Ultra-Low Duty Cycle MAC with Scheduled Channel Polling

Wei Ye and John Heidemann

Abstract— **Energy consumption is a critical factor in sensor networks. Since radio costs remain a large part of the energy costs in sensor network hardware, there has been much focus on minimizing energy consumption in radio medium access control (MAC) protocols. Scheduled protocols such as S-MAC, T-MAC, and TRAMA reduce energy consumption by coordinating nodes into sleep/wakeup schedules, allowing them to remain awake only for brief contention periods and to coordinate. Their premise is that the cost of coordination is minimal compared to the savings in coordinated access. Recently a class of low-power listening (LPL) protocols, such as WiseMAC and B-MAC, reduce this overhead by replacing polling in contention periods with very low power "channel active" probes, replacing explicit coordination with per-message coordination via long pre-message preambles. Since testing channels for activity is about 10x less expensive than listening for full contention period, LPL protocols consume less energy than the above scheduled protocols in lightly used networks. However, both of these protocols are limited to duty cycles of 1–2%: scheduled protocols are limited by the delay one can tolerate between schedules, and LPL-based protocols are limited by the increasing transmit costs due to longer preambles. We explore a new approach that can achieve *ultra-low* duty cycles of 0.01–0.1%, potentially reducing energy consumption by a factor of 10–100. To do this, we examine the the fundamental question of the relative benefits of coordinating network access compared to unsynchronized polling. This paper proposes a new MAC protocol based on scheduled channel polling (SCP-MAC). We argue that the use of LPL-like channel probing is necessary, but it must be combined with scheduled access to minimize energy consumption of the radio. We use theoretical analysis to find the best possible operating points for LPL and SCP. Through analysis and testbed experimentation we demonstrate that the use of scheduling in addition to LPL can extend network lifetime by a factor of 2–2.5. In addition, SCP-MAC can reduce transmit latency by avoiding long message preambles, and is more flexible to changing traffic requirements.**

I. Introduction

Energy consumption is a critical factor in sensor networks. Current applications such as habitat monitoring [2], [16] target sensor deployments of months or years.

With small sensor nodes such as Berkeley Motes [9], [12] the radio is a major source of energy consumption. The Chipcon radio draws 22mW when idle or receiving and more when transmitting [11], a power draw about equal to CPU energy consumption and larger than other typical components. Thus it is not surprising that protocols that optimize radio energy consumption have been a major research focus.

The key to reducing radio energy consumption is controlling its power and duty cycle. In this paper we assume fixed hardware and short-range communication. At short ranges, variable transmit power is a second-order effect, so we focus on turning the radio on and off to control energy requirements. Other work has consider routing-layer topology management [28], [22] or application involvement [20]. We focus here on link-layer optimizations because they are transparent to higher layers and complement work at layers that are above or below.

Two primary approaches have been considered in the MAC layer. The first approach uses *scheduled* protocols such as S-MAC [29], [30], T-MAC [25], and TRAMA [19] adopt common sleep/wakeup schedules, nodes remain awake only for brief contention periods to coordinate. Scheduling allows nodes to operate in low duty cycles. Another major benefit of scheduling is that a sender can efficiently transmit – it only wakes up and sends when a receiver is listening. The premise of the scheduled protocols is that the cost of coordination is minimal compared to reductions in time spent in listening for potential transmissions. In addition, these approaches also take steps to reduce collisions from concurrent transmission and overhearing of packets sent to others.

A second approach is *low-power listening* (LPL), present in WiseMAC [6] and B-MAC [18]. LPL allows a sleeping node to check channel activity with a very brief, low power "channel active" probes. We also call this action channel polling in this paper. These protocols replace the relatively long wakeup interval (including contention) in S-MAC and T-MAC with a very short channel polling time. In these LPL protocols, nodes randomly poll the channel with a pre-defined polling period. To wake up a receiver, a sender uses a long preamble before each packet, which is at least the length of the polling period. Therefore, explicit coordination is unnecessary, since all neighbors will hear the preamble and wake for the message. Since testing channels for activity is about 10x less expensive than listening for full contention period, LPL protocols consume less energy than the above scheduled protocols in lightly loaded networks.

However, both types of existing protocols are limited to duty cycles of 1–2%. Scheduled protocols are limited by the relatively long wakeup interval and the delay one can tolerate between schedules. LPL-based protocols are limited by the increasing transmit costs due to longer preambles. In this paper, we design a new MAC protocol that can achieve *ultra-low* duty cycles of 0.01–0.1%, potentially reducing energy consumption by a factor of 10–100.

Our protocol employs a new approach called scheduled channel polling (SCP). It combines the strengths of scheduling and low power listening. Some researchers [18] pointed out that the overhead in schedule synchronization may largely offset its benefits. The conclusion was drawn based on an unoptimized implementation of schedule synchronization in S-MAC. This paper carries out thorough theoretical analysis and experiments

W. Ye (weiye@isi.edu) and J. Heidemann (johnh@isi.edu) are with the Information Sciences Institute (ISI), University of Southern California (USC).
This research is partially supported by the National Science Foundation (NSF) under grant number NeTS-NOSS-0435517, "Sensor Networks for Undersea Seismic Experimentation", and by a hardware donation from Intel Corporation. It is also partly supported by CiSoft (Center for Interactive Smart Oilfield Technologies), a Center of Research Excellence and Academic Training and a joint venture between the University of Southern California and Chevron Corporation.

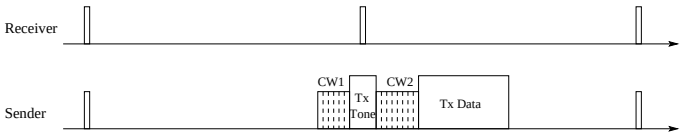

Fig. 1. Data transmission with synchronized channel polling.

on how to optimize the synchronization procedure. Our results show that the synchronization cost is minimal if we select optimized parameters.

There are two main contributions in this paper. First, we propose a new MAC protocol based on scheduled channel polling (SCP-MAC). The central novelty in SCP-MAC is the combination of scheduling and polling; we also describe novel additions including split contention windows and piggybacked synchronization. Second, we examine the the fundamental question of the relative benefits of coordinated network access compared to unsynchronized polling. We argue that the use of LPL-like channel probing is necessary, but it must be combined with scheduled access in order to operate in ultra-low duty cycles. We use theoretical analysis to find the best possible operating points for LPL and SCP. We demonstrate that SCP-MAC can operate for 2–3 times longer than to LPL-based MACs for the same energy budget when each is tuned for a completely periodic workload. Scheduled polling as a better match for *unpredictable* traffic when tuned for low-duty cycle operation. LPL energy consumption suffers when mismatched to changing traffic loads because of preamble length. By contrast, SCP only pays penalty in latency, not in energy, and even the latency penalty can be eliminated with algorithms such as adaptive listen. We show in testbed experiments that LPL consumes 8 times more energy than SCP when presented with short-term bursty traffic.

II. Design of Scheduled Channel Polling

As described above, reducing the duty cycle is key to conserving energy in frequently idle networks. Current protocols are limited to duty cycles of 1%; with SCP-MAC we seek to reduce the duty cycle by a factor of 10 by combining very short channel polling in LPL with scheduling. A secondary goal in SCP-MAC is to provide efficient operation over a wide range of traffic conditions at run time.

A. SCP-MAC Overview

The basic scheme of SCP-MAC combines the strengths of channel polling and scheduling. Channel polling minimizes the cost of wakeup checking for the presence or absence of network activity rather than checking what that activity is. Similar to low power listening (LPL), SCP puts nodes into periodic sleep state when there is no traffic, and they perform channel polling periodically. Unlike LPL, we synchronize the polling time of all neighboring nodes. The major advantage of synchronized polling is that a very short wake-up tone can be sent to wake up a node. The short wakeup tone largely reduces the overhead of transmitting long preambles in LPL.

