Physical Layer Driven Protocol and Algorithm Design for Energy-Efficient Wireless Sensor Networks

Eugene Shih, Seong-Hwan Cho, Nathan Ickes, Rex Min,
Amit Sinha, Alice Wang, Anantha Chandrakasan
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA, USA 02139-4307
[eugene, chosta, nickes, rmin, sinha, aliwang, anantha]@mtl.mit.edu

ABSTRACT

The potential for collaborative, robust networks of microsensors has attracted a great deal of research attention. For the most part, this is due to the compelling applications that will be enabled once wireless microsensor networks are in place; location-sensing, environmental sensing, medical monitoring and similar applications are all gaining interest. However, wireless microsensor networks pose numerous design challenges. For applications requiring long-term, robust sensing, such as military reconnaissance, one important challenge is to design sensor networks that have long system lifetimes. This challenge is especially difficult due to the energy-constrained nature of the devices. In order to design networks that have extremely long lifetimes, we propose a physical layer driven approach to designing protocols and algorithms. We first present a hardware model for our wireless sensor node and then introduce the design of physical layer aware protocols, algorithms, and applications that minimize energy consumption of the system. Our approach prescribes methods that can be used at all levels of the hierarchy to take advantage of the underlying hardware. We also show how to reduce energy consumption of non-ideal hardware through physical layer aware algorithms and protocols.

1 INTRODUCTION

In recent years, the idea of wireless microsensor networks has garnered a great deal of attention by researchers, including those in the field of mobile computing and communications [1, 2]. A distributed, ad-hoc wireless microsensor network consists of hundreds to several thousands of small sensor nodes scattered throughout an area of interest. Each individual sensor contains both processing and communication elements and is designed to monitor the environment for events specified by the deployer of the network. Information about the environment is gathered by the sensors and is delivered to a central basestation where the user can extract the desired data. Because of the large number of nodes in such a network, sensors can collaborate to perform high quality sensing and form fault-tolerant sensing systems. With these advantages in mind, many applications have been proposed for distributed, wireless microsensor networks such as warehouse inventory tracking, location-sensing, machine-mounted sensing, patient monitoring, and building climate control [1, 3, 4, 5].

While the applications enabled by wireless microsensor networks are very attractive, in order to build well-functioning, robust systems, there are many system challenges to resolve. Furthermore, because the proposed applications are unique, wireless microsensor systems will have different challenges and design constraints than existing wireless networks (e.g. cellular networks and wireless LANs). For instance, since the number of sensors will be large, node densities will be high (up to 20 nodes/m^3) and large amounts of data will be produced. Thus, large-scale data management techniques will be needed. Secondly, user constraints and environmental conditions, such as ambient noise and event arrival rate, can be time-varying in a wireless microsensor network. Thus, the system should be able to adapt to these varying conditions. In addition to these challenges, the energy consumption of the underlying hardware is also of paramount importance. Because applications involving wireless sensor networks require long system lifetimes and fault-tolerance, energy usage must be carefully monitored. Furthermore, since the networks can be deployed in inaccessible or hostile environments, replacing the batteries that power the individual nodes is undesirable, if not impossible.

This need to minimize energy consumption and to maximize the lifetime of a system makes the design of wireless sensor networks difficult. For example, since packets can be small and data rates low, low-duty cycle radio electronics will be used in the system. However, designing such circuits to be energy-efficient is technically challenging. As we will show, current commercial radio transceivers, such as those proposed for the Bluetooth standard [6], are not ideal for microsensor applications since the energy overhead of turning them on and off is high. Thus, innovative solutions in transceiver and protocol design are required to achieve efficient transmission of short packets over short distances. Another challenge arises due to the remote placement of these nodes and the high cost of communication. Since sensors are remotely deployed, transmitting to a central basestation has high energy cost. Thus, the use of data aggregation schemes to reduce the amount of redundant data in the network will be beneficial [7]. Finally, since environmental conditions and user constraints can be time-varying, the use

To appear in the Seventh Annual ACM SIGMOBILE Conference on Mobile Computing and Networking, July 2001, Rome, Italy.

of *static* algorithms and protocols can result in non-optimal energy consumption. Thus, wireless microsensors must allow adaptation of underlying hardware by higher level algorithms. By giving upper layers the opportunity to adapt the hardware in response to changes in system state, the environment and the user's quality constraints, the energy consumption of the node can be better controlled.

In summary, reducing energy consumption to extend system lifetime is a primary concern in microsensor networks. Thus, protocols and algorithms should be designed with saving energy in mind. Unfortunately, without knowledge of the underlying computation and communication hardware, performing intelligent power management can be difficult. If system designers want to create energy-efficient algorithms and protocols, correct and accurate modeling of the underlying hardware is critical. Otherwise, system designers run the risk of designing algorithms and protocols that do not reduce the overall system energy consumption significantly. If physical parameters are ignored, one could end up with an energy *inefficient* solution that only survives a short time.

In this paper, we introduce the design of physical layer aware protocols, algorithms, and applications that minimize energy consumption of the system and individual nodes[1]. In addition, we show how to use software to deal with non-ideal physical hardware. First, we describe a hardware implementation with parameters that can be adjusted by software. We will introduce algorithms that take advantage of these adjustable hooks and knobs to minimize energy. Data-link and media-access protocols that adapt parameters of the underlying physical layer in order to minimize energy will be presented. At the same time, we will illustrate how protocols designed without knowledge of the hardware can be energy inefficient. Because our aim is to explore how to minimize energy consumption at different levels of the protocol stack, we will not expound on any one of these techniques in too much detail.

2 RELATED WORK

Due to recent advances in integrated circuit and MEMS technology, it is clear that small, low power sensing devices will be ready to be deployed in sensor networks in the near future. Many research groups are exploring the issues related to the design of nodes for deployment in wireless sensor networks. The WINS [8] and PicoRadio [5] projects are seeking ways to integrate sensing, signal processing, and radio elements onto a single integrated circuit. Meanwhile, researchers involved in SmartDust [3] aim to design particle-sized nodes for wide-area distributed sensing.

Network protocols for wireless microsensor networks, such as *directed diffusion* [9] and *LEACH* [10], are also beginning to emerge. In directed diffusion, routes are dynamically formed as data is sensed. Initially, routes called gradients that link *sources* of interesting data to *sinks* are formed. Through data aggregation techniques, caching, and reinforcement messages, the appropriate link is dynamically selected from the candidates. Links are created only when data of interest is sensed. Thus, less energy will be used by this protocol. LEACH is a protocol that uses hierarchy to reduce the data collected by the sensors before sending it on to a central base station. Reducing the data that needs to be sent helps make LEACH more energy efficient.

While research into energy-efficient protocols for sensor networks is relatively new, many energy-efficient network protocols for ad-hoc wireless networks have been presented. In [11, 12, 13], techniques and metrics to evaluate and design energy-efficient routing and MAC protocols for wireless networks are presented. Energy-efficient protocols that adapt transmit output power and/or error correction control parameters have been explored extensively by a number of researchers [14, 15, 16, 17]. In [14], the authors use an adaptive radio designed for wireless multimedia communications over ATM as a model. In that paper, frame length and forward-error correction parameters are adapted to lower energy consumption of the radio and improve throughput as conditions of the channel change. A similar study is performed by [16] in the context of a cellular-style network, but the output transmit power is also considered. In [17], an energy-efficient protocol that adjusts both RF transmit power and error control strategy is examined for 802.11 wireless LANs. The authors of [15] offer an in-depth study of the error process and then introduce a probing ARQ-type scheme that is designed for energy-constrained devices.

In order to understand energy consumption, the underlying hardware devices cannot be ignored. However, in much of the research presented, only the impact of the radio is considered. In those cases where the processing hardware *is* considered, a generic mobile terminal is used as the model. In general, the majority of the studies have concentrated on networks intended for mobile terminals and multimedia applications such as video and audio. In this paper, the effect of the entire hardware architecture on the design of protocols and algorithms is considered.

The authors of [18] do explore the idea of minimizing energy consumption from a system perspective. That is, the energy consumption of all devices in the mobile device are considered. The authors consider both the energy required during active communication and the energy required to run protocols on the device's CPU. By considering the overall communication energy of the system, the authors show that protocols that attempt to minimize the number of messages in ARQ-type protocols do not necessarily minimize overall system power. As such, they present an adaptive strategy to tune global communication parameters and reduce system energy consumption.

While similar to some previous work, the research presented in this paper differs in several respects. First, we describe an actual node that is targeted for use in sensor networks. Given the hardware, we introduce models that expose underlying properties of the hardware. These models allow more energy-efficient protocols and algorithms to be developed and also point to the need for designing better hardware. In addition, in most of the previous studies, the devices considered are far less energy-constrained than the nodes intended for wireless sensor networks. Mobile, wireless hand-held devices can be recharged periodically, while wireless sensor nodes have small fixed energy sources.

3 APPLICATION SCENARIOS

Wireless microsensor networks can be used in a variety of application spaces. Different configurations and network architectures will be appropriate depending on the application. In this paper, the protocol and algorithm design choices discussed are motivated by applications for distributed, ad-hoc collaborative sensing networks.

For the most part, the protocols and algorithms discussed in this paper are designed for one of two application scenarios. The first

[1] By physical layer, we not only mean the radio component, but also the computational electronics.

involves vehicle tracking and detection. In this application, we are interested in detecting when a vehicle is in the region of interest and in its velocity and direction of travel. In this application, hundreds to thousands of energy-constrained nodes are scattered over the terrain. Due to the high density of nodes, *clusters* can be formed that will enable reduced energy consumption. The data collected by the sensors may be processed locally or sent directly to a central basestation. Communication in this system may occur both among and between sensors. Typically, inter-node data rates are quite low (≤ 1 Mbps) and packets sizes are relatively small (< 5 kbits). Furthermore, transmission distances between nodes are fairly short (≤ 10 m). On the other hand, the distance between a node and the basestation can be as much as several kilometers. Since communication costs over such large distances is energy-draining, communication to the basestation should be avoided unless absolutely necessary.

