Revisiting Sensor MAC for Periodic Monitoring: Why Should Transmitters Be Early Birds?

Daewoo Kim, Jinhwan Jung, Yoonpyo Koo, and Yung Yi

***Abstract*—We propose a new sensor MAC protocol, called *Bird-MAC,* which is highly energy efficient in the applications where sensors periodically report monitoring status with a very low rate, as in structural health monitoring and static environmental monitoring. Two key design ideas of Bird-MAC are: (a) no need of early-wake-up of transmitters and (b) taking the right balance between synchronization and coordination costs. The idea (a) is possible by allowing a node (whether it is a transmitter or receiver) to wake up just with its given wake-up schedule, and letting a late bird (which wakes up later) notify its wake-up status to its corresponding early bird (which wakes up earlier), where the early bird just infrequently waits (i.e., nods) for the late bird's wake-up signal. The idea (b) is realized by designing Bird-MAC to be placed in a scheme between purely synchronous and asynchronous schemes. We provide rigorous mathematical analysis that is used to choose the right protocol parameters of Bird-MAC. We demonstrate the performance of Bird-MAC through extensive simulations, and real experiments using a 26 node testbed at an underground parking lot of our office building to monitor its structural health, where we confirm that energy consumption is reduced by about up to 45% over existing sensor MAC protocols.**

I. Introduction

In this paper, we consider the sensing applications where battery-powered sensors monitor and *periodically* send their status to sinks but the monitoring period is highly long. Such applications include monitoring of structural health, environment, or smart grid. It is reported that the global structure health monitoring market was valued at USD 505.0 million dollars in 2014 and is expected to grow at a CAGR of 24.7% between 2015 and 2020 [1]. For example, it is reported that even a period of an order of weeks is enough in monitoring the health (e.g., the existence of crack due to fatigue) of a large bridge [2][1], whose period is even longer than that typically considered in literature (from an order of hours to a few days) [2]–[5]. The major goal in those applications is to deliver monitoring status with high energy-efficiency with reasonably low delay.

It is widely known that the main energy source is due to listening to the wireless channel and transceiving packets, where *duty cycling* is a natural way of saving energy by periodically switching on and off the radio. In the sensor MAC protocols based on duty cycling, energy consumption is due to the following combination of two cost sources: (i) *coordination cost* which corresponds to the energy consumed to coordinate communication in the presence of clock drift, (ii) *synchronization cost* which is the amount of energy to exchange synchronization control messages. These two energy costs often form a trade-off, depending on target applications (e.g., volume and pattern of traffic from sensors). Asynchronous protocols, e.g., [10], [11], [17], where communication over a link is coordinated in such a way that whenever a TX has a packet to send, it wakes up its intended RX by sending a long preamble signal, are regarded as the ones that have no synchronization cost, but a large amount of coordination cost. Purely synchronous protocols, e.g., [6], however, have large synchronization cost due to frequent signaling for synchronization, but small coordination cost.

Thus, an energy-efficient sensor MAC protocol should be designed so as to choose a good trade-off and minimize its energy waste. We claim that despite a large array of existing sensor MAC protocols in sensor networks (see Section I-A), a large room for saving energy still exists, if MAC is smartly designed in a *customized* manner considering periodic traffic pattern with highly long period. We propose a new MAC, called *Bird MAC*, which achieves high energy efficiency by finding the right tradeoff between coordination and synchronization costs and exploiting the traffic periodicity in environmental monitoring. The key design features are summarized in what follows:

- ***Avoiding early wake-up of transmitters.*** In most existing duty-cycled MAC protocols, communication is coordinated by the following design guideline: a transmitter (TX) *wakes up earlier* than its designated receiver (RX) and coordinates the communication with its RX. This TX's "early-wake-up" rationale is popularly applied because TX's backlog status is regarded unpredictable, and thus TX with data backlog becomes responsible for coordination so as to minimize RX's unnecessary energy waste. Under this design, TX is required to wake up an amount of maximum clock drift earlier than RX not to miss the communication coordination chance, resulting in a large amount of energy waste. Bird-MAC allows a node to wake up just with its given wake-up schedule, and the node (TX or RX) which wakes up later to initiate communication. This idea becomes possible because in the applications with periodic data generation, sensing data's availability is predictable. This feature of Bird-MAC

The authors are with the Department of Electrical Engineering, KAIST, South Korea (e-mails: {dwkim,jhjung,ypkoo,yi}@lanada.kaist.ac.kr.)

[0]This work was supported by the Center for Integrated Smart Sensors funded by the Ministry of Science, ICT & Future Planning as Global Frontier Project (CISS-2012M3A6A6054195) and Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No.B0717-17-0034,Versatile Network System Architecture for Multi-dimensional Diversity)

[1]This is because it is typical that large-scale structures such as bridges or building show symptoms of problems in their health for a non-negligible time before actual collapse occurs.

TABLE I
COMPARISON: RELATED WORK

Protocols	Sync/Async	TX/RX-initiated	Early wake-up
S-MAC [6], T-MAC [7], DS-MAC [8], D-MAC [9]	Sync	TX	O
B-MAC [10], X-MAC [11], TICER [12], SpeckMAC [13]	Async	TX	O
SCP-MAC [14], WiseMAC [15], Dozer [16]	Partial	TX	O
RI-MAC [17], A-MAC [18], RCMAC [19]	Async	RX	O
PW-MAC [20], AS-MAC [21]	Partial	RX	O
LB-MAC [22]	Async	TX-RX	O
This paper: Bird-MAC	Partial	TX-RX	X

enables nodes to consume energy only in proportion to *actual* clock drift (often being much smaller than maximum clock drift), leading to large energy saving.

- ***Balance between synchronization and coordination costs.*** As mentioned earlier, asynchronous and purely synchronous protocols are not the good candidates for periodic sensing with highly long period, because of too much sync message overhead (synchronous protocols) and long preamble signal (asynchronous protocols).This motivates us to infrequently synchronizing nodes' clock, where synchronization period is optimally chosen so as for the total energy consumption to be minimized (see our mathematical analysis in Section IV).

Evaluation. We evaluate the performance of Bird-MAC by implementing it on top of Contiki OS [23], which enables us to carry out both simulations and real experiments. In simulations, we use the Cooja simulator inside the Contiki OS, which allows a variety of controllable setups and microbenchmarks. In real implementations, we build a testbed equipped with 16 Z1 and 10 MICAz motes in an underground parking lot of our office building. In our evaluations, we observe that the average energy consumption for Bird-MAC is at most 60% of other existing protocols. The full source code of our implementation is available in [24].

