TAS-MAC: A Traffic-Adaptive Synchronous MAC Protocol for Wireless Sensor Networks

Pei Huang, Chin-Jung Liu, Li Xiao
Department of Computer Science and Engineering
Michigan State University
Email: {huangpe3, liuchinj, lxiao}@cse.msu.edu

Abstract—**Duty cycling improves energy efficiency but limits throughput and introduces significant end-to-end delay in wireless sensor networks. In this paper, we present a traffic-adaptive synchronous MAC protocol (TAS-MAC), which is a high throughput low delay MAC protocol tailored for low power consumption. It achieves high throughput by using Time Division Multiple Access (TDMA) with a novel traffic-adaptive allocation mechanism that assigns time slots only to nodes located on active routes. TAS-MAC reduces the end-to-end delay by notifying all nodes on active routes of incoming traffic in advance. These nodes will claim time slots for data transmission and forward a packet through multiple hops in a cycle. The desirable traffic-adaptive feature is achieved by decomposing traffic notification and data transmission scheduling into two phases, specializing their duties and improving their efficiency respectively. Simulation results and tests on TelosB motes demonstrate that the two-phase design significantly improves the throughput of current synchronous MAC protocols and achieves the similar low delay of slot stealing assisted TDMA with much lower power consumption.**

I. Introduction

Traffic in a wireless sensor network (WSN) can be quite dynamic. Considering typical surveillance applications [1] [2], the network generates very little or no data traffic when no event of interest is detected. However, once an event of interest is detected, large bursts of data packets may be generated. Because of high sampling rate, high throughput support of the MAC layer is desired, and low end-to-end delay is required for detecting back-to-back events [2]. Therefore, it is desirable to have a MAC protocol that can adapt to the swift switch between the two extreme conditions.

In WSN MAC protocol designs, duty cycling is widely adopted to trade throughput and delay for energy efficiency. In some applications such as event detection, high throughput and low delay are also critical because the detected event might be an alarm that should be received by the sink as soon as possible and sufficient data need to be collected to analyze the event. To support high throughput, Time Division Multiple Access (TDMA) is demonstrated to outperform Carrier Sense Multiple Access (CSMA) when there are multiple contending senders [3]. When few nodes have data to send, the channel utilization of TDMA is low because each node can transmit only in its assigned time slots. One solution is to adopt the slot stealing technique [3] [4], where nodes contend for sending in a slot if the slot is abandoned by the owner. To ensure that the owner of a slot indeed abandoned the slot, contending nodes have to listen to the channel for a sufficient amount of time, and then they have to back off randomly to reduce collisions. The utilization of a slot is reduced. In addition, each node has to wake up in every slot to check whether there are data for it or whether it can send. The additional idle listening and overhearing sacrifice energy efficiency. The slot stealing method thus deviates away from the goal of low power operation in WSNs.

This paper introduces a novel traffic-adaptive synchronous MAC (TAS-MAC) protocol that assigns time slots only to nodes that are located on active routes, addressing the underutilization problem in TDMA from a new perspective. The novelty of TAS-MAC is in separating the traffic notification and the data transmission scheduling. Each process is tailored according to its special duties, which greatly improves the efficiency of the proposed protocol. More specifically, the traffic notification packets are only responsible for notifying nodes on active routes of incoming data packets. The size of a traffic notification packet is minimized and a traffic notification packet is transmitted in a "pulse" mode to achieve fast traffic notification. In data transmission scheduling phase, a concise representation of channel access schedule is proposed along with a low overhead schedule exchange mechanism. The schedule exchange ensures that time slots are distributed only among nodes that are on active routes and thus the channel utilization is improved in a different way from the slot stealing, providing similar high throughput and low delay with much lower power consumption. Another superiority of TAS-MAC is that nodes claim time slots according to their traffic loads. By allocating time slots abandoned by nodes of light traffic load to nodes of heavy traffic load, the channel utilization is further improved.

In summary, our contribution is the development of a novel Traffic Adaptive Synchronous MAC (TAS-MAC) protocol, which has the following unique features:

- Traffic notification and data transmission scheduling are separated and refined independently according to their functionalities, improving the medium access scheduling efficiency.
- A new time slot assignment method is introduced, which assigns time slots only to nodes that will participate in data delivery, reducing overhearing and idle listening.
- An efficient schedule exchange mechanism is developed to adapt the time slot assignment to nodes' traffic loads, enhancing end-to-end delay and channel utilization.

The category of synchronous MAC protocols is often criticized for the existence of time synchronization overhead. However, in many sensor network applications the

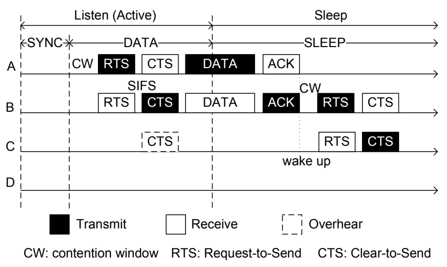

Fig. 1: The adaptive listening in S-MAC.

time synchronization is necessary to provide spatial-temporal correlation between reports provided by multiple sensors [1] [2]. Many events in surveillance and target tracking can be understood only on an accurate time sequence. Therefore, using a synchronous MAC protocol will not add additional overhead for these applications.

II. Related Work

We present the most representative synchronous MAC protocols in this section. Several related TDMA protocols are also discussed. A comprehensive study is available in [5].

S-MAC [6] represents the typical synchronous duty cycling sensor MAC protocols. Time is divided into repeated cycles and each is further divided into three periods: SYNC, DATA, and SLEEP. All nodes wake up at the beginning of the SYNC period to synchronize clocks with each other and then nodes with packets to send contend for exchange of Request-to-Send (RTS) and Clear-to-Send (CTS) frames in the DATA period. Nodes that are not involved in communication return to sleep at the start of the SLEEP period; other nodes return to sleep after they finish transmission of data packets and acknowledgement (ACK) frames. As a result, a packet can be forwarded through one hop per cycle. Later, adaptive listening is introduced to overcome this deficiency.

S-MAC with adaptive listening [7] is illustrated in Fig. 1. Nodes can hear only from their immediate neighbors (e.g., node B can hear only from node A and node C). Because node C can overhear the CTS sent by node B, it goes back to sleep at the beginning of the SLEEP period but wakes up at the end of the current transmission. Node B thus can immediately forward the data packet to node C instead of waiting for the next cycle. As a result, S-MAC with adaptive listening can relay a packet through at most 2 hops per cycle but generally cannot go beyond this because the next hop (i.e., node D) will not wake up since it cannot overhear the CTS sent by node B and does not know when the current transmission will end.

