runtime failure occurred during a well-documented experiment conducted in the Sonoma redwood forest [8], [11]: In this case, a startling 52 of the 80 deployed sensor nodes did not deliver any results. The malfunctioning was not detected until the end of the experiment, however, because there was no management system available. Unfortunately, such failures of single nodes or inter-node communication are all too common, because resources (like energy and communication bandwidth) are scarce and redundancy and reliable protocols are expensive. Thus, there is obviously an urgent need for a system that allows the operators of wireless sensor networks to set relevant parameters and ensure that everything is functioning smoothly.

Generally speaking, management functions can either be integrated into the sensor application itself or set up as a separate dedicated system. The first approach implies several severe disadvantages: Every time a new application is developed, the management functionality has to be reimplemented. This means that application developers cannot focus on their primary goal but must instead worry about how to manage the infrastructure on which their application is supposed to run. It therefore makes sense to strive for a modular generic management system that is independent of the actual sensing application.

Several such management frameworks have been proposed for wireless sensor networks. They cover aspects like monitoring the network (battery state, number of sent packets, number of dropped packets, etc.), setting parameters, deploying executable code, and logging events that are relevant for the purpose of network management. Unfortunately, these frameworks impose a certain burden for a wireless sensor network, because all management requests and responses have to be sent, received, and processed. This overhead eats into the network's energy budget which tends to be strictly constrained in the first place. This is especially true if the management framework uses a separate network stack as proposed in [8]. This scenario leads to an apparent paradox: The management framework built to enhance the sensor network's robustness and longevity instead hastens the depletion of the system's energy reserves. It can also gobble up available communication bandwidth and thus cause the network to partition.

This paper contributes a more efficient management framework for monitoring a wireless sensor network. Based on our experimental results demonstrating its potential energy savings and our analysis of the different strategies for cooperation between the management system and the sensing application, the proposed framework significantly reduces the overhead for management purposes by utilizing the unused space in data packets. Because the functionality of the management framework is completely independent, it can effortlessly be reused or combined with any sensing application. At the same time, the framework allows communication costs to be kept as low as those in a system whose management functionality is deeply embedded in the sensing application. Our framework therefore contributes to the improvement of wireless sensor network management in two ways: It increases the energy- efficiency of management operations and eases the development of managed applications. Measurements based on a prototype implementation of the framework in an experimental setup show that up to 61% of the energy previously needed for management requests can be saved.

The remainder of this paper is structured as follows. In Section II, we illustrate the potential advantage of our approach, i.e. how much energy can be saved if the sensor application data and the management system data are merged into a single packet. We also discuss and analyze the various possible cooperation strategies between the sensor application and the management system. Details of an implementation of the framework as a prototype for a sensor network consisting of MICAz Motes are presented in Section III. In Section IV we evaluate the approach by setting up an experiment to explore the trade-off between increasing the latency of management requests and correspondingly decreasing the management overhead. We then wrap up the paper with some concluding remarks.

II. Energy-efficient Management

Radio communication is one of the predominant energy consumers on most wireless sensor network platforms. There are different medium access protocols that allow the radio chip to be put into a low power sleep mode when it is not sending or receiving data(e.g. BMAC, XMAC, SMAC). Simply put, avoiding radio communication saves energy. The implications of this simple observation for a management system are discussed below.

A. Reducing the Communication Costs

It is no secret that sending and receiving data consumes a lot of energy in wireless sensor networks. However, the amount of energy that is actually spent on a single send or receive operation depends on the deployed hardware and protocols. To illustrate the energy costs of sending data, we provide measurements with MICAz Motes running on the TinyOS operating system and using BMAC [6] and DYMO [1] as the network stack. This popular combination of hardware and software is deployed in numerous wireless sensor networks.

The following experiment exploits the energy-saving potential of our approach. In two runs we measured the energy costs for sending a piece of sensing information (in this example, four bytes containing the reading of the light sensor) and a piece of management information (in this example, a nine-byte-long enumeration response, containing a node identifier and information about the node type). To measure the current, we used a Sensor Node Management Device (SNMD) [3] that was developed at Karlsruhe Institute of Technology. We considered two different approaches:

1) Sending the management data and the sensing data separately: The management data was sent first, followed by the sensing data after a gap of four seconds.
2) Sending the management data and the sensing data together in a single packet.

