Flexible Power Scheduling for Sensor Networks

Barbara Hohlt
Computer Science Division
University of California, Berkeley
Berkeley, CA, USA 94720-1776
hohltb@eecs.berkeley.edu

Lance Doherty
Electrical Engineering Division
University of California, Berkeley
Berkeley, CA, USA 94720-1770
ldoherty@eecs.berkeley.edu

Eric Brewer
Computer Science Division
University of California, Berkeley
Berkeley, CA, USA 94720-1776
brewer@cs.berkeley.edu

ABSTRACT
We propose a distributed on-demand power-management protocol for collecting data in sensor networks. The protocol aims to reduce power consumption while supporting fluctuating demand in the network and provide local routing information and synchronicity without global control. Energy savings are achieved by powering down nodes during idle times identified through dynamic scheduling. We present a real implementation on wireless sensor nodes based on a novel, two-level architecture. We evaluate our approach through measurements and simulation, and show how the protocol allows adaptive scheduling and enables a smooth trade-off between energy savings and latency. An example current measurement shows an energy savings of 83% on an intermediate node.

Categories and Subject Descriptors
C.2 [**Computer Communication Networks**]: Network Architecture and Design, Network Protocols, Network Operations, Distributed Systems;

C.3 [**Special-Purpose and Application-Based Systems**]: Real-time and embedded systems*;*

D.4.4 [**Operating Systems**]: Communications Management

General Terms
Algorithms, Management, Measurement, Design, Economics, Experimentation

Keywords
Sensor Networks, Communication Scheduling, Power Management

1. INTRODUCTION
The combination of technological advances in integrated circuitry, MEMS, communication and energy storage has driven the development of low-cost, low-power sensor nodes [14,2].

Networking many nodes through radio communication allows for data collection via multi-hop routing, but the practical limits on available power and the lack of global control present challenges. Constraints imposed by limited energy stores on individual nodes require careful selection of tasks, and as communication is the most costly task in terms of energy, it must be used particularly sparingly. In general, sensor nodes comprising the network have the ability to sense the environment, control actuators, make simple computations, and communicate data either to other nodes or to a centralized observer. We will consider networks consisting of the Crossbow MICA sensor nodes [6] running the UC Berkeley TinyOS operating system [13].

Power consumption limits the utility of sensor networks. Replacing batteries every week in building networks is a laborious task and replacing them in a less friendly environment may not be possible. Researchers agree, above all functions, radio communication dominates the power consumed in wireless sensor networks [2,8,25,19]. At the communication distances typical in sensor networks, listening for information on the radio channel is of a cost similar to transmission of data [21], so unnecessary radio operation must be pared to increase node lifetimes. In addition, the energy cost for a node in idle mode is approximately the same as in receive mode. Therefore, protocols that assume receive and idle power are of little consequence are not efficient for sensor networks. *Idle listening*, the time spent listening while waiting to receive packets, is a significant cost. Stemm et al. [27] observed that idle listening dominated the energy costs of network interfaces in hand-held devices. It became clear that to reduce power consumption in radios, the radio must be *turned off* during idle times. Mangione-Smith and Ghang [18] proposed such a scheme for an energy-efficient MAC layer for one-hop mobile devices.

This paper presents *Flexible Power Scheduling* (FPS), a distributed power management protocol for sensor networks that reduces radio power consumption while supporting fluctuating demand in the network. A novel, two-level architecture combines coarse grain scheduling at the routing layer to plan radio on/off times and fine grain medium access control to provide channel access. The protocol provides local communication schedules for a multi-hop sensor network and acts only on locally acquired information. We detail a distributed algorithm that exploits a tree based topology in combination with an adaptive slotted communication schedule to route packets, synchronize with neighbors, and schedule radio on/off times. The assignment and modification of schedules is based on a *supply and demand* algorithm that allows for implicit and explicit deletion of nodes from the network without global control or re-initialization.

IPSN'04, April 26–27, 2004, Berkeley, California, USA.

We use sense-to-gateway data collection as the application driver for our protocol. Sense-to-gateway applications represent a large class of wireless sensor network applications and have a natural tree topology that can be exploited for flexible power scheduling. These applications collect data from the environment and forward the data to a base station where it can be stored in a central database for evaluation. Such applications include equipment tracking, building-wide energy monitoring, habitat monitoring [17], conference room reservations [5], art museum monitoring [24], and automatic lawn sprinklers [7]. The communication is primarily one way: from the data-collecting node to the base station. Our protocol may be extended to two-way communication, but in this paper we will focus primarily on network to gateway communication. Our protocol does not support arbitrary many-to-many communication such as would be required in event tracking applications.

The remainder of the paper is organized as follows: Section 2 describes sensor network issues, Section 3 introduces our flexible power scheduling protocol, Section 4 makes the algorithm precise, Section 5 describes an implementation on UC Berkeley nodes and gives some experimental results (energy, adaptation, latency), Section 6 discusses related work, and Section 7 provides concluding remarks.

2. BACKGROUND DISCUSSION

Multi-hop topologies play a significant role in sensor networks. In terms of wireless communication, it is more energy efficient to transmit over several short distances than it is to transmit over a few long distances [19]. Short distances require less energy to transmit and have better signals resulting in fewer retransmissions due to packet loss.

Multi-hop networks have problems with congestion at the sink and unfair end-to-end bandwidth allocation. The farthest nodes do not have as fair a chance of getting their data to the base station as the nearest nodes. Due to the lossy nature of wireless links, the chance of packet loss grows geometrically with each hop. Traffic in sensor networks tends to be highly correlated as well. Asynchronous events can trigger sudden bursts of traffic that can lead to collisions, congestion, and channel capture [28].

