An asynchronous scheduled MAC protocol for wireless sensor networks

Beakcheol Janga,⇑Jun Bum Limb, Mihail L. Sichitiuc

aDivision of Digital Media Technology, Sang Myung University, Seoul, Korea
bQualcomm Inc., Raleigh, NC, United States
cDepartment of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC, United States

Abstract

Wireless sensor networks (WSNs) comprise a number of autonomous sensors and one or more sinks to cooperatively monitor physical or environmental conditions. Energy efficiency is a key design factor of a MAC protocol for WSNs. Due to the importance of the problem, a number of energy efficient MAC protocols have been developed for WSNs. Preamble-sampling based MAC protocols (e.g., B-MAC and X-MAC) have overheads due to their preambles, and are inefficient at large wakeup intervals. SCP-MAC, a synchronous scheduled energy-efficient scheduling MAC protocol, minimizes the preamble by combining preamble sampling and scheduling techniques; however, it does not prevent energy loss due to overhearing; in addition, due to its synchronization procedure, it results in increased contention and delay. In this paper, we present an energy efficient MAC protocol for WSNs that avoids overhearing and reduces contention and delay by asynchronously scheduling the wakeup time of neighboring nodes. We provide an energy consumption analysis for multi-hop networks. To validate our design and analysis, we implement the proposed scheme in TinyOS. Experimental results show that AS-MAC considerably reduces energy consumption, packet loss and delay when compared with existing energy efficient MAC protocols.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction

Wireless sensor networks (WSNs) hold significant potential for distributed sensing applications for large geographical areas that monitor plants and animals, natural phenomena (e.g., weather pollution and earthquakes) and military battlefield surveillance [13]. Energy is often the scarcest resource of WSN nodes, and it determines the lifetime of WSNs; therefore energy efficiency is one of the most important requirements in designing a WSN, and the radio is recognized as a major source of the energy consumption in sensor nodes [7,11,22,23,30,33]. The Medium Access Control (MAC) protocol [14], in addition to controlling medium access, can be designed to reduce the energy consumption of the radio in WSNs [8]. Idle listening is often the largest source of energy waste [5,9,15,18,20,28,32,33], and duty cycling mechanism (i.e., periodically putting the radio in a sleep state) is considered as one of the necessary techniques to reduce energy consumption in WSN MAC protocols [24,25].

Several MAC protocols using duty cycling have been developed for WSNs [5,20,28,32,33]; however, some of them (e.g., S-MAC [32] and T-MAC [28]) are not energy efficient because they have long awake times. Others, like B-MAC [20] and X-MAC [5], minimize the awake-time by using Low-Power Listening (LPL); however, the use of a long preamble limits the possible energy savings at long wake-up intervals. SCP-MAC [33] uses LPL to minimize the uptime and synchronizes the wake-up time of sensor nodes to reduce the need for a long preamble. However, the requirement on synchronizing the wake-up times results in high overhearing, contention and delay penalties.

In this paper, we present a simple but very energy-efficient Asynchronous Scheduled MAC protocol (AS-MAC). AS-MAC employs duty cycling like the previous schemes...
to avoid idle listening and uses Low-Power-Listening (LPL) to minimize the periodic wakeup time. The nodes store the wakeup schedules of their neighbors; therefore they know when their neighbors wake up and they do not need to add long preambles at the beginning of transmission. AS-MAC asynchronously coordinates the wakeup times of neighboring nodes to reduce overhearing, contention and delay unavoidable in synchronous scheduled MAC protocols such as S-MAC, T-MAC, and SCP-MAC.

To determine the performance of AS-MAC, we implement our proposed scheme in TinyOS [12] on two well-known WSN platforms (TELOSB [4] and MICAz [3]). We evaluate our protocol design comparing it with two important WSN MAC protocols: CSMA (default MAC protocol) and SCP-MAC [33] (synchronous scheduled energy-efficient MAC protocol) in single and multi-hop scenarios. The experimental results show that AS-MAC considerably reduces energy consumption while providing low latency and packet loss in comparison with existing WSN MAC protocols.

The remainder of this paper is organized as follows: we provide a summary of related work in Section 2. In Section 3, the design of AS-MAC is described in detail. In Section 4, we present the energy consumption model. In Section 5, we compare the performance of AS-MAC with existing energy efficient WSN MAC protocols. Section 6 concludes this paper.

2. Related work

The preamble-sampling MAC protocols (i.e., B-MAC [20] and X-MAC [5]) exploit Low-Power-Listening (LPL) for sampling the preambles of the packets. LPL minimizes the duty cycle when there are no packet exchanges, but during transmissions, the preamble needs to be longer than the wakeup interval to guarantee that the receiver detects the channel activity. Thus, the overhead of preambles becomes large as the wakeup interval increases. X-MAC reduces the overhead of receiving long preambles by using short and strobed preambles allowing unintended receivers to sleep after receiving only one short preamble and the intended receiver to interrupt the long preamble by sending an ACK packet after receiving only one strobed preamble. However, even in X-MAC, the overhead of transmitting the preamble increases with the wakeup interval, limiting the efficiency of the protocol at very low duty cycles.

S-MAC [32] and T-MAC [28] are scheduling MAC protocols that synchronize the wakeup schedules of sensor nodes in a neighborhood. To synchronize the wakeup schedule, nodes periodically exchange SYNC packets. S-MAC was the first duty cycling WSN MAC protocol where all nodes in a neighborhood simultaneously wake up and listen to the channel. A drawback of this scheme is the need for a long periodic wakeup time that has to include the collision avoidance backoff, RTS-CTS exchange and compensation for clock drift as well as waiting for eventual transmissions from the neighbors. T-MAC reduces the long wakeup time of S-MAC by using a timer that shortens the wakeup time if the channel is idle; however its wakeup time is also much longer than LPL time because the timeout should be longer than the summation of the length of the contention interval, the length of an RTS packet and the turn-around time.

