An Asynchronous Duty Cycle Adjustment MAC Protocol for Wireless Sensor Networks

Yu-Chia Chang[1], Jehn-Ruey Jiang[1] and Jang-Ping Sheu[2]

[1]Department of Computer Science and Information Engineering
National Central University, Jhongli, 32001, Taiwan
[2]Department of Computer Science
National Tsing Hua University, Hsinchu 30013, Taiwan

Abstract

In this paper, we propose an asynchronous duty cycle adjustment MAC protocol, called ADCA, for the wireless sensor network (WSN). ADCA is a sleep/wake protocol to reduce power consumption without lowering network throughput or lengthening transmission delay. It is asynchronous; it allows each node in the WSN to set its own sleep/wake schedule independently. The media access is thus staggered and collisions are reduced. According to the statuses of previous transmission, ADCA adjusts the duty cycle length for shortening transmission delay and increasing throughput. We implement ADCA and T-MAC protocols on WSNTB (Wireless Sensor Network TestBed) and simulate them by ns-2 simulator for the sake of performance evaluation and comparison. The experiment results show that ADCA has better performance in terms of energy saving, network throughput and transmission delay.

Keywords: Wireless sensor network, sleep/wake schedule, duty cycle, energy efficiency, medium access control

1. Introduction

The rapid progress of wireless communications and micro-electro-mechanical system (MEMS) technology has made the *wireless sensor network (WSN)* a hot research topic recently. A WSN consists of many spatially distributed, resource-constrained sensor nodes equipped with microcontrollers, short-range wireless radios, and analog/digital sensors. Sensor nodes sense environmental conditions, such as temperature, light, sound, or vibration, etc., and transmit the sensed data to the sink node through multi-hop communication links. There are many applications of WSNs, such as battlefield surveillance, target tracking, environment monitoring, habitat sensing, home security, etc [1, 2].

Energy conservation is one of the most important issues in WSNs, since sensor nodes are usually powered by batteries. The radio transceiver is the most power consuming component in a sensor node. A typical radio transceiver consists of four possible modes with different power consumption: *transmitting*, *receiving*, *listening*, and *sleeping*. The first three modes are also called *active* or *wake* modes, in which more energy is consumed. For example, the power consumption of the four modes of MICAz mote [3] is 52.5, 59.1, 59.1 and 1.278 mW, respectively. Observing *idle listening*, the status that a sensor node turns on the radio to monitor wireless medium but do not receive any packets, wastes a lot of energy, some researchers propose energy-efficient *medium access control (MAC)* protocols [4, 5] to tune the radio into sleeping mode as long as possible to save energy for prolonging the network lifetime. However, the radio should be scheduled to be in wake mode periodically to monitor, send or receive data packets. Those MAC protocols that make the radio alternate between sleep and wake modes are called *sleep/wake protocols*. As shown in [6], when the *duty cycle* (i.e., active period) of the radio is reduced to 1 percent, the power consumption of the sensor node can be reduced by a factor of 50.

In addition to idle listening, sleep/wake protocols should also try to avoid *overhearing*, which occurs when a node receives data not destined to it, and to reduce *collision*, which occurs when a node receives one or more packets at the same time. The well-known RTS/CTS scheme [7] can be used to avoid overhearing as well as to reduce packet *collision* caused by the hidden terminal problem. However, its overhead is relatively large when used in in WSNs since WSN packets are usually very small. For example, in the well-known product MICA Mote, the maximum data packet size is 41 bytes and the size of an RTS/CTS packet is 18 bytes [8]. The size of an RTS/CTS packet is almost a half of one data packet, so the RTS/CTS scheme has low efficiency; other more energy-efficient mechanisms are required for WSNs.

There are many sleep/wake MAC protocols proposed in the literature trying to save energy of nodes in WSNs by avoiding idle listening, collision and/or overhearing. They can be classified into three categories: preamble-based, slot-based, and duty-cycle synchronization-based. In preamble-based protocols [9-11], nodes asynchronously turn on the radio for a short time per cycle period. Before transmitting data, a sender sends a preamble signal lasting longer than the cycle time for all neighbors to sense properly. When a node senses a preamble signal, it keeps the radio on to receive data; otherwise, it turns the radio off. Basic preamble-based protocols are simple; however, they have the drawback that the sender consumes much energy in sending long preambles and all neighbors of the sender should stay in receiving mode even though they do not send or receive data, which causes overhearing. In slot-based protocols [12-15], timers of sensor nodes are synchronized and the time axis is divided into slots for assigning to nodes. A node transmits data only within slots assigned to it. Slot-based protocols can avoid idle-listing, overhearing, and collision efficiently; however, time synchronization is expensive and slot allocation is complex and also costly. In duty cycle synchronization-based protocols [16-18], all

nodes loosely synchronize their sleep/wake schedules and periodically wake up at the same time to contend for sending data in particular periods. Duty cycle synchronization-based protocols are energy-efficient; however, schedule synchronization causes large overheads and leads to high contention, which degrades performance significantly.

In this paper, we propose an *Asynchronous Duty Cycle Adjustment (ADCA)* MAC protocol to achieve low energy consumption in WSNs without sacrificing performance, such as transmission latency or throughput. Like duty cycle synchronization-based protocols, ADCA makes nodes periodically wake up to contend to send data in specific periods. Unlike duty cycle synchronization-based protocols, ADCA allows nodes to set their own sleep/wake schedules independently. So, the schedule synchronization overhead is avoided. Furthermore, since nodes wake up at different time instances, the contention is reduced. ADCA also tries to increase the throughput and to decrease the transmission delay by adjusting two time periods: the *extended period* and the *next contention period*. The extended period is used to compensate for failed transmission, which is indicated by the happenings of overhearing or packet collision. The transmission delay can therefore be shortened dramatically. The next contention period is adjusted for nodes to adapt to current traffic conditions. If traffic is light, the length of the period is decreased; otherwise, the length is increased. In this way, channel utilization and throughput are improved.

We implement ADCA protocol on WSNTB (Wireless Sensor Network TestBed) [19] and simulate it by ns-2 simulator [20]. Since ADCA is most related to the duty cycle synchronization protocols, we also implement and simulate T-MAC, a representative duty cycle synchronization protocol, for the sake of performance comparison. The experiment results show that ADCA has better performances in terms of energy saving, network throughput and transmission delay.

