Olaf Landsiedel, Klaus Wehrle, Ben L. Titzer, Jens Palsberg

Enabling Detailed Modeling and Analysis of Sensor Networks

Olaf Landsiedel is a PhD student in the Protocol Engineering and Distributed Systems Group at the University of Tübingen. He received his Masters Degree in Computer Science from the University of Kansas, where he graduated in 2003 with honors. His studies were funded by a Fulbright and a Direct Exchange Scholarship. Olaf received his undergraduate degree from the University of Kiel. His research interests include protocol and systems engineering as well as network modeling and simulation with a focus on embedded systems. He is the lead developer of the sensor-network energy model AEON.

Klaus Wehrle is head of the Protocol-Engineering and Distributed Systems Group at University of Tübingen funded within the DFG-Emmy Noether Program. Klaus received his Diploma and PhD from University of Karlsruhe in 1999 and 2002, respectively, both with honors. From 2002 to 2003, Klaus was Postdocoral Fellow at ICSI Berkeley. His research activities are focused on protocol and systems engineering, modeling and network simulation, Peer-to-Peer systems and operating system issues of networking. Klaus actively participated in the IETF; recently RFC 3662 and RFC 3754 were published.

Ben L. Titzer is a PhD student at UCLA. He received his Masters Degree from UCLA in 2004 and an undergraduate degree from Purdue University in 2002. His research interests include simulation and analysis of sensor network programs as well as programming languages and compilers for sensor networks. He is the lead developer of the Avrora microcontroller and sensor network simulator.

Jens Palsberg is a Professor of Computer Science at UCLA. He received a Ph.D. in Computer Science from University of Aarhus, Denmark in 1992. In 1992-1996 he was a visiting scientist at various institutions, including MIT. In 1996-2002 he was an Associate Professor and, in 2002-2003, Professor and Associate Head of Computer Science at Purdue University. His research interests span the areas of compilers, embedded systems, programming languages, software engineering, and information security.

Abstract—Simulation is the de-facto standard tool for the evaluation of distributed and communication systems like sensor networks. Most simulation efforts focus on protocol- and algorithmic-level issues, thus depending on the right choice and configuration of models. However, as such models commonly neglect time dependent issues, many research challenges, like energy consumption and radio channel utilization still remain.

In this article we present two new tools to model and analyze sensor networks: Avrora, a fast and accurate sensor network simulator, and AEON, a novel tool built on top of Avrora, to evaluate the energy consumption and to accurately predict the lifetime of sensor networks.

Avrora is a highly scalable instruction-level simulator for sensor network programs. It simulates the execution of the program down to the level of individual clock cycles, a time quantum of about 135 ns. By incorporating state of the art simulation techniques, including an efficiently maintained event queue, fast-forward through sleep-time, and parallel simulation, it can simulate entire networks of nodes in real time.

AEON's energy model is based on Avrora and makes use of the cycle accurate execution of sensor node applications for precise energy measurements. Due to limited energy resources, power consumption is a crucial characteristic of sensor networks. AEON uses accurate measurements of node current draw and the execution of real code to enable accurate prediction of the actual power consumption of sensor nodes. Consequently, it prevents erroneous assumptions on node and network lifetime. Moreover, our detailed energy model allows to compare different low power and energy aware approaches in terms of energy efficiency. Thus, it enables a highly precise estimation of the overall lifetime of a sensor network.

I. Introduction

Research advances in highly integrated and low power hardware, wireless communication technology, and highly embedded operating systems enable sensor networks. A sensor network may consist of several thousands of nodes, each with very limited processing, storage, sensing and communication abilities.

