PMAC: An adaptive energy-efficient MAC protocol for Wireless Sensor Networks

Tao Zheng
School of Computer Science
University of Oklahoma
Norman, Oklahoma 73019–6151
Email: tao@ou.edu

Sridhar Radhakrishnan †
School of Computer Science
University of Oklahoma
Norman, Oklahoma 73019–6151
Email: sridhar@ou.edu

Venkatesh Sarangan
Computer Science Department
Oklahoma State University
Stillwater, Oklahoma 74078
Email: saranga@cs.okstate.edu

***Abstract*— We propose a novel adaptive MAC protocol for wireless sensor networks. In existing protocols such as SMAC [1], the sensor nodes are put to sleep periodically to save energy. As the duty cycle is fixed in such protocols, the network throughput can degrade under heavy traffic, while under light loads, unwanted energy consumption can occur. In the proposed Pattern-MAC (PMAC) protocol, instead of having fixed sleep-wakeups, the sleep-wakeup schedules of the sensor nodes are adaptively determined. The schedules are decided based on a node's own traffic and that of its neighbors. Our analytical and experimental results show that in comparison to SMAC, PMAC achieves more power savings under light loads, and higher throughput under heavier traffic loads. Furthermore, unlike SMAC, only the sensor nodes involved in communication wake up frequently in PMAC and hence energy is conserved in other sensor nodes.**

I. Introduction

Wireless sensor networks are new types of special-purpose ad hoc networks. They have numerous applications in various fields such as monitoring animal/plant habitation, target tracking, homeland security, etc. These networks are usually deployed in an ad hoc manner with the nodes in the network sharing the same communication medium. The sensor nodes are usually operated by batteries and left unattended after deployment. Therefore, power saving is a critical issue in wireless sensor networks. Many research efforts in the recent years have focused on developing power saving schemes for wireless sensor networks. These schemes include power saving hardware design, power saving topology design [2], [3], power-efficient MAC layer protocols [1], [4], [6] and network layer routing protocols [7], [8], to name a few.

A MAC protocol is required in sensor networks to coordinate the sensor nodes' access to the shared medium. Designing power efficient MAC protocols is one of the ways to prolong the lifetime of the network. In addition to energy efficiency, latency and throughput are also important features for consideration in MAC protocol design for sensor networks. Commercial standards like IEEE 802.11 have a power management scheme for ad hoc networks, wherein the nodes remain in *idle listening* state at low traffic to conserve power. Studies show that significant power is wasted even in the idle listening mode [5]. Hence, 802.11 is not suitable for sensor networks.

†Author to whom all correspondence should be sent.

SMAC [1] is a MAC protocol specifically designed for wireless sensor networks. It forces sensor nodes operate at low duty cycle by putting them into periodic sleep instead of idle listening. Sensor nodes also sleep during overhearing to save power. Although, SMAC saves more power than 802.11, it does not adapt to network traffic very well since it uses a fixed duty cycle for all the sensor nodes. A duty cycle tuned for high traffic loads results in energy wastage when the traffic is low, while duty cycle tuned for low traffic loads results in low throughput under high traffic loads. The Timeout-MAC protocol (TMAC) [4] improves on SMAC by using an adaptive duty cycle. If there is no activity in the vicinity of a node for a time T_A, the node goes to sleep. Such an adaptation frees the application from the burden of selecting an appropriate duty cycle. TMAC has the same performance as SMAC under constant traffic loads, but saves more energy under variable traffic.

The down-side of TMAC's aggressive power conserving policy is that nodes can go to sleep rather early, resulting in increased latency and lower throughput. Another drawback in both SMAC and TMAC is that, they group the communication during small periods of activity. As a result, the protocols collapse under high traffic loads [5]. Data-gathering MAC (DMAC) [6] is another protocol that uses adaptive duty cycle. It provides low node-to-sink latency in convergecast communication by staggering the wake-up times of the nodes in the convergecast tree. While DMAC outperforms SMAC in terms of latency, throughput and energy efficiency, it remains to be seen if DMAC can support communication paradigms other than convergecast.

In this paper, we propose a new MAC protocol called Pattern-MAC (PMAC) for sensor networks that adaptively determines the sleep-wake up schedules for a node based on its own traffic, and the traffic patterns of its neighbors. We analytically and experimentally show that by doing so, our protocol is able to achieve a better throughput at high loads, and conserve more energy at light loads than SMAC.

We organize this paper as follows: In Sections II and III, we describe the working of our protocol. In section IV, we qualitatively analyze our protocol, and in section V, we present an analytical model to calculate the steady state average power savings in PMAC. Section VI shows the experimental results

Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)
1530-2075/05 $ 20.00 **IEEE**

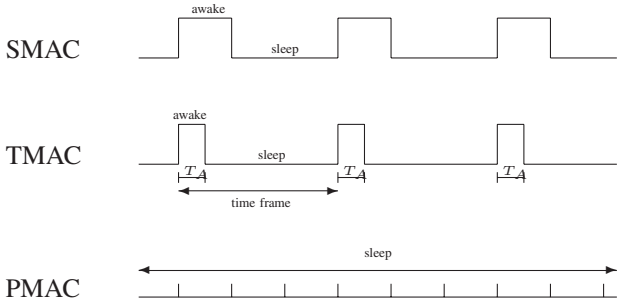


Fig. 1. Comparisons of the lengths of idle listening periods among SMAC, TMAC and PMAC with no traffic.

comparing SMAC and PMAC. We state our future work and conclude in section VII.