Comparing Figure 1 to Figure 2 clearly illustrates the benefit of sending the sensing data and the management data together. In the figures, the x-axis shows the elapsed time and the y-axis shows the current. The narrow spikes with a current of about 30 mA show the increased energy consumption whenever the radio chip is switched to active mode. The BMAC protocol periodically puts the radio chip into low power mode for a fixed interval (local sleep time) in order to save energy. After that the radio chip is put into active mode and samples the channel in order to see if some other device is trying to transmit a packet.

Figure 1 shows two longer periods of increased energy consumption. The first one is caused by the transmission of the management data. Because the receiving node might be asleep when the sender starts the transmission, the sender has to send an extended preamble that is at least as long as the local sleep time of the receiving node. At about four seconds on the x-axis, the sensing application samples the light sensor and sends a packet containing the light value. This results in another period of increased energy consumption.

In contrast, in Figure 2 the data provided by the sensing application and the data provided by the management agent are sent together in a single packet. This results in reduced energy consumption, because the radio chip is put into active mode only for short periods of time to sample the channel, as can be seen between 0 sec and 1 sec.

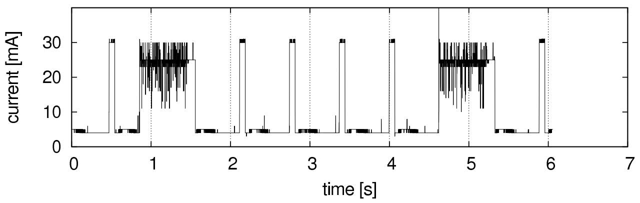

Fig. 1. Current for sending management data and sensing data separately. Two send operations cause two periods of increased energy consumption (0.9s - 1.6s and 4.6s - 5.2s).

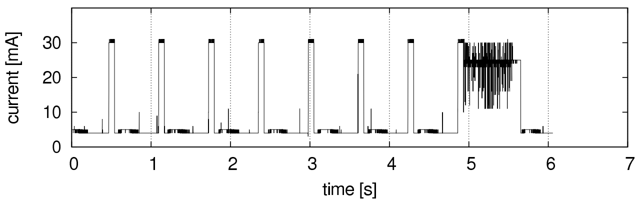

Fig. 2. Current for sending management data and sensing data together. Because there is only one send operation, there is only one period of increased energy consumption (4.9s - 5.5s).

The figures clearly show that although in each case the sensor nodes executes the very same sensing operation and the very same management operation, resulting in the same amount of transmitted application data, the energy consumption differs significantly. According to our measurements, the total energy consumption in the eight second measurement period was

- 66.029 mAs when the management data and the sensing data were sent separately; and
- 55.573 mAs when the management data and the sensing data were sent together.

So in the considered period of time, we were able to decrease energy consumption by about 15% merely by sending one large packet instead of two smaller packets. Considering the fact that sending just the sensing data and no management data at all consumed 55.546 mAs (see Figure 3) but sending both types of data together consumed only slightly more energy, i.e. 55.573 mAs, the management data essentially received a ”free ride” by traveling with the sensing data.

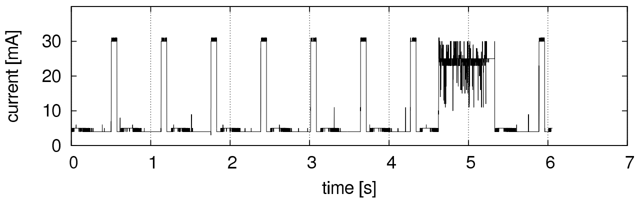

Fig. 3. Current for sending a reading from the light sensor only. Note that the energy costs for sending the management data and the sensing data together are only negligibly higher.

Following from the fact that energy consumption is directly related to the total number of send and receive operations rather than to the size of the transmitted application payload, one way to reduce the number of send and receive operations is to use a single radio packet for several data sets. This is common practice for example in sensing applications that deliver several samples in a single send operation. We extended this concept by enabling the management framework to share radio packets with the sensing application running on the sensor nodes, thereby preserving the autonomy of both the management framework and the sensing application. The approach is generic, so it can be applied to all kinds of modular applications. In the following we will call the sharing of packets by independent applications *cooperative behavior*.

B. Different Degrees of Cooperative Behavior

If packets are to be shared between the sensor network application and the management framework, either the application or the management framework has to wait for the next packet to be sent. Because we wanted to keep the management subsystem as transparent as possible to the application, we decided that there should be no delay for messages sent by the application. The management messages could thereby possibly encounter additional delays, but this would also create less overhead. We denote a management system that behaves in the described way a *cooperative management system*.