Simulation is the most frequently used technique to evaluate protocols and distributed systems, as it enables the evaluation and validation of systems and applications before deployment. Simulation provides a cost efficient, controllable and repeatable environment for evaluation. However, most simulation efforts focus on protocol- and algorithmic-level issues, thus depending on the right choice and configuration of models. However, as such models neglect time dependent issues, many research challenges, like energy consumption and radio channel utilization, still remain. To solve such challenges we present Avrora, which simulates the operation of each sensor by interpreting the machine code of the sensor network program while retaining

cycle accuracy. It simulates the operation of individual hardware devices present on the sensor node, such as timers, a serial port, sensors, and a software-controlled radio that nodes use to communicate short messages with each other. Unlike other cycle-accurate instruction simulators, Avrora uses an event queue to maintain synchronization among the devices and the CPU's clock. This asynchronous model improves performance and reduces the overhead of accurate device simulation. Because Avrora simulates execution at the machine code level, it can simulate sensor network applications written in any language with any operating system components, including TinyOS and SOS, two popular sensor node platforms.

Avrora can handle networks of thousands of nodes with good performance. Each node is simulated in its own thread, enabling parallel execution for better scalability on multiprocessor machines. An efficient synchronization strategy preserves the time-dependent properties of the node interactions. It ensures that network-wide behavior such as routing in a multi-hop scenario is not only ordered correctly in time, but is accurate down to individual clock cycles as well. Such precise timing is a prerequisite for accurately modeling and analyzing many properties of sensor networks, including energy.

Energy consumption in sensor networks is a crucial design factor, because sensors are commonly battery driven. For developers it is important to evaluate the power consumption of applications accurately, since the choice of algorithms heavily influences energy consumption. Once nodes are deployed, it may be challenging or impossible to change batteries in the field. Poor lifetime projection may cause high costs and even render a sensor network useless before its purpose is fulfilled. Although recent research provides many energy efficient or energy aware applications, many measurement schemes lack sufficient detail (for example counting the number of packets sent) for qualitatively estimating the power consumption.

Based on the execution of real code and measurements of node current draw, AEON (Accurate Prediction of Power Consumption in Sensor Networks) enables accurate prediction of the actual power consumption of sensor nodes. Thus, it helps to prevent erroneous assumptions on device and network lifetime. Such a detailed prediction allows the comparison of different low power and energy aware approaches in terms of energy efficiency.

Avrora provides a fast and flexible simulator base for the energy model. It is implemented in Java and has an open and extensible framework for simulating sensor network programs, allowing instrumentation of many types without requiring modification to the source code. This extensibility has proved very useful in implementing the energy model; an intuitive structure and good modularity allowed rapid integration of the energy model without any major modifications. Avrora enjoys many of the benefits of the Java programming language including flexible extension points, absence of memory leaks, dynamic class loading for extensibility, excellent portability, and others. A careful design and attention to critical performance issues has allowed it to achieve excellent performance on the latest commercial Java VMs, as demonstrated by several numeric benchmarks against other available simulators.

The remainder of this paper is structured as follows. First, we discuss background and related work in Section II. Section III introduces the highly scalable and cycle accurate sensor network simulator Avrora. Section IV presents the energy model as the basis of AEON and a detailed validation. Section V presents possible applications of AEON by quantitatively evaluating energy aware components, e.g. Power Management and Multihop Routing. Next, Section VI introduces energy profiling of source code to give feedback about energy efficiency of the implementation. Section VII compares the level of detail and accuracy of AEON with related work. Finally, Section VIII concludes the paper and presents future work.

II. Related Work

AtEmu [1] was the first instruction-level sensor node simulator. Like Avrora, it simulates the execution of individual instructions of sensor network programs and the operation of each device. However, AtEmu's performance has limited its adoption; it is up to 20 times slower than Avrora in simulating networks of tens or hundreds of nodes. Implemented in C, AtEmu suffers from some portability and extensibility problems. In contrast, Avrora's flexible architecture allows instrumentation of many parts of the simulation without making modifications to the either the simulator or the application source code. The energy model of AEON was first implemented in AtEmu, but AtEmu was abandoned in favor of Avrora due to its superior performance and flexibility.

For embedded systems development, many energy profiling tools have been presented [2][3]. However, none of these tools include models for the devices external to the microcontroller, such as the radio, external memory and sensor devices. Thus, they can only be used for the evaluation of one separate component without inter-device communication and environment interaction. However, as communication and sensing are the main purpose of sensor nodes, these tools offer only limited usability for energy evaluation in sensor networks.

