Energy-Efficient Link Layer for Wireless Microsensor Networks

Eugene Shih, Benton H. Calhoun, SeongHwan Cho, and Anantha P. Chandrakasan
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
{*eugene, bcalhoun, chosta, anantha*}*@mtl.mit.edu*

Abstract

Wireless microsensors are being used to form large, dense networks for the purposes of long-term environmental sensing and data collection. Unfortunately, these networks are typically deployed in remote environments where energy sources are limited. Thus, designing fault-tolerant wireless microsensor networks with long system lifetimes can be challenging. By applying energy-efficient techniques at all levels of the system hierarchy, system lifetime can be extended. In this paper, energy-efficient techniques that adapt underlying communication parameters will be presented in the context of wireless microsensor networks. In particular, the effect of adapting link and physical layer parameters, such as output transmit power and error control coding, on system energy consumption will be examined.

1. Introduction

Recently, researchers have been increasingly interested in wireless microsensor networks [1, 2, 5]. This rising interest is in large part due to the compelling applications that will be enabled once wireless microsensor networks have been deployed [5]. In a wireless microsensor network, hundreds to thousands of small, sensor nodes are scattered over some environment for the purpose of gathering data. Each of these distributed sensors contain computation and communication elements and can be designed to provide long-term, remote autonomous environmental monitoring. When events of interest are detected, sensors may process the data before sending information about the environment to a remote basestation. Because of the remote nature of these networks and the size of the individual nodes, however, nodes do not have access to unlimited energy. Thus, in order to prolong system lifetimes, energy-efficient algorithms and protocols should be used. At the same time, these techniques should also be aware of user-specified quality requirements such as latency and data precision. Since these parameters express an application's quality needs, energy-efficient algorithms and protocols must be careful not to compromise this quality while minimizing energy consumption.

The need to minimize energy consumption while maintaining user constraints makes the design of wireless microsensor networks challenging. While techniques to minimize the energy consumption of portable, multimedia devices have been studied extensively [3, 4], these techniques may not be applicable to wireless microsensors. For example, while conventional hand-held devices only need to last hours or days, microsensor nodes need to last several years. Therefore, different energy-efficient techniques will need to be applied. In this paper, energy-efficient techniques for *communication* among the nodes will be presented. Since wireless communication over long distances can be expensive, minimizing the energy for communication is very important. In general, the minimum output transmit power required to transmit a signal over a distance d is proportional to d^n where $2 \leq n \leq 4$.

In the design of any communication system, one parameter of interest to users is the reliability of the links between a transmitter and receiver. Reliable data transfer can be provided either by increasing the output transmit power (P_{out}) of the radio or by adding forward error correction (FEC) to the data. With the use of FEC, we can decrease the probability of bit error (P_b) for any fixed value of the output transmit power. However, FEC will also require additional processing and thus, additional energy at the transmitter and receiver. Depending on the FEC algorithm used and the implementation of the algorithm, the additional processing may require so much power that any savings made in the reduction of the output transmit power will become negligible. In this paper, we attempt to minimize the system energy required to send data between a transmitter and receiver by partitioning the communication energy between the output transmit power and the processing required by error-correction coding.

0-7695-1056-6/01 $10.00 © 2001 IEEE

2. Energy Cost of Computation

During communication between two nodes, energy is expended during transmission of the data (output transmit power) *and* when framing and error correction is performed. We define the *communication energy* to be the sum of the energy required to transmit data using a radio *and* the energy required to perform encoding and decoding of the data[1]. A similar definition is introduced in [6] in the context of portable terminals.

In order to evaluate the communication energy in wireless microsensor networks, we have implemented a sensor node with appropriate processing and communication elements. Before discussing the tradeoffs that can be made between coding energy and output transmit power, we will briefly describe the implementation of our node.

2.1. The μAMPS Wireless Sensor Node

The μAMPS wireless sensor node has the ability to scale the energy consumption of many different subcomponents in response to changes in the environment, state of the network, *and* application requirements in order to maximize system lifetime and reduce energy consumption of the node. Thus, all layers of the system can adapt layer-specific parameters (e.g. error correction scheme) to minimize energy usage. Figure 1 gives an overview of the architecture of the sensor node. The node is designed with collaborative sensor applications in mind. However, the flexibility of the architecture allows it to be used in different application scenarios.

