Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2013, Article ID 784509, 8 pages
http://dx.doi.org/10.1155/2013/784509

Research Article

A+MAC: A Streamlined Variable Duty-Cycle MAC Protocol for Wireless Sensor Networks

Sang Hoon Lee and Lynn Choi

School of Electrical Engineering, Korea University, Seoul 136-701, Republic of Korea

Correspondence should be addressed to Lynn Choi; lchoi@korea.ac.kr

Received 25 March 2013; Revised 24 July 2013; Accepted 25 July 2013

Academic Editor: Tai-hoon Kim

To improve the energy efficiency and the transmission performance of a sensor network MAC protocol under time varying traffic conditions, recent researches have adopted a variable duty cycle operation that makes each node dynamically adjust its own wakeup and sleep schedule according to a predefined trigger condition. However, most of the existing protocols still waste energy on a long preamble packet for waking up a receiver or long idle listening for checking potential communications. To address the energy waste problem, this paper introduces a hybrid MAC protocol called A+MAC that exploits a complementary cooperation between CSMA/CA and preamble sampling. In A+MAC CSMA/CA is used for carrying out communication processes, and preamble sampling is used for checking potential communications. Therefore, A+MAC minimizes both idle listening and the length of a preamble packet by exploiting a short preamble that makes nodes check only the event occurrence. A+MAC also optimizes control packet formats and eliminates both virtual carrier sensing and a separate clock synchronization period from conventional CSMA/CA based MAC protocols. We evaluated both the energy and the network performance of the protocol by using both NS-2 and MICA2 platforms. Our experimentation results show that A+MAC can achieve an order of magnitude energy savings while providing near optimal latency compared to the existing solutions.

1. Introduction

Cyber physical systems (CPSs) have been introduced as a new method for obtaining environmental information and controlling the surrounding environments, which is a system featuring a tight combination of and coordination between the system's computational and physical elements. CPSs fall under sensor-based systems for monitoring environments and autonomous systems for controlling physical objects. Wireless sensor network (WSN) can expand CPS applications into inaccessible area by eliminating the geographical restriction of wired sensors. A few companies put commercial WSN products [1, 2] into industrial management service. However, these products are not sufficiently small and cheap to apply them to large-scale WSNs.

Since wireless sensor networks (WSNs) assume limited and irreplaceable energy source, all the commercial products [1, 2] and academia researches for sensor network MAC protocols [3–13] employ a duty cycle operation that makes nodes periodically wake up and sleep to achieve lower energy consumption. According to the controllability of duty cycle, there are two types of duty cycle operations: a fixed duty cycle operation and a variable duty cycle operation. With a fixed duty cycle operation, all the sensor nodes operate on the predefined duty cycle while nodes with a variable duty cycle operation can adjust their duty cycle on demand. Early researches [3, 4] exploit the fixed duty cycle operation. The duty cycle of nodes need to be big enough to satisfy the latency performance requirement of a given application under the worst case scenario. In other words, nodes need to frequently wake up to prepare to report an event which might not occur. Therefore, the fixed duty cycle operation is open to improving the energy efficiency.

To increase the energy efficiency and the transmission performance of a MAC protocol, recent researches [5–8] have adopted a variable duty cycle with a predefined trigger condition such as traffic occurrence [5, 8], a communication buffer state [7], and the remaining energy [6]. Although they have a different trigger condition, their approaches are the same: a busy node increases its duty cycle for high throughput, and an idle node decreases its duty cycle for low energy consumption.

To our knowledge, our previous work called AMAC [8] is one of the first WSN MAC protocols that support variable duty cycle operations. The main ideas underlying AMAC are twofold. First, each node can adjust the duration of the periodic interval dynamically depending on the network traffic. A node participating in the communication can operate with a higher duty cycle by reducing this interval. A node, which is idling for a long time, lowers its duty-cycle by increasing the interval. Thus, not all the nodes in the field wake up or sleep at the same time. Second, a node can also adjust the duration of its wakeup period dynamically depending on the traffic. A sender can broadcast a special control packet at the very beginning of an active period to inform the existence of communication. Thus, all listeners can adjust the size of its active period by receiving the control packet. Thus, when there is no traffic, a node can minimize its active period by removing unnecessary listen period for control packets. This dynamic adjustment of the active period as well as the dynamic adjustment of the periodic interval enables the duty cycle of each sensor node to adapt to the network traffic, resulting in significant energy savings for idle nodes and improved communication performance for busy nodes at the same time.

