CA-TSL: Energy Adaptation for Targeted System Lifetime in Sparse Mobile Ad Hoc Networks

Pei Zhang, *Member*, *IEEE*, and Margaret Martonosi, *Fellow*, *IEEE*

Abstract—With the proliferation of mobile devices, an increasing number of sensing applications are using *mobile* sensor networks. These mobile networks are severely energy-constrained, and energy usage is one of the most common causes of failure in their deployments. In these networks, nodes that exhaust their energy before the targeted system lifetime degrade system performance; nodes that run past the system lifetime cannot fully utilize their stored energy. Although much work has focused on policies to reduce and regulate energy usage in fixed and dense networks, intermittently connected networks have been largely overlooked. Due to variations in hardware, software, node mobility, and environment, it is especially difficult for intermittently connected mobile networks to improve operations *collectively* in a dynamic environment. Here, we present and evaluate Collaborative Adaptive Targeted System Lifetime (CA-TSL), an adaptive policy that enforces a system-wide targeted lifetime in an intermittently connected system by adapting node energy usage to an estimated desired energy profile. For evaluation, we present both real-system and large-scale simulated results. Our approach improves sink data reception by an average of 50 percent, and an additional 30 percent when a density estimation technique is also employed. In addition, it reduces system lifetime variations by up to $5.5\times$.

Index Terms—Mobile sensor networks, energy budgeting, intermittently connected networks, recharged networks, parameter estimation.

1 Introduction

Sensor nodes, especially sparse mobile sensor nodes, are subject to highly unpredictable and varied conditions that can significantly affect their energy profile. Many external factors, such as weather conditions, unknown movement patterns, and inherent low-level hardware variations, can affect a node's energy consumption and charging patterns. These variations create performance variations, and thus, energy consumption variations. This results in premature system performance degradation when nodes exhaust their energy before the intended end of the system deployment, or when nodes waste energy by lasting beyond the desired lifetime of the deployment.

A predictable system lifetime allows a deployment to be properly planned and maintained. Battery or node replacement can be planned as part of deployment planning, and thereby, reduce deployment cost. In addition, the system will have improved performance during deployment time.

To enhance not just *node*, but also *system* performance, system variations need to be reduced or controlled. However, coupled with the intermittent connectivity of a sparse network, these variations make it extremely difficult to predict and control the energy profile of one node, and seemingly impossible to predict the behavior of *many* nodes in order to meet a particular system performance goal.

Past deployments of both fixed [6], [21], sparse mobile sensor networks [25], and our experience with ZebraNet [30] have shown that energy is one of the most common failure points. The system functionality is degraded as nodes lose power due to unpredictable lifetimes. These deployments also make clear that merely reducing energy consumption is insufficient. Accurate projections of system lifetime are needed for properly planning deployment maintenance and data gathering. In addition, well-balanced energy usage within the system improves system functionality by maintaining more nodes up to the system lifetime. Thus, our goal is to improve the *predictability* of system lifetime, as well as improve node performance.

While there are adaptive methods that can reduce variations in node performance, and hence, variations in lifetime, they rely on real-time node collaboration [25], [30]. Hence, they require extensive real-time communication through the entire network, and therefore, are not feasible in intermittently connected systems. Furthermore, most methods focus on extending, rather than achieving, a given system lifetime. In contrast, our goal is to enable nodes to target a particular *global* system lifetime efficiently, *locally*, and independently of unpredictable variations in a variety of network situations. We discuss related work in more detail in Section 6.

This paper presents the collaborative adaptive targeted system lifetime (CA-TSL), an efficient and effective node-level distributed policy aimed at sparse Delay-Tolerant Networks (DTNs). CA-TSL also allows nodes to make real-time estimates of system-wide parameters as well as adapt

- *P. Zhang is with the Department of Electrical Engineering, Carnegie Mellon University, Silicon Valley Campus, 23-11 NASA Research Park, Moffett Field, CA 94035. E-mail: peizhang@cmu.edu.*
- *M. Martonosi is with the Department of Electrical Engineering, Princeton University, Engineering Quad B216, 34 Olden Street, Princeton, NJ 08544-5263. E-mail: mrm@princeton.edu.*

Manuscript received 25 Mar. 2009; revised 18 Jan. 2010; accepted 28 Feb. 2010; published online 21 July 2010.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2009-03-0101. Digital Object Identifier no. 10.1109/TMC.2010.138.

based on node level and precalculated information. In order to estimate global parameters in sparse and intermittently connected DTNs (such as system energy and system density) in real time, the system uses prediction, selection, and combinations of opportunistic neighboring information to make system-wide estimates. The system then uses the estimates to adjust a node's energy consumption dynamically, thereby achieving a fixed global system lifetime even in *very sparse*, intermittently connected sensor networks, such as the real-world deployment of ZebraNet [30].

In particular, CA-TSL makes the following contributions:

- Adapts to a specified system lifetime to the assumptions of the designer.
- Accurately predicts system parameters (such as energy and density) in intermittently connected networks.
- Improves data-to-sink packet delivery by up to 50 percent without prediction of system parameters, and an additional 30 percent with such prediction.
- Reduces data delay by more than a factor of 2 for intermittently connected mobile nodes due to predictive energy usage.

This paper is organized as follows: Section 2 provides the details of our system. Section 3 shows results from implementation of CA-TSL on ZebraNet hardware in an intermittently connected situation. Section 4 shows extensive Monte Carlo simulation results for charged and uncharged networks based on parameters measured in ZebraNet nodes under various situations. Section 5 discusses related work, and Section 6 gives our conclusions.

2 System Description

This section gives an overview of CA-TSL. Our goal is to let the overall, possibly disconnected, mobile sensor network collaboratively agree on and target a particular system lifetime. CA-TSL is designed to be a flexible approach, responding both to node-level and system-level energy variations that make the lifetime of mobile networks unpredictable. While this collaborative technique can apply to other aspects of the system, this paper focuses on targeting a predetermined lifetime, especially important in our application of ZebraNet [30]. In addition, it is designed to be easily implementable and to function in a variety of networks, particularly very sparse, intermittently connected, mobile sensor networks.

Fig. 1 shows the overview of CA-TSL, which provides information to the scheduling or routing layer on each node. Designers summarize their lifetime and energy expectations using the global expected energy usage curve (Section 2.1). A node's energy budgeting module uses this input and the measurement of a node's available energy to determine the energy budget (Section 2.2). The energy controller then schedules the node's applications using the energy budget as a reference to change the schedule through lower layers or through application support (Section 2.3). The actual energy consumed by the application, combined with other uncontrollable variables, in turn, affects the current node energy. Finally, predictive system parameters are used to adjust the expected usage curve to adapt to current system environments (Section 2.4).

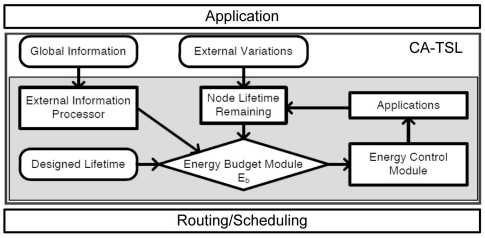

Fig. 1. Block diagram of CA-TSL, which provides information to the scheduling or routing layer on each node. With a globally defined energy expectation (Section 2.1), the node budget (Section 2.2), and then control the schedule through the application or the routing layers (Section 2.3). Global information is processed to improve understanding of its surroundings (Section 2.4).

Below are its major characteristics:

- The policy allows each node to automatically adjust to obtain a fixed system lifetime by adapting to predefined system energy usage target. This fixes the target system lifetime while reducing the variation of node lifetimes around the system lifetime.
- By using a shared energy usage target, this policy limits the need to flood node information through the system and thereby reduces overhead. We discuss the overhead of our system in Section 3.2.
- In response to the policy suggestions, each local node adjusts its energy usage in a variety of ways under its local control. These include adjusting how often it communicates data with other nodes, varying how often it turns on sensing and other devices, or other energy adjustments. It works along with other energy-saving and energy-balancing algorithms, such as routing and radio optimizations. This property allows the policy to be implemented easily in existing systems, and it allows additional improvements to be developed for different hardware peripherals offer opportunities for energy control.
- The policy uses local feedback to achieve a global goal and adapts to an expected usage curve defined by the designer, regardless of topology or density. This allows the designer to explicitly control the deployment energy usage and properly plan the deployment.
- The policy only requires hardware to monitor total energy and infers other information from system peripherals. This allows easy implementation on many platforms.

Fig. 2 shows an example of the collaborative adaptive targeted system lifetime (CA-TSL) controlling system communications. It shows two nodes with different amount of energy both trying to follow the energy profile and accumulating energy units that cannot be used.

2.1 Expected Energy Usage Curve

In order for a sensor network to achieve a long operational lifetime, the node policy must have a global view. Yet, for a very sparse, intermittently connected, mobile network, it is impossible for individual nodes to achieve this through communications alone. Therefore, in order for distributed nodes to achieve a targeted global lifetime, all nodes target a

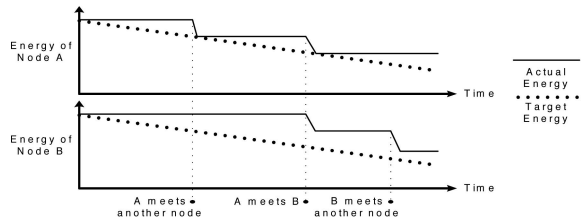

Fig. 2. CA-TSL policy adapting to an energy usage curve. In this example, node A meets another node first and uses its allocated energy. Then, node A meets node B, and node A uses all its allocated energy while node B does not. Later, Node B uses its energy with another node.

designer-defined expected energy usage curve, which defines the expected time-varied system energy usage. Curves could be picked at design time or can be collaboratively adjusted based on a global predefined algorithm as the system runs. The system designer defines the basis of this curve in advance by basing it on the expectation of the particular sensor application. CA-TSL then automatically modifies it by incorporating predicted global parameters.