Using short wakeup tone also makes SCP-MAC more robust in the face of varying traffic load. The performance of LPL is sensitive to the channel polling period. The optimal value to minimize energy consumption requires knowledge of network size and completely periodic traffic. While traffic in some classes of sensor network applications is completely periodic, a much larger set of applications mix periodic and bursty traffic or consist of unpredictable traffic mixes. A worst case is a monitoring application where there is no traffic to send most of the time, but bursts of activity when a target is detected. Such a network does not have a single good operating point, since it just employ a low duty cycle to match long idle periods, but then is penalized with long preambles and expensive transmission costs during busy cycles. Although one could imagine a LPL-based network dynamically adjusting its configuration (such adaptation has been done in scheduled MACs [15]), such adaptation must be conservative and will likely have large transition costs. While we cannot characterize all sensor network applications, experience in the Internet suggests that that traffic is very bursty across a wide range of timescales [13].

Figure 1 illustrates the wakeup and data transmission scheme we propose for SCP-MAC. When a node has a packet to send, it waits in sleep state until the receiver's time to poll the channel. It will send a short *wakeup tone* to activate the receiver. Before sending the tone, it performs carrier sense within the first contention window (denoted as CW1 in the figure). As with typical CSMA protocols, nodes randomly select a slot in a fixed-length contention window to reduce chances of collision. If the node detects channel idle it will send the wakeup tone. Otherwise, it goes back to sleep and will perform regular channel polling. After a sender wakes up a receiver, it enters the second contention window (CW2 in Figure 1). If the node still detects channel idle in the second contention phase, it starts sending data.

The major advantage to separate the contention phases for tone and packet is to achieve lower collision probability with shorter overall contention time. The collision probability is about inversely proportional to the contention window size. Suppose we have m slots in a single contention window, the collision probability is roughly proportional to $1/m$. If we split the window into two with half the size, the collision probability probability will become proportional to $4/m^2$. Therefore when $m > 4$, two-phased contention will have better performance than the single-phased one. Alternatively, we can use fewer contention slots (to save energy) to achieve the same collision performance.

The reason that we can split the contention with fewer overall slots is that SCP tolerates the collisions on tone transmissions—the wake-up tone must indicate network activity, not actually send data. Thus, we can use a very small contention window for phase one. After phase one, only surviving nodes enter the second phase. With fewer competing nodes, the collision probability on data transmission can be largely reduced. Our current implementation defaults to use 8 and 16 slots for tone and data contention windows, respectively.

On top of this basic wakeup and contention mechanism, SCP-MAC includes several algorithms from prior MAC protocols as compile- and run-time options. These extensions can be configured to match the requirements of different applications or traffic patterns. We include RTS-CTS exchanges [1] to support

applications that may have high levels of network contention.

We have extended overhearing avoidance [23], [29] to work both with and without RTS-CTS. When RTS-CTS is enabled, overhearing avoidance is performed the same way as that in S-MAC [29]. When RTS-CTS is disabled, we propose to perform overhearing by examine packet headers. After a node receives the MAC header of a packet, it immediately examine the destination address. If it is a unicast packet destined to another node, it abandons receipt of the rest of the packet and places the radio into sleep. Although we have not validated the checksum of the packet header at this stage, if we sleep because of a corrupted header we would have eventually dropped the packet anyway.

We support adaptive listen [30] to automatically adapt to bursty traffic. After transmission of a packet the MAC layer immediately polls the channel for additional traffic. This approach is similar to S-MAC, but replaces the more expensive contention intervals with wake-up tone transmission and LPL-like channel polling.

We also plan to add support for fast-path schedule allocation [15], where a user can coordinate the schedules of all nodes along a path to avoid all schedule-based delay. As with adaptive listen, fast-path scheduling can exploit channel polling in place of full contention.

B. Synchronization Mechanism

Synchronizing schedules with its neighbors is an essential component of SCP-MAC. To coordinate, all nodes broadcast their schedule information to their neighbors every *synchronization period*. How often synchronization is required is a function of clock drift and node density but synchronization is required only every 10–60 minutes.

SCP-MAC uses two approaches for sending schedule information. The first one is to piggyback it onto data packets when they are present. For example, periodic sensing data reports from each node can carry such information. The overhead of of piggybacking is very small—two bytes for unicast message, or free on broadcast messages in our implementation (details are in Section IV-C). If data traffic is more frequent than the synchronization interval, explicit synchronization can be suppressed. We evaluate the optimal frequency of synchronization in Section III-C.1). With typical clocks, synchronization is required very infrequently (tens of minutes); we expect most applications will be able to make use of piggybacking.

Section III-C also compares the energy consumption with and without piggybacking. This subsection investigates the relationship of the synchronization period and the wake-up tone duration.

There are several factors that affects the synchronization period and the wake-up tone length. Among them, the clock drift rate is a fundamental physical limit. Current CMOS crystal oscillators, such as those used on the UC Berkeley motes, have a drift rate of 30–50 parts per million (ppm) [12]. To accommodate potential clock drift we extend the wake-up signal by a guard time.

Denote the synchronization period as T_{sync} (a configuration parameter) and the clock drift rate as r_{clk}. The maximum clock difference between a sender and a receiver is

$$t_{diff} = 2T_{sync}r_{clk} \tag{1}$$

where the factor of two reflects the worst case when each node's clock drifts in the opposite direction.

Since the relative time difference between two nodes can be in two directions, the guard time needs to be twice t_{diff}. If a node has n neighbors, each of them will send SYNC packets at the period of T_{sync}. Since every SYNC packet re-synchronizes all nodes in the neighborhood, $(n+1)$ nodes effectively reduce the clock drift by $(n+1)$ times. Thus the guard time becomes

$$t_{guard} = 2t_{diff} = \frac{4T_{sync}r_{clk}}{n+1} \tag{2}$$

The duration of the wake-up tone is the guard time plus a short, fixed time

$$t_{tone} = \frac{4T_{sync}r_{clk}}{n+1} + t_{mtone} \tag{3}$$

where t_{mtone} is the time required to detect the tone. Since the time needed for the receiver to sample the channel (not including the radio transition time) and determine channel activity is around 0.5–2ms, depending on the radio speed, carrier sense algorithm, and channel condition, we simply set $t_{mtone} = 2\text{ms}$ for easy analysis.

There is a trade-off in determining T_{sync}: increasing T_{sync} reduces the energy cost of sending SYNC packets, but increase the cost on guard time. In Section III-C we evaluate the optimal T_{sync} to minimizes the energy cost.

III. Analysis of Energy Performance

This section analyzes the energy consumption in low duty cycle MAC protocols. It compares the two schemes of channel polling: random and synchronized. We first describe the models and metrics used in our energy analysis.

A. Models and Metrics

Our analysis only considers a local network, where all nodes can directly hear from each other. Each node has n neighbors, and n is referred to as the neighborhood size of a node. The traffic is generated by each node, which periodically sends a data packet. The packet can be either broadcast or unicast. For now we only consider broadcast. SCP-MAC should have much better savings in unicast, as it has overhearing avoidance. The radio is in any of the four states: transmitting, receiving, listening, and sleeping, each with different power consumption (energy consumption per unit time) of P_{tx}, P_{rx}, P_{listen} and P_{sleep} respectively. The channel polling is different than normal listening in that the radio is turned on very briefly to detect possible wake-up signals. Its duration, denoted as t_{p1}, consists of the radio transition time from sleep to listen and the sampling time to detect channel activity. We denote the average power consumption in channel polling as P_{poll}. The radio transitions can be ignored in other states.

Both LPL and SCP are contention-based MACs, and transmitting a data packet requires carrier sense. To simplify the analysis, this section assumes that there is only one contention

Symbol	Meaning	Typical Value
P_{tx}	Power consumption in transmitting	60mW
P_{rx}	Power consumption in receiving	45mW
P_{listen}	Power consumption in listening	45mW
P_{sleep}	Power consumption in sleeping	90μW
P_{poll}	Avg. power consumption in polling channel	5.75mW
t_{p1}	Time needed to poll channel once	3ms
T_p	Channel polling period	Varying
T_{data}	Data packet period	Varying
r_{data}	Data packet rate ($1/T_{data}$)	Varying
L_{data}	Data packet length	50B
t_{cs1}	Average carrier sense time for one packet	7ms
t_B	Time to transmit or receive a byte	416E-6s
n	Number of neighbors	10

TABLE I
SYMBOLS USED IN ENERGY ANALYSIS

phase in SCP-MAC, and the contention window size is the same as that of the LPL. We denote the average time in carrier sense as t_{cs}. After carrier sense, a node first sends a wake-up tone and then followed by the real packet. We denote the transmitting time for the tone and packet as t_{tx}. Besides carrier sense and transmitting, a node can either poll the channel, receive a packet (or tone) or sleep. We denote the time a node in these states as t_{poll}, t_{rx} and t_{sleep} respectively. In our analysis, all the above time values are normalized to one second. They represent the fractions of time in one second the node in different states. We refer to them as normalized time. Table I lists some symbols used in our analysis.

