System Software Techniques for Low-Power Operation in Wireless Sensor Networks

Prabal K. Dutta and David E. Culler
Computer Science Division
University of California, Berkeley
Berkeley, CA 94720
{prabal,culler}@cs.berkeley.edu

***Abstract*—**
The operation of wireless sensor networks is fundamentally constrained by available energy sources. The underlying hardware determines the power draw of each possible mode of operation. System software attempts maximize the use of the lowest possible modes of each of the subsystems. This tutorial paper describes the system software techniques used at several levels. At the application sensing level, this includes duty-cycling, sensor hierarchy, and aggregation. At the communication level, it includes low-power listening, communication scheduling, piggybacking, post-hoc synchronization, and power-aware routing. At the node OS level, it includes event driven execution with split-phase operation and cooperative power management interfaces. At the lowest level, it includes managment of primary and secondary energy storage devices coupled with intelligent charge transfer scheduling. All of these aspects must be integrated in a systematic software framework.

I. Introduction

WIRELESS sensor networks ("sensornets") represent a new computing class consisting of large numbers of resource-constrained nodes called *motes* [12], [19], [42] which are often embedded in their operating environments [38], [53], [59], distributed over wide geographic areas [7], [17], [47], or located in remote and largely inaccessible regions [4], [27], [57]. Sensornets are enabling previously impossible applications but since they are often battery-powered – typically by a pair of AA alkaline batteries that can supply 3 volts at 2000mAhr – sensornet operations are fundamentally constrained by energy availability. The four main ways in which nodes consume energy are sensing, communication, computation, and storage [11], [12], [34], [39]. Each of these processes consumes a different amount of energy for each unit of useful work that it performs. Table I presents the startup latency and power draw characteristics of a typical sensornet node. The relevance of startup latency is discussed in detail in Section II.

Energy constraints, coupled with the need for longer lifetimes, have led to a variety of techniques for modeling [50] and managing energy consumption. While abstractions that are consistent over the diversity of power management techniques remain elusive [31], several effective techniques have emerged [9]. The most common techniques include:

- **Duty-cycling:** Cycling power to a subsystem to reduce its average power draw.

TABLE I
THE STARTUP LATENCY FROM THE OFF STATE TO THE LABEL STATE AND POWER DRAW AT 3VDC AS A FUNCTION OF THE GIVEN SUBSYSTEM AND STATE FOR THE EXTREME SCALE MOTE ("XSM") [12]. LPL IS AN ACRONYM FOR LOW-POWER LISTEN [40].

Subsystem	State	Startup Time	Power
Acoustic	off	–n/a–	3 μW
Acoustic	on	$<$ 1 ms	1.73 mW
Magnetic	off	–n/a–	3 μW
Magnetic	on	41 ms	19.4 mW
Infrared	off	–n/a–	3 μW
Infrared	on	$>$ 1000 ms	0.88 mW
Processor	sleep	–n/a–	30 μW
Processor	active	0.2 ms	24 mW
Radio	off	–n/a–	3 μW
Radio	receive	2.5 ms	24 mW
Radio	transmit	2.5 ms	48 mW
Radio	LPL	2.5 ms	411 μW
Buzzer	off	–n/a–	3 μW
Buzzer	on	–n/a–	45 mW

- **Batching:** Buffering multiple operations and executing them in a burst in order to amortize a high startup or overhead cost.
- **Hierarchy:** Ordering (boolean) operations by their energy consumption and invoking low-energy operations before high-energy ones when the desired result is a conjuction of the operations.
- **Redundancy reduction:** Reducing or eliminating redundancy through compression, aggregation, or message suppression.

Duty-cycling and hierarchical sensing are commonly used to lower the power consumption of sensors. Duty-cycling, when applied to sensing, follows a *sleep-wakeup-sample-compute-communicate* cycle in which nodes spend the majority of their time sleeping [43], [54]. The hierarchical model of sensing uses low-power sensors to trigger the operation of high-power sensors. These techniques are discussed in Section II.

Communications provide a plethora of opportunities for low-power operation. Duty-cycling the radio is a very effective technique for reducing energy consumption. However, duty-cycling leads to more complex communication patterns that include polling [40] and scheduling [14], [22], [29] the channel. Using a low-power [48] or zero-power [16] secondary radio to trigger the main radio upon channel activity and dynamically adjusting radio parameters [49] have been proposed as well.

Other techniques include buffering packets into packet trains to amortize the channel acquisition costs [41], dynamically adjusting transmission power [25], routing packets through nodes that have significant energy reserves [33], piggy-backing control messages and snooping on on application traffic [58]. Section III reviews some of these techniques.

At the computation level, operating system support for low-power operation is possible. TinyOS [20], [21] uses an event-driven execution model in which all computation occurs in response to internal or external events. The operating system powers down the processor between events, reducing or nearly eliminating wasted energy due to an idle processor. These techniques are possible due to fast processor wakeup times [43]. To support application-cooperative power management, software components implement interfaces that are invoked by the operating system [31]. Section IV reviews these techniques.