In this paper, we introduce A+MAC which is an improved version of AMAC [8]. To minimize the idle listening, A+MAC applies preamble sampling to AMAC, which is a CSMA/CA based MAC protocol. In a preamble sampling scheme, an active period can be minimized, but a sender is required to send a preamble long enough to cover the active period of the receivers. In a CSMA/CA based MAC protocol, each node coordinates its wakeup schedule with neighbor nodes, and an active period is long enough to resolve contentions and notify the communication occurrence. To exploit the minimum active period of preamble sampling on a CSMA/CA based MAC, A+MAC employs a new wakeup technique called wakeup preamble period, which is small only enough to notify the event occurrence. Before resolving contentions, each node only checks the existence of traffic during the wakeup preamble period. If a node detects carrier signal during this period, then the busy channel tells all the listeners to keep awake for preparing following communication. Since A+MAC coordinates the wakeup schedule of nodes, it requires a preamble short enough to cover the wakeup preamble period. Therefore, A+MAC can minimize both the idle listening and the length of a wakeup preamble. In addition, A+MAC simplifies the existing CSMA/CA based MAC protocols by optimizing control packet formats and eliminating both virtual carrier sensing and a separate clock synchronization period.

The rest of this paper is organized as follows. In Section 2 we survey the existing variable duty cycling schemes. In Section 3 we introduce A+MAC. In Section 4 we show the simulation results of A+MAC. In Section 5 we present the real world performance of A+MAC with MICA2 motes. In Section 6 we conclude the paper.

2. Related Works

Existing contention-based WSN MAC protocols can be classified into synchronous protocols and asynchronous protocols. In synchronous protocols [3, 6–8], each node coordinates its wakeup schedule with its neighbor nodes. For example, in SMAC [3], all the nodes wake up at the same time to listen if there is any communication. In asynchronous protocols [4, 5, 14], a rather different approach called preamble sampling is used. A sender is required to send a preamble long enough to cover the active period of the receivers. If a receiver hears a preamble, it can further extend its wakeup and participate in the communication. This allows each node to wake up asynchronously.

SMAC [3] is one of the pioneering works in contention-based MAC protocols, specifically targeting sensor networks. SMAC proposes a periodic sleep and wakeup, providing a low duty cycle operation. To address the bandwidth and latency issues raised by the periodic wakeup, they also propose a message passing to send a long message of multiple packets in a single transaction by trading off the fragment-level fairness.

A rather different approach called preamble sampling [14] was used in Tiny OS project [15]. In the protocol, a sender is required to send a preamble long enough to cover the active period of the receivers. This can not only minimize idle listening, but it also allows each node to wake up asynchronously independent of other nodes. Thus, no network-wide clock synchronization is necessary. This idea is extended by WiseMAC [4]. In WiseMAC a sender can minimize the length of this wakeup preamble by receiving the sampling schedules of its neighbors during communication, reducing both the preamble transmission overhead and the preamble sampling time of the receivers. A recent scheme called X-MAC [16] is also based on preamble sampling but uses multiple short preambles containing receiver address information. With the address information, overhearing nodes can immediately go back to sleep while only the intended receiver participates in the communication by replying an ACK packet. By hearing the ACK packet, the sender can stop sending the preambles, reducing the preamble transmission overhead.

The authors of SMAC proposed SCP-MAC [17] that applies preamble sampling to a slotted CSMA protocol. Similar to A+MAC, a sender transmits a short preamble to its receiver before transmitting a data packet. After the successful transmission of the preamble, the sender transmits its data packet without resorting to RTS and CTS packets. Due to the absence of the control packets, SCP-MAC may suffer from frequent contentions and collisions. To mitigate this issue SCP-MAC uses two separate contention window. Before sending a preamble, a sender with SCP-MAC performs carrier sense during the first contention window. Each sender transmits its preamble from a random slot to the last slot of the first contention window. Therefore, the length of a preamble depends on the random slot number selected within the contention window. Only nodes that successfully transmit a preamble enter the second contention window to transmit a data packet. However, under the burst traffic pattern, the two-phase contention window may not effectively reduce collisions. Therefore, the authors of SCP-MAC suggest the optional use of RTS/CTS packets depending on user configurations.

However, SCP-MAC has several drawbacks. The first problem is the energy waste due to overhearing. Although

SCP-MAC provides an overhearing avoidance technique by sleeping upon the receipt of a header for a different destination, this approach requires low level access to hardware that cannot be supported by sensor platforms with a packet level radio such as MicaZ and TelosB that use Chipcon CC2420. The second problem is the performance degradation due to collisions. SCP-MAC basically uses only contention windows to avoid collisions. This approach would increase packet loss and degrade the energy efficiency and throughput due to the resulting collisions. To address this issue the authors of SCP-MAC suggest the optional use of RTS/CTS packets depending on user configurations. But, if we use RTS/CTS control packets, the first contention window would be unnecessary and it may cause nodes to waste energy for the contention window.

To our knowledge, AMAC [8] is the first sensor network MAC protocol that can support variable duty cycle operations. The main ideas underlying AMAC are twofold. First, each node can adjust the duration of the periodic interval depending on the network traffic. Second, a node can also adjust the duration of its active period depending on the traffic. This dynamic adjustment of both the active period and the periodic interval enables the duty cycle of each sensor node to adapt to the network traffic, resulting in significant energy savings for idle nodes and improved communication performance for busy nodes at the same time. In comparison with AMAC, following other variable duty cycling schemes [5, 6, 8] may be regarded as a subset of AMAC since they only control the wakeup interval.