For both LPL and SCP, the average energy consumption per second, *i.e.*, average power consumption, on each node can be computed as

$$\begin{aligned} E &= E_{cs} + E_{tx} + E_{rx} + E_{poll} + E_{sleep} \\ &= P_{listen}t_{cs} + P_{tx}t_{tx} + P_{rx}t_{rx} \\ &\quad + P_{poll}t_{poll} + P_{sleep}t_{sleep} \end{aligned} \tag{4}$$

We next derive the average power consumption for both random and scheduled channel polling schemes.

B. Random Channel Polling: LPL

In LPL, nodes randomly wake up. A sender wakes up a receiver by sending a long preamble before each packet. (Each packet has a short, fixed preamble to synchronize the transmitter and receiver. For simplicity, we considered it as part of a packet. The length of each packet in our analysis includes 10B preamble.) The duration of the preamble should be at least the same as the polling interval T_p, and thus the preamble length is

$$L_{preamble} = \frac{T_p}{t_B} \tag{5}$$

where, t_B is the time needed to transmit or receive a byte.

Before sending the preamble, a node needs to perform carrier sense for each data packet. Recall that the average carrier sense time is t_{cs1}. The normalized time a node spends in carrier sense is

$$t_{cs} = \frac{t_{cs1}}{T_{data}} = t_{cs1}r_{data} \tag{6}$$

where r_{data} is the rate of sending data packet on each node. The normalized time a node is in transmitting state is

$$\begin{aligned} t_{tx} &= (L_{preamble} + L_{data})t_B r_{data} \\ &= (T_p + L_{data}t_B)r_{data} \end{aligned} \tag{7}$$

The second line in the above equation is due to (5).

Assume each node periodically generates packets at the same rate of r_{data}. A node will periodically receive n packets from its n neighbors. The normalized time it is in receiving state is

$$t_{rx} = nt_{tx} = n(T_p + L_{data}t_B)r_{data} \tag{8}$$

The normalized time that a node uses to poll the channel is

$$t_{poll} = \frac{t_{p1}}{T_p} \tag{9}$$

The normalized time in sleep state is the portion in a second that a node's radio is not in the above active states.

$$t_{sleep} = 1 - t_{cs} - t_{tx} - t_{rx} - t_{poll} \tag{10}$$

Substituting Equations (6)–(10) into (4) and using Equation (5), we obtain the energy consumption with random channel polling as

$$\begin{aligned} E_r &= P_{listen}t_{cs1}r_{data} + (P_{tx} + nP_{rx})(T_p + L_{data}t_B)r_{data} \\ &\quad + P_{poll}t_{p1}/T_p \\ &\quad + P_{sleep}(1 - t_{cs1}r_{data} - (n+1)(T_p + L_{data}t_B)r_{data} \\ &\quad - t_{p1}/T_p) \end{aligned} \tag{11}$$

Assuming data length is fixed, we can see from the above equation that the energy consumption of a node changes as a function of its neighborhood size n, data rate r_{data}, and channel polling period T_p.

Equation (11) also shows a tradeoff with T_p: reducing T_p reduces the cost of polling the channel, but it increases the energy spent in transmitting and receiving. An interesting question is, what is the optimal value of T_p that minimizes the energy consumption when n and r_{data} are fixed? We can obtain the answer by solving the following equation.

$$\frac{dE_r}{dT_p} = 0 \tag{12}$$

Substituting Equation (11) into (12), we have the optimal value of T_p for random polling as

$$T^*_{p,r} = \sqrt{\frac{(P_{poll} - P_{sleep})t_{p1}}{r_{data}(P_{tx} + nP_{rx} - (n+1)P_{sleep})}} \tag{13}$$

Figure 2 shows $T^*_{p,r}$ as a function of the data rate using the typical values shown in Table I. When $T_{data} = 300$s, $T^*_{p,r} = 100$ms (the same as the default value in LPL as shown in its Table 3). When $T_{data} = 100$s, $T^*_{p,r} = 58$ms.

The optimal energy consumption in the random channel polling scheme is the one expressed by Equation (11) when $T_p = T^*_{p,r}$. We will show a numerical result in Section III-C.2.

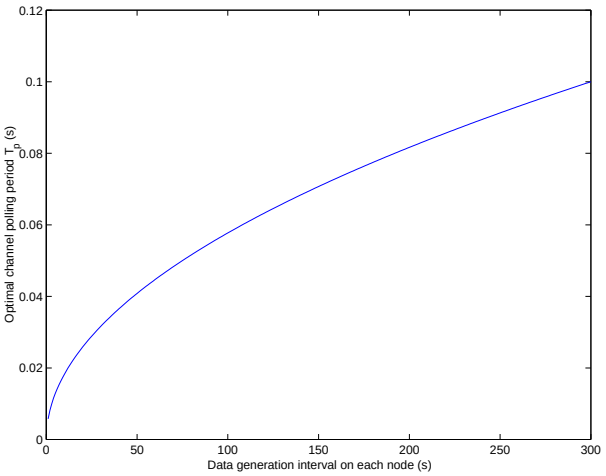

Fig. 2. Optimal channel polling period in LPL. Neighborhood size is 10.

Symbol	Meaning	Typical Value
T_{sync}	SYNC packet period	Varying
r_{sync}	SYNC packet rate ($1/T_{sync}$)	Varying
L_{sync}	SYNC packet length	18B
L_{sB}	SYNC bytes piggybacked to data	2B
t_{mtone}	Minimum duration of wake-up tone	2ms
t_{tone}	Duration of wake-up tone	Varying

TABLE II
ADDITIONAL PARAMETERS IN SCP-MAC

C. Scheduled Channel Polling: SCP

Now we look at the energy consumption in the scheduled channel polling scheme. In this scheme, a node will roughly synchronize with its neighbors on channel polling. An additional cost in such a network is exchanging synchronization information among neighbors. However, there is no need to send long preamble before each packet to wake up a receiver. SCP-MAC uses a short wake-up tone instead. Table II shows additional parameters in SCP-MAC.

Equation (4) is still applicable to the scheduled channel polling scheme if we include the cost of sending and receiving synchronization information. To reduce the synchronization cost, such information should be piggybacked to data packets if possible. In the following two subsections, we investigate the two cases with and without piggybacking separately.

C.1 Best Case: Energy Consumption With Perfect Piggybacking

Given the fact that many types of data transmissions in sensor networks are periodic, synchronization information can be easily piggybacked on data. For example, all synchronization information can be piggybacked if $r_{data} \leq r_{sync}$. This subsection investigates the energy consumption for this case and assumes $r_{data} = r_{sync}$.

Since the transmission rate does not change, the normalized carrier sense time is still expressed by Equation (6). The transmission time now is

$$t_{tx} = (t_{tone} + L_{sB}t_B + L_{data}t_B)r_{data} \tag{14}$$

Similarly, the reception time is

$$t_{rx} = n(t_{tone} + L_{sB}t_B + L_{data}t_B)r_{data} \tag{15}$$

The channel polling time and the sleep time can still be represented by Equations (9) and (10).