The second application we will discuss concerns factory machine sensing. The details of this application scenario will be provided in Section 7.2. Before discussing how to leverage the physical layer to produce energy-efficient protocols and algorithms at various layers of the protocol stack, we will present the architecture and implementation of a node intended for use in collaborative sensing applications.

4 THE μAMPS WIRELESS SENSOR NODE

The μAMPS (micro-Adaptive Multi-domain Power-aware Sensors) node is a wireless sensor node that exposes the underlying parameters of the physical hardware to the system designer. Our node has the ability to scale the energy consumption of the entire system in response to changes in the environment, state of the network, *and* protocol and application parameters in order to maximize system lifetime and reduce global energy consumption. Thus, all layers of the system, including the algorithms, operating system, and network protocols, can adapt to minimize energy usage. Figure 1 gives an overview of the architecture of the sensor node. The overall node can be broken down into different 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 user. As a result, the energy consumption of every component in the system can be exploited at the software level to extend system lifetime and meet user constraints. Figure 2 shows the node implemented in actual hardware.

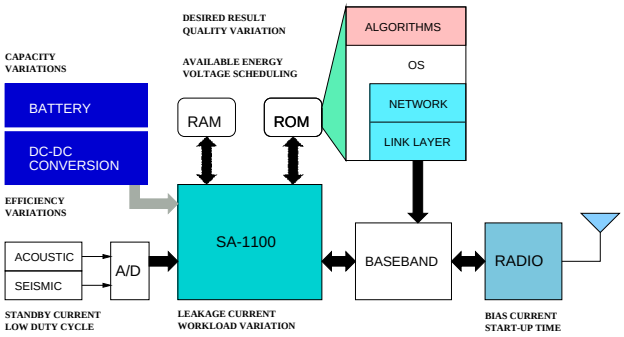


Figure 1: Architectural overview of our microsensor node. The node allows algorithms to gracefully scale its energy consumption by modifying underlying hardware parameters.

Whether for equipment monitoring, military surveillance, or medical sensing, information about the environment must be gathered using some *sensing subsystem* consisting of a sensor connected to an analog-to-digital (A/D) converter. Our initial node contains an electret microphone for acoustic sensing. However, a wider variety of sensors is supported. The acoustic sensor is connected to a 12-bit A/D converter capable of converting data at a rate of 125 kilosamples per second (kSPS). In the vehicle tracking application, the required conversion rate is about 1 kSPS. An envelope detector is also included to allow ultra-low lower sensing.

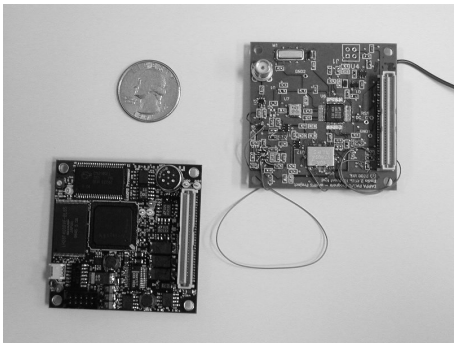


Figure 2: Hardware implementation of the node compared to a U.S. quarter. The lower left board contains the sensor and processor, while the board on the upper right contains the radio transceiver.

Once enough data is collected, the *processing subsystem* of the node can digitally process the data or it can relay the data to a nearby node (or faraway basestation). The primary component of the data and control processing subsystem is the StrongARM SA-1110 microprocessor. Selected for its low power consumption, performance, and static CMOS design, the SA-1110 runs at a clock speed of 59 MHz to 206 MHz. The processing subsystem also includes RAM and flash ROM for data and program storage. A multithreaded "μ-OS" running on the SA-1110 has been customized to allow software to scale the energy consumption of the processor. Code for the algorithms and protocols are stored in ROM.

In order to deliver data or control messages to neighboring nodes, the data from the StrongARM is passed to the *radio subsystem* of the node via a 16-bit memory interface. Additional protocol processing and data recovery is performed by a Xilinx FPGA. The primary component of the radio is a Bluetooth-compatible commercial single-chip 2.4 GHz transceiver [19] with an integrated frequency synthesizer. The on-board phase-locked loop (PLL), transmitter chain, and receiver chain can be shut-off via software or hardware control for energy savings. To transmit data, an external voltage-controlled oscillator (VCO) is directly modulated, providing simplicity at the circuit level and reduced power consumption at the expense of limits on the amount of data that can be transmitted continuously. The radio module, with two different power amplifiers, is capable of transmitting at 1 Mbps at a range of up to 100 m.

Finally, power for the node is provided by the *battery subsystem* via a single 3.6 V DC source with an energy capacity of approximately 1500 mAH. Switching regulators generate 3.3 V, and adjustable 0.9-2.0 V supplies from the battery. The 3.3 V supply powers all digital components on the sensor node with the exception of

the processor core. The core is specially powered by a digitally adjustable switching regulator that can provide 0.9 V to 2.0 V in thirty discrete increments. The digitally adjustable voltage allows the SA-1110 to control its own core voltage, enabling the use of a well-known technique known as *dynamic voltage scaling* [20, 21].

5 HARDWARE MODELS

Using the node, an energy model of the device was developed. Since the radio and data processing subsystems have the greatest impact on the energy consumption of the node, we will focus on those two components. Note that while our node uses an SA-1110, many of our measurements were obtained using the SA-1100, which is very similar in performance and energy consumption.

5.1 Processor Energy Model

The energy consumption of static CMOS-based processors designed in the past couple of decades has been primarily due to switching energy. The switching energy is expressed as

$$E_{switch} = C_{total}{V_{dd}}^2$$

where C_{total} is the total capacitance switched by the computation and V_{dd} is the supply voltage.

As circuit designers become more concerned with reducing power consumption, switching energy will become less dominant. Supply voltages (V_{dd}) are constantly being lowered as an effective way to reduce power consumption. At the same time, to satisfy ever demanding performance requirements, the threshold voltage (V_{th}) is also scaled proportionately to provide sufficient current drive and reduce the propagation delay of signals in an integrated circuit. As the threshold voltage is lowered, the sub-threshold leakage current becomes increasingly dominant. Moreover, when scaling of device thresholds for low-voltage operation is coupled with the low duty cycle operation of a sensor, the dominance becomes more apparent.

To summarize, processor leakage energy is an important parameter to model when designing a wireless microsensor network because energy is wasted while no work is done. The energy lost due to leakage current can be modeled with an exponential relation to the supply voltage [22]:

$$E_{leakage,\mu P} = (V_{dd}t)I_0 e^{\frac{V_{dd}}{n'V_T}}$$

where V_{th} is the device threshold voltage and V_T is the thermal voltage. Experimentally, we have determined that for the StrongARM SA-1100, $n' = 21.26$ and $I_0 = 1.196$ mA and that the leakage energy accounts for about 10% of the total energy dissipated. Since the leakage energy of future processors will be responsible for more than 50% of the total power consumption [23], techniques to reduce the energy consumption penalty of low-duty cycle operations must be devised.

Software-based techniques such as dynamic voltage scaling and the progressive shutdown of idle components are some high-level leakage control techniques. These methods will be discussed in greater detail when we present some physical layer aware, low-power protocols and algorithms.

5.2 Radio Model

The average power consumption of the radio in our design can be described by:

$$\begin{aligned} P_{radio} = N_{tx}[P_{tx}(T_{on-tx} + T_{st}) + P_{out}T_{on-tx}] \\ + N_{rx}[P_{rx}(T_{on-rx} + T_{st})] \end{aligned} \quad (1)$$

where $N_{tx/rx}$ is the average number of times per second that the transmitter/receiver is used, $P_{tx/rx}$ is the power consumption of the transmitter/receiver, P_{out} is the output transmit power, $T_{on-tx/rx}$ is the transmit/receive on-time (actual data transmission/reception time), and T_{st} is the startup time of the transceiver as shown in Figure 3. Note that $N_{tx/rx}$ will largely depend on the application scenario and the media-access control (MAC) protocol being used. Also note that $T_{on-tx/rx} = L/R$, where L is the packet size in bits and R is the data rate in bits per second. In this radio model, the power amplifier is on only when communication occurs. During the

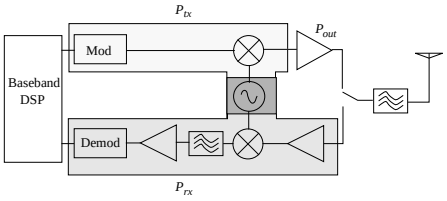


Figure 3: A diagram of the radio model.

startup time, no data can be sent or received by the transceiver. This is because the internal phase-locked loop (PLL) of the transceiver must be locked to the desired carrier frequency before data can be demodulated successfully. Figure 4 shows the measured startup transient of the low power transceiver in our node as described in Section 4. The control input to the voltage-controlled oscillator (in volts) is plotted versus time.

It is necessary to highlight a few key points about the radio we use in our design. First, the transceiver power does not vary over the data rate, R. At the 2.4 GHz frequency band, the power consumption of the transceiver is dominated by the frequency synthesizer which generates the carrier frequency. Hence, to a first order, R does not affect the power consumption of the transceiver [24]. Second, the startup time can have a large impact on the average energy per bit, E_b, since wireless sensor networks tend to communicate using short packets. In order to save power, a natural idea is to turn off the radio during idle periods. Unfortunately, when the radio is needed again, a large amount of power is spent to turn it back on; transceivers today require an initial startup time on the order of hundreds of microseconds during which large amounts of power is wasted. Given that $P_{tx} = 81$ mW and $P_{out} \approx 0$ dBm, the effect of the startup transient is shown in Figure 5, where the energy per bit is plotted versus the packet size. We see that as packet size is reduced, the energy consumption is dominated by the startup transient and not by the active transmit time. Hence it is important to take this inefficiency into account when designing energy-efficient communication protocols. The radio parameters used here are based on a state-of-the-art commercial low power transceiver available today [19]. Note that P_{rx} is usually 2 to 3 times higher than P_{tx} since more circuitry is required to receive a signal. In our radio, $P_{rx} = 180$ mW.