Energy-efficient storage techniques are also possible due to the differences in access times and power profiles of RAM, EEPROM, and Flash. RAM buffers allow energy-efficient manipulation of data which can then be persisted onto more power-hungry EEPROM and Flash memory for durable storage. Buffering sensor samples in RAM before logging to Flash allows compressed data to be written to the log. As flash memory becomes cheaper, archiving data at the sensor nodes and communicating data only when queried becomes a viable model of data collection [15]. Section V presents these techniques.

Some applications require longer lifetimes than is possible with single-use batteries. If frequent battery replacement is not a viable solution, then it is possible to equip such nodes with energy harvesting systems. However, such systems require management of the harvesting devices, the primary and secondary storage devices, and the transfer of charge between these devices. These considerations are discussed in Section VI.

II. Sensing

Sensors can account for a significant proportion of the power budget and unless their operation is managed carefully, can render inconsequential any gains from low-power operation in the other parts of the system. Duty-cycling is an effective and commonly used technique to lower the rate of energy consumption. Hierarchy is sometimes used in conjunction with or in place of duty-cycling and works by sampling low-power sensors first and higher-power sensors only if the low-power sensors indicate that turning on the higher-power sensors would provide additional information. Aggregation of sensor values across space, time, or both is a form of redundancy reduction that allows fewer radio messages to communicate the same amount of information.

A. Duty-Cycling

Duty-cycling is a general and broadly applicable technique so we review it in some depth. Although our focus is on sensors, the central ideas can be applied to other subsystems. Duty-cycling lowers the *average* power consumed by a sensor by cycling its power on and off. The duty-cycle period, T_{DC}, is the sum of the on-time, T_{on}, and the off-time, T_{off}.

$$T_{DC} = T_{on} + T_{off} \tag{1}$$

The duty-cycle, DC, is the ratio of T_{on} and T_{DC}.

$$DC = \frac{T_{on}}{T_{on} + T_{off}} \tag{2}$$

The range of T_{on} and T_{off} values are constrained by several factors. The startup latency, $T_{startup}$, of a sensor is the amount of time required for the sensor to stabilize after power is applied. The acquisition time, $T_{acquire}$, of a sensor is the amount of time required to acquire a sample. For analog sensors, this is the time to perform an analog-to-digital conversion while for digital sensors, this is the time required to read the data over a digital bus.

Many data collection applications collect sensor readings of physical phenomena that change at low frequencies [54]. Duty-cycling under this model is relatively simple and follows a *sleep-wakeup-sample-compute-communicate* cycle in which nodes spend the majority of their time sleeping. For such applications, T_{on} is constrained.

$$T_{on} \geq T_{startup} + T_{acquire} \tag{3}$$

The duty cycle period, T_{DC} is set equal to the desired sampling interval. Computing the average power with duty cycling is straightforward.

$$Power_{avg} = \frac{1}{T_{DC}} \int_{T_{DC}} Power(t)dt \tag{4}$$

In constrast with data collection, exceptional event detection attempts to detect *rare*, *random*, and *ephemeral* events. The events of interest are usually parameter changes in ambient signals with spectra ranging from 1 Hz to 5 kHz and expected event durations, $\mathcal{E}[T_{event}]$, of a few seconds [12]. Naïve duty cycling, as described above, must be augmented with additional constraints to be usable for exceptional event detection.

Obviously, T_{off} must be less than $\mathcal{E}[T_{event}]$ to ensure that the sensor is on during the event. Indeed, the sensor must be turned on long enough to *detect* the event. This period, T_{detect}, usually consists of N consecutive sensor samples taken at a sampling frequency, f_s.

$$T_{detect} = (N-1)/f_s \tag{5}$$

Multiple samples are used to improve the signal-to-noise ratio or ensure enough data is available for a block operation like an FFT. The minimum on time must be increased by T_{detect}.

$$T_{on} \geq T_{startup} + T_{acquire} + T_{detect} \tag{6}$$

Since T_{on} is the time needed to detect an event, including the overhead of powering up the sensor and acquiring the samples, T_{off} must be short enough to ensure at least one complete T_{on} period occurs during the event. Hence, T_{off} is constrained as follows.

$$T_{off} \leq \mathcal{E}[T_{event}] - 2 \times T_{on} \tag{7}$$

The factor of two in Equation 7 ensures that the sensor is fully operational for at least T_{detect}.

There are two important cases in which duty-cycling does not sufficiently lower power consumption. The first case is when the inequality in Equation 7 cannot be satisfied because an event is *too* ephemeral, and the subsystem must be powered continuously. The second case occurs when $Power_{avg}$ is close to or in excess of the power budget and duty-cycling does not lower the average power to an acceptable level.

B. Hierarchy

When duty-cycling alone fails to meet the system power budget, arranging sensors in a hierarchy with one sensor triggering another allows power consumption to be further lowered. Two approaches which implement hierarchy are power-based query optimization [36] and passive vigilance [12].