A. Related Work

There exists an extensive array of research on sensor MAC protocols, where we make our best effort with space limited to appropriately position Bird-MAC. We classify the existing sensor MAC protocols by two criteria: **(a)** asynchronous or synchronous, and **(b)** TX-initiated or RX-initiated. In **(a)**, being asynchronous or synchronous is determined by the existence of explicit synchronization phase, and in **(b)**, protocols are differentiated by who initiates the communication (see Table I for the key difference of Bird-MAC from existing protocols).

(a): Synchronous protocols [6]–[9] synchronize nodes' clock "very frequently" so as not to need additional coordination, whereas asynchronous ones do not synchronize time of sensor nodes. B-MAC [10] and X-MAC [11] are the representative examples of asynchronous protocols that do not synchronize the clock. In B-MAC, TX with backlogs sends a long preamble signal which lasts longer than RXs' sleep period. X-MAC improves B-MAC by, rather than using a long preamble signal, TX's transmitting the strobed preamble packets which can be cut off by RX's ACK signal for energy saving. Protocols that share similar design include [12], [13], [17], [18], [22], [25]. In partially synchronous protocols, nodes synchronize explicitly by exchanging synchronization control messages [14], [16], or implicitly in which the nodes roughly predict the wake up time of intended RX based on a simple scheduling information [15], [20], [21]. Since they synchronize or update scheduling information infrequently, when they exchange data packet, additional coordination is required to combat against clock drift.

(b): Aforementioned MAC protocols can also be classified into TX-initiated and RX-initiated schemes by the following criterion, as shown in Table I. In TX-initiated schemes, a TX examines the wake-up status of its RX and initiates a communication, whereas in RX-initiated ones, a backlogged TX just waits for its RX's wake-up notification. The strength of RX-initiated protocols lies in avoiding unnecessary channel occupation by TX's polling, however, RX consumes more energy for notification than that in TX-initiated (where RX just monitors the channel). LB-MAC [22] is a hybrid scheme that either TX or RX can be adaptively selected as an coordination-initiating node in a situation dependent manner (e.g., the remaining energy of both TX and RX).

Our work. Bird-MAC is partially synchronous, but we run it with the optimal synchronization period (mathematically studied as a function of system parameters). Bird-MAC also works in such a way that TX or RX can initiate the communication coordination depending on who wakes up earlier, so as the coordination cost to be mainly determined by the actual clock drift rather than the maximum one. Our work is close to LB-MAC [22] in the sense that LB-MAC is also TX-RX-initiated. However, LB-MAC is proposed as an asynchronous scheme, and it is less suitable for periodic sensing traffic, because as in Bird-MAC, such a periodicity can offer a lot of information for a MAC to operate much more effectively. Bird-MAC is designed to parameterize a protocol beween purely synchronous and asynchronous and behave under a good tradeoff point, whereas LB-MAC is designed to operate in an asynchronous manner and parameterize between purely TX-initiated (e.g., [11]) and purely RX-initiated (e.g., [17]). Due to LB-MAC's asynchronous feature, LB-MAC is unable to avoid TXs' early-wake-up, whereas Bird-MAC does not require TXs to wake up early, which we believe is one of the major source to save energy in periodic status monitoring.

II. BIRD-MAC: FRAMEWORK

In this section, we describe the overall framework of Bird-MAC. We assume that when nodes are initially deployed, a certain routing protocol initially runs and produces routing

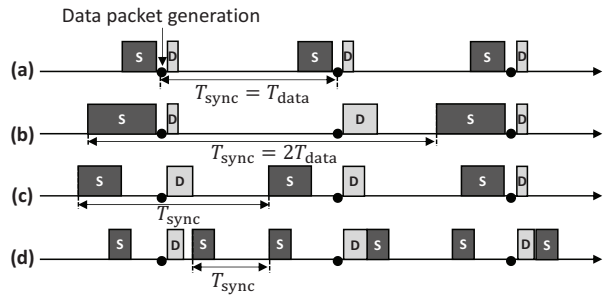


Fig. 1. Various cases of sync ('S') and data ('D') phases. The width of each block is proportional to the energy cost due to coordination.

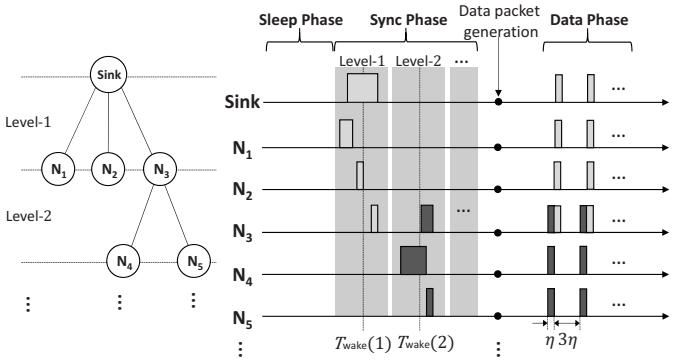


Fig. 2. Framework of Bird-MAC. Three phases: Sleep, Sync, and Data. Nodes are scheduled based on its level, but due to clock drift they wake up differently within shaded area which means maximum possible clock drift.

paths (from sensors to a sink), configured by a form of *tree*[2] (see the left of Fig. 2). Tree-like routing paths are popularly considered in other papers in sensor networks, see e.g., [26]. From this routing path construction phase, all nodes are aware of topology information, e.g., the number of children nodes and level (depth in a routing tree).

Three phases. In Bird-MAC, time consists of repetition of the following three phases: (i) *sleep phase* where each node is dormant by turning off its radio, (ii) *sync phase* where nodes synchronize their clock to the sink's reference clock in a top-down fashion (from the sink to lower-level nodes in the routing tree), and (iii) *data phase* where sensed data is transferred to the sink in a bottom-up fashion. We believe that most sensor MAC protocols with synchronization would require the above three phases. However, the key design choice towards high energy efficiency is how to organize these phases on which Bird-MAC chooses the followings:

How often and when to synchronize? Let T_{data} and T_{sync} be the data generation and synchornization periods, respectively. Fig. 1 shows various ways of organizing three phases. In Figs. 1(a) and 1(b), sync and data phases are aligned with T_{sync} being a multiple of T_{data} around when sensed data is generated, whereas in Fig. 1(c) sync and data phases are unaligned and in Fig. 1(d) time is synchronized more often than data generation. Note that the width of a rectangle corresponds to the amount of energy in the corresponding phase. For example, the energy consumed for sync phase in Fig. 1(b) is higher than that in Fig. 1(a) due to a larger clock drift, but synchronization frequency becomes less. In Bird-MAC, we choose the policy of *"equal data and sync frequency with alignment"* as in Fig. 1(a), i.e.,

$$T_{\text{sync}} \quad = \quad T_{\text{data}}, \tag{1}$$

where (i) time synchronization first occurs with a possibly large clock drift, and then (ii) sensing is made, and finally (iii) sensed data transfer to the sink is performed. Thus, the major energy consumption in Bird-MAC is due to communication coordination of sync message exchanges in the presence of a large amount of clock drift (because sync phase immediately follows the long sleep phase), whereas data communication can be done with a negligible clock drift. Our design in the above comes from our energy-efficient medium access control and communication coordination between a TX-RX pair (see Section III) and the rigorous mathematical analysis (see Section IV).