II. Overview of PMAC

PMAC is a 'time slotted' protocol like SMAC. In SMAC, a node can stay awake for a certain duration of a time slot, and go to sleep in the remaining duration; while in PMAC, a node can either be awake or asleep during a time slot.

A. Rationale behind PMAC

Idle listening is one of the main sources of energy wastage. Energy saving MAC protocols try to minimize the length of the idle listening period. Fig. 1 shows the lengths of idle listening periods in SMAC, TMAC and the proposed PMAC protocols in the extreme case of no traffic in the sensor network. In SMAC, sensor nodes have to wake up periodically even when there is no traffic in the network, thus wasting power. A small duty cycle can reduce this wastage, but it will cause low throughput when the traffic becomes heavy. In TMAC, sensor nodes need to wake up at the beginning of each time frame for a time T_A even when there is no traffic in the network, as a node needs to check for any activity in its neighborhood. In PMAC, a sensor node gets information about the activity in its neighborhood before hand through *patterns*. Based on these patterns, a sensor node can put itself into a long sleep for several time frames when there is no traffic in the network. If there is any activity in the neighborhood, a node will know this through the patterns and will wake up when required. Thus PMAC tries to save more power than SMAC and TMAC, without compromising on the throughput.

B. Pattern vs Schedule

A sleep-wakeup *pattern* is a string of bits indicating the *tentative* sleep-wakeup plan for a sensor node over several slot times. Bit 1 in the string indicates that the node *intends* to stay awake during a slot time, while 0 indicates that the node *intends* to sleep. For example, a pattern of 001 for a node indicates that, the sensor node tentatively plans to be asleep for two consecutive slot times, and stay awake in the third. Since the *pattern* is only a tentative plan, it is subject to change.

A sleep-wakeup *schedule* for a sensor node is a string of bits indicating the *actual* sleep-wakeup itinerary which the node will follow. Bit 1 in the string indicates that the node *will* stay awake during a slot time, while 0 indicates that the node *will* remain asleep.

The above definitions imply that a node's sleep-wakeup *pattern* need not be its sleep-wakeup *schedule*. In PMAC, the *schedule* for a node is derived from its own *pattern* and, the patterns of its neighboring nodes. Therefore patterns do affect the sleep and wakeup times of a node, and thus the protocol's performance.

In the next few paragraphs, we explain our approach for arriving at a node's pattern and schedule.

III. Protocol Details

As explained before, a node's pattern alters its sleep and wakeup times. In order to achieve a good throughput without compromising on the energy savings, it is important that the generated pattern should adapt to the network traffic.

A. Pattern Generation

Let P^j be the binary string representing the pattern of a node j. This pattern is associated with node j over N time slots. We call this sequence of N time slots as a *period*. In case the length of P^j is less than N, then the pattern gets repeated for the remaining duration. For example if $P^j = 01$, and if $N = 5$, then tentative plan for node j over the next five time slots will be 01010, i.e., the node will intend to *sleep* during slots 1, 3, and 5, and to remain *awake* during slots 2 and 4. In PMAC, we restrict a pattern to be 0^m1, where $m = 0, 1, \cdots N-1$. The number of 0 bits in a pattern, denoted by m, indicates the traffic load around the node having the pattern. A large m indicates the traffic load is light, while a small m (even a 0) indicates the traffic load is heavy.

In order to adapt to the traffic conditions, a node's pattern is updated during each period using the local traffic information available at the node and exchanged at the end of each period. Let P_i^j be the working pattern of node j during period i, where $i = 1, 2, \cdots$. Note that P_i^j can be different from P_{i+1}^j depending upon the node j's traffic conditions observed during period i. P_{i+1}^j can be obtained from P_i^j through either single or multiple updates occurring in period i. Let x_i be the number of pattern updates during period i and $P_{i,n}^j$, where $n = 0, 1, \cdots x_i$, be the n^{th} new pattern obtained in the sequence of updates. The starting pattern in the sequence during period i, $P_{i,0}^j$, is the working pattern P_i^j. The last updated pattern in the sequence, P_{i,x_i}^j, is going to be the working pattern in the next period i.e., $P_{i+1}^j = P_{i,x_i}^j$.

When the network is activated, the working pattern at every node has just one bit during the first period, which is 1, i.e., $P_1^j = 1$, $\forall j$ in the network. This simply assumes the traffic load is heavy at the beginning and every node should be awake. Pattern updates during the first period start with the working pattern P_1^j, i.e., $P_{1,0}^j = P_1^j = 1$. If there is no data[1] for a node j to send at the first time slot of bit 1, then it indicates that

[1]This data can be either the node's own data, or the data generated by other nodes which it has to relay.

Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)
1530-2075/05 $ 20.00 **IEEE**

the traffic load around node j is potentially light. Therefore, the node can afford to sleep for some time. Hence, node j updates its pattern to 01, i.e., $P^j_{1,1} = 01$. If we find that the node has no data to send during the second time slot of pattern bit 1, the node is encouraged to sleep longer by doubling the number of 0 bits in $P^j_{1,1}$, i.e., $P^j_{1,2} = 001$. This doubling effect continues in the following time slots of bit 1, until the number of 0 bits in the updated pattern reaches a predefined threshold δ. Beyond δ, the number of 0 bits is linearly increased. If there is no data for node j to send during period 1, the following sequence of patterns is generated at node j:

$$1, \quad 01, \quad 0^2\,1, \quad 0^4\,1, \quad \cdots \quad 0^\delta\,1, \quad 0^\delta\,01,$$
$$0^\delta\,0^2\,1, \quad 0^\delta\,0^3\,1, \quad \cdots \quad 0^{N-1}\,1.$$

This approach of exponential increasing the sleep time during light traffic allows the nodes to save considerable amount of energy. As you can see from the above, the sleep pattern that is generated mimics the slow-start algorithm of TCP [9].

If node j has any data to transmit at any time slot regardless of the pattern bit at that time slot, then the next pattern in the sequence goes back to 1. This enables node j to wake up quickly to handle the traffic load. The following update, if any, is going to start with this new pattern.

The pattern generation scheme used in PMAC is summarized in the following expression:

$$P^j_{i,n+1} = \begin{cases} 01 & \text{if } P^j_{i,n} = 1 \text{ and node } j \text{ has no data to send during the next slot of bit 1;} \\ 0^{2m}1 & \text{if } P^j_{i,n} = 0^m1 (0 < m \leq \delta/2) \text{ and node } j \text{ has no data to send during the next slot of bit 1;} \\ 0^{m+1}1 & \text{if } P^j_{i,n} = 0^m1 (\delta \leq m < N-1) \text{ and node } j \text{ has no data to send during the next slot of bit 1;} \\ 0^m1 & \text{if } P^j_{i,n} = 0^m1 (m = N-1) \text{ and node } j \text{ has no data to send during the next slot of bit 1;} \\ 1 & \text{if node } j \text{ has data to send during a slot, irrespective of the slot's pattern bit.} \end{cases} \quad (1)$$

It is easy to see that by increasing δ, the application can increase the aggressiveness of the sensor nodes to conserve energy. Similar to this multiplicative increase - acute decrease of the sleep times, other schemes such as additive increase - multiple decrease, additive increase - acute decrease, etc. can be employed, if the applications prefer them.

B. Pattern Exchange

A node's *pattern* is just the tentative sleep-wake up plan. In PMAC, the actual sleep-wakeup *schedule* is derived based on the node's own pattern and the pattern of its neighbors. New patterns that are generated for the subsequent period are broadcast by the nodes at the end of the current period.

To accommodate this pattern exchange, time is divided into super time frames (STF) as shown in Fig. 2. Each STF consists

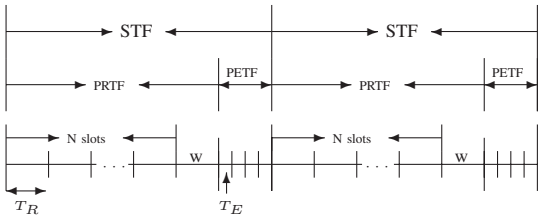


Fig. 2. Division of Time Frames

of two sub-frames. The first is called *Pattern Repeat Time Frame* (PRTF), during which each node repeats its current pattern. PRTF in turn is divided into different time slots of duration T_R. PRTF is nothing but the sequence of N time slots that we referred to as a *period* in the previous discussions. At the end of these N slots, PRTF has one additional time slot during which all the sensor nodes stay awake. This special time slot is used to speed up communication. Long delay may happen if the downstream neighbors are in a long sleep mode when upstream nodes have data destined for them. The upstream nodes cannot send data since they know the destination nodes are not ready, while the downstream nodes might think there is no traffic destined for them, and thus update their patterns for even longer sleep. During this special time slot, data from upstream nodes can be sent to downstream nodes so that downstream nodes can update their patterns to 1 and wake up quickly. This special time slot can also be used for broadcasting.

The second sub-frame of STF is called *Pattern Exchange Time Frame* (PETF), during which new patterns are exchanged between neighbors. PETF again, is divided into various time slots of duration T_E. New patterns are generated during PRTF at every node to reflect the latest traffic information by following the rules summarized in equation 1. The last generated pattern during a particular PRTF becomes the pattern for the next PRTF, and will be advertised to the neighbors during the PETF. The pattern is cyclically repeated during PRTF such that each time slot has one pattern bit assigned. Patterns received from its neighbors during the preceding PETF are also repeated in the same way. If a node j receives no new patterns from some of its neighbors during the preceding PETF (probably due to collisions), it then repeats their old patterns.

