This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2007 proceedings.

RMAC: A Routing-Enhanced Duty-Cycle MAC Protocol for Wireless Sensor Networks

Shu Du Amit Kumar Saha David B. Johnson
Department of Computer Science, Rice University, Houston, TX, USA

***Abstract*—Duty-cycle MAC protocols have been proposed to meet the demanding energy requirements of wireless sensor networks. Although existing duty-cycle MAC protocols such as S-MAC are power efficient, they introduce significant end-to-end delivery latency and provide poor traffic contention handling. In this paper, we present a new duty-cycle MAC protocol, called RMAC (the Routing enhanced MAC protocol), that exploits cross-layer routing information in order to avoid these problems without sacrificing energy efficiency. In RMAC, a setup control frame can travel across multiple hops and schedule the upcoming data packet delivery along that route. Each intermediate relaying node for the data packet along these hops sleeps and intelligently wakes up at a scheduled time, so that its upstream node can send the data packet to it and it can immediately forward the data packet to its downstream node. When wireless medium contention occurs, RMAC moves contention traffic away from the busy area by delivering data packets over multiple hops in a single cycle, helping to reduce the contention in the area quickly. Our simulation results in *ns-2* show that RMAC achieves significant improvement in end-to-end delivery latency over S-MAC and can handle traffic contention much more efficiently than S-MAC, without sacrificing energy efficiency or network throughput.**

I. Introduction

Large-scale wireless sensor networks have a significant potential in applications such as monitoring of natural and man-made environmental phenomena and events, but this potential may be limited due to the limited battery capacity of sensor nodes. Many traditional wireless MAC protocols used in wireless ad hoc networks, such as IEEE 802.11, require a wireless device to remain awake to monitor the medium, even when the node is not transmitting or receiving a packet. Since a typical sensor network application usually generates very light traffic on the network, this idle-listening mechanism is very inefficient and wastes significant energy.

To mitigate this energy consumption of idle listening, duty-cycling mechanisms have been introduced in sensor network MAC protocols. For example, in S-MAC [1], each sensor node follows a periodic synchronized listen/sleep schedule. An overview of the operation of S-MAC is shown in Figure 1, in which a node S sends a packet to a node D. The listening period, in which the node's radio is enabled, is divided into a SYNC period and a DATA period. During the SYNC period, an independent synchronization protocol is used to synchronize the clocks of the sensor nodes, so that they can be awake simultaneously with their neighbors. During the DATA period, packets from applications can be sent. Similar to IEEE 802.11, S-MAC uses the RTS/CTS mechanism to avoid collisions between multiple transmitting nodes, and when a node receives a data packet, it returns an ACK to the sender.

At the start of a SLEEP period, a node turns off its radio and goes to sleep to save energy, unless it is still in the middle of data transmission, in which case, the sender and the receiver go to sleep after the transmission completes. In the example in Figure 1, neither node S nor D will go to sleep until the ACK frame is received by S. The above S-MAC operational cycle is repeated endlessly during the life of the nodes.

As in IEEE 802.11, nodes in S-MAC maintain a Network Allocation Vector (NAV) for virtual carrier sensing. For example, node X in Figure 1 is a neighbor of node D and overhears the CTS sent by D. Node X will set its NAV to indicate this virtual carrier and will not send any traffic while its NAV is nonzero. Inter-frame spacing, such as SIFS (Short Inter-Frame Spacing) and DIFS (Distributed Inter-Frame Spacing), are also used in S-MAC, as in IEEE 802.11. Before a node transmits an RTS, it waits a random time in its contention windows (CW) in order to decrease the possibility of collision when multiple nodes try to send data in the same DATA period.

Duty-cycle MAC protocols are more energy efficient than traditional MAC protocols, but they have some limitations. Most importantly, end-to-end delivery latency may be increased substantially; for example, with S-MAC, in each operational cycle, a packet can be forwarded over a single hop only, since an intermediate relaying node has to wait for its next downstream node to wake up to receive the packet.

In addition, because a sensor node using a duty-cycle MAC protocol is synchronized to be awake during the same short period as its neighbors, the probability of network contention increases. Although sensor network applications usually generates very light traffic, decreasing the importance of this concern, in some applications such as event monitoring, communication demands may suddenly increase in a burst in a small neighborhood. For example, when a fire starts, several temperature monitoring sensors in the area will report to the sink node at the same time. If the transmission contention among these sensors is not handled well, the emergent data may be lost or will experience a long delivery latency.

Finally, existing duty-cycle MAC protocols significantly limit network throughput, since nodes can be active only during a small fraction of the time. Although network throughput in sensor networks is not as important as in traditional networks, throughput is still an important factor, for example to support high throughput during a temporary traffic burst, such as in event-monitoring applications.

Motivated by the above problems, in this paper, we present the design and evaluation of a new duty-cycle MAC protocol, called RMAC (the Routing enhanced MAC protocol), that

0743-166X/07/$25.00 ©2007 IEEE

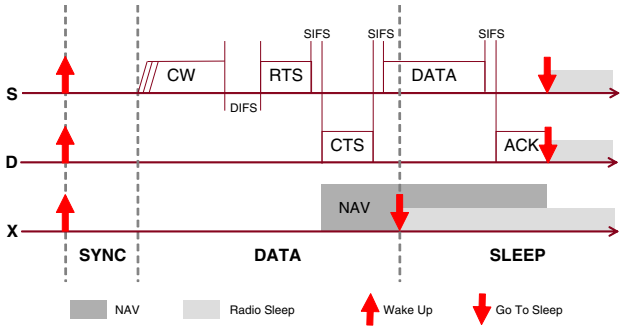

Fig. 1. S-MAC: A typical duty-cycle MAC protocol for sensor networks

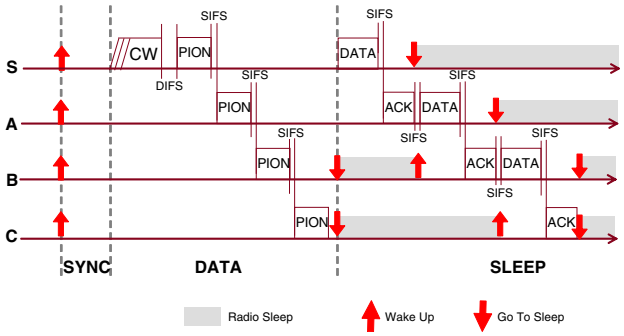

Fig. 2. RMAC overview

exploits cross-layer routing information in order to avoid these problems without sacrificing energy efficiency. Most importantly, RMAC can deliver a data packet *multiple* hops in a single operational cycle. During the SLEEP period in RMAC, a relaying node for a data packet goes to sleep first and then intelligently wake up when its upstream node has the data packet ready to transmit to it. After the data packet is received by this relaying node, it can also immediately forward the packet to its next downstream node, as that node has also just woken up and is ready to receive the data packet.