A future request-to-send (FRTS) mechanism introduced in T-MAC [8] can extend the delivery to up to 3 hops per cycle by sending a FRTS frame to the next hop upon overhearing a CTS. Inspired by the FRTS mechanism, RMAC [9] presents a novel approach to reduce latency in multi-hop forwarding. A control frame, called PION (pioneer frame), is forwarded multiple hops to inform nodes on a routing path of when they should wake up to receive possible data packets in the SLEEP period. When a node receives a PION, it finds its position on the path. If it is the kth hop, it will wake up at the kth slot in the SLEEP period. The drawback of RMAC is that two hidden terminals may always cause collisions of data transmission because they use the same slot. DW-MAC [10] introduces a one-to-one mapping function to this design to ensure collision-free data transmission in the SLEEP period. The name of PION is changed to SCH (Scheduling Frame). Because one SCH is mapped to a slot in the SLEEP period, if a SCH is collided, no data will be transmitted in the corresponding slot in the SLEEP period. This ensures collision-free data transmission but wastes a lot of time slots. In addition, if a SCH is not confirmed, the sender will not wake up in the corresponding slot for sending, but the receiver assumes that the packet will arrive and it reserves a time slot for sending the packet. Any failure of a sequential reservation wastes a series of time slots and it significantly limits the number of packets that can be delivered in a cycle.

A common problem of prior work is that the traffic notification and the data transmission scheduling are coupled. The control packets used for traffic notification are also responsible for scheduling the time to transmit data packets (e.g., RTS/CTS in S-MAC [6], SCH in DW-MAC [10]). The dual function of control packets limits the scheduling performance and increases the control overhead as the scheduling is packet-based. Therefore, we separate the two functions and perform a node-based data transmission scheduling for the sleep period.

In the TDMA topic, the proposed TAS-MAC improves the channel utilization from another perspective compared with slot stealing [3] [4]. As a representative example, Z-MAC [3] adopts DRAND [11] for time slot assignment and then allows non-owners of a time slot to contend for the slot with lower access probabilities than that of the owner. Because in each slot a node has to wake up to check whether it can send or whether it is the intended receiver, the method introduces nontrivial additional power consumption.

TRAMA [12] switches between random access period and scheduled access period. The scheduled access period is divided into slots for data transmission. Nodes claim the ownership of each slot in the random access period in a contention way. Because downstream nodes do not know they will have data to send, they will not reserve time slots for data transmission until they get data. TRAMA adopts the time slot assignment algorithm proposed in NAMA [13], which yields low channel utilization because of the sequential priority order problem that we will discuss later. To improve the channel utilization, TRAMA lets a node reassign a time slot to one of its neighbors if the node does not need the slot. Same as slot stealing, the operation requires nodes to wake up to check whether they can send or whether they are the intended receiver, introducing additional overhead and the reassignment can no longer guarantee collision-free data transmission.

III. Overview

In this section we provide an overview of TAS-MAC, which includes two important phases: traffic notification and data transmission scheduling. We target at static WSNs that are deployed for applications with bursty traffic (e.g., fire detection, intrusion detection, target tracking, and etc.). Nodes act as sources and generate reports to the sink if they detect an event of interest. Because the WSN under study has limited dynamics, an ETX (Expected Transmission Count) [14] tree

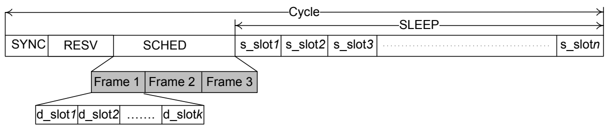

Fig. 2: Time division in TAS-MAC.

is constructed by reusing the periodic time synchronization packets. Same as in Collection Tree Protocol (CTP) [15], the ETX is updated according to the actual data transmission performance. A node chooses the neighbor with the minimum ETX as its next hop during the traffic notification phase. Because the ETX reflects the average link quality, the next hop rarely changes. The parent of a node is thus assumed to be fixed in a cycle similar to other synchronous MAC protocols [8] [9] [10] [12]. Although the predetermined parents may not be the optimal forwarders in the entire cycle, they help relay packets through multiple hops in a cycle. The cycle length is adjustable to meet a desired routing update frequency so that the best relaying point is updated in every cycle.

TAS-MAC repeats time cycles that are divided into four periods: SYNC, RESV, SCHED, and SLEEP as illustrated in Fig. 2. The RESV and SCHED periods are corresponding to the DATA period in traditional synchronous MAC protocols [6] [8] where nodes are all active. However, we do not try to deliver data in the common active period. Instead, we complete the traffic notification and the data transmission scheduling in the common active period. The data transmission is deferred to the SLEEP period so that more data can be delivered in a cycle without keeping all nodes awake. Because both the SCHED period and the SLEEP period are time slotted, we use "*d_slot*" to represent time slots in the SCHED period and "*s_slot*" to represent time slots in the SLEEP period. Each d_slot and s_slot is long enough to transmit a single data packet of the maximum size and receive the ACK frame with sufficient guard intervals.

In TAS-MAC, nodes wake up to synchronize clocks with each other at the beginning of the SYNC period. In the RESV period, source nodes inform other nodes on the routing paths of incoming data packets rapidly. In the SCHED period, nodes that will participate in the data delivery agree on a channel access schedule that specifies their time slots for data transmission in the SLEEP period. Finally, in the SLEEP period, a node wakes up in a particular s_slot if it has data to send and it is the owner of the s_slot, or it is a receiver in the s_slot. Nodes follow TAS-MAC's procedures in the subsequent order.

- Source nodes inform intermediate nodes of incoming traffic (Traffic Notification).
- Nodes that are on active routes construct their transmission schedules (Time Slot Assignment).
- Nodes exchange schedules to improve channel utilization (Schedule Exchange Mechanism).

IV. Traffic Notification

When a node detects an event of interest, it needs to inform nodes on the routing path of incoming data packets so that they will claim time slots for data transmission in the SLEEP period. This ensures that a packet can be delivered through multiple hops in a cycle. When the RESV period starts, a source node initiates a notification packet, called NOTI that includes a *Src* (Source ID) field, a *Con* (Confirmation ID) field, and a *Nxh* (Next Hop) field as shown in Fig. 3. Multiple nodes may detect the same event and they send the NOTI packets in a contention-based way. The *Src* field lets the next hop know which neighbor will send data to it and it can construct a list of children according to the *Src* field.

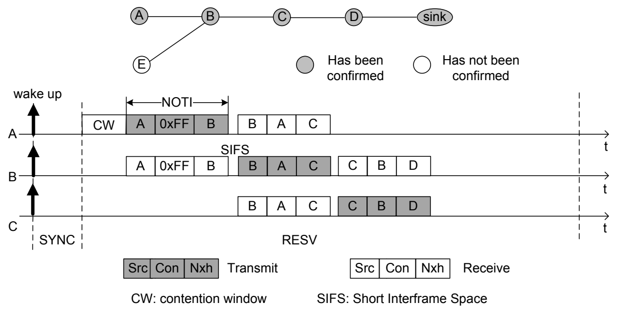

Fig. 3: A simple chain topology.

