Low-Power Wireless Sensor Networks

Rex Min, Manish Bhardwaj, Seong-Hwan Cho,
Eugene Shih, Amit Sinha, Alice Wang, Anantha Chandrakasan
Department of EECS
Massachusetts Institute of Technology
Cambridge, MA 02139

e-mail: {anantha, manishb, chosta, rmin, eugene, sinha, aliwang}@mtl.mit.edu

Abstract - Wireless distributed microsensor systems will enable fault tolerant monitoring and control of a variety of applications. Due to the large number of microsensor nodes that may be deployed and the long required system lifetimes, replacing the battery is not an option. Sensor systems must utilize the minimal possible energy while operating over a wide range of operating scenarios. This paper presents an overview of the key technologies required for low-energy distributed microsensors. These include power aware computation/communication component technology, low-energy signaling and networking, system partitioning considering computation and communication trade-offs, and a power aware software infrastructure.

I. Introduction

The design of micropower wireless sensor systems has gained increasing importance for a variety of civil and military applications. With recent advances in MEMS technology and its associated interfaces, signal processing, and RF circuitry, the focus has shifted away from limited macrosensors communicating with base stations to creating wireless networks of communicating microsensors that aggregate complex data to provide rich, multi-dimensional pictures of the environment. While individual microsensor nodes are not as accurate as their macrosensor counterparts, the networking of a large number of nodes enables high quality sensing networks with the additional advantages of easy deployment and fault-tolerance. These characteristics that make microsensors ideal for deployment in otherwise inaccessible environments where maintenance would be inconvenient or impossible [1][2][3].

The potential for collaborative, robust networks of microsensors has attracted a great deal of research attention. The WINS [5] and PicoRadio [6] and projects, for instance, aim to integrate sensing, processing and radio communication onto a microsensor node. Current prototypes are custom circuit boards with mostly commercial, off-the-shelf components. The Smart Dust [4] project seeks a minimum-size solution to the distributed sensing problem, choosing optical communication on coin-sized "motes." The prospect of thousands of communicating nodes has sparked research into network protocols for information flow among microsensors, such as *directed diffusion* [7].

The unique operating environment and performance requirements of distributed microsensor networks require fundamentally new approaches to system design. As an example, consider the expected performance versus longevity of the microsensor node, compared with current battery-powered portable devices. The node, complete with sensors, DSP, and radio, is capable of a tremendous diversity of functionality. Throughout its lifetime, a node may be called upon to be a data gatherer, a signal processor, and a relay station. Its lifetime, however, must be on the order of months to years, since battery replacement for thousands of nodes is not an option. In contrast, much less capable devices such as cellular telephones are only expected to run for days on a single battery charge. High diversity also exists within the environment and user demands upon the sensor network. Ambient noise in the environment, the rate of event arrival, and the user's quality requirements of the data may vary considerably over time.

A long node lifetime under diverse operating conditions demands power-aware system design. In a power-aware design, the node's energy consumption displays a graceful scalability in energy consumption at all levels of the system hierarchy, including the signal processing algorithms, operating system, network protocols, and even the integrated circuits themselves. Computation and communication are partitioned and balanced for minimum energy consumption. Software that understands the energy-quality tradeoff collaborates with hardware that scales its own energy consumption accordingly. Using the MIT μAMPS project as an example, this paper surveys techniques for system-level power-awareness.

II. Node Architecture Considerations

Figure 1 outlines the architecture of a μAMPS sensor node. The power subsystem consists of a battery with DC-DC conversion to the appropriate voltages required by the system. Digitized data from analog sensors are processed by a Stron-

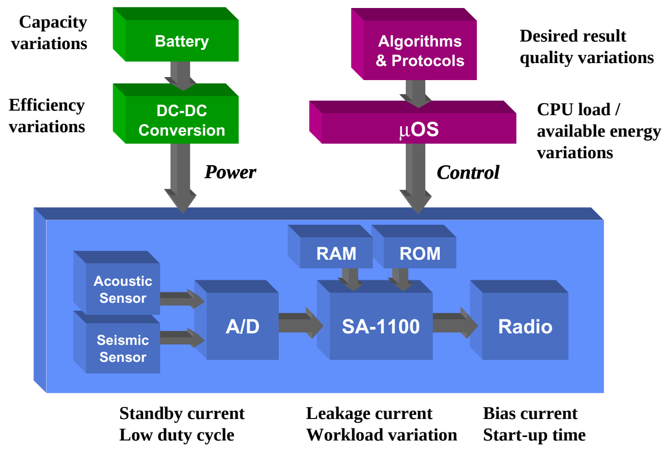

Figure 1: Architectural Overview of the MIT μAMPS sensor node.