RMAC can thus deliver a data packet much faster without sacrificing the energy efficiency achieved by the duty-cycle mechanism. RMAC can also efficiently handle traffic contention by moving the contention traffic quickly away from the contention area. Also, when a burst of traffic occurs, RMAC is able to make multiple transmissions in a single SLEEP period, thus taking better advantage of the SLEEP period than previous duty-cycle MAC protocols.

The organization of the rest of this paper is as follows. In Section II, we describe the basic ideas behind RMAC and give an introduction to the protocol. We then present the details of RMAC in Section III, including the control and data forwarding algorithms and the exception handling methods. Simulation-based performance evaluation is presented and discussed in Section IV. In Section V, we discuss the related work in the area of sensor network MAC designs. Finally, Section VI draws conclusion and discusses future work.

II. RMAC Overview

In order to reduce end-to-end delivery latency with a duty-cycle MAC protocol, the protocol should be able to forward a data packet *multiple* hops within a single operational cycle. The design of RMAC is guided by the fact that to achieve this, nodes along the data forwarding path need to be awake only when actually transmitting or receiving the packet. RMAC thus sends a small control frame along the data forwarding path to allow all nodes along the path learn when to be awake in order to receive the data packet from the immediate upstream node and forward it to the immediate downstream node.

Figure 2 shows an overview of the operation of RMAC. An operational cycle of a sensor node in RMAC can be divided into three stages: SYNC, DATA, and SLEEP. Similar to prior work, RMAC assumes that a separate protocol (e.g., [2], [3]), operating during the SYNC period, synchronizes the clocks on sensor nodes with the required precision.

When a data packet is to be sent to a destination node that is multiple hops away, a control frame is sent during the DATA period to initiate the communication with the downstream nodes. Instead of using a pair of RTS and CTS frames between just two nodes, RMAC uses a series of control frames, named PIONs (Pioneer frames), across multiple hops. A PION is used to request communication, like an RTS frame, and to confirm a request, like a CTS frame. Most importantly, a node transmits a single PION to confirm receipt of a PION from its upstream node and to simultaneously request communication from a downstream node. This dual function makes the multihop relaying of PIONs very efficient. We will present the PION mechanism in detail in Section III.

During a SLEEP period, nodes go to sleep except for those that have communication tasks, as set up by the PIONs. Every node that has sent or relayed a PION must wake up at some specific time to transmit or forward the data frames; each node goes back to sleep after completing its communication task.

III. RMAC Protocol Details

A. *Pioneer Control Frame (*PION*)*

When a node has data to send, the node initiates its request at the start of a DATA period. For example, if a source node S has data to send to some destination (Figure 2), node S first picks a random period from the contention window and waits for the medium to be quiet for that period and an additional DIFS period (as in IEEE 802.11) before sending a PION to the next-hop node A.

This PION includes all fields as in an RTS, such as current node's address, the next-hop address, and the duration of the transmission. More importantly, the PION also includes some cross-layer information: the final destination address of the current flow and the number of hops the PION has traveled. This final destination address is passed down by the networking layer, and the hop count is set to zero when the data packet is generated by the source node.

When A receives S's PION, unless A is the final destination of this flow, A gets the next-hop address for this destination from its own network layer. A then waits a SIFS period (as in IEEE 802.11) before transmitting its own PION. The PION contains three addresses, apart from the final destination address: its own address, the previous-hop address (S), and the

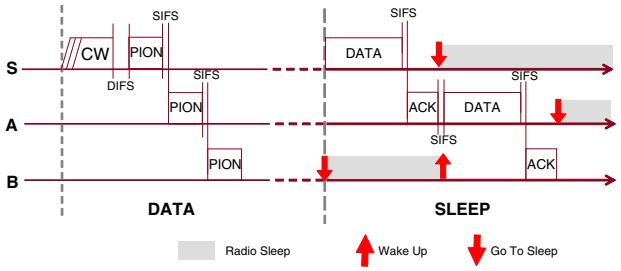

Fig. 3. Data transmission example

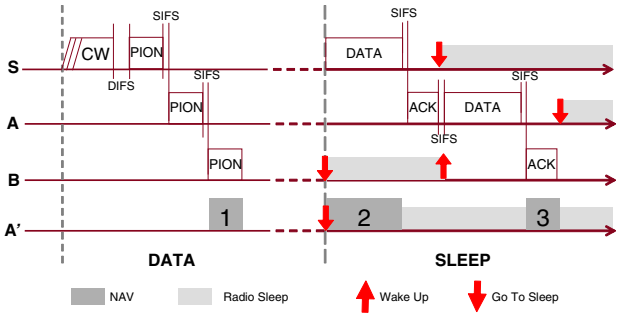

Fig. 4. Network Allocation Vector (NAV) example

new next-hop address (e.g., B). Additionally, the hop count in this new PION is set to 1 more than that in the received PION; the use of the hop count will be explained in Section III-B. This PION from A serves both as a CTS to S and as an RTS to B. Unlike other protocols, when S receives A's PION, S does not send its data frame immediately but waits for the start of the SLEEP period to transmit the data frame; the DATA period of the operation cycle is used only to send and receive PION frames, setting up the schedule for the actual data transmission. Data frames are transmitted and received only during the SLEEP period. Upon receiving A's PION, B performs the same steps as A. This process of receiving a PION and immediately transmitting another PION continues until either the final destination has received the PION or the end of the current DATA period is reached.

B. Data Transmission

As mentioned in Section III-A, all data frames are transmitted in the SLEEP period. In the example in Figure 3, when the first node S receives the PION confirmation from node A, it waits until the start of the SLEEP period to transmit the data frame. Node A stays awake to receive the data frame at the start of the SLEEP period, and after node A receives the data frame, it sends an ACK frame to S. After receiving the ACK, node S goes to sleep mode.