The rest of the paper is organized as follows. Section 2 introduces some related sleep/wake MAC protocols. The proposed ADCA protocol is then described in Section 3. The simulation results and comparisons of protocol performance are shown in Section 4. And at last, Section 5 concludes this paper.

2. Related Work

Over the past few years, several sleep/wake MAC protocols have been developed for WSNs. The goals of those protocols are to decrease the energy consumption of wireless sensor nodes without degrading performance such as network throughput or transmission delay. The protocols can be classified into three categories: preamble-based, slot-based and duty cycle synchronization-based (see Fig. 1). Below, we introduce some representative protocols category by category.

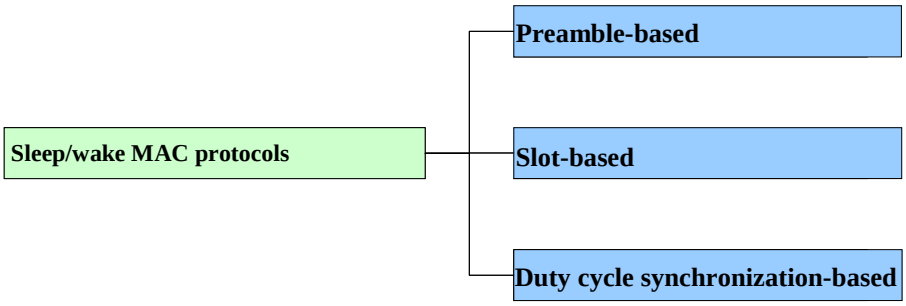

Fig. 1: The classification of sleep/wake MAC protocols for WSNs

2.1 Preamble-based protocols

B-MAC [9] uses preamble signaling for a sender to wake up the receiver. In the protocol, nodes do not need to synchronous their duty cycles. They periodically wake up for a short time at every cycle period for checking preamble signals. They keep their radios on if a preamble is detected; otherwise, they turn off the radios. It is noted that the preamble should be long enough so that the periodically waking receiver can

detect it. Consequently, the sender usually consumes a lot of energy in transmitting preamble signals, and the sender needs to wait until the receiver wakes up for sending data, which causes long transmission delay. Furthermore, since a preamble signal will wake up all neighbors of the sender to receive data, some energy is wasted due to overhearing.

Wise-MAC [10] also uses preamble signals for a sender node to notify the periodically waking receiver node of incoming data. Unlike B-MAC, Wise-MAC requires each node to keep track of the sleep/wake schedules of all its neighbors so that preamble signals can be shortened. When a sensor node has packets to send, the node will send a short wake-up preamble (called WUP) just before the receiver is active. Then it transmits data to the receiver and waits for an ACK packet from the receiver. Since WUP is short, Wise-MAC is more energy-efficient than B-MAC. However, like B-MAC, Wise-MAC requires a sender to wait until the receiver's wake-up time to send WUP and data, and thus the transmission delay may be long.

SyncWUF [11] combines both Wise-MAC's WUP concept and a new wake-up frame (WUF) technique together, where WUP is meaningless signal and WUF contains meaningful information. The idea of SyncWUF is that the sender records the receivers' schedules. To transmit a data packet, a sender node checks the receiver's schedule first. If the schedule is up-to-date, a short WUP is used as in the Wise-MAC protocol. If the schedule is out-of-date, a long WUF are used. Since a WUF is comprised of multiple short wake-up frames (SWUFs), each of which contains information like destination MAC address and the current SWUF position in the whole WUF, a receiver can decide when to turn on radio to receive data for reducing unnecessary waiting time. In SyncWUF, if a sender misses the receiver's active period, it must wait until next period to send data. The transmission delay of SyncWUF may thus be long.

2.3 Slot-based protocols

P-MAC [12] divides time axis into frames, each of which consists of two parts: the Pattern Repeat part and the Pattern Exchange part. Both parts contain many slots. During the Pattern Exchange part, nodes advertise their intended sleep/wake patterns, which represent one slot by one bit (0 for sleeping mode and 1 for active mode) and can be dynamically adjusted based on traffic conditions. And during the Pattern Repeat part, a node wakes up according to the advertised pattern. A node also wakes up at a time slot *t*, if one of its neighbors has advertised to be awake at the time slot *t* and it has data for sending to the node. Since a node decides its tentative sleep/wake schedule based only on its own traffic, P-MAC has the drawback that a receiver node may have a low duty cycle even though it has a lot of data to receive, which lengthens the transmission delay and decreases the throughput.

TRAMA [13] divides time into slots which are grouped as random access control slots and scheduled access data slots. A node arbitrarily chooses a control slot to announce the list of its one-hop neighbors and its traffic; it listens during other control slots for gathering neighboring nodes' announcements to figure out the information of topology and traffic patterns of two-hop neighbors. By the information, a node can determine the data slots in which it must sleep, transmit, or receive. A node owns a slot if the hash value of its ID and the slot number is the largest among the values calculated by all its two-hop neighbors. If a node has data to send, it sends the data in its owned slot(s). A node must stay awake to receive data in a slot when the owner of the slot indicates the node as the intended receiver in traffic pattern announcements. A node sleeps to conserve energy if it does not need to send or receive data. Because two neighboring nodes may have different set of two-hop neighbors, the two nodes may have different view of slot owners, which degrades the protocol performance.

Z-MAC [14] assigns a time slot to each node, but allows nodes to use unassigned slots through a prioritized backoff-based medium access mechanism. A slot owner has a definitely shorter backoff time than others. Therefore, when a slot owner has data to send, it always has the highest priority to do so. However, when the slot owner has no data to send, non-owners can access the slot by contention. Z-MAC needs local synchronization among senders in a two-hop neighborhood so that all two-hop neighboring nodes are assigned different slots. Such a slot assignment guarantees that no transmission by a node to any of its one-hop neighbors interferes with any transmission by its two-hop neighbors. However, the slot assignment and synchronization may lead to high costs especially when significant network changes occur frequently.