gARM SA-1100, which communicates with adjacent nodes through a 2.4 GHz radio transceiver. A small operating system (µOS), sensor algorithms and network protocols are resident in ROM. The power-aware system is sentient of the many variables that define the energy consumption at each architectural block, from leakage currents in the integrated circuits, to the output quality and latency requirements of the end user, to the duty cycles of radio transmission.

A. Computation and Dynamic Voltage Scaling

Energy consumption in a static CMOS-based processor can be classified into switching and leakage components. The switching energy is expressed as $E_{switch} = C_{tot}V_{dd}^{2}$ where C_{tot} is the total capacitance switched by the computation and V_{dd} is the supply voltage. Energy lost due to leakage currents is modeled with an exponential relation to the supply voltage [8]:

$$_{leak} = (V_{dd}t)I_O e^{\frac{V_{th}}{(nV_T)}} \qquad (1)$$

where V_{th} is the device threshold voltage and V_T is the thermal voltage. While switching energy is usually the more dominant of the two components [9], the scaling of device thresholds for low-voltage operation coupled with the low duty cycle operation of a sensor node can induce precisely the opposite behavior. Figure 2 demonstrates that, for sufficiently low duty cycles or high supply voltages, leakage energy can exceed switching energy. For example, when the duty cycle of the SA-1100 processor is 10%, the leakage energy is more than 50% of the total energy consumed. Techniques such as dynamic voltage scaling and the progressive shutdown of idle components (discussed in Section IV.) mitigate the energy consumption penalties of low duty cycle operation.

Dynamic voltage scaling (DVS) exploits variabilities in processor workload and latency constraints and realizes this energy-quality trade-off at the circuit level [10][11]. As discussed above, the switching energy of any particular computation is $E_{switch} = C_{tot}V_{dd}^{2}$, a quantity that is independent of time. Reducing V_{dd} offers a quadratic savings in switching energy at the expense of additional propagation delay through static logic. Hence, if the workload on the processor is light, or the latency tolerable by the computation is high, we can reduce V_{dd} and the processor clock frequency together to trade off latency for energy savings.

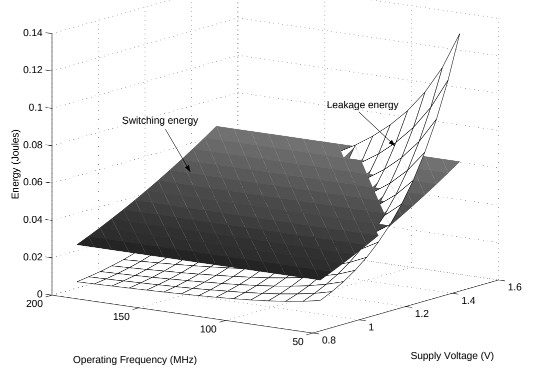

Figure 2: Comparison of leakage and switching energy in SA-1100.

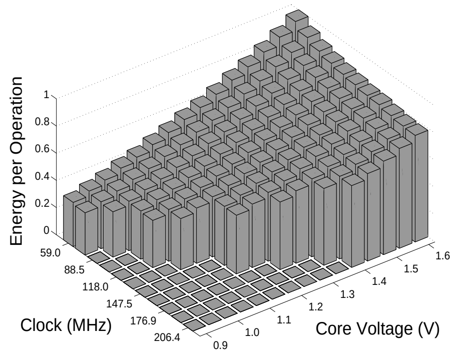

Figure 3: Measured energy consumption characteristics of SA-1100.