A few more recent schemes [5–7] have studied the use of dynamic duty cycle operations. In PL-MAC [6] each node can reduce its duty cycle if its remaining energy is not enough to guarantee the predefined lifetime. If the expected lifetime of a node is longer than the predefined lifetime, a node may increase its duty cycle to reduce the communication latency. DR-MAC [7] focuses on the traffic condition. If a packet starts being buffered as traffic increases, a source node may double its duty cycle by transmitting an extra control packet to notify new duty cycle information. A+MAC is different from PL-MAC and DR-MAC in that it can adjust the duration of an active period as well as the duration of the periodic interval. Compared to PL-MAC and DR-MAC, A+MAC does not require extra control packets to adjust duty cycle.

Different from PL-MAC, DR-MAC, and A+MAC, MaxMAC [5] exploits a variable duty cycle concept on a preamble-based MAC scheme. MaxMAC gradually reduces cycle time of nodes when traffic exceeds a predefined threshold while A+MAC uses the minimum cycle time reduction which will be discussed in Section 3. When a single packet is delivered, the gradual cycle time reduction may reduce the energy consumption from overwork due to the minimum cycle time reduction which makes nodes operate with minimum cycle time. However, when there is burst traffic, it would increase the message latency until traffic exceeds a predefined threshold.

3. A+MAC Design

3.1. Simplification of CSMA/CA. Control packets used in CSMA/CA based MAC protocols for WSNs originate from the conventional wireless MAC protocol such as IEEE 802.11. To reduce the transmission overhead for control packets, A+MAC simplifies the control packet format as illustrated in Figure 1. First, A+MAC eliminates the duration field from RTS and CTS packets. The duration field is used for virtual carrier sensing. However, if nodes overhear RTS or CTS, they go back to sleep. Therefore, we no longer need virtual carrier sensing. Second, A+MAC removes the transmitter address field from CTS and ACK packets since it is simply unnecessary; the receiver of CTS and ACK packets already knows who will send the packets. Since A+MAC does not use a SYNC packet, it also removes the SYNC interval from SMAC, thus eliminating unnecessary idle listening.

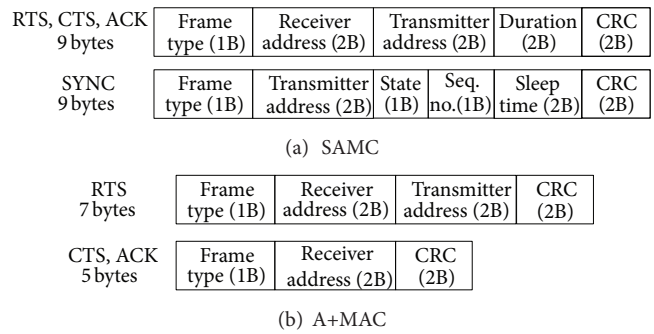

Figure 1: Control packet formats used in SMAC and A+MAC. The number in parentheses denotes the size of each field in bytes.

3.2. Wakeup Preamble. Most of the existing protocols still waste energy on a long preamble packet for waking up a receiver or long idle listening for checking potential communications. This issue can be addressed by a complementary cooperation between CSMA/CA and preamble sampling; nodes selectively benefit from advantages of two approaches: contention resolution ability and synchronized schedules of CSMA/CA based scheduling and short idle listening of preamble sampling. To accomplish this, A+MAC exploits preamble sampling period called wakeup preamble period at the beginning of a cycle as shown in Figure 2. During the wakeup preamble period, each node checks only the channel state. If channel is busy, then all the listeners extend their active period to include SYNC/RTS periods. Otherwise, the node goes back to sleep immediately, reducing the duration of the active period and thus minimizing idle listening without performance loss. A wakeup preamble contains no information since it only notifies the presence of communications. Therefore, A+MAC tolerates collisions between multiple wakeup preambles.

Since with A+MAC each node coordinates wakeup schedule with neighbor nodes, a sender can send a short preamble just at the receiver's wakeup preamble period. In A+MAC the length of the period, P_w, has to be long enough to check the channel state. P_w depends on the time for radio initialization, radio turn-on, switching a communication module to Rx or Tx, and sampling channel. For an example, CC1000 consumes 2.5 ms for all the operations [15]. Note that clock synchronization is carried out before clock drift

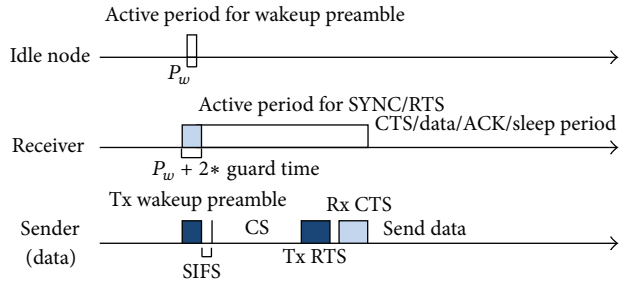

Figure 2: Operations of nodes in A+MAC.

becomes bigger than a guard time [17]. The guard time usually depends on the accuracy of a clock, synchronization error, and channel switching time. Since a sender's clock may be faster or slower than a receiver's clock, a wakeup preamble is as long as (P_w + 2 ∗ guard time).

