Software-based On-line Energy Estimation for Sensor Nodes

Adam Dunkels, Fredrik Österlind, Nicolas Tsiftes, Zhitao He
{adam,fros,nvt,zhitao}@sics.se
Swedish Institute of Computer Science

ABSTRACT

Energy is of primary importance in wireless sensor networks. By being able to estimate the energy consumption of the sensor nodes, applications and routing protocols are able to make informed decisions that increase the lifetime of the sensor network. However, it is in general not possible to measure the energy consumption on popular sensor node platforms. In this paper, we present and evaluate a software-based on-line energy estimation mechanism that estimates the energy consumption of a sensor node. We evaluate the mechanism by comparing the estimated energy consumption with the lifetime of capacitor-powered sensor nodes. By implementing and evaluating the X-MAC protocol, we show how software-based on-line energy estimation can be used to empirically evaluate the energy efficiency of sensor network protocols.

1. INTRODUCTION

Energy is of primary importance in wireless sensor networks. Sensor nodes have limited energy supplies and hence must make intelligent and informed decisions that can help conserve energy. Being able to make decisions based on knowledge of the current energy consumption can increase the lifetime with up to 52% [6]. Also, by distributing energy information to neighboring sensor nodes, routing can be made more energy-efficient [10, 13].

Current hardware platforms such as the Tmote Sky [7] and the ESB [9] do not provide hardware mechanisms for measuring the energy consumption of the sensor node. While it is possible to measure the battery voltage, the battery voltage is affected by the battery capture effect and is not a good estimate of the current energy consumption. Furthermore, the unique characteristics of sensor network applications make hardware-based energy measurement difficult [3].

In this paper we investigate the use of a software-based on-line energy estimation mechanism for small sensor nodes. The mechanism runs directly on the sensor nodes and provides real-time estimates of the current energy consumption.

EmNets '07, June 25-26, 2007, Cork, Ireland

Our target platforms are low-end sensor nodes, such as the ESB [9] and the Tmote Sky [7]. The mechanism uses an intentionally simple linear model that is easy to implement and add to existing sensor node operating systems. No modifications to existing applications or network protocols are required.

The idea of doing software-based on-line energy estimation is not new; for example, Younis and Fahmy have previously suggested the use of a simple linear model for estimating energy consumption [12]. However, their model requires extensive changes to all applications and protocols that use it. We are not aware of any previous work that evaluates the efficiency of software-based on-line energy estimation.

The contribution of this paper is that we show that an intentionally simple mechanism for on-line node-level energy estimation can provide a good estimation of energy consumption. We have implemented our mechanism in Contiki [2] but the mechanism is general enough to be used in any operating system for sensor nodes. We evaluate the accuracy of the energy estimates by comparing the results from the energy estimation mechanism with the lifetime of sensor nodes. Our results show that the energy estimation provides good estimates, but further study is needed to quantify the accuracy of the mechanism.

Hardware-based energy measurement mechanisms are typically difficult to add to existing hardware designs as they require a significant amount of modifications [3]. The cost increase for adding energy measurement is expected to be 100% [3] In contrast, our software-based energy estimation mechanism is easily added to existing hardware and software designs, without any additional per-unit cost. In this paper we show that a software-based energy estimation mechanism can be added to an existing sensor node operating system by adding only a few lines of code in specific operating system functions. No changes are needed to the applications or network protocols.

The rest of this paper is structured as follows. We present our energy estimation mechanism in Section 2 and its implementation in Contiki in Section 3. We discuss calibration in Section 4. We evaluate the mechanism in Section 5 and review related work in Section 6. We discuss future work in Section 7 and conclude the paper in Section 8.