Bird-MAC is not coupled with any specific synchronization protocol, and thus a popular pair-wise synchronization scheme, e.g., see [27], can be utilized. In our implementation, this pair-wise synchronization is performed sequentially from the sink to the leaves at each level in the routing tree, whereas in data phase, nodes are scheduled sequentially from the leaves to the sink based on their levels in the routing tree as shown in Fig. 2.

Pipelined wake-up scheduling for delay reduction. In performing the operations in sync and data phases, we employ a pipelined wake-up scheduling that renders each node at different levels to wake up at different times, enabling data delivery from sensors to be completed in one wake-up period. This wake-up scheduling, in addition to our smart selection of synchronization period, highly helps for small delay (see Section V).[3] We call the set of nodes with depth l and $l+1$ by the *level-l nodes*. For example, in Fig. 2, the level-1 nodes are the sink, N_1, N_2, and N_3 and the level-2 nodes are N_1, N_2, N_3, N_4, and N_5. The wake-up time $T^l_{\text{wake}}(s)$ of level-l nodes at s-th period is set such that they are guaranteed to meet and grab the chance to communicate. To this end, the wake-up time of level-l node is set as ΔT_{wake} earlier (resp. later) than level-$(l+1)$ nodes in sync phase (resp. data phase), i.e., for every period s,

$$|T^{l+1}_{\text{wake}}(s) - T^l_{\text{wake}}(s)| = \Delta T_{\text{wake}}, \quad \text{where} \quad \Delta T_{\text{wake}} = 2\bar{\gamma}\tau + \eta.$$

In the above, τ is the time elapsed since the last synchronization, and $\bar{\gamma}$ is maximum clock drift rate. η is the slack time due to some overhead caused by multiple children (e.g., resolving contentions). Note that the value of τ differs for sync and data phases in Bird-MAC. In sync phase, with $T_{\text{sync}} = T_{\text{data}}$, maximum clock drift becomes $2\gamma T_{\text{data}}$, but in data phase, $\tau = 0$ due to negligible clock drift. We just describe our rule of selecting τ coming from a different configuration of sync and data phases. In our design, η is set as $\bar{n} \cdot 15$ msec in sync phase

[2]We assume that there exists a single sink for simplicity, but our protocol can be readily extended to multiple sinks.

[3]This pipelined scheduling was proposed in [9], [28], [29], and we only present how to adopt such an idea with the right parameters in our design.

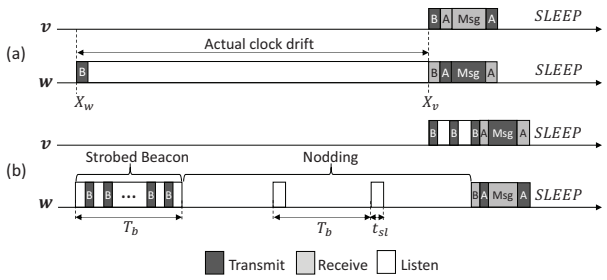


Fig. 3. Two key concepts for energy-efficient coordination in Bird-MAC: (a) late-bird initiated and (b) early-bird nodding.

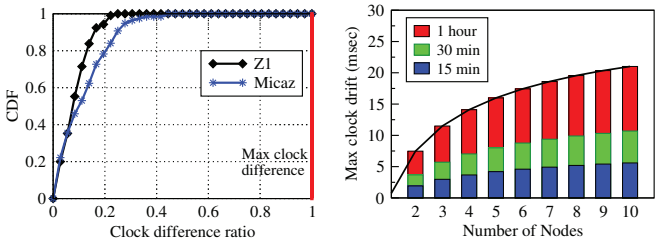


Fig. 4. Clock drift measurement of Z1 and MICAz motes. (a) Clock difference distribution of two nodes: 90% of samples are less than 30% of maximum possible clock difference specified in the data sheet. (b) Expectation of maximum clock drift among n nodes with varying measurement intervals.

to give chances to all children to coordinate safely, and 15 msec in data phase, where $\bar{n}$ is the maximum number of children in routing tree (see Section III-B for details).

Each level-l nodes at each phase s perform message exchange when they wake up, which we call *level task*, following an appropriate control of communication coordination and contention, which is the key contribution of Bird-MAC, elaborated in the next section.

III. Bird-MAC: Level Task

In this section, we explain how the level task is performed in whose major tasks are described as (i) energy-efficiently coordination control and (ii) pure medium access control in Sections III-A and III-B, respectively.

A. Coordination Control: Avoiding Early-Wake-Up

Communication coordination is required to let nodes obtain the chance to meet after a certain dormant duration (in which case a random amount of clock drift at each node is generated) and perform data transfers among them. The set of nodes of any level, say l-level, can be decomposed into a collection of small subtrees with depth one (i.e., one parent and multiple children, if any). For example, in Fig. 2, level-2 nodes are decomposed into the subtrees $\{N_1\}, \{N_2\}$, and $\{N_3, N_4, N_5\}$. Each of these subtrees is the basic unit of communication coordination[4]. For expositional convenience, we describe the coordination control for the case when the subtree consists of a single parent-child pair (each of which can be either of TX or RX depending the direction of data transfer in sync and data phases). See the end of this section for the case of multiple children. In a TX-RX pair we henceforth call a node which wakes up earlier (resp. later) *early bird* (resp. *late bird*).

1) Late-bird Initiated: The basic concept of the late-bird initiated coordination is described as follows: in a communication pair of nodes v and w,

(a) Each node v transmits a beacon signal to its partner w to notify its wake-up.
(b) Then, w returns an ACK to signal for its reception of v's beacon.
(c) If v fails to receive w's ACK, indicating that v is an early bird, then it waits for the beacon of w, which is the late bird.
(d) Upon reception of ACK from the early bird, the late bird immediately starts communication (thus communication is initiated by the late bird).