3.3. Variable Cycle Time. In A+MAC each node can dynamically change its cycle time depending on the traffic by extending or reducing its sleep period. This requires a careful coordination of the cycle time. For such coordination A+MAC requires the cycle time of a node to be defined as a 2^n multiple of the minimum cycle time T. The maximum cycle time is determined by the latency requirement of an application. However, a cycle time may not be changed any time to avoid wakeup schedule discrepancies. Assume that a node can change its cycle time every T. A node A may change its cycle time from T to $2T$ at odd time units while another node B may change its cycle time from T to $2T$ at even time units. However, they can never synchronize each other. To avoid such a wakeup skew, each node must change its cycle time on the natural cycle alignment boundary. For example, if a node with a cycle time of $8T$ wants to double its cycle time, it must change its cycle time at a multiple of $16T$, that is, 0, 16, 32, and 48 time units. This way, a slower node can always synchronize with faster nodes. For instance, a node with a cycle time of $2T$ is able to synchronize with the nodes with $2T$, $4T$, $8T$, and so on. This is illustrated in Figure 3(a).

Application requirements would restrict the bounds of the maximum cycle time, T_{MAX}, that affects both the energy consumption during an idle state and the sleep delay of the first packet of burst traffic. The upper bound of T_{MAX} would depend on networking performance such as the tolerable packet latency and the throughput requirement while the lower bound of T_{MAX} would be affected by the energy requirement or the lifetime requirement. Although the characteristics of traffic also affect the energy consumption and the latency performance, it is hard to precisely predict the future traffic. Therefore, an administrator can decide T_{MAX} in accordance with the given application requirements and the characteristics of traffic assumed by the application.

3.4. Cycle Time Adjustment. In A+MAC, the decision of the cycle time adjustment is controlled by the traffic. If there is no traffic and the current cycle is at the natural alignment boundary for a new cycle time, all the idle listeners double

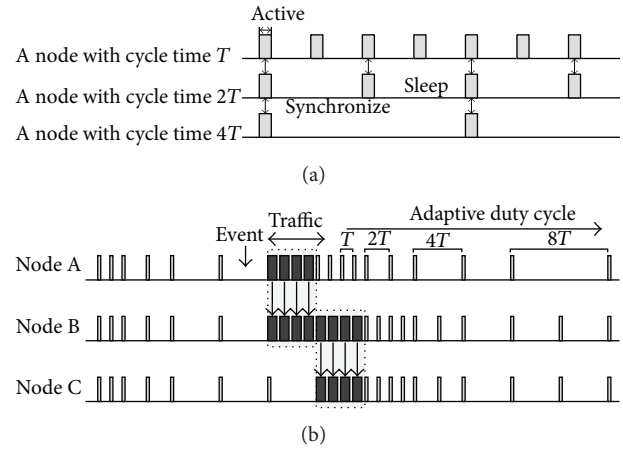

Figure 3: (a) Synchronization among the nodes with different cycle times, (b) dynamic cycle time adjustment.

their cycle times. This strategy is called automatic doubling, which is illustrated in Figure 3(b). Thus, a long idleness would cause a sensor node to eventually operate with the maximum cycle time, leading to the maximum power saving mode. When a communication event occurs, the cycle time reduction is performed. A receiver reduces its cycle time to the minimum cycle time T if the node is specified as the receiver in the RTS packet. Note that T implies that the node operates with a full speed to process the on-going event. This strategy is called the minimum reduction, which generally leads to the best result. Reducing the cycle time gradually would increase the message latency without saving any more energy.

3.5. Schedule Synchronization. One key issue in A+MAC is how a node can synchronize its active period with its neighbors. For this purpose every node maintains a schedule table, which records the cycle time of its neighbors. Initially, this value is initially set to the minimum time T. Since then each node updates its schedule table independently every cycle. The absence of the wakeup preamble suggests that all of its neighbors are idle during the current cycle. Thus, each node can keep track of the cycle time adjustment of its neighbors without explicitly asking the schedule information. If a neighbor has been idle and the current cycle is at the natural cycle alignment boundary, then the node updates the cycle time of its neighbor by doubling its cycle time. If there is an RTS packet, all the listeners update the schedule of both the sender and the receiver by setting their cycle time to the minimum. This way, each node can perform the schedule table update independently for every cycle without incurring any extra traffic.