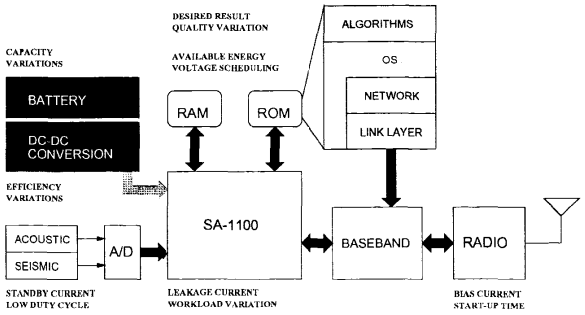

Figure 1. Architectural overview of our microsensor node.

Since the μAMPS node is designed with sensing applications in mind, our initial node contains an acoustic sensor and a seismic sensor. Data sensed by either sensor is passed through an anti-alias filter and directed to an analog-to-digital (A/D) converter. Once data is sampled by the sensing subsystem, it is transferred to on-board RAM. The sensor, analog filters, and A/D collectively use 5 mA at 5 V.

The data collected is processed by a StrongARM (SA-1100) microprocessor. Selected for its low power consumption, high performance, and static CMOS design, the SA-1100 can be adapted to support dynamic voltage scaling. The clock speed can be dynamically adjusted from 50 to 206 MHz, while the voltage supply has a range of 0.9 to 1.5 V. In addition to the processor, ROM and RAM are also on-board. A lightweight, multi-threaded "μ-OS" running on the SA-1100 has been customized to allow software to scale the energy consumption of the processor. The μ-OS, signal processing algorithms, and the network protocols are stored in ROM.

In order to deliver data or control messages to neighboring nodes, the data is transmitted wirelessly using a radio based on a commercial single-chip 2.4 GHz transceiver with an integrated frequency synthesizer [9]. The on-board phase-locked loop (PLL), transmitter chain, and receiver chain are capable of being shut-off via software or hardware control for energy savings. To transmit data, an external voltage-controlled oscillator (VCO) is directly modulated. By directly modulating the VCO, the circuit implementation is simpler and power consumption is reduced. However, the amount of data that can be transmitted continuously is limited. The radio module is capable of transmitting up to 1 Mbps at a range of up to 10 m.

2.2. Computation Energy Model

The encoding and decoding of error-correcting codes can be performed on different platforms. In our initial system, coding will be performed on the StrongARM using C. Instead of modeling the energy required for encoding and decoding the data, the energy consumed will be directly measured.

2.3. Radio Energy Model

The average energy consumption of radio communication can be modeled by:

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

where $P_{tx/rx}$ is the power consumption of the transmitter/receiver, P_{out} is the output transmit power which drives the antenna, $T_{on-tx/rx}$ is the transmit/receive on-time (actual data transmission/reception time), and $T_{startup}$ is the start up time of the transceiver as shown in Figure 2. Note

[1] This definition does not fully capture the energy of all tasks required for communication. However, in this paper, we will ignore the energy required to perform higher protocol layer operations.

that if L is the size of the packet in bits and R is the data rate in bits per second, then $T_{on-tx/on-rx} = L/R$.

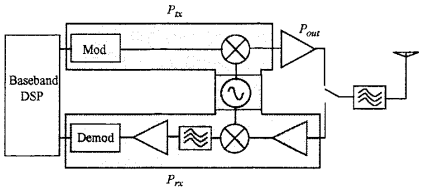

Figure 2. A diagram of the radio model.

In this radio model, the power amplifier needs to be on only when communication occurs. In addition, during the 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.

It is necessary to highlight a few key points about the radio we use in our design. First, note that the power consumption of the transceiver ($P_{tx/rx}$) dominates the output transmit power (P_{out}). Since wireless sensor networks are designed to operate over short distances, this is a reasonable assumption. In addition, the transceiver power does not vary over the data rate, R. At the 2.4 GHz frequency band (as in other gigahertz bands), 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 [10]. 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 very 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, $P_{rx} = 180$ mW, $T_{startup} = 450$ μs and $P_{out} \approx 0$ dBm, the effect of the startup transient is shown in Figure 3, 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 and receive 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 the commercial low power transceiver we are using in the μAMPS node.