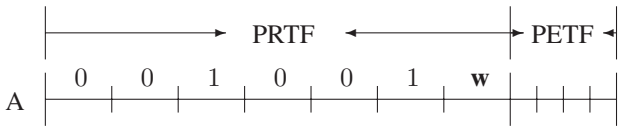


Fig. 3. Illustration of Pattern Exchange

Fig. 3. illustrates how pattern generation process takes place in PMAC. In Fig. 3, we assume that sensor node A has a pattern 001 at period i. Node A repeats its pattern during PRTF. Let $\delta = 4$ and $N = 6$. Pattern updates start with $P^A_{i,0} =$

Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)
1530-2075/05 $ 20.00 **IEEE**

TABLE I

RULES TO SET THE SCHEDULE BIT FOR A SLOT IN PRTF FOR NODE j

Pattern bit at node j	Packet to send	Pattern bit at the receiving node	Schedule at node j
1	1	1	1
1	1	0	$1-$
1	0	$*$	$1-$
0	1	1	1
0	1	0	0
0	0	$*$	0

001. If A has no data to transmit during the entire PRTF period, it generates a new pattern, $P^A_{i,1}$, at time slot 3, since the node has pattern bit set to 1 at that time slot. $P^A_{i,1}$ becomes $0^4\,1$ based on the pattern generation rules presented earlier. $P^A_{i,1}$ will be updated at time slot 6, at which another pattern bit 1 appears. $P^A_{i,2}$ now becomes $0^5\,1$ and is the last updated pattern. It is set as the pattern to be exchanged during PETF. However, if there is data at node A at any time slot, the new pattern goes back to 1. For instance, if there is data at time slot 2, $P^A_{i,1}$ is set to 1. Thereafter, if there is no data at time slot 3, $P^A_{i,2} = 01$.

The span of a time slot T_R is chosen such that it is long enough to handle a complete data transmission (contention window + RTS + CTS + DATA + ACK). The choice for N, the number of time slots in PRTF depends on the application. If N is high, then it is possible for the sensor nodes to have more sleep time, and thus more energy can be saved. However, this may also increase the latency in data transmission. Thus there is a tradeoff between energy saving and latency.

The number of time slots in PETF is set to the maximum number of neighbors a sensor node could have. The span of a time slot T_E in PETF is chosen long enough to broadcast a pattern. A large contention window may be needed at the beginning of each PETF time slot to avoid collision. However, longer PETF is, more overheads are introduced and thus more energy gets wasted. Here is a tradeoff between energy saving and reliability.

C. Schedule Generation

So far, we have explained how a node generates and exchanges its patterns with its neighbors. The purpose of the above exercise is to come up with the sleep-wakeup schedule for a node. To recall, the sleep-wakeup *schedule* for a node is a string of bits indicating the *actual* sleep-wakeup itinerary which the node will follow. Each bit in the string indicates the actual state of the node during a slot time. Bit 1 indicates that the node *will* stay awake, while 0 indicates that the node *will* remain asleep.

For the first N slots in PRTF, a schedule bit of 1 or 0 is obtained based on the pattern bit values of the node and its neighbors corresponding to that slot. The rules for arriving at the schedule bit value for node j for a given slot are enumerated below:

1) Let the pattern bit at node j be 1 and let there be a packet in its buffer to be sent to a neighbor. If the pattern bit for the receiving node is also 1, then the schedule bit for node j is set to 1. This means that node j will wake up at that particular time slot and send the data, since it knows that the receiver might be awake.
2) Let the pattern bit at node j be 1 and let there be a packet in its buffer to be sent to a neighbor. However, the pattern bit for the receiving node is 0. In this case, the schedule bit at node j to be $1-$, where $1-$ implies node j should wake up at the beginning of that time slot and listen for a certain period of time. If it hears nothing from its neighbor within that period, it can go to sleep. At the first glance, it would appear that it is better to set the actual schedule at node j to 0. That is, let node j sleep from the right beginning of that time slot to save more power. However, since the pattern bit of node j is 1, it could be a potential receiver and its neighbors may try to send data to it. If node j ignores this possible happening, and if it goes to sleep, the packet destined to it will be lost, and the energy spent on transmitting this packet is wasted.
3) The pattern bit at node j is 1 and there is no packet in its buffer. In this case, irrespective of the pattern bits of its neighbors, the schedule bit at node j is set to $1-$. The reason is the same as we explained in case 2.
4) The pattern bit at node j is 0 and there is a packet in its buffer to be sent. If the pattern bit at the receiving node is 1, the schedule bit of node j is set to 1. This implies that, node j is going to wake up at that time slot for transmission, although it intended not to. This can improve the throughput without consuming any additional energy. Throughput is improved by waking up node j earlier than it is supposed to. No additional energy is consumed because the packet in the buffer needs to be transmitted sooner or later.
5) The pattern bit at node j is 0, and there is a packet in its buffer to be sent. If the pattern bit at the receiving node is 0, the schedule bit for node j is set to 0. This would imply putting node j into sleep, since the destination node is not ready to receive. To send the packet, node j must wait until the time slot at which the destination node has pattern bit 1.
6) The pattern bit at node j is 0 and there is no packet in

Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)
1530-2075/05 $ 20.00 **IEEE**

its buffer. In this case, no matter what pattern bits its neighbors have, the schedule bit of node j is set to 0. This means that node j is going to sleep mode. If some neighbors have packets for node j, they have to wait until the time slot at which the pattern bit of node j becomes 1. This would introduce longer delays for the first few packets when the traffic becomes heavy, but the subsequent packets will experience lower delays as node j's pattern adapts to the new traffic.

The above rules of setting the schedule bit for a slot are summarized in table I.

IV. Qualitative Discussion

In this section, we give a qualitative discussion on the efficacy of PMAC.