3.6. Clock Synchronization. Since each node coordinates its wakeup schedule with neighbor nodes, A+MAC requires periodic clock synchronization. Most of the existing WSN MAC protocols derived from SMAC use a SYNC packet to perform broadcast-based clock synchronization, which requires a separate clock synchronization period. A+MAC

can also use the same synchronization scheme. In this case, a node may consume more energy due to the additional SYNC period, diminishing the energy saving benefits of preamble sampling. In addition, broadcasting node cannot ensure that all the neighbor nodes receive its SYNC packet. Therefore, a node which does not receive the SYNC packet may fail to synchronize its clock with its neighbor. This would lead to communication failures since nodes' wakeup schedule may be in discord with each other.

To address the synchronization failure and to reduce the energy consumption due to a separate clock synchronization period, A+MAC proposes an on-demand clock synchronization that can be embedded in data communications. Note that not all the nodes have to synchronize their clocks at the same time since they have different clock accuracy with reference to each other. Therefore, A+MAC allows each node to dynamically adjust its synchronization period to its neighbor node, which is the maximum time interval between two consecutive synchronization processes. Each node has to synchronize its clock at least once during this period. The length of the synchronization period depends on the amount of clock skew from the neighbor nodes. We may have to use the largest value among the clock skews from the neighbor nodes to calculate the synchronization period. To accomplish this each node monitors the clock drift from its neighbor nodes by overhearing neighbor's synchronization processes. When a node overhears its neighbor's packet containing the clock information of the neighbor node multiple times, it can calculate the clock skew of the node. Eventually, each node can figure out its neighbor that has the largest clock skew compared to its own clock.

For further optimization, A+MAC can exploit the convergent traffic pattern of WSNs. Due to the converge-cast traffic pattern, the communication traffic would form a tree shape. In a tree topology each node transmits a data packet only to its parent. Therefore, each node has to synchronize its clock only with its parent's clock. Thus, each node can calculate its synchronization period by using the clock skew from its parent. After a node carries out a clock synchronization process, the clock offsets from its children are changed. To avoid a communication failure, each node notifies its children of the new adjusted clock information. Then, the children update their synchronization period according to the received information. Since all the nodes synchronize their clock with their parent clock, a sink node which would be the root of a communication tree plays the role of a reference node.

4. Experimentation and Results

We have implemented A+MAC, AMAC, X-MAC, MaxMAC, and SMAC on NS-2 simulator [18]. X-MAC is one of the representative WSN MAC protocols that employ preamble sampling. To address the overhearing issue in the preamble sampling, X-MAC uses multiple short preambles instead of a single long preamble. By encoding the receiver address in the preamble, a node can stop overhearing if the preamble is destined to a different node. MaxMAC is one of the recent WSN MAC protocols that employ dynamic duty cycling. Unlike A-MAC and A+MAC that assume synchronous scheduling, MaxMAC applies variable duty cycle operations to asynchronous scheduling where each node wakes up independently.

Table 1: Simulation parameters.

Parameters	Values
Data packet size	100 bytes
Contention window	for SYNC: 31 slots, for others: 31 slots
Power consumption	Tx: 31 mW Rx and Idle: 15 mW Sleep: 2 μW
Minimum cycle time, T	361 ms
T_{MAX}	$4T$~$32T$
Number of nodes	200 random nodes
Communication range	250 m
Field size	2000 m × 2000 m
Simulation time	2000 seconds
Number of data packets	100 packets/source node
Packet generation interval	0~10 seconds
Guard time	1 ms

We model different versions of A+MAC by varying the maximum cycle time T_{MAX} from $4T$ to $32T$, which translates to duty cycles ranging from 0.04% to 0.31%. The maximum cycle time of AMAC and MaxMAC is $4T$. Both X-MAC and SMAC have a fixed cycle time of $4T$, which leads to 11% fixed duty cycle. We use a random topology of 200 nodes on a 2000 m ∗ 2000 m field. A source generates 100 packets to a sink. Each packet is 100 bytes long. We vary the traffic load by changing the packet generation interval on the source node from 0 to 10 seconds. Table 1 shows the parameters for our simulations.