TinyDB [35], a database that presents a sensornet through a declarative SQL-like interface, uses power-based query optimization [36] to transforms a declarative query into a query plan that will yield the lowest overall power consumption. Consider the query shown in Figure 1.

```
SELECT accelerometor, magnetometer
FROM   sensors
WHERE  accelerometer > c1
AND    magnetometer  > c2
SAMPLE INTERVAL 1s
```

Fig. 1. A TinyDB SQL query.

This query can be implemented with at least three possible query plans: sampling the accelerometer and magnetometer before applying the selections, sampling the magnetometer and applying its selection first, or sampling the accelerometer and applying its selection first. The Analog Devices ADXL202 accelerometer [2] draws 1.8 mW and can be sampled in 0.1 ms on the Atmel ATMega128 processor [3] used in the Mica2 mote [8]. The Honeywell HMC1002 magnetometer [23] draws 15 mW and can also be sampled in 0.1 ms. If the selectivity of the accelerometer and magnetometer selections is high, then clearly the first query plan will be the most expensive, the second query plan slightly better, and the third query plan the best since it likely will not even need to power up the relatively power-hungry magnetometer.

Passive vigilance arranges sensors in a directed acyclic graph, called a trigger network, in which lower-power sensors trigger higher-power sensors, which in turn trigger computationally expensive signal processing and data fusion algorithms, which eventually trigger radio transmission [12]. Consider the trigger network in Figure 2.

In this trigger network, *wake* represents a simple threshold based passive infrared wakeup sensor which turns on passive infrared signal processing algorithm labeled *pir* when triggered. If the output of the *pir* stage is true, then the microphone, *mic*, magnetometer, *mag*, and sensor fusion algorithms, *fuse*, are powered on or invoked. If the output of the fusion algorithm is true, only then does a radio transmission, *tx*, occur.

C. Aggregation

Many sensornet applications require the ability to extract data from the network and often the data consists of summaries (or aggregations) rather than then raw sensor readings. Because aggregation is central to sensornet applications, [37] has argued that it should be provided as a core service by the system software and has presented the design of an in-network aggregation service called the Tiny AGgregation (TAG) service. TAG allows users to express simple, declarative queries and have them distributed and executed in-network. Nodes are arranged in a tree, receive data from their children, and pass on partially aggregates to their parents.

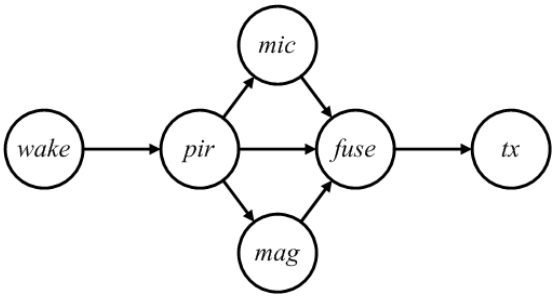

Fig. 2. Trigger network. $X \rightarrow Y$ denotes X triggers Y.

This type of in-network aggregation is generally far more communications-, and hence energy-, efficient than computing aggregates centrally. The TAG aggregates `COUNT`, `MIN`, `AVERAGE`, and `HISTOGRAM` generally transmit far fewer bytes than their centralized versions while the `MEDIAN` and `COUNT DISTINCT` aggregates transmit the same or slightly fewer number of bytes, respectively. The actual performance is, of course, dependent on the network topology. TAG provides no benefit over a one-hop, centrally aggregated network but does provide a substantial benefit over a n-hop line of n nodes that are centrally aggregated (n messages for TAG versus $n^2/2$ messages in the central case).

III. Communication

Communications is an essential, but power-hungry, aspect of sensornet operation. Energy-efficient operation requires careful management of all devices and layers participating in communications. Techniques as diverse as duty-cycling the radio, buffering packets into packet trains to amortize the channel acquisition costs, and routing algorithms that forward packets to neighbors with the greatest remaining battery life are employed.

A. Radio Management

The radios [5], [6], [45] commonly used in sensor nodes can consume a significant proportion of the system power budget when operated continuously. Several techniques have emerged to reduce radio energy consumption. *Polled* operation is the simplest technique and works by sampling the channel periodically and powering down the radio between samples. *Scheduled* operation works by coordinating in advance when radios may (or may *not*) transmit and receive, which allows the radios to be powered down during periods of scheduled inactivity. *Triggered* operation uses a low-power or zero-power secondary radio to signal a more capable, but also more power-hungry, main radio to wake up.

1) Polled Operation: Polled radio operation, sometimes called low-power listening [10], [19], [40], periodically samples the channel for radio activity and powers down the radio between successive samples. The power savings from this technique depends on the radio startup time, the channel sample time, and the sampling period. Smaller startup and sample times improve the achievable efficiency, so they are criticial radio parameters that constrain system operation. These parameters are usually determined at system design time, but can also be adjusted by system software [40]. Duty cycles of 1-2% are common, resulting in similar adjustments to the power draw.