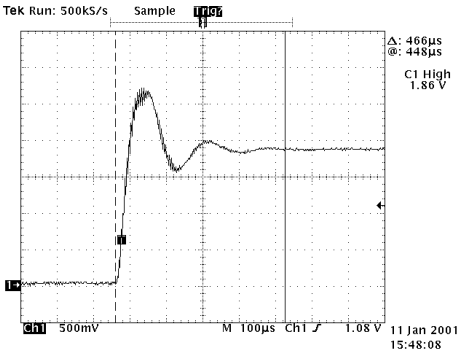


Figure 4: Measured startup transient ($T_{st} \approx 470\mu$s) of a commercial low power transceiver. The control input to the VCO (in volts) is plotted versus time. When the PLL is not on, the control input to the VCO is low. Once the PLL is turned on, it takes T_{st} for the control input to settle to the right voltage.

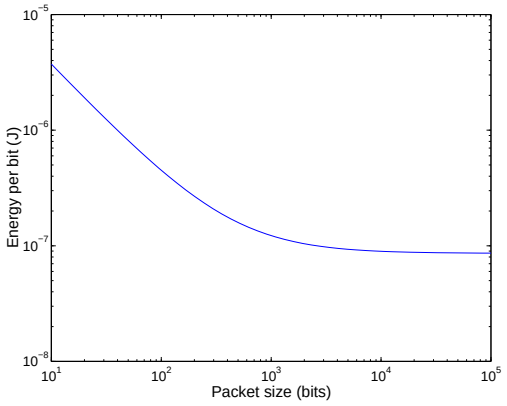


Figure 5: Effect of startup transient where $R = 1$ Mbps, $T_{st} \approx 450\mu$s, $P_{tx} = 81$ mW, and $P_{out} = 0$ dBm.

6 PHYSICAL LAYER IMPACT ON ALGORITHM DESIGN

The underlying physical layer can be exploited by both system-level and node-level algorithms to achieve energy-efficient solutions in wireless sensor networks. In this section, we will present energy-efficient techniques at both levels that take advantage of hardware hooks.

6.1 System-Level Computation Partitioning

One way to improve energy efficiency is to design algorithms and protocols that take advantage of the dense localization of nodes in the network. For example, in acoustic sensing applications, closely located sensors will have highly correlated data. Thus, to reduce redundant information in the network, sensors are grouped in clusters and signal processing is done locally within a cluster. Through signal processing, nodes can remove redundant information, thereby reducing communication costs. Another straightforward way to reduce energy consumption is to adapt the underlying hardware parameters of the node. In this section, we will present an algorithm that can be performed in a distributed fashion to find the location of the vehicle. By distributing the computation, we show that the energy efficiency of the algorithm can be vastly improved.

Suppose a vehicle is moving over a region where a network of acoustic sensing nodes has been deployed. In order to determine the location of the vehicle, we first find the line of bearing (LOB) to the vehicle. In this scenario, we assume that nodes are clustered and that multiple clusters autonomously determine the source's LOB from their perspective. Individual sensors will send data to a "clusterhead". The intersection point of multiple LOBs will determine the source's location and can be calculated at the basestation. In this application, we will assume that due to user requirements, the latency constraint on the computation is 20 ms. Figure 6 shows the scenario described.

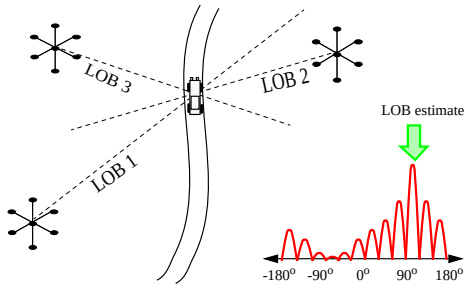


Figure 6: Vehicle tracking application scenario. The basestation is not shown. In the figure on the bottom, the signal energy for twelve different directions is shown. The LOB estimate is the direction which has the most signal energy.

One approach to locate the source is to first estimate the LOB at the cluster and then transmit the result to the basestation. Alternatively, all the sensors could transmit their raw, collected data directly to the basestation for processing. Figure 7 shows the energy required for the first approach compared to the energy required for the second approach. As the distance from the sensor to the basestation increases, it is more energy-efficient to do signal processing locally, at the sensor cluster.

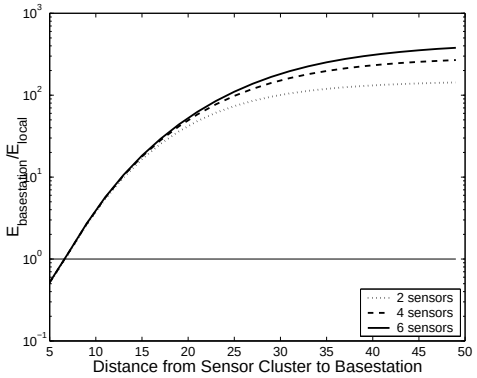


Figure 7: If LOB estimation is done in the network, energy dissipation is reduced. As the basestation is located farther away, it is more energy-efficient to do signal processing at the clusterhead.

To perform LOB estimation, one can use frequency-domain delay-

and-sum *beamforming* [25]. Beamforming is the act of summing the outputs of filtered sensor inputs. In a simple delay-and-sum beamformer, the filtering operations are delays or phase shifts. The first part of frequency-domain beamforming is to transform collected data from each sensor into the frequency domain using a 1024-pt Fast-Fourier Transform (FFT). Then, we beamform the FFT data in twelve uniform directions to produce twelve candidate signals. The direction of the signal with the most energy is the LOB of the source. Given LOB's from multiple clusters, we can then calculate the location of the source. The source localization algorithm is summarized below:

Source Localization Using LOB Estimation
Input: Let $s_i(n)$ be the set of data for the ith sensor.
Output: Source location (LOB).

1. Let $S_i(f) \leftarrow FFT(s_i(n))$ for $1 \leq i \leq m$.
2. Set $j \leftarrow 1$. Repeat the following 12 times:
   2.1 (Beamforming) Using m filters $W_i^{(j)}(f)$, apply the ith filter to $S_i(f)$. $W_i^{(j)}(f)$ represents the set of m filters to use for the jth trial.
   2.2 Let $y_j = \sum_{i=1}^{m} W_i^{(j)}(f) S_i(f)$. Increment j.
3. $LOB \leftarrow \mathsf{dir}(\max \|y_1\|^2, \|y_2\|^2, \ldots, \|y_{12}\|^2)$ where $\|y_j\|^2$ is the signal energy of y_j.

The source localization algorithm can be implemented in two different ways. Assume each sensor i has a set of acoustic data $s_i(n)$. This data can be sent first to a local aggregator or "clusterhead" where the FFTs and beamforming are performed. We will call this technique the *direct* technique, which is demonstrated in Figure 8(a). Alternatively, each sensor can transform the data locally before sending the data to the clusterhead. This method will be called the *distributed* technique and is illustrated in Figure 8(b). If we assume the radio and processor models discussed in Section 5, then performing the FFTs locally, while distributing the computational load and reducing latency, has no energy advantage. This is because performing the FFTs locally does not reduce the amount of data that needs to be transmitted. Thus, communication costs remain the same. However, because our node supports dynamic voltage scaling, we can take advantage of the parallelized computational load by allowing voltage and frequency to be scaled while still meeting latency constraints.

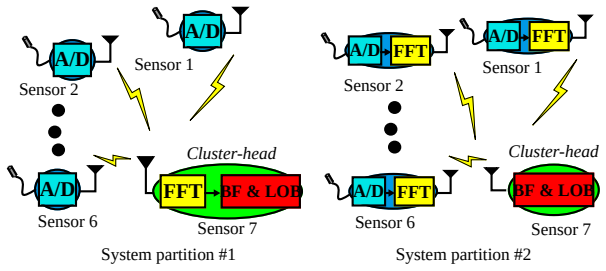


Figure 8: (a) System partition 1: All of the computation is done at the clusterhead ($E_{total} = 6.2$ mJ, Latency = 19.2 ms). (b) System partition 2: Computation of FFT is distributed among sensors in a cluster ($E_{total} = 3.4$ mJ, Latency = 18.4 ms).

Dynamic voltage scaling (DVS) exploits variabilities in processor workload and latency constraints and realizes this energy-quality trade-off at the circuit level. As shown previously, the switching energy E_{switch} of any particular computation is independent of time. In addition, 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 reduced, 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 9 depicts the measured energy consumption of the SA-1100 processor running at full utilization [26]. The energy consumed per instruction is plotted with respect to the processor frequency and voltage. As expected, a reduction in clock frequency allows the processor to run at a 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. This result has been applied by several researchers for CPU scheduling and algorithm design [27, 28]. However, we show below that DVS can also be used to allow system partitioning of computation.

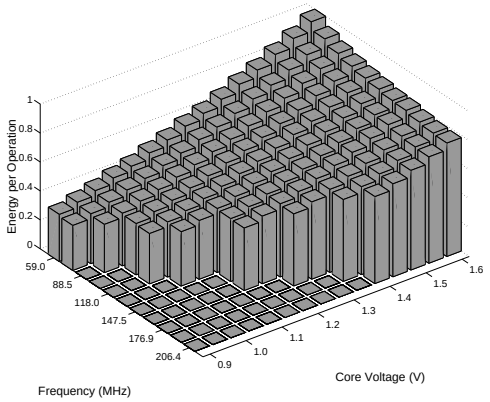


Figure 9: Measured energy consumption characteristics of SA-1100.