Figure 4(a) shows the average packet latency. As the packet generation interval increases, the latencies of all the protocols start to decrease since the delay due to contention diminishes. A+MAC$_{4T}$ shows the highest performance since only the cycle time reduction is enabled. A+MAC with a larger T_{MAX} shows competitive performance compared to SMAC for packet generation rates of 0 s to 6 s since the cycle time reduction feature of A+MAC can effectively alleviate the impact of contention under burst traffic. However, when the packet generation interval is long enough, the latency of A+MAC is higher due to the long hop delay incurred by a larger maximum cycle time. It is difficult to recognize the difference between the average packet latency of A+MAC$_{4T}$ and AMAC$_{4T}$. Note that A+MAC is an energy efficient version of AMAC. To reduce idle listening, A+MAC inserts a preamble sampling period into the beginning of a cycle time. Therefore, A+MAC can efficiently reduce idle listening, but the preamble sampling period incurs additional delay. However, A+MAC can reduce cycle time by eliminating a SYNC period compared to AMAC. Although the length of a clock synchronization period depends on the number of contention slots, the difference between the reduction in cycle time due to the elimination of the SYNC period and the additional

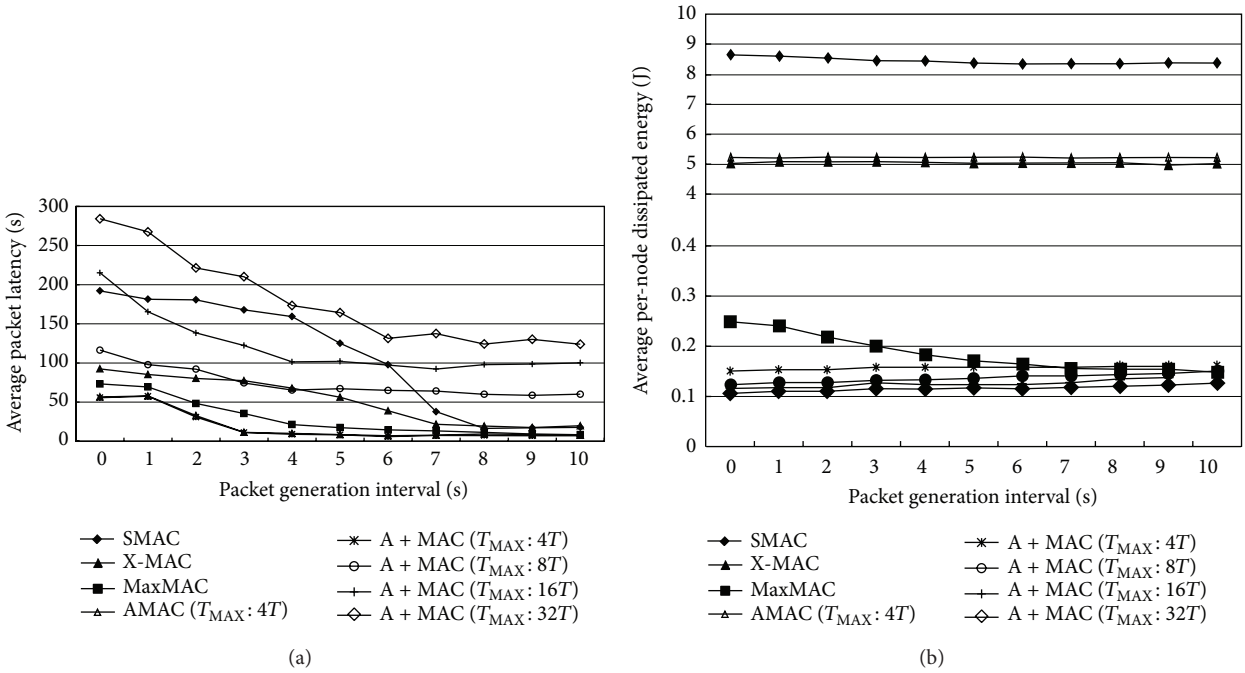

FIGURE 4: (a) The average latency and (b) the per-node energy consumption of A+MAC, X-MAC, and SMAC in terms of the packet generation interval.

delay due to preamble sampling is quite negligible compared to the minimum cycle time. Therefore, two protocols have comparable latency performance with each other.

According to our results, A+MAC is the fastest protocol among all the protocols compared assuming the same maximum cycle time 4T. Since MaxMAC has no collision avoidance, it carries out more retransmissions than AMAC and A+MAC. In addition, MaxMAC gradually reduces cycle time as the traffic volume increases while AMAC and A+MAC immediately minimize cycle time when there is traffic. Therefore, MaxMAC suffers from more sleep delay until the traffic volume is heavier than the predefined threshold. A+MAC is slightly faster than AMAC that requires a dedicated clock synchronization period. Therefore, in terms of the packet latency, A+MAC is the best protocol.

Figure 4(b) shows the average per-node energy consumption of A+MAC compared to other protocols in delivering 100 packets. Since both X-MAC and MaxMAC use a preamble sampling technique, it can efficiently reduce the idle listening. However, frequent retransmissions due to collisions under the burst traffic pattern increase their energy consumption. Moreover, each sender with X-MAC wastes energy to transmit multiple preambles. Therefore, although both X-MAC and MaxMAC reduce the energy consumption due to idle listening, the average energy consumption of X-MAC is much higher than MaxMAC. Since AMAC adaptively wakes up during RTS period according to the traffic condition, AMAC reduces idle listening in half. However, A+MAC more aggressively reduces the energy consumption due to idle listening. By eliminating idle listening due to SYNC/RTS period from active period, A+MAC$_{4T}$ can reduce the energy by more than 98% compared to SMAC although it can also substantially reduce the packet latency. A+MAC with a larger T_{MAX} can further reduce the energy consumption by increasing the cycle time adaptively. A+MAC$_{32T}$ consumes only about 1.5% of SMAC's energy consumption for all the traffic scenarios we tested.