A variety of energy curves can be defined, depending on the application. Fig. 3 shows examples of various cases. To allow the network to achieve consistent performance throughout the system lifetime, a linear expected usage curve (Fig. 3a) with starting point equal to the initial energy budget can be used. In charged systems, a constant (Fig. 3b) can be used. In periodically charged systems, a sawtooth curve (Fig. 3c) can be used to allow for different charge rates during day and night. Curves need not be linear. The introduction of an expected energy curve simplifies the system design for different applications, since only this curve needs to be changed and the system will adapt. This centralizes energy use goals of the system and allows for the stable estimated global parameter incorporation described later.

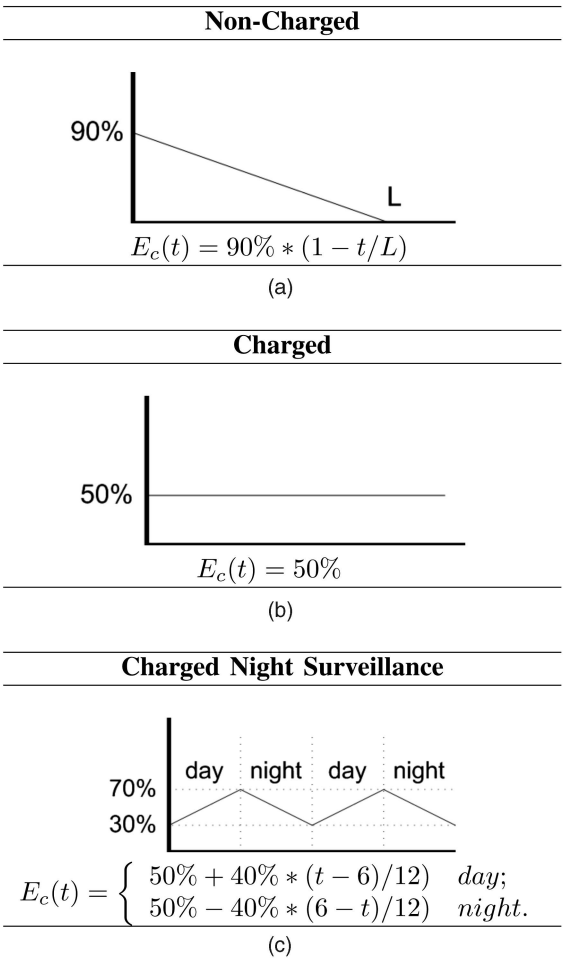

Fig. 3. Some possible options for the expected energy usage curve; L is the targeted lifetime for noncharged systems. (a) Expected usage for a noncharged system. (b) Simplistic usage for a charged system. (c) Expected usage for a solar-charged system.

2.2 Energy Budgeting Module

CA-TSL's energy budgeting module runs and allocates energy on each node during each time frame. The example shown in Fig. 2 demonstrates the energy units allocated in each time frame, based on the difference between the expected energy curve and the actual node's current energy.

Deployed sensor networks can exhibit large energy variations, even within the same deployment. These large variations make it difficult for many control algorithms to react quickly and maintain stability. To mitigate this problem, we implement a step feedback method, where several different thresholds set the quantity of change in energy budget. With this method, when the node's current energy state exceeds certain thresholds around the expected energy usage, the node sets its allowable energy budget corresponding to that threshold. The current energy state, that corresponding to the difference from the desired energy level, is calculated as shown in (1):

$$E = E - E_c(t) + I_{app}. \tag{1}$$

Here, E is the energy measured, $E_c(t)$ is the function describing the expected energy reserve at a given time, and I_{app} is the application importance input described later.

Budgeted energy E_b is discontinuous and is calculated by (2).

$$E_b = E_o \sum_{i=0}^{N} u(E - E_l(i)); \quad u(a) = \begin{cases} 0, & a < 0; \\ 1, & a > 0. \end{cases} \tag{2}$$

Here, E_o is the maximum total amount of energy that all applications can consume in a given time frame, $u(a)$ is the step function, $E_l(i)$ is the threshold energy for each energy level, and N is the maximum number of energy levels.

The energy budget module is the decision center for the control loop of CA-TSL (Fig. 1). This one-cycle control method has some characteristics of both on/off control and proportional control. While providing smoother feedback than on/off control, it does not exhibit stability issues related to continuous proportional control. In addition, with $max(E_l(i)/2)$ away from the optimal energy budget, it allows a closer approximation to the optimal energy budget than does simple on/off control. Furthermore, when the energy status changes, this approach gives a rapid transition to the optimal operating range with negligible over- or undershoot. With N set to a low number, this calculation is quick, only consisting of addition and if statements.

Fig. 4. Two methods to vary the energy profile: (left) duration modification, where length of operation is modified; (right) rate modification, where some operations are skipped depending on available energy.

Fig. 5. External information processor: after each time period, the system goes through a prediction phase, a filter phase, and then an update phase to incorporate all new information acquired from its communication with neighbors, if there are any.

CA-TSL requires knowledge of the current node's energy level in order to determine the energy budget; a battery gauge is used for this purpose. While battery gauges have been known to be inaccurate, their performance in this respect has recently improved. After initial battery pack calibration, some lithium ion battery gauges are able to maintain an accuracy as high as 99 percent even after months of use [22].

2.3 Energy Control Module

During each time frame, the energy control module receives a energy budget; it then controls the energy usage of the node it resides on using two control methods, shown in Fig. 4, depending on the environment and the node design. One method is duration management, which modifies the length of time for which a peripheral can execute. For example, the radio is given different maximum bandwidths in order to alter its energy profile. Another method is operation rate modification, which modifies how often the application runs [2], [10]. Both of these methods apply to typical sensor node peripherals such as radios and sensors. We use these methods in our experiments in Sections 3 and 4. Our framework is general enough, however, to employ other energy modification approaches offered by other hardware.

Multiple applications can be present on each node, such as GPS data gathering, other sensing, and radio communication. Within each time frame, the control module runs multiple applications based on a predetermined schedule for serialized applications. The energy budget in this case is used to determine the number of times the application will run in each time frame. The controller uses a fixed ratio to share energy between applications that run in parallel, but priority-based schemes are also possible. Each fraction of the energy budget is then used to determine the run schedule of the application as in the serial case.

Variations in the network can prevent nodes from using up their energy budget. For example, in the communication module, nodes may not always have neighbors to communicate with and thereby use their energy. To account for this, unused energy in CA-TSL nodes is rebudgeted, and so it accumulates in each budget cycle until it can be used. The actual communication window for the radio application is also limited by the minimum window between the sender and the receiver. In addition, this method also places the overhead on the higher energy node. Nodes that have more energy will turn on and attempt to discover neighbors, while nodes with lower energy do not turn on. This ensures fairness by minimizing energy consumption by lower energy nodes.

2.4 System Parameter Estimation and Integration

In this section, we first discuss the delay-tolerant collaborative technique used in the external information processor to estimate system parameters in general (Fig. 1), and then we use this technique in the next section to provide an example of density estimation and its integration into CA-TSL. Although many applications can take advantage of system information, we focus on producing an automatic energy target profile of CA-TSL.

The delay-tolerant collaboration technique is designed especially for sparse intermittently connected networks, such as DTNs. Devices in these networks rarely encounter neighbors, making continuous direct communication among them impossible. Furthermore, the movement of these devices is often unpredictable and uncontrollable, so communications cannot be prescheduled. Despite these issues, it is desirable to obtain a best effort method for aggregating knowledge about the network's parameters, as well as offering confidence levels for our estimates. This is accomplished by not only storing information received from previous encounters but also providing measurements to predict parameter changes actively. Using this technique, as shown in Fig. 5, a device/node waits for a period of time before updating predictions of the current estimates of system parameters. Then the nodes search for neighbors in order to obtain new information, and this information is then filtered for independence from the current predictions. Finally, the information is used to update the current predictions. Each of these steps is described in more detail in the following paragraphs. In addition, an implementation of CA-TSL is presented in Section 2.5.

The *prediction* maintains the estimation for the device during long periods of disconnected operation. It extends the usefulness of the received information beyond the immediate reception time and enables delayed collaboration. In this phase, the device makes a new prediction of the parameter based on its current value and its sensor readings. When a neighboring device with new information comes into range, the system enters the filter phase.

The *transform* prevents repeated incorporation of the same or outdated information when nodes remain in communication range for several communications. This phase prepares the new information for inclusion into the new estimate. When a neighbor is found and new information is received, the filter compares it with previously received information and determines whether it is new information, replicated information, or an update of previous information (time stamp and ID). Repeated information is removed, while new and update information is labeled and passed to the update phase.

The *update* reduces estimation error that accumulates during the prediction phase. This phase adds the new information to existing estimations, and thereby, improves the latter. While different applications can implement this phase differently, they should all combine newer information based on their likelihood of accuracy.

2.5 Parameter Estimation Example

This delay-tolerant collaborative technique can be applied in many situations involving system parameters such as system energy, node density, sensor calibration, and data routing. Application to system energy and density estimation is presented below and evaluated in Section 4.3.

2.5.1 Global Energy Estimation

Global system energy usage varies with respect to time. Below we present the detail of the steps of how CA-TSL estimates global energy.

Prediction. In the delay-tolerant collaborative implementation, the prediction phase for system power estimation is accomplished by tracking the rate of battery discharge. A node tracks its average battery discharge rate by averaging the 100 most recent energy usage measurements, which are recorded after each communication attempt. When neighboring nodes are found, energy levels and average discharge rates are also exchanged. During the prediction phase, nodes simply reduce the energy of each observation according to the predicted rate.