In a DVS-enabled system, there is an advantage to doing distributed signal processing. By distributing computation, the clock rate can be reduced at each sensor, allowing for a reduction of the supply voltage. In System Partition 1, the direct technique, all sensors sense data and transmit their raw data to the clusterhead, where the FFTs and beamforming are done. The clusterhead performs the beamforming and LOB estimation *before* transmitting the result back to the user. In order to be within the user's latency requirement of 20 ms, all of the computation is done at the clusterhead at the fastest clock speed, $f = 206$ MHz at 1.44 V. The energy dissipated by the computation is 6.2 mJ and the latency is 19.2 ms.

In System Partition 2, the distributed technique, the FFT task is parallelized. In this scheme, the sensor nodes perform the 1024-pt FFTs on the data before transmitting the data to the clusterhead. The clusterhead performs the beamforming and LOB estimation. Since the FFTs are parallelized, the clock speed and voltage supply of both the FFTs and the beamforming can be lowered. For example, if the FFTs at the sensor nodes are run at 0.85 V at 74 MHz, while the beamforming algorithm is run at 1.17 V at 162 MHz,

then with a latency of 18.4 ms, only 3.4 mJ is dissipated. This is a 45.2% improvement in energy dissipation. This example shows that efficient system partitioning by parallelism can yield large energy reductions.

Figure 10 compares the energy dissipated for System Partition 1 versus that for System Partition 2 with optimal voltage scheduling as the number of sensors is increased from 3 to 10 sensors. This plot shows that a 30-65% energy reduction can be achieved with the system partitioning scheme. Therefore, the use of DVS coupled with system partitioning of computation should be considered by protocol designers when designing an algorithm for sensor networks.

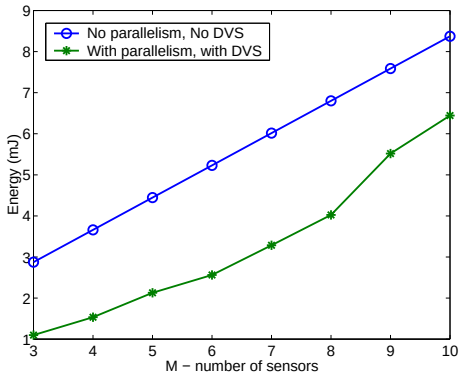


Figure 10: Comparing computation energy used for System Partition 1 vs. System Partition 2.

6.2 Node-Level Power-Mode Scheduling

Because not all algorithms can be partitioned over the entire system, node-level techniques to extend lifetime are also needed. In this section, we present a *power-mode scheduling algorithm* intended for use on nodes in a sensor network. In general, a power-mode scheduling algorithm is one that manages the active and sleep modes of the underlying device in order to increase node lifetime. The advantage of power-mode scheduling is that wasted energy due to leakage can be reduced. If a device is completely turned off, no leakage energy is dissipated.

Many techniques that attempt to minimize energy consumption of devices and CPUs through power management and scheduling have been considered [29, 30, 31]. Algorithms to schedule processes on CPUs in order to minimize power consumption are the focus of [29] and [30]. In [31], the authors use a semi-Markov decision process model to devise a shutdown policy for various portable devices. A shutdown policy that adapts to event arrival statistics is presented here.

6.2.1 Power-Modes for the Sensor Node

In Section 5, we have introduced energy models for the radio and processor of our node. Observe that there are several modes of operation for each. For example, the SA-1100 can operate in several different energy modes: active, idle, or sleep mode. Using these different modes, we can define different sleep states or operating modes for the node. Each operating mode corresponds to a particular combination of component power modes. In general, if there are n components labelled $(0, 1, \ldots, n-1)$, each with k_i number of sleep states, the total number of node-sleep states is $\prod_{i=0}^{n-1} k_i$. Every component power mode is associated with a latency overhead for transitioning to that mode. Therefore, each mode is characterized by a power consumption and a latency overhead. However, from a practical point of view, not all the states are useful. Table 1 enumerates the component power modes corresponding to five different useful energy modes for the sensor node. Each of these

Table 1: Useful sleep states for the sensor node.

State	SA-1110	Sensor, A/D	Radio
Active (s_0)	active	sense	tx/rx
Ready (s_1)	idle	sense	rx
Monitor (s_2)	sleep	sense	rx
Observe (s_3)	sleep	sense	off
Deep Sleep (s_4)	sleep	off	off

node-sleep modes corresponds to an increasingly deeper sleep state and is therefore characterized by an increasing latency and decreasing power consumption. These sleep states are chosen based on actual working conditions of the sensor node. We assume that s_0 is the only active state and that processing can only occur when the SA-1110 is active. The power-aware sensor modes presented here is similar to the system power model in the ACPI standard [32]. An ACPI compliant system has five global stetes. `SystemStateS0` is the working state, while `SystemStateS1` to `SystemStateS4` correspond to four different levels of sleep states. The sleep states are differentiated by the power consumed, the overhead required in going to sleep, and the wakeup time. In general, the "deeper" the sleep state, the lower the power consumption and the longer the wakeup time. With these sleep states, a system designer can devise a policy of transitioning between states based on observed events to maximize energy efficiency. The statistical nature of the events may dictate the nature of the transition policy. An example of such a policy is discussed in the next section.

6.2.2 Sleep State Transition Policy

The presence of a hierarchy of sleep states has to be exploited intelligently for optimum energy savings in the network. The various shutdown stages are characterized by progressively lower power consumptions and increasing transition overhead in terms of time. If these overheads were ignored, a simple greedy algorithm that puts the node into the deepest sleep state whenever possible would work. However, in practice using such a scheme could result in cases where more energy is consumed compared to when the node was not put to sleep. An example of this scenario follows.

Assume an event is detected by node k at some time and it finishes processing the event at t_1 and the next event occurs at time $t_2 = t_1 + t_i$. At t_1, node k decides to transition to a sleep state s_k from the Active state as shown in Figure 11. Each state s_k has a power consumption P_k. Define the transition time to s_k from the active state by $\tau_{d,k}$ and the transition time from the active state to s_k by $\tau_{u,k}$. 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 $i > j$.

We will now derive a set of sleep time thresholds $\{T_{th,k}\}$ corresponding to the states $\{s_k\}$ (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 $t_i < T_{th,k}$ due to the transition energy over-

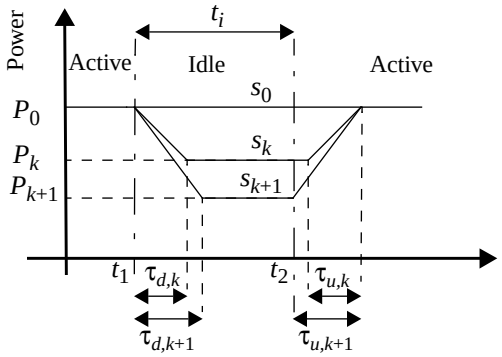


Figure 11: State transition latency and power.

head. No productive work can be done in the transition period since the transition time will consist of the time needed to lock the PLL, the time to stabilize the clock, and the time required to restore the processor context. The energy saving E_s due to transitioning is given by

$$E_{s,k} = P_0 t_i - \left[\frac{P_0 + P_k}{2}\right](\tau_{d,k} + \tau_{u,k}) - P_k(t_i - \tau_{d,k})$$
$$= (P_0 - P_k)t_i - \left[\frac{P_0 - P_k}{2}\right]\tau_{d,k} - \left[\frac{P_0 + P_k}{2}\right]\tau_{u,k}$$

and such a transition is only justified when $E_{s,k} > 0$. This leads us to the transition time threshold

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

which implies that the longer the delay overhead of the transition $s_0 \rightarrow s_k$, the longer the transition time threshold, while the greater the difference between P_0 and P_k, the shorter the threshold. Using Poisson theory, it can be shown that the probability that at least one event occurs in time T_{th} at node k is given by $p_{th,k}(T_{th}) = 1 - e^{-\lambda_k T_{th}}$ where λ_k is the event arrival rate at node k. Table 2 lists the power consumption of the sensor node

Table 2: Sleep state power, latency and thresholds.

State	$\mathbf{P_k}$ (mW)	$\tau_\mathbf{k}$ (ms)	$\mathbf{T_{th,k}}$ (ms)
Active	1040	-	-
Ready	400	5	8
Monitor	270	15	20
Look	200	20	25
Sleep	10	50	50

described in Section 6.2.1. Since the node consists of off-the-shelf components, it is not optimized for minimum power consumption. However, we will use the threshold and power consumption numbers detailed in Table 2 to illustrate our basic shutdown algorithm. When an event is detected at node k, it wakes up and processes the event (this might involve classification, beamforming, transmission etc.). It then updates a global counter which stores the total number of events registered by node k. The average arrival rate, λ_k, for node k is then updated. This requires the use of an μ-OS timer-based system function call which returns the time elapsed since the node was turned on. The μ-OS then tries to put the node into sleep state s_k (starting from the deepest state s_4 through s_1) by testing the probability of an event occurring in the corresponding sleep time threshold $T_{th,k}$ against a system defined constant p_{th0}.