5. Empirical Evaluation

To evaluate the real world performance we implemented A+MAC on the MICA2 platform. We constructed a linear topology with 14 MICA2 motes including a single sink node. A sink node and a source node are located at both ends. We varied the traffic load by changing the packet generation interval from 1 s to 10 s. We independently carried out the test for 10 times under each traffic load. In each test, a source node sends 20 packets that contain 100-byte data.

5.1. Communication Latency. Figure 5(a) shows the average packet latency according to the packet generation interval. The average latencies of all protocols decrease since the decline of the contention between neighbor nodes and the interference between consecutive transmissions. In particular, when there is a burst traffic nodes which operate at 32T suffer neither contention nor interference because a source node sends only 20 messages and they can be sent to next hop in a single cycle. Although A+MAC$_{32T}$ does not suffer contention and interference under burst traffic, nodes also cannot benefit from the adaptive cycle time scheme. Therefore, A+MAC$_{32T}$ has the highest message latency for burst traffic. As the packet generation interval increases, nodes operate at T due to the adaptive cycle time and the message latency is decreased to 10.1 s.

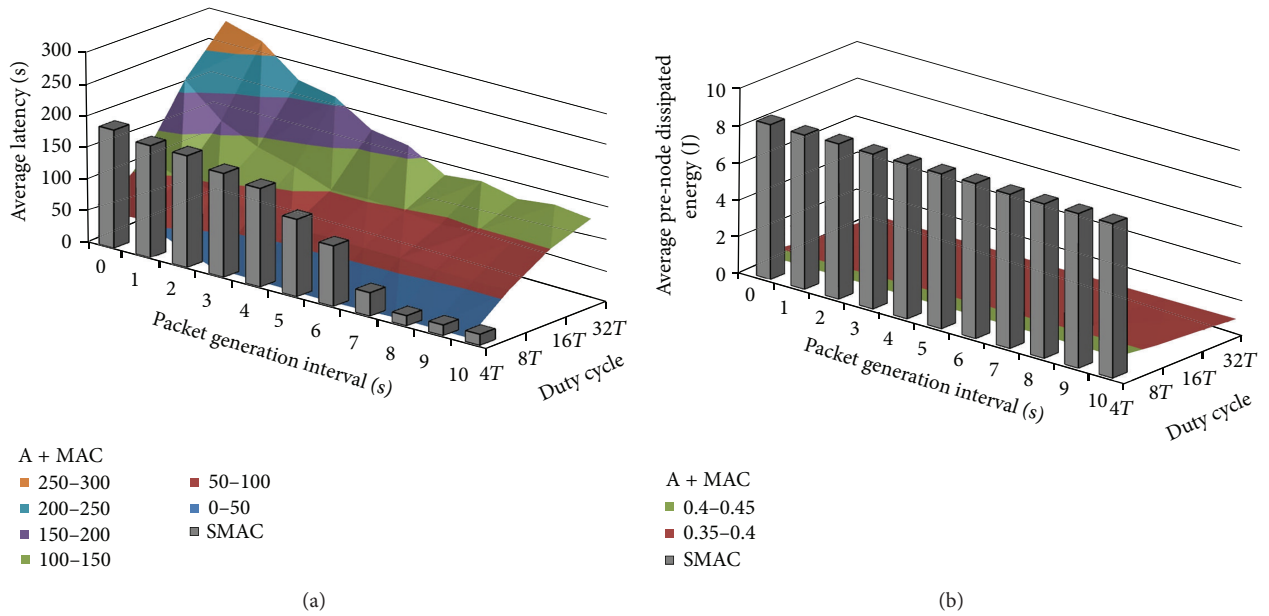

FIGURE 5: (a) The average packet latency and (b) the per-node energy consumption of A+MAC and SMAC in terms of the packet generation interval and the duty cycle.

A+MAC$_{4T}$ which has the highest communication performance enhances the performance up to 13.5% from SMAC since the adaptive cycle time can decrease the impact of sleep delay on the message latency. When the packet generation interval is higher than 6 s, the communication performance of A+MAC$_{4T}$ saturates at 7.5 s. Under these traffic loads A+MAC$_{4T}$ still has better performance than SMAC because A+MAC$_{4T}$ still operates at shorter cycle time due to the automatic doubling than SMAC. As the packet generation interval increases, SMAC catches up A+MAC$_{8T}$. In other words, A+MAC$_{8T}$ can improve the energy efficiency; moreover, it can also reduce the message latency caused by contention under burst traffic. However, when the packet generation interval is long enough or T_{MAX} is higher than 16T, the latency of this A+MAC becomes high since a larger cycle time increases the hop-to-hop delay.