In theory, slotted time division schemes can solve these problems. Slotted time division has a natural structure that leaves traffic uncorrelated and provides end-to-end fairness. More importantly, slotted time division schedules are *energy efficient* because it is known *when the radio will be idle*. Global schedules can be generated in such away that bandwidth is essentially reserved from source to sink and from the schedule it is clear when to turn the radio on and off locally.

By and large, slotted time division schemes require centralized control, have static global schedules, and require fine-grain time synchronization. The algorithm proposed in this paper exploits a tree-based topology that enables a distributed algorithm using no centralized control, has adaptive local schedules, and requires only coarse-grain time synchronization.

We propose a two level architecture: coarse-grain scheduling at the routing layer to schedule all communication for the purpose of powering the radio on-and-off during idle times, and fine-grain medium access control at the MAC-layer to handle channel access.

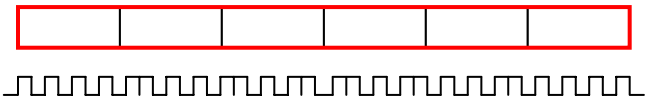

Figure. 1: Coarse-grain scheduling (above) and fine-grain medium access (below).

Fairness requires a global view of the network. We need some information about traffic and topology, which requires a higher-level operation. Slotted systems require network-wide fine-grain time synchronization for use of discrete time slots. This is easy to achieve in centralized networks, but much more difficult in multi-hop networks. If we choose a coarse-grain schedule then time synchronization becomes easy and will be sufficient for the algorithm proposed here.

The algorithm is distributed. There is no oracle to generate and distribute a network-wide global schedule. However, discovery can be done locally because we are using a constrained tree topology. Through the exchange of messages a local schedule can be generated at each node and adapted to the changing needs of the network. On the TinyOS platform messages are delivered at the Active Message layer [12] and the routing layer is above. It is the routing layer that has knowledge of route through traffic as well as source-generated traffic in a sensor node.

Although the coarse-grain schedule reduces contention, it is not the case the medium will be absolutely contention free. The algorithm does not generate the perfect distributed schedule, so we still need a MAC layer to handle those cases where in different parts of the network nodes are transmitting at the same time, but can overhear each other.

The coarse-grain schedule reduces contention because it can coordinate transmission times and distribute traffic. A MAC layer is still required, but will have less work to do. The combination of a coarse-grain schedule and MAC-layer protocol reduces contention and increases end-to-end fairness. The distributed schedule has the effect of doing connection-less flow control and the distributed algorithm provides reserved bandwidth from source to sink.

3. FPS PROTOCOL

3.1 Goals of the Protocol

The main goal of flexible power scheduling is to reduce power consumption while supporting fluctuating demand in the network for data collection. It schedules transmit and receive time slots in each node's *power schedule* and sleeps during idle periods. Local power schedules dynamically adapt as network demand changes. The assignment and modification of schedules is distributed without global control and without a network-wide global initialization phase. The protocol supports data collection applications and communication is assumed to be primarily one-way, toward a base station, and there may be multiple base stations.

3.2 Definitions

Time is broken up into slots. Each slot s corresponds to a length of time T_s. Slot numbering is periodic modulo m, i.e. slot $s+m$ is called slot s. Generally, the same events will occur periodically

and during the same slot in different cycles. A cycle *c* hence represents a length of time $T_c = mT_s$.

Each node maintains a local *power schedule* of what operations it performs over the course of a cycle. During each slot of the cycle, the node can have one of six states that, for now, can be considered as one of three states: (i) Receiving, (ii) Transmitting, or (iii) Idle. In general, the number of receive slots in the local schedule of node *a* plus its own original source demand represents the *demand* at node *a* and the number of transmit slots in the local schedule at node *a* represents the *supply* at node *a*. Transmit and receive slots in the local schedule allow the node to know when its neighbors are ready to transmit to or receive from this node.

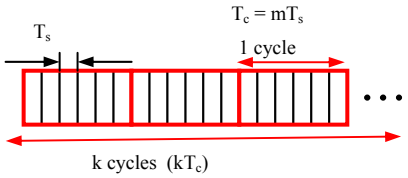

Figure. 2: Definition of the terminology used in power scheduling

Conceptually, we represent the collection of local schedules as an *N*x*m* matrix *Ω* where *N* is the total number of nodes. For example, *Ω*(*n,s*) tells which of the three states node *n* is in during slot *s*. In implementation, a node *n* would only have access to the *n*th row of *Ω*. The schedule is initialized to all Idle and evolves over the run of the scheduling algorithm. We assume that the node can power down during idle slots. Figure 2 depicts *k* cycles of a *power schedule* at node *n*.

Let the number of slots that a node is not idle during a cycle be *b*(*n*), *b*(*n*) <= *m*. Now the duty cycle of a node will be:

$$DutyCycle(n) = \frac{b(n)}{m} \qquad (1)$$

Equation (1) emphasizes two ways that duty cycle can be reduced at a node: either reduce the amount of non-idle slots, or lengthen the cycle. Each node can then adapt its particular power schedule to minimize its energy consumption independent of activity at other nodes.

For simulation purposes, a network is defined by two parameters: a binary connectivity matrix *C* and an initial demand vector *d*. *C*(*a,b*) = 1 if node *a* can receive transmissions from node *b* and 0 otherwise. In general, this matrix could be a function of time to represent mobile nodes or changing transmission channels, or could be a real number between 0 and 1 representing the probability of packet loss. The demand vector indicates how many packets node *a* seeks to forward during each cycle. Initially, every node has a demand of 1 for itself. In the our current scenario, *d*(*a*) is the *integer demand* at node *a*. Integer demand counts one packet per cycle per unit and *d(a)* represents the sum of the demand at node *a* and the children of node *a*. Our protocols can easily be adapted to accommodate finer units of demand. For example one unit could represent one packet every 32 cycles. These parameters completely specify the network and the goal state, namely that each node in the network is scheduled to forward one packet upstream per unit of demand.