The *Nxh* field gives a NOTI packet dual functions. When a node receives a NOTI, it responds a NOTI immediately. This NOTI confirms the received request NOTI, and also serves as a request NOTI to the next hop. For example, in Fig. 3, when node B responds a confirmation NOTI to node A, it also uses the NOTI as a request NOTI to node C. It sets *Con* as A, *Src* as B, and *Nxh* as C. When node A receives the confirmation NOTI, the notification is finished for it. When node C receives the NOTI, it responds a confirmation NOTI to node B because it finds that the *Nxh* is equal to itself. In the confirmation NOTI, node C sets the *Nxh* to its own next hop (i.e., node D). In this way, a NOTI packet is forwarded sequentially in a pulse mode, quickly notifying nodes on the routing path of an incoming data flow.

Suppose node A does not receive a confirmation NOTI before NOTI times out. It retransmits the request NOTI after random backoff. If node B has already informed its next hop (i.e., node C) of the incoming data flow, it responds a confirmation NOTI with *Nxh* set to *0xFF* so that the NOTI serves as a confirmation NOTI only. The notification works in the same way when node B receives a new data flow notification. If its request NOTI has been confirmed by its next hop, it only responds a confirmation NOTI instead of a dual function NOTI. In other words, no more request NOTI packets will be generated. Therefore, although NOTI packets are sent in a contention manner, the congestion is not severe. Traffic in WSNs usually converges at the sink and several flows share the same path. Only one NOTI packet needs to be delivered to the sink and others will be terminated when they arrive at an intermediate node that has been confirmed by the next hop (e.g., node E in Fig. 3 will not trigger another flow of NOTI).

It is impossible to set a RESV period that is just long enough to inform all nodes on active routes of incoming data packets. There is a tradeoff between initial delay and energy efficiency in duty cycling MAC protocols. Because the SYNC and SCHED durations are fixed, the RESV period is constrained by the target duty cycle. Although a NOTI packet is not guaranteed to be delivered to the sink in one cycle in a large network, the initial delay in synchronous MAC protocols has been reduced by a significant amount with a pulse mode delivery of tiny control packets. If a NOTI packet cannot be delivered to the sink in one cycle, the intermediate nodes that have not been confirmed will initiate the NOTI packets in the next cycle. After several cycles, all nodes on active routes will be notified, and all intermediate nodes are ready for delivering data packets through multiple hops in a cycle.

V. Data Transmission Scheduling

After traffic notification, a node knows whether it needs time slots for data transmission in the SLEEP period. To determine the ownership of a time slot without incurring high control overhead, we exploit the deterministic characteristic of pseudo-random functions. Before we introduce how nodes work in the duty cycling mode, we briefly describe the entire network operation first.

A. Initial Setup Phase

The initial setup phase is consistent with the TDMA framework. When the network is deployed, nodes send HELLO packets to discover neighbors. Each node is aware of neighbors within its two-hop communication neighborhood. After neighbor discovery, nodes are synchronized through protocols such as FTSP [16] and PulseSync [17]. Once the network is synchronized, a time frame, in which each node is assigned a time slot to transmit without incurring any collision in its two-hop communication neighborhood, is constructed through DRAND [11]. The network is then ready to operate in the TDMA mode by repeating the time frame (e.g., Z-MAC [3]). TDMA, however, is not suited to WSNs that produce few active flows of bursty traffic. TAS-MAC thus adopts the time slotted structure, but assigns time slots only to nodes that are on active routes. The time frame defined by DRAND is only used for exchanging schedules in the SCHED period.

The DRAND is performed only once in the beginning. A node may occasionally hear a packet from a new node due to temporal channel variations. It refuses to consider the new node as a new neighbor because the light interference can be addressed by the capture effect of modern radios. The neighbor list of a node is thus relatively static. If a new node indeed became a neighbor of a node, the node can trigger the local slot assignment of DRAND to change the slot assignment locally. Since nodes are stationary, the DRAND rarely incurs additional overhead after the initial setup phase. Once the initialization is done, nodes switch between active and sleep modes periodically. TAS-MAC mandates nodes to finish time slot assignment for the SLEEP period in a short common active period. The efficient time slot assignment relies on the deterministic characteristic of pseudo-random functions.

B. Time Slot Assignment

A pseudo-random function generates the same sequence of pseudo-random numbers for the same seed. The priority p_k^i of node k for the ith s_slot in the set $N_2(u) \cup u$ can be calculated as:

$$p_k^i = rand(k \oplus i) \oplus k, k \in N_2(u) \cup u \qquad (1)$$

where $rand(x)$ is a pseudo-random number generator that produces uniformly distributed random numbers with the seed x, the sign $\oplus$ means concatenating its operands, and $N_2(u)$ denotes the set of node u's neighbors in its 2-hop communication neighborhood. Node u wins the ith s_slot if Equation 2 is met.

$$\forall v \in N_2(u), p_u^i > p_v^i \qquad (2)$$

This ensures that no two nodes within a two-hop communication neighborhood are assigned to the same slot. Although $rand(x)$ may generate the same number on different inputs of x, each priority p_k^i is unique because the result of $rand(k \oplus i)$ is concatenated with the unique node ID k. Therefore, each node can determine which node in its 2-hop communication neighborhood is the winner of a particular time slot without exchanging any information if it knows the IDs of its two-hop neighbors. Here the node ID is used to break ties between nodes' priorities. This design does not lead to preference towards nodes of greater IDs because an unsigned integer wraps around on an overflow. If a 16 bits representation is used, the results of p_k^i are still uniformly distributed in the range of $[0, 65535]$ if $rand(x)$ is a uniform random number generator. The low overhead of the method enables the repetition of the time slot assignment in each cycle to accommodate dynamic traffic of WSNs. However, there are several issues need to be addressed. First, nodes win a slot with unequal chances because nodes that have more contending neighbors are harder to win a slot. This results in bottlenecks on routing paths. Second, the channel utilization is low due to nodes' local view of the priority distribution. Third, a s_slot is wasted if the winner is not on any active route or has already obtained enough s_slots. We address these challenges in the following discussions.

1) Fairness: Although $rand(x)$ generates uniformly distributed random numbers, nodes do not get fair chances to win a slot. A simple chain topology illustrated in Fig. 3 shows that node C contends with four neighbors while nodes A and E contend with two neighbors. The probability of winning a s_slot for node C is less than that of nodes A and E if they use a uniform random number generator. Packets may be queued at node C. To allocate bandwidth resource fairly, we compensate nodes with more neighbors by generating more random numbers for them. Although this may not ensure absolute fairness, during schedule exchange phase, nodes that are not satisfied can grab redundant s_slots abandoned by other nodes so that the bandwidth resource is allocated based on nodes' traffic loads.