Ideally, a cooperative management system allows a human operator to define the maximum delay he is willing to tolerate for a response to a management request. A longer maximum delay results in a reduced number of messages sent in response to his request. In general there are three different kinds of management requests, each with varying degrees of cooperative behavior. The following three actors are involved in management operations:

- *Manager:* A manager is a piece of software that acts on behalf of a human operator. It sends management requests to one or more management agents and processes the corresponding management responses. We assume that the manager is running on a central computer that also acts as the data sink for the sensing application.
- *Management Agent:* A management agent is a piece of software that runs on every managed sensor network element. It is responsible for processing received management requests and creating the corresponding management responses.
- *Sensing Application:* The sensing application is the actual application in a wireless sensor network. Attached sensors are sampled and the retrieved values are transmitted to a data sink.

1) Non-Cooperative Management Request: A human operator can decide to send a management request in a non-cooperative manner. This means that the agent is supposed to respond as quickly as possible without regard for packets the application might send in the near future. That way there is no additional delay for the management requests. However, this approach means that the potential of the unused space in the application packets remains unexploited. In other words, the increased overhead in packets is tolerated in favor of a fast reply. Such a non-cooperative request and the corresponding reply is depicted in Figure 4. The manager sends a request that is non-cooperative and the agent responds immediately, ignoring any opportunities for cooperation with the sensing application.

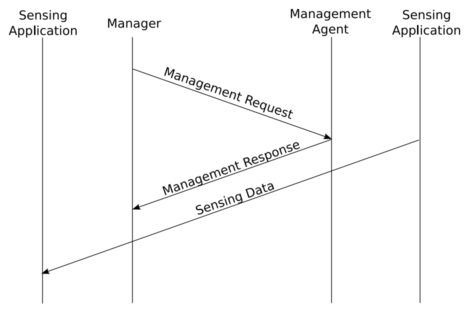

Fig. 4. Sequence of operations if the sensing application and the management agent do not cooperate.

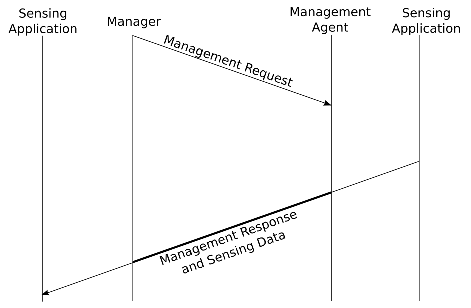

Fig. 5. Sequence of operations if the manager specifies an infinite timeout.

2) Fully Cooperative Management Requests: A management request can also be served in a fully cooperative manner.

This means that the manager signals the agent not to send a response until the next application packet offers enough space to cooperate. Obviously, this scenario offers only best effort reliability, because if there is no next application message, there will be no response at all. However, there will also be no additional packet sent in response to the request. The possible delay is therefore unbounded and the packet overhead is kept to a minimum. This type of request is especially suited for scenarios in which applications send data very frequently. An example of this kind of message exchange scheme is illustrated in Figure 5.

3) Cooperative Management Request: If the human operator needs an upper bound for the latency of a management response, the previously described concept of fully cooperative management requests is obviously not sufficient. However, it is possible to define a maximum delay. The management agent sets up a timer with the value specified by the manager within the management request. If the sensing application does not execute a send operation before the specified deadline, the agent sends its management data in a dedicated management response. Such a scenario is illustrated in Figure 6. If the application executes a send operation before the timeout occurs and there is enough space left inside the application's packet, the agent fills up the packet with its management data, as shown in Figure 7.

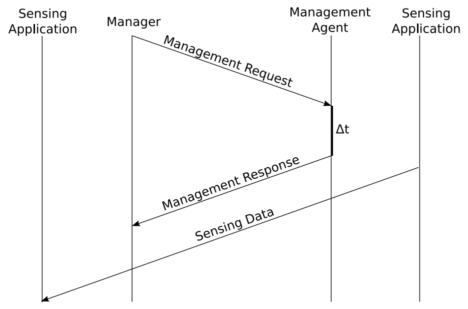

Fig. 6. Sequence of operations if the manager specifies a timeout and the sensing data are sent before the timeout occurs.