Note that v can be either TX or RX. As an example in Fig. 3(a), w is the early bird and thus transmits a beacon, but no ACK, because v, which is the late bird, is still dormant. Thus w waits for v's beacon to be sent some time later, and returns ACK for v's beacon to start communication.

We comment that this coordination scheme is similar to that of [22]. In Bird-MAC, this late-bird initiated coordination removes the case when TXs unnecessarily wake up early, whereas in LB-MAC TXs should still wake up early because it is an asynchronous protocol. We modify LB-MAC's scheme in our partially synchronous framework and improve it to be more energy efficient by applying early-bird nodding and supporting coordination aggregation where multiple TX's transmissions are handled by one coordination, as explained in Section III-A3.

Rationale. This simple, yet powerful idea of *late-bird initiated* helps a lot to save communication, because the energy consumption due to communication coordination in presence of clock drift is in proportion to the length of actual clock difference between two nodes in a TX-RX pair. This is in stark contrast to most of existing MAC protocols, where TX always wakes up earlier than RX as much as maximum possible clock difference, and waits until RX wakes up to coordinate communication. To intuitively understand, let $X_v, X_w \in [-T_{\max}, T_{\max}]$ be the random variables with zero mean, representing the times when nodes v and w wake up (nodes are scheduled to wake up at 0), and $T_{\max}$ is maximum possible clock drift of a sensor node. Then, the expected wake-up time for coordination in *late-bird initiated* and *TX-early-wakeup* (which is the philosophy of many existing protocols) are $E[|X_v - X_w|]$ and $E[X_w - (X_v - 2T_{\max})] = 2T_{\max}$, respectively. Note that the expected actual clock drift between two nodes $E[|X_v - X_w|]$ tends to much shorter than maximum clock drift $T_{\max}$, as demonstrated by our measurement results in Fig. 4(a) and that in [30].

The benefit of late bird-initiated idea comes from our intention of design a MAC by tailoring into the target applications's feature: *periodic monitoring*, which as we belive constitutes a non-negligible portion of sensing applications, where data backlog status is predictable. However, conventional TX-early-

[4]There may exist interference among different subtrees, which will be discussed in Section III-B.

wake-up schemes are designed to cope with unpredictable traffic generation, where RX does not know when data is ready, thus requiring TX to wake up earlier than RX.

2) Early-bird Nodding: This corresponds to an idea that an early bird does not continue to be in the listen mode to catch the wake-up signal from the late bird, but repeats switching on and off with nodding interval T_b. Then, the late bird's wake-up notification with beacon signal should last at least for T_b, so that the early bird can catch this signal, as depicted in Fig. 3(b). This additionally saves a lot of energy, especially when synchronization is done very infrequently. To further save energy, a beacon signal is strobed, i.e., the sequence of sub-beacons with a short interval, during which ACK from its partner can be received. [5] Note that to make early-bird nodding and beacon strobing work, inter-beacon time should be long enough to receive and decode at least one ACK. In order not to miss this beacon packet, once the node is switched on while nodding, it listens the channel for the duration t_{sl}, which is slightly larger than inter-beacon time. In our implementation of Bird-MAC over Z1 mote, we set inter-beacon time to be 5.5 msec considering the transmission and processing delays of 7-byte ACK packet including 5-byte header, and t_{sl} is set as 7 msec.

Choice of nodding interval T_b**.** The nodding interval T_b is an important parameter that trades off energies consumed for nodding and transmitting a strobed beacon. As T_b grows, a node can nod less but should transmit a longer strobed beacon. In Bird-MAC, we suitably choose T_b, so that the consumed energy is minimized considering a given environment such as the number of children n in a subtree and clock drift (see Section IV for mathematical derivation). In our design, we choose T_b to be:

$$T_b = 2\sqrt{\frac{c(n\sqrt{\log 2}+\sqrt{\log(n+1)})t_{sl}T_{\text{data}}}{(3n+4)(1-\beta)\gamma}} \quad (2)$$

where c is the parameter representing the distribution of clock drift which is determined by the variance of clock drift (see Section IV for more details), and β is so-called a *beacon suppressed ratio* corresponding to the portion of beacons from multiple nodes that do not have to be sent due to overhearing beacons in the neighborhood (see Section III-B2). As (2) shows, T_b depends on n, so that each subtree should set T_b differently using the information from the routing protocol. It is intuitive that T_b increases as T_{sync} and n grow, because larger clock drift and more children increase the waiting time of the early bird, and in this case, nodding less by setting T_b larger can reduce the total energy consumption in spite of a longer strobed beacon. We will provide a rigorous analysis to present how the optimal nodding interval in (2) is derived in Section IV.

[5]Similar beacon strobing was applied in asynchronous protocols, e.g., [11], but it is first applied to a (partially) synchronous protocol in Bird-MAC with the optimal nodding interval derived mathematically, and we further apply it to *coordination aggregation* for additional energy saving (see Section III-A3 for details).

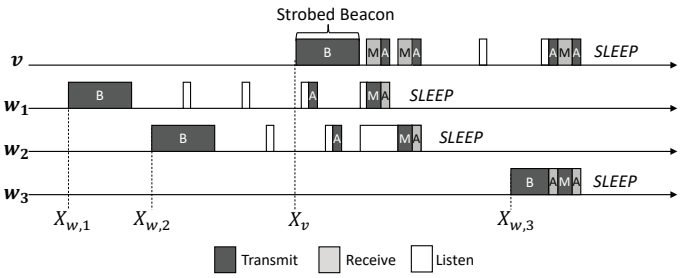


Fig. 5. Coordination aggregation in one parent and three children.

3) Multiple Children (One to many): We have so far presented how the nodes in a TX-RX pair coordinate themselves for a communication. However, in practice, as in Fig. 2, there may exist multiple children for one parent in the routing tree. For further energy efficiency, Bird-MAC uses a notion of *coordination aggregation*, where the parent v waits long enough to finish the coordination with all of their children (whose number is available from the routing protocol) rather than individually coordinates with each child.