If node A earlier received the confirmation PION from its next hop B in the DATA period, A immediately relays the data frame to B. This data frame relaying process continues at each hop until the final destination is reached or the data frame reaches some node that did not receive a confirmation PION from its next hop, in which case the node just holds the data frame until the DATA period of the next operational cycle. At that time, this node sends a fresh PION to the next hop with the hop count reset to zero. This entire process is repeated until the final destination is reached.

In the above case, when the SLEEP period starts, nodes S and A start their data frame sending/receiving immediately. Other nodes in the multihop path that took part in the PION transmission in the current DATA period go to sleep to save energy. Each node later wakes up at the right time to receive the data frame from the upstream node and send it to the downstream node. For example, node B can go to sleep when the SLEEP period begins, but it wakes up at the scheduled time when A is ready to forward the data frame to B.

The data relaying process is very much like a pipeline process; the correct wake up time of a node can be calculated from the hop count from in the PION frame. Suppose a node is the ith hop during the PION transmission. Its wake up time $T_{wakeup}(i)$ should be

$$T_{wakeup}(i) = (i-1) \cdot (durDATA + \text{SIFS} + durACK + \text{SIFS}) \quad (1)$$

where *durDATA* and *durACK* are the times to send a single data frame and an ACK frame, respectively. If all data frames in the sensor network are the same size, *durDATA* could be a preset value; otherwise, *durDATA* can be included in the PION, to let every node along the path calculate the correct wake up time.

C. Setting the Network Allocation Vector (NAV)

The Network Allocation Vector (NAV) at each node is used in IEEE 802.11-style MAC protocols for virtual carrier sense, to avoid packet collisions. A non-zero NAV implies a busy medium and hence prevents a node from transmitting. Unlike existing sensor network MAC protocols, the control sequence is different in RMAC, since the PIONs schedule data transmissions expected in the future. Thus, the NAV in RMAC records *segments* of time, rather than a single duration, during which the medium is considered busy. All nodes that overhear a PION set a segment in their NAV based on the *durDATA* and the hop count i in the PION. For example, in Figure 4, if node A' is a neighbor of node A, in order to avoid collision at node A, A' should not transmit if node A is potentially receiving anything. Therefore, if node A' overhears a PION from A to B, it should set its own NAV to reserve the following three segments of time (in the format $[starttime, endtime]$):

Confirmation PION *Segment (1):* $[now, now+durPION]$, where *durPION* is the transmission duration of a PION frame. This segment ensures that A' will not transmit when A is receiving the confirmation PION from B.

Data Segment (2): $[t_{datastart}, t_{dataend}]$, where $t_{datastart}$ can be calculated by A' based on the next sleep time and T_{wakeup} of A. The value $t_{dataend}$ can then be further calculated by adding a *durDATA* to $t_{datastart}$. This segment ensures that A' will not transmit when A is receiving the data frame from the previous hop.

ACK *Segment (3):* $[t_{ackstart}, t_{ackend}]$, where $t_{ackstart}$ can be calculated by adding *durDATA* plus *durACK* plus $3 \times$ SIFS to $t_{dataend}$. The value t_{ackend} is $t_{ackstart}$ plus *durACK*, the transmission time of an ACK frame. This segment ensures that A' will not transmit when A is receiving the ACK frame from its next hop.

After setting the above NAV segments, if node A' receives

another PION destined to itself that requests A' to do data relaying, then A' only transmits a confirmation PION, thereby agreeing to relay the data frame, if the relaying assignment does not conflict with its current NAV settings. This relaying assignment involves an immediate PION and future ACK and data transmissions. If any of these assignments conflict with the current NAV at A', then A' does not transmit a PION, thus rejecting the future relaying of the corresponding data frame; the previous hop will have to request A' in the DATA period of the next operational cycle.

D. Handling Frame Losses

If a PION is lost, the upstream node of the PION's sender will not get a confirmation and so will not try to send the data frame to the downstream node. The upstream node will initiate a new PION in the next DATA period. Unfortunately, the downstream node does not know that the upstream node did not get its confirmation, so it will wake up and receive nothing. After a predefined timeout, it will go back to sleep. Therefore, the downstream node wastes some energy on the unnecessary wakeup and the data packet cannot travel as many hops as it could have. In the worst case, the upstream node may lose the PION from the next downstream node, but that PION may successfully traverse several hops further along the downstream. In this worst case, every node along the downstream will wake up at the scheduled time, wait, receive nothing, and then go back to sleep after time out.

If a data or a ACK frame gets lost, no retries are made in the current operational cycle, since the next hop is not scheduled to be awake to receive the retransmitted packet. Consequently, the upstream node goes back to sleep and tries again in the next DATA period, starting with a fresh PION. In summary, RMAC does not require any retry effort in a single operational cycle. The node that identifies a loss will start with a fresh PION in the next operational cycle.

E. Synchronization and Data Fusion

In sensor networks, multiple duty-cycle schedules may co-exist due to the limitation of the synchronization algorithms or hardware. Irrespective of the duty-cycle based MAC protocol being used, if a node has neighbors of different duty-cycle schedules, the node needs to keep track of all these different schedules. In RMAC, if a node receives a PION and the next hop has a different duty-cycle schedule, the PION relaying stops; the relaying node will first receive the data and then try to deliver the packet to the next hop following the next hop's duty-cycle schedule.

Since the clock rates of the sensors may not be the same, sensor clocks can drift apart over time. Therefore, when a node calculates the wake up time using Equation 1, it can further deduct a small time value from the calculated result to ensure it will always wake up before its upstream node starts transmitting.

Many sensor networks apply data fusion algorithms when a sensor relays data packets for the others. For example, data values can be aggregated and compressed on their way to the sink node. Therefore, the wake-up time and NAV segments cannot be easily calculated using a fixed *durDATA*. To solve this problem, a PION can further include a field of accumulated data transmission duration. When a PION is forwarded, the relaying node adds its own *durDATA* value into the accumulated data transmission duration in the PION. The neighboring nodes and downstream nodes can thus correctly set their NAV segments or wake-up timers.