H-MAC [15] uses a slotted frame structure to achieve high energy efficiency. Each frame contains multiple short wakeup slots and multiple data slots. Each node needs to choose a wakeup slot and notifies all its neighbors of the chosen slot number with a technique proposed in HAMA [21] during the deployment phase, so that the wakeup slot number can be received properly with high probability (>0.99). It is noted that nodes can also use specific data slots to announce its chosen slot number after the deployment phase for some special occasions. The data slots are assigned on an on-demand basis. A sender *s* first sends a message during the chosen wakeup slot of receiver *r* to notify *r* of the data slots during which *s* would like to send data to *r*. The receiver *r* will then wake up during the specified data slots to receive data from *s*. Because a data slot may have multiple contenders, RTS/CTS/DATA/ACK mechanism is used to avoid collision. H-MAC has good performance in terms of channel utilization and transmission delay. However, H-MAC needs very accurate time synchronization which causes a large overhead.

2.2 Duty cycle synchronization-based protocols

S-MAC [16] is probably the most famous MAC protocol for WSNs. In S-MAC, time is divided into fixed-length cycles, each of which is further divided into SYN, contention and sleep periods (see Fig. 2). Nodes try to synchronize their duty cycle (sleep/wake) schedules by broadcasting locally SYN packets in the SYN period. A node not hearing any SYN packet will choose its own schedule and broadcast locally a SYN packet containing the schedule. On hearing the first SYN packet, a node adopts the schedule contained in the SYN packet and rebroadcasts the SYN packet. On hearing multiple, sufficiently different SYN packets, a node adopts all schedules contained in them but just rebroadcast the first SYN packet. In this manner, nodes are divided into several clusters. All nodes in a cluster have the same schedule, and nodes residing within the boundaries of two or more clusters follow the schedules of the clusters. After synchronizing schedules, nodes contend for sending data in the contention period and turn off radios to save energy in the sleep period. It is noted that traditional RTS/CTS/DATA/ACK mechanism is applied in S-MAC to reduce collisions and to avoid the hidden terminal problem. S-MAC has low cost and fair performance. However, it has the following drawbacks. First, nodes adopting multiple schedules may deplete energy soon. Second, S-MAC can only adjust the start time of duty cycles but not the cycle structure (i.e., the lengths of SYN, contention and sleep periods), so it cannot adapt to traffic conditions. Third, since nodes wake up and contend to send data at the same time, contention is high and channel utilization and throughput are thus harmed.

T-MAC [17] tries to improve S-MAC by making it adapt to traffic conditions with adjustable contention periods (see Fig. 2). In T-MAC, sensor nodes tune the radio into sleeping mode when there is no activity during a time period $T_A = 1.5\ (C + R + T)$ after the SYN period, where C is the length of the contention period, R is the

time period of *RTS* packet transmission, and *T* is a short time between the end of the *RTS* packet and the beginning of the *CTS* packet. In this way, a node can go to sleep early if there is no traffic, and a node stays awake longer when traffic is higher. Consequently, T-MAC has lower power consumption and better throughput than S-MAC under variable traffic. However, like S-MAC, T-MAC suffers from high contention due to synchronized duty cycle schedules, which casts bad influences on channel utilization and throughput.

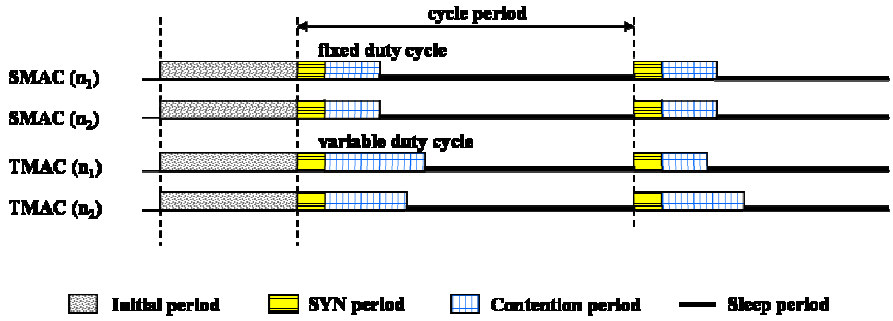

Fig. 2: The duty cycle structures of S-MAC and T-MAC protocols

U-MAC [18] also improves S-MAC by assigning different duty cycles to nodes based on channel utilization. The calculation of utilization takes transmitting time, receiving time and idle listening time into consideration. If the current utilization is larger (resp., less) than the high (resp., low) utilization threshold, the duty cycle will be increased (resp., decreased) by a pre-specified fraction. For not lengthening the transmission delay, the duty cycle will not be decreased if the average packet delay is larger than the maximum tolerable delay. And for not consuming too much energy, the duty cycle will not be increased if it is larger than a pre-specified maximum value. U-MAC can save more energy than S-MAC. However, U-MAC's performance depends heavily on the parameters for the high and low utilization thresholds, maximum delay and energy consumption. Good parameter setting varies case by case and is thus hard to derive.

3. ADCA protocol

3.1 Overview

ADCA (asynchronous duty cycle adjustment) protocol is most related to duty cycle synchronization-based protocols in the sense that nodes in ADCA periodically wake up to contend to send data in specific periods. However, unlike typical duty-cycle synchronization-based protocols that synchronize neighboring nodes' schedules, ADCA allows each node to asynchronously set its own sleep/wake schedule. In ADCA, time is divided into cycles of fixed length, and each cycle is further divided as a *contention period*, a *control period*, an *extended period* and a *sleep period* as shown in Fig. 3. When a node starts up, it broadcasts locally its own schedule and collects and stores all neighbors' schedules in the *neighbor-schedule table* for an arbitrary-length *initial period*. Nodes then start their cycle periods asynchronously, and turn on radios at the beginning of the period for data exchange and schedule broadcast; they then enter sleeping mode for conserving energy.