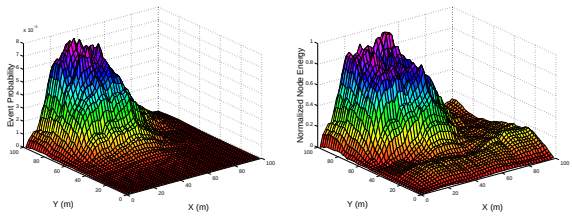
All the sleep states, except the Deep Sleep state have the actual sensor and A/D circuit on. When an event is detected, that is, the signal power is above a threshold level, the node transitions to the Active state and processes the event. The only overhead involved is latency (worst case being about 25 ms). However, in the Deep Sleep state, the node is almost completely off and it must rely on a preset timer to wake up. In sparse event sensing systems, (e.g. vehicle tracking, seismic detection etc.) the inter-arrival time for events is much greater than the sleep time thresholds $T_{th,k}$. Therefore, the sensor node will frequently go into the Deep Sleep state. In this state, the processor must watch for preprogrammed wake-up signals that are set by the CPU prior to entering the sleep state. To be able to wake up on its own, the node must be able to predict the arrival of the next event. An optimistic prediction might result in the node waking up unnecessarily while a pessimistic strategy will result in some events being missed. In our context, being in the Deep Sleep results in missed events as the node has no way of knowing if anything significant occurred. In general, the strategy selected is a design decision based on how critical the sensing task is. There are two possible strategies:

- *Completely disallow* Deep Sleep. If the sensing task is critical and any event cannot be missed this state must be disabled.
- *Selectively disallow* Deep Sleep. This technique can be used if events are spatially distributed and not totally critical. Both random and deterministic approaches can be used. In our scheme, every node k that satisfies the sleep threshold condition for Deep Sleep goes to sleep with a system defined probability p_{s4} for a time duration given by

$$t_{s4,k} = -\frac{1}{\lambda_k}\ln(p_{s4}). \quad (2)$$

The steady state behavior of the nodes is described by (2). Thus, the sleep time is computed such that the probability that no events occur in $t_{s4,k}$ is equal to p_{s4}. However, when the sensor network is switched on and no events have occurred for a while, λ_k will become zero. To account for this case, we disallow the transition to state s_4 until at least one event is detected. We can also have an adaptive transition probability p_{s4}, which is zero initially and increases as events are later detected. The advantage of this algorithm is that efficient energy tradeoffs can be made with only a fraction of events missed. By increasing p_{s4}, the system energy consumption can be reduced while the probability of missed events will increase and vice versa. Therefore, our overall shutdown policy is governed by two implementation specific probability parameters, p_{th0} and p_{s4}.

To evaluate the performance of our probabilistic scheme, we used a custom-made Java-based event-driven simulator to simulate a 1000 node system distributed uniformly at random over a 100×100 m area. The visibility radius of each sensor was assumed to be $r = 10$ m. The sleep state thresholds and power consumption are shown in Table 2. Figure 12 shows the overall spatial node energy consumption for an event with Gaussian spatial distribution centered around $(25, 75)$. It can be seen that the node energy consumption tracks the event probability. In a non-power managed scenario we would have uniform energy consumption in all nodes.



(a) (b)

Figure 12: (a) Spatial distribution of events (b) Spatial energy consumption of nodes.

7 IMPACT OF HARDWARE ON PROTOCOL DESIGN

In this section, we will explore the impact of the hardware on the design of the link, MAC, and physical layers of the protocol stack.

7.1 Link Layer Considerations

In any protocol stack, the link layer has a variety of purposes. One of the tasks of the link layer is to specify the encodings and length limits on packets such that messages can be sent and received by the underlying physical layer. The link layer is also responsible for ensuring reliable data transfer. In the following section, the impact of varying error control on the energy consumption of our node will be discussed. In [14], a similar exploration of the impact of adapting packet size and error control on system energy efficiency was conducted.

7.1.1 Data Reliability

The level of reliability provided by the link layer will depend on the needs of the application and user-specified constraints. In many wireless sensor networks, such as machine monitoring and vehicle detection networks, the actual data will need to be transferred with an extremely low probability of error.

In our application, we assume that the objects of interest can have high mobility (e.g. vehicles moving) and that the nodes themselves are immobile. Thus, the coherence time of the channel is not much larger than the signaling time of $1\mu s$. Given this scenario, we can assume that nodes communicate over a frequency non-selective, slow Rayleigh fading channel with additive white Gaussian noise. This is a reasonable channel model to use for communication at 2.4 GHz where line-of-sight communication is not always possible[2].

Consider one node transmitting data to another over such a channel using the radio described in Section 5.2. The radio presented uses non-coherent binary frequency-shift keying (FSK) as the modulation scheme. For comparison purposes, the best achievable probability of error using raw, non-coherent binary FSK over a slowly fading Rayleigh channel will be presented. Let P_e be a function of the received *energy per bit to noise power ratio* $(\gamma_{b,rx})$.

[2]This model does not apply to all wireless sensor networks; Rician or log-normal fading may be more appropriate. However, all of these fading channels have similar characteristics.

In general, $\gamma_{b,rx} = \alpha^2(E_b/N_0)$, where α is a random variable for a fading channel. It is shown in [33] that the probability of error using non-coherent, orthogonal binary FSK is $P_e = \frac{1}{2+\overline{\gamma}_{b,rx}}$, where $\overline{\gamma}_{b,rx}$ is the average $\gamma_{b,rx}$. Unfortunately, this does not directly tell us the amount of transmit power P_{out} that must be used in order to get a certain probability of error. In order to determine P_e as a function of P_{out}, we must consider the implementation of the radio. In general, one can convert $\gamma_{b,rx}$ to P_{out} using

$$\left(\frac{E_b}{N_0}\right)_{rx} = \frac{P_{out}}{P_{loss}\overline{\alpha}} \cdot \frac{1}{WN_{th}N_{rx}}$$

where P_{loss} represents the large-scale path loss, $\overline{\alpha}$ is the average attenuation factor due to fading, W is the signal bandwidth, N_{th} is the thermal noise and N_{rx} is the noise contributed by the receiver circuitry known as the noise figure. In general, $P_{loss} \propto \frac{1}{4\pi d^k}, 2 \leq k \leq 4$.

A conservative estimate for $P_{loss}\overline{\alpha}$ is about 70 dB. With a signal bandwidth of $W = 1$ MHz, $N_{th} = -174$ dBm and $N_{rx} \approx 10$ dB, we find that $P_{out} = E_b/N_0 - 34$ dBm assuming a data rate of 1 Mbps. This equation can be used to find the transmit power needed to obtain a certain average E_b/N_0. The uncoded curve in Figure 13 shows the probability of error plotted against the output power of the transmitter.

Since using a power amplifier alone is highly inefficient, forward error correction (FEC) can be applied to the data to decrease the probability of error. Many types of error-correcting codes can be used to improve the probability of bit error. However, we will only consider convolutional codes with base coding rates of $R_c = 1/2$ and punctured derivatives. For a frequency non-selective, Rayleigh fading channel, a bound on the P_e can be determined by applying

$$P_e < \frac{1}{k} \sum_{d=d_{\text{free}}}^{\infty} \beta_d P(d).$$

Here d represents the Hamming distance between some path in the trellis decoder and the all-zero path, the coefficients $\{\beta_d\}$ can be obtained from the expansion of the first derivative of the transfer function, $P(d)$ is the first-event error probability, and d_{free} is the minimum free distance [33]. Figure 13 shows the P_e for codes with varying rates R_c and constraint lengths K. Note that the probabilities shown assumes the use of a hard decision Viterbi decoder at the receiver. We see that greater redundancy (lower rate) or more memory (higher constraint length) lowers the output power for a given P_e. From this perspective, coding should *always* be used.

7.1.2 Energy Consumption of Coding

As shown, the use of FEC can decrease the transmit power. However, the additional processing required will increase the energy of computation. Depending on the underlying architecture, energy cost can be significant. Additional processing energy, denoted by E_{dsp} must be expended in order to encode and decode the data. Additional energy cost will be also be incurred during the communication of the message since encoding a bit stream will increase the size of the packet by approximately $1/R_c$, thereby increasing T_{on} and the radio energy required to transmit a packet. If we denote the energy to encode as $E_{dsp}^{(e)}$ and decode data as $E_{dsp}^{(d)}$, then the total energy cost of the communication can be derived from (1)

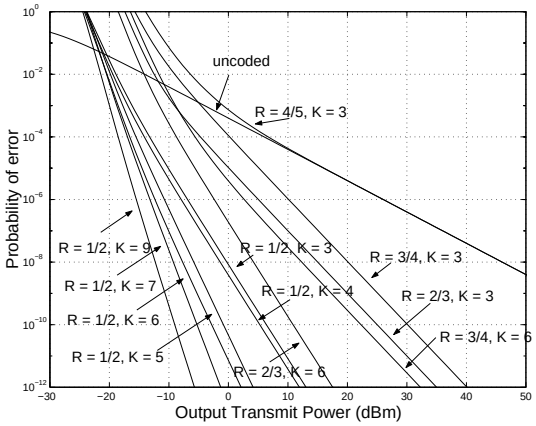


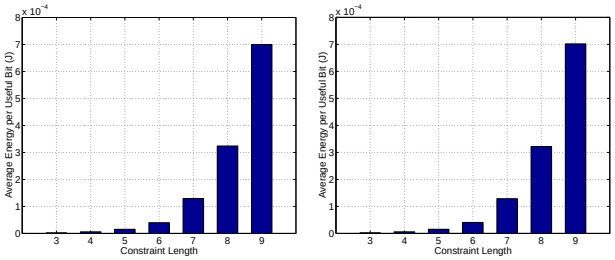
Figure 13: The probability of error of different rate convolutional codes plotted versus the transmit power for the radio described in Section 5.2. $P_{loss} = 70$ **dB,** $N_{rx} = 10$ **dB, and** $R = 1$ **Mbps.**

as

$$E = P_{tx}(T_{on-tx} + T_{st}) + P_{out}T_{on-tx} + E_{dsp}^{(e)} + P_{rx}(T_{on-rx} + T_{st}) + E_{dsp}^{(d)} \quad (3)$$

Given this model, we can then derive the average energy to transmit, receive, encode and decode each information bit. If R_c is the code rate and L is the packet length transmitted, then the number of information bits is $L' \approx LR_c$. Thus, the energy per useful bit is $E_b = E/L'$.