Transform. If a neighbor is found, the nodes exchange their local logs of energy estimations and discharge rates. The filter phase processes each received energy log. The log's discharge and energy information is filtered to ensure that only the more recent information from each node is passed to the update phase, whereas repeated and outdated entries are discarded.

Update. The update phase takes the data from previously unseen nodes and incorporates them into the log. In addition, it takes new update information from previously seen nodes in order to replace the existing entry. To restrict system memory usage, the energy log records the energy and the discharge rate for the 10 most recent node measurements. Average system energy estimation is performed by averaging the energy levels of the log entries. Furthermore, each entry is weighted by the reciprocal of the time elapsed since its measurement.

2.5.2 Global Density Estimation

CA-TSL energy estimation functions not only estimate global energy usage but also the likelihood of using energy with global density estimations. For communication or other functions that require the radio, as the system density decreases, the likelihood of using the allocated energy for communication becomes low. If the system statically allocates energy usage as the node enters a sparse area, there will tend to be a large amount of energy left unused at the end of the desired lifetime. Information about the likelihood of energy use allows CA-TSL to predict expected delays in energy use and preallocate energy budgets given the estimated density. This is especially useful in data passing with the radio peripheral, as energy usage is limited by the existence of neighbors.

In density estimation, the prediction phase simply calculates the average of periods between previous encounters. When nodes encounter each other, they exchange density measured by the the neighbors and the transform phase will only reject old or duplicated data. Finally, the update phase updates the internal table for averaging.

2.5.3 Targeting Lifetime with Global Estimation

With collaborative estimations, the node can estimate the probability of meeting a neighbor as described above, and then estimate the likelihood of energy use, and finally, modify the expected use curve. In order to alter the curve based on this new information, we must first explore the relationship between density and system lifetime. Here, we assume a charged system with a constant expected energy. Lifetime in this context refers to the operational time that is required after charge is removed (e.g., overcast days for solar charging). This section thus provides a comparison between expected lifetime with density. The calculations here can easily be extended to other types of systems.

Following the steps described before, the nodes can collaboratively obtain the probability of meeting each other (p). Therefore, the probability of a given session having no connections is $1-p$ or q. Thus, the expected lifetime of any node is

$$E(lifetime) = time\ on\ reserve\ threshold + time\ on\ energy\ above\ threshold.$$

When the system falls below the reserve battery threshold (E_c), the system goes into conservation mode. This mode can support minimum communications and some data sampling. Since the discharge is low, we assume it to have a roughly constant discharge rate (R_s). Therefore, time operating on the reserve battery capacity is roughly fixed at E_c/R_s.

However, time operating on extra energy above the reserve threshold is not fixed. This is because at any time, this can be used by communication with an encountered neighbor. So this time is calculated by

$$p(N\ periods) = p * (1 + q + q^2 \cdots + q^N), \tag{3}$$

which simplifies to

$$p(N\ periods) = 1 - q^{N+1}. \tag{4}$$

Knowing the probability of connections, we can solve for the number of additional periods contributed by the energy above the threshold. Solving this equation requires the system to compute logarithms, which is processor intensive and unsuitable for most sensor network nodes. Therefore, an approximation can be used in order to reduce processing overhead. The expected value can be used to approximate the additional lifetime due to the extra energy available from infrequent connections:

$$E(extra) = p + 2 * p * q + 3 * p * q^2 + \cdots + (N+1) * p * q^N. \tag{5}$$

Here, N denotes the maximum time period considered for the expected extra lifetime. A closed form can be obtained by calculating

$$E(extra) - q * E(extra) = p * (1 + q + q^2 \cdots + q^N). \tag{6}$$

The right-hand side gives the form shown in (3). Solving this equation yields the results for (5), in closed form:

$$E(extra) = \frac{1 - q^{N+1}}{1 - q}. \tag{7}$$

Approximating the expected lifetime and setting this lifetime to infinity, we obtain

$$E(extra) = \lim_{N\to\infty} \frac{1-q^{N+1}}{1-q} = \frac{1}{p}. \quad (8)$$

This calculation requires only one division, and is further simplified by measuring the periods between connections instead of calculating the connection frequency (that is, measure $1/p$ directly). Therefore, no division is needed with this estimation.

However, the lifetime of this system is in fact finite, and there is a limit on how long the system can operate even without connection to other nodes due to power used in conservation mode. This is taken into account by adding a cap to the possible extra lifetime. Therefore, the true expected extra lifetime value is

$$E(extra) = min\left(\frac{1}{p}, \frac{100\% - E_c}{idlerate}\right), \quad (9)$$

and total lifetime is

$$E(lifetime) = \frac{E_c}{R_s} + min\left(\frac{1}{p}, \frac{100\% - E_c}{idlerate}\right). \quad (10)$$

Since the expected lifetime after charge disconnection ($\frac{E_c}{R_s}$) is constant, we can allow the system to adjust (E_c). Given the solution shown above, this method adjusts E_c, corresponding to changes of $\frac{1}{p}$, while keeping $E(lifetime)$ constant. By doing so, more battery capacity is allocated to communications as density decreases. This increased communication energy budget reduces the time during which the battery remains fully charged, allowing more energy to be scavenged. To understand the effectiveness of this process, we evaluate this estimation in Section 4.3.

3 CA-TSL: Real System Prototyping

To validate the CA-TSL policy, test its real in-system performance, and collect simulation parameters, we designed and implemented CA-TSL on ZebraNet nodes used in our second Kenyan deployments [26]. We estimate and adjust system lifetime by modulating software's use of the radio. The radio used is the XTend OEM RF Module [17], with in-system power consumption of 4.1 W in transmit mode, 0.4 W in receive mode, and 10 mW in idle mode. The system generates 32 packets of 64 bytes every two minutes. The second generation of ZebraNet nodes incorporates a battery module with a battery gauge BQ27000 [23] paired with a 2 amp-hour Li-ion battery; the battery gauge has an energy reading accuracy within 5 percent [23]. In this implementation, eight possible run schedules corresponding to different energy levels are managed by CA-TSL. In order to reduce the experimentation time, the expected lifetime of the system is set to 60 hours.

3.1 Periodically Connected Nodes

Our test example involves intermittent connectivity with four nodes in order to evaluate the general effectiveness of CA-TSL. In this experiment, nodes 2 and 4 were sensors that generate data, while nodes 1 and 3 were forwarders. The nodes were connected as shown in Fig. 6, with node 2 able to forward data through both nodes 1 and 3, while node 4 could forward data only through node 3. At start time, the system ran normally in steady state. Then node 3 was physically moved out of range, leaving node 4 without a communication link to the base station; node 4 then stored data to be sent later. Node 3 was restored after some time, and the backlogged data from node 4 flowed through it to the base station.

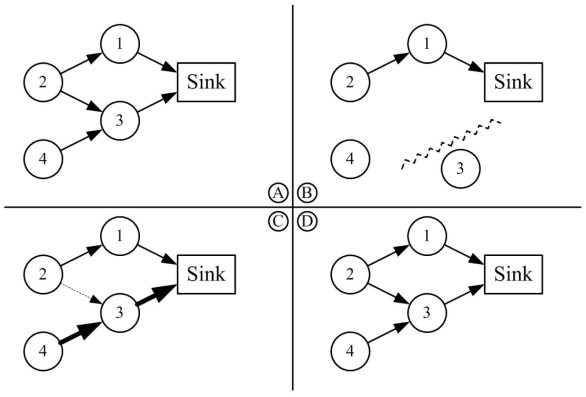

Fig. 6. System configuration diagram of periodically connected tests.

Fig. 7a shows the battery energy in each node. Node 1 and node 2 followed the desired discharge profile because they were able to maintain and adjust their communication. Energy for nodes 3 and 4 dropped at a much slower rate when node 3 was out of range because it had no useful way of using communication energy. At t_1, node 3 was moved away and energy stopped dropping. As expected, once node 3 was moved back at t_2, CA-TSL allocated more of the unused energy, to get the node back on the targeted lifetime schedule. Fig. 7b compares this to tuned-energy reduction (tuned-ER) in which a fixed schedule is selected to achieve the desired lifetime, Tuned-ER does not attempt to "catch up" on communication after a disconnection, and thus, transmits at a slower rate during this time [21], [30]. Fig. 7b shows that both methods accumulated data at the same rate but backlogged data moved quickly out of node 4 and finished data transfer at time t_3, compared to a finish at t_4 under tuned-ER.

CA-TSL greatly improves the transfer rate of backlogged data in intermittently connected systems. In this experiment, the maximum delay was reduced by more than $2\times$ with only eight energy levels. The backlogged delay would have decreased further in this experiment if the maximum bandwidth was higher. The opportunistic nature of CA-TSL is most beneficial for sparse mobile networks, where rare connections can be short-lived.

We feel that this implementation on ZebraNet hardware is indicative of the sparse intermittently connected mobile networks, and that this implementation can easily be ported to other platforms. The parameters collected from the hardware are used as simulation parameters for larger scale experimental results in the next section.