Substituting Equations (6), (14), (15), (9) and (10) into (4), we obtain the energy consumption of the scheduled channel polling with piggybacked synchronization as

$$\begin{aligned} E_{sp} \quad = \quad & P_{listen}t_{cs1}r_{data} \\ & + (P_{tx} + nP_{rx})(t_{tone} + L_{sB}t_B + L_{data}t_B)r_{data} \\ & + P_{poll}t_{p1}/T_p \\ & + P_{sleep}[1 - t_{cs1}r_{data} \\ & \quad - (n+1)(t_{tone} + L_{sB}t_B + L_{data}t_B)r_{data} \\ & \quad - t_{p1}/T_p] \end{aligned} \tag{16}$$

Ideally, with the periodic traffic from all neighbors, a node should only poll the channel when there is a transmission from a neighbor. Thus the optimal polling period T_p for scheduled polling is

$$T^*_{p,sp} = \frac{1}{n(r_{data})} \tag{17}$$

The optimal energy consumption can be obtained by substituting Equation (3) into (16) and letting $T_p = T^*_{p,sp}$ and $T_{sync} = 1/r_{data}$. It is only a function of r_{data}. A numerical result will be shown in Section III-C.2.

Piggybacking does require a slightly larger header to include clock and schedule information. This cost is reflected in L_{sB}. We can evaluate the overhead of piggybacking by comparing E_{sp} to $E_{sp\text{-}free}$, the ideal case where L_{sB} is set to zero. For the 10 neighbor scenario, the overhead of piggybacking is always less than 2%, and is less than 1% when $T_{data} > 150$s, with overhead dropping at longer data intervals.

C.2 Worst Case: All Synchronization via SYNC Packets Without Piggybacking

In this case, nodes spend more time in transmitting and receiving SYNC packets, since the packet transmission rate has been increased by r_{sync}. Here we assume the worst case where no SYNC packets can be piggybacked on data packets.

Since SYNC packets also require carrier sense, the normalized time in carrier sense is

$$t_{cs} = t_{cs1}(r_{data} + r_{sync}) \tag{18}$$

where r_{data} is the data packet rate, and r_{sync} is the SYNC packet rate.

After carrier sense, a node first sends a wake-up tone to wake up the receiver and then sends the packet. The normalized time in transmitting state is

$$t_{tx} = (t_{tone} + L_{data}t_B)r_{data} + (t_{tone} + L_{sync}t_B)r_{sync} \tag{19}$$

Compared with Equation (7), the long preamble is replaced with a short tone, but the packet rate is increased by r_{sync}.

Assuming all the data packets are broadcast, we have the normalized time in receiving state as

$$t_{rx} = n(t_{tone} + L_{data}t_B)r_{data} + n(t_{tone} + L_{data}t_B)r_{sync} \quad (20)$$

The normalized time that a node polls the channel and sleep can still be expressed by Equations (9) and (10) respectively. But the values of t_{cs}, t_{tx} and t_{rx} in (10) are replaced by Equations (18)–(20).

Substituting Equations (18)–(20) and (9)–(10) into (4), we have the energy consumption in scheduled channel polling with independent SYNC packets as

$$\begin{aligned} E_{snp} &= P_{listen}t_{cs1}(r_{data} + r_{sync}) \\ &\quad + (P_{tx} + nP_{rx})(t_{tone} + L_{data}t_B)r_{data} \\ &\quad + (P_{tx} + nP_{rx})(t_{tone} + L_{sync}t_B)r_{sync} \\ &\quad + P_{poll}t_{p1}/T_p \\ &\quad + P_{sleep}[1 - t_{cs1}(r_{data} + r_{sync}) \\ &\qquad - (n+1)(t_{tone} + L_{data}t_B)r_{data} \\ &\qquad - (n+1)(t_{tone} + L_{sync}t_B)r_{sync} \\ &\qquad - t_{p1}/T_p] \end{aligned} \quad (21)$$

If we ignore the energy consumption in sleep state, the energy consumption with scheduled channel polling changes monotonically with the polling period T_p. The larger the T_p, the smaller the E_{snp}. This is different than the random channel polling as shown in Equation (11), since here the cost of sending and receiving a packet does not change with T_p. Ideally, with the periodic traffic from all neighbors, a node should only poll the channel when there is a transmission from a neighbor. Thus the optimal polling period for scheduled polling with independent SYNC packets is

$$T^*_{p,snp} = \frac{1}{n(r_{data} + r_{sync})} \quad (22)$$

Now we go back to the question "what is the optimal synchronization period T_{sync} that minimizes E_{snp}?" To answer the question, we substitute Equations (3) and (22) into (21), and solve the following equation

$$\frac{dE_{snp}}{dT_{sync}} = 0 \quad (23)$$

Thus the optimal T_{sync} is obtained as

$$T^*_{sync} = \sqrt{\frac{n(n+1)(E_l + P_t t_t + E_p)}{2r_{data}r_{clk}P_t}} \quad (24)$$

where

$$\begin{aligned} E_l &= P_{listen}t_{cs1}, \\ P_t &= P_{tx} + nP_{rx} - (n+1)P_{sleep}, \\ t_t &= t_{mtone} + L_{sync}t_B, \\ E_p &= n(P_{poll} - P_{sleep})t_{p1}. \end{aligned}$$

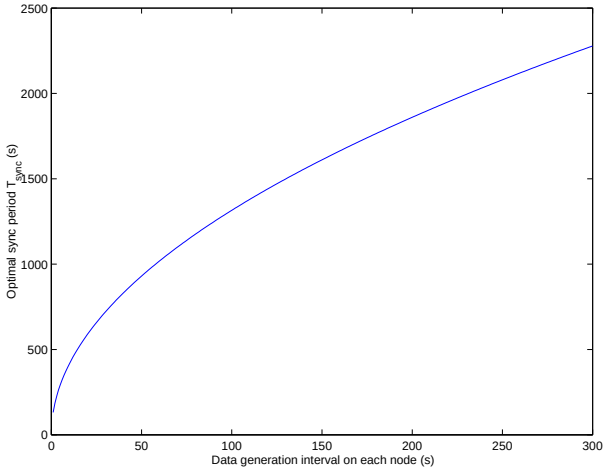

Fig. 3. Optimal SYNC period for SCP-MAC.

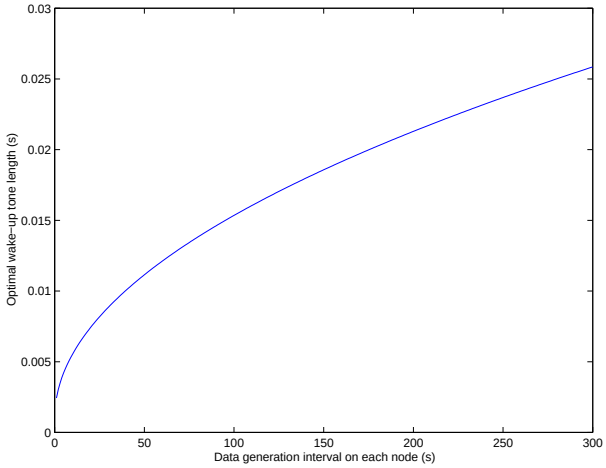

Fig. 4. Optimal wake-up tone length for SCP-MAC.

Once T^*_{sync} is known, we can obtain the optimal tone duration by substituting Equation (24) into (3), which is

$$t^*_{tone} = \frac{4T^*_{sync}r_{clk}}{n+1} + t_{mtone} \quad (25)$$

Figures 3 and 4 show the optimal synchronization period and the optimal wake-up tone length respectively.

From these results so far we can make several observations about how the parameters of a scheduled MAC compare to an unscheduled one. First, Figure 3 suggests that the clock synchronization can be quite rare, about $7\times$ data transmission frequency during light loads ($T_{data} = 300$s) to $16 \times T_{data}$ during heavier loads ($T_{data} = 50$s). This observation suggests that synchronization overhead can be low. Second, clock synchronization and scheduled polling allows *much* shorter preambles than are possible with unsynchronized media access. Finally, when piggybacking is used, synchronization happens "for free" on top of data, allowing much shorter tone lengths because of careful clock synchronization. The cost of piggybacking is also quite low, only 2 bytes.