For the ith s_slot, the jth priority for a particular node $k \in N_2(u) \cup u$ is modified to be:

$$p_{k \oplus j}^i = [rand(k \oplus i)]_j \oplus k, k \in N_2(u) \cup u, j = 1, 2, ..., \chi_k \qquad (3)$$

where χ_k denotes the number of 1-hop neighbors of node k, and $[rand(k \oplus i)]_j$ is the jth random number generated by $rand(x)$. We use the number of one-hop neighbors because it is easy to maintain consistency among neighbors. A node only needs to attach its number of 1-hop neighbors to the SYNC packets. Node u wins the ith s_slot if Equation 4 is true.

$$\forall v \in N_2(u), \forall j \in [1, \chi_v], \exists m, p_{u \oplus m}^i > p_{v \oplus j}^i, 1 \leq m \leq \chi_u \qquad (4)$$

Equation 4 states that for the ith s_slot node u generates χ_u priorities for itself, if there exists a $p^i_{u \oplus m}$ that is greater than all of the priorities generated for all of its one-hop and two-hop neighbors, it wins the s_slot. Further, in order to randomize the s_slot assignment for each cycle, a random number generated by using the current cycle number as the seed is added to the seed for priority calculation:

$$p^i_{k \oplus j \oplus l} = [rand(k \oplus i + rand(l))]_j \oplus k \quad (5)$$
$$k \in N_2(u) \cup u, j = 1, ..., \chi_k$$

where l is the current cycle sequence that is synchronized by the SYNC packets.

2) Channel Utilization: Although pseudo-random functions provide us a low overhead time slot assignment solution, being overconfident in this solution leads to low channel utilization. As an example, for the ith s_slot, suppose the priority order generated by the aforementioned method for nodes in Fig. 3 is $P^i_A > P^i_B > P^i_C > P^i_D > P^i_E$. Because node D believes that node B, which has the highest priority in node D's two-hop communication neighborhood, will win the ith s_slot, node D cannot claim the s_slot. However, in the view of node B, node A has the highest priority. Consequently, although node D can use the ith s_slot along with node A, only node A will utilize the s_slot and the channel is underutilized. The worst case is when only one node in the entire network can win a particular s_slot if the priority order is globally sequential. Although the priority order is unlikely to be globally sequential, the locally sequential order is prevalent. To improve the channel utilization, it is necessary to check whether there are some underutilized slots. We address the issue through schedule exchange.

C. Schedule Exchange Mechanism

Each s_slot in the SLEEP period can be represented by one bit (1: occupied, 0: vacant). Consequently, a SLEEP period can be represented by several 32 bit unsigned integers. In implementation, a long SLEEP period is represented by repeating a short schedule $\frac{n}{k \times 32}$ times where n is the total number of s_slots in the SLEEP period and k is the number of 32 bit unsigned integers used for schedule representation. A node maintains seven schedule entries as below.

- sending_slot_assignment indicates slots owned by the node for data transmission.
- children_list records IDs of one-hop neighbors that will send data to the node.
- receiving_slot_assignment indicates slots in which the node should wake up to receive possible data packets.
- 1hop_slot_assignment indicates slots occupied by the node and its one-hop neighbors.
- 1hop_finalized_list records IDs of its one-hop neighbors that have finalized their sending_slot_assignment.
- 2hop_slot_assignment indicates slots occupied by the node and its neighbors in its two-hop neighborhood.
- 2hop_finalized_list records IDs of its one-hop and two-hop neighbors that have finalized their sending_slot_assignment.

Only the sending_slot_assignment, 1hop_slot_assignment, and the finalized_list will be exchanged among nodes as shown in Fig. 4. Others are derived from the three lists and maintained on each node for sending/receiving schedule construction.

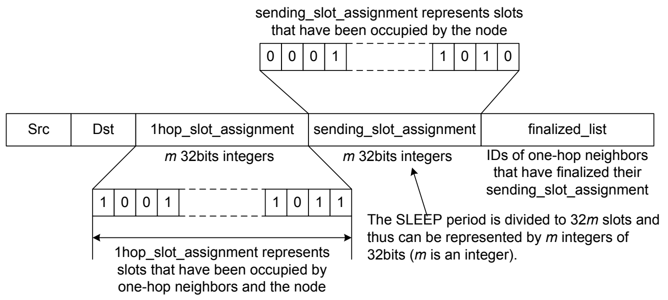
sending_slot_assignment represents slots that have been occupied by the node
0 0 0 1 1 0 1 0
Src
Dst
1hop_slot_assignment
sending_slot_assignment
finalized_list
m 32bits integers
m 32bits integers
IDs of one-hop neighbors that have finalized their sending_slot_assignment
1 0 0 1 1 0 1 1
The SLEEP period is divided to 32m slots and thus can be represented by m integers of 32bits (m is an integer).
1hop_slot_assignment represents slots that have been occupied by one-hop neighbors and the node

Fig. 4: Schedule packet format.

As discussed in Section V-A, a time frame is constructed through DRAND in the setup phase. A SCHED period consists of three DRAND frames. Each node broadcasts its schedule in its assigned d_slots in the three DRAND frames. After the 1st DRAND frame, each node learns the s_slot assignments of all its one-hop neighbors. After the 2nd DRAND frame, the schedule of a node is relayed to its two-hop neighbors and now each node knows whether a s_slot has been abandoned by its neighbors in the two-hop communication neighborhood. In the 3rd DRAND frame, a node can grab slots abandoned by its neighbors and broadcasts the finalized schedule to indicate when its neighbors should wake up to receive data packets.

1) Sending schedule packets: During the SCHED period, each node constructs and broadcasts a schedule packet in its assigned d_slot. The structure of the schedule packet is illustrated in Fig. 4. Whether a node owns a particular s_slot in the SLEEP period depends on its priority among all of its neighbors in the two-hop neighborhood and the schedules it has received. A node scans its 2hop_slot_assignment to look for s_slots that are not occupied by any of its neighbors in the two-hop communication neighborhood. Initially, all s_slots are available. For each unoccupied s_slot, the node checks whether it has the highest priority or whether all nodes that have higher priorities have already abandoned the s_slot. If any of the condition is true, the node claims the possession of the s_slot. Once it acquires enough number of s_slots, the loop is terminated and the node claims that its schedule has been finalized. The number of slots needed is calculated based on its queue size and the average incoming data rate measured in the last cycle. By claiming schedule as finalized, nodes of light traffic load release redundant winning slots to their neighbors.