Recently, Power Tossim [4] has been presented as an extension to the TinyOS [5] simulator Tossim [6] to estimate the power consumption of the Mica2 senor node. Although Power Tossim benefits from the high scalability of Tossim, the hardware abstraction approach of Tossim results in lack of detail and poor accuracy in power consumption prediction of Power Tossim, which we will address in Section VII. SensorSim [7] and Sens [8] also have extensions to model power consumption, but their high level of abstraction results in inaccurate and only coarse approximations.

III. The Avroa Simulator

Many aspects of the sensor network domain present challenges for software development that can be addressed with accurate simulation. First, debugging microcontroller programs is notoriously difficult. There is often no operating system available to provide a stable platform for experimentation, which makes on-chip debugging difficult. Often a testbed microcontroller is used with a narrow serial

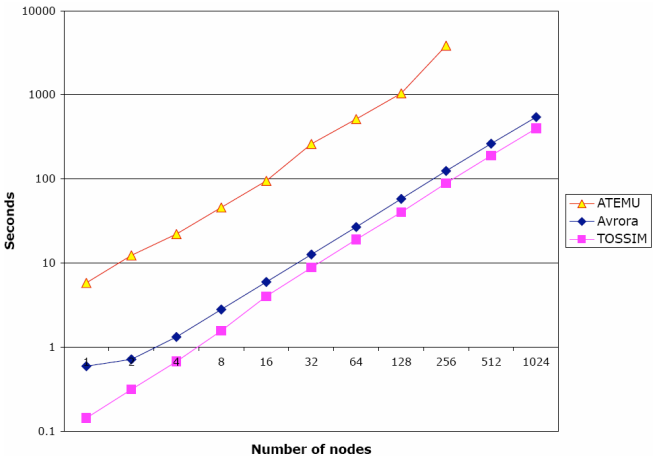


Figure 1: Performance evaluation of Avrora, Tossim and AtEmu: Avrora enables cycle accurate simulation while being 20 times faster than AtEmu and merely 50 % slower than Tossim.

Device	Current	Device	Current
CPU		**Radio** (900 MHz)	
Active	7.6 mA	Core	60 μA
Idle	3.3 mA	Bias	1.38 mA
ADC Noise	1.0 mA	Rx	9.6 mA
Power down	116 μA	Tx (-18 dBm)	8.8 mA
Power Save	124 μA	Tx (-13 dBm)	9.8 mA
Standby	237 μA	Tx (-10 dBm)	10.4 mA
Ext Standby	243 μA	Tx (-6 dBm)	11.3 mA
		Tx (-2 dBm)	15.6 mA
LED (each)	2.2 mA	Tx (0 dBm)	17.0 mA
		Tx (+3 dBm)	20.2 mA
Sensor Board	0.7 mA	Tx (+4 dBm)	22.5 mA
		Tx (+5 dBm)	26.9 mA

Table 1: Measurements of current draw form the base of the energy model.

interface which is controlled by portions the software being tested; a deadlock or crash can cause this interface to become mysteriously unresponsive. Secondly, low-level code that interfaces with hardware devices such as a software-controlled radio may have timing constraints that may be difficult to observe and verify in this way. Third, the distributed nature of sensor networks makes capturing and analyzing detailed interactions between nodes difficult for a physical network. Simulation can address these concerns by creating a controlled environment where the operation of the software and the interactions between nodes can be observed in detail.

Avrora provides a centerpiece for research and development of sensor networks: a high-resolution simulation with clock-cycle accurate execution of sensor nodes that scales to an entire network. In addition to emulating the execution of the program's instructions, each device on each node (e.g. a serial port, timers, externally wired data pins, or a software-controlled byte-oriented digital AM radio) is emulated faithfully, allowing detailed analysis and verification of node behavior. Additionally, encapsulation and modularity in Avrora's design allows new device simulations to be added easily, allowing exploration of new hardware sensors and the corresponding software control before the hardware is actually built. Accurate simulation of device behavior coupled with the modularity of the design allowed the fine-grained energy accounting of AEON to be implemented without any major modifications to Avrora.