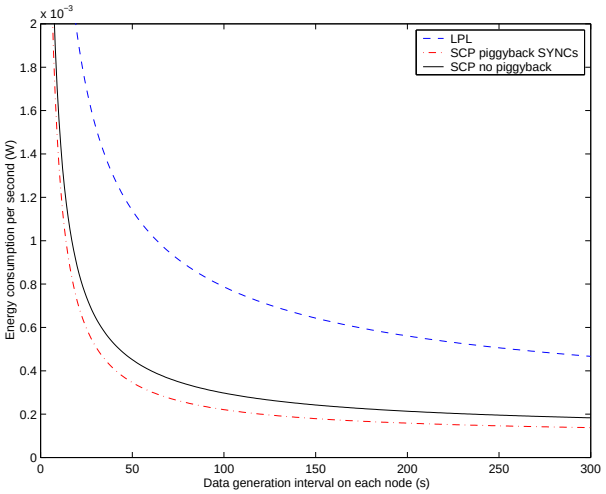

Fig. 5. Comparison of optimal energy consumptions in random and scheduled channel polling.

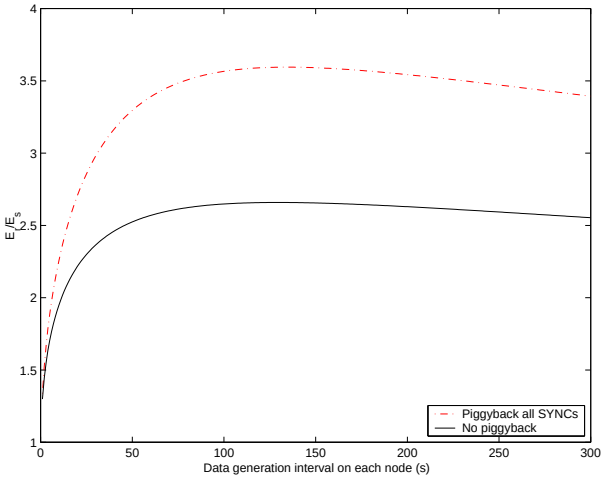

Fig. 6. Gain in energy consumption of scheduled polling over random polling.

The optimal energy consumption in scheduled channel polling with independent SYNC packets can be obtained by substituting Equations (22)–(25) into (21).

Figure 5 compares the optimal (minimum) energy consumptions in the three cases we have analyzed: random channel polling, scheduled channel polling with all sync information piggybacked on data, and scheduled channel polling with all independent SYNC packets. We can see that the random channel polling consumes the most energy. Scheduled polling with sync information piggybacked on data consumes the least energy. In a real network where only partial SYNC packets can be piggybacked on data, its energy consumption will between two lines of piggybacking and no piggybacking in the figure. Figure 6 shows the actual gain in energy of scheduled polling over random polling.

IV. Protocol Implementation

We have implemented SCP-MAC in TinyOS [10] over the Mica2 motes [12]. To provide a clean comparison of LPL and scheduled LPL, we implement scheduling as a layer over basic LPL. We describe this architecture, how it integrates with TinyOS, and details about piggybacking synchronization information next.

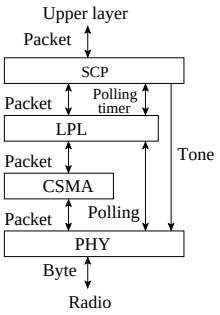

Fig. 7. Software architecture of the SCP-MAC implementation in TinyOS.

A. Software Architecture

We first describe the software architecture of SCP-MAC in TinyOS. Our implementation emphasizes on modular design and reusable components. Figure 7 shows the general architecture, illustrating the relationships of the major components. In addition to these modules, several parameters and options are configurable at compile time, including RTS-CTS handling, overhearing avoidance, and adaptive listen. We have implemented all these components; a version of this implementation is available from the authors at their web site.

At the bottom is the physical layer (PHY). It handles the radio states (sending, listening, receiving, off, and warming up). On packet transmission, it passes each byte to the radio at the its transmission speed. On reception, it buffers all bytes from a packet and and passes the packet to the MAC when complete. It also implements and exports interfaces for physical carrier sense, transmission of the wakeup tone, CRC check, and time stamping on transmitting and receiving of packets (for accurate time synchronization). For performance measurement, the PHY can record time spent in each radio state. The PHY module is designed to be MAC-independent and able to support contention-based or TDMA protocols, so it leaves backoff and similar functions to higher layers.

Above the PHY, we first implemented a basic CSMA protocol. Since both LPL and SCP are contention-based protocols, the CSMA component can be used by both of them. It includes preamble length as a parameter to packet transmission, allowing support for LPL. The CSMA is responsible for performing carrier sense and random backoff. It also includes, as a compile-time option, support for full RTS-CTS-DATA-ACK or simply DATA-ACK exchanges for unicast traffic. If ACKs are enabled, it does retransmission of unicast packets. It also includes virtual carrier sense (avoiding transmission during control message exchanges) and overhearing avoidance.

LPL is implemented on top of the CSMA component. Its major purpose is to periodically poll the channel and send the radio to sleep when there is no activity. It adjusts preamble lengths on transmitted packets to ensure they intersect with polling fre-

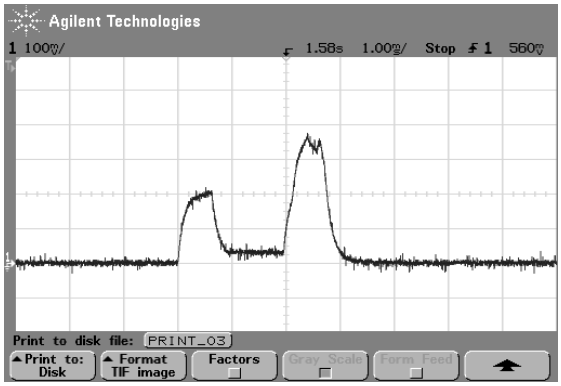

Fig. 8. Channel polling process implemented in SCP-MAC.

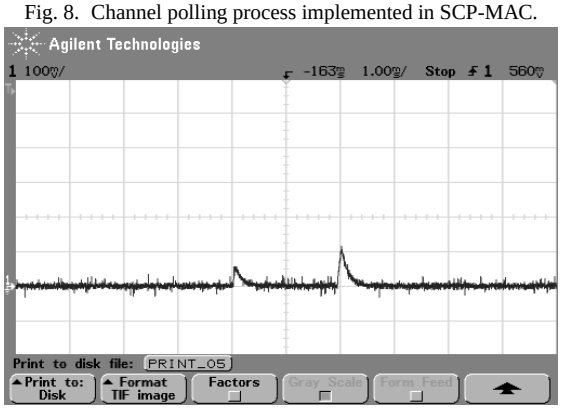

Fig. 9. CPU overhead on timer firing events.

quency, and coordinates concurrent polling and transmission. To support SCP, LPL exports interfaces to query and adjust channel polling times.

Scheduling is implemented above the LPL module in the SCP module. It uses basic LPL to bootstrap schedules with SYNC packets. Once it has synchronized polling times with neighbor nodes, it switches to reduced-length preambles and wake-up tone transmission. It coordinates packet transmission timing to ensure short-duration wake-up tones are sent when neighbors are listening. It also implements the SCP-level randomized contention window before wake-up, which combines with CSMA-level contention for the data transmission to provide two independent contention periods. Finally, it includes a number of optional compile-time optimizations, including SYNC piggybacking on broadcast data packets. (As future work we expect to also piggyback SYNC information in unicast exchanges.)

All three MAC components, CSMA, LPL and SCP export the same interface for message transmission and reception. An application can easily switch MAC protocols by changing its component wiring. Such implementation promotes component reuse. This architecture also provides a common foundation for our performance evaluation in Section V.

B. Interaction with TinyOS

Although we control radio activity, we depend on TinyOS for CPU power management and timers. Our PHY layer coordinates with TinyOS to allow the CPU to sleep when the radio is not needed.

Based on our PHY and the CPU power management component in TinyOS and the implementation from B-MAC we implemented the low-power channel polling. Figure 8 shows the current draw for channel polling captured by an oscilloscope (each x-axis tick is 1ms, each y-axis tick is 4mA). Our implementation provides similar results as the B-MAC implementation ([18], Figure 3).