3.3 A Simple Example

As an illustration of the protocol, consider the network depicted in Figure 3. Nodes 0 through 5 represent the nodes of a network each having different demand. The squares represent nodes with an initial demand of 1. The solid circles represent demand that originates at the node. The lines between squares are the connectivity graph in terms of communication links.

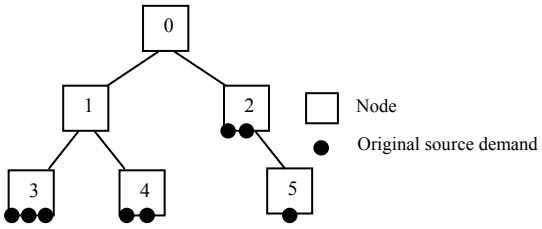

Figure. 3: Network used in the simple example.

Suppose that node 0 is the base station and that all other nodes initialize with at least a demand of 1, i.e. they each need at least one communication slot with their parent. Nodes always begin with one extra unit of demand. Starting with the leaves of the tree, nodes 3 and 4 both require transmission slots to communicate to their parent, node 1. Thus, the effective demand of node 1 is 8 and node 1 requires 8 communication slots with the base station. On the other branch of the tree, node 2 requires 5 communication slots with the base station. From this simple example, it is apparent that nodes closer to the root will have higher duty cycles as is the case in general with sinks in sensor networks.

Another observation is that the data from nodes on level *l* can be delayed by as many as *l* cycles upon reception at the root. In the worst case, data from nodes 1 and 2 in the example are received after 1 cycle while data from nodes 3-5 are delayed by 2 cycles. If a particular latency is required, the number of slots *m* may have an upper bound that increases the overall duty cycle in the network. The tradeoff between latency and duty cycle will be further explored in later sections.

4. ALGORITHM DETAILS

4.1 Node States and Reservations

Each node maintains its own transmit and receive schedule of time slots known as a power schedule. One power schedule cycle is equivalent to *m* time slots in length. A *reservation slot* refers to a receive/transmit pair between a parent node and child node.

Each time slot a node can be in one of 6 states:

1. Transmit (T) – Transmit a message to parent node, remains in schedule until topology or supply/demand changes

2. Receive (R) - Receive a message from child node, remains in schedule until topology or supply/demand changes
3. Advertisement (A) - Broadcast an Advertisement from parent node for an available reservation slot, remains at most one cycle
4. Transmit Pending (TP) – Send a reservation request to parent node, remains at most one cycle
5. Receive Pending (RP) – Receive a reservation request from child, remains at most two cycles
6. Idle (I) – After all current demand at this node has been met, the node can power down during idle slots

When a parent node broadcasts an advertisement for a reservation it marks the reservation slot (not the advertisement slot) in its own schedule as Receive Pending. If a child node has unmet demand and it hears an advertisement, it marks the reservation slot in its own schedule as Transmit Pending and sends a reservation request when the time slot arrives. An explicit example of the standard operation for two nodes (node 1 is closer to the base station than node 2) is shown in Table 1. Here node 1 and node 2 reserve time slot R for communication.

Table 1 shows the interaction between two nodes to set up communication. The headings *C* and *S* denote cycle number and slot number respectively. Node 1, the parent, begins by selecting an advertisement slot A and a reservation slot R at random from its idle slots. The reservation process requires three cycles. The first is the advertisement from the parent, the second is the response from the child and immediate confirmation, and the third is the first actual transmission of data. Thereafter, node 2 transmits during slot R. Advertisement messages carry the node id, number of hops from the base station, the current slot number, current demand, and reservation slot number. The return request messages carry the same information.

Table 1: Reservation between two nodes

C	S	Node 1	Node2
1	A	Advertise slot R →	Receive advertisement
2	R	Send immediate confirmation	← Request slot R
3	R	Receive from Node 2	← Transmit to Node 1
.	.	…	…
N	R	Receive from Node 2	← Transmit to Node 1

4.2 Synchronization

When a child node selects a parent it synchronizes its current time slot and slot number to that of its parent. Periodically, thereafter, the child node resynchronizes with its parent. Only coarse-grain synchronization is required due to the relatively large time slots. In the example from Table 1, node 2 synchronizes with node 1 during cycle 1 and time slot A. If available, link layer acknowledgements containing timestamps can be used for subsequent resynchronization. Otherwise, node 2 can snoop during a known transmission time of node 1. This requires node 1 to include a list or partial list of its transmission slots in its advertisement messages. A simple, but not optimal, solution is for node 2 to listen for node 1's periodic advertisements. This requires node 2 to turn on its radio periodically to listen for an advertisement from node 1.

4.3 Initialization

There is no global initialization. The base station initializes by picking a reservation slot at random and listens for a return message during this slot during the following cycle. If a node is new to the network (i.e. it does not have a schedule), it sets its schedule to all idle and listens for at least one cycle for advertisements. Nodes set their demand to 1 + their current demand. Nodes are fully powered-on in the initialization phase until they acquire an advertised slot. Nodes will choose a parent one hop away toward the base station having the least demand (smallest load). Additional metrics, such as link quality, can be supported by our protocol as well. Once a node has acquired enough reservation slots to meet its initial demand it begins to advertise itself and turn off the radio during idle slots. Note that in the algorithm below, Receive Pending (RP) and Transmit Pending (TP) refer to the same time slot in the parent schedule and child schedule respectively.