In contrast, system software can control the sampling period, allowing for an energy-latency-channel capacity tradeoff. The degree of the tradeoff is again dictated by the radio startup and sample times. The effective power consumption of the radio using this scheme is equal to the duty-cycle of the radio (the percentage of time the radio is powered). To be useful, polled operation requires the ability to efficiently detect channel activity, where efficiently means the energy cost of sampling the channel for activity is significantly less than the cost of packet reception. Practically, a radio must be able to detect channel activity within a few frame symbol periods (e.g. a bit, byte, or preamble period).

The Mica family of sensor nodes, which use the RF Monolithics TR1000 radio [45], employ slow, periodic sampling to detect the preamble and then increase the sampling frequency to perform start symbol detection [19]. Since the Mica radios can sample the channel very quickly – on the order of a bit period – and frame preambles are several bytes in length, this approach leads to significant power savings on the receive side. In this scheme, a packet preamble must be longer than the slow sampling period to be detected. Preamble length increases with the channel sample period so the energy savings that the receiver realizes is actually an additional energy burden placed upon the transmitter. However, for lightly-loaded networks with relatively few transmissions, this technique incurs relatively small energy cost for the transmitter and provides considerable benefit for the receiver.

Low-power listen can reduce channel capacity, perhaps significantly, depending on the radio startup and channel sample time [10]. For a given duty-cycle level, a longer startup or sample time implies a longer sample period, which in turn requires a longer preamble, which in turn lowers the useable channel capacity. The three generations of Berkeley motes illustrates this point [8], [19], [42].

2) Scheduled Operation: Most radios draw a substantial fraction of the transmit power when the radio is on and receiving nothing. In sensornets, a device will only be transmitting for short periods of time but must be listening more often in order to forward data for the surrounding nodes. Scheduled radio operation coordinates communications activity between neighboring nodes by establishing time slots when a node may or may not transmit.

One way to reduce the cost of idle listening (i.e. listening when no radio transmission is occurring) is through *periodic listening*. By creating time periods when it is illegal to transmit, nodes need only listen part of the time [10]. This type of scheduled sleeping, which is really a form of duty-cycling, is also used in S-MAC [60] to conserve energy. The energy-efficiency of these protocols can be estimated using the duty cycle ratio presented in Equation 2.

In contrast with periodic listening, which schedules illegal time slots, legal time slots can be scheduled as well. The IEEE 802.15.4 standard [56] defines a MAC layer time division multiple access (TDMA) protocol for scheduled operation. Network-layer protocols like FPS [22] can have access to multihop flow information that allows them to schedule radio activity. TDMA can be used for communications scheduling on a (perturbed) grid [29]. Applications like TinyDB, which have *a priori* knowledge of communication patterns, can implement their own scheduling policies and, for example, turn off the entire network stack when the network is expected to be inactive [37]. The main drawback of scheduled operation is the energy cost and complexity of establishing and maintaining the schedules and synchronized clocks.

3) Triggered Operation: Triggered operation uses a low-power secondary radio to signal a more capable, higher-power main radio to wake up, with the system software responsible configuring and coordinating the activities of the radios. Bluetooth radios have been used as secondary radios for 802.11b radios [1]. Unfortunately, the idle power draw of the *secondary* radio is 40mW – more than the *main* radio in most sensornet nodes. Customized secondary radios called "Mini Bricks" were presented as part of a "Wake on Wireless" infrastructure scheme [48]. The Mini Bricks use the same RFM TR1000 [45] radios found in the Mica motes [19], so the power draw is substantially lower than Bluetooth radios – 7mW receive and 8mW transmit.

Zero-power secondary radios have been proposed that, like crystal radios, operate by coupling the RF signals using simple detector circuits [16]. The output of the detector circuit is connected to an interrupt line on the processor and asynchronously interrupts the processor upon radio activity. This idea, while promising, has not been reduced to practice.

B. Middleware Services

Several sensornet middleware services like time synchronization, routing, and dissemination have power awareness integrated into their design or operation.

1) Time Synchronization: Many sensornet applications require events to be timestamped so that they can be correlated with other events or an external frame of reference. Time synchronization services can help address this need by establishing the temporal ordering of events (X happened before Y) and real-time issues (X and Y happened within a certain interval) [13], [46]. Timesync also may be used to coordinate future actions at two or more nodes (X, Y, and Z will all happen at time T).

Proactive time synchronization algorithms that attempt to maintain continuously synchronized clocks through the use of periodic messages consume energy, often unnecessarily, by sending messages every few seconds or tens of seconds. Many common data collection and event detection applications do not require a proactively maintained global timebase and can

instead use reactive or post hoc time synchronization in which event times are conveyed after the fact [30]. Such post-hoc schemes do not waste any energy unnecessarily synchronizing clocks. Post-hoc synchronization requires MAC-level timestamping of packets on transmit and receive as well as an API layer that makes using post-hoc synchronization simpler.