SCP-MAC [33] combines preamble sampling with scheduling techniques. It synchronizes the wakeup time of neighboring nodes, which minimizes the length of preamble. It also minimize the periodic wakeup time by LPL. In comparison with other MAC protocols it is very energy efficient, especially at very low duty cycles; however the synchronous mechanism of SCP-MAC has several drawbacks. The first problem is that since all nodes in a neighborhood wake up at the same time, nodes cannot avoid overhearing the packets from and for each of their neighbors. S-MAC and T-MAC prevents overhearing by using RTS/CTS; however an RTS/CTS exchange has relatively high overhead (especially since data packets in WSNs are short). SCP-MAC also provides a mechanism to avoid overhearing by sleeping upon the receipt of a header for a different destination; however this approach only avoids overhearing the payload of the packet and also requires sufficiently low level access to hardware (such as for MicaZ using a bit-streaming radio, Chipcon CC1000 [1]), and cannot be implemented on sensor platforms with a packet level radio (such as Mica2Z and TelosB that use Chipcon CC2420 [2]).

The second problem is that SCP-MAC results in increased contention. At each synchronized wakeup time, every sender in a neighborhood has to contend to acquire the channel. This high contention increases packet loss and degrades the energy efficiency and throughput due to the resulting collisions. It can also result in congestion, which, in turn, deteriorates application-level reliability. Finally, SCP-MAC incurs relatively large delays in multi-hop scenarios. The per-hop delay is at least equal to the wakeup interval. It can be much longer because all losing nodes in the contention have to postpone their transmissions to the next synchronized wakeup time. SCP-MAC provides an adaptive channel polling mechanism that reduces the average delay by adding a high-frequency LPL in the same frame, immediately following its regular LPL, when receiving a packet. It decreases the average delay when bursty traffic occurs (the next packet is then transmitted within the additional LPLs). However, adaptive polling is not effective in reducing delay (and increases energy consumption) when traffic is light. Our proposed scheme, AS-MAC aims to solve all the above-mentioned problems of SCP-MAC while inheriting many of its advantages.

The receiver-initiated MAC protocols ([i.e., RI-MAC [31], PW-MAC [17] and EM-MAC [16]]) uses the beacon message for the reliable data transmission. In RI-MAC, each node announces its wakeup by broadcasting a beacon message whenever it wakes up, and a sender starts to transmit a data packet upon receiving a beacon from its intended receiver. Therefore it guarantees that the intended receiver is awake when the sender transmits the data. When a sender in RI-MAC has a packet to transmit, it immediately wakes up to wait for the receiver, which results in a large sender duty cycle due to its idle listening until the receiver wakes up. PW-MAC uses the receiver-initiated wakeup mechanism using a beacon message as in RI-MAC; however, it minimizes the long idle listening time of senders, unavoid-
able in RI-MAC, by enabling senders to predict receiver wake-up times with pseudo random wakeup scheduling mechanism, which also avoids and efficiently resolves any radio collisions caused by multiple concurrent traffic flows. However, broadcasting and receiving beacon messages always before the data transmission still incurs increasing energy consumption while AS-MAC is more flexible because Hello interval is different from the wakeup interval. EM-MAC utilizes the multiple orthogonal radio channels available in common sensor node devices inheriting all aspects of PW-MAC (i.e., pseudo random wakeup scheduling mechanism). In EM-MAC, every node dynamically selects wireless channels based on the channel conditions it senses. EM-MAC achieves good energy efficiency, latency and packet delivery ratio.

SPARE MAC [6] proposes a distributed scheduling solution that assigns to each sensor its own time slots for reception. It shares the basic principle of the asynchronous wakeup with our proposed protocol, which makes it reduce energy consumption. However, it is a collision free protocol that relies on a dynamic TDMA slotted structure. Z-MAC [21] presents a hybrid MAC protocol that combines the strengths of TDMA and CSMA. Z-MAC achieves a high channel utilization under both low contention and high contention.

Several MAC protocols for IEEE 802.11-based ad hoc networks [26,34] have used an asynchronous wakeup mechanism similar to our proposed mechanism. Their purpose is not to decrease the energy consumption but rather to increase the robustness of network: they trade energy for network reliability. Tseng et al. propose three asynchronous wakeup protocols [26]. The protocols overlap periodic wakeup times of neighboring nodes; therefore, they can increase the reliability of the network, but the wakeup times of nodes are very large, which leads to a large energy consumption. In [34], nodes store the wakeup schedules of neighbors like our proposed approach. The approach in [34] uses the scheduling information to allow the detection of neighbor departures. That is, maintaining neighbor schedules in the approach does not decrease energy consumption.

Wise-MAC [10] uses a mechanism similar to AS-MAC to predict the wake-up time of the intended receiver; however, Wise-MAC was designed for single hop networks relying on an access point with sufficient energy to be always on. In contrast, the proposed AS-MAC is designed to be a general purpose multi-hop MAC protocol.

Zhang and Shin presents E-MiLi [29], an energy-minimizing idle listening mechanism for IEEE 802.11. They show that more than 60% of energy is consumed in idle listening through real world traffic, even with IEEE 802.11 power-saving mode enabled. They show that radio power consumption decreases proportionally to its clock-rate and E-MiLi adaptively down-clocks the radio during idle listening, and reverts to full clock-rate when an incoming packet detected or a packet has to be transmitted. It incorporates sampling rate invariant detection for accurate packet detection and opportunistic down-clocking mechanism for efficient switching clock-rate. Experimental results shows that E-MiLi very accurately detects packets while reducing energy consumption.