4.4 Main Operation

Each node *n* runs the following procedure after initialization:

For each cycle *c*,

Pick an advertisement slot A randomly from idle slots

Pick a reservation slot RP randomly from idle slots

If (supply >= demand)

Radio off during Idle slots

Schedule an advertisement during an A slot

Else

Radio on and listen for advertisements

Schedule a reservation request during a TP slot

For each slot *s*, check power schedule $\Omega(n,s)$

Case (T) – Transmit a message

Case (R) – Receive a message

Case (A) – Broadcast an advertisement for slot RP

Case (RP) – Listen for reservation requests

Case (TP) – Transmit a reservation request

Case (I) – Power down radio or Listen for advertisements if demand is unmet

End

Clear the current A, the previous TP, and the previous RP from the schedule.

End

Note that nodes always begin with *one extra unit of demand*, so bandwidth is immediately available all the way to the sink once a

reservation is made locally. A slot is officially reserved when a parent node receives and accepts a request during its current RP slot. If more than one request is received, the parent accepts the first request. The parent node sends an immediate confirmation acknowledgement and increases its demand by one unit and schedules an R during this slot. The child node increases its supply by one and schedules a T during this slot. The reservation does not have to be renegotiated and remains in effect indefinitely until a child cancels the reservation or the parent times out the reservation if no transmissions occur after some number of cycles. A parent node may also receive a request for decreased demand from a child during a scheduled R slot at which time the parent node sets the R slot to I and decrements its demand.

Notice here the radio on and off times. A node powers off its radio during idle slots when its demand is met. In sensor networks it is the *minority* of nodes that are varying their demand or moving about the network. The design of the protocol puts the onus on the node increasing its demand, the sending node, to listen for advertisements not the receiving node. Advertising nodes need only listen during one time slot, the Receive Pending slot. Similarly, the protocol design is also advantageous for initialization. The onus is on the joining node to listen until it attaches to the network. The advertising nodes need only listen during their Receive Pending slot. The emphasis is on conserving energy.

This algorithm runs on all nodes following initialization. Once all demand has been met at a node, it can power down its radio during idle slots to conserve energy otherwise it listens for advertisements. If more nodes are added to the network, the existing nodes will get new reservation requests and will re-enter the algorithm to maintain a state of extra supply.

4.5 Collisions and Message Loss

The primary function of the local power schedules is to determine when to turn the radio on and off. The schedules derive solely from local information and do not give a global view of the network. Power scheduling relies on the presence of a MAC layer to handle channel access. A simple CSMA MAC will suffice because time division scheduling significantly reduces media contention. The minimum width of a time slot is set to allow transmission of at least two packets in the same slot, so at a minimum, two packets may transmit during the same time slot and the CSMA MAC will handle the channel access.

Collisions can occur in the network due to hidden terminals. This is because carrier sense can only detect potential collisions at the sender and not the receiver. In our protocol, collisions can occur when two children *out of radio range from each other* respond to the same reservation advertisement. When a child fails to acquire a reservation for some number of cycles, it may assume that such collisions are occurring, in which case, the child will send subsequent reservation requests with a probability $P_{request} < 1$. Some collisions may be due to one-off advertisement messages, the slots for which are randomly selected. In this case, message loss will not perennially happen during the same slot.

In the case where a node expects a message from a child (i.e. has R scheduled) and does not receive the message for several cycles, the topology of the network can be assumed to have changed and the R slot can be recycled and demand decremented. Through link layer acknowledgements a parent can implicitly let a child know when messages have ceased to arrive and the child can recycle the T slot and decrement its supply.

5. EVALUATION

The sensor nodes in our implementation are the MICA nodes manufactured by Crossbow running the TinyOS operating system and nesC [10] language developed at UC Berkeley. The TinyOS power management feature exports an interface that allows a program to easily power the radio on and off. MICA has an ATMEGA 128L processor [3], with 128K bytes of programmable flash, 4K bytes of SRAM, and 4K bytes of programmable EEPROM. It uses a TR1000 RF Monolithics [22] radio transceiver with a carrier frequency of 916.50 MHz and transmission rate of 40 Kbps.

5.1 A Simple Application

To verify our protocols we implemented a simple inventory tracking application consisting of 5 stationary nodes, 7 mobile nodes, base station node, and database with user interface. The network continuously receives data at a rate of one packet per minute from mobile nodes used to track various pieces of equipment and forwards the packets to a base station. The area covers two rooms and adjoining corridor. The base station can receive packets wirelessly and send them to a PC through its serial port. The packets received at the PC are then logged to a remote database.

Using a similar setup we ran two simple tests to verify our assumptions about slotted scheduling and end-to-end fairness. One test uses slotted scheduling (via FPS) and the other test does not use slotted scheduling (Naïve store-and-forward). Six nodes send 100 36-byte packets at a rate of one packet per 3.2 seconds. Packets are sent across a 3-hop network with one exception: Node 3 in the Naïve test sends across 2 hops. The test begins after a start message is injected into the network.

Table 2: Average number of packets received at the base station over 10 trials.

Node	1	2	3	4	5	6
FPS	96.00	96.91	96.09	97.45	94.55	97.55
Naïve	28.64	11.55	54.36	18.45	18.18	16.91

Table 2 displays the average number of packets received at the base station over 10 trials. With slotted scheduling almost all of the packets are received. Without slotted scheduling most of the packets are lost due to contention and interference.

This test illustrates that slotted scheduling can provide end-to-end fairness and significantly reduce contention in multi-hop networks.

5.2 Experimental Setup

Figure 4: Network used in MICA experiments.