In general, for convolutional codes, the energy required to encode data is negligible. However, performing Viterbi decoding on a StrongARM using C is energy-intensive. We have measured the energy per useful bit required to decode $1/2$ and $1/3$-rate convolutional codes with varying constraint length on the StrongARM. The results are shown in Figure 14. Two observations can be derived



(a) Rate 1/2 (b) Rate 1/3

Figure 14: (a) Decoding energy per useful bit for $R_c = 1/2$ **codes with** $K = 3$ **to** 9 **(b) Decoding energy per useful bit for** $R_c = 1/3$ **codes with** $K = 3$ **to** 8**.**

from these graphs. First, the energy consumption scales exponentially with the constraint length. This is expected since the number of states in the trellis increases exponentially with constraint length.

Another observation to make is that the energy consumption seems to be independent of the coding rate. This is reasonable since the rate only affects the number of bits sent over the transmission. A lower rate code does not necessarily increase the computational energy since the number of states in the Viterbi decoder is unaffected. In addition, the cost of reading the data from memory is dominated by the updating of the survivor path registers in the Viterbi algorithm. The size of the registers is proportional to the constraint length and is not determined by the rate. Therefore, given two convolutional codes $\mathcal{C}_1$ and $\mathcal{C}_2$ both with constraint lengths K, where $R_{\mathcal{C}_1} < R_{\mathcal{C}_2}$, the per bit energy to decode $\mathcal{C}_1$ and $\mathcal{C}_2$ is the same even though more bits are transmitted when using $\mathcal{C}_1$.

Given the data in Figure 14, we can now determine which convolutional code to use to minimize the energy consumed by communication for a given probability of error. In Figure 15, the total energy per information bit E_b is plotted against P_b. Figure 15 shows

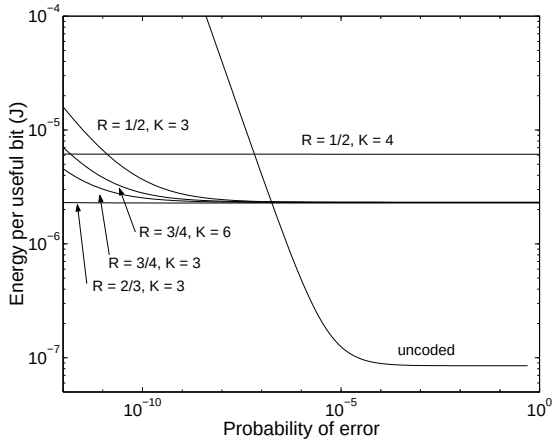


Figure 15: Energy per useful bit plotted versus P_b **of an uncoded signal and a few convolutional codes with varying rates and constraint lengths (**$P_{loss} = 70$ dB**,** $N_{rx} = 10$ dB**,** $R = 1$ Mbps**). The number of information bits is 10000.**

that the energy per bit using no coding is lower than that for coding for $P_b > 10^{-5}$. The reason for this result is that the energy of computation, i.e. decoding, dominates the energy used by the radio for high probabilities of error. For example, assuming the model described in (3) and $P_{out} = 0$ dBm, the communication energy to transmit and receive per useful bit for an $R_c = 1/2$ code is 85 nJ/bit. On the other hand, the energy to decode an $R_c = 1/2, K = 3$ code on the SA-1100 is measured to be 2200 nJ per bit.

At lower probabilities of error, the power amplifier energy begins to dominate. At these ranges, codes with greater redundancy have better performance. These results imply that coding the data is not always the right thing to do if energy-efficiency is a crieria. One may suspect that this result is due to the inefficiency of the StrongARM in performing error correction coding. However, we will show that his result will hold even if a more efficient implementation of the Viterbi decoder is used[3].

[3]Note that the x-axis of the graph extends below 10^{-9}. At such low P_b, these results are likely invalid. They are shown so that the different codes can be distinguished.

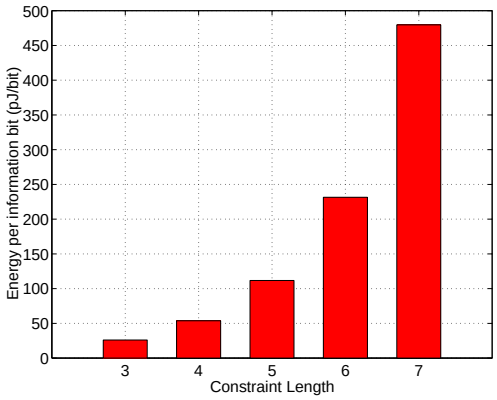


Figure 16: Measured decoding energy per useful bit for $R_c = 1/2$ codes with $K = 3$ to 7 using our synthesized VLSI implementation.

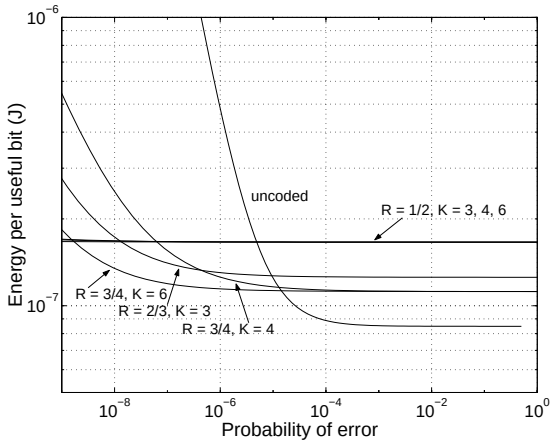


Figure 17: The energy per useful bit plotted against P_b using no coding and various convolutional codes ($P_{loss} = 70$ dB, $N_{rx} = 10$ dB, $R = 1$ Mbps). The number of information bits is 10000.

Since the use of the StrongARM to perform Viterbi decoding is energy inefficient, a dedicated integrated circuit solution to perform decoding is preferred. To explore the power characteristics of dedicated Viterbi decoders, we implemented $1/2$-rate decoders with different constraint lengths and synthesized them using 0.18 μm TSMC ASIC technology. Our designs are fully parallel implementations of the Viterbi algorithm where a separate add-compare-select (ACS) unit is used for each state. Using Synopsys Power Compiler, we estimated the energy per bit used by our designs during the decoding of 20000 bits. Figure 16 shows the energy per bit for various constraint lengths. Using our implementation, in addition to our radio model, we determined the minimum energy code to use for a given probability of error. In Figure 17, the energy per useful bit is plotted against P_b. From the graph, one can see that the communication/computation scheme to use will be dependent on the probability of error desired at the receiver. For $P_b > 10^{-4}$, no coding should be used. This is due to the fact that the transceiver power ($P_{tx/rx}$) is dominant at high probabilities of error. Since coding the data will increase the on time (T_{on}) of the transceiver, using coding will increase the overall energy per useful bit[4]. Note that once $P_b < 10^{-5}$, the overall communication energy with coding is smaller since the energy of the power amplifier (P_{out}) will begin to dominate. Figure 17 reinforces the idea that coding the data may not necessarily be the best solution if energy-efficiency is a criteria. Indeed, the coding strategy is highly dependent on the desired output quality of the user.

7.2 Low Power MAC Protocols

In this section, we investigate how the non-ideal behavior of physical layer electronics in our system can affect the design of the media-access control (MAC) protocol. We evaluate the energy-efficiency of various MAC schemes for use in an asymmetric, *coordinated* sensor network where clusters or cells are formed around high-powered basestations. An example of such a network is machine diagnosis in an industrial setting. Energy-constrained sensors communicate to a single high-powered basestation nearby (< 10 m). In this application, the latency requirements of the data must be strictly guaranteed. Note that data aggregation techniques cannot be applied since sensors monitor different properties (e.g. temperature, pressure) of the machines. Table 3 summarizes the application specification.

Table 3: Machine monitoring specification.

Cell Density	$<$ 300 in 5 m $\times$ 5 m $<$ 3000 in 100 m $\times$ 100 m
Range of Link	$<$ 10 m
Message rate (L = 2 bytes)	average : 20 msgs/s maximum : 100 msgs/s minimum : 2 msgs/s
Error Rate and Latency	10^{-6} after 5 ms 10^{-9} after 10 ms 10^{-12} after 15 ms
System Lifetime	5 years
Frequency band	2.400 - 2.4835 GHz (ISM)

7.2.1 *Determining N_{rx} and N_{tx}*

Recall from Section 5.2 that N_{rx} and N_{tx} are factors that relate to the underlying MAC protocol and intended application. In this application, these parameters depend largely on the latency requirement specified by the user. Given these requirements, we derive the best parameters to use for a low power MAC protocol for a single cell where a high-powered basestation gathers data from the sensors.

First, we examine a few candidate MAC protocols. For the purposes of this discussion, we limit our choice of MAC protocols to time division multiple access (TDMA) and frequency division multiple access (FDMA) schemes. For this application, other more complex multi-access schemes may not be appropriate. In particular, on-demand schemes (e.g. CSMA-CA) that require handshaking not only increase the latency of the data, but detrimentally affect energy. Furthermore, in this application, efficient use of bandwidth is not a concern.

In a TDMA scheme, the full bandwidth of the channel is dedicated

[4]At high P_b, the required transmit power is not very high and the energy to perform decoding is small. Thus, the transceiver energy is dominant. Since coding increases the number of bits to transmit, the energy with coding is greater.