Fig. 5 shows how coordination aggregation works. The parent v transmits a full strobed beacon without stopping it, even if it receives ACK from some of its children, which is necessary to guarantee that all early-bird children receive v's wake-up notification. Moreover, if a child receives the beacon packet, (e.g., w_1, w_2 in Fig. 5) it falls asleep and wakes up again at the end of parent's strobed beacon. This sleeping period can be computed using the sequence number marked in the beacon packet, and the child grabs the channel through contention with other children (see Section III-B). If the parent receives ACK for the strobed beacon, it stays awake for a certain time whose length depends on how seriously contention occurs, to wait for Msg packets after finish transmitting the strobed beacon, otherwise it enters nodding immediately.

B. Contention, Reliability Control, and Adaptation to Changes

1) Handling collisions: One of a MAC's roles is to control contentions in presence of multiple wireless nodes intending to transmit data. As a basic contention control mechanism, we employ CSMA. The main target is data being transmitted correspond to strobed beacons and Msgs, but we focus on the contentions due to the strobed beacons, since as we discuss, there are two unique challenges ***C1*** and ***C2*** that do not arise in the conventional contention control over a wireless network.

C1: Strobed beacon is hard to sense. A beacon signal is strobed with a series of small beacon packets that are repetitively transmitted every t_b interval. Thus, even if a node occupies the channel, other nodes can kick in between two beacon packets, resulting in beacon collisions that ongoing strobed beacon is interrupted by other nodes, if a classical sensing mechanism (i.e., just sensing the channel for a short time) is applied.

Our design: Long listen before transmit. To tackle the challenge above, we let a node sense for sufficiently long to know whether there exists a strobed beacon, where its length should last longer than the inter-beacon time (5.5 msec in our implementation), as depicted in Fig. 6. We set this sensing time to be

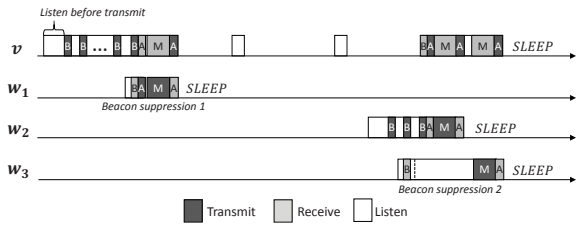


Fig. 6. Beacon suppression. Coordination of one RX and three TXs in Fig. 2. Nodes wake up in the sequence of Sink, N_1, N_2, and N_3.

10 msec, in order not to interrupt an ongoing strobed beacon as well as Msg packet whose maximum back-off time slot is 9.92 msec determined by the back-off rule in *inter-beacon sensing* below. This sensing functionality is implemented at the MAC-level due to implementation simplicity, i.e., the default classical carrier sensing is in RF chip, so hard to reprogram it, at least at the platform used by ours. In the CC2420 as the RF chip in Z1 mote the sensing time is as short as only 128 μsec.

C2: No ACK does not always imply collision. In spite of our design for avoiding collisions, it may be imperfect, in which case, we need to alleviate contentions e.g., a backoff scheme. Another challenge lies in detecting collisions, which is typically done by ACK. Strobed beacons lead to false negatives, i.e., the absence of ACK does not always imply a collision. This is because a TX's receiver cannot send while it is asleep.

Our design: Inter-beacon sensing. In Bird-MAC, while a node transmits a strobed beacon, it is able to listen to the channel over a inter-beacon period to receive ACK, and detect the collision when some signal is detected, in which case it backs off for a random amount of time. In CC2420 of Z1 mote, one back-off time slot is 0.32 msec and maximum slot size is 2^5-1, and the node retransmits at most 7 times.

Contention control for Msg pacekts. Once nodes are coordinated by exchanging beacon and ACK in sync phase or just waking up in data phase, multiple nodes can intend to transmit Msg packets. Since Msg packets are not strobed, convetional CSMA with ACK can be used, where the rules for back-off and retransmission are the same as those of the strobed beacon.

2) Beacon suppression: Further energy saving can be possible if a certain portion of strobed beacons can be suppressed. To this end, we are able to use nodes' ability of overhearing the communications in their neighborhood. Thanks to our MAC-level sensing in the previous section, a node is capable of often decoding overheard packets, and then suppresses its strobed beacons in the following ways: First, when a child w overhears its parent's beacon, w suppress its beacon and directly replies ACK to its parent and communicate at the end of parent's beacon (see *Beacon suppression 1* in Fig. 6). Second, when w_3 overhears w_2's beacon which is a sibling of w_3, it keeps listening until the ongoing beacon is terminated. If the ongoing beacon ends with successful communication, w_3 tries to communicate with its parent v at the end of communication between v and w_2 (see *Beacon suppression 2* in Fig. 6), and otherwise it infers that the parent is in the sleep state, and thus starts nodding to wait for the parent's beacon. Third, when a parent w receives its child's beacon, w sends back ACK to the child and communicate immediately. If all communications with the entire children are finished prior to transmission of its own beacon, it directly falls into sleep without its beacon.

3) Handling transmission failures: Despite the contention control in Section III-B1, transmission failure is sometimes unavoidable due to collisions, hidden nodes, and time-varying wireless conditions, in particular, in wild areas. In applications with highly long periods just as those considered in this paper, the cost of a transmission failure is high. To provide more reliability to Bird-MAC, we furnish each transmission with a final chance of retransmission in what follows: If a node v fails to communicate since an early bird misses its associated late bird's beacon or fails to transfer its Msg packet in spite of retransmission, v is provided a final chance to communicate driven by the early bird's transmission of the final strobed beacon (F.B.) after a certain timeout. The timeout duration is chosen as maximum clock difference (i.e., $2\gamma T_{\text{sync}}$) to guarantee the communication coordination. We also deal with the hidden node problem as follows: The parent v wakes up first, and w_1 and w_2 which are hidden from each other, wake up at a similar time, where the final beacon of v coordinates both nodes, and thus they can communicate. We confirmed that this transmission failure occurs very infrequently, thus energy-efficiency is not highly affected.

IV. Analysis: Parameter Selection

We design Bird-MAC to work for a wide variety of hardware configurations, e.g., clock accuracy and the consumed energy when dormant. At the heart of such a flexible design lies our parameter selection given by the theoretical analysis. In this section, we provide theoretical analysis that verifies our choice of two important parameters: T_{sync} that determines how often nodes should be synchronized ((1) in Section II) and T_b that corresponds to the nodding interval ((2) in Section III-A2). Our derivation is based on the following assumptions.

