Accurate Prediction of Power Consumption in Sensor Networks

Olaf Landsiedel, Klaus Wehrle, Stefan Götz
Protocol Engineering and Distributed Systems Group
University of Tübingen, Germany
{olaf.landsiedel, klaus.wehrle,stefan.goetz}@uni-tuebingen.de

Abstract

Energy consumption is a crucial characteristic of sensor networks and their applications as sensor nodes are commonly battery-driven. Although recent research focuses strongly on energy-aware applications and operating systems, energy consumption is still a limiting factor. Once sensor nodes are deployed, it is challenging and sometimes even impossible to change batteries. As a result, erroneous lifetime prediction causes high costs and may render a sensor network useless before its purpose is fulfilled.

In this paper, we present AEON (Accurate Prediction of Power Consumption), a novel evaluation tool to quantitatively predict energy consumption of sensor nodes and whole sensor networks. Our energy model, based on measurements of node current draw and the execution of real code, enables accurate prediction of the actual energy 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.

1. 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.

Sensor nodes are usually battery driven. Due to this limited energy resource, energy consumption is a crucial design factor for hardware and software developers. Although hard- and software are strongly tied together in mobile and embedded device development, energy consumption is still a minor issue for software development. To enable efficient co-design of hard- and software for such devices, a deep evaluation of applications and systems in terms of energy consumption is crucial.

Furthermore, no breakthroughs in battery technology are to be expected in the next years. As long as new technologies like fuel cells do not advance from prototype level to mass production, batteries and thus energy consumption will be the limiting factor. Batteries often constitute more than 50 % of the device's weight and volume. For example, the Mica2 [9] sensor node, has a weight of 18 g. However, its two AA batteries weigh between 20 to 30g each.

For developers, it is crucial to evaluate the energy consumption of applications accurately since the choice of algorithms and programming styles may strongly influence energy consumption. Once nodes are deployed, it is challenging and sometimes even impossible to change batteries. As a result, erroneous lifetime prediction may cause high costs and even render a sensor network useless before its purpose is fulfilled.

In this paper, we present AEON (Accurate Prediction of Power Consumption), a novel evaluation tool to predict energy consumption of sensor nodes. Based on the execution of real application and OS code and measurements of node current draw, our model enables accurate prediction of the actual energy consumption of nodes. Thus, it prevents erroneous assumptions on device and network lifetime. Moreover, AEON gives adequate and very fine grained feedback to an software engineer on the energy efficiency of his/her code. Such a detailed prediction allows the comparison of different low power and energy aware approaches in terms of energy efficiency and the estimation of the overall lifetime of a sensor network.

The remainder of this paper is structured as follows. First, we discuss background and related work in Section 2. Section 3 presents the energy model as the basis of our tool and a detailed validation. Section 4 presents possible applications of our tool AEON by quantitatively evaluating energy and power aware components, e.g. Power Management, Low Power Listening, and Multihop Routing. Next, Section 5 introduces energy profiling of applications and operating systems source code to give feedback about en-

ergy efficiency of the developed code. Section 6 compares the level of detail and accuracy of AEON with related work. Finally, Section 7 concludes the paper and presents future work.

2. Related Work

Coarse approximations of the energy consumption are usually derived from the number of transmitted packets and CPU duty cycles. However, such approximations fail to capture low-level details and states of a device. For a quantitative evaluation, a precise and detailed low-level model of the device is needed. Although recent research provides many energy-efficient or energy-aware applications, only a very limited number [23, 4] has been evaluated deeply in terms of quantitative energy consumption by measurements of current draw. Most implementations [10, 1, 22, 14] count the number of packets sent and use this information as the only source to qualitatively estimate energy consumption.