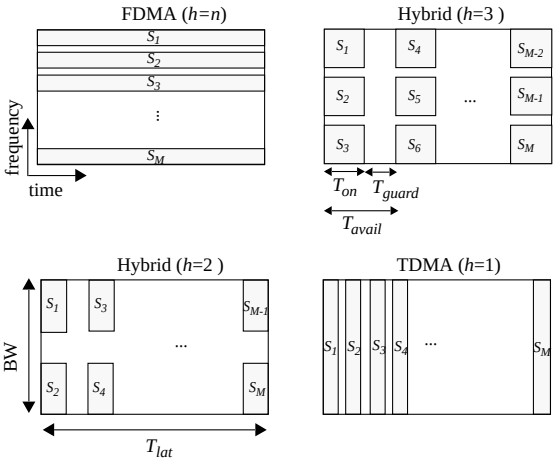


Figure 18: Multiple access methods.

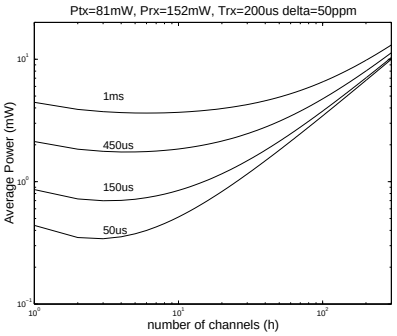


Figure 19: Energy with different T_{st}.

to a single sensor for communication purposes. Thus, the signal bandwidth per sensor is equal to the available bandwidth and sensors can transmit at the highest data rate. Since the transmit on-time (T_{on-tx}) of the radio model described in (1), is inversely proportional to the signal bandwidth, the T_{on-tx} is minimized in TDMA schemes. On the other hand, in an FDMA scheme, the signal bandwidth (total available bandwidth divided by number of sensors) is minimal. Thus, T_{on-tx} is at its maximum. A hybrid scheme involving both TDMA and FDMA (TDM-FDM) is also possible. In a TDM-FDM scheme, both time and frequency are divided into available transmission slots. Figure 18 illustrates each of the different multiple-access schemes considered, where a shaded area indicates a valid transmission slot for sensor S_i.

In the schemes where TDM is employed, note that a downlink from the basestation to the sensors is required to maintain time synchronization among the nodes in the network. Due to the finite error among each sensor's reference clock, the basestation must send out *synchronization* packets (SYNCs) to avoid collisions among transmitted packets. Hence, the receiver circuitry of each sensor must be activated periodically to receive the SYNC signals. As explained in Section 5.2, the receiver uses more power than the transmitter. Thus, we need to reduce the average number of times the receiver is active. The number of times the receiver needs to be active (N_{rx}) will depend on T_{guard}, the minimum time difference between two time slots in the same frequency band, as shown in Figure 18. During T_{guard}, no sensor is scheduled to transmit any data. Thus, a larger guard time will reduce the probability of packet collisions and thus, reduce the frequency of SYNC signals and N_{rx}.

If two slots in the same frequency band are separated by T_{guard}, it will take T_{guard}/δ seconds for these two packets to collide, where δ is the percent difference between the two sensors' clocks. Hence the sensors must be resynchronized at least δ/T_{guard} number of times every second. In other words, the average number of times the receiver is active per second, $N_{rx} = \delta/T_{guard}$. Assuming that the total slot time available $T_{avail} = T_{on} + T_{guard}$, we can derive a formula relating N_{rx} to T_{lat}, the latency requirement of the transmitted packet, as follows,

$$N_{rx} = \frac{\delta}{T_{guard}} = \frac{\delta}{T_{avail} - T_{on}} = \frac{\delta}{(\frac{T_{lat}}{M} - \frac{L}{W})h} \qquad (4)$$

where W is the available bandwidth, L is the length of the data packet in bits, T_{lat} is the latency requirement of the transmitted packet, h is the number of channels in the given band W, and M is the number of sensors. Here we have assumed that the data rate R equals the signal bandwidth. Hence, $T_{on} = \frac{L}{R} = \frac{L}{W/h}$. From (4), we see that as the number of channels decreases, the guard time becomes larger and N_{rx} is reduced. It is also apparent that the advantage of ideal FDMA is that a receiver at the sensor is not needed (i.e., as $T_{guard} \to \infty, N_{rx} \to 0$).

From (1) and (4), we can determine an analytical formula to find h_{opt}, the number of channels which gives the lowest power consumption.

$$h_{opt} = \sqrt{\frac{\delta P_{rx}(T_{on-rx} + T_{st})}{(\frac{T_{lat}}{N_{cell}} - \frac{L}{W})N_{tx}(P_{tx} + P_{out})\frac{L}{W}}} \propto \sqrt{\frac{P_{rx}}{N_{tx}P_{tx}}}. \qquad (5)$$

Clearly, we see that h_{opt} is determined by the ratio of the power consumption of the transmitter to the receiver. As expected, if the receiver consumes less power, a TDMA scheme is favored. On the other hand, if the receiver uses more power, FDMA is more appropriate.

An example of the previous analysis is performed in a scenario where a sensor on average sends twenty 100-bit packets/s ($N_{tx} = 20$ times/s, $L = 100$ bits) and the latency requirement is 5 ms ($T_{lat} = 5$ ms). Also, we assume that $W = 10$ MHz and the number of sensors in a cell is $M = 300$. The resulting average power consumption is plotted in Figure 19 and 20 where the horizontal axis represents the number of channels available ($h = 1$: TDMA, $h = 300$: FDMA) and the vertical axis is the average power consumption.

In Figure 19, the average power consumption is plotted for various startup times ($T_{st} = 50$ μs to 1 ms). We can see that the average power reaches a minimum value when a hybrid TDM-FDM scheme is used. The variation in power consumption for different h gets smaller as the T_{st} is increased since the overall power consumption is dominated by the startup time. In Figure 20, we can see how the power consumption curve will vary if different radio receivers are used. That is, we vary P_{rx} while maintaining a constant P_{tx}. We see from this figure that h_{opt} increases as receiver power increases. Notice that despite the fact that a TDMA scheme will have the minimum transmit on-time, the TDMA scheme does

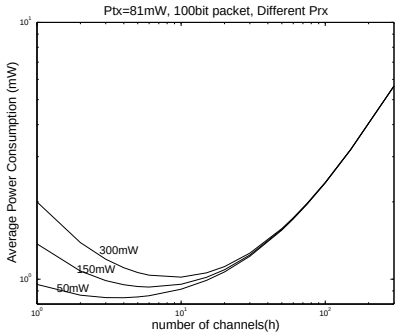


Figure 20: Energy with different E_{rx}/E_{tx}.

not achieve the lowest power. Indeed, as the number of channels is reduced, the guard time decreases. This implies that more synchronization is needed and thus, the receiver power starts consuming a large portion of the total power.

7.3 Modulation Schemes

The modulation scheme used by the radio is another important factor which can strongly impact the energy consumption of the node. As evidenced by (1), one way to increase the energy efficiency of communication is to reduce the transmit on time of the radio. This can be accomplished by sending multiple bits per symbol, that is, by using M-ary modulation. Using M-ary modulation, however, will increase the circuit complexity and power consumption of the radio. In addition, when M-ary modulation is used, the efficiency of the power amplifier is also reduced. This implies that more power will be needed to obtain reasonable levels of transmit output power.

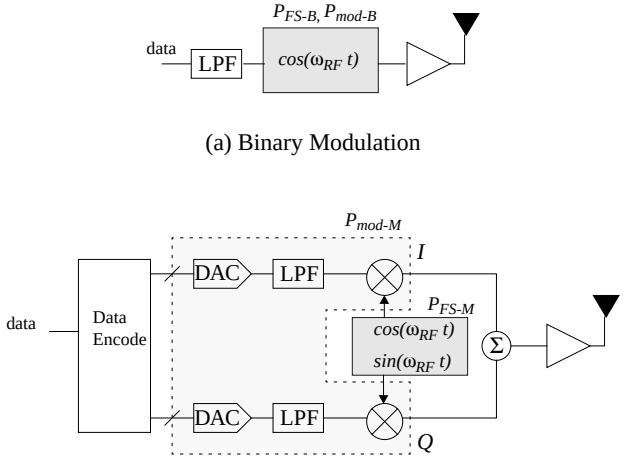
The architecture of a generic binary modulation scheme is shown in Figure 21(a), where the modulation circuitry is integrated together with the frequency synthesizer [19, 34]. To transmit data using this architecture, the VCO can be either directly or indirectly modulated. The architecture of a radio that uses M-ary modulation is shown in Figure 21(b). Here, the data encoder parallelizes serially input bits and then passes the result to a digital-to-analog converter (DAC). The analog values produced serve as output levels for the in-phase (I) and quadrature (Q) components of the output signal. The energy consumption for the binary modulation architecture can be expressed as

$$E_{bin} = (P_{mod-B} + P_{FS-B})T_{on} + P_{FS-B}T_{st} + P_{out-B}T_{on} \quad (6)$$

while the energy consumption for M-ary modulation is

$$E_M = (P_{mod-M} + P_{FS-M})\frac{T_{on}}{n} + P_{FS-M}T_{st} + P_{out-M}\frac{T_{on}}{n} = \frac{(\alpha P_{mod-B} + \beta P_{FS-B})T_{on}}{\log_2 M} + \beta P_{FS-B}T_{st} + \frac{P_{out-M}T_{on}}{\log_2 M}. \quad (7)$$

In these equations, P_{mod-B} and P_{mod-M} represents the power consumption of the binary and M-ary modulation circuitry, P_{FS-B} and P_{FS-M} represent the power consumed by the frequency synthesizer, P_{out-B} and P_{out-M} represent the output



(a) Binary Modulation

(b) M-ary Modulation

Figure 21: Binary vs. M-ary Modulation.

transmit power for binary or M-ary modulation, T_{on} is the transmit on time, and T_{st} is the startup time. As mentioned, for the same number of bits, T_{on} for M-ary modulation is less than T_{on} for binary modulation. Note that $n = \log_2 M$, the number of bits per symbol. The factors of α and β can be expressed as

$$\alpha = \frac{P_{mod-M}}{P_{mod-B}} \qquad \beta = \frac{P_{FS-M}}{P_{FS-B}}.$$

Here, α represents the radio of the power consumption of the modulation circuitry between M-ary and binary modulation, while β is the ratio of synthesizer power between the M-ary and binary schemes. Basically these parameters represent the overhead that is added to the modulation and frequency synthesizer circuitry when one switches from a binary modulation scheme to an M-ary modulation scheme.