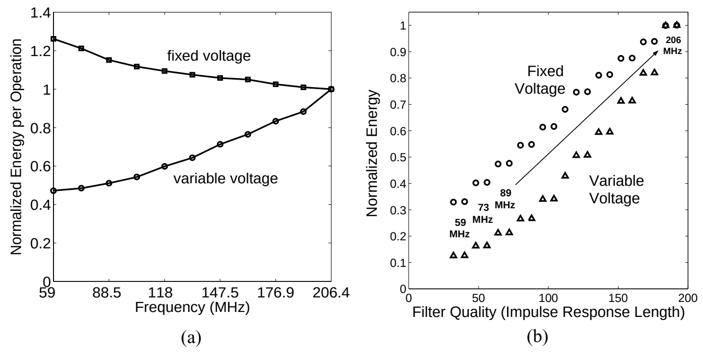

Figure 4: (a) Measured energy savings in an energy *vs.* latency trade-off. (b) Energy *vs.* filter performance for scalable FIR filter.

Figure 3 depicts the measured energy consumption of the SA-1100 processor running at full utilization. The energy consumed per operation is plotted with respect to the processor frequency and voltage. As expected, a reduction in clock frequency allows the processor to run at lower voltage. The quadratic dependence of switching energy on supply voltage is evident, and for a fixed voltage, the leakage energy per operation increases as the operations occur over a longer clock period.

Using a digitally adjustable DC-DC converter, the SA-1100 in the µAMPS sensor node can adjust its own core voltage to demonstrate energy-quality tradeoffs with DVS. In Figure 4a, for a fixed computational workload, the latency (an inverse of quality) of the computation increases as the energy decreases. In Figure 4b, the quality of a FIR filtering algorithm is varied by scaling the number of filter taps. As we sacrifice filter quality, the processor can run at a lower clock speed and thus operate at a lower voltage. In each example, our DVS-based implementation of energy-quality tradeoffs consumes up to 60% less energy than a fixed-voltage processor.

B. Radio Communication Hardware

The characteristics of sensor networks call for interesting considerations in communication models that differ from multimedia networks. The average energy consumption for a sensor radio (Figure 5) when sending a burst packet is given by the following equation:

$$
\begin{aligned}
E &= P_{tx} \cdot (T_{on-tx} + T_{startup-tx}) + P_{out} \cdot T_{on-tx} \\
&\quad + d \cdot P_{rx} \cdot (T_{on-rx} + T_{startup-rx}) \qquad (1)
\end{aligned}
$$

$P_{tx/rx}$ is the power consumption of the transceiver, $T_{on\text{-}tx/rx}$ is the transmit/receive on-time (actual data transmission/reception time), $T_{startup\text{-}tx/rx}$ is the start-up time of the transceiver, P_{out} is the output transmit power which drives the antenna and d is the duty cycle of the receiver. Although the primary purpose of the sensor node is to transmit data, a receiver is also necessary to support a communication protocol in the network (i.e., time synchronization, acknowledgment signal, etc.).

It is important to note that the power consumption of the transceiver ($P_{tx/rx}$) does not vary with the data rate to first order. For short-range transmission (e.g., under 10 meters) at gigahertz carrier frequencies, the radio's power is dominated by the frequency synthesizer which generates the carrier frequency rather than the actual transmit power. Hence, data rate, to first order, does not affect the power consumption of the transceiver [15]. But as packets become shorter, the radio's start-up time becomes significant. To reduce energy, the node's radio module is duty cycled, or turned on/off during the active/idle periods. Figure 6 illustrates the effect of start-up time on transmitter energy consumption when sending a 100 bit packet at 1 Mbps. As the start-up time increases, the radio energy becomes dominated by the start-up transient rather than the active transmit time. Unfortunately, transceivers today require initial start-up times on the order of milliseconds due to an inherent feedback loop in the PLL-based frequency synthesizer. The start-up time must be lowered to a few tens of microseconds to minimize energy consumption for the short packets expected in microsensor communication.

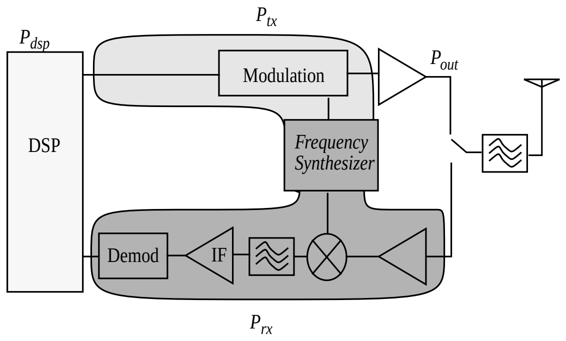

Figure 5: Radio Architecture.