For hardware development, many energy profiling tools have been presented [7, 6], focusing on components of traditional CPUs, e.g. memory, cache, data- and control path. New tools [20, 16] model the energy consumption of processors and microcontrollers for embedded systems as they add models for the memory, serial communication, and other parts of the microcontroller. These tools mainly focus on hardware development and not on the evaluation of software for distributed systems. None of the tools include models for devices next to the microcontroller like communication, external memory, sensors, and actuators. Thus, they can only be used for the evaluation of single separate components without inter-device communication and interaction with the environment interaction. However, as communication and sensing are the main purpose of sensor nodes, these tools are only of limited use for energy evaluation in sensor networks.

Recently, Power Tossim [15] has been presented as an extension to the TinyOS [9] simulator Tossim [8] to estimate the energy consumption of the Mica2 senor node. Since Tossim provides an abstract model of the node and compiles applications to x86 executables, it does not model all hardware details, such as interrupts and execution time. As an accurate prediction of code execution time is important to estimate the energy consumption of a device, Power Tossim maps Mica2 binaries to x86 executables by matching basic code blocks (code blocks without branches) to determine execution time. Although Power Tossim benefits from the high scalability of Tossim, hardware abstraction in Tossim results in a lack of detail and accuracy in the energy consumption prediction of Power Tossim, which we will address in Section 6. SensorSim [11] and Sens [19] also have extensions to model energy consumption but their high level

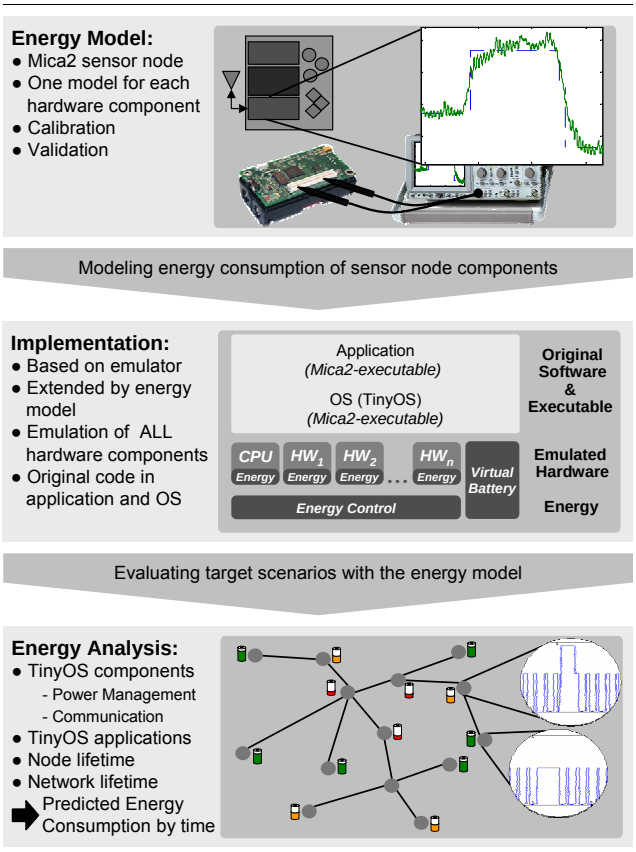


Figure 1. AEON: Modeling and analyzing energy consumption of sensor node systems and applications.

of abstraction results in inaccurate and only coarse grained approximations.

3. AEON's Energy Model

Since the Mica2 node [9] is currently the most commonly used sensor node, we focus on modeling the energy consumption of the Mica2 platform in this paper. However, AEON can be easily applied to other sensor node platforms by repeating the following procedure on the target platform.