3.2 System Overhead

Because it is a local, low-overhead approach, CA-TSL only very modestly increases processor overhead. The adaptive

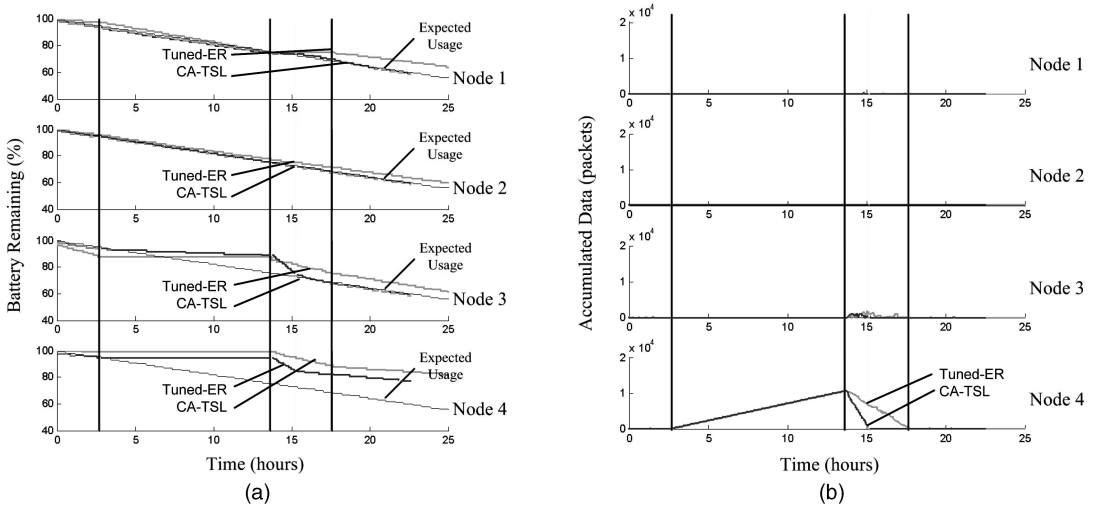

Fig. 7. Results for four periodically connected nodes when a critical node is moved away. The data show CA-TSL's fast data upload speed for an intermittently connected node. (a) Battery state of charge. (b) Accumulated data.

policy implemented in our experiments requires 238 bytes of code, 174 bytes of RAM for schedule generation, energy use curve generation, a 16-entry parameter table, and the adaptive policy code. Most of the code is run once when the system starts. At scheduling points, the system needs only to calculate the expected energy value, which takes 36 μs on a 4 MHz MSP430. This is a very small overhead even for a low-capability processor. In addition, parameter sharing requires an additional 2 bytes of transmission and is incorporated in the handshaking packets in this implementation.

4 CA-TSL Performance

A full evaluation of CA-TSL performance requires it to be evaluated at scale, and in a realistic environment. While the previous section demonstrates small-scale, real-system performance, this section presents results of large-scale simulation based on the parameters collected in the previous section. We focus on intermittently connected, mobile networks, for which it is difficult to predict the connection pattern and preset the nodes accordingly. Due to frequent changes in connections and neighbors, these systems are more susceptible to the unexpected energy profile variations that CA-TSL targets.

These Monte Carlo experiments were performed using the measured power consumption data from the ZebraNet nodes, which consume 4.1 W when the system is in transmit mode, 0.4 W in receive mode, and 10 mW when idle [30]. Simulated node behavior (energy, hardware, mobility, etc.) was based on observations of nodes in the field [26].

Our Monte Carlo experiments were run on a grid of 100×100 km with 10, 100, or 1,000 nodes. Each node had a radio communication range of 2 km, giving the average neighbor density shown in Table 1. We use both random walk and the zebranet model. At the beginning of each communication slot, each node used one packet for peer discovery. If no neighbors were found, the node shut off. During each communication session, the energy budget was translated into a packet count. This max-data-count was set to be the maximum sum of RX and TX packets. Nodes exchanged sensor data with their neighbors until the max-data-count was reached for the node with the least count, accounting for the energy of both the RX and TX nodes. Energy usage was normalized to the transmit energy usage of the ZebraNet node. Each node used one unit of energy to transmit a packet and 0.1 unit of energy to receive each packet [17]. The node with more energy was penalized by an additional 0.1 unit of energy to reflect the time required to time out after the other node had stopped communicating upon reaching the communication limit imposed by CA-TSL. In all these cases, data flowed opportunistically in the network and traveled only one hop per communication slot. Although other communication protocols exist for mobile networks, we focused here on flooding since our networks are sparse enough that other ad hoc routing techniques cannot provide a data rate significantly better than nonnetworked systems. However, since this technique is routing protocol independent, other protocols can be used.

Similar to the ZebraNet deployment, nodes have the opportunity to perform sensing and/or communication every eight minutes. The expected lifetime is set to 1,080 communication attempts or six days starting with sufficient energy to send 15,000 packets for noncharged systems with CA-TSL. Each solar-charged node has a reservoir, with maximum energy sufficient to send 1,500 packets. Any charging beyond the maximum energy level is lost. A node is considered dead when it does not have enough energy to

TABLE 1
Average Number of Neighbors per Communication Slot for Each Node in Our Sparse Network Experiments over a $100 \text{ km} \times 100 \text{ km}$ Field and Radio Range of 2 km

Number of Nodes	Movement Model	Neighbors per Communication
10	Random	0.02
100	Random	0.24
100	Zebra Movement	0.16
1000	Random	2.45

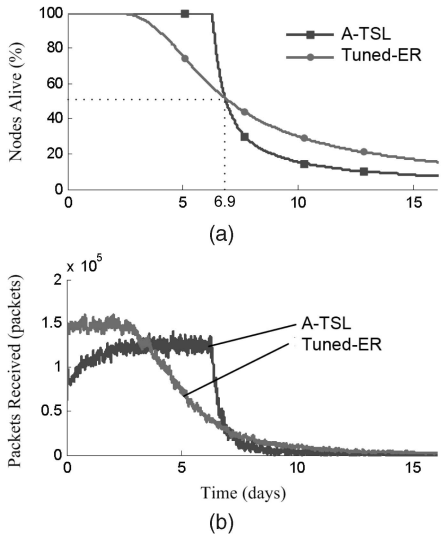

Fig. 8. Results for 100 nodes over a $100\ \mathrm{km} \times 100\ \mathrm{km}$ area, averaged over 100 runs. (a) Percentage of nodes operating. (b) Average packets transmitted. The data show CA-TSL reduced lifetime variability compared to the tuned method.

both transmit and receive one packet. In these experiments, we defined actual system lifetime as the time during which more than 50 percent of the nodes were operational. While in some systems 50 percent can still be considered sufficient, in our experiences with sparse deployments, this is generally the point at which systems can no longer function as expected.

In noncharged experiments, we selected a linear expected energy use curve, similar to Fig. 3a. Energy started at 95 percent and crossed 0 at the targeted lifetime of six days. In the system where batteries recharged continuously (for example, from solar cells), we set the lifetime to infinity. Each charged node used a discharge curve that was, on average, horizontal and piecewise linear due to day-night cycles, as shown in Fig. 3c. We selected these discharge curves because they were typical of the expectations that system designers usually have for real-world sensor networks.

Due to the lack of prior work on system-wide policies that can function at such low node densities, we compared CA-TSL with the tuned-ER method, which is widely used in today's system deployments [30], [21]. We also compared it to a collaborative method based on a price-bidding strategy, which entails communication overhead and manual tuning of price in these sparse systems. To test the effectiveness of this approach in a variety of system environments, we used both random walk with uniform distribution over the maximum zebra movement speed of 7 km per hour and also a movement model based on collected ZebraNet GPS traces [26].

Based on its energy, each node in the experiment could select from 16 possible communication bandwidths, each with a different max-data-count. The minimum max-data-count was 16 packets per communication slot and this increased by 16 packets at a time, up to a total of 256 packets per communication slot. With the tuned-ER method, in contrast, the nodes attempted to communicate every communication slot, but the packets per slot were fixed.

TABLE 2
Q1 and Q3 Represent the First and Third Quartiles

System Lifetime Range

(a)	Median	Q3 - Q1	Max - Min
Tuned-ER	6.9	0.50	2.0
CA-TSL	6.9	0.13	0.32

The table shows system lifetime and its ranges in days, and they indicate an approximately five times increase in the predictability of the system lifetime.

To compare fairly between the two methods, the max-data-count for tuned-ER was set at 62 packets per communication slot. This led to an average system lifetime similar to that of CA-TSL.

4.1 Noncharged Mobile Networks

In this first experiment, we evaluated the performance of CA-TSL on battery-operated mobile nodes. These nodes had an initial energy supply and could not be recharged. This is often the case in many mote-based or mobile phone-based networks. We first simulated an ad hoc mobile network of 100 nodes using the simulation parameters just presented. With an average of only 0.24 neighbors, this network is not dense enough to have a reliable link to the data sink. Instead, data are transmitted to the sink via opportunistic pairwise data transfers. We used the random walk mobility model in the first experiment to simulate a purely random memory less movement pattern.

4.1.1 Variation Reduction

CA-TSL is compared first with tuned-ER, where nodes have a fixed bandwidth that is tuned over multiple runs to give the desired performance. The graph in Fig. 8a shows that both CA-TSL and tuned-ER achieved the same system lifetime of 6.93 days. CA-TSL, however, retained almost all of its nodes up until the end of the system lifetime. This allowed its nodes to achieve better success rates in data delivery than tuned-ER. The graph in Fig. 8b shows the data transmitted over the course of the experiment. The system bandwidth for tuned-ER was greatest at the beginning. CA-TSL, in comparison, had a more uniform system bandwidth throughout its targeted lifetime. At the maximum, the CA-TSL approach had approximately 40 percent more nodes operating than tuned-ER, resulting in better connectivity.