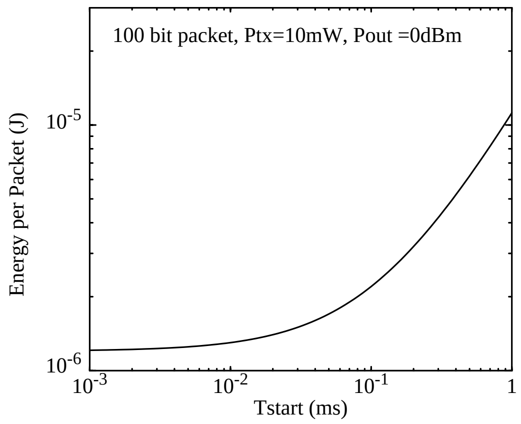

Figure 6: Effects of startup time on short packet transmission.

III. Energy-Efficient Networks

Once the power-aware microsensor nodes are incorporated into the framework of a larger network, additional power-aware methodologies emerge at the network level. Decisions about local computation versus radio communication, the partitioning of computation across nodes, and error correction on the link layer offer a diversity of operational points for the network.

A. Signal Processing in the Network

A network protocol layer for wireless sensors allows for sensor collaboration. Sensor collaboration is important for two reasons. First, data collected from multiple sensors can offer valuable inferences about the environment. For example, large sensor arrays have been used for target detection, classification and tracking. Second, sensor collaboration can provide tradeoffs in communication versus computation energy. Since it is likely that the data acquired from one sensor are highly correlated with data from its neighbors, data aggregation can reduce the redundant information transmitted in the network. Figure 7 shows the amount of energy required to aggregate data from 2, 3 and 4 sensors and to transmit the result to the basestation, compared to all sensors' transmitting data to the basestation individually. When the distance to the basestation is large, there is a large advantage to using local data aggregation (e.g. beamforming) rather than direct communication. Since wireless sensors are energy-constrained, it is important to exploit such trade-offs to increase system lifetimes and improve energy efficiency.

The energy-efficient network protocol LEACH (Low Energy Adaptive Clustering Hierarchy) utilizes clustering techniques that greatly reduce the energy dissipated by a sensor system [12]. In LEACH, sensor nodes are organized into local clusters. Within the cluster is a rotating *cluster-head*. The cluster-head receives data from all other sensors in the cluster, performs data aggregation, and transmits the aggregate data to the end-user. This greatly reduces the amount of data that is sent to the end-user for increased energy-efficiency. LEACH can achieve up to

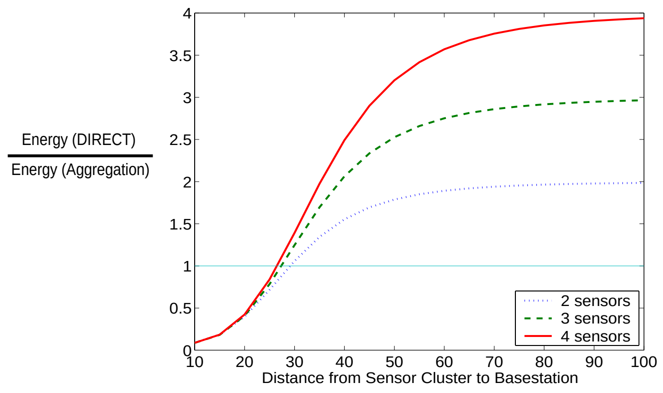

Figure 7: Local data aggregation can reduce energy dissipation.

a factor of eight reduction in energy over conventional routing protocols such as multi-hop routing. However, the effectiveness of a clustering network protocol is highly dependent on the performance of the algorithms used for data aggregation and communication. It is important to design and implement energy-efficient sensor algorithms for data aggregation and link-level protocols for the wireless sensors.