As sensor nodes consist of several components such as microcontrollers, radios, sensors, and memory, a detailed low-level model of all these devices is necessary to enable accurate prediction of energy consumption. The application and external events influence the program execution and so the state of a node. For example, applications turn on and off components like the radio and timer interrupts change the CPU from sleep to active mode. As such state changes happen frequently and each state consumes a different amount of power, the states and the timing of the state changes need to be modeled accurately. The single states of every com-

Device	Current	Device	Current
CPU		**Radio** (900 MHz)	
Active	$7.6\,mA$	Core	$60\,\mu A$
Idle	$3.3\,mA$	Bias	$1.38\,mA$
ADC Noise	$1.0\,mA$	Rx	$9.6\,mA$
Power down	$116\,\mu A$	Tx (-18 dBm)	$8.8\,mA$
Power Save	$124\,\mu A$	Tx (-13 dBm)	$9.8\,mA$
Standby	$237\,\mu A$	Tx (-10 dBm)	$10.4\,mA$
Ext Standby	$243\,\mu A$	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.

ponent form the state of the whole node. The total current draw of a node is the sum of the currents of each component in the respective states.

The first challenge for the accurate prediction of energy consumption is to build a precise and detailed low-level energy model of the sensor node. Our approach for such a model consists of three steps (see Fig. 1): (1) We measured the current draw of each state of all sensor node components to calibrate our model. (2) The model derived from these measurements, e.g. the power consumption of all component states, is implemented in a sensor node emulator. (3) the model is validated by oscilloscope measurements and battery lifetime tests running TinyOS applications. We address these steps in the following sections.

3.1. Calibrating the Energy Model

To calibrate our energy model we implemented specific applications which keep all node components in a certain state during program execution. These test applications allow us to measure the current draw of various combinations of component states with highly precise ampere meters. Based on this data, we extracted the draw of current of each component's state and estimated its power consumption. Finally, the estimated power consumption of each component forms our energy model for this type of sensor node (see Table 1). We measured over twenty different states of three Mica2 nodes. The measurements deviated for each node 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 does not draw more current than any other CPU instruction.

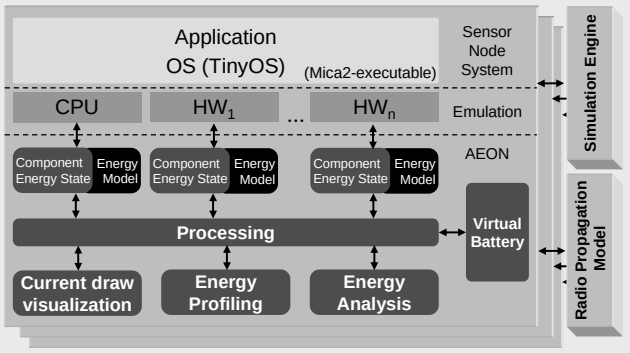


Figure 2. Block diagram of AEON's architecture.

3.2. AEON's Implementation

To enable exact modeling of application execution and device state changes, we base our model on a sensor node emulator. The emulation of the node and the execution of real application and OS code allows to model the state of every single component at every point in time during program execution. Furthermore, emulation is independent of the OS, so any operating systems and applications for the Mica 2 node can be evaluated.

We implemented AEON on top of AVRORA [21], a highly scalable sensor node emulator. Performance analysis [21] shows that AVRORA is only 50 % slower than the simulator TOSSIM [8] and about 20 times faster than the sensor node emulator ATEMU [13]. The slowdown of 50 % is more than acceptable for energy modeling as the execution of real code and realistic device timing is crucial for accurate energy modeling – this is not the case for high level simulation.

The energy model extends the implementation of each hardware component in AVRORA by monitoring its power usage during emulation (see Fig. 2). Furthermore, we added energy profiling (see Sect. 5) to enable a breakdown to individual source code routines and components. Additionally a radio propagation model provides realistic node communication. Our performance analysis shows that the overhead added to AVRORA is minimal, as only the state changes of the components are monitored.

3.3. Validation of the Energy Model

Validating the model is important for reliable and accurate results. For validation, we use two different approaches: (1) Oscilloscope measurements of TinyOS applications. (2) Long time validation to evaluate whether the predicted lifetime matches the actual node lifetime.