2) Routing: Routing packets through nodes that have greater energy reserves than their neighbors is a technique used to prolong network lifetime and share the routing load equitably [33]. These routing algorithms incorporate the remaining energy of nodes into the routing metric. Forwarding messages along routes that would result in the minimum *expected* number of transmissions, rather than shortest path for example, optimizes for energy [58].

3) Dissemination: Dissemination is the process of propagating data to a large number of of nodes. Flooding is a naïve approach for dissemination in which every node transmits each unique message exactly once. Flooding has many problems including unnecessary transmissions, which wastes power, and excessive channel contention, which results in many nodes not receiving the message. More effective and energy-efficient algorithms for dissemination have been proposed for sensornets. Sensor Protocols for Information via Negotiation ("SPIN") uses meta-data to reduce redundant transmissions and uses knowledge of the resources available to nodes, which allows energy-efficient dissemination [18].

Trickle, using techniques from the epidemic/gossip, scalable multicast, and wireless broadcast literature, implements a "polite gossip" policy for propagating and maintaining code updates in a sensornet [32]. Trickle periodically broadcasts a summary of a node's data to local neighbors. However, this broadcast is inhibited if the node has recently received a summary identical to its own. When a node receives a summary that older than its own, the node broadcasts its own summary. Instead of flooding a network with packets, the algorithm controls the transmission rate so each node receives a small trickle of packets which allow it to stay up to date. The Trickle algorithm is used in the Deluge bulk data transfer protocol [24] which itself serves as the default TinyOS network programming system.

C. Miscellaneous Optimizations

Both the PAMAS [51] and S-MAC [60] medium access control layers conserve energy by powering off the radio in nodes that are not actively participating in ongoing communications in a node's neighborhood. PAMAS and S-MAC decide whether a node's radio can be powered off after receiving a packet's destination address and checking whether the node is an intended recipient.

Snooping on application-level packets allows network layers to obtain information about neighbors, channel activity, and packet transmission yields without originating packets of their own, which saves energy. Similarly, piggybacking control information on application traffic allows packet transmission overhead to be shared [58].

Batching message transmission can help amortize the cost of channel acquisition. MAC layers that implement low-power listening, like B-MAC [40], have to transmit lengthy preambles to ensure the receiver detects the packet. If multiple packets can be sent in a "packet train" after the long preamble, then the cost of the preamble can be spread across all of the packets, thereby reducing the average energy consumed per packet.

IV. Computation

Most sensor network applications are neither CPU nor I/O bound, so a system's processor and peripherals are often idle. Since many systems are built around microcontrollers and peripherals that feature low-power sleep states and the ability to wakeup quickly from these states, it is possible to put the processor and peripherals to sleep when the system is idle. For example, the Atmel ATMega128 [3] used in the Mica2 mote [8] can wake up in 180μs [42] while the Texas Instruments MSP430F1611 processor [55] used in the Telos mote [42] can wake up in 6μs. In comparison, the M25P80 serial Flash memory used in the Telos mote requires between 1 ms and 10 ms to wake up [52]. TinyOS [20], [21], [31] utilizes these fast wakeup times to provide operating system-level power management in an application-transparent manner.

Program execution in TinyOS is initiated in response to *events* and *tasks*. Event sources include timer, analog-to-digital converter completion, and communication device interrupts. Tasks are a form of deferred computation that can be initiated or *post*-ed from an event handler or another task. Pending tasks are placed in a queue and execute on a strictly first-come-first-serve basis with run-to-completion semantics. Task processing can be preempted by events. However, if the event posts a task, that task is placed at the back of the task queue and its execution is deferred until the preempted task, and all other tasks in the queue, complete.

When TinyOS's task queue is empty, the system goes into a low-power state until the next interrupt. Typically, this involves putting the processor into a sleep state. In addition to the processor, the system can put other peripherals to sleep as well. TinyOS provides the `StdControl` interface, shown in Figure 3, to support such system-initiated power management.

```
interface StdControl
{
  // Init component and subcomponents.
  command result_t init();

  // Start component and subcomponents.
  command result_t start();

  // Stop component and subcomponents.
  command result_t stop();
}
```

Fig. 3. The TinyOS 1.x StdControl interface. Note: the TinyOS 2.x startup and power management interfaces are different.

All components that require initialization or can be powered down should provide the `StdControl` interface. The TinyOS system uses this interface as follows. On system boot, the `StdControl.init` function of every module is called. The

`StdControl.start` function indicates that the module, and any subsystems that the module encapsulates, should be powered up. Subsequently, whenever TinyOS's task queue is empty, the system calls `StdControl.stop` on each module to indicate that the system is going into a low-power sleep state. This call gives application modules or device drivers an opportunity to place peripherals into a power-save mode.

V. Storage

Traditional memory hierarchies exist in computer systems for reasons of performance, cost, and persistance. While this remains true for sensornets, energy-efficiency also presents a case for employing a memory hierarchy. RAM, EEPROM, and Flash are the most common memory devices in use but they have vastly different read, write, and erase latencies; they offer different access granularities; they are orders of magnitude different in size; and they consume different amounts of power to perform their operations. These feature differences mean that buffering data in RAM and batching read, write, and erase operations can be more energy-efficient than immediate operations on EEPROM or Flash.