We added a new timer implementation to TinyOS to add support for dynamically adjusting timer values and asynchronous, low-jitter triggers. The synchronized channel polling in SCP-MAC requires to receive the timer firing events with very low jitter to minimize synchronization errors. Our timer implementation is based on the 8-bit hardware counter on Mica2. This timer runs independently from the CPU, allowing the CPU to sleep when no other activity is present. Because this timer uses an 8-bit counter running at 1kHz, the timer overflows and must wake-up the CPU four times per second. We measured the energy cost of this event via an oscilloscope in Figure 9. Compared to the cost of activating the radio (Figure 8, with the same scale), the energy requirements to maintaining the timer is minimal.

C. Efficient Piggybacking of Synchronization Information

To minimize the cost of synchronization we wish to avoid explicit SYNC packets. One SCP-MAC optimization is to piggyback synchronization information in broadcast packets. We are able to do so with no additional to packet length. Our normal MAC header includes 3 fields: packet type, source address and destination address. For broadcast data packets the destination address is normally set as the common broadcast address (0xFFFF) in TinyOS. However, the packet type field also redundantly indicates that the packet is a broadcast packet. We therefore use the type field to indicate broadcast packets and reuse the address field to piggyback schedule information.

On the receiver side, when SCP receives a broadcast data packet, it extracts piggybacked schedule information from the destination field, and performs schedule synchronization. It then replaces the destination field with the broadcast address before it passes the packet to its upper layer. Our approach piggybacks synchronization information onto broadcast packets for free, and it does not affect the operation of upper layers.

V. Experimental Evaluation

The main contributions of this paper are to highlight the relative benefits of LPL and scheduling in energy conservation, and to propose a specific new MAC protocol, SCP-MAC. We have implemented SCP-MAC to validate both of these contributions.

In this section we focus explicitly on validating our analysis of the relative benefits of scheduling, LPL, and scheduled LPL. Since the performances of LPL alone over scheduling alone have been demonstrated, here we compare only LPL against scheduled LPL.

All actual MAC implementation has hundreds of specific design choices, many of which have effects on performance. To control these details in comparing LPL and scheduled LPL, we compare our own implementation of these protocols. This approach controls for algorithms such as CSMA, physical-layer

carrier sense mechanism, back-off, and other MAC algorithms. In addition, in these experiments, we disable the more advanced SCP-MAC features, including overhearing avoidance and adaptive listen, again to provide a simplified comparison of the core algorithms.

The second contribution of this paper is the set of design choices and optimizations described in Section IV. We do not attempt to compare those to an actual LPL implementation such as B-MAC [18] at this time for several reasons. First, those details would distract from our main question of comparing the advantages of scheduling on top of LPL. In addition, the current implementation of B-MAC (as of July 4, 2005) supports only a few pre-defined duty cycles, thus it would be impossible to explore a wide range of duty cycles directly. We identify a full evaluation of SCP-MAC's advanced features and a comparison to other full MAC implementations as future work.

A. Optimal Setup with Periodic Traffic

We first compare the energy performance of SCP and LPL under optimal configuration with completely periodic, known traffic. With static traffic loads we can optimize each for maximum energy conservation. We can use our implementation to validate the analysis leading to Figure 5. While known, periodic traffic is somewhat artificial, this configuration models a environmental monitoring applications where sensors are periodically sampled.

MAC parameters vary based on network size and data rate. For this test we place 10 nodes, all within range of each other, forming a single hop mesh. Each node periodically generates a 40B data message (not including preamble) and broadcasts it it to the network. Logically one would place base station in the middle of this mesh to record the data or relay it to the Internet, but we omit that node to focus on wireless performance.

We vary the data transmit rate to study how MAC performance varies. For this test we consider very light traffic loads typical for very long-lived sensor networks: we vary each node's message generation interval from 50–300s. (Thus the aggregate data rate for the whole network varies from 1 message every 5–30s.)

For each static traffic load, we find out the optimal polling period of LPL and SCP from Equations (13) and (17). We run each experiment for 5 message periods, generating 50 total messages over each experiment.

A central node begins and ends the experiment by a broadcast packet received by all nodes. We measure the energy consumption at each node by recording (in software) the time spent by the radio state between each state transition[1]. At the end of the experiment we collect this information from all nodes to a central measurement point.

Figure 10 shows the mean energy consumption of each node to send and receive all the messages. As expected, a lower traffic rate (corresponding with a larger inter-packet delay towards the right of the graph) results in a higher total energy cost. This result is because of a longer total experiment time and a higher cost to keep the network synchronized over longer gaps between messages. For LPL, the optimal polling interval is longer for slower traffic rates (see Figure 2), therefore the optimal preamble length is longer and so the cost of each message is longer. For SCP, the optimal sync period grows (Figure 3), and the optimal wake-up tone length grows slightly (Figure 4), but but the rate of growth is lower than for LPL. In addition, the absolute cost of SCP is much lower than LPL: we can see that LPL requires 2–2.5 times more energy than SCP to send the same amount of data. This savings is because scheduling allows much shorter preambles on each data message.

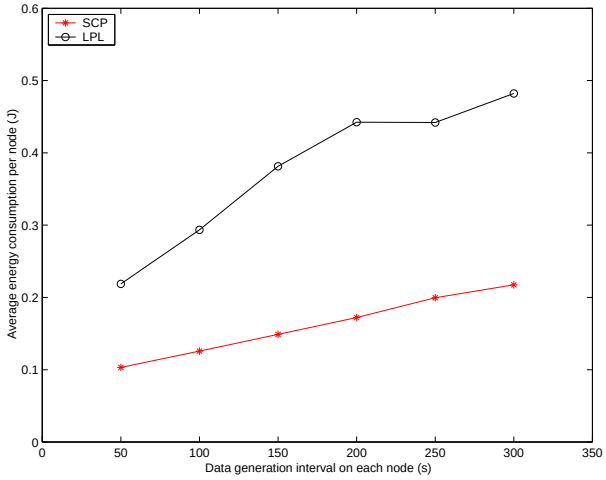

Fig. 10. Mean energy consumption (J) for each node as traffic send rate varies. (Assumes optimal LPL and SCP configurations, completely periodic traffic, and a 10-node network.)

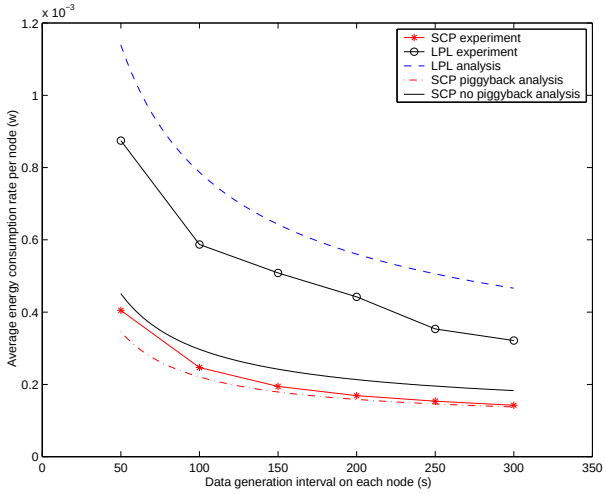

Fig. 11. Mean rate of energy consumption rate (W, or J/s) for each node as traffic send rate varies. (Assumes optimal LPL and SCP configurations, completely periodic traffic, and a 10-node network.)

Figure 10 shows the absolute total energy required to send a fixed amount of data over a given time (Joules per experiment). We can also express energy in terms of energy *rate*: Joules per second or Watts. We expect slower traffic rates correspond to lower rates of energy consumption. Figure 11 shows the total energy consumed (Figure 10) normalized by experiment dura-

[1]We do not explicitly model CPU energy, but in our experiments there are no CPU costs other than timer maintenance when radio is off. In Figure 9 we demonstrate that timer energy costs are not significant compared to radio costs.

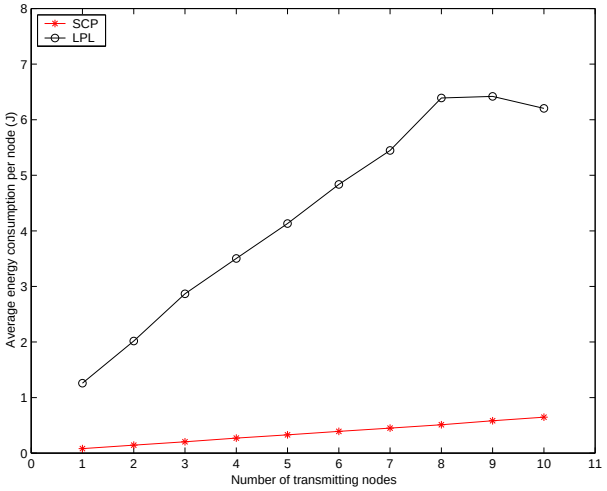

Fig. 12. Energy consumptions on heavy traffic load with very low duty cycle configurations.