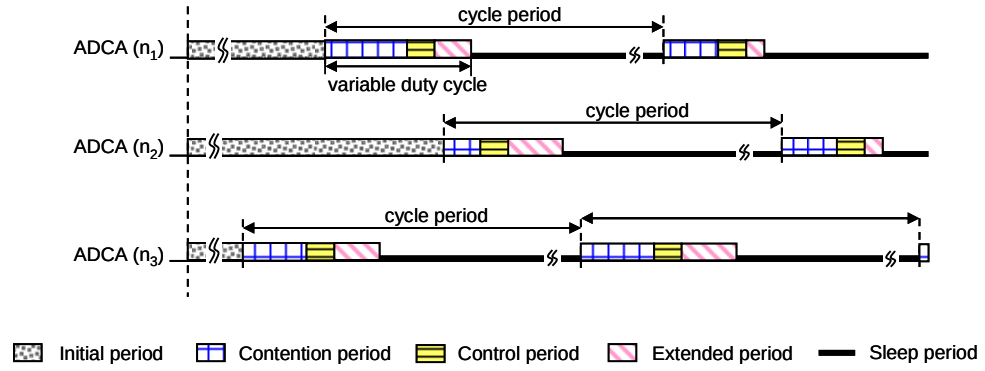

Fig. 3: The duty cycle structure of ADCA protocol

In ADCA, a node listens to the channel for possible incoming data packets during the contention period and broadcasts locally a control packet to announce its schedule during the control period. An extended period immediately follows the

control period to prolong the active time. A node turns its radio into sleeping mode to enter sleep period to save energy. When a node has a packet to send, it checks its neighbor-schedule table and contends to send the data packet during the receiver's contention period. If a sender fails to send the data packet in the receiver's contention period, it switches the radio into the receiving mode to wait for the receiver's control packet which indicates the length of receiver's extended period and next contention period. The sender then tries to retransmit the data packet in the receiver's extended period. If the transmission still fails, the sender waits for the contention period in the receiver's next cycle for retransmitting the data. It is noted that the sender in waiting can turn the radio off to save energy.

Nodes in ADCA do not synchronize their schedules; they maintain schedules independently. Therefore, the schedules are staggered and the transmission success rate and channel utilization are thus increased. Furthermore, ADCA allows nodes to dynamically adjust the contention period and the extended period based on current transmission statuses and traffic loads. In this way, the throughput is increased and the transmission delay is decreased without scanting energy efficiency. Below, we show how ADCA adjusts the two periods in the next subsection.

3.2 Duty cycle adjustment

In ADCA, a node adjusts its duty cycle according to transmission statuses and traffic loads. Each node records the time of channel idle (T_i), the time of channel busy (T_b) and the number of overheard packets (N_{oh}) during the *observed active periods*, i.e., the last extended period and the current contention period. It then calculates, at the end of the contention period (or called the *adjustment point*), the length of the extended period (A_1) and the length of the next contention period (A_2) accordingly (see Fig. 4). The node then broadcasts locally during the control period a control

packet containing its new schedule with the newly calculated extended period and contention period.

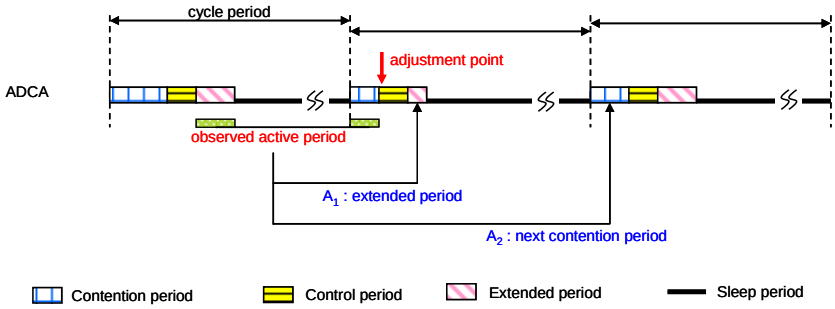

Fig. 4: The duty cycle adjustment of ADCA protocol

The length of extended period (EP) is adjusted according to Eq. 1. T_{bad} in Eq. 1 represents the time duration of collision and channel interference, and N_{oh} stands for the number of overheard packets. T_{data} is defined in Eq. 2 as the average transmission time of a data packet including the time for transmitting data (packet size/data rate), and the average random back-off time within a fixed-sized contention window of *cw* slots, each with length T_{slot}.

$$EP = \left(\left\lfloor \frac{T_{bad}}{T_{data}} \right\rfloor + N_{oh} \right) \times T_{data} \tag{1}$$

$$T_{data} = \frac{cw}{2} \times T_{slot} + \frac{packet\ \ size}{data\ \ rate} \tag{2}$$

Fig. 5 shows some bad receiving situations such as collision, overhearing and interference, which will increase the transmission delay and decrease the channel utilization. A node should lengthen the extended period to compensate for the bad receiving situations. If a receiver detects more collisions or overhearing events, it knows that the sender has smaller probability to complete data packet transmission

successfully. Therefore, the receiver's extended period is made proportional to the number of overheard packets and the duration of channel unstableness (interference) and collision.

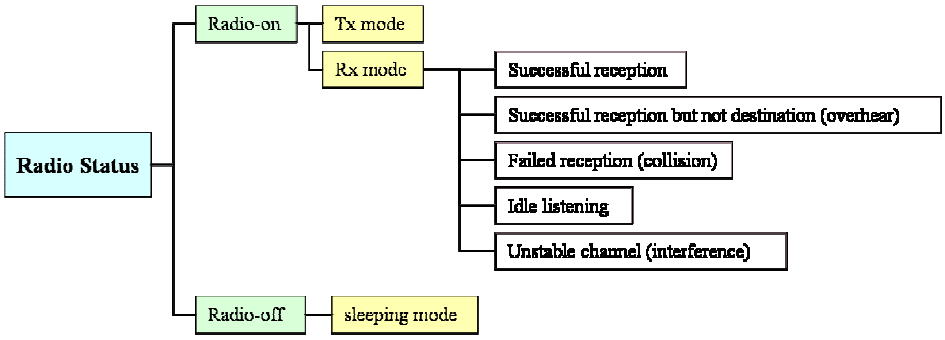

Fig. 5: The radio status of a sensor node

The next contention period adjustment is for the purpose of adapting to the traffic conditions of the observed active period. To be more precise, the length of the next contention period is proportional to traffic loads. The length of the next contention period (CP) is adjusted according to Eq. 3, where T_{rx} is the total time that a node is in the receiving mode during pervious cycle period, *CCP* means the current contention period length, T_i is the channel idle time and T_b is the channel busy time (T_i+T_b=T_{rx}). Eq. 3 takes channel idle time T_i and channel busy time T_b into consideration, and α and β are weight parameters associated with the two time spans. In general, α should be negative so that a longer channel idle time will lead to a shorter contention period, while β should be positive so that a longer channel busy time will lead to a longer contention period. The values of α and β can be determined according to specific application requirements. We suggest setting α=-1 and β=1 in this paper. Therefore, if $T_i > T_b$, then *CP* gets smaller; otherwise, *CP* gets larger. Certainly, *CP* should be larger than a pre-specified minimum value and should only last until the end of the cycle period.