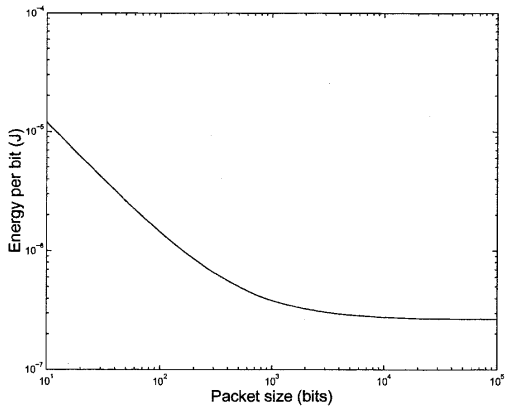

Figure 3. Effect of startup transient where R = 1 Mbps, $T_{startup} \approx 450$ μs, $P_{tx} = 81$ mW, $P_{tx} = 180$ mW and $P_{out} = 0$ dBm.

3. Impact of Link Layer Parameters on System Energy

In general, the "link layer" has a variety of purposes in the protocol stack. 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 [8, 7], a similar exploration of the impact of adapting packet size and error control on system energy efficiency was conducted.

3.1. Data Reliability

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

In a wireless microsensor network, we will assume that nodes communicate over a frequency non-selective, slow Rayleigh fading channel. Consider one node transmitting data to another over such a channel using the radio described in Section 2.3. 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.

In general, $\gamma_{b,rx} = \alpha^2(E_b/N_0)$, where $\gamma_{b,rx}$ is the

received *energy per bit to noise power ratio* and α is a random variable describing the attenuation property of the fading channel. It is shown in [11] that the probability of bit error using non-coherent, orthogonal binary FSK is $P_b = \frac{1}{2+\overline{\gamma}_{b,rx}}$. Unfortunately, this does not directly tell us the transmit power P_{out} that must be used in order to get a certain probability of error. In order to determine P_b 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

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

where P_{loss} represents the large-scale path loss, $\bar{\alpha}$ 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, $P_{loss} \propto \frac{1}{4\pi d^n}, 2 \leq n \leq 4$.

An estimate for $P_{loss}\bar{\alpha} \approx 70$ 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 4 shows the probability of bit error plotted against the output power of the transmitter for an uncoded signal.

Since using a power amplifier alone is highly inefficient, forward error correction (FEC) is applied to the data to decrease the probability of error. However, some additional processing energy must be expended. The additional energy, denoted by E_{dsp}, to both encode and decode the data will need to be considered. Additional energy cost will also be incurred during the communication of the message since the length of each frame will increase. If the raw transmission rate R remains the same, then both the transceiver and output amplifier will be on for a longer duration.

Many types of error-correcting codes can be used to improve the probability of bit error. In this paper, we will only consider convolutional codes with varying coding rates. The upper bound on P_b can be determined by applying

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

where d represents the Hamming distance between some path in the trellis decoder and the all-zero path, $\{\beta_d\}$ are the coefficients of the first derivative of the transfer function $T(N, D)$ with respect to N, $\{P(d)\}$ are the first-event error probabilities, and d_{free} is the minimum free distance [11].

Figure 4 shows the P_b for different code rates and varying constraint lengths. These results were obtained using MATLAB simulation. Note that the probabilities shown assumes the use of a hard decision Viterbi decoder at the receiver.

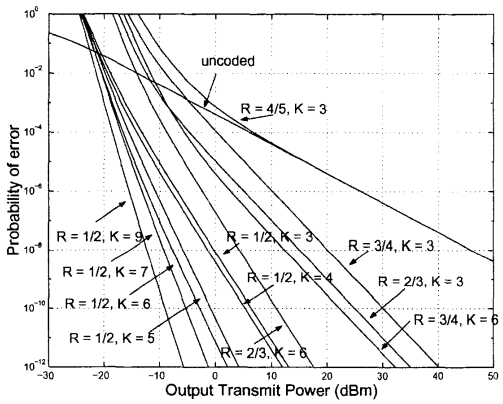

Figure 4. The probability of bit error of several different rate convolutional codes plotted versus the transmit power for the radio described in Section 2.3. $P_{loss} = 70$ dB, $N_{rx} = 10$ dB, and $R = 1$ Mbps. The number of information bits is 10000. Results were generated using MATLAB.