To release redundant slots, a node informs its neighbors that it will not acquire any more slot other than what has been claimed in its sending_slot_assignment. Fig. 4 shows that a schedule packet includes a list of IDs whose schedules have been finalized. With this field, a node is able to check whether a s_slot has been abandoned by all neighbors of higher priorities in its two-hop communication neighborhood. In other words, when a node identifies an unoccupied slot, it checks whether all potential owners of higher priorities are in the 2hop_finalized_list. If the condition is true, the node can utilize the slot because all higher priority nodes have finalized their schedules without claiming the slot.

2) Receiving schedule packets: Upon receiving a schedule packet, a node uses bitwise OR (|) to merge the schedule it received with the schedule it maintains. More specifically, the node merges the *1hop_slot_assignment* in the received schedule packet with its maintained *2hop_slot_assignment*. In other words, a node aggregates its one-hop neighbors' *1hop_slot_assignments* to obtain the *2hop_slot_assignment*. The 2hop_slot_assignment is updated to indicate whether a slot is occupied by any of its neighbors in the two-hop communication neighborhood. For the *sending_slot_assignment* in the received schedule packet, the node merges it with its *1hop_slot_assignment* and if the node will receive packets from the sender, the node also merges it with its *receiving_slot_assignment*, which indicates when it should wake up to receive. Each node ID listed in the finalized_list of the received schedule packet is merged with the *1hop_finalized_list* and the *2hop_finalized_list*. The *1hop_finalized_list* is used to construct the *finalized_list* in a schedule packet and the *2hop_finalized_list* is used to check whether a node with a higher priority has finalized the schedule.

3) Case study: A simple example of a chain topology is illustrated in Fig. 5. The DRAND [11] produces a time frame of size 3, in which node C can transmit in the 1st d_slot, nodes A, D use the 2nd d_slot, and nodes B, E transmit in the 3rd d_slot as shown in the 'DRAND frame' block on top of the figure. The priority order for each s_slot shown in the 'Priority order' block in the figure is generated by the pseudo-random function Eq. 5. We show how five s_slots are assigned to nodes according to their traffic loads. Note that when a node determines the ownership of a s_slot, it only considers priorities of its neighbors in the two-hop communication neighborhood. In Fig. 5, we illustrate which node takes each s_slot for easy understanding, but in implementation, no node ID will be transmitted, only the binary occupancy representation will be transmitted.

Fig. 5 illustrates the operations of each node in the first two DRAND frames. The time progress is shown vertically. When the SCHED period starts, node C scans its 2hop_slot_assignment. Although initially all s_slots in the 2hop_slot_assignment are marked as available (i.e., '0'), node C has the highest priority for s_slot 1 only. For all other s_slots, it does not know whether higher priority nodes will use them or not. Therefore, node C only claims the possession of s_slot 1. If node C only needs one s_slot, it sets its schedule to finalized, which indicates that it will not acquire any more s_slot. However, if node C wants to acquire more s_slots by taking unoccupied s_slots owned by nodes of higher priorities, it should not claim its schedule as finalized. Suppose node A is not on any active route. It claims an empty finalized schedule. Node B notices that the schedule of node A is finalized and s_slot 3 and s_slot 5 are available. Node B can take both s_slots, but here we assume that node B only takes the s_slot 5 so that the s_slot 3 can be grabbed by node C. Node B must broadcasts the IDs of its one-hop neighbors who have finalized their schedules (i.e., finalized_list). Otherwise, node C is unable to utilize the s_slot 3 because it does not know whether the s_slot3 has been abandoned by node A or not.

In this design, a node cannot take any s_slot that belongs to a neighbor who is not in the *1hop_finalized_list*. Therefore, the loss of schedule packet will not cause any collision. It

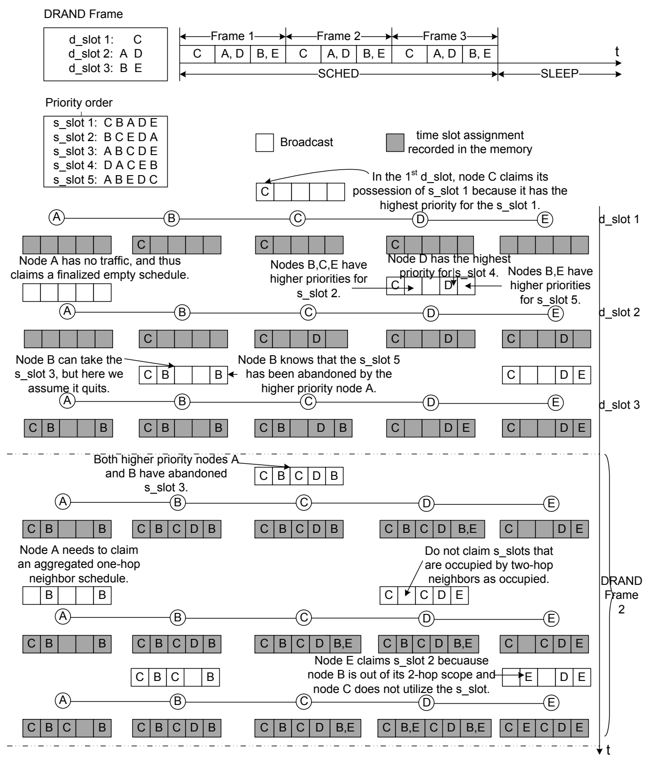
DRAND Frame
d_slot 1: C
d_slot 2: A D
d_slot 3: B E
Frame 1
Frame 2
Frame 3
C | A, D | B, E | C | A, D | B, E | C | A, D | B, E
SCHED
SLEEP
t
Priority order
s_slot 1: C B A D E
s_slot 2: B C E D A
s_slot 3: A B C D E
s_slot 4: D A C E B
s_slot 5: A B E D C
Broadcast
time slot assignment recorded in the memory
In the 1st d_slot, node C claims its possession of s_slot 1 because it has the highest priority for the s_slot 1.
d_slot 1
Node A has no traffic, and thus claims a finalized empty schedule.
Nodes B,C,E have higher priorities for s_slot 2.
Node D has the highest priority for s_slot 4.
Nodes B,E have higher priorities for s_slot 5.
d_slot 2
Node B can take the s_slot 3, but here we assume it quits.
Node B knows that the s_slot 5 has been abandoned by the higher priority node A.
d_slot 3
Both higher priority nodes A and B have abandoned s_slot 3.
Node A needs to claim an aggregated one-hop neighbor schedule.
Do not claim s_slots that are occupied by two-hop neighbors as occupied.
DRAND Frame 2
Node E claims s_slot 2 becuause node B is out of its 2-hop scope and node C does not utilize the s_slot.
A
B
C
D
E
t

Fig. 5: Time slot assignment in TAS-MAC.

just decreases the channel utilization because some abandoned slots cannot be utilized. A node chooses a neighbor of the minimum ETX as its next hop. It is unlikely that the next hop cannot receive the schedule packet and there are three chances in total. Even if all of the schedule packets are lost, the next hop can learn most of the sending slots of its children by checking the priorities for each slot.

