Energy Accounting Support in TinyOS

Simon Kellner
kellner@ira.uka.de

Frank Bellosa
bellosa@ira.uka.de

System Architecture Group
Universität Karlsruhe

ABSTRACT

Energy is the most limiting resource in sensor networks. This is particularly true for dynamic sensor networks in which the sensor-net application is not statically planned. We describe three components of our energy management system for nodes in such dynamic sensor networks: A flexible energy model and an accounting infrastructure for making sensor nodes energy-aware, and Resource Containers for managing the energy accounting information.

1. INTRODUCTION

Energy still is the most critical resource in sensor networks. Current energy supplies already take up most of a sensor node's space, but can provide the desired node lifetimes of years only when sensor-net application designers give a high priority to a long sensor-net lifetime. Sensor-Net Operating Systems (OSes) like TinyOS [2] encourage energy saving by not providing a convenient CPU-abstraction such as threads, which could, for example, tempt application developers into creating CPU-intensive waiting loops and thus into wasting energy.

Database interfaces to sensor nets like TinyDB [5] make it easy for users to retrieve sensor data: A sensor-net application is formulated as a request in an SQL-like language and interpreted by the sensor network until the request expires. The program on the sensor nodes only needs the ability to interpret and execute such requests. This eliminates the need to reprogram sensor nodes and allows multiple queries to be processed simultaneously.

Such dynamic systems can support multiple users in a sensor net, each with his own set of queries. In this scenario it is desirable to account the energy consumption of each query, e.g. to bill users based on their sensor-net usage, or to find the query with the highest energy consumption and cancel it before it wears down the energy supplies.

In this paper we describe several parts of a solution to energy management on sensor nodes that addresses multi-user dynamic sensor networks. The presented solution is currently being implemented for MICAz nodes in TinyOS 2. First we describe our energy model for a sensor node, then the infrastructure used in taking measurements and accounting the energy. As a third part, we describe the concept of Resource Containers, which will be employed to fairly distribute the accounted energy in our dynamic sensor-net setting.

2. RELATED WORK

The management of energy in sensor networks has received a significant share of research over the last years, as it concerns the primary resource of such networks.

PowerTOSSIM[6] is perhaps the approach most similar to our own model and accounting infrastructure. It instruments OS components or simulations thereof to track power states and uses an energy model to compute energy consumption for one or more sensor nodes. However, there is no implementation for current versions of TinyOS. Its energy model only considers hardware states, not the transitions in-between. The most significant difference is in the intended use: Our instrumentation and model are designed to be used in on-line energy accounting as opposed to off-line simulation.

AEON[4] is the energy model used in the AVRORA[7] simulator. It models the hardware states of a MICA2 node. Our model is based primarily on the MICAz node and additionally considers transitions between hardware states.

Energy measurements of a MICAz node can be found in [3]. The measurements of the AtMega128 controller are detailed, but the measurements of the ZigBee controller (CC2420) severely lack details. Our measurements show a real difference between the listen and the transmit state, regardless of the programmed output power in the latter.

Resource Containers are an OS-abstraction introduced by Banga, Druschel and Mogul [1] in 1999 for accounting on web-servers and consist basically of OS-provided storage for accounting data. The idea is to separate OS abstractions for CPU and resource accounting, so one can base accounting on other, more suitable abstractions. In the PC world, for example, Resource Containers (RCs) give administrators and users the ability of accounting all activity connected to a user request, which usually has a higher significance than process-based accounting.

3. ENERGY MODEL

Our sensor node energy model is designed to be used both for off-line simulations and on-line accounting. To this end, the model is specified in a more formal manner than usual.