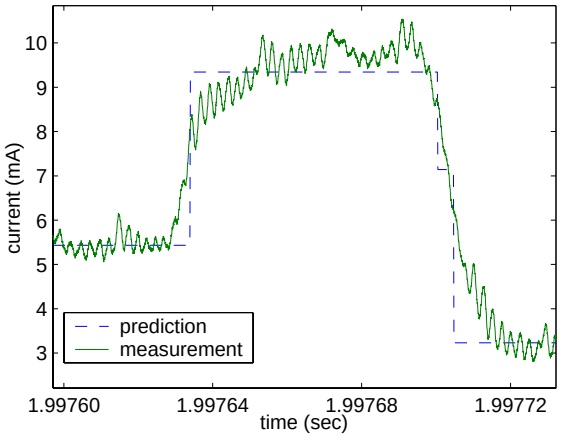


Figure 3. Comparing detailed current measurements of a oscilloscope and AEON's prediction of current draw (TinyOS `Blink` application).

To validate our model, we measured standard sensor node applications with an oscilloscope over various amounts of time. For example, Fig. 3 shows a (noise filtered) measurement and corresponding results predicted by AEON for the TinyOS `Blink` application. The average error for this 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 [18] for the Mica2 node (433 MHz radio). Using the `CntToLedsAndRfm` TinyOS application, the node operated for 172 hours before the battery voltage dropped below the Mica2 operating limit of 2.1 V. According to the battery datasheet [3], two AA batteries of this type provide 2400 mAh of energy before their total voltage drops below 2.1 V. With an average voltage of 2.4 V, our model 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 5 % as mentioned in section 3.1. Based on these results, we consider our energy prediction tool AEON to be very precise.

4. Evaluating TinyOS and Applications

In this section, we show how the energy model can be used to evaluate the energy efficiency of sensor node applications and operating systems. Based on this analysis, we are also able to determine where and how sensor node applications can be improved in terms of energy efficiency. Although our approach is independent of the sensor node operating system, we focus on TinyOS for this evaluation as it is the defacto operating system for sensor nodes. As an example, we analyze the application `CntToLedsAndRfm` as well as the energy savings of the TinyOS Power Management, Low Power Listening, and routing schemes.

As mentioned before, a detailed analysis of the energy efficiency of applications and schemes like Power Management and routing is crucial for sensor node applications. However, we are not aware that the applications and schemes, which we will evaluate in the following sections, have been analyzed by their developers in terms of energy efficiency. With AEON, system and software developers can now do this while implementing future systems and applications.

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. Furthermore, a detailed current graph is provided for the `CntToLedsAndRfm` application (see Fig. 4). The figure and table show that the radio consumes most of the energy as it is not turned off during the transmission intervals. During reception and transmission, the radio fires interrupts approximately every $460\,\mu s$ to clock in or out data. As a result, the CPU changes frequently from idle to active mode. Thus, also the CPU consumes a substantial amount of energy – much more than one would expect from such a simple application.

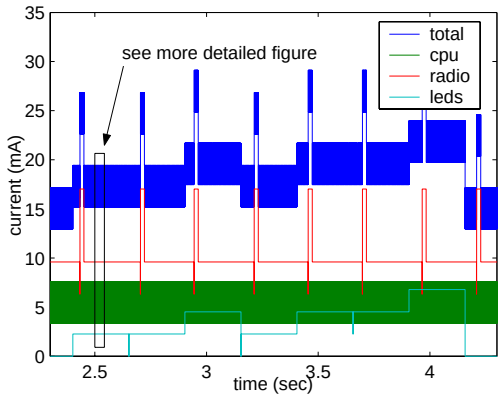
4.1. Evaluating TinyOS Power Management

Efficient CPU Power Management is crucial for long node and network lifetime. The ATmega 128L of the Mica2 sensor node provides six different sleep modes, which consume between $3.3\,mA$ and $243\,\mu A$ (see Table 1). When more parts of the controller are shut down, it consumes less power during sleep. However, it wakes up from fewer sources and wake-up takes longer. The selection of an appropriate sleep mode heavily depends on the application and operating system properties.