3.2. Energy Consumption of Coding

While the bounds on the performance of convolutional codes can be theoretically determined as shown above, the energy required to encode and decode will vary depending on the underlying architecture. In addition, since convolutional encoding of a bit stream will increase the size of the packet by approximately $1/R_c$, L will also increase, thereby increasing 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) as

$$E = P_{tx}(T_{on-tx} + T_{startup}) + P_{out}T_{on-tx} + E_{dsp}^{(e)} + P_{rx}(T_{on-rx} + T_{startup}) + E_{dsp}^{(d)} \quad (2)$$

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 total energy per information bit, $E_b = E/L'$.

In general, for convolutional codes, the energy required to encode data is negligible. However, performing Viterbi decoding on the StrongARM using C can be computationally-intensive and is likely to be energy-intensive[2]. We have measured the energy per useful bit required to decode rate $1/2$ and $1/3$ convolutional codes

[2] By optimizing and rewriting the code in assembly, it is possible to gain a factor of 5 to 10 reduction in energy consumption.

on the SA-1100. We denote the average decode energy per information bit as $E^{(d)}_{dsp,b}$. The energy to decode $1/2$-rate codes is shown in Figure 5. From our measurements,

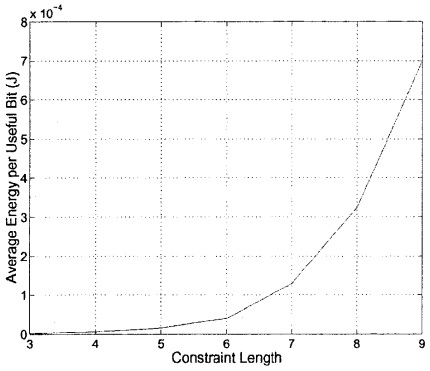

Figure 5. Measured decoding energy per useful bit for $R_c = 1/2$ codes with 3 to 9

we were able to make a couple of key observations. 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. We also observed that the energy consumption seems to be independent of the 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 power consumption since the number of states in the Viterbi decoder is unaffected. 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 5, we can now determine which convolutional code to use to minimize the energy consumed by communication for a given probability of error. In Figure 6, the total energy per information bit E_b is plotted against P_b. Figure 6 shows that the energy per bit using no coding is lower than that for coding. The reason for this result is that the energy of computation, i.e. decoding, dominates the energy used by the radio for the channel we have described in Section 3.1. For example, assuming the model described in (2) and $P_{out} = 0$ dBm, the communication energy to transmit and receive per useful bit for an $R_c = 1/2$ code is 168 nJ. Even if we assume $P_{out} = 20$ dBm, the communication energy per bit is merely 366 nJ. On the other hand, the energy to decode an $R_c = 1/2, K = 3$ code on the SA-1100 is measured to be 2.2 μJ per bit with a latency of 64 ms. These results imply that using the StrongARM to perform error correction coding is extremely inefficient[3].

[3]Note that the x-axis of the graph extends below 10^{-9}. For such low probabilities of error, these results may not be entirely valid. We only show the results below 10^{-9} so that the different codes can be distinguished.

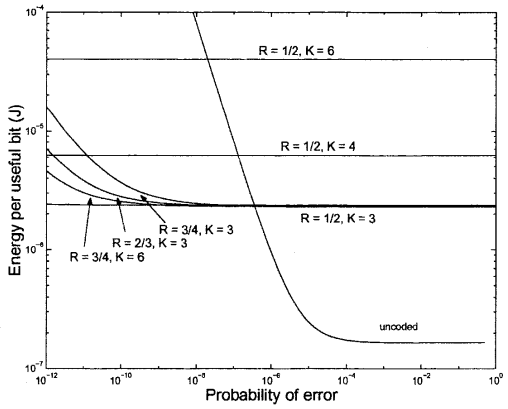

Figure 6. The energy per useful bit plotted against P_b of an uncoded signal and a few convolutional codes with different rates and constraint lengths. ($P_{loss} = 70$ dB, $N_{rx} = 10$ dB, $R = 1$ Mbps). The number of information bits is 10000. The decoder is implemented in C on a SA-1100.