2. ON-LINE NODE-LEVEL ENERGY ESTIMATION

To conserve energy, sensor nodes frequently switch on and off their components, such as the communication device, LEDs, or sensors. Our energy estimation mechanism is in-

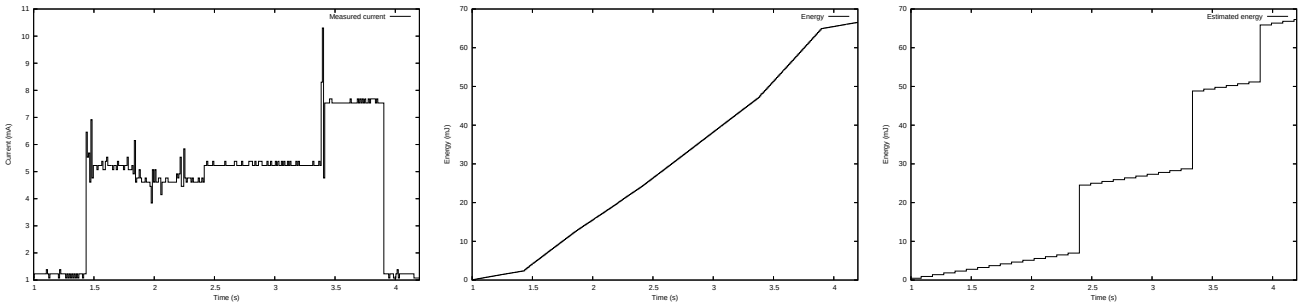

Figure 1: Left: measured current draw. Middle: energy consumption over time, calculated by integrating the left figure. Right: estimated energy consumption over time, obtained from the on-line energy estimation mechanism.

voked every time a hardware component is switched on or off. When a component is switched on, the estimation mechanism stores a time stamp. As the component is switched off again, a time difference is produced and added to the total time that the component has been turned on. The estimation mechanism keeps a list of all components and the time of which they have been turned on. The mechanism then uses the current draw of each component to produce an estimate of the total energy consumption.

2.1 Example

Figure 1 shows the operation of the energy estimation mechanism. The left graph shows the measured current draw for a part of a typical sensing application. The application turns on the on-board temperature sensor and reads its value (1.5s - 2.5s). It then turns on the radio for one second to listen for any incoming traffic before it sends its sensor reading to its neighbor (2.5s - 3.5s). After sending its value, it signals that it is alive by blinking the green LED for half a second (3.5s - 4s). The middle graph shows the energy consumption and is obtained by integration of the left graph.

The right graph in Figure 1 shows the estimated energy consumption of the above activity, as obtained from our on-line energy estimation mechanism. As the mechanism estimates the energy consumption only when a component is turned off, the graph shows jumps when the components are powered down. The energy consumption of the CPU is estimated every time the processor is switched from low power mode to normal mode, and therefore shows up as an increasing line. The saw tooth-pattern in the right graph is due to the intervals at which the energy estimate was sampled.

2.2 Energy Model

The on-line energy estimation mechanism uses a linear model for the sensor node energy consumption. The total energy consumption E is defined as

$$\frac{E}{V} = I_m t_m + I_l t_l + I_t t_t + I_r t_r + \sum_i I_{c_i} t_{c_i}, \qquad (1)$$

where V is the supply voltage, I_m the current draw of the microprocessor when running, t_m the time in which the microprocessor has been running, I_l and t_l the current draw and the time of the microprocessor in low power mode, I_t and t_t the current draw and the time of the communication device in transmit mode, I_r and t_r the current draw and time of the communication device in receive mode, and I_{c_i} and t_{c_i} the current draw and time of other components such as sensors and LEDs. The energy model does not contain a term for the idle current draw of the board itself; this is embedded in the low power mode draw of the microprocessor.

In many cases, the voltage V does not need to be explicitly computed, as the energy estimate often is used only for comparison between different nodes. If all nodes have the same voltage, computed E/V values can be compared directly, without the need of a multiplication operation.

3. IMPLEMENTATION

The implementation of the on-line energy estimation mechanism requires only small changes to existing operating system source code. We have implemented the mechanism in the Contiki operating system [2] but the mechanism can easily be incorporated into other sensor node operating systems.