Next, we examined variations between different randomized runs of the system under the same conditions. This experiment shows the likelihood that a particular deployment will have a lifetime close to the target. Table 2 shows the resulting system lifetime from 100 simulations. While the two methods are designed to yield the same median lifetime, 50 percent of CA-TSL lifetimes had a range of 0.13 days and 100 percent had a total range of 0.32 days, whereas tuned-ER gave 0.5 and 2.01, respectively. The range in lifetime for CA-TSL was lower than that of tuned-ER by a factor of 5. This variance reduction is by design: we wanted to define system lifetime more accurately, and hence, plan better for deployment. Table 3 shows the number of packets the sink received at the end of the system lifetime; the number of packets shown was normalized to

TABLE 3
Q1 and Q3 Represent the First and Third Quartiles

Packets Received (Normalized)

(b)	Min	Q1	Median	Q3	Max
Tuned-ER	0.46	0.83	1.0	1.2	1.8
CA-TSL	0.69	1.3	1.6	1.8	2.4

For packets received by the sink entries in the table, values are normalized to the median of tuned-ER. The table shows a roughly 50 percent improvement in median packet reception.

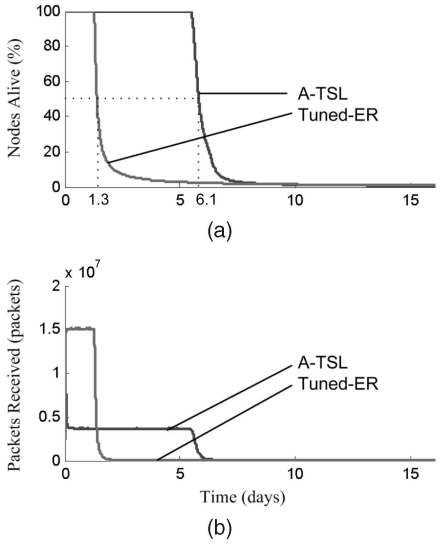

Fig. 9. Results for 1,000 nodes over a $100\ \mathrm{km} \times 100\ \mathrm{km}$ area, averaged over 100 runs. (a) Percentage of nodes operating. (b) Average packets transmitted. The data show CA-TSL greatly reduced lifetime variability to varying node densities when compared with the tuned method.

the median number of packets received in the tuned-ER case. Due to the enhanced system connectivity, CA-TSL increases the amount of data received at the sink by more than 50 percent in nearly all quartiles.

4.1.2 High-Density Experiment

We also explore cases where node density cannot be determined beforehand, for example, deployments that are affected by such logistical factors as multiround deployments. We simulated 1,000 nodes with an average of 2.45 neighbors per communication (a $10\times$ increase in node density compared to the previous experiment), and we ran both CA-TSL and tuned-ER with the same parameters as described above. Fig. 9 shows operation with 1,000 nodes. The tuned-ER lifetime was greatly reduced due to the increase in the amount of sensing data from the 900 additional nodes. The top graph shows that the lifetime of the tuned-ER method reduced 5.5 days with a mean lifetime of 1.3 days compared to the 100-node experiment. In contrast, with CA-TSL, lifetime changed by only 0.8 day to 6.1 days, an improvement of more than $5.5\times$ compared to tuned-ER.

4.1.3 ZebraNet Movement Traces

We used the ZebraNet movement model to test the adaptive energy policy performance under a realistic and highly correlated movement model [26]. This model is based on GPS location traces collected in the 2005 ZebraNet deployment.

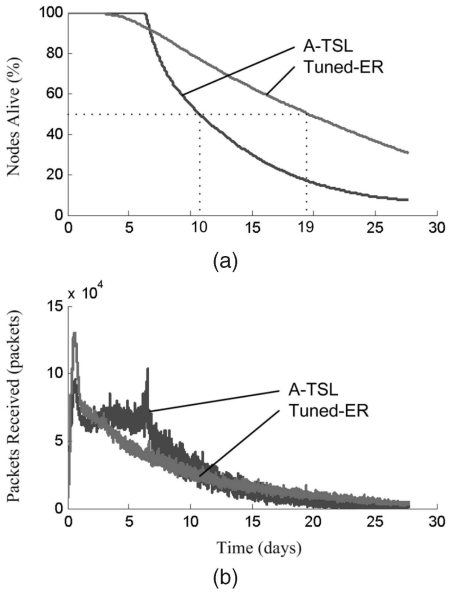

Fig. 10. Results for 100 nodes on a $100\ \mathrm{km} \times 100\ \mathrm{km}$ using ZebraNet movement traces, averaged over 100 runs. (a) Percentage of nodes operating. (b) Average packets transmitted. The figure shows CA-TSL adapting to the affect of node connection variability due to different movement patterns.

Due to animal feeding patterns, nodes in the ZebraNet model tend to remain static for long periods of time, causing an even lower apparent neighbor density of 0.16. Fig. 10 plots operating nodes in these experiments. In this case, under tuned-ER, the system lifetime increased by 12 days from the random movement lifetime of 6.9 days to 19 days. CA-TSL, in contrast, was more stable in the face of mobility changes, deviating by only 3-10 days. There is less communication than under a more mobile network. This is reflected in a much slower node die-off with both CA-TSL and tuned-ER methods, as shown in the top graph of Fig. 10. More importantly, the bottom graph shows that the communication bandwidth was more stable for CA-TSL, whereas tuned-ER produced a peak transfer and then fell. This means that CA-TSL is more likely to return a steady stream of sensor data representative of the entire experiment. In this case, CA-TSL was able to reduce lifetime variation by $4\times$ when faced with unanticipated connection patterns caused by unknown mobility patterns.

4.1.4 Comparison with an Adaptive Bidding Policy

While the results with CA-TSL are impressive compared to tuned-ER, it is also important to compare our method with more sophisticated approaches. We compared the CA-TSL policy with a policy based on a price-bidding approach in which nodes decide how and when to relay data by bidding on resources. In this bidding policy, "price" is set based on remaining node energy and willingness to pay is set by the amount of data stored by a node similar to [32]. Because of the sparseness of the system, price change policies cannot be communicated throughout the system, but are instead constant and obtained through multiple experiments. We assumed the best-case scenario in which the bidding policy does not incur a handshaking overhead to communicate its pricing information. Fig. 11 shows the percentage of operating nodes and the amount of data sent. The top graph

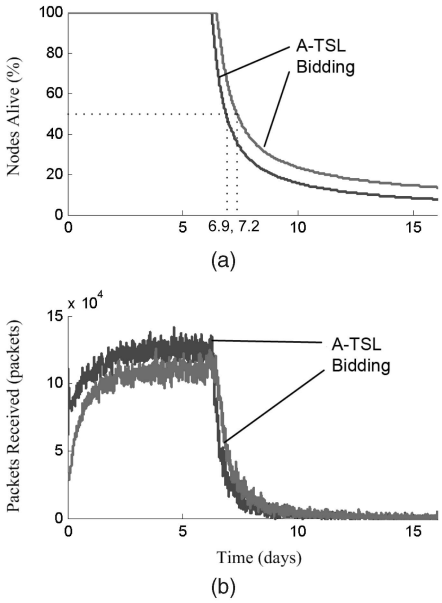

Fig. 11. Results for 100 nodes on a $100\ \mathrm{km} \times 100\ \mathrm{km}$ area comparing CA-TSL with a bidding policy, averaged over 100 runs. (a) Percentage of nodes operating. (b) Average packets transmitted. The figure shows CA-TSL has similar performance to high-communication overhead bidding policies.

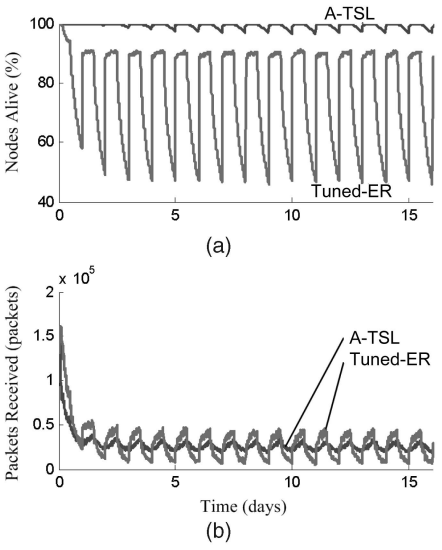

Fig. 12. Results for 100 solar-charged nodes using random walk on a $100\ \mathrm{km} \times 100\ \mathrm{km}$ area, averaged over 100 runs. (a) Percentage of nodes operating. (b) Average packets transmitted. The figure shows CA-TSL is resilient to different charging patterns.

shows that the lifetime of the two policies is similar, with CA-TSL at 6.9 and bidding at 7.2 days. The bottom graph shows that the number of packets communicated is also similar.

While it appears that CA-TSL performs only slightly better than the more complicated global bidding policy, CA-TSL has a few important advantages over bidding. The bidding policy relies on communications to establish an agreement on resource use. However, because we did not have a real-world implementation of the bidding policy, we removed this potentially significant extra overhead. More importantly, due to the lack of neighbors, pricing information cannot be set across multiple nodes, making competitive bidding difficult. Therefore, in order for the bidding policy to maintain stability, tuning has to be carried out over multiple runs in order to find the globally optimal pricing structure. Despite the experiment designed to be a best-case scenario for the bidding policy, CA-TSL was still able to perform closer to system expectations than the much more complex global policy.

4.2 Solar Charged Mobile Sensor Networks

Thus far, we have focused on uncharged systems in which the battery level drops monotonically during the node lifetime. In a charged system, additional challenges arise due to variations in charging conditions. For example, in a solar-charged system, the charge is variable due to day-night cycles as well as weather conditions and shading. For these charged systems, it is desirable to maintain node energy at a specific level. If the steady-state energy level is too low, nodes will be susceptible to frequent failure when the charging condition varies. On the other hand, if the steady-state energy level is too high, extra charge energy that cannot be stored is wasted, reducing the energy available for the system to exploit. These situations are especially undesirable if the time between node connections is long, as in sparse mobile networks.