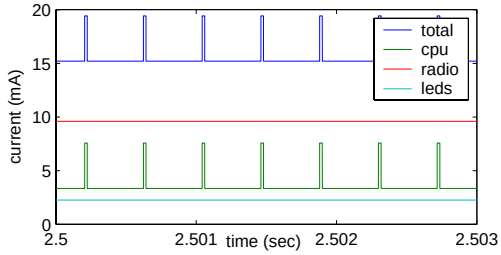
TinyOS provides HPLPowerManagement [9] as an efficient power management implementation. The main advantage of this approach is that it dynamically adopts to the current load of a node. It does not use static or global sleep schedules like other approaches [23] do. However, we are not aware of a evaluation of the energy savings this power management scheme provides. Based on this, we used AEON to evaluate the lifetime extension this scheme provides for TinyOS applications (see Fig. 5). With the `Blink` application, for example, the CPU energy consumption is reduced by a factor of 24, resulting in a lifetime extension of factor 3.5. The CPU only consumes 11 % of the total energy, without power management it is 75 %. The remaining 89 % and 25 %, respectively, are consumed by the LEDs. For the `CntToLeds` application, power manage-

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 emulated seconds.



(a) Coarse grained view.



(b) Fine grained view. Spikes are mainly caused by radio interrupts (CPU: change from idle to active mode).

Figure 4. Analyzing TinyOS applications with AEON: Predicted current draw of the hardware components (`CntToLedsAndRfm`).

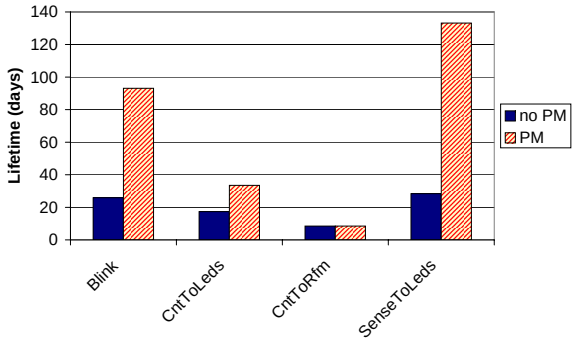


Figure 5. Predicted lifetime extension when using TinyOS Power Management.

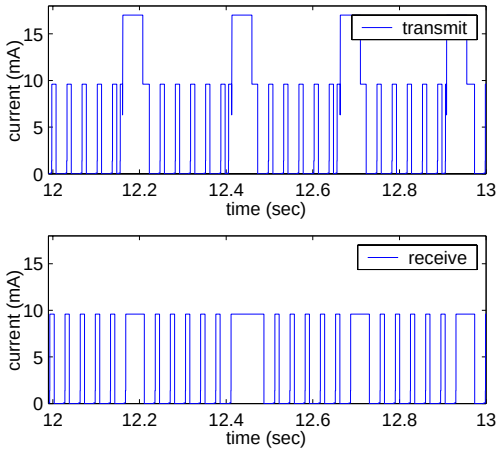
ment even reduces the CPU's fraction of the total energy consumption down to 4 %.

4.2. Analyzing TinyOS Low Power Listening

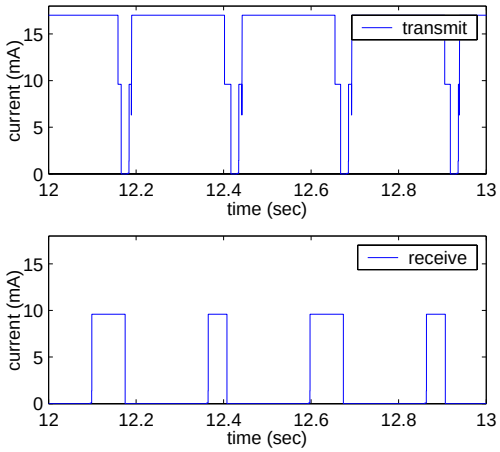
However, applications using the radio do not benefit directly from the power management approach described in the previous section. The Mica2 microcontroller cannot wake up from sleep modes (except from the Idle mode) by radio interrupts. As a result, the microcontroller cannot be sent to an efficient sleep mode when the radio is turned on.