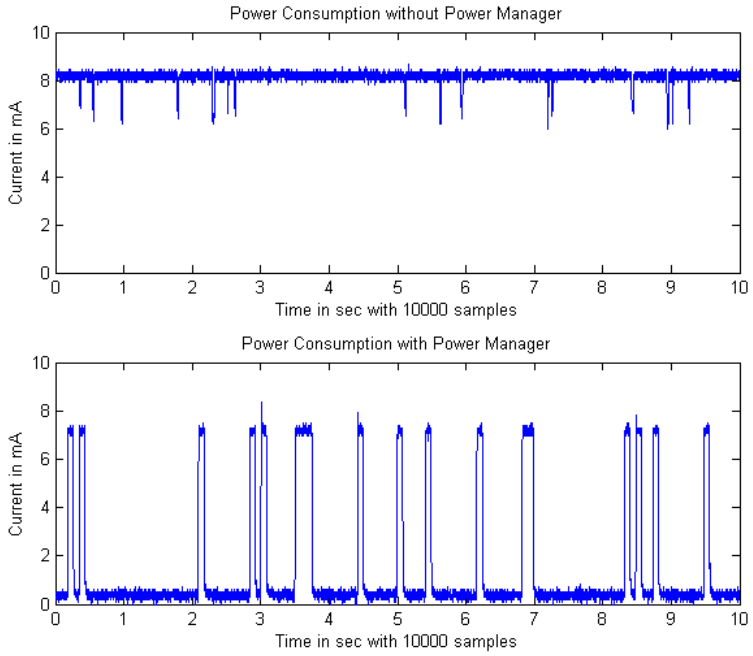

Figure 5: Measured current at node 1 with and without power management over 10 seconds.

To obtain the measurements in sections 5.2 and 5.3, we use a simple 4-node setup where all nodes are in proximity of the base station in order that their messages may be over-heard and logged to a file. Figure 4 shows the 3-hop network we used in energy savings and network adaptation experiments. Node 6 is the sender, node 1 and node 66 are intermediate nodes, and node 0 is the base station node connected to a PC.

5.3 Energy Savings

In this experiment we measure the current at an intermediate node in its steady state while it is forwarding packets. One measurement is taken with power management enabled and one measurement is taken with power management disabled. When power management is enabled we run our protocol and power down the radio during idle time slots. When power management is disabled we run our protocol and leave the radio on during idle time slots. The energy consumption is measured at node 1. Node 6 sends a 36-byte data packet once per cycle, every 2.6 seconds. There are 40 times slot per cycle and each time slot is 65 ms.

Figure 5 shows the current of node 1 in mA without power management (top) and with power management (bottom). Ten seconds of data are captured at a 1 ms sample rate. MICA 128L nodes have a boost converter; we took our measurements with the boost converter turned off. We use an instrumentation amplifier with a gain of 85 and measure voltage across a 1.1 Ohm resistor with an oscilloscope. Hence the current in mA is:

$$\text{Current [mA]} = (\text{voltage [V]} / (85 * 1.1)) * 1000 \quad (2)$$

With power management disabled the average current measured is 8.144 mA. With power management enabled the average current is 1.412 mA. A high-performance AA battery has a capacity of ~1800 mA-hours. So, when power management is disabled the average battery life is 221.02 hours (~9 days) and with power management enabled the average battery life is 1274.79 hours (~53 days).

When in a steady state each node can be in 1 of 5 states in each time slot: Transmit (T), Receive (R), Receive Pending (RP), Advertisement (A), or Idle (I). In the steady state we will not observe Transmit Pending states since no advertisements are being responded to. From analyzing our session log we can determine the schedules of each node.

Table 3: Calculation of duty cycle from observed schedules

N	T	R	RP	A	I	Duty
66	3	2	2	1	32	20%
1	2	1	2	1	34	15%
6	1	0	0	0	39	2.5%

Now we can determine the expected duty cycle for each node. Table 3 displays the count of slot states in the schedule of each node *N*. The duty cycle is the number of active slots per cycle divided by the total number of slots in a cycle. For a cycle length of 40 time slots, the expected duty cycle for Node 1 is 15% (6/40).

To determine the actual duty cycle measured at node 1 over 10 seconds, we count the number of samples that node 1 registers a current greater than a 4 mA and divide this by the total number of samples in the 10 second period:

$$1455 \text{ samples} / 10000 \text{ samples} = 14.55\% \text{ duty cycle}$$

This empirical calculation closely matches the theoretical 15%. Since the 10-second window represents about 3.85 cycles and not an integral number, we expect some small deviation from the theoretical results. As a comparison to the time-based duty cycle computation, we can equally calculate the change in energy expenditure when the power management protocol is enabled.

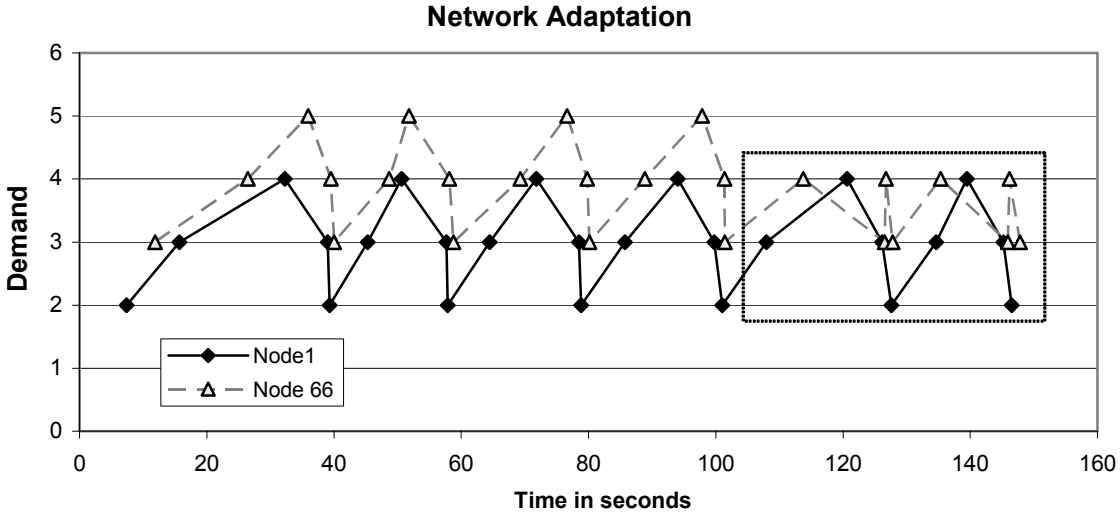

Figure 6: Nodes adapting to changing demand in the network.