Beamforming algorithms are one class of algorithms which can be used to combine data. Beamforming can enhance the source signal and remove uncorrelated noise or interference. Since many types of beamforming algorithms exist, it is important to make a careful selection based upon their computation energy and beamforming quality. Comparing the Max Power beamforming algorithm and the LMS beamforming algorithm, for instance, measurements on the SA-1100 indicate that the Max Power algorithm requires more than 5 times the energy of the LMS algorithm [13].

B. System Partitioning

Algorithm implementations for a sensor network can take advantage of the network's inherent capability for parallel processing to further reduce energy. Partitioning a computation among multiple sensor nodes and performing the computation in parallel permits a greater allowable latency per computation, allowing energy savings through frequency and voltage scaling.

As an example, consider a target tracking application that requires sensor data to be transformed into the frequency domain through 1024-point FFTs. The FFT results are phase-shifted and summed in a frequency-domain beamformer to calculate signal energies in 12 uniform directions, and the line-of-bearing (LOB) is estimated as the direction with the most signal energy. By intersecting multiple LOB's at the basestation, the source's location can be determined. Figure 8a demonstrates the tracking application performed with traditional clustering techniques for a 7 sensor cluster. The sensors (S1-S6) collect data and transmit the data directly to the cluster-head (S7), where the FFT, beamforming and LOB estimation are performed. Measurements on the SA-1100 at an operating voltage of 1.5V and frequency of 206 MHz show that the tracking application dissipates 27.27 mJ of energy.

Distributing the FFT computation among the sensors reduces energy dissipation. In the distributed processing scenario of Figure 8b, the sensors collect data and perform the FFTs before

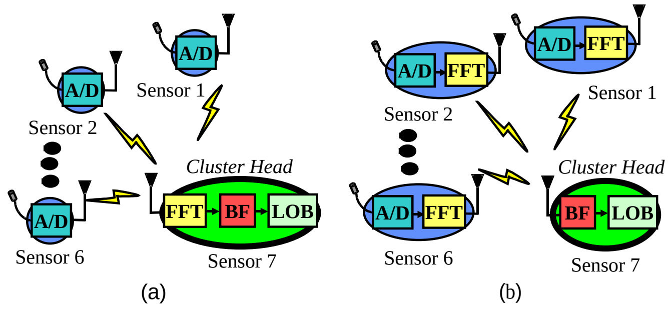

Figure 8: a) Approach 1: All computation is done at the cluster-head. b) Approach 2: Distribute the FFT computation among all sensors.

transmitting the FFT results to the cluster-head. At the cluster-head, the FFT results are beamformed and the LOB estimate is found. Since the 7 FFTs are done in parallel, we can reduce the supply voltage and frequency without sacrificing latency. When the FFTs are performed at 0.9V, and the beamforming and LOB estimation at the cluster-head are performed at 1.3V, then the tracking application dissipates 15.16 mJ, a 44% improvement in energy dissipation.

C. Energy-Efficient Link Layer

Energy-quality tradeoffs appear at the link layer as well. One of the primary functions of the link layer is to ensure that data is transmitted reliably. Thus, the link layer is responsible for some basic form of error detection and correction. Most wireless systems utilize a fixed error correction scheme to minimize errors and may add more error protection than necessary to the transmitted data. In a energy-constrained system, the extra computation becomes an important concern. Thus, by adapting the error correction scheme used at the link layer, energy consumption can be scaled while maintaining the bit error rate (BER) requirements of the user [14].

TABLE I : ENERGY PER USEFUL BIT FOR BCH CODES (@1.5V)

	Encode			Decode		
Code	**Current (mA)**	**Time (s)**	**Energy (nJ/bit)**	**Current (mA)**	**Time (s)**	**Energy (nJ/bit)**
(15,7,2)	240	53	191	240	75	270
(31,6,7)	237	159	562	240	227	817
(31,11,5)	238	111	396	240	182	655
(31,16,3)	238	77	275	240	132	475
(63,7,15)	235	334	117	240	596	2146
(63,16,11)	236	228	807	240	401	1444
(63,24,7)	239	197	706	240	332	1195
(63,39,4)	239	124	445	240	209	752
(63,45,3)	240	94	338	241	193	698

Error control can be provided by various algorithms and techniques, such as convolutional coding, BCH coding, and turbo coding. The encoding and decoding energy consumed by the various algorithms can differ considerably. Table I shows the energy per useful bit to encode and decode messages using various BCH codes on the SA-1100. As the code rate increases, the algorithm's energy also increases. Hence, given bit error rate and latency requirements, the lowest power FEC algorithm that satisfies these needs should continuously be chosen. Power consumption can be further reduced by controlling the transmit power of the physical radio. For a given bit error rate, FEC lowers the transmit power required to send a given message. However, FEC also requires additional processing at the transmitter and receiver, increasing both the latency and processing energy. This is another computation versus communication trade-off that divides available energy between the transmit power and coding processing to best minimize total system power.