TABLE I
NETWORKING PARAMETERS

Bandwidth	20 Kbps	Sleep Power	0.05 W
Rx Power	0.5 W	Carrier Sensing Range	550 m
Tx Range	250 m	Contention window (CW)	64 ms
Tx Power	0.5 W	DIFS	10 ms
Idle Power	0.45 W	SIFS	5 ms

TABLE II
TRANSMISSION DURATION PARAMETERS

	Frame Size (bytes)	Tx Latency (ms)
RTS/CTS	10	11.0
ACK (in S-MAC/RMAC)	10	11.0
PION	14	14.2
DATA (in S-MAC/RMAC)	50	43.0

IV. SIMULATION EVALUATION

To evaluate our RMAC design, we evaluated it using version 2.29 of the *ns-2* simulator. We simulate the Two Ray Ground radio propagation model and a single omni-directional antenna at each sensor node. We compare RMAC against S-MAC without the adaptive listening mode [4]. We did not include adaptive listening in S-MAC because the end-to-end latency results from adaptive listening can be easily derived from the basic S-MAC simulation results (the latency will be reduced by half), but adaptive listening also consumes much more energy than the basic S-MAC. Table I shows the key parameters we used in our simulations; these are the default settings in the standard S-MAC simulation module distributed with the ns-2.29 package. According to the *ns-2* documentation, the default 250m transmission range and the 550m carrier sensing range are modeled after the 914MHz Lucent WaveLAN DSSS radio interface, which is not typical for a sensor node. However, similar proportions of carrier sensing to transmission range are also observed in some state-of-art sensor nodes [5]. In future work, we will investigate the impact of smaller carrier sensing range.

In our simulations, traffic loads are generated by constant bit rate (CBR) flows, and all data packets are 50 bytes in size. Intermediate relaying nodes do not aggregate or compress data. We also assume that the application data processing at any node can be finished within a SIFS period; thus, data processing introduces no extra delivery delays. The transmission latencies for different types of packets are shown in Table II.

The transmission latencies in Table II are calculated as:

$$durFrame = \frac{p + (Frame\ Size \cdot E)}{Bandwidth} + 1 \text{ ms} \quad (2)$$

where we use the default 5 bytes for the preamble size p and the default encoding ratio E of 2 in our simulations. The duration of the DATA period in S-MAC can then be calculated as:

$$T_{DATA}(S\text{-}MAC) = CW + \text{DIFS} + durRTS + \text{SIFS} + durCTS \quad (3)$$

where *durRTS* and *durCTS* are the transmission latencies of the RTS and CTS frames, respectively. The duration of the DATA

Fig. 5. Chain scenario

period in RMAC is calculated as:

$$T_{DATA}(RMAC) = CW + \text{DIFS} + durPION + N \cdot (\text{SIFS} + durPION) \quad (4)$$

where *durPION* is the transmission duration of a PION frame. The PION relaying number N defines how many PION frames can be forwarded in each DATA period. In order to enable multihop relaying of a PION, N should be greater than 1. In our simulations, we chose $N = 4$.

Finally, the duty cycle R is defined as the proportion of the radio awake time to the entire cycle time of a sensor node:

$$R = \frac{T_{awake}}{T_{cycle}} = \frac{T_{SYNC} + T_{DATA}}{T_{SYNC} + T_{DATA} + T_{SLEEP}} \quad (5)$$

We keep the same duty cycle (5%) for both RMAC and S-MAC, although this makes the whole cycle in RMAC longer than in S-MAC due to RMAC's longer DATA period. The duty cycle-related settings are shown in Table III.

In our simulations, we assume all the nodes in the network have already been synchronized to use a single wake-up and sleep schedule. There is no synchronization traffic during our simulations, but nodes still wake up at the beginning of the SYNC period and listen to the medium. Also, we do not include any routing traffic in the simulations, as we assume the existence of a routing protocol deployed to provide the shortest path between any two nodes.

A. Overview of Scenarios

We use three types of scenarios in our simulations: *chain* scenarios, *cross* scenarios, and *realistic* scenarios.

Figure 5 shows an example of a *chain* scenario. All nodes are equally spaced in a straight line, and neighboring nodes are placed 200 m apart. One single CBR (constant bit rate) flow sends packets from the node 0 to the node n. The length of the chains varies from 1 hop to 24 hops. The chain scenario helps us to study the protocols for basic multihop delivery.

Figure 6 shows an example of a *cross* scenario. Two straight chains of nodes cross each other at a center node. The two chains are of the same length, and the single node at the crossing point is shared by the two chains. Therefore, the length of the chains must be of a even number of hops. All the neighboring nodes are placed 200 meters apart as well. There are two CBR flows, each along one chain of nodes, from one end of a chain to the other. The two CBR flows generate packets at the same time and at the same rate, and thus their traffic contends with each other at the center. The length of the two chains is varied from 2 hops to 24 hops. Cross scenarios are used to study the protocols for basic inter-flow contention.

Finally, Figure 7 shows an example of a *realistic* scenario composed of 200 sensor nodes and a sink node. The 200 sensor nodes are uniform randomly distributed in a 2000 m by 2000 m square area, and the sink node is located at the top right corner of the square. Figure 8 shows the histogram of the path lengths from the sensors to the sink. The maximum path length from a sensor to the sink is 15 hops, and most of the sensors are about 7 to 13 hops from the sink. All traffic in the network is from a sensor node to the sink, generated as follows: at a periodic interval, a random sensor node is selected to send one data packet to the sink node. If a node is selected to send a packet, it is taken out from the selection pool. If the selection pool becomes empty, which means each of the 200 nodes has sent a data packet to the sink, then the selection pool is reset to contain all 200 nodes.

TABLE III
CYCLE DURATION PARAMETERS

	T_{SYNC} (ms)	T_{DATA} (ms)	T_{SLEEP} (ms)	T_{cycle} (ms)
S-MAC	55.2	104.0	3025.8	3185.0
RMAC	55.2	168.0	4241.8	4465.0

TABLE IV
RESULTS OF 24-HOP NETWORKS

	Scenario	Latency (seconds)	T_{cycle} (seconds)	$\frac{Latency}{T_{cycle}}$ (cycles)	$\frac{PathLength \cdot T_{cycle}}{Latency}$ (hops/cycle)
S-MAC	chain	74.9	3.185	**23.52**	**1.02**
S-MAC	cross	87.0	3.185	**27.32**	**0.88**
RMAC	chain	17.4	4.465	**3.90**	**6.16**
RMAC	cross	20.4	4.465	**4.57**	**5.25**

B. Latency Evaluation

In this subsection, we evaluate the performance of end-to-end delivery latency. We use a typical light traffic load for sensor networks. For the chain and cross scenarios, each CBR flow generates a traffic load of 100 packets at the rate of 1 packet every 50 seconds; the entire simulation runs for 5500 seconds of simulation time. For the realistic scenario as well, data is generated every 50 seconds, but each sensor sends only one data packet to the sink node; the simulation runs for 10300 seconds of simulation time.