A. Adaptability to traffic conditions

As we stated in the previous section, the number of 0 bits in a new pattern is grows exponentially when the traffic load is light. This means that sensor nodes can fall into a long sleep quickly under light loads. Hence, PMAC is able to save more power than SMAC. If any data is detected during the current PRTF, the new pattern generation process will start over from 1. This enables, a sensor node to wake up quickly when the traffic load becomes heavy. PMAC is thus able to adapt to the traffic conditions. We also note that the pattern repeating process may compromise the speed of adapting to the network traffic, since new patterns must wait until the next PRTF to be effective.

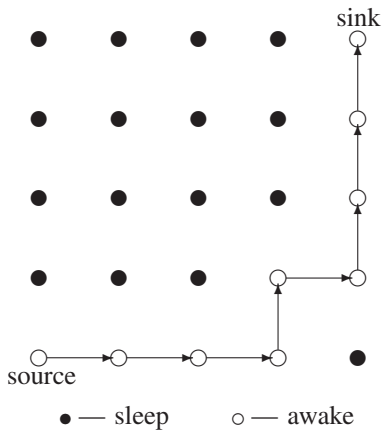


Fig. 4. Illustration of Power Saving Through Localization

B. Power Savings through Localization

In PMAC, only those sensor nodes involved in a communication will wake up frequently. Other sensor nodes that do not participate in the data gathering/relaying process will sleep for longer times. In other words, PMAC selectively wakes up sensor nodes. Fig. 4 illustrates the power saving through localization feature of PMAC on a 5×5 mesh. Suppose that the only traffic in the network is from the source node to the sink node along the path indicated by the arrows. At steady state, only those sensor nodes on the path will stay awake to handle the traffic. Other nodes not involved in the communication will sleep through most of the PRTF period.

C. Power savings through reduced Idle listening

We have just described how PMAC saves energy by allowing sensor nodes that are not involved in any communication to remain asleep. This in turn, reduces the energy wasted due to idle listening during the periodic wakeups that take place in SMAC. PMAC can also potentially introduce additional idle listening than SMAC. This occurs if in the actual schedule of a sensor node, there are two consecutive wakeup time slots, but during the second time slot no communication is associated with the sensor node. This is the case where its pattern has two consecutive 1's when the traffic load is light. Fortunately, because of the sparse spurts of traffic in sensor networks, this kind of pattern does not occur quite often.

The time-out scheme used in TMAC can be even introduced into PMAC to save more power by allowing sensor nodes go to sleep during their wakeup time slots if no activity in the vicinity of those nodes.

D. Time Synchronization

Since time is slotted, some level of synchronization among the sensor nodes is needed in PMAC as with SMAC. However, as only large time scales are involved in PMAC (in the order of hundred milliseconds), small clock drifts would not be a problem. The sensor nodes can use some loose time synchronization schemes, such as the one proposed in SMAC [1] to synchronize the sensor nodes. At the end of a PETF, a node can advertise a SYNC packet after waiting for a random duration. The neighboring nodes that receive this SYNC packet can adjust their clocks and need not advertise their SYNC packets. In case a node receives SYNC from two neighbors, it remembers and follows both of them.

V. Analytical Model

We present a simple analytical model to study the pattern generation process and calculate the steady state average power savings in PMAC under light traffic. We calculate the average power savings by estimating the number of zero bits appearing in the pattern. For simplicity, we ignore the pattern repetition during PRTF presented earlier, and study PMAC with the following model. Starting with a working pattern 1, if no data is available at a node for transmission at the time slot with bit 1, the pattern for the next two slots is set to be 01. Again, if no data to be sent during the previous two time slots, the next working pattern for the subsequent slots becomes 001, and so on. While the above model may not completely capture the complete traits of the protocol, the analysis does provide good insight into the results that follow.

Let p be the steady state probability that the buffer at node j is non-empty at a particular time slot. As we are interested in the steady states, it is reasonable to assume p is a constant over all the time slots. For the sake of simplicity, we also assume $\delta = N = 2^M$. Now we can use the Markov chain

Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)
1530-2075/05 $ 20.00 **IEEE**

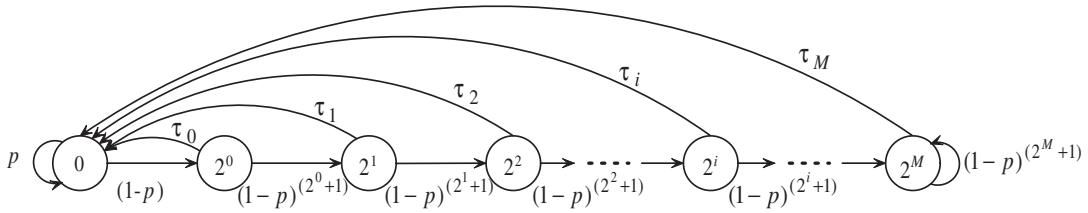


Fig. 5. Markov chain

(Fig. 5) to model the pattern evolution process. In the figure, each state is indicated by the number of 0 bits in its pattern. Let $P_0, P_{2^0}, P_{2^1}, P_{2^2}, \cdots, P_{2^i}, \cdots, P_{2^N}$ be the probability that the sensor node is in each of those states. We can have the following equations based on the Markov chain:

$$\begin{aligned}
P_{2^0} &= (1-p)P_0 \\
P_{2^1} &= (1-p)^{(2^0+1)}P_{2^0} \\
P_{2^2} &= (1-p)^{(2^1+1)}P_{2^1} \\
&\vdots \quad \vdots \\
P_{2^i} &= (1-p)^{(2^{i-1}+1)}P_{2^{i-1}} \\
&\vdots \quad \vdots \\
P_{2^M} &= (1-p)^{(2^{M-1}+1)}P_{2^{M-1}} + (1-p)^{(2^M+1)}P_{2^M}
\end{aligned}$$