First, let $X_i(\tau)$ denote the amount of clock drift for node i for the duration of time τ. Then, $X_i(\tau)$ has zero mean, following a distribution that satisfies $E[\max_{1\le i\le n} X_i(\tau)] = c\tau\sqrt{\log n}$. [6] Second, We assume that link is reliable, thereby no link-level transmission failures exist. Under the extremely low data rate application, backoff and retransmission costs due to collisions are much smaller than coordination cost. Third, We focus on a subtree of depth one with one parent and n children for analytical tractability. This analysis can be readily extended to an entire network, because the coupling between two level-trees does not highly impact on the analysis. Finally, in an unaligned case (see Fig. 1), the starting time of sync phase is randomly chosen.

Decomposition of consumed energy. Let P_{on} (watt) be the amount of energy when in RX mode (i.e., sum of MCU and RF

[6] In Gaussian distribution with zero mean, following bounds hold: $\frac{\sigma}{\pi\log 2}\sqrt{\log n} \le E[\max_{0\le i\le n}\{X_i\}] \le \sigma\sqrt{2}\sqrt{\log n}$ [31]. Thus, the distribution in **A1** can represent that of clock drift in Fig. 4. Using MMSE (Minimum mean square error) fitting, we set c as 3.58×10^{-6} in Z1 mote, which fits well to the clock drift measurement result (see Fig. 4(b)).

powers), and the RF consumes γP_{on} watt in TX mode, where γ depends on the chip-dependent transmission power. Similarly, let P_{off} denote the amount of power when a node is asleep. We normalize these powers, so that $P_{\text{off}} = 0$ and P_{on} is the differential power relative to P_{off}.

The entire energy E consumed per unit time by the nodes in a subtree can be decomposed into: $E = E_{\text{co\_sync}} + E_{\text{sync}} + E_{\text{co\_data}} + E_{\text{data}}$, where E_{sync} and E_{data} denote energies for exchanging sync and data packets between one parent and n children nodes, respectively. These two energies will be determined by both T_{sync} and T_{data}, and time for exchanging packets (t_s and t_d) which are related to the size of each packet and n, e.g. $E_{\text{sync}} = nt_s(1+\gamma)P_{\text{on}}/T_{\text{sync}}$ and $E_{\text{data}} = nt_d(1+\gamma)P_{\text{on}}/T_{\text{data}}$. $E_{\text{co\_sync}}$ and $E_{\text{co\_data}}$ are the energies consumed for coordinating communication for data and sync messages, respectively. Typically, as T_{sync} shrinks (thus more frequent synchronizations), E_{sync} grows and $E_{\text{co\_sync}}$ decreases. Recall that two design parameters T_{sync} and T_b affect E, thus $E = E(T_{\text{sync}}, T_b)$. In particular, as discussed in Section II, E_{sync}, $E_{\text{co\_sync}}$, and $E_{\text{co\_data}}$ are the functions of T_{sync}, and $E_{\text{co\_sync}}$ and $E_{\text{co\_data}}$ are the functions of T_b.

A. Optimal T_b and T_{sync}

First, we say that sync and data phases are *k-aligned,* when $T_{\text{sync}} = kT_{\text{data}}$ for some positive integer k, i.e., synchronization is performed every k sensing data generation. In this case, synchronization is carried out, and then sensed data is transferred, and thus $E_{\text{co\_data}}$ becomes negligible. Recall that t_{sl} is the time required to receive a beacon when the node wakes up while nodding (see Section III-A2) and β is the beacon suppressed ratio (see Section III-B2). Theorem 4.1 states which choices of T_b and T_{sync} minimize the total energy consumption under what conditions. The proof is presented in our technical report [24] due to space limitation.

Theorem 4.1: Under the assumption that $T_{\text{data}} > T_{\text{th}}$, where $T_{\text{th}} = \left(\frac{nt_s(1+\gamma)}{\alpha(\sqrt{2}-1)}\right)^2$ with

$$\alpha = \sqrt{c(n\sqrt{\log 2} + \sqrt{\log(n+1)})(3n+4)(1-\beta)\gamma t_{sl}},$$

E is minimized when sync and data phases are 1-aligned (i.e., $T_{\text{sync}} = T_{\text{data}}$ as described in (1)), and the following choice of T_b and T_{sync} is made:

$$T_b = 2\sqrt{\frac{c(n\sqrt{\log 2} + \sqrt{\log(n+1)})t_{sl}T_{\text{data}}}{(3n+4)(1-\beta)\gamma}}. \quad (3)$$

Furthermore, with the choices of (3) and (1), the expected energy consumed per unit time E is given by:

$$E = \frac{P_{\text{on}}}{T_{\text{data}}}\left(\alpha\sqrt{T_{\text{data}}} + n(1+\gamma)(t_s+t_d)\right). \quad (4)$$

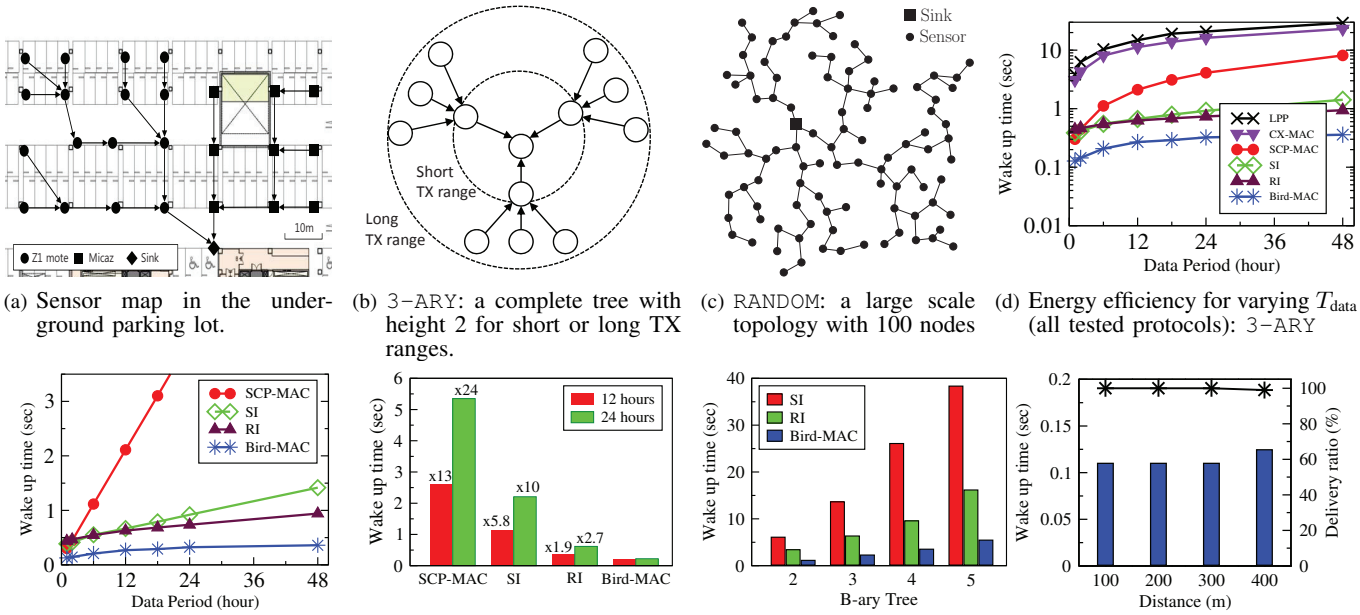
Interpretation. A few interpretations are in order. *(a)* First, T_{th} is about 1 minute in the subtree with one child in practice, and it is upper-bounded by 3 mins.[7] Thus, optimal choices in (3) of Theorem 4.1 works for a fairly large class of data sensing generation frequency. *(b)* Second, aligning helps in energy efficiency because aligning highly eliminates the need of additional coordination of data phase. Our choice $k = 1$ takes the best tradeoff between synchronization and additional coordination costs for data phase (not overlapped with the sync phase). Especially when data generation period is long, additional coordination cost exceeds synchronization cost, thereby setting $T_{\text{sync}} = T_{\text{data}}$ is optimal. *(c)* Third, the effect of early-bird nodding is that we see that the coordination cost for one synchronization, $\alpha P_{\text{on}}\sqrt{T_{\text{data}}}$ in (4) is proportional to $\sqrt{T_{\text{data}}}$, whereas the clock drift at a sync phase is proportional to T_{data}. Thus, early-bird nodding helps in coordinating communication in a more energy efficient manner when data packets are generated with a highly long period.