$$CP = CCP \times (1 + \alpha \frac{T_i}{T_{rx}} + \beta \frac{T_b}{T_{rx}}) \quad (3)$$

4. Performance Evaluation

4.1 The Experiment Environment

As we have shown, ADCA is most related to duty cycle synchronization-based MAC protocol. Thus, we only compare it with duty cycle synchronization-based protocols. Since the performance of U-MAC is affected significantly by threshold parameter setting and the best parameter setting can only be derived after extensive experiments, we do not compare ADCA with U-MAC. And the research results in [17, 18] show that T-MAC undoubtedly has better performance than S-MAC. So, we only compare ADCA with T-MAC. We implement ADCA and T-MAC protocols on WSNTB (Wireless Sensor Network TestBed) and simulate them by ns-2 simulator for the sake of performance comparison.

WSNTB [19] is an indoor wireless sensor network testbed which consists of a number of Octopus II sensor nodes as shown in Fig. 6. Each Octopus II sensor node is equipped with a MSP430 microcontroller and a CC2420 radio module, which operates at 2.4 GHz and transmits at 250 Kbps. And each node is also attached to a USB interface that provides both power supply and a backchannel for programming and data collection. The sensor nodes in WSNTB run Tmote tools 2.04; they send data with at most three retransmissions and have “quasi-reliable” data links. Furthermore, nodes only apply the CSMA (carrier sense multiple access) scheme but not the RTS/CTS scheme to avoid collision.

For WSNTB experiments, we deploy 35 testbed nodes on the third floor of our office building, as shown in Fig. 7. Two scenarios are investigated in the experiments. One is the all-to-one scenario where all the nodes report data to a sink node periodically (see Fig. 8). The other is the end-to-end scenario where six random pairs of nodes are selected for exchanging data. The data packets and ACK packets are respectively 44 and 10 bytes in length. The traffic loads are assumed to have constant

bit rates (CBR) which are set to 1, 5, 10, 15 and 20 packets per second. And we assume routing information is already stored in nodes memory beforehand so that we can focus only on investigating the effects of MAC protocols.

Fig. 6: Octopus II sensor node

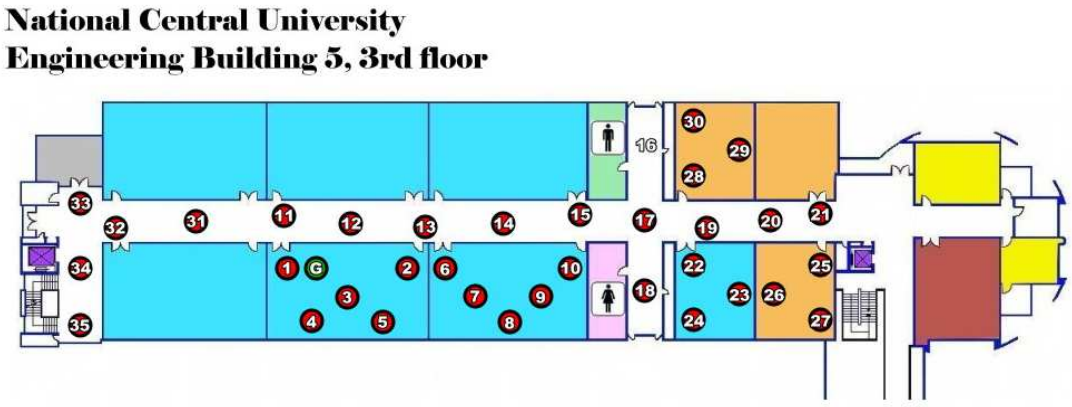

Fig. 7: The deployment of WSNTB sensor nodes

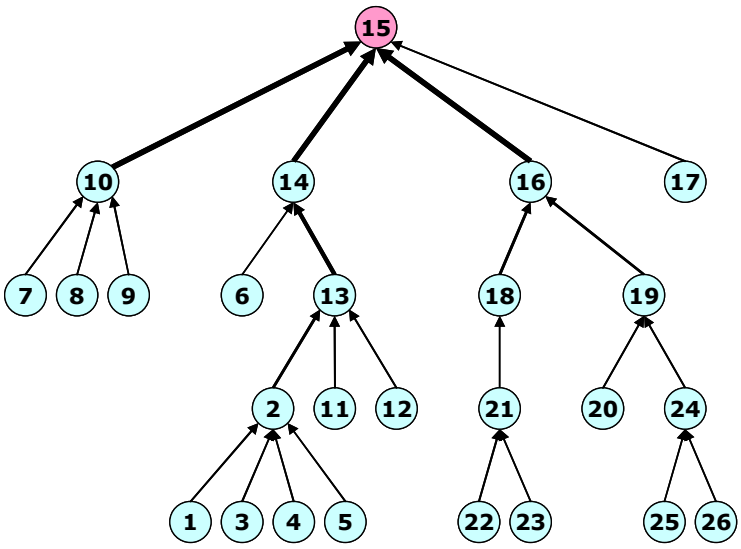

Fig. 8: The routing topology of the all-to-one scenario

We compare ADCA and T-MAC protocols by WSNTB testbed experiments and ns-2 simulations in terms of the following three metrics: (1) the *energy consumption*, which is defined as the average energy consumption of sensor nodes, (2) the *transmission delay*, which stands for the average transmission delay in a single hop, and (3) the *packet transmission success rate*, which is defined as the ratio of the number of packets received properly to the total number of packets sent. It is noted that we try to make the testbed and the simulation experiments have the same setting. However, some environmental parameters are out of our control, so the setting of the two types of experiments may not be exactly the same. For example, we assume the transmission area of a node is a circle with fixed radius and a node's neighbors are decided when nodes are deployed. This assumption can be realized easily in simulation experiments. But in testbed experiments, the practical transmission range of a node is affected by many dynamically changing environmental factors, such as the temperature, the humidity, the positions of antennas, and the interference from surroundings, etc. Therefore, testbed and simulation experiments have results with subtle differences. Below, we use ADCA (resp., T-MAC) and ADCA(sim) (resp., T-MAC(sim)) to stands for the testbed and simulation experiment results for ADCA (resp., T-MAC) protocol. Note that each experiment lasts 1000 seconds and each result is obtained by averaging outcomes of 30 experiments.