By substitution, we have for $0 \le i \le M-1$

$$\begin{aligned}
P_{2^i} &= (1-p)^{(2^{i-1}+1)} \cdots (1-p)^{(2^0+1)}(1-p)P_0 \\
&= P_0(1-p)^{\left[\left(\sum_{j=0}^{i-1} 2^j\right)+(i+1)\right]} \\
&= P_0(1-p)^{(2^i+i)}
\end{aligned} \tag{2}$$

and

$$\begin{aligned}
P_{2^M} &= (1-p)^{(2^{M-1}+1)}P_0(1-p)^{(2^{M-1}+M-1)} \\
&\quad +(1-p)^{(2^M+1)}P_{2^M} \\
&= \frac{P_0(1-p)^{(2^M+M)}}{1-(1-p)^{(2^M+1)}}
\end{aligned} \tag{3}$$

From the Markov chain, we should also have

$$P_0 = pP_0 + \tau_0 P_{2^0} + \tau_1 P_{2^1} + \cdots + \tau_i P_{2^i} + \cdots + \tau_M P_{2^M} \tag{4}$$

where $\tau_i = 1-(1-p)^{2^i+1}$ is the transition probability from state P_{2^i} to P_0 for all $0 \le i \le M$. We can verify that (4) holds by plugging (2) and (3) into it. Since a sensor node must be at one of those states, the summation of the probabilities should be 1.

$$\begin{aligned}
1 &= P_0 + P_{2^0} + P_{2^1} + P_{2^2} + \cdots + P_{2^i} + \cdots + P_{2^M} \\
&= P_0 + P_0 \sum_{i=0}^{M-1} (1-p)^{(2^i+i)} + P_0 \frac{(1-p)^{(2^M+M)}}{1-(1-p)^{(2^M+1)}}
\end{aligned} \tag{5}$$

Hence,

$$P_0 = \frac{1}{1+\sum_{i=0}^{M-1}(1-p)^{(2^i+i)} + \frac{(1-p)^{(2^M+M)}}{1-(1-p)^{(2^M+1)}}} \tag{6}$$

Now the probability at (2) becomes

$$P_{2^i} = \frac{(1-p)^{(2^i+i)}}{1+\sum_{i=0}^{M-1}(1-p)^{(2^i+i)} + \frac{(1-p)^{(2^M+M)}}{1-(1-p)^{(2^M+1)}}} \tag{7}$$

for $0 \le i \le M-1$. The probability at (3) becomes

$$P_{2^M} = \frac{1}{1+\sum_{i=0}^{M-1} \frac{1-(1-p)^{(2^M+1)}}{(1-p)^{(2^M-2^i+M-i)}} + \frac{1-(1-p)^{(2^M+1)}}{(1-p)^{(2^M+M)}}} \tag{8}$$

The average number of 0 bits, denoted by $E(0)$ in a pattern, can then be obtained as follows

$$E(0) = \sum_{i=0}^{M} \left(2^i P_{2^i}\right) \tag{9}$$

When the traffic is heavy, p is close to 1. From (6), we can see that P_0 is also close to 1 and thus $E(0)$ is close to 0. When the traffic is light, p is close to 0. From (8), we can see that P_{2^M} tends to 1 and thus $E(0)$ is close to 2^M.

We will now derive an expression to determine the additional amount of power saved at a particular node in PMAC over SMAC. Consider a time interval of length $E(0) * T_R$, where T_R is the slot time in PMAC. It is easy to see that, over this entire duration, a sensor node will be asleep in PMAC. If T is the frame duration in SMAC and d is the duty cycle, then over the same time interval of $E(0) * T_R$, a node in SMAC will be awake for the duation $\frac{E(0)*T_R}{T} * d$. Thus the amount of additional energy saved at a particular node in PMAC over SMAC is given by

$$E_{save} = \frac{E(0) * T_R * d * P_{idle}}{T} \tag{10}$$

where P_{idle} is the power consumption when sensor nodes are in idle listening state. Let S be the number of sensor nodes in the network not involved in the data gathering/relaying process, then the idle listening energy saved by PMAC in the sensor network during the time interval can be calculated as

$$E_{save}^{total} = S * \frac{E(0) * T_R * d * P_{idle}}{T} \tag{11}$$

VI. Experimental Results

We have simulated both PMAC and SMAC using the latest version of *ns*-2. Simulations were done on mesh topology, as shown in Fig. 4, with different sizes. In both simulations, we have used *constant bit rate* traffic source with different time intervals and UDP as the transport layer protocol. The

Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)
1530-2075/05 $ 20.00 **IEEE**

TABLE II
PARAMETERS USED IN THE SIMULATIONS

Parameter	Value
initial energy	100 Joules
transmission power	0.5 Watts
receiving power	0.3 Watts
idle power	0.05 Watts
bandwidth	20 kbps
data cwnd	63 ms

simulation time is set to 1500 seconds. For PMAC, $T_R = 258$ms and $T_E = 104$ms, where T_R and T_E are the slot time in PRTF and PETF, respectively. The number of time slots in PRTF is 64 and the number of time slots in PETF is 4. The duty cycle for SMAC is set to 10. Some of the parameters used in the simulations are listed in Table II. We have set the idle power to be smaller than the receiving power in our simulations, but a larger value for idle power is going to favor PMAC even more.