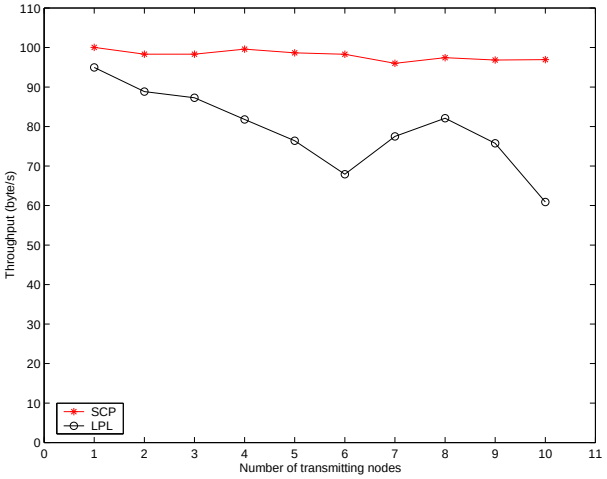

Fig. 13. Throughput on heavy traffic load with very low duty cycle configurations.

tion to give this rate. For easy comparison with analytical results, we put relevant portions of Figure 5 into Figure 11. We can see that both SCP and LPL experimental results closely match the trends of their analytical results with some fixed differences. The results validate the correctness of our analysis.

B. Energy Use Under Unanticipated Traffic Loads

In the prior section we consider optimal conditions for LPL and SCP with a completely known, periodic load. In many applications the traffic load is less predictable. For example, in tracking or monitoring applications such as fire detection, there are long stretches of operation with no events, but then a detection causes a flurry of activity and very bursty traffic. Such a network *must* be optimized for the common case when nothing happens, yet it must also be able to handle waking up out of this case and handling bursty traffic. Even if the MAC is customized with multiple modes, it must still operate conservatively before it is shifting to highly active.

To evaluate these scenarios we next consider MAC performance when operating outside its optimal regime. We tune LPL and SCP for a 0.3% duty cycle, polling every second. Since the polling interval is the same for both MACs, energy draw without traffic is almost identical. (SCP-MAC will require slightly higher energy to preemptively keep schedules synchronization.)

To simulate a sensor detection, we trigger all nodes to enter "busy" mode at the same time. When busy, each node generates 20 100B-long messages as rapidly as possible, starting to send the next as soon as the prior message is transmitted. This burst of traffic exercises the network at an operating point different from its optimal.

To vary the degree of offered load, we vary the number of nodes in the network that start sending from 1 to 10. This traffic causes severe contention as the number of transmitting nodes increases in the network. Figure 12 shows the average energy consumption of each node as the number of transmitting nodes increases. We can see that at this heavy traffic, LPL consumes about 8 times more energy than SCP to transmit an equal amount of data.

The main reason for this higher cost is the expense of LPL preambles. When optimized for low duty cycle with a 1s polling interval, each packet sent with LPL includes a 1s preamble. SCP avoids this overhead.

Of course, additional algorithms could improve both LPL and SCP performance. LPL could shift to shorter preambles for busy periods, however such a shift must be done conservatively to ensure all nodes agree to the transition—effectively a form of synchronization. SCP could benefit from adaptive listen [30] or T-MAC-style Future RTS [25]. While all of these optimizations are feasible, here we focus on an understanding of the core algorithm trade-offs before such optimizations.

C. Throughput Under Unanticipated Traffic Loads

Finally, we briefly explore one optimization in SCP-MAC: the use of separate contention windows for the wakeup tone and data.

Figure 13 shows the throughput of SCP and LPL under the same conditions as Section V-B. As the offered load increases the contention algorithms of each protocol is stressed. Both protocols do CSMA, however concurrent CSMA probes can miss each other. To avoid this, all contention-based MAC protocols use randomization. LPL uses a single contention window of randomized listening before sending its preamble; here we configure it to 32 slots. With 10 transmitters, there is roughly a one-third chance of two nodes selecting the same slot and therefore colliding.

As described in Section II-A, SCP uses *two* contention periods, one for the wakeup tone and the second for the data period. To keep the total time spent contending identical, we divide the 32 slots into 16 slots for each window. The two-phase contention window reduces overall collisions because even though there is a 10/16 (62%) chance of collision during the wakeup tone, collisions there do not matter since even multiple concurrent tones succeed in indicating the presence of traffic. Only nodes that collide in the wakeup contention window will compete in the data contention period, thus it has only a $10/16^2$ (4%)

effective collision rate.

We see the result of this more effective algorithm in Figure 13 as minimal reductions in SCP throughput as the number of transmitting nodes (and hence contention) increases. By comparison, LPL shows significantly lower throughput.

VI. Related Work

Energy-efficient MAC protocols have been a very active research area in wireless embedded and sensor networks. Existing work mainly focuses on two directions: TDMA and contention-based protocols.

TDMA protocols are naturally energy efficient. Their major limitations are the requirement of centralized control and strict time synchronization. Centralized control often requires nodes to form clusters and coordinated by cluster heads. Examples of such TDMA protocols include Bluetooth [7], LEACH [8], and BMA [14]. To extend the flexibility of TDMA, some distributed slot assignment schemes have been proposed, such as LMAC [26] and TRAMA [19]. Sohrabi and Pottie also proposed a protocol for distributed assignment of TDMA schedules [24]. Although TDMA protocols are attractive, we believe that contention-based protocols are better suited to dynamic sensor networks because of their flexibility and robustness.

Compared to TDMA protocols, contention-based protocols are more widely used in wireless sensor networks. Woo and Culler [27] examined different configurations of CSMA and proposed an adaptive rate control mechanism. However, the work does not focus on energy efficiency.

Reducing idle listening is one of the major challenge in contention-based protocols, and the major solution is to put nodes into low duty cycles. Two important techniques have been developed to make sensor nodes efficiently work in low duty cycle mode: scheduling and low-power listening (LPL). The power-save mode in IEEE 802.11 [17] adopts a centralized approach with the access point coordinating sleep times of nodes in a single-hop network. S-MAC [29], [30] developed a distributed coordination scheme to synchronize node sleep schedules in a multi-hop network. By scheduling node wakeup times, S-MAC enables nodes to run at duty cycles of 1–10%; it also fully decentralizes control, making it suitable for a multi-hop network. T-MAC [25] improves S-MAC by reducing the wakeup duration controlled by an adaptive timer and introducing future-RTS. We have also recently described how scheduling can be controlled to minimize latency in multi-hop communication [15]. The major advantage of scheduling is that a sender can determine a receiver's wakeup time and transmit efficiently. The major disadvantage in S-MAC and T-MAC is the relatively long listen time, as they incorporate the contention time. The long listen time is the major obstacle for these protocols to run at ultra-low duty cycles.

Low-power listening is an approach where the network channel is polled for presence of activity rather than for specific data. Exploiting much shorter network poll times, LPL by itself reduces energy consumption compared to alternatives. However, we are currently aware of no research that has explored scheduling these very low-power polls as we propose here. Instead, nodes perform channel polling in an uncoordinated fashion. To wake up a receiver, a sender needs to send a wakeup signal that is at least as long as the polling interval. STEM [21] explored this idea with a low-power paging channel. It uses the paging channel to transmit the wakeup tone and a normal channel to transmit data. Hill [9] and El-Hoiydi [4] independently developed the approach of sending the wakeup signal by simply adding preambles in front of each transmitted packet. WiseMAC [5] tries to further reduce the long preambles of packets after an initial packet with a long preamble. The improvement only works for certain traffic patterns, and long preambles have to be used for all broadcast packets. B-MAC [18] implemented the idea of LPL in TinyOS with a well-defined interface for applications to control the MAC behavior. It also developed an algorithm for clear channel assessment. The major advantage of LPL is that it minimizes the overhead of listening time. However, without scheduling on polling time, these existing protocols have large overhead on transmission side, which essentially prevent them from going to ultra-low duty cycles.