Variation reduction. In this experiment, we test the mobile system's ability to maintain maximum connectivity when it experiences changes in day-night charging in the case of solar charging. The top graph of Fig. 12 shows the node availability for both CA-TSL and tuned-ER methods with 100 nodes. Under tuned-ER, the system drops to only 50 percent availability every night, before charge resumes. In contrast, our adaptive method was successful in confining the failure rate to less than 5 percent throughout the simulation, corresponding to $10\times$ fewer failures than with the tuned-ER method. The bottom graph of Fig. 12 shows that CA-TSL was able to maintain a more stable bandwidth while also providing better connectivity.

Sparse networks. In this experiment, we test the case of sparse charged networks, where 10 nodes are spread over a $100\ \mathrm{km} \times 100\ \mathrm{km}$ area with an average of only 0.02 neighbor per attempted communication. In such systems, it is especially difficult to revisit nodes once deployed; therefore, a conservative low-rate ER method is often used. We simulated sparse networks using 1) CA-TSL, 2) tuned-ER, and 3) a low-rate ER with connectivity similar to that of CA-TSL. As seen in the top graph of Fig. 13, because connections are infrequent, nighttime node failure is not as prevalent as in the previous case. However, a simulated 5-day rainy period, during which charging was not possible, caused 45 percent node failure in the tuned-ER case. In contrast, CA-TSL and the low-rate ER achieved a $4.5\times$ improvement with only 10 percent node failure. The bottom graph shows that while the low-rate ER case had similar numbers of operating nodes, the amount of data passed was much lower than with CA-TSL because CA-TSL was able to capture more charge during long idle periods. This experiment shows that the CA-TSL method can maintain high connectivity with near-optimal data transfer rates.

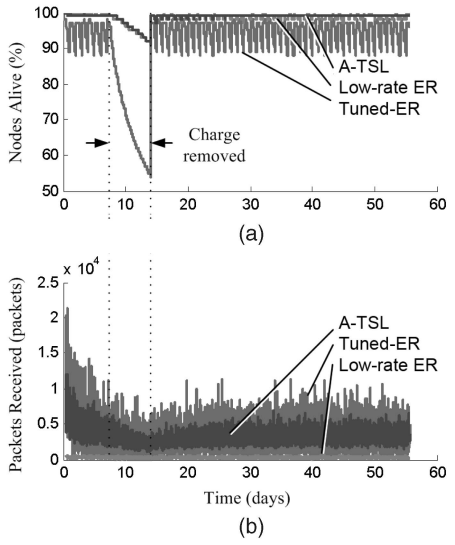

Fig. 13. Results for 10 solar-charged nodes using random walk on a $100\ \mathrm{km} \times 100\ \mathrm{km}$ area, averaged over 100 runs. (a) Percentage of nodes operating. (b) Average packets transmitted. The figure shows CA-TSL is resilient to different densities and charging patterns.

4.3 Dynamic System Parameter Estimation

We next implement and evaluate a variant policy in which collaborative techniques estimate both system power and system density. We compare this to existing policies used to gain knowledge of averages for connected neighboring nodes. The implementation is based on the three major phases outlined in Sections 2.4 and 2.5.

4.3.1 System Power Estimation

Figs. 14 and 15 illustrate experimental results for system power with sparse collaborative estimation. These graphs show the percentage of functioning nodes on the left axis, and the estimation error as calculated with respect to total available energy on the right axis. The 25th and 75th percentiles for the error are also shown as error bars. We see in Fig. 14 that as time progresses and the node's energy levels diverge, the median error and error range both increase until nodes begin to exhaust their energy under the existing policy. In contrast, the sparse collaborative estimation technique, shown in Fig. 15, does not exhibit large changes as time progresses. Furthermore, the sparse

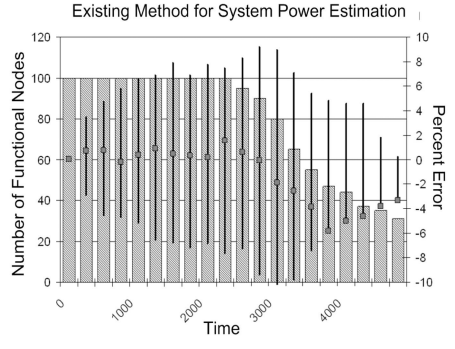

Fig. 14. Energy estimation without collaboration. Dots in the center indicate median of the error, vertical lines connect the 25th percentile and 75th percentile of the error, and bars indicate the number of nodes still operational in the system.

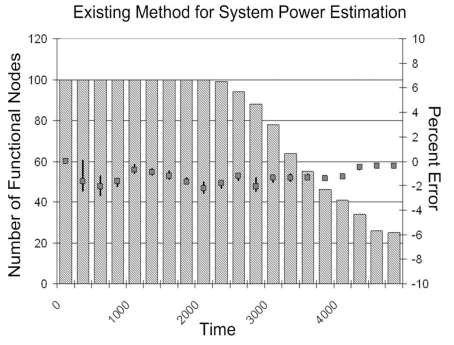

Fig. 15. Energy estimation with collaboration. Dots in the center indicate median of the error, vertical lines connect the 25th percentile and 75th percentile of the error, and bars indicate the number of nodes still operational in the system.

collaborative technique has an error range as narrow as -3 to 0 percent. In contrast, the existing policy achieved errors between -10 and 10 percent. This corresponds to a nearly $7\times$ improvement in estimation error. Variation in estimation is also greatly reduced in this sparse network. While an error range of 20 percent with the existing technique is not acceptable in most applications, the sparse collaborative policy has a maximum error of 3 percent from the true value. This opens up possibilities for many higher level techniques to be applied in very sparse networks.

This experiment shows that system energy cannot be estimated reliably using existing dense network policies, whereas it can be predicted reliably with the sparse collaboration policy.

4.3.2 Density Estimation

We also apply the sparse estimation technique to estimate the density of the sparse network. This information is especially useful in estimating the likelihood of communication, and hence, energy use. Density is difficult to estimate due to the unpredictable nature of node communications in intermittently connected networks. Yet knowing this parameter can increase the flexibility of various system policies. We show its impact on CA-TSL in Section 4.3.3.

In density estimation, each node maintains a log for density estimation. Each entry contains the estimation from other nodes (L), as well as the time (T) when the information was first measured from the original node. The estimation is weighted by the reciprocal of the time since information was received. For example, if the estimated density predicts that a meeting is due every a cycles, on average, and it is $n * a + x$ cycles (where $x < a$) since the most recent information was received, this information will be multiplied by $1/n$. However, because this method is intended to be implemented on a 4 MHz integer processor where division is expensive, we constrain 1/n to be a power of 2. In this context, m is defined as the minimum number of times that T needs to be shifted until all entries of T are less than 1. Then, L is shifted left m times to obtain an intermediate value $S = L \ll m$. The estimation is then obtained by $\frac{SUM(All\ S\ in\ log)}{SUM(All\ T\ in\ log)}$.

Prediction. The prediction phase predicts the certainty of the log entries based on the number of meetings predicted. Since certainty is determined by the time elapsed since the

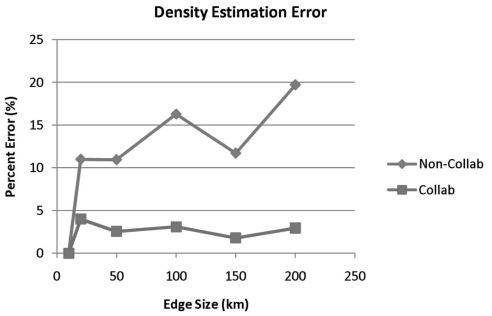

Fig. 16. Error of connection period estimation comparing local and collaborative estimation with 100 nodes on fields of different edge sizes.

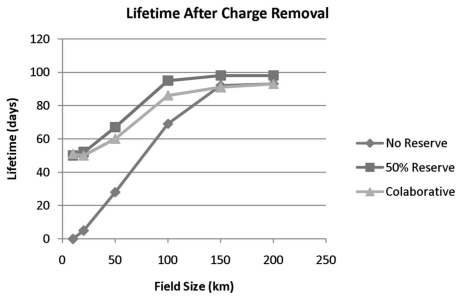

Fig. 17. Cycles when system reaches 50 percent working nodes after charge is removed. The experiment featured 100 nodes in fields of different sizes.

data were received, certainty decreases during each cycle by $T = T + 1$.

Transform. Similar to the power estimation technique described before, if a neighbor is found, the nodes exchange their local logs of density estimations and time elapsed since information was first received.

Update. The update phase takes the data from previously unseen nodes and incorporates them into the log. In addition, it takes update information from previously seen nodes and replaces the existing entry. All information is capped by the size of the log.

Fig. 16 shows the accuracy of the density estimation at different densities. The experiment places 100 nodes in a square field with different sizes. We see that accuracy is roughly constant except when density is high enough that nodes have neighbors during all communication attempts (left). The error with only local information is greater at a range of between 10 and 20 percent. However, collaborative estimation provides better results, with a maximum error of only 4 percent. The results show that the collaborative method successfully predicts density with an acceptable error. We use this estimation in the next section, for dynamically selecting expected energy levels of CA-TSL.

4.3.3 CA-TSL in Dynamic Environments

With accurate estimation of system parameters even in sparse networks, CA-TSL can measure and account for system characteristics. This section evaluates the performance of CA-TSL using density estimation to adjust the expected energy curve of CA-TSL as described in Section 2.1.

In each experiment, nodes were given a radio range of 1 km and a square field within which the nodes could move. The movement model used was a random walk with a uniform distribution from 0 to 1 km and a random direction. One hundred nodes were used in each experiment, and they were charged periodically, with charging during 10 cycles and no charging during 190 cycles.