The design of the Avrora simulator has three main contributions. First, it provides a layered, well encapsulated object-oriented software architecture that has well-defined extension points and a clean profiling interface for programs. Second, it uses the well-known concept of the event queue that keeps device operation synchronized with the main clock in an efficient manner, rather than the simple and slow approach of each device being advanced one clock cycle at a time. Third, it uses an efficient synchronization strategy that allows multiple nodes to run in parallel while preserving the timing and ordering of node interaction, resulting in increased performance on multiprocessor machines. For more on each of these, see [9].

We evaluated the scalability of Avrora over network size by comparing its performance with two other sensor network simulators. TOSSIM is a simulator included in the TinyOS distribution that compiles sensor network programs for the host platform and links them with hardware emulation libraries. Like Avrora, AtEmu is an instruction-level simulator for sensor network programs that is cycle accurate. In Figure 1, we plot the time needed to simulate a network of the given size, where half of the nodes run the *CntToRfm* TinyOS application and half of the nodes run the *RfmToLeds* application. In this experiment, we simulate a dense network where all nodes can communicate with each other, which presents a worst-case scenario for synchronization and communication possibilities. As the figure shows, Avrora's superior architecture allows it to outperform AtEmu by a factor of 20 with the same clock cycle accuracy, while being merely 50\% slower than TOSSIM, which is not cycle accurate. Such scalability coupled with cycle-accurate precision solves an open problem posed in [6]. A more detailed performance evaluation is given in [9]. In our experiments, to simulate a 512-node network with each node running *CntToRfm*, Avrora requires just 45MB of heap space.

IV. AEON'S ENERGY MODEL

In many energy consumption estimates, coarse approximations of the power consumption are derived from the number of packets transmitted and CPU duty cycles [10][11][12][13]. However, such approximations fail to capture low-level details and states of a device. For quantitative evaluation a precise and detailed low-level model of all device components, e.g. microcontroller, radio, sensors, and memory, is needed. The single states of every component form the state of the whole node. Thus, the total draw of current of a node is the sum of the currents of each component in the respective states. Our approach for modeling power consumption consists of three steps: (1) We first measure the current draw of each state of all sensor node components to calibrate our model. (2) We then implement the model resulting from these measurements, e.g. the energy consumption of each component, in a sensor node emulator. (3) We validate the results from simulation with oscilloscope measurements and battery lifetime tests

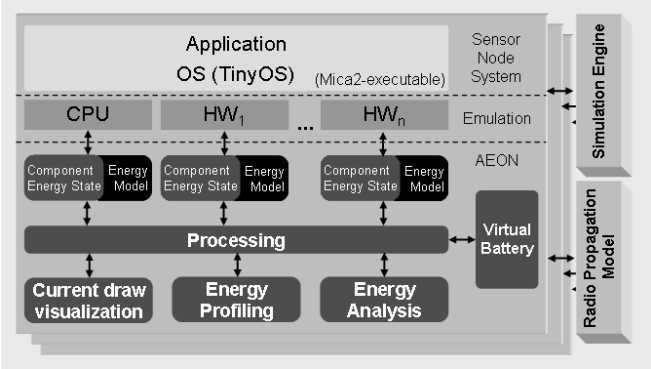


Figure 3: Block diagram of AEON's architecture.

running TinyOS applications on real hardware. We explain each of these steps in the following sections.

For AEON we focus on modeling the power consumption of the Mica2 platform. However, the energy model can be easily adapted to other platforms like ScatterWeb [14], BTnode [15] or Nymph [16], assuming the existence of an emulator for the corresponding platform.

A. *Calibrating the Energy Model*

To calibrate our energy model we first construct specific sensor node applications that keep all node components in a known state during program execution. These test applications allow us to measure the current draw of various component state combinations with highly precise ampere meters. Based on this data we extract the draw of current of each component's state and estimate its energy consumption. Finally, the power consumption of each component forms our energy model for the sensor node (see Table 1). We measured over twenty different states of three Mica2 nodes. Measurement deviation for each node was less than 0.5 %. However, due to electronic components tolerances, the results of different nodes varied by approximately 5 %. Further, our measurements indicate, that CPU access to the ADC, UART or SPI bus do not draw more current than any other CPU instruction.