IV. POWER-AWARE SOFTWARE

The overall energy efficiency of wireless sensor networks crucially depends on the software that runs on them. Although dedicated circuits can be substantially more energy-efficient, the flexibility offered by general purpose processors and DSPs have engineered a shift towards programmable solutions. Power consumption can be substantially reduced by improving the control software and the application software.

A. Energy Efficient Node Operating Systems

The embedded operating system can dynamically reduce system power consumption by controlling *shutdown*, the powering down all or parts of the node when no interesting events occur, and *dynamic voltage scaling*, which has been discussed above. Dynamic power management using node shutdown, in general, is a non-trivial problem. The sensor node consists of different blocks each characterized by various low power modes and overheads to transition to them. The node sleep states are a combination of various block shutdown modes. If the overheads in transitioning to sleep states were negligible, then a simple greedy algorithm could makes the system go into the deepest sleep state as soon as it is idle. However, in reality, transitioning to a sleep state and waking up has a latency and energy overhead. Therefore, implementing the right policy for transitioning to the available sleep states is critical. Assume that an event is detected by $node_k$ at some time, it finishes processing it at time t_1, and the next event occurs at time $t_2 = t_1 + t_i$. At time t_1, $node_k$ decides to transition to a sleep state s_k from the active state s_0 as shown in Figure 9. Each state s_k has a power consumption P_k, and the transition time to it from the active state and back is given by $\tau_{d,k}$ and $\tau_{u,k}$ respectively. By our definition of node-sleep states, $P_j > P_i$, $\tau_{d,i} > \tau_{d,j}$ and $\tau_{u,i} > \tau_{u,j}$ for any $i > j$.

It has been shown in [15] that there exist thresholds $\{T_{th,k}\}$ corresponding to the states $\{s_k\}$, $0 \le k \le N$ (for N sleep states) such that transitioning to a sleep state s_k from state s_0 will result in a net energy loss if the idle time $t_i < T_{th,k}$ because of the transition energy overhead. This threshold is given by

$$T_{th,k} = \frac{1}{2}\left[\tau_{d,k} + \left(\frac{P_0 + P_k}{P_0 - P_k}\right)\tau_{u,k}\right] \qquad (2)$$

which implies that the longer the delay overhead of the transition s_0 -> s_k, the higher the threshold, and that the greater the difference between P_0 and P_k, the smaller the threshold. These observations are intuitively appealing too. Figure 10 shows the simulation results of the spatial energy consumption of a 1000 node sensor network distributed randomly over a 100mx100m area implementing a hierarchical node shutdown policy based on thresholds and statistical event prediction. From the figure, it can be seen that the shutdown policy allows energy consumption to track event activity. Complete details of the shutdown policy can be found in [15].

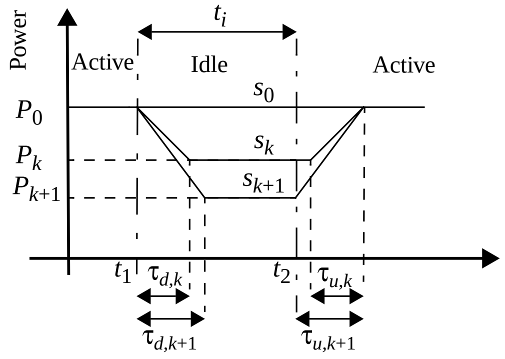

Figure 9: State transition latency and power.