[7] In Z1 mote, $P_{\text{on}} = 68$mW, $\gamma = 1$ (TX power is set as 0dBm), $t_{sl} = 7$ msec, and $t_s = 0.96$ msec (two packets with sizes of 6 and 9 bytes are exchanged under the data rate of CC2420, which is 250 kbps).

V. Implementation and Evaluation

Implementation. We use both Z1 and MICAz motes to run our implementation of Bird-MAC. The radio in both Z1 and MICAz is CC2420 supporting the data rate of 250 kbps. Z1 and MICAz motes use TI MSP430 and ATMEGA128L MCUs whose maximum clock drift rate is 25 ppm. We implement Bird-MAC on top of Contiki OS [23]. The source codes and all the test scripts are available in [24]. The key modules of Bird-MAC such as pipelined wake-up scheduling, synchronization, coordination control, and contention control are implemented at the RDC (Radio Duty Cycle) layer of Contiki, where we disable the default contention control module, and modify the CC2420 radio driver to implement the overhearing capability for beacon surpression. Routing paths are made by RPL routing protocol [26] in Contiki OS, and the metric of RPL is set as minimizing ETX (Expected Transmission Count). In simulations for constructing a controlled environment, we use the Cooja simulator inside Contiki. In simulations, to emulate more practical situations, we base our simulations on Cooja's multi-path ray tracing model (MRM) which models radio hardware properties, background noise and interference through SINR.

Setup, parameters, and metric. In simulations, traffic is periodically generated with the random clock drifts following the distribution from our measurement (see Fig. 4). For real experiments, we build up a testbed with 26 motes (16 Z1 motes and 10 MICAz motes) in an underground parking lot of our office building as depicted in Fig. 7(a). Following the analytical result in Section IV, we choose T_b and T_{sync} with dependence on the given topology and other hardware-dependent values. Other parameters such as the backoff counts, contention window sizes and retransmission counts are chosen based on the 802.15.4 Zigbee. Our primary interest is energy-efficiency, delay, and delivery ratio. Especially in experiments, to estimate the energy consumption of a mote, we use the energy estimating module in Contiki OS, which records the active and sleep times of MCU, and the RF chip. Due to the fact that a mote consumes an extremely small power during the sleep state, which is also a baseline power consumption, we plot the energy consumption of just the active state.



(a) Sensor map in the underground parking lot.

(b) `3-ARY`: a complete tree with height 2 for short or long TX ranges.

(c) `RANDOM`: a large scale topology with 100 nodes

(d) Energy efficiency for varying T_{data} (all tested protocols): `3-ARY`

(e) Energy efficiency for varying T_{data} (SCP-MAC, SI, RI, Bird-MAC): `3-ARY`

(f) Energy efficiency for `RANDOM` and T_{data} = 12, 24 hours

(g) `B-ary` trees (B = 2, 3, 4, 5), T_{data} =1 day, short TX range

(h) Impact of link quality by varying inter-sensor distance.

Fig. 7. Evaluations of Bird-MAC by varying diverse environmental factors.

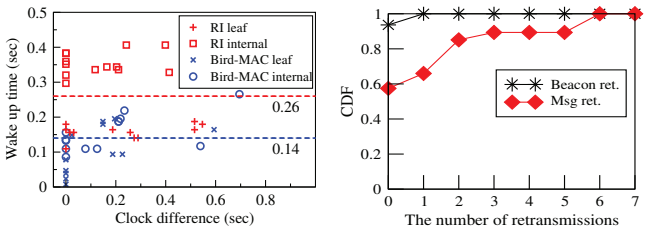
Tested protocols. In simulations, we compare Bird-MAC with other asynchronous and partially synchronous protocols. For asynchronous protocols, we test CXMAC and LPP in Contiki OS. CXMAC is a sender-initiated MAC protocol based on X-MAC [11], and LPP is a receiver-initiated MAC protocol similar to RI-MAC [17]. We do not present LB-MAC [22]'s performance due to space limitation, but its total energy consumption was similar to or slightly better than CXMAC and LPP. For partially synchronous protocols, we directly implement SCP-MAC [14] and two "artificial" protocols *SI* and *RI*. *SI* and *RI* contain time synchronization in sender-initiated and receiver-initiated manners, respectively. Note that these two artificial protocols are not for our convenience, but for fair comparison to evaluate the impact of our late-bird initiated idea, where they are exactly the same as Bird-MAC, e.g., early-bird nodding, except for who initiates in communication coordination. Those are generalized and more energy-efficient versions of SCP-MAC [14] and PW-MAC [20] that are partially synchronous protocols in literature, because SCP-MAC and PW-MAC's nodding interval is 0. In the real testbed, we compare Bird-MAC with *RI* which turns out to show the best energy efficiency out of all tested protocols in most simulations.