B. *AEON's Implementation*

To enable exact modeling of application execution and device state changes, we base our model on Avrora. The execution of real code guarantees accurate device timing, which allows modeling the state of every component at every point in time during program execution.

The energy model extends the implementation of each hardware component in Avrora (see Figure 3) to monitor power usage. Furthermore we added energy profiling (see Section VI) to enable a breakdown to individual source code routines and components as well as a radio propagation model to provide realistic node communication.

C. *Validation of the Energy Model*

Model validation is important for reliable and accurate results. For validation we use two different approaches: (1) oscilloscope measurements of hardware running TinyOS applications. (2) running to battery exhaustion to evaluate whether the predicted lifetime matches the actual node lifetime. To validate our model we measured standard sensor node applications with an oscilloscope over various amounts

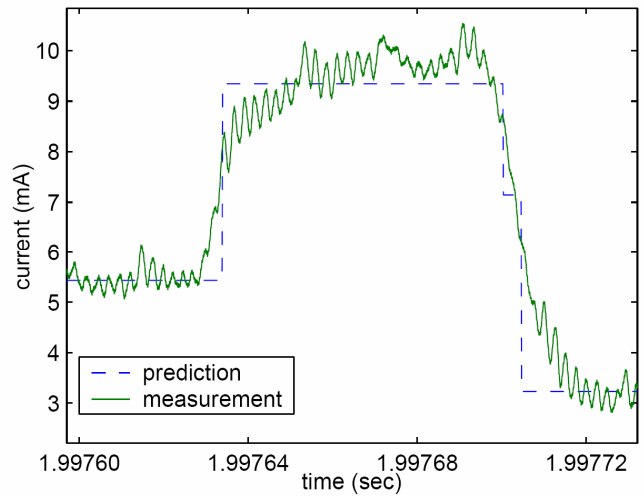


Figure 2: Comparing detailed current measurements of a oscilloscope and AEON's prediction of current draw (TinyOS *Blink* application).

of time. As example, Figure 2 shows a (noise filtered) measurement and corresponding results predicted by AEON for the TinyOS *Blink* application. The average error for these measurements is about 0.4% (standard deviation: 0.24); measurements of other applications are in the same range.

Recently, the sensor node manufacturer Crossbow published battery life tests [17] for the Mica2 node (433 MHz radio). These tests indicated that the TinyOS *CntToLedsAndRfm* application operated for 172 hours. For the corresponding battery type, AEON predicts a lifetime of approximately 168 h and 165 h for the 433 MHz radio and the 933 MHz radio, respectively. The prediction error of 2% is due to voltage fluctuation, battery tolerance and the fact that the nodes' current draws differ by as much as 5% as mentioned in section IV.A. Based on these results we consider our energy prediction tool AEON as highly precise.

V. Evaluating Applications With AEON

A detailed analysis of the energy efficiency of applications and schemes like power management and routing is crucial for sensor node applications. With AEON, system and application developers can now perform this analysis for future systems and applications using simulation. Many of the applications and schemes that we will evaluate in the following section may have been already been analyzed by their developers for energy efficiency to improve node lifetime, but we believe our tool allows this analysis in greater detail than previously available. Although our approach is independent from the sensor node operating system, we focus on TinyOS for this evaluation, as it is the de-facto operating system for sensor nodes.

Our model enables detailed energy evaluation for each state of all node components. Table 2 presents a breakdown for various TinyOS applications based on AEON's predictions. The table shows that the radio consumes most of the power, as it is not turned off during the transmission intervals.