4.2 The results of the energy consumption

In this subsection, we observe the average energy consumption of sensor nodes in experiments. We make each node record the accumulated time in transmitting (tr), receiving (rx), idle listening (idle) and sleeping (slp) states during the entire experiment duration. The power consumption of the four states is 52.2, 59.1, 59.1 and 1.28 mW, respectively. At the end of the experiment, each node calculates the total

energy consumption by Eq. 4, and then the average energy consumption can be derived accordingly.

$$E = E_{tx} + E_{rx} + E_{idle} + E_{sleep} \tag{4}$$

Fig. 9 and Fig. 10 show the average energy consumption results for the all-to-one and the end-to-end scenarios. As we can see, the testbed experiment has worse results than the simulation experiment. However, both experiments show that the energy consumption of ADCA is lower than that of T-MAC. In T-MAC protocol, schedules are synchronous and nodes wake up at the same time, which results in high collision probability. ADCA also suffers from collision, but its asynchronous schedule strategy staggers the active periods of nodes. Therefore, the collision probability is decreased, the packet retransmission is reduced, and the energy is conserved. By Fig. 9, we can observe that ADCA can be 45% better than T-MAC in terms of energy consumption for the all-to-one scenario with 10 packets per second traffic. By Fig. 10, ADCA can be 42% better than T-MAC in terms of energy consumption for the end-to-end scenario with 15 packets per second traffic.

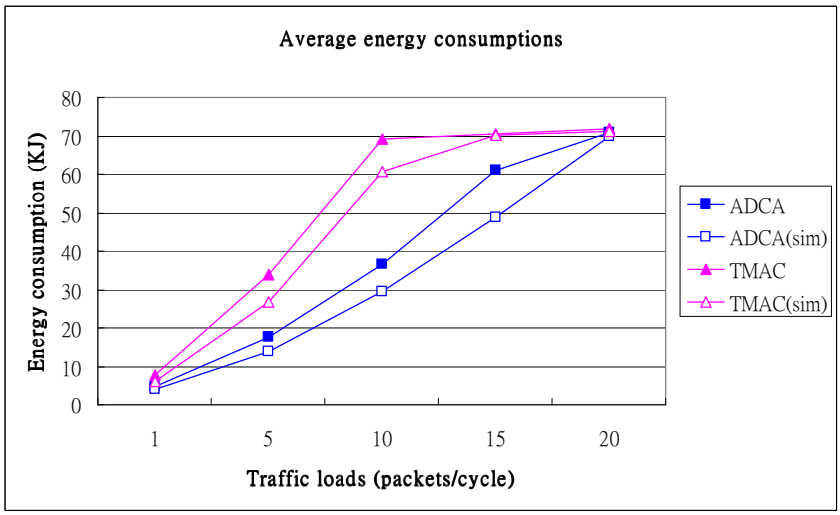

Fig. 9: The average energy consumption for the all-to-one scenario

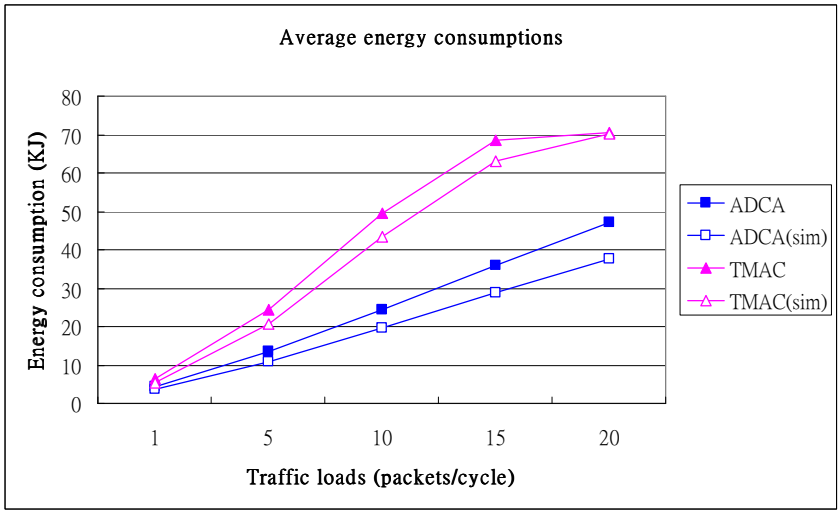

Fig. 10: The average energy consumption for the end-to-end scenario

4.3 The results of the transmission delay

In sleep/wake schedule MAC protocols, transmission delay consists of a waiting time and a processing time. The waiting time for a sender is the duration from the time the sender is ready to send a data packet to the time the receiver tunes its radio into the receiving mode. The length of the waiting time is dependent on both the cycle duration and the active/sleep ratio. Because we assume that all the nodes have the same cycle duration, the active/sleep ratio becomes the major factor affecting the waiting time. The processing time is the duration from the time the sender contends to send the receiver a data packet to the time an ACK packet is received by the sender successfully. It consists of the back-off time, packet propagating time and ACK waiting time. Therefore, the duty cycle adjustments and the collision will directly affect the transmission delay.

Fig. 11 and Fig. 12 show the results of the average one-hop transmission delay for the one-to-all and the end-to-end scenarios. The delay times of both ADCA and T-MAC grow with the traffic loads. For light traffic cases (e.g., 1 packet per second), senders in ADCA need to wait for the receiver's active period to transmit data packet,

but senders and receivers in T-MAC wake up simultaneously to handle the data transmission. Thus, T-MAC's delay time is shorter than ADCA's for light traffic cases. However, in other cases, ADCA has shorter delay than T-MAC. This is because T-MAC maintains a global schedule and thus sensor nodes contend to send data during the same period, leading longer delay. On the contrary, ADCA maintains asynchronous schedules and the number of contenders is thus decreased. Therefore, the data transmission can be staggered and the delay time is decreased.