Similar to Aloha [2], Low Power Listening [12] reduces the duty cycles of the radio by periodically switching the radio on and off. When the radio is turned off, the microcontroller can be sent to an efficient sleep mode. To ensure reliable data reception, the length of the preamble equals the interval that the radio is turned back on. Thus, if the radio is turned on every $50\,ms$ to check for incoming packets, the preamble must be at least $50\,ms$ long to ensure reception.

Using low radio duty cycles, the time between the "radio on" intervals is long, resulting in a long transmission preamble and thus a long transmission time. A long transmission time results in high energy consumption for the sender as the radio is switched on and transmitting for a long interval. Therefore, low radio duty cycles result in high transmission costs in terms of energy, whereas receiving packets does not consume a large amount of energy since the radio can be switched off for long durations. For example, Low Power Listening with $1.3\,kbps$ reduces the energy con-



(a) Low Power Listening mode 1 (5.7kbps)



(b) Low Power Listening mode 4 (1.3kbps)

Figure 6. Predicted radio current draws for different Low Power Listening modes. Low radio duty cycles result in low receiver- and high sender energy consumption

sumption of `RfmToLeds` by 59 %, extending node lifetime by factor 2.4 (see Table 3 and Fig. 6). However, as mentioned above this mode increases the transmitter energy consumption, resulting in a 30 % lower node lifetime for `CntToLedsAndRfm`. Low Power Listening at $1.3\,kbps$ is the most efficient radio duty cycle for the "count" applications, as their communication rates match the available bandwidth of this mode best. Lower radio duty cycles result in packet loss.

The main disadvantage of this approach is that schedules are static and apply to all nodes. Usually, nodes at the perimeter of a network send much less packets than nodes close to the base station. The closer a node is to the base station, the more packets it has to route. Since the schedule is static for the whole network, the node sending and receiving the most packets dictates the schedule of all other nodes.

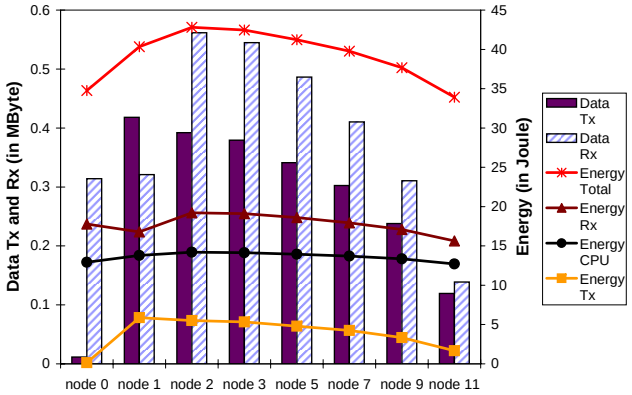


Figure 7. Data transmitted and energy consumed by 12 nodes running `Surge` during thirty virtual minutes analyzed with AEON.

4.3. Multihop Networking: `Surge`

We started our TinyOS evaluation with the analysis of the power management of a single node, then focused on the inter node communication with an analysis of Low Power Listening. We conclude the analysis with an evaluation of the multihop application `Surge`. `Surge` periodically measures the ambient light and sends the measurement results across the network to the base station. Additionally each node works as a router, forwarding packets to the base station.

Using Low Power Listening and power management, we analyze the lifetime of the individual nodes in the network and determine whether their position in the network has an impact on their lifetime. To have a deterministic packet flow, we put twelve nodes in a row so that each node can only communicate to its predecessor and successor in the line. Node 0 is the base station and node 11 is the furthest away from the base station.