To increase spatial reuse of s_slots, a node does not indicate s_slots taken by its 2-hop neighbors as occupied. The reason is that a one-hop neighbor of this node may reuse these s_slots. In the example above, node E does not care which s_slot is occupied by node B and can reuse it no matter whether it is occupied by node B or not. Node D thus should not indicate that the s_slot 2 is occupied.

Through the schedule exchange, the adverse impact of the sequential priority order is mitigated. Otherwise, for the five s_slots, only one node can transmit in each s_slot and no one will transmit in the s_slots owned by node A. Nodes that do not need s_slots or do not need to acquire more s_slots finalize their schedules in the first DRAND frame. Their neighbors only take the redundant s_slots that are abandoned by them and thus there is no collision. If a node intends to acquire more s_slots, it will not set its schedule to finalized and thus lower priority neighbors will not take any s_slot that might be taken by the node. Consequently, even though a node claims new s_slots in the 2nd or the 3rd DRAND frame, there is no collision. In total, each involved node broadcasts three schedule packets per cycle. The overhead of three schedule packets per cycle is trivial if the SLEEP period is long and we typically pursue a low duty cycle. A cycle is tens of seconds long while a schedule packet is around 10 ms, which is three

TABLE I: The key simulation parameters

Bandwidth	250 Kbps	Tx Power	31.32 mW
Transmission range	30 m	Rx Power	33.84 mW
Carrier sensing range	60 m	Idle Power	33.84 mW
SIFS	192 μs	Sleep Power	1.8 μW
slot time	30.5 μs	Transition Power	27 mW
d_slot/s_slot	7 ms	Transition Time	0.6 ms

to four orders of magnitude less than a cycle. Compared with prior protocols [6] [8] [9] [10] that need one control packet per node per packet, node-based scheduling of TAS-MAC has reduced the control overhead to three schedule packets per node per cycle and brings the benefit of allowing a number of packets to be forwarded through multiple hops per cycle. TAS-MAC seamlessly integrates opposite metrics: energy efficiency, high throughput, and low delay.

VI. Performance Evaluation

Among synchronous duty cycling MAC protocols, DW-MAC [10] provides low end-to-end delay for data delivery by allowing packets to be forwarded through multi-hops in a sequential way in the SLEEP period. The data transmission scheduling, however, is packet-based. Therefore, if a packet fails to be reserved for transmission in the current cycle due to the limited length of the DATA period, it will be postponed to the next cycle. We compare our TAS-MAC with DW-MAC to show that our node-based data transmission scheduling can avoid this unfair treatment toward packets and thus leads to lower average delay. Second, the traffic-adaptive time slot assignment of TAS-MAC improves the channel utilization from another perspective compared with the slot stealing method proposed in Z-MAC [3]. Although slot stealing increases the throughput, it sacrifices energy efficiency as it wakes nodes up in every slot. We show that the traffic-adaptive time slot assignment can achieve the similar high throughput and low delay performance with much lower power consumption.

We evaluated TAS-MAC in both ns-2 and on TelosB motes. Table I summarizes the key parameters used in our simulations. We draw parameters from the CC2420 datasheet [18] and the TelosB datasheet [19]. To account for the energy consumed for random number generation, we measured that on average a TelosB mote takes about 75 μs to generate a random number. When the TelosB mote is active, the current draw is 1.8 mA [19] and thus the power is 5.4 mW with an input voltage of 3 V. We add the additional energy consumption solely in our TAS-MAC to account for the computation cost.

Due to space limitation, we omit the study on a chain topology. The chain topology facilitates DW-MAC because there is no interference from other flows and thus it is unlikely that the important control packet SCH will be corrupted by collisions. A successful SCH reservation gives a packet low end-to-end delay as it flows continuously from the source to the destination. However, the length of the DATA period is upper bounded by the ratio between the control frame size and the data frame size because of the one-to-one mapping $\frac{T_{DATA}}{T_{SLEEP}} = \frac{SCH\_size}{DATA\_size}$. Therefore, some packets cannot get time slot reservation for transmission and have to be deferred to the next cycle. As a result, high variations on packet delivery delay are observed.

A. Simulations in random topologies

In a random topology, the SCH packets of DW-MAC may collide with each other and this significantly affects the performance of DW-MAC. We constructed networks in which 100 nodes are uniformly distributed in a $200 \times 200\ m^2$ field. To simulate a number of short flows due to event detection, we randomly select source nodes to generate packets of 128 bytes at a speed of 250 packets per second for a random period of time. The high data rate is used to examine how well a protocol can handle the bursty traffic. The number of concurrent flows is increased from 1 to 15 in a step of 2. The common active period of DW-MAC is set to the upper bound to achieve the maximum throughput, and the cycle length is set to 30 s because many synchronization protocols [16] [17] require a beacon interval of tens of seconds.

Fig. 6 presents the average throughput for different number of flows. We average results over ten random scenarios for each setting of the flow numbers. Depending on the topology, increasing the number of flows may increase or decrease the throughput of DW-MAC. If two flows do not interfere with each other, the throughput is increased. However, if they interfere with each other, increasing collisions of SCHs result in more wasted time slots and thus lower throughput. Since more flows bring higher probability of collision, the throughput of DW-MAC drops as the number of flows increases.

In TAS-MAC, increasing the number of flows increases the channel utilization. If there is only one flow, a downstream node may win a slot but has no data to send. Slot stealing would be helpful, but as the number of flows increases, the drawback of contention in slot stealing appears. Due to the hidden terminal problem, a node may incorrectly regard that a slot is available and commence transmission. The collisions introduced by slot stealing make the throughput of Z-MAC lower than that of TAS-MAC.

The average delay shown in Fig. 7 again confirms that DW-MAC is vulnerable to the loss of SCHs. More flows introduce more contentions and thus more failures of data transmission reservation. Once the DATA period ends, no reservation can be made and the remaining packets have to be postponed to the next cycle. The average delay thus increases steadily along with the number of flows. On the contrary, data packets can always be transmitted without reservation in TAS-MAC and Z-MAC. Whenever packets are ready, they can be forwarded in assigned time slots, leading to short delay. TAS-MAC is superior over Z-MAC in achieving the similar high throughput and low delay with much lower power consumption as shown in Fig. 8.