Fig. 6 compares the energy consumption, throughput and power efficiency between PMAC and SMAC with different number of nodes when the traffic load is heavy (100bytes/s). Fig. 6(a) shows that SMAC consumes more energy than PMAC. Fig. 6(b) shows that PMAC achieves a much higher throughput than SMAC when the traffic load is heavy. This is because in SMAC (without adaptive listening), sensor nodes have to go to sleep periodically even when the traffic load is heavy. In PMAC, patterns vary with the traffic load. If traffic load is heavy, the patterns will contain less 0's forcing the sensor nodes to stay awake to handle the traffic, resulting in the improvement. Our interest is in the calculation of throughput as opposed to the end-to-end goodput because, in a sensor network traffic may be aggregated at the intermediate nodes without traveling all the way down to the sink node. Fig. 6(c) compares PMAC and SMAC in terms of their power efficiency. Power efficiency, which is the throughput achieved per unit of energy consumed, is given as

$$\text{power efficiency} = \frac{\text{total throughput}}{\text{total energy consumption}}$$

We observe that PMAC outperforms SMAC in terms of this metric too.

Fig. 7 shows the contour maps of the remaining energy distribution of a 5×5 mesh network with the same communication path as shown in Fig. 4. We observe that in both PMAC and SMAC, the sensor nodes along the communication route consumes more energy, as they are involved in more transmission and receiving. However, in PMAC, those nodes at the upper-left corner and the lower-right corner, which are not involved in the communication, have more remaining energy in comparison with SMAC. This is because in SMAC, those nodes not involved in the communication still have to wake up periodically and in PMAC, these nodes fall into long sleep, saving the energy that is otherwise wasted due to unnecessary idle listening.

Fig. 8 compares the energy consumption, throughput and power efficiency between PMAC and SMAC under different traffic loads. This experiment was performed on the same 5×5 mesh network, and the number of time slots in PRTF is set to 64. Fig. 8(a) shows that PMAC consumes less energy than SMAC for all traffic loads. Fig. 8(b) shows that the throughput of PMAC and SMAC are the same when the traffic is light, due to the fact that sleeping in SMAC will not affect traffic flowing through. However, when the traffic is heavy, PMAC has a significant improvement on the throughput. Periodic sleeping blocks the traffic flowing through in SMAC. We also observe from Fig. 8(c), that PMAC has a better power efficiency, especially when the traffic load is heavy.

VII. CONCLUSIONS AND FUTURE WORK

This paper proposes a new MAC protocol, called PMAC, where the sleep-wakeup times of the sensor nodes are adaptively determined. The schedules are decided based on a node's own traffic and that of its neighbors. Our experimental results show that in comparison to SMAC, PMAC achieves more power savings under light loads, and higher throughput under heavier traffic loads. The improved performance of PMAC suggests that 'pattern exchange' is a promising framework for improving the energy efficiency of the MAC protocols used in sensor networks.

Our on-going research efforts on PMAC are along many dimensions. Currently we are carrying out comparative analysis of the proposed PMAC with other adaptive MAC protocols such as TMAC and DMAC for variable bit-rate data transfers. We also plan to carry out a detailed analysis of PMAC under various kinds of traffic such as broadcast, convergecast, and point-to-point. We believe that PMAC can be enhanced with timeout mechanisms along the lines of TMAC, and thereby achieve even better performance. We are developing better pattern and schedule generation schemes to further improve the energy efficiency and data transfer latency in PMAC.

REFERENCES

[1] W. Ye, J. Heidemann and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *INFOCOM 2002*, New York, Jun. 2002, pp.1567-1576.

[2] A. Salhieh, J. Weinmann, M. Kochhal and L.Schwiebert, "Power Efficient Topologies for Wireless Sensor Networks," in *ICPP'01*, Valencia, Spain, Sept. 2001, pp.156-163.

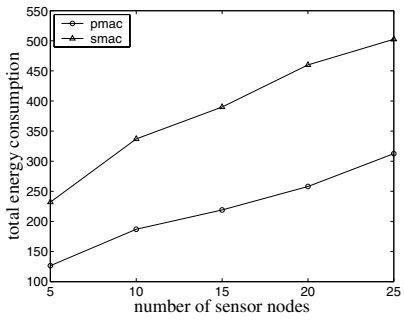
[3] M. Khan, G. Pandurangan and B. Bhargava, "Energy-Efficient Routing Schemes for Wireless Sensor Networks," *Tech. Rep.*, CSD TR 03-013, Department of Computer Science, Purdue University, July 2003.

[4] T. V. Dam and K. Langendoen, "An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *SenSys'03*, Los Angeles, Nov. 2003, pp.171-180.