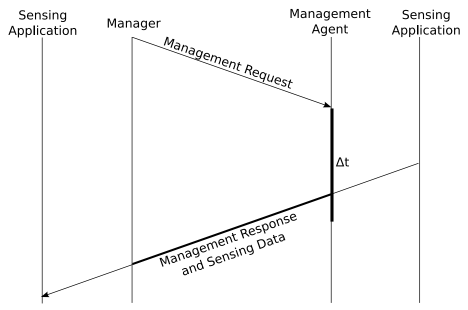

Fig. 7. Sequence of operations if the manager specifies a timeout that occurs before the sensing data are sent.

By specifying a timeout value, the human operator can precisely determine how much delay he is willing to tolerate in exchange for the energy savings due to a reduced number of sent packets. The longer the agent is allowed to wait for a cooperative send operation from the application, the lower the number of packets sent for management purposes will be. In Section IV we will present measurements that explore this trade-off. The specified timeout value represents the maximum delay for a management response in the worst case scenario.

C. Modeling the Trade-off

Considering the correlation between sent packets and energy consumption (as shown in Section II-A), the trade-off between reduced overhead (in terms of the total number of sent packets P_{Node}) and increasing latency ($T_{Latency}$) that we expect is of great interest. Assuming a certain sensor network application and a random uniform distribution of management requests over time, we can derive a mathematical model of this trade-off.

The following equations model the number of packets sent by a managed sensor node and the resulting latency of management responses depending on the degree of cooperativeness. We assume a sensor network application that periodically sends small payloads towards the data sink. This model was evaluated in an experiment as described in Section IV.

The following input quantities are considered in the model:

- $Resp$: Number of management responses;
- App: Number of send operations for the actual application;
- R: Number of packets sent in the considered time for establishing a route to the data sink;
- T_{App}: Time in seconds between two periodical send operations from the sensor network application;
- $T_{Timeout}$: Time in seconds that a management response might be delayed in order to establish a cooperation. It is necessary that $T_{Timeout}$ is in the interval $[0, T_{App}]$ since a management request will be answered by the next application message at the latest.

Note that the model can be customized to model all degrees of cooperative behavior described in Section II-B. Non-cooperative management requests are modeled by setting $T_{Timeout} = 0$, whereas $T_{Timeout} = T_{App}$ models fully cooperative management requests. Selecting $T_{Timeout} \in]0, T_{App}[$ models an arbitrary degree of cooperative behavior in between these two extremes.

1) Expected Number of Packets: The number of packets originating from a sensor node can be calculated as a summation consisting of three summands: The packets needed for application data or application data and management responses, and second, the management responses that were returned due to a timeout without cooperation. Additionally, in a multi-hop network there is some overhead to establish and maintain a routing structure towards the data sink. As the arrival of management requests is supposed to be uniformly distributed, the number of packets needed for management responses exclusively should scale with $(1 - \frac{T_{Timeout}}{T_{App}})$. This leads to the following equation with P_{Node} denoting the total number of sent packets:

$$P_{Node} = R + (App + (1 - \frac{T_{Timeout}}{T_{App}}) \cdot Resp)$$

The model considers a single sensor node. The total number of messages in a network further depends on several other factors (like network topology and routing protocol) that are not considered in this model. However, reducing the number of messages sent by a single node will still decrease the total number of messages in the network.

2) Expected Latency: The expected latency $T_{Latency}$ in turn can be calculated from the probability of the case that the management response and application data can be sent in the same packet and the probability of the case that one packet for each is sent:

$$T_{Latency} = \frac{T_{Timeout}}{T_{App}} \cdot \frac{T_{Timeout}}{2} + (1 - \frac{T_{Timeout}}{T_{App}}) \cdot T_{Timeout}$$

3) Discussion of Expected Benefit: According to this model, the amount of messages sent by a sensor node for management purposes decreases linearly with an increasing maximum timeout. In contrast, the expected latency for management responses grows more slowly than linearly with the specified maximum timeout (Figure 8). This implies that selecting a degree of cooperative behavior that is close to fully cooperative management requests will significantly reduce energy consumption with a tolerable increase of latency.