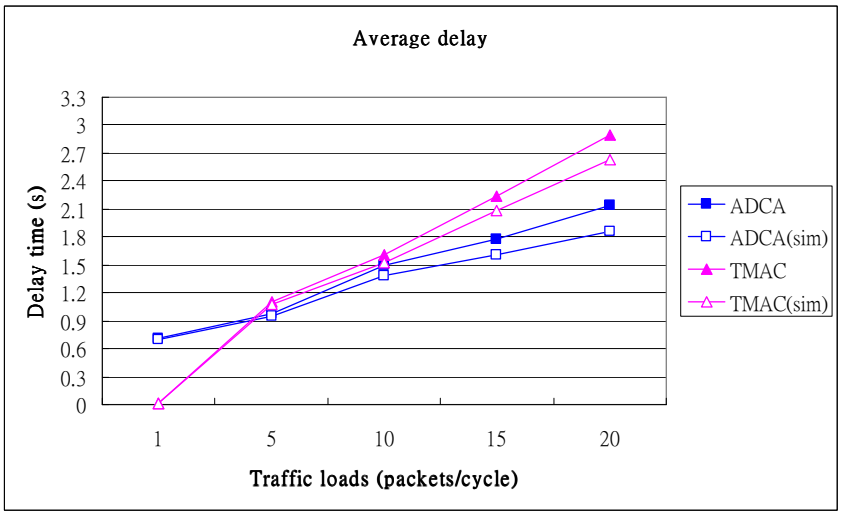

Fig. 11: The average transmission delay for the all-to-one scenario

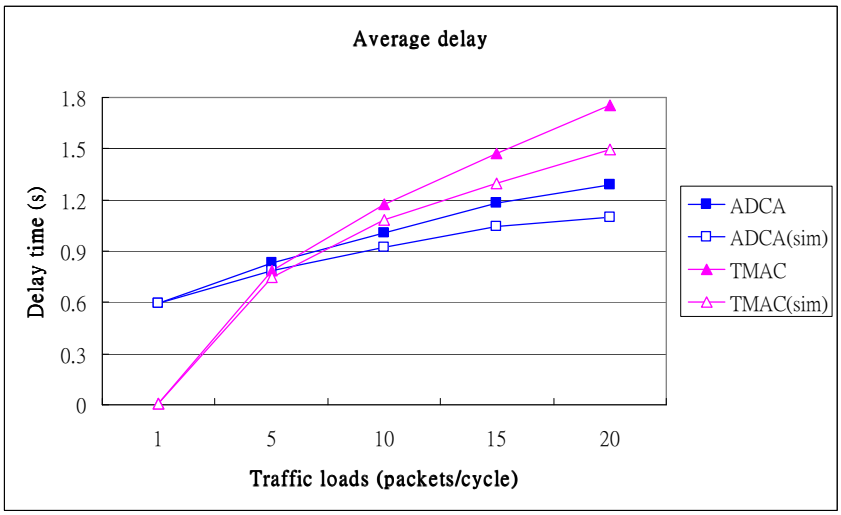

Fig. 12: The average transmission delay for the end-to-end scenario

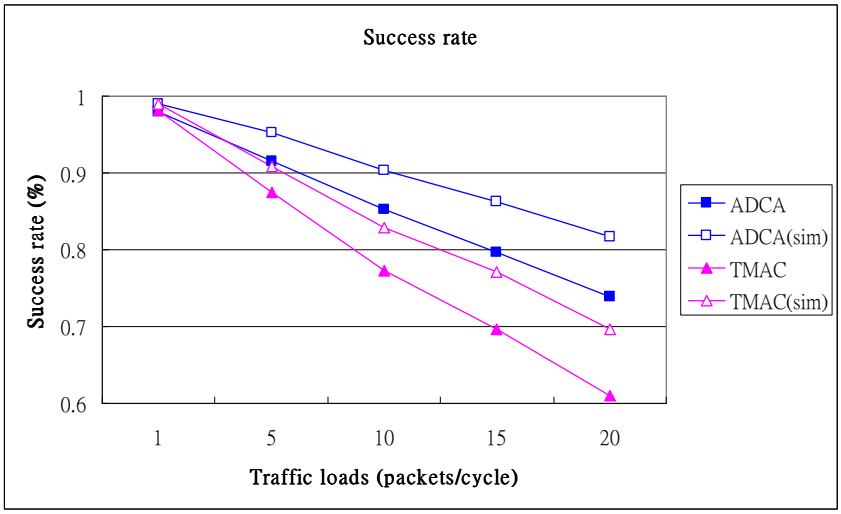

Fig. 13: The average packet transmission success rate for the all-to-one scenario

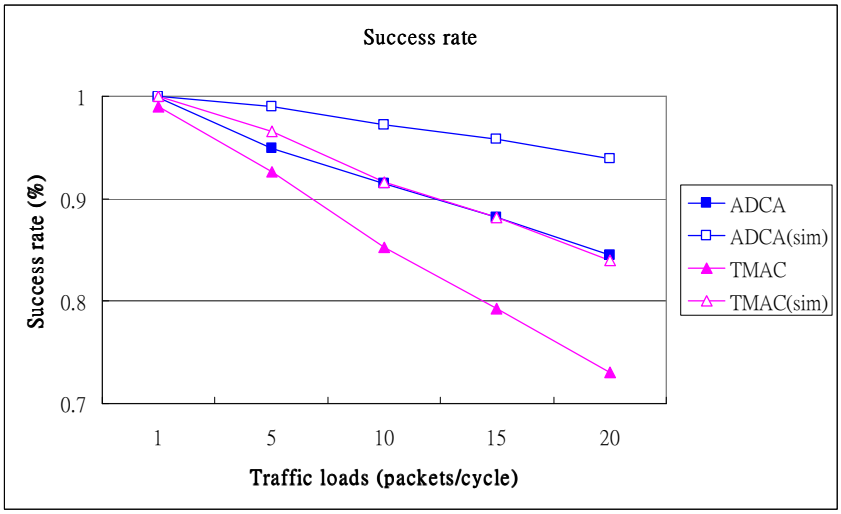

Fig. 14: The average packet transmission success rate for the end-to-end scenario

4.4 The results of the packet transmission success rate

We use the packet transmission success rate (just called the *success rate* for short) as a measurement of the throughput. It is evident that the throughput increase with the success rate. Fig. 13 and Fig. 14 show the results of the success rate for the one-to-all and the end-to-end scenarios. As the traffic load increases, the success rate goes down for both ADCA and T-MAC protocols. We can easily observe that ADCA outperforms T-MAC in terms of success rate. For example, the success rate of ADCA can be 12% (resp., 10%) higher than T-MAC for the all-to-one (resp., end-to-end) scenario, as shown in Fig. 13 (resp., Fig. 14). This is because ADCA staggers the active periods of

nodes to reduce the collision probability, and thus the packet transmission success rate is increased.