3. AS-MAC design

In this section, we describe the proposed Asynchronous Scheduled WSN MAC protocol (AS-MAC). The basic idea in AS-MAC is that nodes wake up periodically (but asynchronously from their neighbors) to receive packets. Nodes intending to transmit wake up at the scheduled wakeup time of the intended target node. We first describe the initialization phase and the periodic listening and sleep phase then introduce the collision detection and retransmission mechanism.

3.1. Initialization phase

When a new node joins a WSN, it performs the initialization phase, in which it builds the neighbor table that stores its neighbors' scheduling information, and chooses and announces its own unique offset of the periodic wake-up. Existing nodes may be in the initialization phase or the periodic listening and sleep phase. Nodes in the periodic listening and sleep phase perform LPL every wakeup interval, \( I_{\text{wakeup}} \), and send Hello packet every Hello interval, \( I_{\text{hello}} \). Hello packets are used to publish scheduling information: \( I_{\text{wakeup}} \), \( I_{\text{hello}} \), and offset of the periodic wakeup, \( O_{W} \).

First, the new node receives the Hello packets from its neighbors, listening to the channel for a fixed amount of time, which should be longer than \( I_{\text{hello}} \) to guarantee that the new node receives the Hello packets from its all neighbors. Each time the node receives a Hello packet from a neighbor, it determines the starting time of the reception of the Hello packet to the neighbor's \( O_{W} \) and stores \( I_{\text{hello}}, I_{\text{wakeup}} \) and \( O_{W} \) with the neighbor ID in its neighbor table.

After building the neighbor table of its neighbors, the new node will set its \( O_{W} \) uniquely. One possible approach is the node sets its \( O_{W} \) to a random value different from its neighbors' \( O_{W} \). However, it is better to distribute the \( O_{W} \)s of neighboring nodes as evenly as possible. To achieve this distribution, one possible approach is that the node sets its \( O_{W} \) at the half point of the longest interval among the neighbors' \( O_{W} \)s. Fig. 1 presents an example of setting the unique \( O_{W} \). Node C is the new participant. The interval from B's \( O_{W} \) to A's \( O_{W} \) is longer than the interval from A's \( O_{W} \) to B's \( O_{W} \). In result, node C sets the half point of the former interval as its \( O_{W} \). If the new node does not receive any Hello packet from neighbors and its neighbor table is empty, it sets its \( O_{W} \) at a random time.

Once the new node sets its unique \( O_{W} \), it enters the periodic listening and sleep phase. To publish its schedule to its neighbors, it sends its scheduling information to its known neighbors (discovered during the initialization phase) at their upcoming \( O_{W} \)s separately.

We assume that \( I_{\text{wakeup}} \) is large enough to asynchronously accommodate \( O_{W} \)s of all nodes in communication range of each other. To prevent that any two neighboring nodes miss each other forever, nodes perform periodic neighbor discovery like existing scheduled MAC protocols [28,32,33]. For neighbor discovery, nodes listen to the channel for a whole \( I_{\text{hello}} \) interval, adds new neighbors' schedule in its neighbor table and informs its scheduling
information to new neighbors by sending Hello packets separately. This allows the discovery of missed or mobile nodes. The frequency of the neighbor discovery depends on the rate of change of the network topology and can be done adaptively. Similarly, to account for link fluctuations, stale entries in its dynamic table should be periodically flushed.

### 3.2. Periodic listening and sleep phase

After the initialization phase, a node enters the periodic listening and sleep phase. Fig. 2 presents the simple finite state machine [19] for the periodic listening and sleep phase. The node starts the periodic listening and sleep phase setting wakeup interval, \( I_{\text{wakeup}} \). A node performs LPL every \( I_{\text{wakeup}} \) timeout to receive an incoming packet. If the channel is busy, the node receives the incoming packet. If the wakeup time of the node is also Hello time, the node receives the packet after sending a Hello packet. When a node has a packet to send, it waits in sleep state until the receiver is scheduled to wake up, and it wakes up when the receiver does. If the wakeup time of the receiver is Hello time, it receives the Hello packet and then sends the packet. If not, it directly sends the packet with the preamble compensating clock drift. We first describe the operation of a node acting as a receiver, and then the operation as a sender.

As a receiver, if the wakeup time is also a Hello time, if the channel is clear, the node broadcasts a Hello packet. After that, the node waits to receive a packet until a timeout, \( t_0 \). If a packet is sent before \( t_0 \), the node receives it (e.g., receiver in Fig. 3). The value of \( t_0 \) should be only slightly longer than the maximum backoff time of the senders. After receiving a packet, the receiver goes back to sleep. When the wakeup time is not a Hello time, if the channel is busy, the node immediately goes back to sleep. If the channel is busy, the node stays in the listen state and receives the incoming packet. After reception, the node returns to sleep (e.g., receiver in Fig. 4).