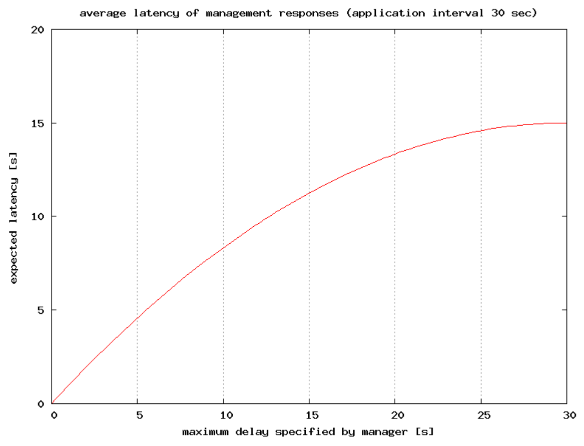

Fig. 8. Expected latency.

III. System Design

Based on the results regarding the potential energy savings, this section presents the system design of a generic management framework with an adjustable degree of cooperative behavior. The framework consists of two major components: the *manager*, running on a PC connected to the sensor network via a base-station, and a *management agent*, running on each sensor node. In our approach the manager is located on top of the network stack just as the actual sensing application (see Figure 9).

As described above, there are no dependencies between the manager and the sensing application. Cooperatively used packets are handled transparently from the application by the manager. The manager's tasks are to react to certain events according to predefined policies and to map actions and properties to numeric keys. We decided to use a numeric

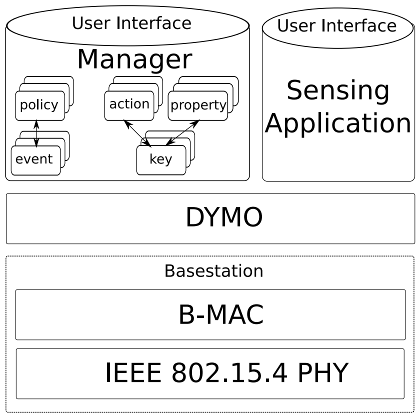

Fig. 9. Architecture on the host computer.

representation for describing requests for management actions instead of human-readable textual representation in order to save precious space in data packets. Finally, the manager offers an interface for a human operator. This might be a GUI or a simple command line tool.

On sensor nodes the architecture looks slightly different, as depicted in Figure 10. The management agent is located as a shim between the actual sensing application and the network stack. Replicating the interfaces of the network stack, it offers the same network services to the application as the network stack itself. Whenever the application is willing to support a cooperative send operation, it calls the send method provided by the management agent. In this way the management agent can create data structures containing the application's payload as well as the management payload. Then the management agent calls the send method of the network stack.

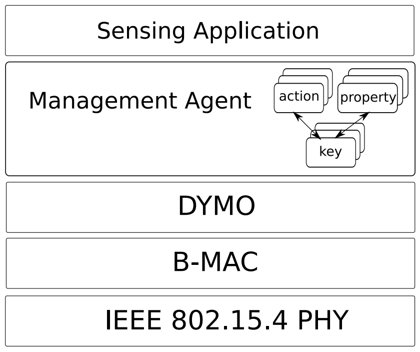

Fig. 10. Architecture on the sensor node.

A. Packet Format

Two message formats have been defined for the purpose of requesting management information and answering these requests. Management requests are created by the manager and sent to the corresponding sensor node. Management responses are delivered from the respective node to the data sink. The packet formats are explained in Sections III-A1 and III-A2.

1) Management Request: The actual management request data structure is sent and received as the payload of a DYMO data packet. Using the message id, it is handed over to the management agent on the receiving sensor node. The three fields of this data structure are described in the following.

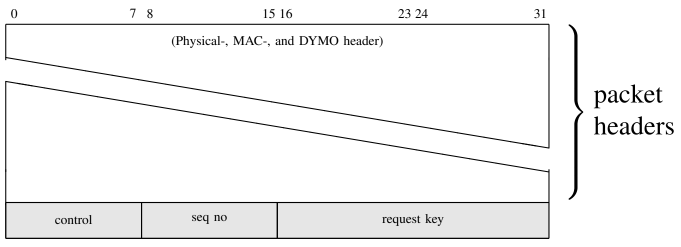

Fig. 11. Management request packet format.

control: Eight control bits allow to encode the delay the manager is willing to tolerate for a response. A timeout value of 0 denotes a non-cooperative request whereas 255 denotes a fully cooperative request. All values in between specify a timeout value in seconds.

sequence number: Using the sequence number field, the manager can map incoming responses to the corresponding request. This is necessary because the different sensor nodes might return their answers at different points in time.

request key: As mentioned earlier, each property of a sensor node that is worthy of being monitored can be identified using a numeric key. This request key is a 16-bit integer number.