Since the use of the StrongARM to perform Viterbi decoding is highly energy inefficient, a dedicated integrated circuit solution to perform decoding is preferred. To explore the power characteristics of dedicated Viterbi chips, we implemented one-half rate Viterbi 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 in which a separate add-compare-select (ACS) unit is used for each state. In addition, the designs use the one pointer traceback method of implementing the survivor path registers. Using Synopsys Power Compiler, we estimated the energy per bit used by our designs during the decoding of twenty thousand encoded information bits. Figure 7 shows the energy per bit for various constraint lengths. Using our implementation, in addition to our radio model, we can determine the minimum energy code to use for a given probability of error. In Figure 8, the total energy per information bit E_b is plotted against P_b. From the graph, it is clear that the energy used by the dedicated Viterbi decoder is insignificant compared to that of the radio tranceiver energy. In this case, the use of any convolutional code has very little impact on the energy consumption of communication. Furthermore, the benefits gained from the use of most of the convolutional codes exceed the energy costs in hardware; thus, coding is always recommended. Figure 8 provides the additional insight that high rate codes use less energy per bit despite requiring higher output transmit power (Figure 8) than lower

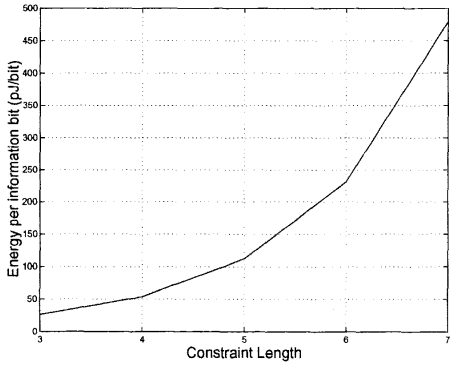

Figure 7. Measured decoding energy per useful bit for $R_c = 1/2$ codes with 3 to 7 using our synthesized VLSI implementation.

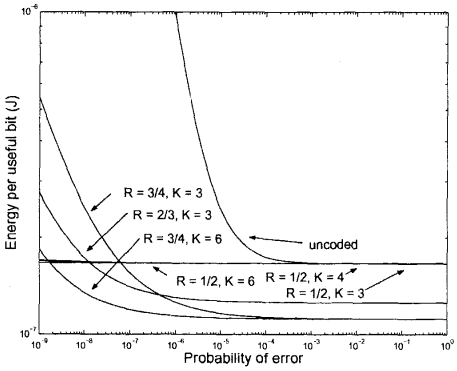

Figure 8. The energy per useful bit plotted against P_b using no coding and various convolutional codes. ($P_{loss} = 70$ dB, $N_{rx} = 10$ dB, $R = 1$ Mbps). The number of information bits is 20000.

rate codes for the same probability of bit error. This is primarily due to the fact that high rate codes use less bits during transmission than $1/2$-rate codes.

4. Conclusion

In this paper, we have demonstrated how to minimize the communication energy required for point-to-point communication. In particular, we considered how to adjust the transmit power and the underlying convolutional coding algorithm in order to achieve a desired probability of error using minimal energy. If a microprocessor (SA-1100) is used to implement the Viterbi algorithm, it is generally better to use no coding since the decoding energy is enormous. On the other hand, the use of a dedicated Viterbi decoder can lower the decoding energy per information bit by up to five orders of magnitude. As a result, sensor data can be encoded using a convolutional code to allow for lower output transmit power.

5. Acknowledgements

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. The authors also thank Manish Bhardwaj, Frank Honore, and Travis Simpkins for their help in estimating the power consumption of the ASIC Viterbi decoder.

References

[1] K. Bult et al. Low Power Systems for Wireless Microsensors. In *Proc. ISLPED '96*, pages 17–21, August 1996.

[2] A. Chandrakasan et al. Design Considerations for Distributed Microsensor Systems. In *Proc. CICC '99*, pages 279–286, May 1999.

[3] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen. Low-power CMOS digital design. *IEEE Journal of Solid-State Circuits*, 24(4):473–484, April 1992.

[4] C. Chien, M. B. Srivastava, R. Jain, P. Lettieri, V. Aggarwal, and R. Sternowski. Adaptive Radio for Multimedia Wireless Links. *IEEE Journal on Selected Areas in Communications*, 17(5):793–813, May 1999.

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

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

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

[8] B. Narendran, J. Sienicki, S. Yajnik, and P. Agrawal. Evaluation of an Adaptive Power and Error Control Algorithm for Wireless Systems. In *IEEE International Conference on Communications (ICC '97)*, 1997.

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

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

[11] J. Proakis. *Digital Communications*. McGraw-Hill, New York City, New York, 4th edition, 2000.