1) Latency Evaluation in Chain Scenarios: Figure 9 shows, for the chain and cross scenarios, how the average packet delivery latency varies with the path length; error bars show the minimum and maximum delivery latencies.

For the chain scenarios, delivery latency in both S-MAC and RMAC increases as the hop count of the path increases. However, delivery latency in S-MAC increases at a much faster rate, although it actually has a shorter operational cycle than does RMAC. This shows the benefit of RMAC's capability of multihop delivery within a single cycle. This capability can be better presented in Table IV. Using the cases of the 24-hop chain and cross scenarios, the last two columns in the table show the total number of operational cycles needed for a packet to finish the 24-hop delivery and the average number of hops over which a packet can be forwarded in a single cycle.

For the 24-hop chain scenario, S-MAC can forward a data packet over only 1.02 hops per cycle. It is slightly more than 1 hop, because for the first hop (from node 0 to node 1), S-MAC does not need a full cycle to deliver the packet. Depending upon when the data packet is generated at node 0, node 0 may be able to send the packet to 1 immediately, if a data packet is generated during a DATA period. Otherwise, node 0 must wait for the start of the next DATA period; this waiting time may vary, but is never more than one cycle.

On the other hand, RMAC, can forward a data packet an

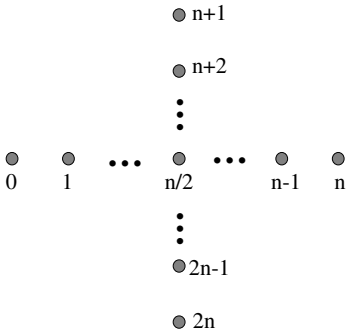

Fig. 6. Cross scenario

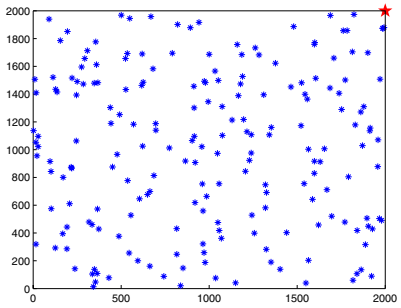

Fig. 7. A realistic 200-node network

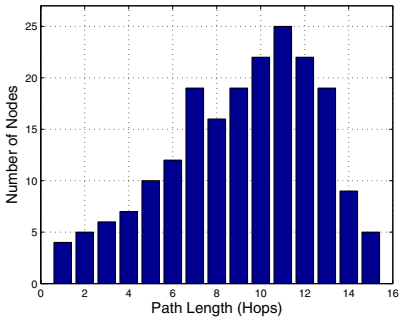

Fig. 8. The histogram of the path lengths of the realistic network

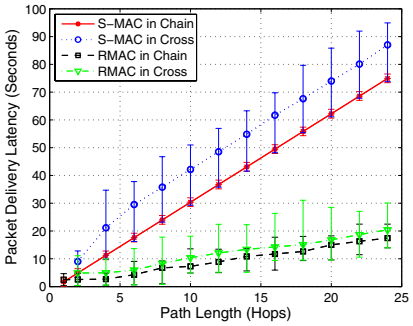

Fig. 9. Delivery latency in the chain and cross scenarios

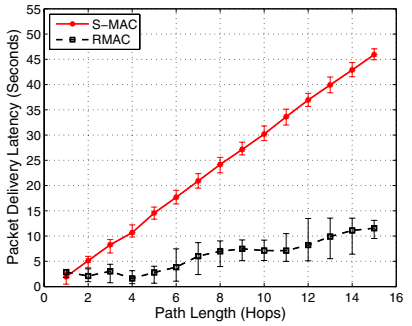

Fig. 10. Delivery latency in the realistic scenario

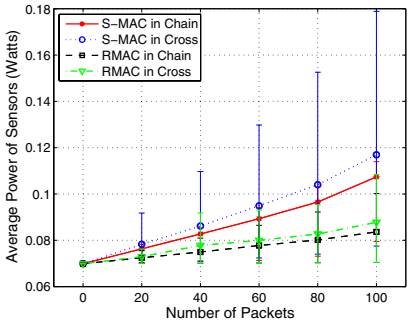

Fig. 11. Average power of sensors in the 24-hop chain and cross scenario

average of 6.16 hops per cycle. This is in spite of a value of $N = 4$ in Equation 4. The PION can be forwarded 6.16 (> 4) hops in a single DATA period because of the size of the contention window. When a source sensor node randomly selects a slot in the CW, it has to wait until that slot arrives before it can initiate its PION frame. As long as the node does not select the last slot in CW, it can use that extra time for relaying the PION over multiple hops. If this extra time is more than *durPION*+SIFS, the PION can be forwarded for one more hop than N in this DATA period. In the best situation, the node selects slot 0 in its CW, and the whole CW may potentially be used for forwarding PIONs. Therefore, the number of extra hops a PION may be forwarded in a cycle is

$$X = \frac{CW}{durPION + \text{SIFS}} = \frac{64 \text{ ms}}{14 \text{ ms} + 5 \text{ ms}} = 3.37 \text{ (hops)} \qquad (6)$$

Thus, theoretically, in each cycle in RMAC, the data packet can be forwarded for at least N hops, but may be as many as $\lfloor N + X \rfloor$ hops. When the SLEEP period starts, if we require a node not to go to sleep immediately if its radio is in transmitting or receiving mode, then we can potentially forward the PION for one more hop, because the last hop will have time to finish its PION confirmation to its previous upstream node. In this case, the maximum forwarded hops is $\lceil N + X \rceil$. This also shows a very good feature of RMAC: RMAC can efficiently use a large contention window (CW), specifically, to deliver a PION frame over more hops ($> N$) in each DATA period. The value for CW is usually decided by the application requirement. If the possibility of simultaneous traffic generation is high in the neighborhood, for example in an event monitoring network, a high value for CW is needed to avoid potential collisions. A large CW may consume more power from the sensor nodes or introduce longer delivery latency if the duty cycle is kept fixed. In RMAC, since a PION can be potentially delivered further when the CW is larger, RMAC mitigates the negative effects of a large contention window while still achieving the MAC contention resolution intended by a larger CW.