Our energy model is based on finite state machines that closely model the hardware's power states and transitions. To account for the concurrency possible on typical hardware, each subsystem on a sensor node is modeled by its own

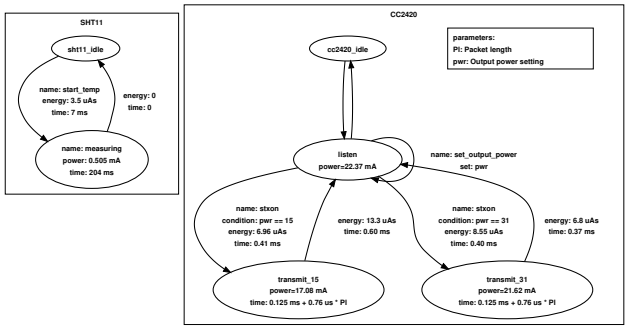

Figure 1: Energy models for the CC2420 and SHT11 chips

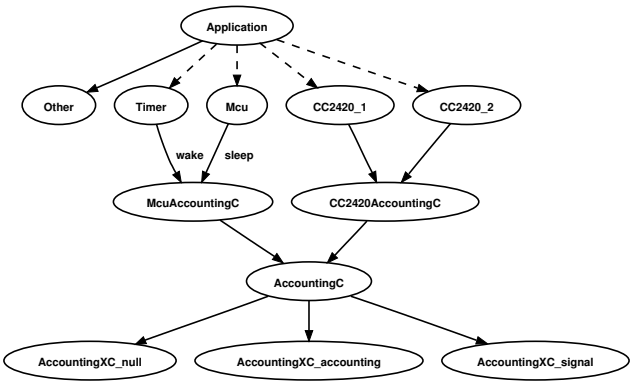

Figure 2: Accounting infrastructure overview

state machine. For example, the OS can finish instructing the radio controller to send a packet, start taking a measurement on a sensor, and get interrupted by a preset timer. In this example, the radio controller, the sensor, and the microcontroller each are modeled by one finite state machine.

States and transitions of these machines are attributed with their physical characteristics such as time (duration), energy, or power. These characteristics often depend on parameters from outside such as the battery voltage or packet length. Values for these parameters have to be supplied by each program that uses this model.

The model states describe the hardware states in as much as they can be distinguished by their power consumption. A small part of such an energy model is shown in Figure 1. Simple hardware like sensors or LEDs can be modeled by a small number of states. The model for the SHT11 chip in Figure 1, for example, covers the whole process of measuring the temperature. For other chips like the ZigBee controller CC2420 this approach can result in multiple *transmit* states, one for each selectable transmission power. Of the 32 available and 8 documented power settings, only two are shown in Figure 1 due to space constraints. Furthermore, the transitions from *cc2420_idle* to the transmit states have been omitted to avoid cluttering.

Transitions in the model describe the time and energy spent on changing the hardware state. A transition can be named or unnamed. Unnamed transitions are used for state changes which are predictable from the hardware layer. For example, upon completing a transmission the CC2420 controller automatically switches back into the *listening* state where it could then receive an acknowledgment message. The time spent transmitting is known from the length of the packet, which is known, as a packet must be stored in the CC2420's transmit buffer prior to transmission. Named transitions, on the other hand, describe changes that are not predictable from a hardware viewpoint, e.g., when the software sends a command to the radio chip to transmit a packet that is currently in the chip's buffers. In the example, *start_temp*, *set_output_power* and *stxon* are of this type.

Some models are further equipped with parameters that can be used to reduce the number of model states or to calculate energy consumption. The packet length parameter in Figure 1 determines the time the CC2420 spends transmitting, so the energy consumed for the transmission directly depends on the packet length. The other parameter, output power setting, is not strictly necessary for modeling. Its primary use is to reduce the number of states in the model: Without it, the number of states would almost triple: As the output power can be programmed in the *idle* and *listen* states, there would have to be 32 versions of both states with bidirectional transitions between all versions of a state. So not only would the number of states nearly triple, the number of transitions would grow quadratically with them.

4. ACCOUNTING INFRASTRUCTURE

Information required to use the model for on-line energy accounting has to come from various places deep inside a sensor node operating system. Usually, this means places where a driver issues commands to or receives interrupts from the hardware it manages. Due to the high degree of modularization in TinyOS 2, the relevant code places for one hardware chip could be scattered over several directories. The purpose of our accounting infrastructure is to collect this information in a consistent manner.

The process of gathering the accounting information should also be transparent to the sensor-net application. This allows developers to base energy-aware applications on traditional, statically planned applications without the need to change a lot of code.