2) Management Response: A management response additionally features a pointer to the application data, the collection id of the application, and the application data itself.

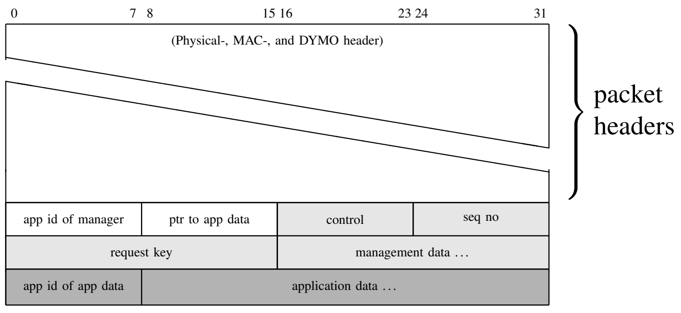

Fig. 12. Cooperative response packet format.

application id of manager: The application id specifies the application that is expecting the data in the data sink (comparable to a UDP port). This field is used to identify the manager as the appropriate handler for this message.

pointer to application data: This field tells the receiving manager if and where application data can be found in this packet. A value of 0 in this field tells the receiving instance that this packet contains no application data at all.

management data: The management data are of variable length depending on what kind of data are being sent. That is why there is a dedicated pointer that tells the receiving manager where the application data starts.

application id of sensing application: The application id specifies the application that is expecting the data in the data sink (comparable to a UDP port).

B. Processing of Cooperatively Used Packets

Whenever a packet contains application data as well as management data, the management data have to be extracted and processed by the manager, while the application data have to be passed to the application. The receiving part of the sensor network application should not experience any difference in handling a cooperatively used packet and a packet used exclusively by the application exclusively.

Thus, a cooperatively used packet arriving at the data sink is handed to the manager first. The manager extracts the management data and creates a new packet which contains the application data. It uses the pointer to the application data and the application id of the application data (see Section III-A2) for this purpose. This packet in turn is processed like every incoming packet by the network stack. When the application receives it, there is no difference at all compared to receiving a non-cooperative packet. The entire processing of a cooperatively used packet is illustrated in Figure 13. It consists of five steps:

1) The PacketDispatcher (PD) receives the packet from the MAC layer;
2) The PD determines the responsible packet handler with ActiveMessageId. In this case, it is the DymoController (DC);
3) Using the application id of the packet, the DC determines the handler that is registered for this packet. In this instance, it is the manager;
4) The manager extracts the management data. Then it hands over the application id of the application along with the application data to the DC;
5) The DC uses the application id to finally deliver the sensing data to the sensing application.

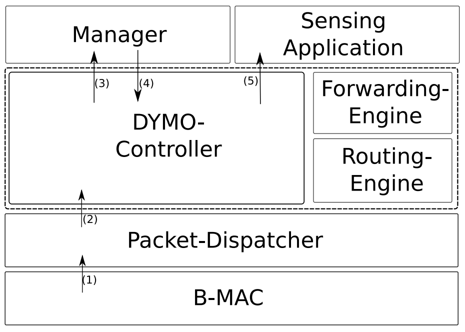

Fig. 13. Processing of a packet that bears management data as well as sensing application data.

IV. EVALUATION

In order to evaluate the proposed concept of a cooperative management framework and to validate the model introduced in Section II-C, a prototype of the management agent was implemented and deployed along with a light measurement application on a MICAz Mote. The sensor node sent four-byte-long light readings towards the data sink every 30 seconds. The data sink, a laptop with an attached base-station, collected these values. We used the DYMO [1], [7] protocol, which comes with the TinyOS operating system, as the network layer. We decided to build upon DYMO and BMAC [6], [4] because they are used in a couple of real-world sensor network projects. However, the management agent and the manager component could be used on other network stacks as well. They do not depend on any specific features provided by DYMO and BMAC.

We performed 13 independent evaluation runs to determine the influence of cooperation on the overhead that the management framework creates. In each run the manager sent 30 management requests to the sensor node and measured the delay between sending the request and receiving the response. The time that passed between two management requests was randomly chosen between 30 and 60 seconds. However, the sequence of randomly chosen time frames was made uniform in all 13 runs in order to create a reproducible setting. We measured the current between the sensor node and the power supply to determine its energy consumption. To obtain a reference value, we determined the the quantity of energy consumed when no management information at all was requested. Again, to measure the current, we used an SNMD [3] with a sample rate of 2kHz.