As a sender, the node does not wake up at neighbors’ wakeup time if it has nothing to send. If a node has a packet to send, it waits in sleep state until the receiver’s wakeup time. Every node stores its neighbors’ scheduling information in its neighbor table; therefore it can predict the remaining time, \( t_{\text{remain}} \), from the current time to the upcoming wakeup time of the receiver:

\[
t_{\text{remain}}(i) = I_{\text{wakeup}}(i) - \left( t_c - O_W(i) \right) / C_i, \quad \text{(1)}
\]

where \( t_c \) is the current time, and \( i \) is the ID of the receiver. When the receiver’s wakeup time is also a Hello time, as shown in Fig. 3, to compensate for the potential clock drift between the sender and the receiver, the sender wakes up earlier than \( t_{\text{remain}} \) by a guard time, \( t_{\text{G}} \), which is equal to or smaller than \( t_{\text{remain}} \) because it cannot be a negative number:

\[
t_{\text{G}}(i) = \min(2C_{\text{drift}}(i) - O_W(i), t_{\text{remain}}(i)). \quad \text{(2)}
\]

where \( C_{\text{drift}} \) is the maximum clock drift rate. The sender waits for the receiver’s Hello packet with a timeout \( t_{\text{C}} \), which is similar to \( I_{\text{wakeup}} \) at most:

\[
t_{\text{C}}(i) = \min(4C_{\text{drift}}(i) - O_W(i) + t_{\text{LPL}}, I_{\text{wakeup}}(i)). \quad \text{(3)}
\]

where \( t_{\text{LPL}} \) is the time for LPL. If the sender does not receive the Hello packet before \( t_{\text{C}} \) seconds elapse, it postpones the transmission to the next wakeup time of the receiver. If the sender does receive the Hello packet from the intended receiver, it updates the receiver’s \( O_W \) in its neighbor table to the start time of the reception of the Hello packet (e.g., Senders I and II in Fig. 3). To avoid the collision with the potential senders, the sender performs collision avoidance backoff and carrier sensing by randomly selecting a slot within the fixed contention window. If it loses the contention, the sender postpones the transmission to the receiver’s next wakeup time (e.g., Sender I in Fig. 3). If it wins the contention, it sends the packet, after that it goes back to sleep (e.g., Sender II in Fig. 3).

When the receiver’s wakeup time is not a Hello time, as shown in Fig. 4, the sender should perform the collision avoidance backoff to avoid collision with other potential senders; therefore the guard time \( t_{\text{C}} \) is longer than \( t_{\text{C}} \) by the maximum contention window time, and it is equal to or smaller than \( t_{\text{remain}} \) like in (2). If it loses the contention, the sender postpones the transmission to the receiver’s next wakeup time. If it wins the contention, the sender sends the data packet with the preamble of \( t_{\text{LPL}} \), which is longer than \( t_{\text{C}} \) by the remaining contention time and similar to \( I_{\text{wakeup}} \) at most like in (3) (e.g., sender in Fig. 4).

---

**Fig. 1.** Initialization phase finding its offset.
Fig. 2. Periodic listening and sleep phase.

Fig. 3. Communication at Hello time.

Fig. 4. Communication at wakeup time.
The disadvantage of the asynchronous wakeup interval in AS-MAC is the inefficiency of broadcast. To broadcast a packet, AS-MAC has to transmit the packet once for each neighbor. Alternatively, it could broadcast a packet with long preamble like B-MAC. In AS-MAC, nodes do not overhear any packets because each receiver has own unique OW. Moreover, only senders trying to send to the same receiver contend to acquire the channel at the receiver’s wakeup time. Therefore, AS-MAC results in fewer collisions than SCP-MAC. In terms of delay, AS-MAC is faster on the average than SCP-MAC by a factor of two. In SCP-MAC, the per-hop delay of a packet is equal to wakeup interval, because the packet can be propagated at only the synchronized wakeup time. In AS-MAC, the per-hop delay of a packet is on the average half the wakeup interval. In AS-MAC, each node stores only its neighbors’ schedules.

3.3. Collision detection and retransmission

For reliable data packet transmission, AS-MAC uses ACKs after data packet receptions and retransmission with exponential backoff up to the maximum retransmission limit. When a sender has a packet to send, it wakes up at the wakeup time of the intended receiver and transmits the data packet. After transmitting a data packet, the sender stays awake for a brief period of time (i.e. processing time and transmission time), $t_{OA}$, to allow it to receive an ACK packet. If the sender does not receive the ACK from the intended receiver in $t_{OA}$, in order to save energy, similar to the approach in PW-MAC [17], the sender goes back to sleep state and retransmits it at the next wakeup time of the receiver. The retransmission is performed with exponential backoff within the maximum retransmission limit, $m$. The receiver wakes up at its periodic wakeup time. If there is an incoming packet and it is valid, the receiver transmits ACK packet and goes back to sleep. If the reception is unsuccessful, the receiver goes back to sleep without transmitting the ACK packet. Fig. 5 illustrates the collision detection and retransmission mechanism of AS-MAC.

An important advantage of AS-MAC is that it can build on existing underlying MAC protocols inheriting their good properties while adding energy efficiency. To avoid the hidden terminal problem, AS-MAC can also incorporate an RTS/CTS [27] mechanism like in [32], but postponing the data transmission with RTS/CTS to the next wake-up time of the intended receiver in loss of RTS packet transmission like in our collision detection and retransmission mechanism. In what follows we assume that AS-MAC builds on a CSMA with collision avoidance and without RTS/CTS and ACKs (primarily for a fair and consistent comparison with existing MAC protocols such as SCP-MAC).

4. Energy consumption analysis

In this section, we present a brief analysis of the performance of AS-MAC and compare it with a very energy-efficient WSN MAC protocol, SCP-MAC. We first provide energy consumption models and then the numerical results based on this model.