Another major difference for the chain scenarios between S-MAC and RMAC in Figure 9 is the shape of their curves. In S-MAC, as the path length increases, the end-to-end delivery latency increases linearly, with very little fluctuation, since S-MAC forwards the data packet with a fixed rate of 1 hop per cycle. However, for RMAC, the number of hops over which data can be forwarded in each cycle depends on the random backoff selected by the source node. Therefore, the curve for end-to-end latency for RMAC has more fluctuations and larger error bars. For the chains shorter than 5 hops, the fluctuations in RMAC's latency are much smaller than those for longer chains, because for shorter chains all the packets can reach their final destinations within a single cycle.

2) Latency Evaluation in Cross Scenarios: Figure 9 also shows the latency results of S-MAC and RMAC in the cross scenario. Traffic contention has much less impact on RMAC than on S-MAC; the gap between the chain curve and the cross curve is much wider in S-MAC. Table IV shows that compared to the chain scenario for a 24-hop flow, the end-to-end delivery latency increases by 12.1 s, or 3.80 cycles, in S-MAC, but only increases by 3 s, or 0.67 cycle, in RMAC. RMAC deals with contention much better than does S-MAC, due to RMAC's ability to deliver packets over multiple hops in a single cycle. Since the carrier sensing range is 550 meters in our simulations,

a data packet in our cross scenario must be more than 3 hops away from the crossing point in order to avoid any potential interference from the other flow. RMAC is efficient in helping packets get through the contention area quickly, and further away, thus avoiding the contention. In a cross scenario, when two packets from the two flows arrive simultaneously at the contention area in the center, one of them wins and gets relayed. This winning packet immediately goes several hops away so that in the next operational cycle, the contention is already removed from the crossing area, and the two packets can be delivered along their respective paths.

3) Latency Evaluation in the Realistic Scenario: Figure 10 shows the results of our latency evaluation for realistic scenarios. Because the data generation interval of 50 seconds is long enough for an earlier generated packet to be delivered to the sink before the next packet is generated, there are no competing flows in the network. Comparing the results in Figure 10 with the chain scenario results in Figure 9 (for chains shorter than 16 hops), they are almost of the same shape, except that in the realistic scenarios, the curves of both S-MAC and RMAC have greater fluctuations than in the chain scenarios due to the small sample size in the realistic scenario; in Figure 9, each point is the average of 100 different samples, whereas in Figure 10, the number of samples at a hop count k depends upon the number of the k-hop neighbors of the sink node; the number of k-hop neighbors is not monotonic with k, as shown in Figure 8.

C. Energy Consumption Evaluation

In this subsection, we evaluate the energy efficiency of RMAC. Here as well, we use the typical light traffic load in a 24-hop chain, 24-hop cross, and the realistic scenarios. We varied our traffic load up to 100 packets in each topology, and we observe the average sensor power consumption during the entire simulated time. If the simulation has multiple packets to send, then for the chain and cross scenarios, each CBR flow generates traffic load at the rate of 1 packet every 50 seconds. For the realistic scenario , the periodicity of data generation is also 50 seconds. Each simulation runs for 5500 seconds of simulated time.

Figure 11 shows the average power over all the sensors in the chain and the cross scenario. Average power consumed is calculated by dividing the total energy consumed by the sensors by the total simulated time. Error bars show the minimum and the maximum values for a single sensor's average power consumption. When there is no traffic in the network, nodes in RMAC consume the same energy as those in S-MAC. This is because both use the same duty cycle ratio R, thus having the same power efficiency. As the traffic load increases, both RMAC and S-MAC increase their energy consumption, but RMAC has a smaller rate of increase than does S-MAC. This is because RMAC has a more concise control frame sequence than does S-MAC. For a multihop delivery of a packet, sensors in RMAC transmit only about half as many total control frames. Less transmitting also implies less receiving or overhearing, which further increases the energy efficiency of the entire network with RMAC. Another reason RMAC is more energy efficient is that sensors in RMAC never consume energy on overhearing a data frame transmission, because during a data frame transmission, all the nodes are in the sleep mode except for the two sensors that are communicating. In S-MAC, however, since the data frame is transmitted right after the CTS is received, part of the data frame transmission may happen in the DATA period, and so all the neighboring sensor nodes spend energy to overhear the transmission.

Figure 11 also shows the impact that traffic contention has on energy efficiency. Both RMAC and S-MAC consume more energy in cross scenarios than in chain scenarios, although the gap between the chain curve and the cross curve in S-MAC is much wider than in RMAC. This is expected, as we have discussed in Section IV-B, due to the difference in contention handling by the two protocols. In a cross scenario, it is always a node in the crossing area that consumes the most energy, which is shown in the figure as the upper limit of the error bars. For the maximum average power value, S-MAC has a large increase in cross scenarios over the chain scenarios of the same path length. A sensor network's lifetime is actually decided by lifetime of the bottleneck links, in this scenario, the nodes that are in the crossing area. Therefore, RMAC's efficient contention handling, together with its energy efficient control sequences, can help to prolong the lifetime of the network.

Figure 12 shows the average power of sensors in the realistic scenario. RMAC is more energy efficient than S-MAC in the realistic scenario. Both curves look flatter than the corresponding curves in Figure 11. This is because each point on the curves is the average of 200 nodes, and many nodes do not participate in packet relaying as much as the nodes in the chain or cross scenarios. Nodes in the bottleneck link, such as the one-hop neighbors of the sink, still consume similar amounts of energy as the ones in the cross scenario, and are shown as the upper limits of the higher error bars.