A. *Evaluating TinyOS Power Management*

Efficient CPU Power Management is crucial for long

Test Application	Predicted Energy Consumption (in mJ) and Node Lifetime							
	CPU		Radio		LEDs	Sensor Board	Total	Lifetime (days)
	active	idle	rx	tx				
Blink	0.37	601.6	0	0	196.2	-	798.2	25.8
CntToLeds	0.77	601.5	0	0	590.6	-	1193	17.4
CntToLedsAndRfm	93	560.7	1651	130	589.6	-	3025	6.9
CntToRfm	92.7	560.8	1651	130	0	-	2435	8.5
RfmToLeds	82.9	565.2	1727	0.6	589.0	-	2965	7.0
SenseToLeds	1.85	601	0	0	0[1]	126	728.8	28.5
SenseToRfm	4.39	560.3	1651	130	0	126	1937	10.7

Table 2: Component breakdown for TinyOS 1.1.7. Applications were executed for 60 virtual seconds and *SenseToLeds* was configured to read sensor value 0.

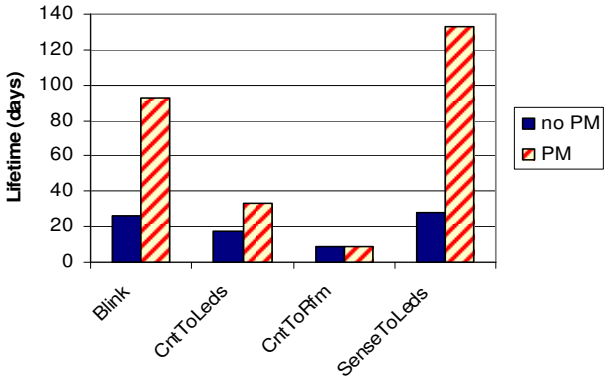


Figure 5: Predicted lifetime extension when using TinyOS Power Management

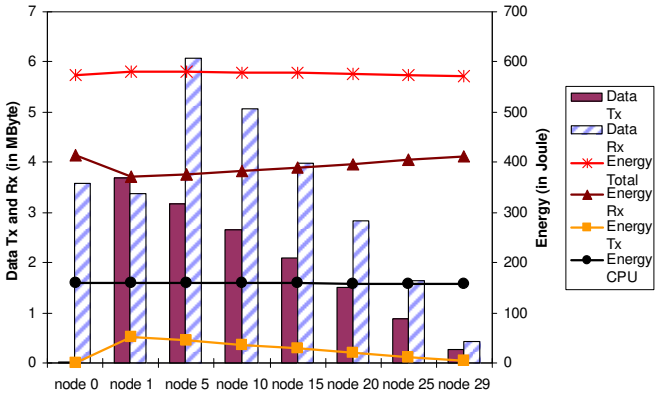


Figure 4: Data transmitted and energy consumed by 30 nodes running Surge during four virtual hours analyzed with AEON.

node and network lifetime. The ATmega128L Microcontroller of the Mica2 sensor node provides six different sleep modes, consuming between 3.3 mA and 243µA (see Table 1). TinyOS provides HPLPowerManagement [5] as an efficient power management implementation. The main advantage of this approach is that it dynamically adjusts to the load of individual nodes. It does not use static or global sleep schedules as other approaches [18]. However, as mentioned before, no deep evaluation of the energy savings of this power management scheme exist. Based on this, we used AEON to evaluate the lifetime extension this scheme provides for TinyOS applications (see Figure 5). E.g. for the *Blink* application, CPU power consumption gets reduced by factor 24, resulting in a lifetime extension of factor 3.5.

B. *Multihop Networking*

We will conclude the energy analysis with an evaluation of the multihop application *Surge*. *Surge* periodically measures the ambient light and sends the measurement over the network to the base station. Additionally each node works as a router, forwarding packets the base station.

We want to analyze the lifetime of the individual nodes in the network and determine whether their position in the network has impact on their lifetime. To have a deterministic packet flow, we arrange 30 nodes in a line so that each node can only communicate to its predecessor and successor in the line. Node 0 is the base station and node 29 is the node the furthest away from the base station.