4.1. Energy model

We develop a power consumption model for AS-MAC multi-hop networks. We assume that the network is organized as a tree rooted at the sink, and every node periodically generates and sends a data packet to its parent; we also assume that there are no collisions and all wake-up intervals are the same. In SCP-MAC, we consider its basic mechanism without the collision avoidance, the two-phase contention and the adaptive channel polling. We assume that SYNC schedules of all nodes are evenly distributed such that SCP-MAC incurs the smallest overhead for transmitting and receiving preambles. We assume that the offsets of wake-up times of all nodes in a neighborhood are different such that neighboring nodes do not overhear each other.

Table 1 presents the notations, and default values of variables used in our analysis. We use the values of MICAz platform for the variables. (We can easily calculate the energy consumption on other platforms such as MICA2 and TELOS B by using the values for the corresponding platforms.) The total energy consumption per second, $E$ includes transmission, reception, listen, sleep, and LPL, denoted as $E_{tx}, E_{rx}, E_{ls}, E_{s}$, and $E_{lpl}$, respectively:

$$E = E_{tx} + E_{rx} + E_{ls} + E_{lpl} + E_{s}.$$ (4)
Table 1
Notations and default values.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
<th>Value (MICAz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l_{data}$</td>
<td>Data packet length</td>
<td>50 bytes</td>
</tr>
<tr>
<td>$l_{sync}$</td>
<td>SYNC packet length</td>
<td>18 bytes</td>
</tr>
<tr>
<td>$l_{hello}$</td>
<td>Hello packet length</td>
<td>18 bytes</td>
</tr>
<tr>
<td>$P_{tx}$</td>
<td>Power in transmission mode</td>
<td>52.2 mW</td>
</tr>
<tr>
<td>$P_{rx}$</td>
<td>Power in reception mode</td>
<td>56.4 mW</td>
</tr>
<tr>
<td>$P_{psleep}$</td>
<td>Power in listen mode</td>
<td>56.4 mW</td>
</tr>
<tr>
<td>$P_{psleep}$</td>
<td>Power in sleep mode</td>
<td>0.003 mW</td>
</tr>
<tr>
<td>$P_{ppl}$</td>
<td>Power in LPL mode</td>
<td>12.3 mW</td>
</tr>
<tr>
<td>$t_{ppl}$</td>
<td>Time for LPL</td>
<td>2.5 ms</td>
</tr>
<tr>
<td>$t_{r}$</td>
<td>Time to RX a byte</td>
<td>0.032 ms</td>
</tr>
<tr>
<td>$t_{s}$</td>
<td>Time for contention slot</td>
<td>0.04 ms</td>
</tr>
<tr>
<td>$S_{cw}$</td>
<td>Contention window size</td>
<td>16</td>
</tr>
</tbody>
</table>

\[
E = T_{tx}P_{tx} + T_{rx}P_{rx} + T_{tp}P_{psleep} + T_{ppl}P_{ppl} + T_{s}P_{psleep},
\]

where $T_{tx}$, $T_{rx}$, $T_{tp}$, $T_{ppl}$, and $P_{psleep}$ are time fractions and power for transmission, reception, listen, LPL, and sleep, respectively. We assume that a node sleeps when it is not doing anything else:

\[
T_{s} = 1 - T_{tx} - T_{rx} - T_{tp} - T_{ppl}.
\]

For SCP-MAC, we follow the energy analysis in [33] (while trivially adapting it to the multi-hop network scenario we consider). In this section, we determine the values of $T_{tx}, T_{rx}, T_{tp}, T_{ppl}$, and $T_{s}$ for AS-MAC.

In AS-MAC, nodes have the Hello interval, $I_{hello}$, and the wake-up interval, $I_{wakeup}$. We assume that $I_{hello}$ is $m$ times larger than $I_{wakeup}$, i.e., $I_{hello} = ml_{wakeup}$, where $m$ is a positive integer. We assume that when a node sends $m$ data packets, on the average it receives the parent’s Hello packet once.

The expected transmission time is:

\[
T_{tx} = T_{txhello} + T_{txdata} + T_{txpre},
\]

where $T_{txhello}$, $T_{txdata}$, and $T_{txpre}$ are the time fractions for transmitting Hello packets, data packets, and preambles respectively. Every hello interval, a node sends a hello packet:

\[
T_{txhello} = \frac{1}{I_{hello}}l_{hello}t_{B}.
\]

Nodes generate and send a packet every data generation interval, $I_{data}$, and receive and forward packets generated by each of their descendants.

\[
T_{txdata} = \frac{n_{descendants} + 1}{I_{data}}l_{data}t_{B},
\]

where $n_{descendants}$ is the number of descendants of a node.

When a node needs to send data packets, it undergoes $(m - 1)$ wake-up times without Hello packet of its parent for $m$ transmissions; i.e., it sends $(m - 1)$ preambles for $m$ transmissions:

\[
T_{txpre} = \frac{m - 1}{m} \frac{n_{descendants} + 1}{I_{data}}l_{pretx}.
\]

The fraction of the time spend receiving is:

\[
T_{rx} = T_{rxhello} + T_{rxdata} + T_{rxpre}.
\]

When a node needs to send data packets, it receives a Hello packet for each $m$ data transmissions:

\[
T_{txhello} = \frac{1}{m} \frac{n_{descendants} + 1}{I_{data}}l_{hello}t_{B}.
\]

Each node receives a data packet generated by every child every data generation interval. The fraction of time spent receiving data packets is:

\[
T_{txdata} = \frac{n_{descendants} + 1}{I_{data}}l_{data}t_{B}.
\]

When a node receives data packets, it receives $(m - 1)$ preambles for $m$ receptions because its adjacent child undergoes $(m - 1)$ wake-up times without Hello packet every $m$ transmissions:

\[
T_{rxpre} = 2ml_{ck}t_{B}.
\]

The expected listening time is:

\[
T_{rx} = T_{tx} + T_{lcw} + T_{lw}.
\]

where $T_{lw}$, $T_{lcw}$ and $T_{lw}$ are the percentage listening time while compensating for clock drift, while in contention and waiting for the time-out respectively.