The energy estimation module maintains a table with entries for all components, the CPU, and the radio transceiver. Each table entry contains the total time that the corresponding component has been turned on.

Energy estimation is implemented in two lines of code in the device driver for the hardware for which energy is to be estimated. When the component is turned on, the energy estimation module is called to produce a time stamp. When the component is turned off, the time difference from when the component was turned on is computed. The time difference is added to the table entry for the component. To add energy estimation to an existing device driver, only two lines of code needs to be added.

Time measurement is implemented using the on-chip timers of the MSP430. Since the on-chip timers work even when the microcontroller is in low-power mode, the time measurement is non-intrusive. We use the 32768 Hz clock divided by 8, producing 4096 clock ticks per second.

4. CALIBRATION

To be able to accurately estimate the energy, the estimation mechanism must know the current draw for the microprocessor, communication device, and the components to be used in Equation 1. Calibration is performed by off-line measurement of the current draw for the sensor board using an oscilloscope. The average current draw for the different

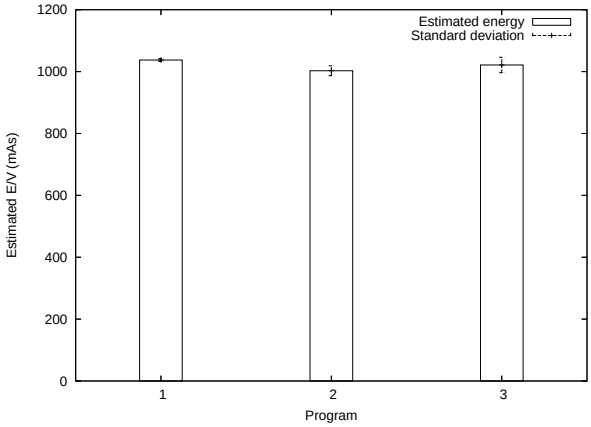

Figure 2: The estimated energy in the last packet before node failure. Ideally, the estimated energy should be the same regardless of program.

parts of the board is then used as input to the estimation mechanism.

5. EVALUATION

To evaluate the efficiency of the on-line energy estimation mechanism, we use the method developed by Ritter el al. [8] where ESB sensor nodes [9] are equipped with 1F capacitors. The capacitors have a predictable energy storage and energy dissipation rate [11]. The capacitor is charged by turning on the power switch on the sensor node, which causes the battery to be connected to the capacitor. When the battery is switched off, the node runs on the energy contained in the capacitor. The energy stored in the capacitor can power the ESB node for a few minutes, depending on the energy consumption of the software running on the node. The program shown in Figure 1 runs for about four minutes from the energy in the capacitor.

We use two different ESB nodes and run all experiments on both nodes. We run three different programs that emulate a standard sensor network application using the sensors, turning the radio and LEDs on and off at regular intervals, and sending data packets over the radio. The intervals for the three programs are configured to be different.

In addition to turning components on and off, the sensor node transmits its estimated energy consumption via radio once every two seconds. The node runs until it dies. We define the lifetime of the node to be the time until the node sends its last packet. A base station node listens to the radio traffic and sends it via a serial cable to a PC, which logs the data to a file.

5.1 Estimation Accuracy

To evaluate the energy estimation accuracy, we run the three programs on capacitor-equipped ESB nodes and capture the estimated energy that the nodes transmits over the radio. Ideally, the nodes should report the same energy estimate in the last packet they transmit before they die. The choice of program should not affect the estimated energy.

Figure 2 shows the energy estimate in the last packet before node failure. The figure shows that the estimated energy varies both between programs and within each program. This is due to the low precision of the experimental evaluation mechanism, which involves a fair amount of manual intervention. For example, we need to manually turn on and off the nodes to charge and discharge the capacitor. We plan to investigate validation mechanisms with higher precision [3] as part of future work.

Table 1: Code and memory footprint.