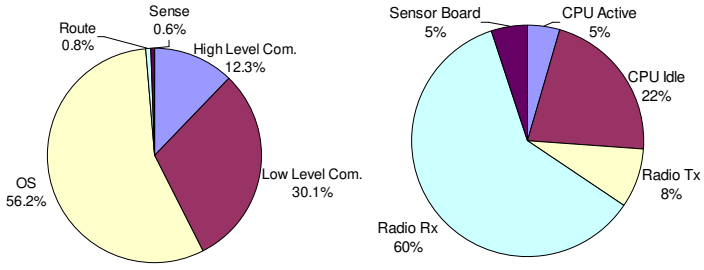
Surge has an initial setup phase where the routes to the base station are set up. To capture steady state behavior, we ran our analysis for 4 virtual hours. Although the node closest to the base station (node 1) sends about fourteen times more data than the node the furthest away from the base station, our analysis shows that it consumes only slightly more energy (see Figure 4). This is due to the fact that the transmission periods are short and so do not have much impact on the power consumption. As no power management and Low Power Listening [19] schemes are applied to *Surge*, the radio is continuously sensing for data, e.g. it is in receive mode, and thereby consuming large amounts of power.

Additionally, the CPU power consumption does not seem to depend on the amount of data transmitted or received, which we will address in the next section.

As mentioned before, recent research provides many energy efficient or energy aware applications. However, most implementations count the number of packets sent and use this information as the one and only source to qualitatively estimate the power consumption. With AEON it is now possible to quantitatively evaluate the energy efficiency of systems and software and to accurately predict the lifetime of a sensor network.

VI. Energy Profiling

Next to the analysis of power consumption of the node components, it can be important to break the CPU power consumption down to individual routines and blocks of the source code. Such an analysis allows us to determine how much energy the CPU spends in various routines such as sensing, routing and communication. Such information can be used to identify procedures consuming large amounts of energy and improve their implementation. We extended AEON to profile the applications during execution and report the energy consumption of their routines. As example we choose *Surge*, as it contains sensing as well as multihop



a) Energy Profiling: Breakdown of CPU energy consumption to system routines.

b) Breakdown to hardware components.

Figure 6: Breakdown of *Surge* to individual components with AEON.

communication. The profiling data is based on the energy profiling done for node 1 (see Section V.B) running the *Surge* application. We partitioned TinyOS and the application into five main components: (1) **High Level Communication:** Packet handling, queuing and radio modes, (2) **Low Level Communication:** SPI data transfer, (3) **Operating System (OS):** Scheduling, interrupts, timer, (4) **Routing** and (5) **Sensing**.

The analysis (see Figure 6) shows that low-level data handling and operating system events like scheduling and timer handling consume much of the CPU processing time. It can be concluded that the basic approach to increase the energy efficiency of sensor node applications is to improve the radio and the operating system implementation.

VII. AEON AND POWER TOSSIM

Although Power Tossim and our approach are based on nearly the same measurements (see Table 1 and [4]), the results in terms of energy consumption are quite different. For example, Power Tossim predicts a power consumption of 2620 mJ per 60 seconds for the *CntToLedsAndRfm* application, our prediction model results in 3023 mJ, a difference of 15%. However, the breakdown to individual node components is not modeled accurately in Power Tossim; it predicts a power consumption of 1.61 mJ per 60 seconds for the CPU in active mode for the same application, while AEON predicts a power consumption of 92.08 mJ, a difference of more than 5700 %. Power Tossim is unable to predict accurately the time spent in idle and active CPU mode, and therefore the resulting energy consumption estimate is flawed.

AEON's prediction of power consumption is based on the execution of real code, capturing all low-level events of the application in a cycle accurate way. For example, the radio causes the SPI interface to fire an interrupt that wakes the CPU from idle mode approximately every 460 µs. However, Tossim uses hardware abstraction to model the device components. Consequently, it does not model such interrupts precisely, resulting in the inaccurate prediction of power consumption in Power Tossim.

VIII. CONCLUSION AND FUTURE WORK

As sensor networks gain more importance in the research communities, we believe that it is crucial to have tools for analyzing and evaluating their behavior accurately.