To compensate for the clock drift, nodes listen to the channel during:

\[
T_{lw} = \frac{1}{S_{cw}t_{slot}}.
\]

where $S_{cw}$ is the contention window size.

When nodes wake up, they listen to the channel on average for half of $t_{O}$ after sending Hello packet to receive an incoming packet:

\[
T_{lw} = \frac{1}{2I_{hello}}t_{O}.
\]

where the timeout $t_{O}$ is longer than the maximum contention time by one slot time:

\[
t_{O} = (S_{mcw} + 1)t_{slot}.
\]

When a node wakes up, they perform LPL:

\[
T_{ppl} = \frac{1}{I_{wakeup}}t_{ppl}.
\]

To calculate the optimal hello interval that minimizes the total energy of AS-MAC, we substitute $ml_{wakeup}$ for $I_{hello}$ of each equation and differentiate the total energy with respect to $m$:
by solving (23), we can calculate \( \text{Opt}_m \) (rounded as it is an integer). Thus, the optimal \( I_{\text{hello}} \) is:

\[
\text{Opt}_{\text{hello}} = \text{Opt}_m I_{\text{wakeup}},
\]

where

\[
\text{Opt}_m = \sqrt{\frac{A_m + B_m + C_m + D_m + E_m + F_m}{G_m + H_m}},
\]

\[
A_m = \frac{1}{I_{\text{wakeup}}} L_{\text{hello}} t_0 (P_{tx} - P_s),
\]

\[
B_m = \frac{n_{\text{descendants}} + 1}{I_{\text{data}}} L_{\text{hello}} t_0 (P_{rx} - P_s),
\]

\[
C_m = \frac{n_{\text{descendants}} + 1}{I_{\text{data}}} t_\text{data} (P_{tx} - P_s),
\]

\[
D_m = \frac{n_{\text{descendants}} + 1}{I_{\text{data}}} S_{\text{encvtw}} t_\text{holo} (P_{tx} - P_s),
\]

\[
E_m = \frac{1}{2I_{\text{wakeup}}} t_0 (P_{tx} - P_s),
\]

\[
F_m = \frac{n_{\text{descendants}} + 1}{I_{\text{data}}} t_\text{data} (P_{s} - P_{tx}),
\]

\[
G_m = 4r_{\text{clk}} (P_{tx} - P_s),
\]

\[
H_m = 2r_{\text{clk}} (P_{tx} - P_s).
\]

In AS-MAC, there is an optimal \( I_{\text{hello}} \) with respect to \( I_{\text{wakeup}} \), and the energy consumption monotonically decreases as \( I_{\text{wakeup}} \) increases.

4.2. Numerical results

In this section we present numerical results based on the energy models in Section 4.1 in a scenario with one hundred nodes in the 10 by 10 uniform rectangular grid with a sink in a corner, as shown in Fig. 6. We assume that all nodes have the same wake-up interval and nodes only communicate with their immediate neighbors and forward data based on the fixed routing presented in Fig. 6. We consider the energy consumption of two nodes. Node I receives and forwards data for six nodes and hears 126 transmissions in each sampling period; Node II receives and forwards data for 63 nodes and hears 143 packets during each sampling period. Because the choice of the wake-up interval effectively allows a trade-off between energy and delay, for fairness, we compare AS-MAC and SCP-MAC at similar per-hop delays (assuming that the per-hop delay for SCP-MAC is equal to the wake-up interval and for AS-MAC is half of the wake-up interval). We apply the optimal values calculated by our energy model for Hello interval and SYNC interval. The data generation interval is fixed at 100 s.

Fig. 7 shows the energy consumption of Node I as a function of per-hop delay. AS-MAC clearly outperforms SCP-MAC at a similar delay penalty. For the per-hop delay equal to 1.2 s, AS-MAC has a energy consumption of only 17.3% of that of SCP-MAC. Fig. 8 depicts the sources of energy consumptions of each protocol for Node I. It is readily apparent that the main advantage of AS-MAC results from the elimination of overhearing, resulting in a considerable reduction of the energy spent in receiving mode. At higher wake-up intervals, AS-MAC can reduce most of the sources of power consumption. In contrast, the savings of SCP-MAC are negatively affected by the consumption due to overhearing that remains unchanged (and continues to dominate the overall energy budget).

Figs. 9 and 10 show the energy consumptions and the sources of energy consumptions of Node II. Because Node II forwards far more packets than Node I, the benefit of AS-MAC in terms of energy consumption is smaller in Node II than in Node I. Qualitatively, the results for Node II are similar to the ones for Node I, with AS-MAC obviously outperforming SCP-MAC at all delays.

In summary, we can expect that at a similar latency, AS-MAC outperforms SCP-MAC primarily due to the overhearing avoidance. The higher the network size and density, the larger the performance gap between AS-MAC and SCP-MAC.

5. Experimental evaluation

To evaluate the design of AS-MAC, we implement AS-MAC in TinyOS [12] on two major platforms: TELOS-B [4]
and MICAz [3] based on the Chipcon CC2420 [2] radio. We compare the performance of AS-MAC with two important WSN MAC protocols, CSMA and SCP-MAC to show the effect of elemental algorithms of AS-MAC, the duty cycling and asynchronous scheduling mechanisms in turn. Before the protocol can be used in large scale deployments, especially since it uses long term scheduling, a large scale experiment is warranted. In particular, the stability of the schedules as nodes drift is not validated by the current experiments (involving a small number of nodes and relatively short duration). While we believe that the proposed protocol is robust in the proposed design, this experimental section does not confirm this robustness. The main goal of the experimental evaluation is to validate the analysis in Section 4.