Energy-relevant code fragments are instrumented by an additional function call. These function calls are routed first to a module for the subsystem and then to a central module named `AccountingC`, as shown in Figure 2. Here they arrive as (*component*, *event*) pairs, where *component* refers to a finite state machine of the energy model, and *event* to a named transition in this machine. The central module then uses a back-end to process the gathered information. The choice of back-end is configurable from the command line at compile time. The default back-end (`AccountinXC_null`) simply discards the information. Thus, in spite of introducing these additional function calls in the source code, this does not result in additional object code, as the compiler inlines all involved functions.

Aside from the actual accounting process, the infrastruc-

ture has shown its flexibility in combination with the TinyOS build system by providing convenient options to indicate the energy-relevant events using a GPIO pin to the measurement hardware outside. For this scenario, the central accounting component is given a different back-end module (`AccountingXC_signal`) to use, which in turn uses a platform-dependent signaling module to access a GPIO pin. This proved to be very helpful for extracting the physical characteristics of our energy model from measurements of sensor nodes. Parts of the data analysis could be automated, as phase lengths and transitions can be detected more easily by these additional signals.

5. RESOURCE CONTAINER

To make energy accounting work not only on a single node but in a large network, we employ the concept of Resouce Containers (RCs). RCs can be used to account energy consumption for each query being executed on the node and aggregate this information throughout the network.

In the following we assume that the sensor-net application responds to user-generated queries and that all packets related to one query carry the same unique ID.

5.1 Normal Resource Containers

A normal RC is associated with a query. As soon as an application learns the ID of the query currently being processed, it informs TinyOS that it wishes to switch to the RC associated with this query. The selected RC is then bound to the current TinyOS `task`.

The energy consumption of all further activities coming from this TinyOS `task` is accounted to the selected RC. If the TinyOS `task` posts a new TinyOS `task` or sets up a new timer, this binding can be stored by the scheduler or timer system, and can be used to switch back to the stored RC automatically on the corresponding wake-up call.

The OS here clearly depends on the application for correct accounting, but this is both feasible and necessary in a sensor-net application. It is necessary to prevent producing hard-to-maintain code, and it is feasible because there should be only few places where this RC-switching occurs, namely when a sensor-net application starts processing a query.

5.2 Anonymous Resource Containers

Since TinyOS applications spend most of their time sleeping and perform only minimal amounts of processing, energy consumed during interrupt handling is not negligible. For example, a timer interrupt may cause the activation of a communication device, which is subsequently used to send stored sensor data to other nodes. The sensor node is not aware of the query ID until it accesses the packet it is about to send. In the meantime, the activation of the communication device can consume a substantial amount of energy that cannot be assigned to the correct RC at that moment.

As a solution, the interrupt handler can allocate a temporary, anonymous RC and use it to account both its own energy consumption and the device activation. Later, when the application becomes aware of the query ID, it can switch to the RC associated with the query, causing the temporary RC to be merged and released.

5.3 Special Resource Containers

It may be necessary to employ special RCs to provide additional information or to handle cases where the correct RC is not known.

5.3.1 Root Resource Container

One RC worth mentioning is the RC for the whole node. It is used to collect the amount of energy consumed by the whole node, regardless of queries. This information is of interest to the nodes themselves in order to estimate the amount of remaining energy. It can also be regarded as another data source and can itself be the target of a query.

5.3.2 Idle Resource Container

Some energy consumption can not be clearly accounted to a query, e.g., the energy spent during sleep (*idle energy*). We call the problem of accounting this energy consumption in a fair manner *accounting fairness*.

One way to address the issue of idle energy accounting is to distribute the accounted idle energy among all queries known to the sensor node. To achieve this, a special RC for this energy class is present in the system. At certain times, this RC is cleared and its content distributed among all existing normal RCs. This has to be done both periodically and on creation/expiration of a query:

- Periodically so that the accounting information remains recent.
- At query instantiation to avoid penalizing this query by accounting sleeping energy spent before its instantiation.
- At query expiration to avoid losing accounted energy.