B. Energy Scalable Node Software

It is highly desirable to structure our algorithms and software such that computational accuracy can be traded off with energy consumption. Transforming software such that most significant computations are accomplished first improves the energy-quality scalability can be improved [16]. Consider an example of a sensor node performing an FIR filtering operation. If the energy availability to the node were reduced, we may want to terminate the algorithm early to reduce computational energy. In an unscalable software implementation, this would result in severe quality degradation. Figure 11 demonstrates the improved energy-quality characteristics of an energy-scalable implementation of the FIR filtering operation. By accumulating the partial products corresponding to the most significant coefficients first (by sorting them in decreasing order of magnitude), the scalable algorithm produces far more accurate results at lower energies.

C. Applications Programming Interface (API)

An application programming interface is an abstraction that hides the underlying complexity of the system from the end-user. Hence, a wireless sensor network API is a key enabler in allowing end-users to manage the tremendous operational complexity of such networks. While end-users are experts in their respective application domains (say, remote climate monitoring), they are not necessarily experts in distributed wireless networking and do not wish to be bothered with the internal network operation. By defining high level objects, a functional interface and the associated semantics, APIs make the task of application development significantly easier.

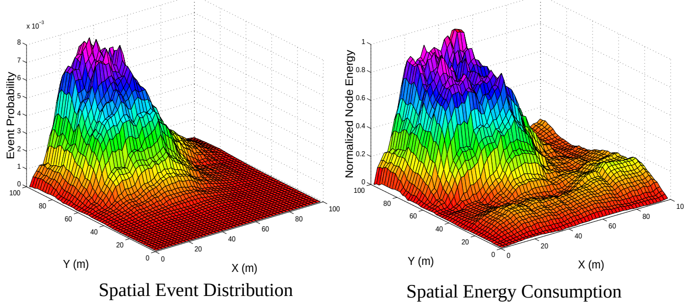

Figure 10: Event driven shutdown of sensor nodes.

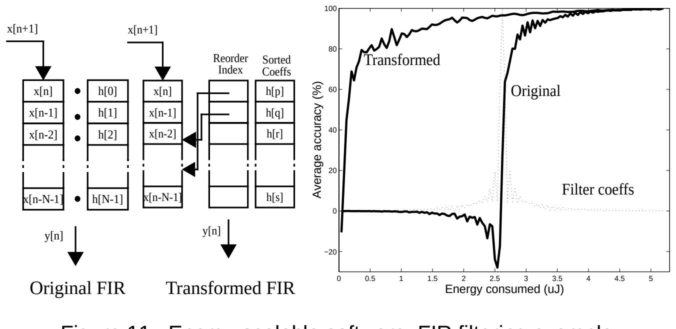

Figure 11: Energy scalable software: FIR filtering example.

An API consists of a functional interface, object abstractions, and detailed behavioral semantics. Together, these elements of an API define the ways in which an application developer can use the system. Key abstractions in a wireless sensor network API are the nodes, basestation, links, messages etc. The functional interface itself is divided into the following:

- Functions that gather the state (of the nodes, part of a network, a link between two nodes etc.)
- Functions that set the state (of the nodes, of a cluster or the behavior of a protocol)
- Functions that allow data exchange between nodes and the basestation
- Functions that capture the desired operating point from the user at the basestation
- Functions that help visualize the current network state
- Functions that allow users to incorporate their own models (for energy, delay etc.)

An API is much more than the sum of its functional interface and object abstractions. This is because of the (often implicit) application development paradigm associated with it. In other words, the API is especially crafted to promote application development based on certain philosophies which the designers of the network consider to be optimal in the sense of correctness, robustness and performance. For example, a good overall application framework for wireless sensor networks is the "Get-Optimize-Set" paradigm. This paradigm basically implies collecting the network state, using this state information along with the knowledge of the desired operating point to compute the new optimal state and then setting the network to this state. The entire application code is based on this template.

Power aware computation and communication is the key to achieving long network lifetimes due to the energy constrained nature of the nodes. An important responsibility of the API is not only to allow the end-user to construct the system in a power aware manner but also encourage such an approach. For starters, functions in a high quality network API have explicit energy, quality, latency and operating point annotations. Hence, instead of demanding a certain function from the network, one can demand a certain function subject to constraints (energy, delay, quality etc.). Next, the API has basic energy modeling allowing the end user to calibrate the energy efficiency of the various parts of the application. For users requiring models beyond the level of sophistication that the API offers, there are modeling interfaces which allow users to register arbitrarily complex models. Next, a good wireless sensor API allows what have come to be known in the software community as "thick" and "thin" clients. These adjectives refer to the complexity and overhead of typical application layers. Finally, the "Get-Optimize-Set" paradigm promulgated by the API allows the network to be at the optimal operating point thus enhancing energy efficiency.