We divide the average current measured with power management enabled by the average current measured with power management disabled.

1.412 mA / 8.144 mA = .1734

If the sleeping current draw of the node were zero, we would expect this 17% value to match with the duty cycle. However, the microprocessor still draws current when the radio is powered-down and hence the ratio is slightly higher. We are not certain why the peak current draw using the power management protocol is below that of the disabled case, but were these two values equal, the 17% calculated would rise another few percentage points.

5.4 Network Adaptation

In this experiment we test that the network adapts to varying demand and measure how long it takes the network to react to the fluctuating demand across two intermediate nodes. Figure 4 shows the network used in this experiment. There are 40 time slots per cycle and each time slot is 80ms. Because of the proximity of the request and confirmation messages it was necessary to slow the experiment down, so that the logger at the base station could capture all the messages. The time slot was lengthened and a delay of 5 ms was added between the two messages. All measurements are taken running our protocol with power management enabled.

Recall from Section 4 that demand is increased through request messages. For decreases in demand, we send negative reservation requests during the next scheduled Transmit slot. (The implicit decreased demand mechanism would not have messages for snooping.)

The experiment runs as follows. After each node initializes with a demand of 1, node 6 repeats the following sequence, {send 2 requests to increase demand, wait, send 2 requests to decrease demand, wait}. The longer wait period ameliorates visualization of subsequent messages. We ran the test 10 times collecting ~30 data points per test.

Figure 6 shows the results of one such test. The X-axis is time in seconds. The Y-axis is the current demand at an intermediate node at the time it sends a reservation confirmation message. The two intermediate nodes, node 1 and node 66, are shown. At the beginning of the experiment node 1 has a demand of 1 and node 66 has a demand of 2. The first data point shows Node 1 sending a confirmation to node 6 with a demand of 2. The second data point shows node 66 sending a confirmation to node 1 with a demand of 3 after 4471 ms. At this point node 6 has joined the network and will begin sending alternating demand in steps of 2. There are 2 positive requests followed by 2 negative requests taking 10792 ms, 3681 ms, 462 ms, and 700 ms respectively to percolate to node 66. These delays are known as the *response time.*

To increase its supply, node 1 first turns on the radio, listens for the next advertisement, waits for the advertised reservation slot, and sends the request. Confirmation is received in the same time slot. Delay is related to the length of the slot cycle; here it is 3.2 seconds. In an ideal situation with no packet loss or collisions we expect to wait less than 2 cycles to send a request: at most 1 cycle to hear an advertisement, and at most 1 cycle to meet the reservation slot. If the radio was not powered down during idle slots, there is no wait time for the next advertisement. However, our primary goal is to conserve the battery life, so we use power management.

The square in Figure 6 shows the last 8 requests from the test above. Node 66 is shown responding to requests seemingly out of order. Node 1 sends requests {+2,-2,+2,-2}. Node 66 responds

with {+1,-1,+1,-1,+1,-1,+1,-1}. These anomalies are caused by the random placement of advertisement and reservation slots, which may reorder the requests slightly. Positive requests are sent in the next reservation slot, which has a random position, while negative requests are sent in the next Transmit slot, which has a fixed position. However, the supply and demand mechanism eventually balances out.

The histogram in Figure 7 is collected from 100 data points collected from 7 of our tests. It shows the time difference between node 66 sending a reservation confirmation and node 1 sending a reservation confirmation. It represents the time when node 66 reacts to a request for demand from the sending node 6.

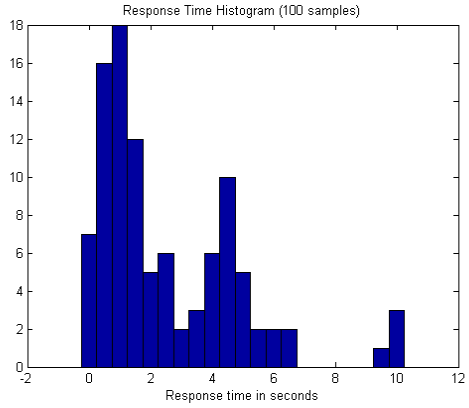

Figure 7: Histogram showing the distribution of response times in the test network.

The majority falls below the expected worst-case response time of 6.4s. The outliers to the right represent lost advertisement messages.

The earlier arrivals to the left are the result of the first-come first-serve nature of packet transmission. There is no notion of bandwidth ownership. Messages are forwarded in the next available Transmission slot

5.5 Latency vs. Energy Tradeoff

In this experiment we use simulation to show the tradeoff between latency and energy. For simplicity, the network used is a 15-node binary tree with a fixed demand of one unit per node, yielding a total network demand of 14. The demand at each node therefore is one unit per child node plus one unit for itself.

Figure 8 is a parametric graph with *m* as the parameter. The X-axis represents latency, measured as the number of slots that occur from the time a message is sent from a leaf node to the time it is received at the base station, as *m* increases from 30 to 1000. In simulation one full cycle passes before forwarding a message one level. For example if $m = 40$ then X = 120 because there are 3 hops between leaf and root. The Y-axis represents the overall duty cycle as *m* increases from 30 to 1000. For example, in this experiment when $m = 40$, we expect to save approximately 90% of the original power in the network to satisfy a demand of 14. Lengthening the cycle time and thus increasing the latency can save more power

6. RELATED WORK

Energy optimizations in wireless sensor networks must be considered at every stage of the hardware and software architectures. In hardware there are energy efficient solutions for the microcontroller, radio, signal processing, sensors, and power supply. Energy-aware software solutions are being addressed as well. There is ongoing research in the areas of energy efficient MAC, routing, topology management protocols, and in-network processing. Energy issues for sensor networks are explored in [8,19,21]. These works make clear one of the most significant sources of power consumption is the radio. Most importantly, because the energy cost of the radio's idle mode differs little from the receive mode the radio must actually be turned off.