Avrora fills a need for precise and scalable sensor node simulation, allowing large networks to be simulated with cycle accuracy. Its flexible framework allows detailed instrumentation of running programs and has proved useful in implementing an energy monitoring and profiling tool.

Energy is a limited resource for sensor nodes making a deep evaluation of energy consumption and accurate lifetime prediction crucial before deployment. Erroneous lifetime prediction may causes high costs and may even render a network of small devices useless, increasing the need for precise lifetime prediction. The detailed energy analysis presented for the TinyOS Power Management and the *Surge* application shows how node lifetime can be analyzed in detail and extended by targeting components that consume the most energy first. Energy Profiling provides the ability to break application power consumption down to individual routines of the operating system.

Avrora does not yet model all of the sensor node components, and currently lacks a uniform mechanism to model the effects of a changing environment, and how a sensor network may react to such changes. Subtle issues such as clock drift due to hardware clocks running at different rates must also be addressed. Currently, much work is ongoing to address these concerns and further refine the performance, scalability, and usability of Avrora.

REFERENCES

[1] G. J. Polley et al., "ATEMU: A Fine-Grained Sensor Network Simulator,"in Proc. of SeCon, Oct. 2004.

[2] T. K. Tan, A. Raghunathan, and N. Jha, "EMSIM: An Energy Simulation Framework for an Embedded Operating System," in *Proc. of ISCAS*, 2002.

[3] A. Sinha and A. P. Chandrakasan, "JouleTrack - A Web Based Tool For Software Energy Profiling," in *Proc. of Design Automation Conference*, June 2001.

[4] V. Shnayder *et al.*, "Simulating the Power Consumption of Large-Scale Sensor Network Applications," in *Proc. of SenSys*, Nov. 2004.

[5] P. Levis *et al.*, "The Emergence of Networking Abstractions and Techniques in TinyOS," in *Proc. of NSDI*, March 2004.

[6] P. Levis *et al.*, "TOSSIM: accurate and scalable simulation of entire tinyOS applications," in *Proc. of SenSys*, 2003.

[7] S. Park, A. Savvides, and M. B. Srivastava, "SensorSim: a simulation framework for sensor networks," in *Proc. of WSWiM*, 2000.

[8] S. Sundresh, W. Kim, and G. Agha, "SENS: A Sensor, Environment and Network Simulator," in *Proc. of Annual Simulation Symposium*, 2004.

[9] B. L. Titzer, "Avrora: The AVR Simulation and Analysis Framework," Master's thesis, UCLA, June 2004.

[10] S. R. Madden *et al.*, "TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks," in *Proc. of OSDI*, Dec. 2002.

[11] M. Duarte and Y. Hu, "Distance Based Decision Fusion in a Distributed Wireless Sensor Network," *Telecomm. Systems*, vol. 26, 2004.

[12] A. Woo and D. E. Culler, "A transmission control scheme for media access in sensor networks," in *Proc. of MobiCom*, July 2001.

[13] R. C. Shah and J. Rabaey, "Energy Aware Routing for Low Energy Ad Hoc Sensor Networks," in *Proc. of WCNC*, March 2002.

[14] H. Ritter *et al.*, "A Highly Flexible Testbed for Studies of ad-hoc Network Behaviour," in *In Proc. of 3rd Int. Workshop on Wireless Local Networks (WLN)*, Oct. 2003.

[15] J. Beutel *et al.*, "Prototyping wireless sensor networks with btnodes," in *In Proc. of EWSN*, Jan. 2004.

[16] S. Bhatti *et al.*, "MANTIS OS: An Embedded Multithreaded Operating System for Wireless Micro Sensor Platforms," *MONET, Special Issueon Wireless Sensor Networks*, 2004.

[17] J. Suh *et al.*, *MICA2 AA Battery Pack Service Life Test*, Crossbow Technology, Inc., 2004.

[18] W. Ye, J. Heidemann, and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *Proc. of INFOCOM*, June 2002.

[19] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," in *Proc. of SenSys*, Nov. 2004.