When we compare (6) and (7), we can see that M-ary modulation achieves a lower energy consumption when the following condition is satisfied.

$$\alpha < n\left[1 + \frac{P_{FS-B}[(1-\frac{\beta}{n})T_{on} + (1-\beta)T_{st}]}{P_{mod-B}T_{on}}\right] + n\frac{P_{out-B}}{P_{mod-B}} - \frac{P_{out-M}}{P_{mod-B}} \quad (8)$$

$$\approx n\left[1 + \frac{P_{FS-B}[(1-\frac{\beta}{n})T_{on} + (1-\beta)T_{st}]}{P_{mod-B}T_{on}}\right] \quad (9)$$

The last two terms of (8) can be ignored since P_{out-B} and P_{out-M} are negligible compared to the power of the frequency synthesizer. A comparison of the energy consumption of binary modulation and M-ary modulation is shown in Figure 22. In the figure, the ratio of the energy consumption of M-ary modulation to the energy consumption of binary modulation is plotted versus the overhead α. In Figure 22 we vary M to produce different M-ary modulation schemes. For each scheme, we also vary the startup time and assume that 100 bit packets are sent at 1 Mbps. This implies that in an M-ary scheme, $1/\log M$ megasymbols are sent per second and the on time is decreased.

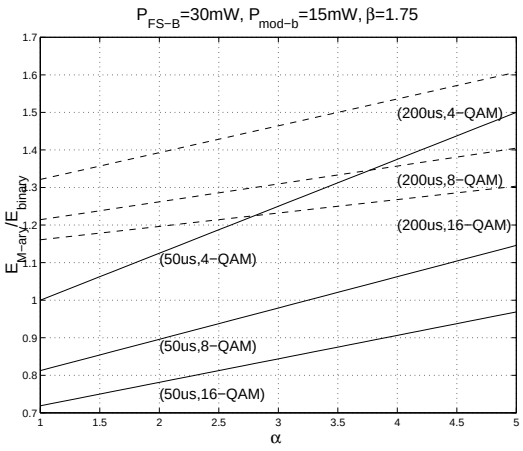


Figure 22: The ratio of the energy consumed by M-ary modulation to the energy consumed by binary modulation is plotted versus α, the ratio of the modulation circuitry power consumption.

As expected, the M-ary modulation scheme achieves the lowest energy when the overhead α is small and T_{st} is about 50μs. When the startup time is about 200μs, however, the energy consumption is higher for M-ary modulation regardless of α. This is because the energy consumption due to startup time dominates the energy consumption due to transmit on-time. Hence, reducing T_{on} by using a larger M has a negligible effect on the total energy consumption.

8 CONCLUSION

Throughout this paper, we have shown techniques at various levels of the system hierarchy that take advantage of underlying hardware to produce more energy efficient solutions. In some instances, we have shown how to take advantage of hooks and knobs in the physical layer to build more energy-efficient protocols and algorithms. In other instances, we demonstrated how non-idealities of the hardware can be mitigated by making careful, yet simple protocol design choices. As a whole, we advocate a physical layer driven approach to protocol and algorithm design for wireless sensor networks. In order to meet the system lifetime goals of wireless sensor applications, considering the parameters of the underlying hardware are critical. If protocol designers treat the physical layer as a black box, system designers may design protocols that are detrimental to energy consumption.

9 ACKNOWLEDGEMENTS

The authors would like to thank ABB Corporation for providing invaluable application information. Thanks also goes out Ben Calhoun, Fred Lee, and Theodoros Konstantakopoulos for their help in designing the node. Finally, the authors would like to thank Manish Bhardwaj and all the reviewers for their insightful comments that helped to improve the paper.

This research is sponsored by the Defense Advanced Research Project Agency (DARPA) Power Aware Computing/Communication Program and the Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-00-2-0551. Alice Wang is funded by a Lucent Graduate Research Fellowship, while Rex Min is funded by an NDSEG Fellowship.

REFERENCES

[1] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next Century Challenges: Scalable Coordination in Sensor Networks," in *Proc. ACM MobiCom '99*, pp. 263–270, August 1999.

[2] K. Bult *et al.*, "Low Power Systems for Wireless Microsensors," in *Proc. ISLPED '96*, pp. 17–21, August 1996.

[3] J. Kahn, R. Katz, and K. Pister, "Next Century Challenges: Mobile Networking for Smart Dust," in *Proc. ACM MobiCom '99*, pp. 271–278, August 1999.

[4] N. Priyantha, A. Chakraborty, and H. Balakrishnan, "The Cricket Location-Support System," in *Proc. of MobiCom '00*, pp. 32–43, August 2000.

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

[6] L. Nord and J. Haartsen, *The Bluetooth Radio Specification and The Bluetooth Baseband Specification*. Bluetooth, 1999-2000.

[7] A. Wang, W. Heinzelman, and A. Chandrakasan, "Energy-Scalable Protocols for Battery-Operated Microsensor Networks," in *IEEE SiPS '99*, Oct. 1999.

[8] G. Asada *et al.*, "Wireless Integrated Network Sensors: Low Power Systems on a Chip," in *Proc. ESSCIRC '98*, 1998.

[9] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," in *Proc. ACM MobiCom '00*, pp. 56–67, August 2000.

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

[11] J.-H. Chang and L. Tassiulas, "Energy Conserving Routing in Wireless Ad-hoc Networks," in *Proc. IEEE INFOCOM '00*, pp. 22–31, March 2000.

[12] S. Singh, M. Woo, and C. Raghavendra, "Power-Aware Routing in Mobile Ad Hoc Networks," in *Proc. ACM MobiCom '98*, Oct. 1998.

[13] V. Rodoplu and T. H. Meng, "Minimum Energy Mobile Wireless Networks," *IEEE Journal on Selected Areas in Communications*, vol. 17, pp. 1333–1344, August 1999.

[14] P. Lettieri and M. B. Srivastava, "Adaptive Frame Length Control for Improving Wireless Link Throughput, Range, and Energy Efficiency," in *Proc. INFOCOM '98*, pp. 564–571, March 1998.

[15] M. Zorzi and R. Rao, "Error Control and Energy Consumption in Communications for Nomadic Computing," *IEEE Transactions on Computers*, March 1997.

[16] B. Narendran, J. Sienicki, S. Yajnik, and P. Agrawal, "Evaluation of an Adaptive Power and Error Control Algorithm for Wireless Systems," in *IEEE Inteernational Conference on Communiations (ICC '97)*, 1997.

[17] J.-P. Ebert and A. Wolisz, "Combined Tuning of RF Power and Medium Access Control for WLANs," in *Proc. MoMUC '99*, 1999.

[18] R. Kravets, K. Schwan, and K. Calvert, "Power-Aware Communication for Mobile Computers," in *Proc. MoMUC '99*, November 1999.

[19] National Semiconductor Corporation, *LMX3162 Evaluation Notes and Datasheet*, April 1999.

[20] A. Klaiber, "The Technology Behind Crusoe Processors." Transmeta Corporation. http://www.transmeta.com, January 2000.

[21] I. Corporation, "Intel XScale Microarchitecture." http://developer.intel.com/design/intelxscale/, 2000-2001.

[22] A. Sinha and A. Chandrakasan, "Energy Aware Software," in *Proc. VLSI Design '00*, pp. 50–55, Jan. 2000.

[23] V. De and S. Borkar, "Technology and Design Challenges for Low Power and High Performance," in *Proc. ISLPED '99*, pp. 163–168, August 1999.

[24] M. Perrott, T. Tewksbury, and C. Sodini, "27 mW CMOS Fractional-N Synthesizer/Modulator IC," in *ISSCC Digest of Technical Papers*, pp. 366–367, February 1997.

[25] S. Haykin, J. Litva, and T. Shepherd, *Radar Array Processing*. Springer-Verlag, 1993.

[26] R. Min, T. Furrer, and A. Chandrakasan, "Dynamic Voltage Scaling Techniques for Distributed Microsensor Networks," in *Proc. WVLSI '00*, April 2000.

[27] K. Govil, E. Chan, and H. Wasserman, "Comparing Algorithms for Dynamic Speed-Setting of a Low-Power CPU," in *Proc. of MobiCom '95*, August 1995.

[28] T. Pering, T. Burd, and R. Brodersen, "The Simulation and Evaluation of Dynamic Voltage Scaling Algorithms," in *Proc. of ISLPED '98*, August 1998.

[29] M. Weiser *et al.*, "Scheduling for reduced CPU energy," in *Proc. of 1st USENIX Symp. on Operating System Design and Implementation*, pp. 13–23, Nov. 1994.

[30] J. Lorch and A. Smith, "Reducing Processor Power Consumption by Improving Processor Time Management in a Single-User Operating System," in *Proc. of MobiCom '96*, 1996.

[31] T. Simunic *et al.*, "Dynamic Power Management for Portable Systems," in *Proc. of MobiCom '00*, pp. 11–19, August 2000.

[32] "The Advanced Configuration & Power Interface." http://www.teleport.com/∼acpi.

[33] J. Proakis, *Digital Communications*. New York City, New York: McGraw-Hill, 4th ed., 2000.

[34] N. Filiol, T. Riley, C. Plett, and M. Copeland, "An Agile ISM Band Frequency Synthesizer with built-in GMSK Data Modulation," in *IEEE Journal of Solid-State Circuits*, vol. 33, pp. 998–1008, July 1998.