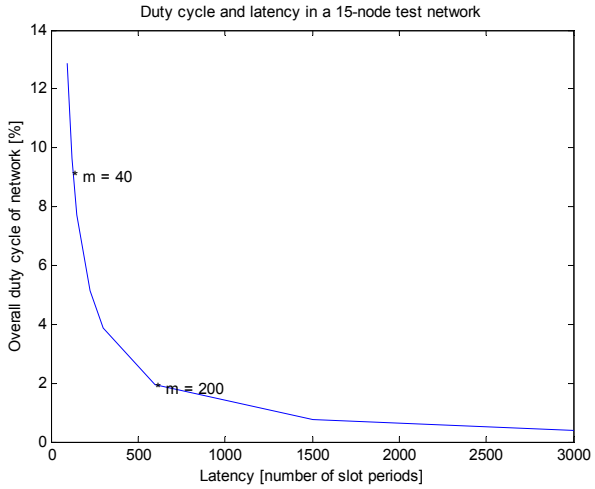

Figure. 8. Overall duty cycle versus latency for varying values of cycle length (*m*)

This is the motivation for the research into energy-efficient MAC layers. There are two broad classes of MAC: contention based [20,30] and TDMA based [26,21]. PAMAS [20] enhances the MACA protocol with the addition of a signaling channel. It powers down the radio when it hears transmissions over the data channel or receptions over the signaling channel. S-MAC [30] incorporates periodic listen/sleep cycles. In order to communicate, neighboring nodes periodically exchange their listen schedules. In the listen phase nodes transmit RTS/CTS packets. They use their normal sleep phase to send data.

Although ours is not a MAC protocol, we draw our inspiration from the TDMA-based energy-aware solutions. TDMA based protocols have natural idle times built into their schedules where the radio can be powered down. Additionally they do not have to keep the radio on to detect contention and avoid collisions. Centralized energy management [1] uses cluster-heads to manage CPU and radio consumption within a cluster. Centralized solutions usually do not scale well because inter-cluster communication and interference is hard to manage. Self organization [26] is non-hierarchical and avoids clusters altogether. It has a notion of super frames similar to TDMA frames for time schedules and requires a radio with multiple frequencies. It assumes a stationary network and generates static schedules. This scheme has less than optimal bandwidth

allocation. Slot reservations can only be used by the node that has the reservation. Other nodes cannot reuse the slot reservation.

Flexible Power Scheduling does not have a strict notion of bandwidth ownership. It is more a reservation for satisfied demand. A message is simply forwarded during the next available Transmit slot and all reserved demand is guaranteed to be met.

Energy-efficient routing in wireless ad-hoc networks has been explored by many authors, see [23,31,15,11] for examples. Topology management approaches exploit redundancy to conserve energy in high-density networks. Redundant nodes from a routing perspective are detected and deactivated. Examples of these approaches are GAF [29] and SPAN [8]. Our approach does not seek to find minimum routes or redundancy. These protocols are designed for systems that require much more general communication throughout the network. Since we are dealing with a more constrained tree topology, we can design a communications protocol with less generality and hence less flooding of messages through the network. As energy considerations are paramount, a specific implementation of a protocol is better suited for our target applications. We also are not interested in a system with a dedicated global initialization phase – demand must be incorporated into the network schedules as data requirements change and as nodes come and go or fail and restart.

In addition to energy conservation, time synchronization is an important consideration for sensor networks. In RBS [9], nodes send reference beacons to their neighbors using physical-layer broadcasts. This methodology indicates that wireless nodes can be synchronized to levels beyond those required by our algorithms. Since we do not require such fine resolutions or the global clock federation we can rely instead on a much simpler local synchronization protocol.

7. CONCLUSIONS AND FUTURE WORK

Flexible Power Scheduling allows us to exploit the natural energy conservation inherent in slotted time division schemes. In this paper we presented a dynamic distributed time division scheduling protocol that facilitates power management by enabling nodes to turn off their radios during idle slots. The algorithm supports variation in traffic load through the use of a supply and demand based protocol. Specifically, the protocol provides local communication schedules for a multi-hop sensor network without global control or re-initialization phase and acts only on locally acquired information.

A novel, two-level architecture combines coarse grain scheduling at the routing layer to plan radio on/off times and fine grain medium access control to provide channel access. Bandwidth allocation is first come first serve. Messages are sent during the next Transmission time slot. They can be aggregated or sent serially bounded by the duration of the time slot.

Implementation and simulation have both shown that power scheduling reduces the energy consumption at all levels of the network and that the network can adapt schedules locally to changing demand. By increasing latency in the network, power requirements can be further reduced. We have shown that we can significantly reduce non-idle slots by locally managing supply and demand. In future work we will investigate the energy savings network-wide or at the macro level.

Also, in future work, we will explore *fractional demand*. Our current algorithm supports integer demand. Consider, for example, an application that sends a message only once per minute. If a schedule has 40 time slots in one cycle and each slot is 50ms long, then a packet will be forwarded every 30 cycles. That leaves 29 cycles where a particular time slot may be used by up to 29 other nodes. If we allocate demand by fractions then downstream nodes can share time slots achieving much better bandwidth utilization.

Flexible Power Scheduling is aimed toward data collection kinds of applications. Currently, communication is assumed to be primarily one-way, toward a base station. Many data collection applications like TinyDB [16] require broadcast or multicast as well and in future work we will extend our algorithm to support such communication.