Lifetime. Many charged systems exhibit periodic charging behavior and possible periodic charge loss. The reason for keeping battery reserves is to maintain system functionality during periods of no charging. These experiments are designed to show how different policies respond to changes in density. In these experiments, we compare performance of CA-TSL with global density estimation against the same policy without parameter estimation (using different assumption of density).

Fig. 17 compares lifetimes of the system after charge loss at different densities. Each experiment was set up with different sizes for the square area where the nodes could roam. CA-TSL was compared with local CA-TSL with reserves of 50 percent and 0 percent battery capacity for the assumption of high and low density, respectively. We see from the graph that while the 50 percent battery capacity maintained a reasonable lifetime at high densities, it kept the level high at different densities as well. Whereas the no-reservoir case had an unacceptably short lifetime at high densities, the collaboration method provided the same lifetime at lower densities. On the other hand, as density dropped, the lifetime of CA-TSL approached that of the no-reservoir case and lifetime was artificially inflated. This result was not surprising because CA-TSL lowers the reserve as connections become sparse. This shows that the collaboration policy provides an acceptable lifetime for various densities.

Data received. While system lifetime is an important factor in system performance, the amount of data passed in the system is an important indicator of system capability. This section examines the data throughput of CA-TSL.

In a charged system, the amount of charge obtained determines the maximum amount of data that can be passed in the system. Fig. 18 shows the total data passed in the system normalized to the CA-TSL policy. We have seen that the no-reservoir policy can pass the most data because it can accumulate the most energy. Since it maintains no reserve, it is able to maintain a maximum reservoir for charging and obtain maximum energy when charge is available. However, Fig. 17 shows that while the amount of data transmitted by a no-reservoir fixed policy is impress-

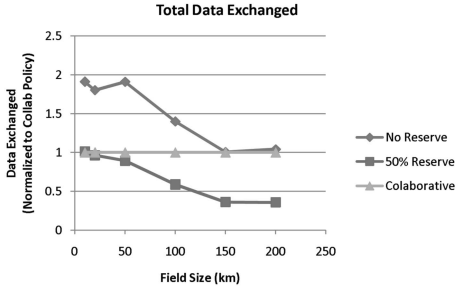

Fig. 18. Total data received by all nodes in the system. The experiment had 100 nodes on fields of different sizes, charged for 10 cycles and not charged for 190 cycles.

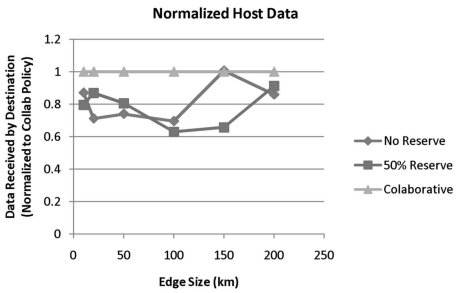

Fig. 19. Data received by the destination normalized to the CA-TSL. The experiment had 100 nodes on fields of different sizes, charged for 10 cycles and not charged for 190 cycles.

ive, the resulting system has an unacceptably short lifetime when charge is unavailable. Fig. 18 also shows that a 50 percent reservoir, on the other hand, passes the least amount of data, because it can charge much less due to capacity remaining in the battery. CA-TSL maintains a reservoir to maintain a lifetime requirement when density is high (left), similar to the 50 percent reservoir. Yet as density is reduced, the data bandwidth of CA-TSL approaches the no-reservoir case, the data bandwidth of the collaboration method improves by a factor of nearly 3.

Communication in a sensor system is often intended for one destination, such as a common data sink. The data sink is commonly connected to a constant power supply and does not depend on charging. Systems are designed to pass their data to the data sink on connection and to ignore the energy budget. Fig. 19 shows the data received by the data sink of both collaborative and local CA-TSL. The collaborative policy is able to pass more data to the sink. At the high-density end (left), it is able to maintain a battery level similar to the 50 percent reservoir case and it can transmit data to the sink. At the low-density end (right), it is able to maintain additional reserve energy, allowing it to transmit more data than in the 0 percent reservoir case. Furthermore, due to the dynamic nature of each node's surroundings, the local system estimate enables the node to adapt dynamically to conditions that vary over time, allowing more data to be transmitted. In addition, each node has more data available to it and can forward more data to the data sink. This results in up to 30 and 37 percent more data received over the no-reservoir case and 50 percent over the reservoir case.

This section has shown that CA-TSL significantly improves system functionality, both by maintaining system lifetime and increasing data transferred to the data sink. Through collaboration, accurate system information is obtained and can be reliably used locally by each device. This information both increases data capacity and maintains system lifetime in a sparse, intermittently connected network.

5 Discussion

CA-TSL simplifies application support by using discontinuous control that only requires applications to have a bounded execution time that is small compared to the lifetime, and per-cycle energy use that is small compared to the battery capacity [31]. To prevent unintended application errors, CA-TSL does not interfere during application execution. Therefore, each run of the application must finish within a fixed time, which means that many applications can be incorporated without modification. However, other application design considerations need to be taken into account.

Data delay. CA-TSL introduces no additional data delay for networks that have ample energy for the needed packets per slot. When a node's actual energy drops below the energy usage target, it reduces communication, and thus, may delay some packets in order to pursue the targeted system lifetime. Once the node is able to transmit again, different application policies can choose whether to forward the delayed data, or drop it in favor of fresher sensor readings. However, as shown in Section 3, in a sparse mobile network for which CA-TSL is designed, the opportunistic nature of this design significantly reduces data delay in the common case, especially when networks are sparse.

Multiple applications. Another design consideration is when there are multiple applications and peripherals to control (e.g., Sensing and Communications). The previous section describes how the energy budgeting module would allocate total energy for all applications. In addition, the applications can adapt further and split the available budget. For example, it can reduce communication energy relative to sensing energy when the energy budget is low.

6 Related Work

There is a wide range of research on constraining variable system lifetimes. Areas relevant to our work include energy minimization to increase system lifetime, routing algorithms to increase data balance, and adaptive methods to increase lifetime. In this section, we discuss some of the previous work and contrast it with our own.

Energy reduction methods. Most of the previous sensor deployments tackle the targeted system lifetime problem by using a fixed schedule based on a conservative energy profile estimate. This method is the simplest one, but unlike CA-TSL, it ignores long-term variations inherent in the system. An energy reduction method based on a very low duty cycle and a conservative estimate of the energy budget was used in [9], [15], [21], [24]. However, in these deployments, node lifetimes varied due to unexpected changes in conditions. Solar-charged network deployments [6], [30] also have trouble maintaining full node operation due to long-term variations in charging. Other work ignores the variation problem by changing the nonrechargeable batteries [27], but this is not always possible in remotely deployed systems. CA-TSL, due to its adaptive nature, can handle both expected and unexpected long-term harvesting variations by using a linear energy prediction model.

Smart routing algorithms. In a dense static network, node redundancy and special network topologies can also be exploited [1], [4], [11], [12], [13], [16], [20], [28], [29]. While these special topologies make route discovery easier and reduce overhead, they are limited to fixed topologies and are not usable for mobile networks, thus, not suitable in application targeted by CA-TSL. Other bidding-style methods have been developed to balance data and energy, where a number of metrics could be used to determine the price of the transaction in a virtual currency [3], [19], [32]. While these can be used in a dense and limited mobile network,

overhead costs for these types of routing scheme are unfortunately high, especially in a sparse mobile network described in this paper. Furthermore, they are difficult to implement in intermittently connected mobile networks, since nodes are not always connected. Furthermore, Unlike CA-TSL, these schemes also do not take into account differences between energy profiles of other peripherals for the nodes, nor do the schemes gain system parameter information when nodes are disconnected, due to their inability to predict and track system parameters.

Hardware variation considerations. An approach has been developed to change the duty cycle according to node availability and to use energy harvesting variations for fixed networks [14]. This method is different from CA-TSL because it requires knowledge of the amount of energy already harvested, in addition to knowing the total energy. Obtaining this requires the addition of either an energy harvest learning algorithm, or a monitor to measure and store charging current. This makes these types of methods impractical, since they require additional hardware components to deal with any change in hardware or topology.

Other analytical methods have been investigated to achieve fixed lifetimes for clustered fixed networks [8], [18]. Unlike CA-TSL, these methods assume invariant power consumption of the nodes and do not adapt to achieve a certain lifetime. Instead, they assume a fixed network that consists of nodes with invariant energy consumption and that is fitted with different hardware or bandwidth characteristics to account for topological differences.

Other domains. Other works related to this research exist in more general-purpose domains. For example, research has been carried out on mobile laptops in order to set targeted battery life and allow the operating system to adapt to the lifetime expectations of the user [7]. Laptops, however, are subject to different requirements and limitations than mobile ad hoc sensor networks; they do not usually need to cooperate with other laptops, whereas CA-TSL allows cooperation among the nodes to improve functionality. These different challenges and limitations prevent implementation of these techniques in sensor network nodes.

7 Conclusion

In this paper, we have shown that CA-TSL reduces deviation of node lifetime from the targeted lifetime by up to $5.5\times$, it increases the efficiency of energy use by passing up to 50 percent more packets, and it reduces node failures by up to 10 times for charged nodes. Results further show that in very dynamic environments, dynamic parameter estimation not only accurately predicts system parameters but also further increases the number of packets received by 35 percent.

The delay-tolerant CA-TSL offers effective energy adaptation and reduces the effects of the many sources of energy profile variations in sparse mobile sensor systems. CA-TSL enables more efficient use of energy, and therefore, better functionality in the system. Collaborative adaptation is effective in enhancing functionality and increasing system efficiency, which would otherwise be impossible in low-density systems. Furthermore, these results demonstrate that by adapting to global information, the system is more efficient than it would be with energy reduction or local adaptation alone. CA-TSL improves not only *node* performance, but also *system* functionality and longevity. More broadly, we think that this work offers an effective and easy-to-implement policy for a wide range of sparse, high-variation, energy-constrained, and mobile sensor networks.