5.1. AS-MAC VS CSMA

We evaluate the performance of AS-MAC in terms of three traditional evaluation metrics: energy consumption, throughput and delay. We use two parameters, wakeup interval and data generation interval, correspondingly duty cycle and traffic load respectively.

The experimental result show that the duty cycling mechanism of AS-MAC that significantly decreases the energy consumption at the cost of a reduced maximum throughput and increased delay in comparison to plain CSMA.

5.1.1. Energy consumption

To measure the energy consumption, we monitor changes in the state of the radio. We used counters that accumulate time in each state of the radio (e.g., transmit, receive, listen, sleep and wakeup). At the end of the experiment, considering the energy consumption in each state, we compute the total energy consumption.

We consider only the energy consumption on the periodic listen and sleep phase of AS-MAC. The energy consumption in the initialization phase of AS-MAC can be ignored because the lifetime of WSN is assumed to be very long, tens or hundreds of days, and the initialization phase is relatively very short.

We use a chain with four nodes with the fourth node as the sink at one end of the chain. For the multi-hop network, we reduce the RF output power of the node to its minimum value and place the nodes about twenty centimeters apart; this results in good communication links between neighboring nodes and intermittent connectivity between nodes two hops apart. In this experiment the first three nodes periodically send packets to their parent (we used static routing), which in turn forwards it toward the sink. Each experiment lasted for 300 s. We consider the energy consumption as a function of the wakeup interval and the data generation interval.

Fig. 11 presents the energy consumption of the second node as a function of the wakeup interval in TELOSb platform. We fix the data generation interval to 10 s. Not surprisingly, the energy consumption of CSMA is constant because energy consumptions of transmission, reception and listen modes are same. The energy consumption of
AS-MAC decreases with the increase in the wakeup interval because the sleep time increases and energy consumption in sleep mode is much smaller than in any other mode.

With a wakeup interval of 100 s, the energy consumption of AS-MAC is only 0.37% of that of CSMA.

Fig. 12 presents the energy consumption of the second node as a function of the data generation interval in TELOS-B platform. We fix the wakeup interval to 10 s. The energy consumption of CSMA is again constant while the energy consumption of AS-MAC decreases as the data generation interval increases. At the data generation interval of 100 s (i.e., one packet every 100 s) the energy consumption of AS-MAC is approximately 0.51% of that of CSMA.

5.1.2. Throughput

We measure the throughput varying data generation rate on a single-hop network with one sender and one receiver. We measured the throughput as:

$$T = \frac{N_{rp}S_p}{T_e},$$

where $T$ is the throughput, $T_e$ is the duration of the experiment, $N_{rp}$ is the number of received packets, and $S_p$ is the packet size. In our experiment, the packet size $S_p$ is 38 bytes, and the duration $T_e$ is 100 s. The wakeup interval of the sender and receiver is fixed to one second.

Fig. 13 shows the throughput as a function of the data generation rate in TELOS-B platform. The maximum throughput of AS-MAC is about 70% of that of CSMA. The root cause of this reduction is that sometimes, the receiver goes back to sleep due to the duty cycling mechanism although the transmitter still has packets in its queue. While the receiver is sleeping, the transmitter’s queue overflows, thus resulting in packet loss.

5.1.3. Delay

Duty cycling MAC protocols trade off delay for energy saving. To quantify the delay of AS-MAC, we use the same chain of four nodes (with the sink being the fourth node and data being generated at the first node). We vary the wakeup interval and data generation rate from 1 s to 40 s.

Fig. 14 shows well the average per-hop delay of the three hops in TELOS-B platform (we measure the end-to-end delay and divide it by the hop count – three). As expected, the delay of AS-MAC is very close to half of the wakeup interval (the variations are due to the random choices of the offsets of the wakeup times during the initialization phase). The per-hop delay of the default CSMA protocol is about 12 ms.

5.1.4. Memory footprint

The main overhead of AS-MAC in terms of memory usage is for maintaining the neighbor table. In our implementation of AS-MAC in the TELOS-B platform, the neighbor table consumes 12 bytes per neighbor, specifically, 2 bytes for neighbor ID, 4 bytes for wakeup interval, 4 bytes for the offset of wakeup interval and 2 bytes for the Hello interval.
The Hello interval is stored as a multiple of the wakeup interval.

Table 2 provides RAM and ROM sizes of AS-MAC and CSMA in TELOSB platform, in which AS-MAC has the neighbor table with five neighbors. RAM and ROM footprints of AS-MAC are larger than those of CSMA because AS-MAC was implemented on top of CSMA. RAM and ROM sizes of AS-MAC without the neighbor table are 724 bytes and 11,238 bytes respectively.

<table>
<thead>
<tr>
<th></th>
<th>RAM (bytes)</th>
<th>ROM (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TELOSB</td>
<td>16,000</td>
<td>48,000</td>
</tr>
<tr>
<td>CSMA</td>
<td>546</td>
<td>9268</td>
</tr>
<tr>
<td>AS-MAC</td>
<td>784</td>
<td>11,906</td>
</tr>
</tbody>
</table>

5.2. AS-MAC VS SCP-MAC

We consider energy consumption, packet loss and delay as evaluation metrics. We evaluate AS-MAC and SCP-MAC by varying three parameters: number of neighbors, data generation interval and wakeup interval, or, in other words, network density, traffic load and duty cycle respectively.