Earlier TinyOS platforms [19] had driver support for RAM-based buffering of reads and writes to EEPROM. Some current TinyOS services like Deluge [24] use use a simple form of application-controlled paging in which EEPROM or Flash pages are read or "paged-in" to RAM buffers for manipulation and then written or "paged-out" to EEPROM or Flash. This simple technique of buffering pages and batching writes can be more efficient than per-byte operations because the overhead of writing can be amortized over the entire page. For example, the Berkeley Telos mote [42] uses the M25P80 8Mbit serial flash [52]. The M25P80 has a typical page program cycle time, t_{PP}, of $0.4 + n/256$ ms, where n is the number of bytes to be written. This translates to a batched write time of 1.4 ms for 256 bytes versus 104 ms if the same 256 bytes are written individually – a 74x difference. If the M25P80 is powered down between write cycles, then the batched writing efficiency becomes even more dramatic. Since the time from power-up to write, t_{PUW}, ranges from 1 to 10 ms, batched writing provides a time improvement ranging from 149x (2.4 ms vs 358 ms) to 233x (11.4 ms to 2662 ms), respectively.

Flash memory, driven by consumer demand and Moore's Law, is becoming less expensive and more energy-efficient. This trend is changing the tradeoffs between storage and communications in sensornets. Just a few years ago, data collection applications chose to communicate to sensor readings rather than store them locally because it was more energy-efficient [39]. In the future, as storage becomes increasingly more energy-efficient, particularly when compared with communications, data collection applications might archive data at the sensor nodes, index the data centrally, and disseminate queries to access data dynamically. Under this model, sensornets would become a federation of tiny databases [15].

VI. Energy Harvesting

Some applications require longer lifetimes than is possible with single-use batteries. If frequent battery replacement is not a viable solution, then it is possible to equip such nodes with energy harvesting systems [26], [44]. However, such systems require management of the harvesting devices, the primary and secondary storage devices, and the transfer of charge between these devices [28]. It has been our experience that exposing this logic to the application, and allowing the application to adapt its duty cycle to the available power, as suggested by [26], makes writing applications difficult. We believe additional research is needed to incorporate energy harvesting into the system software and abstract it from applications.

VII. Conclusions

Sensornet lifetime is severly constrained by the available energy sources so naïve and power-oblivious operation is not appropriate. A number of system software techniques have been proposed to extend the lifetime of sensornet nodes including duty-cycling subsystems, batching operations, leverating power hierarchis, and reducing redundancies. System software techniques are more general, and hence have broader applicability than application-specific techniques, but application-specific techniques can be still more effective. Since the particular mix of sensing, communications, computation, and storage is usually quite application-specific, with different applications demonstrating dramatically different mixes, power profiles can vary widely. System software can provide only general solutions to power management but if the system can expose appropriate power management interfaces or implement power optimizations, as many of the techniques presented in this paper do, then applications can influence or control system components and reduce power without having to be explicitly power-aware.

References

[1] Y. Agarwal, C. Schurgers, and R. Gupta, "Dynamic power management using on-demand paging for networked embedded systems," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC 2005)*, Shanghai, China, Jan. 2005, pp. 755–759.

[2] Analog Devices, Inc., *Low-cost +/-2g Dual-Axis Accelerometer with Duty Cycle Output*, Oct. 2000. [Online]. Available: http://www.analog.com/UploadedFiles/Data_Sheets/53728567227477ADXL202E_a.pdf

[3] Atmel Corporation, *ATmega128(L) Complete Datasheet*, Nov. 2004. [Online]. Available: http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf

[4] V. Bokser, C. Oberg, G. Sukhatme, and A. Requicha, "A small submarine robot for experiments in underwater sensor networks," in *Symposium on Intelligent Autonomous Vehicles*, July 2004.

[5] Chipcon AS, *CC1000: Single Chip Very Low Power RF Transceiver*, Apr. 2004. [Online]. Available: http://www.chipcon.com/files/CC1000_Data_Sheet_2_2.pdf

[6] ——, *CC2420: 2.4 GHz IEEE 802.15.4 /ZigBee-ready RF Transceiver*, June 2004. [Online]. Available: http://www.chipcon.com/files/CC2420_Data_Sheet_1_2.pdf

[7] S. Coleri, S. Y. Cheung, and P. Varaiya, "Sensor networks for monitoring traffic," in *Forty-Second Annual Allerton Conference on Communication, Control, and Computing*, Univ. of Illinois, Sept. 2004.

[8] Crossbow Technology, Inc., *MPR/MIB Mote Hardware Users Manual*, Apr. 2005. [Online]. Available: http://www.xbow.com/Support/Support_pdf_files/MPR-MIB_Series_Users_Manual.pdf

[9] D. Culler, P. Dutta, C. T. Ee, R. Fonseca, J. Hui, P. Levis, J. Polastre, S. Shenker, I. Stoica, G. Tolle, and J. Zhao, "Towards a sensor network architecture: Lowering the waistline," in *HotOS X: Tenth Workshop on Hot Topics in Operating Systems*, Santa Fe, NM, 2005.