Separate from MAC protocols, as number of researchers have proposed higher-level approaches to conserve power, either as a topology management layer [28], [3], [22], integrated with routing, or at the application layer [20]. Such approaches are complementary with MAC-level optimizations to accomplish even lower effective duty cycles.

VII. Conclusions and Future work

This paper proposes a new MAC protocol based on scheduled channel polling, which enables sensor network nodes to operate at ultra-low duty cycles. It combines the strengths of both scheduling and low-power listening. To achieves the goal, we perform theoretical analysis to quantify the overhead of synchronization and relative benefits of scheduling compared to random channel polling. Our analysis also finds out the best operating point for both LPL and SCP.

We have implemented SCP-MAC in TinyOS over Mica2 motes. Our preliminary experiments show that SCP is able to achieve better energy performance than LPL by a factor of 2–2.5 when both of them use optimal configurations. Our experiments also demonstrated the advantage of SCP to handle unexpected traffic that does not match the ideal periodic model.

Our future work includes further implementation of all optimizations in SCP-MAC and thorough evaluation of its performance under different application requirements and traffic conditions. We also plan to make more complete comparisons with other sensor network MAC protocols.

Acknowledgments

We would like to thank Yuan Li for her contributions to the SCP-MAC implementation in its early stage. We also thank Joe Polastre at UC Berkeley for his help on CPU sleep mode on Mica2 motes.

References

[1] Vaduvur Bharghavan, Alan Demers, Scott Shenker, and Lixia Zhang. MACAW: A media access protocol for wireless LAN's. In *Proceedings of the ACM SIGCOMM Conference*, pages 212–225, London, UK, September 1994. ACM.

[2] Alberto Cerpa, Jeremy Elson, Deborah Estrin, Lewis Girod, Michael Hamilton, and Jerry Zhao. Habitat monitoring: Application driver for wireless communications technology. In *Proceedings of the ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, San Jose, Costa Rica, April 2001. ACM.

[3] Benjie Chen, Kyle Jamieson, Hari Balakrishnan, and Robert Morris. Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In *Proceedings of the ACM International Conference on Mobile Computing and Networking*, pages 85–96, Rome, Italy, July 2001. ACM.

[4] A. El-Hoiydi. Spatial TDMA and CSMA with preamble sampling for low power ad hoc wireless sensor networks. In *Proceedings of the SeventhInternational Symposium on Computers and Communications (ISCC)*, pages 685–692, July 2002.

[5] A. El-Hoiydi, J.-D. Decotignie, C. Enz, and E. Le Roux. WiseMAC: An ultra low power MAC protocol for the wisenet wireless sensor networks (poster abstract). In *Proceedings of the First ACM SenSys Conference*, Los Angeles, CA, July 2003. November.

[6] A. El-Hoiydi, J.-D. Decotignie, and J. Hernandez. Low power MAC protocols for infrastructure wireless sensor networks. In *Proceedings of the Fifth European Wireless Conference*, pages 563–569, Barcelona, Spain, February 2004. February.

[7] Jaap C. Haartsen. The Bluetooth radio system. *IEEE Personal Communications Magazine*, pages 28–36, February 2000.

[8] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocols for wireless microsensor networks. In *Proceedings of the Hawaii International Conference on Systems Sciences*, January 2000.

[9] Jason Hill and David Culler. Mica: a wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24, November 2002.

[10] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System architecture directions for networked sensors. In *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 93–104, Cambridge, MA, USA, November 2000. ACM.

[11] Chipcon Inc. CC1000 data sheet. `http://www.chipcon.com/`.

[12] Crossbow Technology Inc. Mica2 data sheet. `http://www.xbow.com/`.

[13] W.E. Leland, M.S. Taqqu, W. Willinger, and D.V. Wilson. On the self-similar nature of Ethernet traffic (extended version). *IEEE/ACM Transactions on Networking*, 2(1):1–15, February 1994.

[14] J. Li and G. Lazarou. A bit-map-assisted energy-efficient MAC scheme for wireless sensor networks. In *Proceedings of the 3rdInternational Symposium on Information Processing in Sensor Networks (IPSN)*, pages 55–60, Berkeley, CA, February 2004. April.

[15] Yuan Li, Wei Ye, and John Heidemann. Energy and latency control in low duty cycle MAC protocols. In *Proceedings of the IEEE Wireless Communications and Networking Conference*, New Orleans, LA, USA, March 2005.

[16] Alan Mainwaring, Joseph Polastre, Robert Szewczyk, and David Culler. Wireless sensor networks for habitat monitoring. In *Proceedings of the ACM Workshop on Sensor Networks and Applications*, pages 88–97, Atlanta, Georgia, USA, September 2002. ACM.

[17] LAN MAN Standards Committee of the IEEE Computer Society. *Wireless LAN medium access control (MAC) and physical layer (PHY) specification*. IEEE, New York, NY, USA, IEEE Std 802.11-1999 edition, 1999.

[18] Joseph Polastre, Jason Hill, and David Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd ACM SenSys Conference*, pages 95–107, Baltimore, MD, USA, November 2004. ACM.

[19] Venkatesh Rajendran, Katia Obraczka, and J.J. Garcia-Luna-Aceves. Energy-efficient, collision-free medium access control for wireless sensor networks. In *Proceedings of the First ACM SenSys Conference*, pages 181–193, Los Angeles, California, USA, November 2003. ACM.

[20] Nithya Ramanathan, Mark Yarvis, Jasmeet Chhabra, Nandakishore Kushalnagar, Lakshman Krishnamurthy, and Deborah Estrin. A stream-oriented power management protocol for low duty cycle sensor network applications. In *Proceedings of the IEEE Workshop on Embedded Networked Sensors*, pages 53–62, Sydney, Australia, May 2005. IEEE.

[21] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava. Optimizing sensor networks in the energy-latency-density design space. *IEEE Transactions on Mobile Computing*, 1(1):70–80, January 2002.

[22] Curt Schurgers, Vlasios Tsiatsis, Saurabh Ganeriwal, and Mani Srivastava. Topology management for sensor networks: Exploiting latency and density. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 135–145, Lusanne, Switzerland, June 2002. ACM.

[23] S. Singh and C.S. Raghavendra. PAMAS: Power aware multi-access protocol with signalling for ad hoc networks. *ACM Computer Communication Review*, 28(3):5–26, July 1998.

[24] Katayoun Sohrabi and Gregory J. Pottie. Performance of a novel self-organization protocol for wireless ad hoc sensor networks. In *Proceedings of the IEEE 50th Vehicular Technology Conference*, pages 1222–1226, 1999.

[25] Tijs van Dam and Koen Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the First ACM SenSys Conference*, pages 171–180, Los Angeles, California, USA, November 2003. ACM.

[26] L. van Hoesel and P. Havinga. A lightweight medium access protocol (LMAC) for wireless sensor networks. In *Proceedings of the 3rdInternational Symposium on Information Processing in Sensor Networks (IPSN)*, pages 55–60, Berkeley, CA, February 2004. April.

[27] Alec Woo and David Culler. A transmission control scheme for media access in sensor networks. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom)*, pages 221–235, Rome, Italy, July 2001. ACM.

[28] Ya Xu, John Heidemann, and Deborah Estrin. Geography-informed energy conservation for ad hoc routing. In *Proceedings of the ACM International Conference on Mobile Computing and Networking*, pages 70–84, Rome, Italy, July 2001. ACM.

[29] Wei Ye, John Heidemann, and Deborah Estrin. An energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the IEEE Infocom*, pages 1567–1576, New York, NY, June 2002. IEEE.

[30] Wei Ye, John Heidemann, and Deborah Estrin. Medium access control with coordinated, adaptive sleeping for wireless sensor networks. *IEEE/ACM Transactions on Networking*, 12(3):493–506, June 2004. A preprint of this paper was available as ISI-TR-2003-567.