D. Throughput Evaluation

In this subsection, we evaluate the network throughput using RMAC. Although network throughput is not a crucial metric in typical sensor networks, it is important when the traffic can potentially come in a burst. We again use a 24-hop chain, a 24-hop cross, and the realistic scenarios in our simulations. We varied our traffic load in terms of the packet generation rate, from 1 packet every 50 seconds to 10 packets every 50 seconds. For the cross scenarios, the two flows split their load equally, for example, if the traffic load is 5 packets every 50 seconds, each of the two flows will generate at the rate of 2.5 packets every 50 seconds. Each simulation runs for 2000 seconds of simulated time, and we record the throughput of the network in terms of the average number of packets successfully received by the final destination in every 50 seconds.

Figure 13 shows our simulation results. Each point in the curve is the average of 4 different runs, and the error bars show the minimum and the maximum values. For all cases, the output rate follows the input rate when the input rate is low and finally the output rate reaches its peak point. If we continue injecting more packets into the system, after the output has peaked, the input creates more contention in the system and decreases

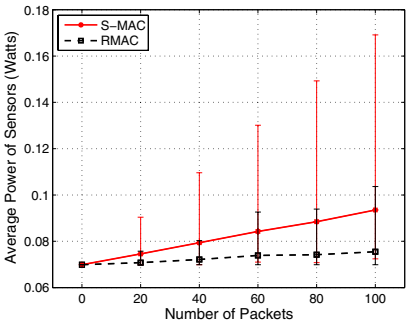

Fig. 12. Average power of sensors in the realistic scenario

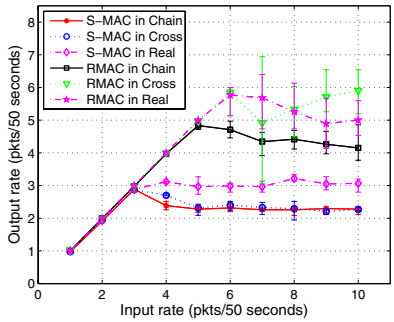

Fig. 13. Throughput in the chain (24-hop), cross (24-hop) and realistic scenario

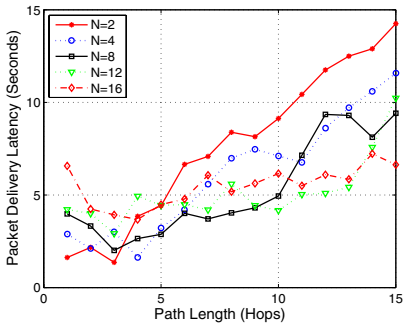

Fig. 14. Packet delivery latency with different PION relaying number

TABLE V
CYCLE DURATIONS WITH DIFFERENT N

N	T_{SYNC} (ms)	T_{DATA} (ms)	T_{cycle} (ms)
2	55.2	129.6	3696
4	55.2	168.0	4465
8	55.2	244.8	6000
12	55.2	321.6	7536
16	55.2	398.4	9072

the throughput slowly until the throughput reaches a steady state value. In all three types of scenarios, RMAC outperform S-MAC. This is again due to RMAC's ability to forward PIONs over multiple hops. Although the medium is saturated when the load is high, PION frames can still be forwarded multiple hops whenever it is possible. Therefore it uses its medium access opportunity more efficiently than with RTS/CTS in S-MAC.

E. PION *Relaying Number (N)*

Finally, we evaluate the impact of the PION relaying number (N) on the network. If we increase N, the length of the DATA period will increase, thus increasing the number of hops over which a PION can be forwarded in each cycle. However, increasing the length of the DATA period increases the whole cycle time, and thus when a packet cannot be delivered to the final destination in a single cycle or when the packet is generated during a SLEEP period, the packet has to wait longer for the next DATA period to begin. Table V shows the PION relaying numbers we used in our simulations, as well as their corresponding cycle time and DATA period time. We use the same light load traffic in the realistic scenario in this evaluation. The interval time for the data generation is 50 seconds and each simulation runs for 10300 seconds of simulated time.

Figure 14 shows the average packet delivery latency for each path length in the realistic scenario. From the figure, it is hard to tell which value of N is the best fit for all the path lengths. For the nodes within 5 hops away from the sink node, $N = 2$ and $N = 4$ provide the best end-to-end delivery latency. For the nodes that are 5 to 10 hops away from the sink, $N = 8$ performs the best. And for the nodes that are even farther away from the sink, $N = 12$ and $N = 16$, have the lowest delivery latency. However, these results show that for a flow with a path length of k hops, PION relaying number $N = k$ should provide the lowest average packet delivery latency. This is because all the packets can be delivered within a single cycle and the DATA period is just long enough to allow this to happen.

In real sensor network path length varies and a fixed number N may not perform the best in all the cases. However, designers can select the number N such that most of the nodes, or the nodes sensing areas or events with higher priority, can be delivered within a single cycle. For example, in our scenario here, a value of N between 8 and 12 may be the best choice, since according to the path length histogram in Figure 8, most of the nodes are within 7 to 13 hops away from the sink.

V. RELATED WORK

Power efficient MAC protocols for sensor networks can generally be divided into two categories: those based on on-demand wakeup and those based on scheduled wakeups. In on-demand wakeup protocols, nodes use some form of out-of-band signaling technique, usually through a separate radio, to wake up nodes (e.g., [6], [7], [8]), adding extra cost to the network deployment. Scheduled wakeup MAC schemes can be further divided based on their synchronization requirements. Asynchronous schemes (e.g., [9], [10], [11]), although simple to implement, are less efficient than synchronous schemes and cannot provide guarantees on the worst-case delay. For synchronous scheduled wakeup protocols, TDMA and duty cycling are the most commonly used techniques. Although TDMA protocols (e.g., [12], [13]) are usually designed to create contention-free medium access for communication, they can schedule the wake-ups of the sensor nodes as well. However, TDMA protocols require a scheme for slot allocation and management, which can be difficult if the network topology is dense and dynamic.

S-MAC [1] is one of the first duty-cycle based MAC protocols for sensor networks. If there is no packet to receive during the active period, T-MAC [14] adapts the duty-cycle of the protocol by dynamically ending the active period of duty cycle. Both of these protocols incur high delay in multihop packet delivery, since a packet can be delivered over only a single hop in a single active/sleep period.

In a later refinement of their work, S-MAC was modified to include adaptive listening [4]. When a node overhears an RTS or CTS, the node wakes up for a short period of time after the transmission of the packet for which the CTS was intended. If the node is the next-hop node, then it can immediately receive the packet from its neighbor. Thus, adaptive listening can deliver a packet up to 2 hops per cycle. However, adaptive

listening also consumes more energy, since many neighboring nodes receive an RTS or CTS and stay awake, but only one of them is the next hop.