5.2. Energy Consumption. It is difficult to directly measure the amount of consumed energy of a MICA2 mote. Therefore, we simulated the energy consumption of a node with AEON tool [19]. The energy model of AEON is based on measurements of node current draw and the execution of real code. The average energy consumption per node is shown in Figure 5(b). A+MAC dramatically improves the energy efficiency. A+MAC$_{4T}$ can decrease 90% of energy consumption of SMAC. As T_{MAX} increases, A+MAC can reduce up to 94% of energy consumption. The diminution in the energy consumption of A+MAC is smaller than that in the result of NS-2 simulation. This is due to the different topologies in two tests. Because all nodes in a linear topology participate in message delivery, there is no node which only overhears neighbor's communication. Therefore, the impact of overhearing on the energy consumption is not observed and the diminution in the average energy consumption of A+MAC is reduced.

6. Conclusion

With a simple tuning of existing CSMA/CA based WSN MAC protocols, A+MAC achieves both higher performance and lower energy consumption through variable duty-cycle operations. With detailed packet-level simulations we verify and demonstrate that A+MAC can achieve as much as two orders of magnitude energy savings with improved network performance compared to a previous solution.

Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2012-0002515). This research was supported by a Korea University Grant K1216991.

References

[1] Banner, "SureCross family," http://www.bannerengineering.com.

[2] MEMSIC, "eKo Environmental Systems," http://www.memsic.com.

[3] 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, 2004.

[4] C. C. Enz, A. El-Hoiydi, J. Decotignie, and V. Peiris, "WiseNET: an ultralow-power wireless sensor network solution," *Computer*, vol. 37, no. 8, pp. 62–70, 2004.

[5] P. Hurni and T. Braun, "MaxMAC: a maximally traffic-adaptive MAC protocol for wireless sensor networks," in *Proceedings of EWSN 2010*, vol. 5970 of *Lecture Notes in Computer Science*, pp. 289–305, Coimbra, Portugal, February 2010.

[6] A. Irandoost, S. Taheri, and A. Movaghar, "PL-MAC; proLonging network lifetime with a MAC layer approach in wireless sensor networks," in *Proceedings of the 2nd International Conference on Sensor Technologies and Applications (SENSORCOMM '08)*, pp. 109–114, Cap Esterel, France, August 2008.

[7] Y. I. Kim, B. G. Choi, C. S. Kim, and M. Y. Chung, "A synchronized MAC protocol considering packet generation intervals in wireless sensor networks," in *Proceedings of the IEEE Region 10 Conference (TENCON '09)*, Singapore, November 2009.

[8] S. H. Lee, J. H. Park, and L. Choi, "AMAC: traffic-adaptive sensor network MAC protocol through variable duty-cycle operations," in *Proceedings of the IEEE International Conference on Communications (ICC '07)*, pp. 3259–3264, Glasgow, UK, June 2007.

[9] A. Nandi and S. Kundu, "Energy level performance of packet delivery schemes in wireless sensor networks over fading channels," *International Journal of Future Generation Communication and Networking*, vol. 4, no. 2, 2011.

[10] A. Nikdel, A. M. Bidgoli, and M. H. Yektaie, "A new scheduling mechanism inspired of artificial immune system algorithm for wireless sensor networks," *International Journal of Smart Home*, vol. 5, no. 4, pp. 1–16, 2011.

[11] H. Lee, J. Hwang, and H. Yoe, "Energy efficient MAC protocol for ubiquitous agriculture," *International Journal of Smart Home*, vol. 4, no. 3, pp. 15–26, 2010.

[12] C. J. Fu, A. H. Lee, M. H. Jin, and C. Y. Kao, "A latency MAC protocol for wireless sensor networks," *International Journal of Future Generation Communication and Networking*, vol. 2, no. 1, 2009.

[13] S. Ghosh, P. Veeraraghavan, S. Singh, and L. Zhang, "Performance of a wireless sensor network MAC protocol with a global sleep schedule," *International Journal of Multimedia and Ubiquitous Engineering*, vol. 4, no. 2, pp. 99–114, 2009.

[14] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)*, pp. 95–107, November 2004.

[15] TinyOS Project, http://www.tinyos.net.

[16] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks," in *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys '06)*, pp. 307–320, November 2006.

[17] W. Ye, F. Silva, and J. Heidemann, "Ultra-low duty cycle MAC with scheduled channel polling," in *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys '06)*, pp. 321–334, November 2006.

[18] "The Network Simulator ns-2," http://www.isi.edu/nsnam/ns/.

[19] Avrora Simulator, http://compilers.cs.ucla.edu/avrora/.