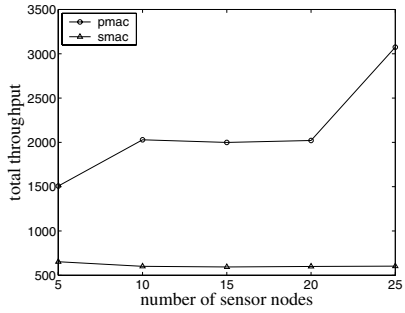
[5] K. Langendoen and G. Halkes, "Energy-Efficient Medium Access Control", Book chapter in the *Embedded Systems Handbook*, R. Zurawski (editor), CRC press, to appear.

[6] G. Lu, B. Krishnamachari and C. Raghavendra, "An Adaptive Energy-Efficient and Low-Latency MAC for Data Gathering in Sensor Networks," in *International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks* (WMAN 04), April 2004.

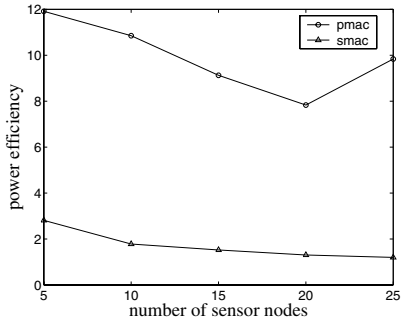
[7] W. Heinzelman, A. Chandrakasan and H. Balakrishnan, "Energy-Efficient Communication Protocols for Wireless Microsensor Networks," in *Proc. Hawaian Int'l Conf. on Systems Science*, Jan. 2000.



(a) The comparison of total energy consumptions between PMAC and SMAC with different number of nodes.

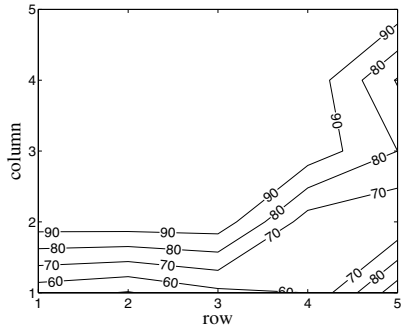


(b) The comparison of total throughput between PMAC and SMAC with different number of nodes.

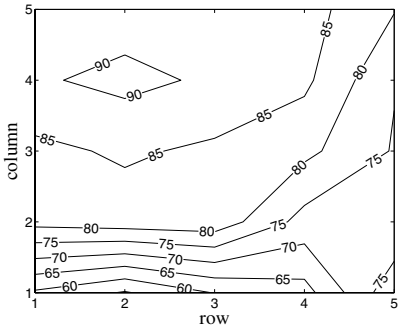


(c) The comparison of power efficiency between PMAC and SMAC with different number of nodes.

Fig. 6. Comparison of power efficiency between PMAC and SMAC with various network size under heavy traffic load.

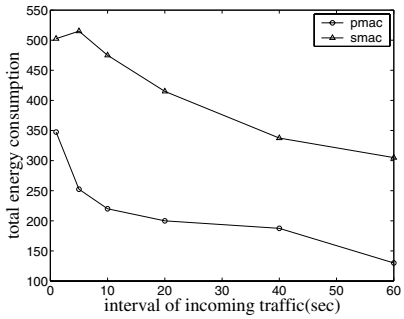


(a) Remaining energy distribution on a 5×5 mesh for PMAC.

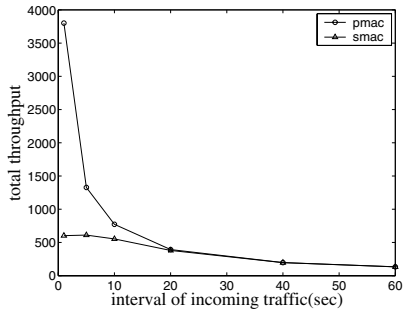


(b) Remaining energy distribution on a 5×5 mesh for SMAC.

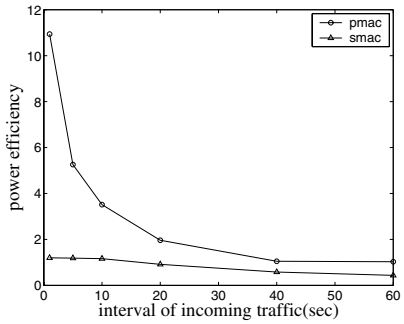
Fig. 7. Contour map of remaining energy on a 5×5 mesh for both PMAC and SMAC.



(a) The comparison of total energy consumptions between PMAC and SMAC with different incoming traffic rates.



(b) The comparison of total throughput between PMAC and SMAC with different incoming traffic rates.



(c) The comparison of power efficiency between PMAC and SMAC with different incoming traffic rates.

Fig. 8. Comparison of power efficiency between PMAC and SMAC under different traffic loads

[8] W. Heinzelman, J. Kulik and H. Balakrishnan, "Adaptive Protocols for Information Dissemination in Wireless Sensor Networks," in *Proc. 5th ACM/IEEE Mobicom Conf.*, Seattle, WA, Aug. 1999.

[9] A. S. Tanenbaum, *Computer Networks*, 3rd edition, prentice hall, Upper Saddle River, NJ, 1996.

Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)
1530-2075/05 $ 20.00 **IEEE**