Module	Code (bytes)	Memory (bytes)
Book keeping	54	48
Summation	340	0

5.2 Overhead of the Estimation Mechanism

The energy estimation mechanism measures and stores the time during which hardware components have been turned on. The mechanism therefore incurs an overhead in terms of processing power. However, the code required to measure and store the time is very small; only 11 processor cycles to store a time stamp before turning on the hardware component and 20 processor cycles to update the total time after the component has been switched off. We measured the number of times the time stamping code was invoked in the data collection case study below and found that, on the average, the time stamping code was run 60 times per second. This incurs an overhead of approximately 1800 cycles per second, or 0.7% of the total processing time. The energy overhead of these few extra cycles is negligible.

The code footprint and memory requirements of the mechanism are small, as shown in Table 1. The book keeping module keeps track of how long the components have been turned on. The summation module calculates Equation 1 and must be altered depending on the calibrated parameters for the particular hardware device on which the mechanism is executed.

5.3 Case Study: X-MAC

As an example of how our energy estimation mechanism is intended to be used, we estimate the energy overhead of the X-MAC duty-cycling radio protocol [1] using the software-based on-line energy estimation mechanism.

The X-MAC protocol switches on and off the radio at regular intervals to conserve the energy of the sensor node. When a node is to send a packet, it first broadcasts a train of short strobe packets. When the other nodes hear a strobe packet, it turns on its radio in preparation of receiving a full packet. As an optimization for unicast packets, the strobe packets include the address of the receiver of the full packet. When the receiver hears the strobe packet, it immediately sends a short acknowledgement packet to the sender of the strobe packets. The sender can then immediately send its full packet. All other nodes that overhear the packets can turn off their radios until the full packet has been transmitted.

In this case study, we are interested in experimentally evaluating the unicast-packet optimization of X-MAC. We implement the X-MAC protocol in Contiki and setup a data collection network consisting of nine Tmote Sky nodes. One node acts as a base station; it collects the data from the network, and writes it to a PC. The other eight nodes are

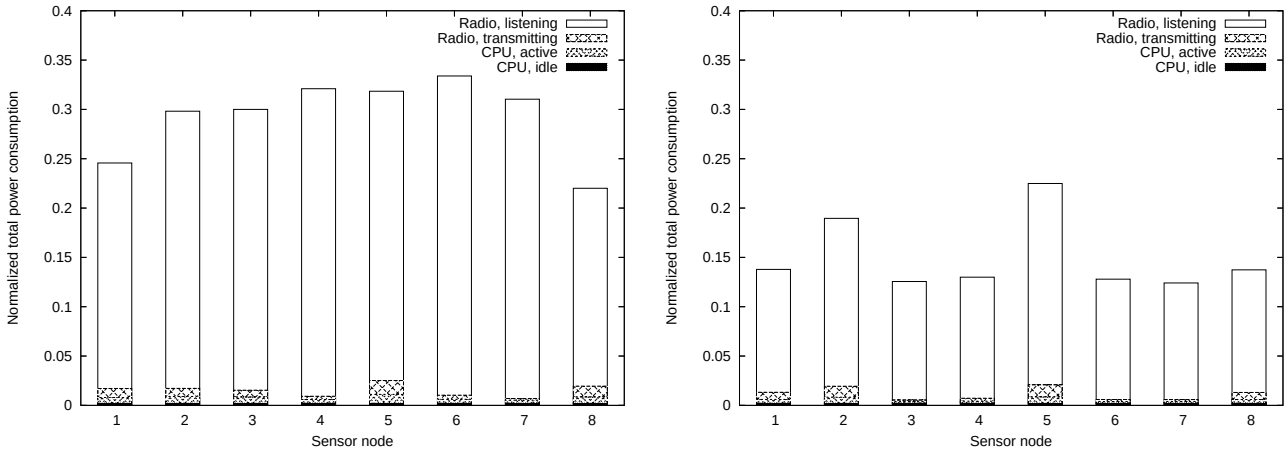

Figure 3: The estimated power consumption of eight nodes running the Contiki data collection protocol with X-MAC. The left graph is without the unicast optimization and the right graph with optimization. The power is normalized to the maximum Tmote Sky power consumption.

arranged so that they form a two-hop network. The data collection protocol sends energy estimates produced by the energy estimation mechanism. We configure the X-MAC protocol to have a duty cycle of approximately 9%, based on the estimated number of packets per second and the analytical results calculated by the authors of X-MAC [1].