References

[1] K. Akkaya and M. Younis, "A Survey on Routing Protocols for Wireless Sensor Networks," *J. Ad Hoc Networks,* vol. 3, pp. 325-349, May 2005.

[2] R. Braynard, A. Silberstein, and C. Ellis, "Extending Network Lifetime Using an Automatically Tuned Energy-Aware MAC Protocol," technical report, Eidgenössische Technische Hochschule (ETH) Zurich, Feb. 2006.

[3] L. Buttyan and J.P. Hubaux, "Nuglets: A Virtual Currency to Stimulate Cooperation in Self-Organized Mobile Ad Hoc Networks," Technical Report DSC/2001/001, Swiss Fed. Inst. of Technology, 2001.

[4] M. Cardei and D.Z. Du, "Improving Wireless Sensor Network Lifetime through Power Aware Organization," *J. Wireless Networks,* pp. 333-340, May 2005.

[5] J.-H. Chang and L. Tassiulas, "Energy Conserving Routing in Wireless Ad-hoc Networks," *Proc. IEEE INFOCOM,* vol. 1, pp. 22-31, May 2000.

[6] P. Dutta, J. Hui, J. Jeong, S. Kim, C. Sharp, J. Taneja, G. Tolle, K. Whitehouse, and D. Culler, "Trio: Enabling Sustainable and Scalable Outdoor Wireless Sensor Network Deployments," *Proc. Fifth Int'l Conf. Information Processing in Sensor Networks (IPSN '06) Special track on Platform Tools and Design Methods for Network Embedded Sensors (SPOTS '06),* pp. 407-415, Apr. 2006.

[7] J. Flinn and M. Satyanarayanan, "Energy-Aware Adaptation for Mobile Applications," *Proc. Symp. Operating Systems Principles,* pp. 48-63, Dec. 1999.

[8] J. Flinn and M. Satyanarayanan, "Rate Allocation in Wireless Sensor Networks with Network Lifetime Requirement," *Proc. Fifth ACM Int'l Symp. Mobile Ad Hoc Networking and Computing,* pp. 67-77, 2004.

[9] T. He, P. Vicaire, T. Yan, Q. Cao, G. Zhou, L. Gu, L. Luo, R. Stoleru, J.A. Stankovic, and T. Abdelzaher, "Achieving Long-Term Surveillance in VigilNet," *Proc. IEEE INFOCOM,* vol. 5, Apr. 2006.

[10] B. Hohlt, L. Doherty, and E. Brewer, "Flexible Power Scheduling for Sensor Networks," *Proc. Int'l Symp. Information Processing in Sensor Networks (IPSN),* pp. 205-214, Apr. 2004.

[11] B. Hong and V. Prasanna, "Optimizing System Life Time for Data Gathering in Networked Sensor Systems," *Proc. IEEE Workshop Algorithms for Wireless and Ad-Hoc Networks (A-SWAN),* July 2004.

[12] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," *Proc. ACM MobiCom,* pp. 56-67, Aug. 2000.

[13] K. Kalpakis, K. Dasgupta, and P. Namjoshi, "Maximum Lifetime Data Gathering and Aggregation in Wireless Sensor Networks," *Proc. IEEE Int'l Conf. Networking (ICN '02),* pp. 685-696, Aug. 2002.

[14] A. Kansal and M. Srivastava, "An Environmental Energy Harvesting Framework for Sensor Networks," *Proc. Int'l Symp. Low Power Electronics and Design,* pp. 481-486, Aug. 2003.

[15] K. Langendoen, A. Baggio, and O. Visser, "Murphy Loves Potatoes: Experiences from a Pilot Sensor Network Deployment in Precision Agriculture," *Proc. 14th Int. Workshop Parallel and Distributed Real-Time Systems (WPDRTS),* Apr. 2006.

[16] S. Madden, M.J. Franklin, J. Hellerstein, and W. Hong, "TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks," *Proc. Fift Symp. Operating System Design and Implementation (OSDI),* pp. 131-146, Dec. 2002.

[17] Maxstream, Inc., *XTend OEM RF Module: Product Manual v1.2.4,* http://www.maxstream.net, Oct. 2005.

[18] V.P. Mhatre, C. Rosenberg, D. Kofman, R. Mazumdar, and N. Shroff, "A Minimum Cost Heterogeneous Sensor Network with a Lifetime Constraint," *IEEE Trans. Mobile Computing,* vol. 4, no. 1, pp. 4-15, Jan./Feb. 2005.

[19] P. Michiardi and R. Molva, "CORE: A COllaborative REputation Mechanism to Enforce Node Cooperation in Mobile Ad Hoc Networks," *Proc. IFIP TC6/TC11 Sixth Joint Working Conf. Comm. and Multimedia Security,* pp. 107-121, Sept. 2002.

[20] S. Soro and W. Heinzelman, "Prolonging the Lifetime of Wireless Sensor Networks via Unequal Clustering," *Proc. 19th IEEE Int'l Parallel and Distributed Processing Symp. (IPDPS '05),* Apr. 2005.

[21] R. Szewczyk, J. Polastre, A. Mainwaring, J. Anderson, and D. Culler, "An Analysis of a Large Scale Habitat Monitoring Application," *Proc. Second ACM Conf. Embedded Networked Sensor Systems (SenSys),* pp. 214-226, Nov. 2004.
[22] Texas Instrument, "SBS 1.1-Compliant Gas Gauge Enabled with Impedance Track(TM) Technology for Use with the bq29312," http://www.ti.com, Oct. 2005.
[23] Texas Instrument, "Single-Cell Li-Ion and Li-Pol Battery Gas Gauge IC For Portable Applications Data Sheet," http://www.ti.com, Oct. 2005.
[24] G. Tolle, J. Polastre, R. Szewczyk, N. Turner, K. Tu, P. Buonadonna, S. Burgess, D. Gay, W. Hong, T. Dawson, and D. Culler, "A Macroscope in the Redwoods," *Proc. Third ACM Conf. Embedded Networked Sensor Systems (SenSys),* pp. 51-63, Nov. 2005.
[25] I. Vasclescu, K. Kotay, D. Rus, P. Corke, and M. Duabablu, "Data Collection, Storage and Retrieval with an Underwater Optical and Acoustical Sensor Networks," *Proc. Third ACM Conf. Embedded Networked Sensor Systems (SenSys),* pp. 154-165, Nov. 2005.
[26] Y. Wang, P. Zhang, T. Liu, C. Sadler, and M. Martonosi, "Movement Data Traces from Princeton ZebraNet Deployments," CRAWDAD Database, http://crawdad.cs.dartmouth.edu, Oct. 2007.
[27] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh, "Fidelity and Yield in a Volcano Monitoring Sensor Network," *Proc. Seventh USENIX Symp. Operating Systems Design and Implementation (OSDI '06),* pp. 381-396, Nov. 2006.
[28] O. Younis and S. Fahmy, "Distributed Clustering in Ad-Hoc Sensor Networks: A Hybrid, Energy-Efficient Approach," *Proc. IEEE INFOCOM,* pp. 629-640, Mar. 2004.
[29] Y. Yu, R. Govindan, and D. Estrin, "Geographical and Energy Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks," Technical Report UCLA/CSD-TR-01-0023, University of California, Los Angeles, Computer Science Dept., May 2001.
[30] P. Zhang, C. Sadler, S. Lyon, and M. Martonosi, "Hardware Design Experiences in ZebraNet," *Proc. Second ACM Conf. Embedded Networked Sensor Systems (SenSys),* pp. 227-238, Nov. 2004.
[31] P. Zhang, C. Sadler, and M. Martonosi, "Middleware for Long-Term Deployment of Delay-Tolerant Sensor Networks," *Proc. First Int'l Workshop Middleware for Sensor Networks (MidSens '06),* Nov. 2006.
[32] S. Zhong, J. Chen, and Y.R. Yang, "Sprite: A Simple, Cheat-Proof, Credit-Based System for Mobile Ad-Hoc Networks," *Proc. IEEE INFOCOM,* pp. 1987-1997, Apr. 2003.

Pei Zhang received the bachelor's degree from the California Institute of Technology and the master's and PhD degrees from Princeton University, all in electrical engineering. He is an assistant research professor in Cylab, the Information Networking Institute, and the Electrical and Computer Engineering Department at Carnegie Mellon Silicon Valley. His primary research interest is in embedded systems with a special focus on mobile sensing networks. He is especially interested in developing practical system solutions to improving system adaptability, robustness, and autonomy. While at Princeton University, he worked on the ZebraNet system, a sparse mobile sensor system. Being the first deployed, wireless, ad-hoc, mobile sensor network, his work was proved during its deployments at Sweetwaters Game Reserve in central Kenya. At Carnegie Mellon, he leads the SensorFly project, which developed the first mote style flying sensor networks platform. He is a member of the IEEE.

Margaret Martonosi received the bachelor's degree from Cornell University and the master's and PhD degrees from Stanford University, all in electrical engineering. She is currently a professor of electrical engineering at Princeton University, where she has been on the faculty since 1994. She also holds an affiliated faculty appointment in the Computer Science Deparment at Princeton University. Her research interests include computer architecture and the hardware/software interface, with particular focus on power-efficient systems and mobile computing. In the field of mobile computing and sensor networks, she led the Princeton ZebraNet Project, which included two real-world deployments of tracking collars on Zebras in Central Kenya. She is a fellow of the IEEE and the ACM.