5. Conclusion

This paper presents an asynchronous duty-cycle adjustment MAC protocol, called ADCA, for saving energy of nodes in wireless sensor networks. ADCA allows nodes to keep schedules asynchronously, so data transmission is staggered and collision and overhearing are reduced. A node in ADCA tunes the radio into sleeping mode as long as possible to save energy for prolonging the network lifetime. However, it adjusts the length of the active period to improve the throughput and to reduce the transmission delay. We implement ADCA and T-MAC protocols on WSNTB and simulate them by ns-2 simulator for the sake of performance comparison. The energy consumption and the packet transmission success rate of ADCA are up to 45% (resp., 42%) and 12% (resp., 10%) better than those of T-MAC in the all-to-one (resp., end-to-end) scenario. The average one-hop transmission delay of ADCA is also shorter than that of T-MAC for most cases in the two scenarios. By the experiment results, we observe that ADCA can reduce energy consumption without sacrificing the throughput and the transmission delay.

References

[1] A. A. Ahmed, H. Shi, and Y. Shang, “A Survey on Network Protocols for Wireless Sensor Networks,” *in Proceedings of International Conference on Information Technology: Research and Education*, pp. 301 – 305, August 2003.

[2] M. A. M. Vieira, C. N. Coelho Jr., D. C. da Silva Jr., and J.M. da Mata, “Survey on Wireless Sensor Network Devices,” *in Proceedings of IEEE International Conference on Emerging Technologies and Factory Automation*, Vol. 1, pp. 537 – 544, September 2003.

[3] http://www.tinyos.net/

[4] I. Demirkol, C. Ersoy and F. Alagoz, “MAC Protocols for Wireless Sensor Networks: A Survey,” *IEEE Communications Magazine*, Vol. 44, pp. 115 – 121, April 2006.

[5] K. Kredo II, P. Mohapatra, “Medium access control in wireless sensor networks,” *Computer Networks*, Vol. 51, No. 4, pp. 961-994, 2007.

[6] J.M. Rabaey, M.J. Ammer, J.L. da Silva, D. Patel, and S. Roundry, “PicoRadio Supports Ad Hoc Ultra-Low Power Wireless Networking,” *Computer*, vol. 33, pp.42-48, July 2000.

[7] K. Xu, M. Gerla, and S. Bae, “Effectiveness of RTS/CTS handshake in IEEE 802.11 based ad hoc networks,” *Ad Hoc Networks Journal*, vol. 1, no. 1, pp. 107–123, 2003.

[8] Chipcon AS, “SmartRF CC2420 PRELIMINARY Datasheet,” rev. 1.2, February 2004.

[9] 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*, pp. 95-107, November 2004.

[10] A. El-Hoiydi and J. D. Decotignie, “WiseMAC: An Ultra Low Power MAC

Protocol for The Downlink of Infrastructure Wireless Sensor Networks," *in Proceedings of the 9th International Symposium on Computers and Communications*, Vol. 1, pp. 244-251, July 2004.

[11] X. Shi and G. Stromberg, "SyncWUF: An Ultra Low-Power MAC Protocol for Wireless Sensor Networks," *IEEE Transactions on Mobile Computing*, vol. 6, pp. 115-125, Jan. 2007.

[12] T. Zheng, S. Radhakrishnan and V. Sarangan, "PMAC: An Adaptive Energy Efficient MAC Protocol for Wireless Sensor Networks," *in Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium*, pp. 8, April 2005.

[13] V. Rajendran, K. Obraczka, and J.J. Garcia-Luna-Aceves, "Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks," *In the Journal of Wireless Networks*, Vol. 12, pp. 63-78, Feb. 2006.

[14] I. Rhee, A. Warrier, M. Aia, and J. Min, "Z-MAC: A Hybrid MAC for Wireless Sensor Networks," *in the Proceedings of the 3rd ACM Conference on Embedded Networked Sensor Systems (SenSys '05)*, pp.90-101, Nov. 2005.

[15] Heping Wang, Xiaobo Zhang and Ashfag Khokhar, "An Energy-Efficient Low-Latency MAC Protocol for Wireless Sensor Networks," *IEEE Global Telecommunications Conference (GLOBECOM '06)*, pp. 1-5, Nov. 2006.

[16] 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, pp. 493-506, June 2004.

[17] T. V. Dam and K. Langendoen, "An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks," *in Proceedings of the 1st international conference on Embedded networked sensor systems*, pp. 171-180 , November 2003.

[18] Shih-Hsien Yang, Hung-Wei Tseng, Wu E.H.-K., Gen-Huey Chen, “Utilization based duty cycle tuning MAC protocol for wireless sensor networks,” *IEEE Global Telecommunications Conference (GLOBECOM '05)*, Volume 6, pp. 5, 28 Nov.-2 Dec. 2005.

[19] Jang-Ping Sheu, Chia-Jen Chang, Chung-Yueh Sun, Wei-Kai Hu, “WSNTB: A testbed for heterogeneous wireless sensor networks,” in *Proc. of the First IEEE International Conference on Ubi-Media Computing*, 2008.

[20] http://www.isi.edu/nsnam/ns/

[21] Lichun Bao, “Hybrid channel access scheduling in Ad Hoc networks,” in *Proc. IEEE Tenth International Conference on Network Protocols (ICNP*), 2002.