[10] D. E. Culler, J. Hill, P. Buonadonna, R. Szewczyk, and A. Woo, "A network-centric approach to embedded software for tiny devices," in *EMSOFT 2001:First International Workshop on Embedded Software*, Oct. 2001, pp. 114–130.

[11] L. Doherty, B. Warneke, B. Boser, and K. Pister, "Energy and performance considerations for smart dust," *International Journal of Parallel Distributed Systems and Networks*, vol. 4, no. 3, pp. 121–133, 2001.

[12] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, and D. Culler, "Design of a wireless sensor network platform for detecting rare, random, and ephemeral events," in *Proceedings of the Fourth International Conference on Information Processing in Sensor Networks (IPSN '05)*.

[13] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," in *Proceedings Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002)*.

[14] S. C. Ergen and P. Varaiya, "Tdma scheduling algorithms for sensor networks," in *submission*, July 2005.

[15] D. Ganesan, B. Greenstein, D. Perelyubskiy, D. Estrin, and J. Heidemann, "Multi-resolution storage and search in sensor networks," *ACM Transactions on Storage (To appear)*, Aug. 2005.

[16] L. Gu and J. Stankovic, "Radio triggered wake-up capability for sensor networks," in *Real-Time Applications Symposium (RTAS'04)*, May 2004.

[17] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, G. Zhou, J. Hui, and B. Krogh, "Vigilnet:an integrated sensor network system for energy-efficient surveillance," *ACM Transaction on Sensor Networks (In submission)*, 2004.

[18] W. R. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in *Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking*. ACM Press, 1999, pp. 174–185.

[19] J. Hill and D. Culler, "Mica: A wireless platform for deeply embedded networks," *IEEE Micro*, vol. 22, no. 6, pp. 12–24, 2002.

[20] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister, "System architecture directions for networked sensors," in *Architectural Support for Programming Languages and Operating Systems*, 2000, pp. 93–104, tinyOS is available at http://webs.cs.berkeley.edu. [Online]. Available: citeseer.nj.nec.com/382595.html

[21] J. Hill, R. Szewczyk, A. Woo, P. Levis, K. Whitehouse, J. Polastre, D. Gay, S. Madden, M. Welsh, D. Culler, and E. Brewer, "Tinyos: An operating system for sensor networks," 2003, submitted for publication.

[22] B. Hohlt, L. Doherty, and E. Brewer, "Flexible power scheduling for sensor networks," in *Proceedings of the Third International Symposium on Information Processing in Sensor Networks*, Berkeley, CA, Apr. 2004.

[23] Honeywell, *HMC1001/HMC1002/HMC1021/HMC1022: 1 and 2-axis Magnetic Sensors*, 2003. [Online]. Available: http://www.ssec.honeywell.com/magnetic/datasheets/hmc1001-2_1021-2.pdf

[24] J. W. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," in *The 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys'04)*, 2004.

[25] J. Jeong, D. Culler, and J.-H. Oh, "Empirical analysis of transmission power control algorithms for wireless sensor networks," in *submission*, 2005.

[26] X. Jiang, J. Polastre, and D. Culler, "Perpetual environmentally powered sensor networks," Apr. 2005.

[27] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.-S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet," in *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS X)*, 2002.

[28] A. Kansal, D. Potter, and M. B. Srivastava, "Performance aware tasking for environmentally powered sensor networks," *SIGMETRICS Perform. Eval. Rev.*, vol. 32, no. 1, pp. 223–234, 2004.

[29] S. S. Kulkarni and M. Arumugam, "Tdma service for sensor networks," in *In Proceedings of the Third International Workshop on Assurance in Distributed Systems and Networks*, Mar. 2004.

[30] B. Kusý, P. Dutta, P. Levis, M. Maróti, A. Lédeczi, and D. Culler, "Elapsed time on arrival: A simple, versatile, and scalable primitive for time synchronization services (to appear)," vol. International Journal of Ad hoc and Ubiquitous Computing, no. 1, p. 2, 2006.

[31] P. Levis, S. Madden, D. Gay, J. Polastre, R. Szewczyk, A. Woo, E. Brewer, and D. Culler, "The emergence of networking abstractions and techniques in tinyos," in *Proceedings of the First USENIX/ACM Symposium on Network Systems Design and Implementation (NSDI)*, 2004.

[32] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A self-regulating algorithm for code maintenance and propagation in wireless sensor networks," in *Proceedings of the First USENIX/ACM Symposium on Network Systems Design and Implementation (NSDI)*, 2004.

[33] Q. Li, J. Aslam, and D. Rus, "Online power-aware routing in wireless ad-hoc networks," in *In Proceedings of the Seventh Annual International Conference on Mobile Computing and Networking (ACM Mobicom'01)*, July 2001, pp. 16–21.