Fig. 8 shows that the average power consumption of nodes in TAS-MAC is much lower than that in Z-MAC and is two thirds of that in DW-MAC. In DW-MAC all nodes stay in idle listening in the DATA period. To achieve high throughput in DW-MAC, the DATA period has to be set long enough and this sacrifices energy efficiency. In Z-MAC all nodes have to wake up in every slot to check whether they are the intended receiver or whether they can send. The slot stealing method incurs high power consumption whereas in TAS-MAC and DW-MAC only nodes that are involved in data transmission will wake up in their assigned time slots.

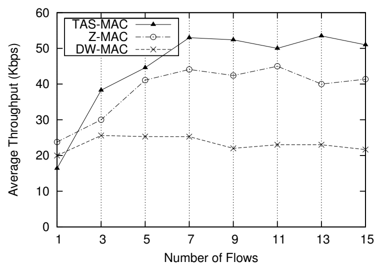

Fig. 6: Throughput comparison in random topologies.

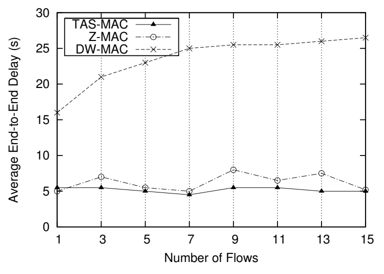

Fig. 7: End-to-end delay comparison in random topologies.

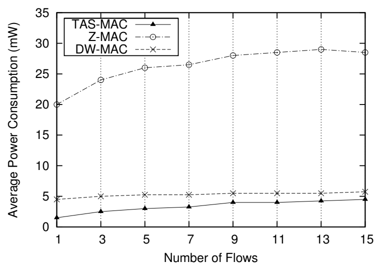

Fig. 8: Power consumption comparison in random topologies.

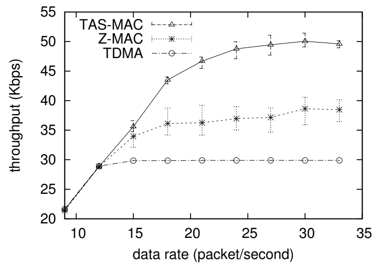

Fig. 9: The maximum throughput with different traffic loads.

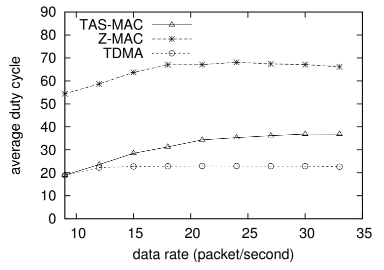

Fig. 10: The average duty cycle with different traffic loads.

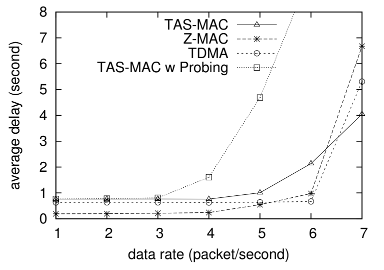

Fig. 11: The average delay with low traffic loads.

B. *Evaluation on TelosB Motes*

Duty cycling MAC protocols are unlikely to achieve the similar high throughput and low delay as Z-MAC because nodes in Z-MAC are always ready to receive as they wake up in every slot. Therefore, we only implemented TAS-MAC, Z-MAC, and a pure TDMA on TelosB motes.

1) Single-hop scenarios: To get the maximum achievable throughput, we first use a star topology to study why the traffic-adaptive time slot assignment can improve the throughput. The topology has 3 out of 5 nodes keep sending packets of 100 bytes to the sink. Fig. 9 compares the throughput of the three protocols. The results are averaged over ten experiments. Because a node can send only in its assigned time slots, the throughput of TDMA stops growing earlier. Both TAS-MAC and Z-MAC achieve higher throughput as a node can reuse the time slots owned by its neighbors. In Z-MAC, when a node intends to utilize an abandoned slot, it has to make sure that the slot is indeed available. After clear channel assessment (CCA), random backoff is used to reduce collisions between contending nodes. The channel utilization is thus not as high as that of TAS-MAC in which a node transmits immediately in its owned time slots. Simulations do not reflect system delays that exist in real implementation (e.g., delay for calling CCA and getting results back). The throughput gain of TAS-MAC over Z-MAC is thus more significant in experiments. The contention-based slot stealing also introduces greater variances on throughput due to collisions.

The difference between slot stealing and traffic-adaptive time slot assignment also resides in power consumption. As shown in Fig. 10, TAS-MAC has higher throughput, but the average duty cycle of nodes in TAS-MAC is 30% less than that of Z-MAC. The traffic notification mechanism in TAS-MAC lets nodes that are not on active routes sleep for the entire SLEEP period. Further, the schedule exchange let nodes on active routes only wake up in their assigned time slots. They do not need to wake up in every slot to contend for sending or to check whether they have data to receive. Because the number of claimed slots is proportional to a node's traffic load, a node increases its sleep period when it has fewer packets to send. Therefore, TAS-MAC provides a more energy-efficient way to improve channel utilization.

2) Multi-hop scenarios: We constructed multi-hop networks of 15 TelosB motes. The leaf nodes are 3 to 4 hops away from the sink. We simulated a target moving along the network boundary by blocking light to some leaf nodes. When a leaf node detects the moving target, it keeps generating packets at a certain rate until the target moves out of the detection range. TAS-MAC has a long initial delay that is incomparable to Z-MAC and TDMA. This is a disadvantage of synchronous MAC protocols. In synchronous MAC protocols, if an event is detected in the SLEEP period, the reports cannot be delivered until the next cycle begins. Therefore, packets experience an average initial delay of half the SLEEP period. To make the delay performance of TAS-MAC comparable to that of Z-MAC and TDMA, we let nodes work in the TDMA mode when no event is detected. We also tested the latest low power probing (LPP) method [20] in which a node broadcasts a beacon to request data when it wakes up periodically. TAS-MAC now can immediately start to deliver data packets even if the event is detected in the SLEEP period.

Fig. 11 shows the average delay in accordance with different data rates. When the data rate is low, a packet can be delivered quickly in Z-MAC because a node can utilize any time slot to transmit the data. However, in TAS-MAC, a node expects very light traffic load and only claims several slots for data transmission. The delay is thus slightly higher at low data rates. When there are many packets to be delivered, packets are queued at intermediate nodes. The delay is thus largely affected by the throughput performance of a MAC protocol. TAS-MAC provides the lowest delay at high data rates as

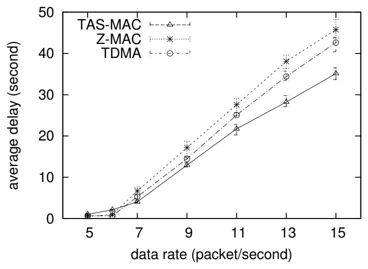

Fig. 12: The average delay with high traffic loads.