V. Conclusion

Distributed microsensor networks hold great promise in applications ranging from medical monitoring and diagnosis to target detection, home automation, hazard detection, and automotive and industrial control. But even within a single application, the tremendous operational and environmental diversity inherent to the microsensor network demand the system's ability to make trade-offs between quality and energy dissipation. Hooks for energy-quality scalability are necessary not only at the component level, but also throughout the node's algorithms and the network's communication protocols. Distributed sensor networks designed with built-in power awareness and scalable energy consumption will achieve maximal system lifetime in the most challenging and diverse environments.

Acknowledgments

This research is sponsored by the Defense Advanced Research Project Agency (DARPA) Power Aware Computing/Communication Program and Air Force Research Laboratory, Air Force Material Command, USAF, under agreement number F30602-00-2-0551. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon[1].

The authors wish to thank Timothy Garnett for work on initial implementations of the API, and Nathan Ickes for work with the node power supply and hardware architecture.

References

[1] K. Bult et al., "Low Power Systems for Wireless Microsystems," *Proc. ISLPED 1996*, pp. 17-21.

[2] A. Wang, W. Heinzelman, and A. Chandrakasan, "Energy-Scalable Protocols for Battery-Operated Microsensor Networks," *Proc. SiPS '99*, Oct. 1999, pp. 483-492.

[3] A. Chandrakasan et al., "Design Considerations for Distributed Microsensor Systems," *Proc. CICC 1999*, pp. 279-286.

[4] J. M. Kahn, R. H. Katz, and K. S. J. Pister, "Next Century Challenges: Mobile Networking for 'Smart Dust'," *Proc. Mobicom 1999*, pp. 271-278.

[5] G. Asada et al., "Wireless Integrated Network Sensors: Low Power Systems on a Chip," *Proc. ESSCIRC '98*,

[6] J. Rabaey et al., "PicoRadio Supports Ad Hoc Ultra-Low Power Wireless Networking," *Computer*, July 2000, pp. 42-48.

[7] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Program for Sensor Networks," *Proc. Mobicom 2000*.

[8] A. Sinha and A. Chandrakasan, "Energy Aware Software," *Proc. Thirteenth International Conference on VLSI Design*, Jan. 2000, pp. 50-55.

[9] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design, A Systems Perspective*, 2nd edition, Reading, Mass.: Addison-Wesley, 1993, p. 236.

[10] G. Wei and M. Horowitz, "A Low Power Switching Supply for Self-Clocked Systems," *Proc. ISLPED 1996*, pp. 313-317.

[11] J. Goodman, A. Dancy, and A. Chandrakasan, "An Energy/Security Scalable Encryption Processor using an Embedded Variable Voltage DC/DC Converter," *Journal of Solid State Circuits*, Vol. 33, No. 11, Nov. 1998, pp. 1799-1809.

[12] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks", *Proc. HICSS '00*, January 2000.

[13] A. Wang, W. Heinzelman and A. Chandrakasan, "Energy-Scalable Protocols for Battery-Operated MicroSensor Networks", *Proc. SiPS 2000*, October 2000.

[14] P. Lettieri, C. Fragouli, and M.B. Srivastava, "Low Power Error Control for Wireless Links," *Proc. MOBICOM '97*, pages 139-150, August 1997.

[15] A. Sinha and A. Chandrakasan, "Operating System and Algorithmic Techniques for Energy Scalable Wireless Sensor Networks", Proc. of the Second International Conference on Mobile Data Management, Jan. 2001, (to appear)

[16] A. Sinha, A. Wang and A. Chandrakasan, "Algorithmic Transforms for Efficient Energy Scalable Computation", Proc. of the IEEE International Symposium on Low Power Electronics and Design, July 2000

1. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Project Agency (DARPA), the Air Force Research Laboratory, or the U.S. Government.