For transmission, we choose Low Power Listening with 5.7 kbps, as placing twelve nodes in a line results in communication rate of about 4.3 kbps. Nodes further away from the base station route less traffic and have longer sleep periods. Thus, they consume less energy. For example, node 11 consumes about 21% less energy than node 2 (see Fig. 7).

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 only source to qualitatively estimate the energy 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.

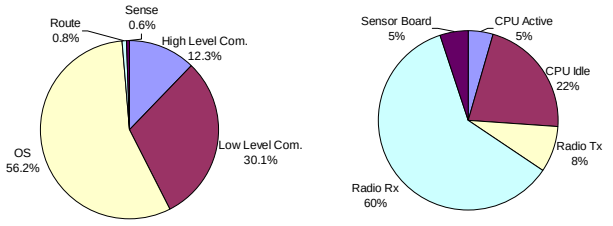
Max. Radio Throughput	Test Application	Predicted Energy Consumption (in mJ) and Node Lifetime								
		CPU			Radio		LEDs	Total	Lifetime	Energy
		active	idle	sleep	rx	tx			(days)	saving (%)
5.7kbps	`CntToLedsAndRfm`	226.5	245.5	9.8	481.2	573.2	590.7	2130	9.6	29.7
	`RfmToLeds`	207.0	217.6	10.9	701.1	1.0	590.2	1731	11.8	41.6
2.5kbps	`CntToLedsAndRfm`	123.6	310.5	16.1	212.4	1367	590.4	2621	7.8	13.5
	`RfmToLeds`	84.3	156.8	15.1	486.6	2.2	589.9	1336	15.5	54.9
1.3kbps	`CntToLedsAndRfm`	131.3	509.4	2.3	135.0	2587	589.9	3955	5.25	-30.6
	`RfmToLeds`	52.9	142.0	16.2	426.0	4.1	587.4	1229	16.8	58.5

Table 3. Predicted energy savings from TinyOS Low Power Listening modes 1, 2, and 4 with a maximum radio throughput of 5.7, 2.5, and 1.3 kbps respectively.

5. Energy Profiling

Apart from the analysis of energy consumption of the node components, it is very important to break the CPU energy consumption down to individual routines and blocks of the source code, which we address in this section. Such an analysis allows to determine how much energy the CPU spends in various routines like sensing, routing, and low level transmission and reception. Thus, it allows to identify procedures consuming a huge amount of energy and so to improve their implementation. We extended AEON to profile the applications during execution and report the energy consumption of their routines. We choose `Surge` as an example, it contains sensing as well as multihop communication. The profiling data is based on the energy profiling done for node 1 (without Low Power Listening, see Sec. 4.3) running the `Surge` application. For profiling, we map source code functions to the corresponding object code addresses and log all functions calls during program execution. We grouped the function calls and 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 Fig. 8) shows that low-level data handling and operating system events like scheduling and timer handling consume a large portion 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.



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

(b) Breakdown to hardware components.

Figure 8. Breakdown of `Surge` to individual components with AEON.

6. AEON and Power Tossim

Although Power Tossim and our approach base on nearly the same measurements (see Table 1 and [15]), the results in terms of energy consumption are quite different. For example, Power Tossim predicts an energy consumption of $2620\,mJ$ per 60 seconds for the `CntToLedsAndRfm` application, while AEON predicts $3023\,mJ$, resulting in an error of 15 %. However, the breakdown to individual node components seems not be modeled accurately in Power Tossim, because it predicts an energy consumption of $1.61\,mJ$ per 60 seconds for the CPU in active mode for the same application. AEON, however, predicts an energy consumption of $92.08\,mJ$, resulting in a difference of more than 5700 %. The cycles and so the energy spent in the idle and active CPU state differ extremely between Power Tossim and AEON. AEON's prediction of energy consumption is based on the execution of real code in an emulator. Thus, it inherently captures all low level events of the application. However, Tossim is a sensor node simulator, it uses hardware abstraction to model the device components. Consequently, it does not model all interrupts and all state changes of the CPU, which results in a lack of accuracy in Power Tossim. When a timer interrupt is fired in TinyOS, the timer often just needs to be reloaded before the CPU switches back to a sleep mode. Reloading the counters does not require many CPU cycles but such interrupts occur frequently. Thereby, they contribute heavily to the energy consumption, so these interrupts and the corresponding CPU cycles need to be considered. The radio causes the SPI interface to fire an in-