The experimental result shows well the effect of the asynchronous mechanism of AS-MAC that decreases the energy consumption, packet loss and delay at the penalty of the memory consumption in comparison with a synchronous scheduled energy-efficient MAC protocol.

5.2.1. Energy consumption

We use a star topology consisting of one receiver and up to five senders, in which all nodes are in communication range with all others. Each sender transmits a packet to the receiver every 10 s 20 times. The wakeup interval is set to one second in both MAC protocols, and the Hello interval of AS-MAC and SYNC interval of SCP-MAC is 60 s. The contention window size is 16 for both MAC protocols. We consider the energy consumption as a function of the number of senders. Ideally, the energy consumption of MAC protocol should not increase with the increase in the number of neighbors. To validate our energy model in Section 4.1, we also show the energy estimations from the model. Before starting the data generation we use a random jitter to avoid periodic collisions.

Fig. 15 presents the average energy consumption of the senders as a function of the number of senders in MICAz platform, and Figs. 16 and 17 show the sources of energy consumptions with two and five senders respectively in MICAz platform. The energy consumption of AS-MAC is almost constant regardless of the number of senders, while the energy consumption of SCP-MAC increases with number of senders. Figs. 16 and 17 show that the difference is due to the fact that AS-MAC can avoid overhearing while SCP-MAC cannot. The energy consumption in receive mode...
for AS-MAC is almost similar in Figs. 16 and 17; however, that of SCP-MAC in Fig. 17 is much larger than that of SCP-MAC in Fig. 16. There are significant differences in the absolute values between the theoretic and the experimental results although qualitatively the models capture the behavior of the two MAC protocols. The root cause of the difference between the experimental result and the theoretical analysis is a discrepancy between the theoretical and real listen time of the protocol. In theory, we assume that the radio can perform LPL perfectly, but in reality, the radio sometimes recognizes a busy channel even when the channel is idle (this may be due to noise in the 2.4 GHz ISM band). In this case, both AS-MAC and SCP-MAC will wait for the packet that does not exist. To prevent this error, both AS-MAC and SCP-MAC set up a timer. This additional listening time is not taken into account in the theoretical analysis of either protocols. Furthermore, we also neglected processing time in the theoretical analysis. We can expect that in a deployment scenario with less noise the energy consumption will closely match the results of the analysis.

5.2.2. Packet loss
To evaluate the packet loss, we set up a chain network with six nodes, with sixth node as the sink at one end of the network. Each experiment lasted until each node generated and sent 10 packets (not counting the forwarded packets). The contention window size is four for both AS-MAC and SCP-MAC. Neither the SCP-MAC, nor the AS-MAC use buffers for retransmissions or queue the packets. We calculate the packet loss at the sink.

Fig. 18 shows the packet loss in the sink in MICAz platform. The packet loss of SCP-MAC is much larger than that of AS-MAC. The major cause of the large packet loss in SCP-MAC is the increased contention (as all nodes wake up at the same time to send packets) – the lack of buffers may amplify this effect.

5.2.3. Delay
To quantify the delay, we use the same chain network of six nodes (with the sink being the sixth node and data being generated at the first node). We vary the wakeup interval and data generation interval from 1 s to 8 s.

Fig. 19 shows the average per-hop delay of the five hops in MICAz platform. As in the experiment in TELOSB platform, the per-hop delay of AS-MAC is very close to half of the wakeup interval. On the other hand, that of SCP-MAC is close to (just a little longer than) the wakeup interval, i.e., double that of AS-MAC. The adaptive channel polling of SCP-MAC cannot reduce the delay in this light traffic scenario.
5.2.4. Memory footprint

Table 3 provides RAM and ROM sizes of AS-MAC and SCP-MAC in MICAz platform, in which AS-MAC has the neighbor table with five neighbors and its RAM and ROM footprints are slightly larger than those of SCP-MAC. RAM and ROM sizes of AS-MAC without the neighbor table are 889 bytes and 16,004 bytes respectively, which are very similar to those of SCP-MAC.

6. Conclusion

We present an duty cycling, asynchronous, scheduled, energy-efficient MAC protocol; in the proposed approach, each node stores the wakeup schedules of their neighbors. The protocol asynchronously coordinates the wakeup times of neighboring nodes to reduce overhearing, contention and delay being unavoidable in synchronous scheduled MAC protocols. The disadvantages of the proposed protocol are the overhead for broadcast and the need to store one hop neighbor table. We present a multi-hop energy consumption model for the proposed protocol and compare its performance with SCP-MAC. To validate the design and the energy model, we implement AS-MAC in two major TinyOS platforms. The experimental results show that AS-MAC considerably reduces energy consumption while providing good delay and packet loss in comparison with existing WSN MAC protocols and that the energy model is a fairly good approximation of the real energy consumption of the proposed MAC layer.

References

Jun Bum Lim received the Ph.D. degree in Computer Science from North Carolina State University, in 2011. His research interest is in performance evaluation of multihop wireless networks, focusing on developing performance evaluation frameworks and designing protocols in MAC and Routing layers for Wireless Sensor Networks and Wireless Mesh Networks. Since August 2011, he has been a senior software engineer at Qualcomm Inc.

Mihail L. Sichitiu received his B.E. and an M.S. in Electrical Engineering from the Polytechnic University of Bucharest in 1995 and 1996 respectively. In May 2001, he received a Ph.D. degree in Electrical Engineering from the University of Notre Dame. He is currently employed as an associate professor in the Department of Electrical and Computer Engineering at North Carolina State University. His primary research interest is in Wireless Networking with emphasis on ad hoc networking and wireless local area networks.