A. Trade-Off: Delay vs. Energy Consumption

Waiting for the next send operation of the sensor network application means accepting an additional delay in receiving the management response. We measured the delay times caused by the different degrees of cooperation in our experiment.

In each of the 13 runs, the manager sent out the management requests with a different degree of cooperation. The degree of cooperation denotes the timeout value in seconds that the agent is allowed to wait for an opportunity to transmit the management data cooperatively.

Figure 14 shows the average latency for all 30 management requests during each run, including the standard deviation. In general, the measurements support the model developed in Section II-C: With increasing cooperativeness, the latency increases less than linearly . However, the measured average latency turned out to be about 1.4 seconds higher than predicted by our model. This is due to the processing time of request and response messages and an additional delay of 1 second resulting from the 500 ms long BMAC preamble.

We conclude from Figure 14 that the average delay of a management response is shorter than the upper bound specified by the manager. The price to pay for increased cooperativeness, which increases latency for management responses, grows more slowly with higher timeouts.

Figure 15 shows that there is a clear benefit to taking the additional delay into account: The amount of energy consumed by the managed sensor node for management purposes can be significantly reduced by cooperative behavior. In Figure 15 just the energy used for management purposes is considered. This quantity was arrived at by subtracting the energy consumption measured when no management requests or responses whatsoever were sent from the measured energy consumption with management functionality enabled. We conclude from Figure 15 that a mean delay time of 15 seconds saves up to 61% of the energy previously spent for management purposes.

When the maximum timeout for management responses is set to 30 seconds or higher, every single management response is sent piggyback with a packet containing sensing data. Even in this case there is obviously some energy overhead for conducting management activities. This overhead is caused by receiving and processing the management requests.

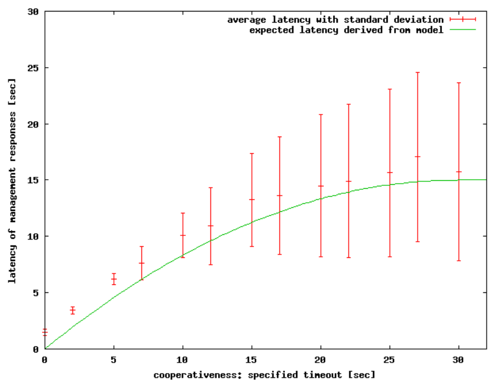

Fig. 14. Latency of management responses depending on the degree of cooperation.

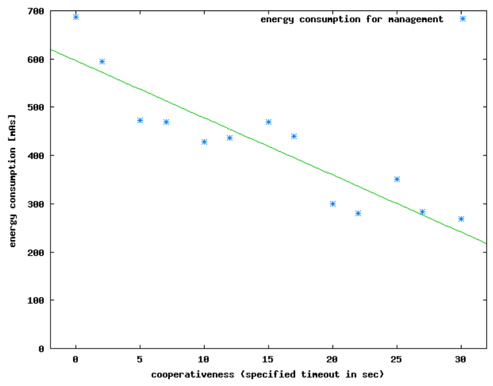

Fig. 15. Energy consumption for management purposes.

V. Related Work

The management of wireless sensor networks has been addressed in several publications: Wagenknecht et al. provide in [10] a survey of the different tasks of a management framework. They distinguish the four major tasks, i.e. monitoring, configuring, code deployment, and management of the sensor readings. We chose to focus on monitoring because we think it is the most frequently used aspect.

In [8] Tolle et al. propose a management platform for the purpose of monitoring parameters that are relevant from a network management perspective. They point out that the management and the actual application of a wireless sensor network should be loosely coupled so that the failure of the application does not necessarily affect the management system. They even run their management platform on top of its own network stack. We think that the additional memory consumption as well as the creation of network traffic for a second routing tree is a high price to pay for the increased robustness. That is why we tried to reduce the communication overhead on a packet basis by using not only the network stack and routing structures but also the unused space in single packets across the border of management and application.

The concept of piggybacking data from different sources in order to reduce communication costs in wireless sensor networks has been used in a different context before. In [9] the overhead of network gossiping is reduced by piggybacking the data of different message streams. However, we focused on the special needs of a network monitoring platform: Without disturbing the actual sensing application, management requests are served transparently. Depending on the network operator's requirements, the additional latency that occurs can be flexibly controlled. In addition, in this project we chose not to target specific network protocols.