[34] S. Madden, "The design and evaluation of a query processing architecture for sensor networks," Ph.D. dissertation, U.C. Berkeley, 2003.

[35] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "Tinydb: An acquisitional query processing system for sensor networks," *ACM Transactions on Database Systems (TODS)*, vol. 30, no. 1, pp. 122–173, Mar. 2005.

[36] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "The design of an acquisitional query processor for sensor networks," in *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*. New York, NY, USA: ACM Press, 2003, pp. 491–502.

[37] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks," in *Proceedings of the Fifth ACM Symposium on Operating System Design and Implementation (OSDI 2002)*.

[38] J. Paradiso, J. Lifton, and M. Broxton, "Sensate media - multimodal electronic skins as dense sensor networks," *BT Technology Journal*, vol. 22, no. 4, pp. 32–44, Oct. 2004.

[39] J. Polastre, "Design and implementation of wireless sensor networks for habitat monitoring," Master's thesis, University of California at Berkeley, 2003.

[40] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *In Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys'04)*, Nov. 2004.

[41] J. Polastre, J. Hui, P. Levis, J. Zhao, D. Culler, S. Shenker, and I. Stoica, "A unifying link abstraction for wireless sensor networks," in *Proceedings of the Third ACM Conference on Embedded Networked Sensor Systems (SenSys 2005)*.

[42] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling ultra-low power wireless research," *The Fourth International Conference on Information Processing in Sensor Networks: Special track on Platform Tools and Design Methods for Network Embedded Sensors (IPSN/SPOTS)*, Apr. 2005.

[43] J. Polastre, R. Szewczyk, C. Sharp, and D. Culler, "The mote revolution: Low power wireless sensor network devices," in *Hot Chips 16: A Symposium on High Performance Chips*, Stanford University, Aug. 2004.

[44] V. Raghunathan, A. Kansal, J. Hsu, J. Friedman, and M. B. Srivastava, "Design considerations for solar energy harvesting wireless embedded systems," Apr. 2005.

[45] RF Monolithics, Inc., *TR1000: 916.50 MHz Hybrid Transceiver*, 1999. [Online]. Available: http://www.rfm.com/products/data/tr1000.pdf

[46] K. Romer, "Time synchronization in ad hoc networks," in *MobiHoc 2001*, June 2004.

[47] C. Sharp, S. Schaffert, A. Woo, N. Sastry, C. Karlof, S. Sastry, and D. Culler, "Design and implementation of a sensor network system for vehicle tracking and autonomous interception," in *Second European Workshop on Wireless Sensor Networks*, Jan. 2005.

[48] E. Shih, P. Bahl, and M. J. Sinclair, "Wake on wireless: An event driven energy saving strategy for battery operated devices," in *Proceedings of the Eighth Annual ACM Conference on Mobile Computing and Networking*, Altanta, Georgia, USA.

[49] E. Shih, S. Cho, F. S. Lee, B. H. Calhoun, and A. Chandrakasan, "Design considerations for energy-efficient radios in wireless microsensor networks," *The Journal of VLSI Signal Processing-Systems for Signal, Image, and Video Technology*, vol. 37, no. 1, pp. 77–94, 2004.

[50] V. Shnayder, M. Hempstead, B. rong Chen, G. Werner-Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," in *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys'04)*, Nov. 2004.

[51] S. Singh and C. S. Raghavendra, "Pamas: power aware multi-access protocol with signalling for ad hoc networks," *SIGCOMM Comput. Commun. Rev.*, vol. 28, no. 3, pp. 5–26, 1998.

[52] ST Microelectronics., *M25P80: 8 Mbit, Low Voltage, Serial Flash Memory With 40MHz SPI Bus Interface*, Aug. 2005. [Online]. Available: http://www.st.com/stonline/books/pdf/docs/8495.pdf

[53] R. Szewczyk, A. Mainwaring, J. Polastre, and D. Culler, "An analysis of a large scale habitat monitoring application," in *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys'04)*, Nov. 2004.

[54] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin, "Application driven systems research: Habitat monitoring

with sensor networks," *Communications of the ACM Special Issue on Sensor Networks*, vol. 47, no. 6, June 2004.

[55] Texas Instruments, *MSP430F1611 Mixed Signal Microcontroller Datasheet*, Mar. 2005. [Online]. Available: http://focus.ti.com/lit/ds/symlink/msp430f1611.pdf

[56] The Institute of Electrical and Electronics Engineers, Inc., "Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)," Oct. 2003.

[57] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh, "Monitoring volcanic eruptions with a wireless sensor network," in *Proceedings of the Second European Workshop on Wireless Sensor Networks (EWSN'05)*, Jan. 2005.

[58] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in *Proceedings of the First International Conference on Embedded Networked Sensor Systems (SenSys)*, 2003.

[59] N. Xu, S. Rangwala, K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, "A wireless sensor network for structural monitoring," in *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys'04)*, Nov. 2004.

[60] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *Proceedings of IEEE Infocom*, 2002.