DMAC [15] overcame the latency problem for the specific communication pattern of a tree by offsetting the sleep schedule of a sensor node (like a pipeline) by an amount dependent on the level in the tree at which the node lies. In DMAC, not all nodes on a multihop path are aware of the data delivery, thus leading to interruption in forwarding. Also, trees need to be rebuilt if the network has different communication patterns. A similar pipelining scheme has also been proposed by Cao et al. [16] and in the fast path algorithm proposed by Li et al. [17]; these works, however, did not discuss in detail on how to handle multiple schedules and the potential pipeline stage conflicts of the multiple schedules. Abtin et al. [18] suggest a new wakeup scheme that takes advantage of the multiple routes usually available from a sensor node to the sink node. They use this in conjunction with the pipelined scheme in order to improve energy efficiency while maintaining delay bounds. Finally, Lu et al. [19] have proved that in the presence of arbitrary communication, scheduling wakeups in order to minimize the end-to-end delay is NP-hard. As a result, existing pipelined schemes are all suboptimal.

Compared with the above scheduled wakeup mechanisms, RMAC is unique, as its wakeup scheduling algorithm is entirely integrated into its medium access mechanism. Therefore the scheduling algorithm is *fully distributed* and *semi-on demand*. The pipelined schedule in RMAC is set up only when there is data to be delivered. The schedule is also *fully dynamic* and fits arbitrary communication patterns: schedules come and go in each cycle, and no extra messages are needed to set up or cancel the schedules other than the medium access control frames. RMAC achieves these features by using its unique PION multihop forwarding mechanism, which not only improves the end-to-end latency but also provides a better contention handling solution in sensor networks.

VI. Conclusions and Future Work

Duty cycle mechanisms have been used in sensor networks to improve energy efficiency, but they also introduce significant increase in end-to-end delivery latency and poor contention handling as well. We have presented the design and evaluation RMAC (the Routing enhanced MAC protocol) as a duty-cycle MAC protocol that is capable of multihop data delivery in a single operational cycle. RMAC exploits cross-layer routing information to allow its control PION (Pioneer) frame to set up a multihop schedule for subsequent forwarding of a data frame. Each node along the forwarding path then wakes up at the correct scheduled time to allow it to receive and forward the data frame. Our simulation evaluation shows RMAC's advantages in reducing delivery latency and in better handling contention, and show that RMAC achieves energy efficiency and throughput improvement as well.

Despite the potential shown by RMAC, there are many issues left open for future research. We are currently exploring use of RMAC's PION mechanism in an asynchronous environment, such as a wireless mesh network. Also, theoretical analysis of RMAC could guide us in the future exploration. Finally, the PION mechanism increases the complexity in packet handling, which may have some negative effects in a real implementation of RMAC on a sensor network platform, such as TinyOS.

References

[1] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Proc. INFOCOM 2002*, Jun. 2002, pp. 1567–1576.

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

[3] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proc. First International Conference on Embedded Networked Sensor Systems (SenSys 2003)*, Nov. 2003, pp. 138–149.

[4] W. Ye, J. Heidemann, and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 3, pp. 493–506, Jun. 2004.

[5] G. Anastasi, A. Falchi, A. Passarella, M. Conti, and E. Gregori, "Performance measurements of motes sensor networks," in *Proc. 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM 2004)*, Oct. 2004, pp. 174–181.

[6] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava, "Topology management for sensor networks: Exploiting latency and density," in *Proc. Third ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2002)*, Jun. 2002, pp. 135–145.

[7] M. J. Miller and N. H. Vaidya, "Power save mechanisms for multi-hop wireless networks," in *Proc. First International Conference on Broadband Networks (BROADNETS 2004)*, Oct. 2004, pp. 518–526.

[8] S. Singh and C. S. Raghavendra, "PAMAS: Power aware multi-access protocol with signalling for ad hoc networks," *SIGCOMM Computer Communications Review*, vol. 28, no. 3, pp. 5–26, 1998.

[9] Y.-C. Tseng, C.-S. Hsu, and T.-Y. Hsieh, "Power-saving protocols for IEEE 802.11-based multi-hop ad hoc networks," in *Proceedings of INFOCOM 2002*, vol. 1, Jun. 2002, pp. 200– 209.

[10] R. Zheng, J. C. Hou, and L. Sha, "Asynchronous wakeup for ad hoc networks," in *Proc. Fourth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2003)*, Jun. 2003, pp. 35–45.

[11] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proc. Second International Conference on Embedded Networked Sensor Systems (SenSys 2004)*, Nov. 2004, pp. 95–107.

[12] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, Oct. 2002.

[13] S. S. Kulkarni and M. Arumugam, "TDMA service for sensor networks," in *Proc. 24th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW 2004)*, May 2004, pp. 604–609.

[14] T. van Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," in *Proc. First International Conference on Embedded Networked Sensor Systems (SenSys 2003)*, Nov. 2003, pp. 171–180.

[15] G. Lu, B. Krishnamachari, and C. S. Raghavendra, "An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks," in *Proc. 18th International Parallel and Distributed Processing Symposium (IPDPS 2004)*, Apr. 2004.

[16] Q. Cao, T. Abdelzaher, T. He, and J. Stankovic, "Towards optimal sleep scheduling in sensor networks for rare event detection," in *Proc. Fourth International Conference on Information Processing in Sensor Networks (IPSN 2005)*, Apr. 2005, pp. 20–27.

[17] Y. Li, W. Ye, and J. Heidemann, "Energy and latency control in low duty cycle MAC protocols," in *Proc. 2005 IEEE Wireless Communications and Networking Conference (WCNC 2005)*, vol. 2, Mar. 2005, pp. 676–682.

[18] A. Keshavarzian, H. Lee, and L. Venkatraman, "Wakeup scheduling in wireless sensor networks," in *Proc. Seventh ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2006)*, May 2006, pp. 322–333.

[19] G. Lu, N. Sadagopan, B. Krishnamachari, and A. Goel, "Delay efficient sleep scheduling in wireless sensor networks," in *Proc. INFOCOM 2005*, vol. 4, Mar. 2005, pp. 2470–2481.