The trade-off between responsiveness and energy consumption in wireless sensor networks has been examined before, but in a different context. In [2] it is proposed to increase the time that it takes to deliver information to the data sink and to thereby decrease the energy consumption of the entire network. But this study explicitly targets sensor networks with moving data sinks. By sending data whenever the moving data sink is nearby the number of hops and thus the over all energy consumption for delivering the data can be reduced.

The authors in [5]approached our problem from a different angle: They address how to assign roles, distribute code, and upgrade code on the fly. Their approach to reducing network traffic caused by the management framework is to create smaller amounts of code that needs to be distributed. This solution complements our approach and could be combined with a cooperative usage of data space in application packets.

VI. Conclusion

In this paper we examined the trade-off between energy consumption for the purpose of network management and the latency of management requests. We proposed the concept of cooperative requests, which allows the data sent by the actual sensor network application and the management agent to be clustered. In this way the number of packets as well as the number of sent bytes can be reduced considerably.

Using a prototype, we proved the feasibility of our approach as well as its effectiveness. The prototype was used to run the management platform concurrently with an exemplary sensor network application. In an experimental setup we measured the number of sent packets and the latency of management requests depending on the degree of cooperation between the management framework and the sensor application. Although the quantitative results are dependent on the particular setup of our experiment, they clearly show that the more latency an operator is willing to tolerate with respect to the fulfillment of management requests, the more he can reduce the number of packets that have to be sent, and this decrease in transmission traffic results in genuine energy savings. This interrelation can be exploited in order to use the tight energy budget of a sensor network in an efficient way.

Acknowledgments

The authors want to thank Detlev Meier, Patrick Armbruster, and Mario Pink for their help concerning the SNMD. This research is supported by the ”Concept for the Future” of Karlsruhe Institute of Technology within the framework of the German Excellence Initiative.

References

[1] I. Chakeres and C. Perkins. Dynamic manet on-demand (dymo) routing. *IETF Internet-Draft*, 2008.

[2] L. Galluccio, A. Leonardi, G. Morabito, and S. Palazzo. A trade-off between energy consumption reduction and responsiveness in information delivery for delay-tolerant sensor networks with mobile sink. In *IWCMC '06: Proceedings of the 2006 international conference on Wireless communications and mobile computing*, pages 563–568, New York, NY, USA, 2006. ACM.

[3] S. Kellner, M.Pink, D. Meier, and E.-O. Blass. Towards a Realistic Energy Model for Wireless Sensor Networks. In *Proceedings of IEEE Fifth Annual Conference on Wireless On demand Network Systems and Services*, pages 97–100, Garmisch-Partenkirchen, Germany, Jan. 2008. ISBN 9781424419586.

[4] K. Klues, G. Hackmann, O. Chipara, and C. Lu. A component-based architecture for power-efficient media access control in wireless sensor networks. In *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 59–72, New York, NY, USA, 2007. ACM.

[5] P. J. Marrón, A. Lachenmann, D. Minder, M. Gauger, O. Saukh, and K. Rothermel. Management and configuration issues for sensor networks. *International Journal of Network Management – Special Issue: Wireless Sensor Networks*, 15(4):235–253, 2005.

[6] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 95–107, New York, NY, USA, 2004. ACM.

[7] R. Thouvenin. Implementing and evaluating the dynamic manet on-demand protocol in wireless sensor networks. Master's thesis, University of Aarhus Department of Computer Science, 2007.

[8] G. Tolle and D. Culler. Design of an application-cooperative management system for wireless sensor networks. In *Wireless Sensor Networks, 2005. Proceeedings of the Second European Workshop on*, pages 121–132, July 2005.

[9] E. Ucan, N. Thompson, and I. Gupta. A piggybacking approach to reduce overhead in sensor network gossiping. In *MidSens '07: Proceedings of the 2nd international workshop on Middleware for sensor networks*, pages 19–24, New York, NY, USA, 2007. ACM.

[10] G. Wagenknecht, M. Anwander, T. Braun, T. Staub, J. Matheka, and S. Morgenthaler. Marwis: A management architecture for heterogeneous wireless sensor networks, 2008.

[11] S. Yang. Redwoods go high tech: Researchers use wireless sensors to study California's state tree http://www.berkeley.edu/news/media/releases/2003/07/28_redwood.shtml. online, 2003.