The fairness of this distribution is subject to discussion and thus should be handled by a project-dependent policy. Policy examples include equal distribution and partitioning according to duty-cycle or used energy.

Concluding, one can picture the RCs in a 3-level hierarchy: the root RC for the node, named RCs for the queries and anonymous RCs to account energy consumed for a (yet) unknown purpose. In this hierarchy, the root RC contains the aggregated accounting data of the named RCs, while the anonymous RCs will eventually be merged with one of the named RCs.

5.4 Shared Data

Caching the acquired sensor data introduces another instance of the accounting-fairness problem. Without additional measures, the first query to sample data bears the cost of acquiring it, subsequent queries can use it at almost zero cost. If the accuracy of timing or accounting can be relaxed, some trade-offs between one of them and accounting fairness can and should be considered.

A trade-off between timing accuracy and accounting fairness can be implemented as a subscriber model for sensor data: The sensor data is sampled either on time-out after the first subscription or when enough parties subscribed to this sensor data. The energy is split among all of the subscribed parties.

A trade-off between accounting accuracy and fairness can be implemented by assigning a value to the sampled sensor data that decays with every access. For example, the initial query bears 3/4 of the costs, the next query 3/4 of the remaining costs, and after a time-out, the rest is distributed across all queries that acquired this data.

5.5 Resource Container Aggregation

The usefulness of Resource Containers becomes apparent when they are shared between all network nodes. With the collected information in these RCs it is possible to account the energy consumption of the whole network individually for each query.

RCs lend themselves quite naturally to sensor nets with dynamically created queries. When receiving a new query, a sensor node allocates an RC for this query, accounts the query's energy costs to that RC and sends the accounted data back together with the responses to this query.

RC contents can easily be aggregated by summation over all RCs with the same query ID. The design of RCs to store all of the energy accounted to it since its creation makes it resilient to occasional packet loss. When accounting information is lost in the network due to occasional packet loss, the aggregated accounting information at the data sink may be incorrect, but it will be correct again after reception of the next packet.

To allow the data sink to compare aggregated RC values from different queries and to detect packet loss, a node should additionally send the number of sensor nodes involved in an aggregate, if this information is not already present in the aggregated sensor data.

6. CONCLUSION

In sensor networks with dynamically created queries, on-line energy accounting is necessary to make the network energy-aware. We presented three parts of an on-line energy accounting solution currently being developed for TinyOS 2. An accounting infrastructure uses an energy model to make each sensor node energy-aware. Resource Containers allow to extend this accounting mechanism to cover the whole network. Together, this is an energy management solution for multi-user dynamic sensor networks.

7. ACKNOWLEDGMENTS

This work is done as part of the BW-FIT project ZeuS.

8. REFERENCES

[1] G. Banga, P. Druschel, and J. Mogul. Resource containers: A new facility for resource management in server systems. In *Proceedings of the Third Symposium on Operating System Design and Implementation (OSDI'99)*, Feb. 1999.

[2] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *ASPLOS-IX: Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 93–104, New York, NY, USA, 2000. ACM Press.

[3] M. Krämer and A. Geraldy. Energy measurements for MicaZ node. In *5. GI/ITG KuVS Fachgespräch „Drahtlose Sensornetze“*, number 2006/07, pages 61–68, Universität Stuttgart, Institut für Parallele und Verteilte Systeme, July 2006.

[4] O. Landsiedel, K. Wehrle, and S. Götz. Accurate prediction of power consumption in sensor networks. In *Proceedings of The Second IEEE Workshop on Embedded Networked Sensors (EmNetS-II)*, May 2005.

[5] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 491–502, New York, NY, USA, 2003. ACM Press.

[6] V. Shnayder, M. Hempstead, B. Chen, G. W. Allen, and M. Welsh. Simulating the power consumption of large-scale sensor network applications. In *SenSys '04: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, pages 188–200, New York, NY, USA, Nov. 2004. ACM Press.

[7] B. L. Titzer, D. K. Lee, and J. Palsberg. Avrora: scalable sensor network simulation with precise timing. In *IPSN '05: Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, page 67, Piscataway, NJ, USA, 2005. IEEE Press.