8. ACKNOWLEDGMENTS

We are much indebted to Rob Szewczyk for his continuous help, support, and advice on the topics of power management and TinyOS. We also thank Cathy Tao for her work on data collection. This work was supported in part by DARPA, grants F33615-01-C-1895 and N6601-99-2-8913, and by Intel Corporation.

9. REFERENCES

[1] K.A. Arisha, M.A. Youssef, M.F. Younis, “Energy-aware TDMA based MAC for sensor networks,” IEEE IMPACCT 2002, New York City, NY, USA, May 2002.

[2] G. Asada, M. Dong, T. S. Lin, F. Newberg, G. Pottie, W. J. Kaiser, H. O. Marcy, "Wireless integrated network sensors: low power systems on a chip," ESSCIRC '98. Proceedings of the 24th European Solid-State Circuits Conference, The Hague, Netherlands, September 1998.

[3] Atmel Corporation: AVR 8-bit RISC processor. http://www.atmel.com/atmel/products/AVR.

[4] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, “Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks,” MobiCom 2001, Rome Italy, July 2001.

[5] W.S. Conner, L. Krishnamurthy, and R. Want, “Making everyday life a little easier using dense sensor networks,” Proceeding of ACM Ubicomp 2001, Atlanta, GA, Oct. 2001.

[6] Crossbow Technology Inc.: http://www.xbow.com/Products/Wireless_Sensor_networks.htm.

[7] Digital Sun, Inc.: http://digitalsun.com

[8] L. Doherty, B.A. Warneke, B.E. Boser, K.S.J. Pister, "Energy and Performance Considerations for Smart Dust," International Journal of Parallel Distributed Systems and Networks, Volume 4, Number 3, 2001, pp. 121-133.

[9] J. Elson, L. Girod and D. Estrin, “Fine-grained network time synchronization using reference broadcasts,” OSDI 2002, Boston, MA, USA, December 2002.

[10] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and C. Culler, “The nesC Language: A Holistic Approach to Networked Embedded Systems," Programming Language Design and Implementation, San Diego, CA, USA, June 2003.

[11] Z. Haas, J. Halpern, and L. Li, “Gossip-based ad-hoc routing,” IEEE INFOCOM 2002, New York, NY, USA, June 2002.

[12] J. Hill, P. Bounadonna, and D. Culler, “Active Message Communication for Tiny Network Sensors,” http://webs.cs.berkeley.edu/tos/media.html.

[13] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K.S.J. Pister, “System architecture directions for networked sensors,” ASPLOS 2000, Cambridge, MA, USA, November 2000.

[14] J.M. Kahn, R.H. Katz, and K.S.J. Pister, “Next century challenges: mobile networking for Smart Dust,” MobiCom 1999, Seattle, WA, USA, August 1999.

[15] B. Karp and H.T. Kung, “GPSR: Greedy Perimeter Stateless Routing for wireless networks,” MobiCom 2000, Boston, MA, USA, August 2000.

[16] S.R. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong, “TAG: a tiny aggregation service for ad-hoc sensor networks,” 5th Symposium on Operating Systems Design and Implementation, Boston, MA, USA, December 2002.

[17] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, J. Anderson, “Wireless sensor networks for habitat monitoring,” WSNA 2002, Atlanta, GA, USA, September 2002.

[18] W. Mangione-Smith and P.S. Ghang, “A low power medium access control protocol for portable multi-media systems,” 3rd International Workshop on Mobile MultiMedia Communications, September 25-27, 1996.

[19] G.J. Pottie, W.J. Kaiser, “Wireless Integrated Network Sensors,” Communications of the ACM, vol. 4, no. 5, May 2000.

[20] C.S. Raghavendra and S. Singh, “PAMAS - Power aware multi-access protocol with signaling for ad hoc networks,” ACM Communications Review, vol. 28, no. 33, July 1998.

[21] V. Raghunathan, C. Schurgers, S. Park, and M.B. Srivastava, “Energy-aware wireless microsensor networks,” IEEE Signal Processing Magazine, vol. 19, no. 2, March 2002.

[22] RF Monolithics: http://www.rfm.com/products/data/tr1000.pdf.

[23] E. M. Royer and C-K. Toh. "A review of current routing protocols for ad-hoc mobile wireless networks,” IEEE Personal Communications, April 1999.

[24] Sensicast Systems: http://www.sensicast.com.

[25] K. Sohrabi, J. Gao, V. Ailawadhi, and G.J. Pottie, “Protocols for self-organization of a wireless sensor network,” IEEE Personal Communications, October 2000.

[26] K. Sohrabi and G.J. Pottie, “Performance of a novel self-organization for wireless ad-hoc sensor networks,” IEEE Vehicular Technology Conference, 1999, Houston, TX, May 1999.

[27] M. Stemm and R. Katz, “Measuring and reducing energy consumption of network interfaces in hand-held devices,” IEICE Transactions on Communications, vol. E80-B, no. 8, pp. 1125-1131, August 1997.

[28] A. Woo and D. Culler, “A transmission control scheme for media access in sensor networks,” in Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking, Rome, Italy, July 2001, ACM.

[29] Y. Xu, J. Heidemann, D. Estrin, “Geography-informed energy conservation for ad hoc routing,” MobiCom 2001, Rome, Italy, July 2001.

[30] W. Ye, J. Heidemann, D. Estrin, “An energy-efficient MAC protocol for wireless sensor networks,” IEEE INFOCOM 2002, New York City, NY, USA, June 2002.

[31] Y. Yu, R. Govindan, and D. Estrin. “Geographical and Energy Aware Routing: a recursive data dissemination protocol for wireless sensor networks,” UCLA Computer Science Department Technical Report UCLA/CSD-TR-01-0023, May 2001.