The estimated energy is shown in Figure 3. The left graph shows the estimated energy without the unicast optimization and the right graph the estimated energy with the optimization. We see that the optimization is able to obtain a significant reduction in energy consumption. Furthermore, we see that idle listening is dominating the total energy consumption.

Finally, we see that the energy consumption of nodes 2 and 5 is significantly higher than the energy consumption of the other nodes. This is because those nodes happened to route packets for the two-hop nodes in our experiment. This behavior would have been difficult to find if we would have measured the energy of a single node only. We take this as an indication that a systems perspective is useful when evaluating the energy consumption of sensor network protocols.

6. RELATED WORK

Jiang et al. [3] show that the unique characteristics of sensor network applications make it difficult to measure the energy consumption of sensor nodes. The authors develop a hardware-based mechanism for measuring the energy consumption of sensor nodes that they expect to have a per-unit cost similar to that of the sensor node. It therefore incurs a significantly higher cost than software-based energy estimation. However, hardware-based mechanisms are able to capture phenomena such as per-node fluctuations in energy consumption that are not possible to study using our software-based mechanism. We expect both hardware-based and software-based methods to be used in the future.

Operating systems for wireless sensor networks such as TinyOS, SOS, Mantis, and Contiki, reduce their energy consumption by powering off the microcontroller and hardware components when they are not used. Our work is orthogonal to this; on-line energy estimation estimates the actual energy consumption of the devices rather than trying to reduce the energy consumption.

Younis and Fahmy [12] use a linear model for on-line estimation of node-level energy consumption, but do not evaluate the mechanism's effectiveness. Furthermore, their model requires all applications to be explicitly rewritten to estimate their own energy consumption. In contrast, our model does not require any modifications to applications or network protocols.

There are many sensor network simulators with energy estimation abilities [4, 5]. However, simulators are run off-line and cannot estimate the energy consumption in an on-line sensor network. Unlike off-line emulation, on-line energy estimation makes it possible to do energy-aware decisions about routing and transmission power scheduling, which potentially can prolong sensor node lifetime [6, 10, 13].

Landsiedel, Wehrle, and Götz [4] estimate the energy consumption of TinyOS-based systems using an off-line emulator. Our work is different in that it does on-line energy estimation. The off-line emulation must accurately capture the time in the processor is awake, and significant effort is devoted to doing accurate time emulation. With on-line energy estimation, time measurements can be directly obtained from on-chip timers provided by the microprocessor. Furthermore, the effects of interrupts and timers, that need to be carefully emulated in an off-line estimator, are automatically included in the energy estimates obtained with on-line estimation.

The need for network-level energy monitoring in sensor networks is discussed by Zhao, Govindan, and Estrin [13]. Their work addresses the problem of transmitting node energy levels across the network. The monitoring mechanism, which is developed for a significantly larger target platform than ours, assumes the existence of a mechanism for measuring or estimating the node-level energy, such as ACPI. Our work addresses the problem of providing a way to estimate node-level energy, even for devices without ACPI, and is thus orthogonal to their work.

7. FUTURE WORK

The results presented in this paper are only initial results and more experiments are needed to fully validate software-based on-line energy estimation. We see at least three possible directions for future work, as described below.

Battery model integration. The on-line energy estimation mechanism tries to estimate the energy consumption of the sensor board. However, for battery-powered sensor boards this cannot be directly translated into a life time estimate of the sensor node because of the non-linearity of batteries. Nevertheless, it might be possible to extend the model from Equation 1 to take non-linear battery effects into consideration to obtain life time predictions based on past energy consumption.