terrupt approximately every $460\,\mu s$ (Fig. 4(b)), which wakes up the CPU from idle mode every time. As mentioned above, Tossim models such SPI and timer interrupts not precisely, resulting in the inaccurate prediction of energy consumption in Power Tossim.

7. Conclusion and Future Work

As sensor networks gain more importance in the research community, we believe that it is crucial to have tools for analyzing and evaluating their behavior accurately. Energy is a limited resource for sensor nodes. Thus, a deep evaluation of energy consumption and accurate prediction of lifetime is crucial before deployment. As devices are commonly embedded into the environment, it is very challenging and sometimes even impossible to change batteries, when nodes run out of energy. As a result, nodes may fail and not fulfill their purpose, long before the expected lifetime is reached. Erroneous lifetime prediction causes high costs and may even render a network of small devices useless, making a deep and accurate energy analysis and precise lifetime prediction a must.

The indepth energy analysis presented for the TinyOS Low Power Listening, Power Management, and the `Surge` application, shows how node lifetime can be analyzed in detail and efficiently extended. Energy Profiling allows to break down application energy consumption to individual routines. Our work shows how applications can be profiled in terms of energy consumption and to detect which parts of the operating system should be improved to increase energy efficiency.

Although AEON enables energy analysis of many sensor node applications, it does not yet support all node components. Currently, we are extending AVRORA with EEPROM emulation and better sensor models. Thus, we will be able to analyze of the energy efficiency of network reprogramming tools like MOAP [17] and Deluge [5].

References

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

[2] A. El-Hoiydi. Aloha with Preamble Sampling for Sporadic Traffic in Ad-Hoc Wireless Sensor Networks. In *In Proc. of IEEE ICC*, April 2002.

[3] Eveready Battery Co. *ENERGIZER NO. A91 Datasheet*.

[4] B. Hohlt, L. Doherty, and E. Brewer. Flexible Power Scheduling for Sensor Networks. In *Proc. of IPSN*, April 2004.

[5] J. W. Hui and D. Culler. The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale. In *Proc. of SenSys*, Nov. 2004.

[6] M. Kamble and K. Ghose. Energy-Efficiency of VLSI Caches: A Comparative Study. In *10th Int. Conf. on VLSI Design*, 1997.

[7] P. Landman and J. Rabaey. Activity-Sensitive Architectural Power Analysis. *IEEE Trans. on CAD*, 1996.

[8] P. Levis et al. TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. In *Proc. of SenSys*, 2003.

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

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

[11] S. Park, A. Savvides, and M. B. Srivastava. SensorSim: a Simulation Framework for Sensor Networks. In *Proc. of WSWiM*, 2000.

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

[13] J. Polley et al. ATEMU: A Fine-Grained Sensor Network Simulator. In *Proc. of SeCon*, Oct. 2004.

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

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

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

[17] T. Stathopoulos, J. Heidemann, and D. Estrin. A Remote Code Update Mechanism for Wireless Sensor Networks. Technical report, CENS UCLA, 2004.

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

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

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

[21] B. L. Titzer, D. K. Lee, and J. Palsberg. Avrora: Scalable Sensor Network Simulation with Precise Timing. In *Proc. of IPSN*, April 2005.

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

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