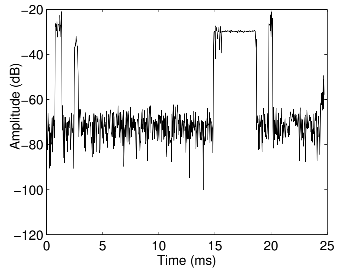

Fig. 13: Channel activities measured by USRP2 for LPP.

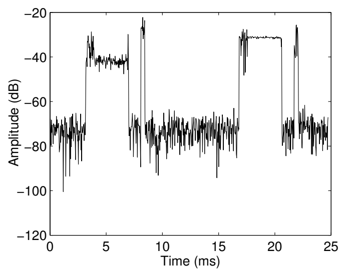

Fig. 14: Channel activities measured for TAS-MAC.

shown in Fig. 12 due to its high throughput.

Fig. 12 shows that TDMA has lower delay than Z-MAC at high data rates. This is because slot stealing introduces collisions of packets. Although Z-MAC utilizes some abandoned slots, the utilization is not high due to nontrivial system delay on low cost sensor nodes. The benefit is offset by the stealing overhead and introduced collisions.

The system delay is also the reason for the high delay of using LPP method introduced in A-MAC [20] in the SLEEP period as shown in Fig. 11. Therefore, we choose TDMA over LPP for the SLEEP period in normal conditions. In LPP, a node broadcasts a beacon to indicate that it is ready to receive when it wakes up. Upon receiving the beacon, neighbors acknowledge the beacon and initiate data transmission after random backoff. On a low cost sensor node, the time consumed to process a beacon, load data to TX FIFO, perform random backoff and CCA is nontrivial. As shown in Fig. 13, it takes time for a node to load data to the TX FIFO of CC2420 and perform random backoff after it acknowledges (the 2nd peak) the beacon (the 1st peak). For the same amount of time, we measured that a node can transmit an additional data packet if it transmits immediately without waiting for the receiver's beacon as shown in Fig. 14. When an event is detected in the SLEEP period, TAS-MAC cannot assign time slots to nodes until the next cycle starts. If the initial delay of a cycle is tolerable, we can use the initial design of TAS-MAC where nodes sleep for the entire SLEEP period. If energy efficiency can be sacrificed to ensure short reporting delay, we adopt TDMA in the SLEEP period when no event has been detected. This ensures that a node will wake up periodically in the SLEEP period and start data transmission soon after an event is detected in the SLEEP period. Once the next cycle starts, the traffic-adaptive time slot assignment of TAS-MAC is activated to assign time slots only to nodes that are located on active routes to achieve higher throughput and lower delay.

VII. Conclusion

High throughput and low delay are critical issues in low duty cycle synchronous MAC protocols because if the intended receiver cannot be notified within the common active period, all unfinished work has to be postponed to the next cycle. The common active period, however, cannot be set too long if we want to maintain the energy efficiency gained by duty cycling. In this paper, we present a novel low overhead, low duty cycle synchronous MAC protocol TAS-MAC that specializes duties in the common active period. The specialization allows us to derive a unique traffic-adaptive MAC protocol that achieves high throughput and low delay with low power consumption. Other traffic notification and data transmission scheduling methods can be designed under the framework to alleviate the tradeoff between energy efficiency and throughput or delay.

References

[1] T. He, S. Krishnamurthy, L. Luo, T. Yan, L. Gu, R. Stoleru, G. Zhou, Q. Cao, P. Vicaire, J. A. Stankovic, T. F. Abdelzaher, J. Hui, and B. Krogh, "VigilNet: An integrated sensor network system for energy-efficient surveillance," *ACM TOSN*, vol. 2, no. 1, Feb 2006.

[2] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh, "Fidelity and yield in a volcano monitoring sensor network," in *Proc. USENIX OSDI*, 2006, pp. 381–396.

[3] I. Rhee, A. Warrier, M. Aia, and J. Min, "Z-MAC: a hybrid MAC for wireless sensor networks," in *Proc. SenSys*, 2005, pp. 90–101.

[4] S. Gobriel, D. Mosse, and R. Cleric, "TDMA-ASAP: Sensor network TDMA scheduling with adaptive slot-stealing and parallelism," in *Proc. ICDCS*, 2009, pp. 458–465.

[5] P. Huang, L. Xiao, S. Soltani, M. Mutka, and N. Xi, "The evolution of MAC protocols in wireless sensor networks: A survey," *Communications Surveys Tutorials, IEEE*, vol. 15, no. 1, pp. 101–120, 2013.

[6] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Proc. INFOCOM*, vol. 3, Jun. 2002, pp. 1567–1576.

[7] ——, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 3, pp. 493–506, 2004.

[8] T. van Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," in *Proc. SenSys*, Los Angeles, CA, Nov. 2003, pp. 171–180.

[9] S. Du, A. K. Saha, and D. B. Johnson, "RMAC: A routing-enhanced duty-cycle MAC protocol for wireless sensor networks," in *Proc. INFOCOM*, May 2007, pp. 1478–1486.

[10] Y. Sun, S. Du, O. Gurewitz, and D. B. Johnson, "DW-MAC: A low latency, energy efficient demand-wakeup MAC protocol for wireless sensor networks," in *Proc. MobiHoc*, May 2008, pp. 53–62.

[11] I. Rhee, A. Warrier, J. Min, and L. Xu, "DRAND: Distributed randomized TDMA scheduling for wireless Ad-hoc networks," in *Proc. MobiHoc*, Florence, Italy, 2006, pp. 190–201.

[12] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves, "Energy-efficient collision-free medium access control for wireless sensor networks," in *Proc. SenSys*, 2003, pp. 181–192.

[13] L. Bao and J. J. Garcia-Luna-Aceves, "A new approach to channel access scheduling for Ad Hoc networks," in *Proc. MobiCom*, 2001.

[14] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris, "A highthroughput path metric for multihop wireless routing," in *Proc. MobiCom*, 2003, pp. 134–146.

[15] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *Proc. SenSys*, 2009, pp. 1–14.

[16] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The flooding time synchronization protocol," in *Proc. SenSys*, 2004, pp. 39–49.

[17] C. Lenzen, P. Sommer, and R. Wattenhofer, "Optimal clock synchronization in networks," in *Proc. SenSys*, 2009, pp. 225–238.

[18] "CC2420 datasheet," http://www.ti.com.

[19] "TelosB datasheet," http://www.xbow.com.

[20] P. Dutta, S. Dawson-Haggerty, Y. Chen, C.-J. M. Liang, and A. Terzis, "Design and evaluation of a versatile and efficient receiver-initiated link layer for low-power wireless," in *Proc. SenSys*, 2010, pp. 1–14.