Non-linearity of the energy consumption of hardware components. Hardware effects can sometimes cause non-linear behavior of the current draw so that the current draw for two components is not the sum of the current draw of the individual components. The linear model in Equation 1 cannot capture such behavior. Also, the current implementation of the model measures individual components without taking other components into consideration. It would be interesting to study how to integrate non-linear current draw effects into the present model and its implementation.

Validation of the model with hardware energy measurement. Recent work has shown that hardware-based energy measurement for sensor nodes is difficult, but not impossible [3]. It would be interesting to compare the results of software-based on-line energy estimation with hardware-based energy measurements.

8. CONCLUSIONS

In this paper we present a software-based on-line energy estimation mechanism for tiny sensor nodes. The mechanism is easy to implement in a sensor node operating system and does not require any changes to applications or network protocols. By experimentally correlating the estimated energy and the life time of real sensor nodes, we show that the energy estimation mechanism is sound, but that further study is needed to accurately quantify the error rate of the mechanism. Finally, we use the mechanism to empirically study the energy consumption of the X-MAC protocol.

Acknowledgments

This work was partly financed by VINNOVA.

9. REFERENCES

[1] M. Buettner, G. V. Yee, E. Anderson, and R. Han. X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 307–320, Boulder, Colorado, USA, 2006.

[2] A. Dunkels, B. Grönvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proceedings of the First IEEE Workshop on Embedded Networked Sensors (IEEE Emnets '04)*, Tampa, Florida, USA, November 2004.

[3] X. Jiang, P. Dutta, D. Culler, and I. Stoica. Micro power meter for energy monitoring of wireless sensor networks at scale. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 186–195, Cambridge, Massachusetts, USA, 2007.

[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)*, Sydney, Australia, May 2005.

[5] P. Levis, N. Lee, M. Welsh, and D. Culler. Tossim: accurate and scalable simulation of entire tinyos applications. In *Proceedings of the first international conference on Embedded networked sensor systems*, pages 126–137, Los Angeles, California, USA, 2003.

[6] C. Park, K. Lahiri, and A. Raghunathan. Battery discharge characteristics of wireless sensor nodes: An experimental analysis. In *Proceedings of the IEEE Conf. on Sensor and Ad-hoc Communications and Networks (SECON)*, Santa Clara, California, USA, September 2005.

[7] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. In *Proc. IPSN/SPOTS'05*, Los Angeles, CA, USA, April 2005.

[8] H. Ritter, J. Schiller, T. Voigt, A. Dunkels, and J. Alonso. Experimental Evaluation of Lifetime Bounds for Wireless Sensor Networks. In *Proceedings of the Second European Workshop on Sensor Networks (EWSN2005)*, Istanbul, Turkey, January 2005.

[9] J. Schiller, H. Ritter, A. Liers, and T. Voigt. Scatterweb - low power nodes and energy aware routing. In *Proceedings of Hawaii International Conference on System Sciences*, Hawaii, USA, January 2005.

[10] C. Schurgers and M. Srivastava. Energy efficient routing in wireless sensor networks. In *Proceedings of MILCOM 2001*, 2001.

[11] T. Staub, T. Bernoulli, M. Anwander, M. Waelchli, and T. Braun. Experimental lifetime evaluation for mac protocols on real sensor hardware. In *Proceedings of ACM Workshop on Real-World Wireless Sensor Networks (REALWSN'06)*, Uppsala, Sweden, June 2006.

[12] O. Younis and S. Fahmy. An experimental study of routing and data aggregation in sensor networks. In *Proceedings of the IEEE International Workshop on Localized Communication and Topology Protocols for Ad Hoc Networks (IEEE LOCAN)*, Washington, DC, USA, November 2005.

[13] Y. Zhao, R. Govindan, and D. Estrin. Residual energy scan for monitoring sensor networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC'02)*, Orlando, Florida, USA, March 2002.