A. Results: Simulation

In simulations, we use two types of topologies: (i) `B-ARY` complete tree with $B = 2, 3, 4, 5$ with height 2 and (ii) random topologies with $5, 10, 20, 50, 100$ nodes, which we denote by `RANDOM`, as shown in Figs. 7(b) and 7(c). For `RANDOM` topologies, we run the RPL routing protocol [26] for constructing a routing tree, whereas in `B-ARY`, the routing tree is set to be the same as the original topology.

(a) *Energy efficiency:* Fig. 7(d) shows the energy efficiency of all tested protocols, where we particularly magnify the results of SCP-MAC, SI, RI, and Bird-MAC in Fig. 7(e) due to too large gap among other protocols. First, we observe that asynchronous protocols consume an order-of-magnitude energy larger than other partially synchronous protocols, because, as expected, asynchronous protocols are optimized for unpredictable data generations. Fig. 7(e) quantitatively illustrates the energy efficiency gain of the late-bird initiated rationale and the early-bird nodding of Bird-MAC, which grows as T_{data} increases (about 2.61, 3.94 and 22.6 times less than RI, SI, and SCP-MAC for 48-hour T_{data}). This trend does not change for `RANDOM`, as shown in Fig. 7(f), where we observe that Bird-MAC outperforms other three protocols by at least 1.9 times and up to 24 times under $T_{\text{data}} = 12$ hours and 1 day.

(b) *Impact of environmental changes:* First, Fig. 7(g) shows the results when T_{data} =1 day, for a varying number of children in `B-ARY`, where $B = 2, 3, 4, 5$. As discussed in Section III, depending on those changes, we appropriately set our protocol parameters and factors, e.g., nodding interval T_b and beacon suppression ratio β. We observe that Bird-MAC sustains its energy efficiency, and its gap from other protocols tends to increase thanks to beacon suppression (see Section III-B2). Second, we examine the impact of link quality by changing the inter-sensor distance, ranging from 50 m to 300 m when $T_{\text{data}} = 1$ day, where in the data sheet of Z1 mote, 100 m distance is recommended for robust connection. This is for testing how reliability is supported and how energy efficiency at that time (using retransmissions and the notion of final beacon in Section III-B3) is achieved. Fig. 7(h) shows 100% delivery ratio is provided up to 200 m distance, and about 99% delivery ratio is achieved for 300 m. Finally, to test the scalability of



(a) Energy efficiency for `Z1` and `MICAz` motes (b) The number of retransmissions of beacons and Msgs

Fig. 8. Experimental result based on our real implementation.

Bird-MAC, we also investigated the impact of node density and network size, where we confirmed that node density does not affect the performance much and delay of data delivery increases only sub-linearly as the network size. We refer the readers to [24] for more details.

B. Results: Real Experiment

In the testbed deployed at an underground parking lot of our office building, each node reports its sensing data to a sink every 12 hours. We run both Bird-MAC and *RI* (which shows the best performance in simulations) for 7 days for comparison. Fig. 8(a) shows the energy consumption of all motes, where x-axis denotes each node's average clock difference with its child and parent. The energy consumption of *RI* doubles that of Bird-MAC on average (0.26 sec vs. 0.14 sec). This is because the amount of clock drift for all motes is relatively much smaller than the maximum clock drift which is the major source of RI's high energy consumption. Fig. 8(b) shows the CDF of the number of retransmissions in beacon and Msg, respectively. More transmission failures occur than in simulations, as expected, but we have confirmed that Bird-MAC achieves 100% delivery ratio. This demonstrates that the maximum retransmission count 7 in Section III-B1 is a good choice at least in our test environment.

VI. Conclusion

We developed a new sensor MAC protocol, called *Bird-MAC,* which is highly energy efficient in the applications where sensors periodically report monitoring status with a very low rate. Two key design features of Bird-MAC are: (i) partially synchronous and (ii) no early-wake-up. A large energy-saving effect of this late-bird initiated scheme (irrespective of whether it is a transmitter or a receiver) is due to the fact that nodes wake up only during actual clock drift between a transmitter-receiver pair, whereas the wake-up duration of existing approaches is proportional to that of the maximum clock drift.

References

[1] Research and Markets, "Structural health monitoring market by technology (wired and wireless), by solution type (hardware and software & services), by application (bridges, dams, tunnels, buildings, stadiums, and other), and by geography - global trend & forecast to 2020," 2015.

[2] P. Liu *et al.*, "Development of a wireless nonlinear wave modulation spectroscopy (NWMS) sensor node for fatigue crack detection," in *Proc. of Smart Structures and Materials and Nondestructive Evaluation for Health Monitoring*, 2014.

[3] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habit monitoring," in *Proc. of ACM International Workshop on Wireless Sensor Networks and App.*, 2002.

[4] J. Beutel, S. Gruber, S. Gubler, A. Hasler, M. Keller, and R. Lim, "The PermaSense remote monitoring infrastructure," in *Proc. of ISSW*, 2009.

[5] M. Erol-Kantarci and H. T. Mouftah, "Wireless sensor networks for cost-efficient residential energy management in the smart grid," *IEEE Transactions on Smart Grid*, vol. 2, no. 2, pp. 314–325, 2011.

[6] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Proc. of IEEE INFOCOM*, 2002.

[7] T. V. Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," in *Proc. of ACM SenSys*, 2003.

[8] P. Lin, C. Qiao, and X. Wang, "Medium access control with a dynamic duty cycle for sensor networks," in *Proc. of IEEE WCNC*, 2004.

[9] G. Lu, B. Krishnamachari, and C. Raghavendra, "An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks," in *Proc. of IEEE PDPS*, 2004.

[10] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proc. of ACM SenSys*, 2004.

[11] M. Buettner, G. Yee, E. Anderson, and R. Han, "X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks," in *Proc. of ACM SenSys*, 2006.

[12] E.-Y. Lin, J. M. Rabaey *et al.*, "Power-efficient rendez-vous schemes for dense wireless sensor networks," in *Proc. of IEEE ICC*, 2004.

[13] K.-J. Wong and D. Arvind, "SpeckMAC: low-power decentralised MAC protocols for low data rate transmissions in specknets," in *Proc. of Workshop on Multi-hop Adhoc Networks: From Theory to Reality*, 2006.

[14] W. Ye, F. Silva, and J. Heidemann, "Ultra-low duty cycle MAC with scheduled channel polling," in *Proc. of ACM SenSys*, 2006.

[15] A. El-Hoiydi and J.-D. Decotignie, "WiseMAC: An ultra low power MAC protocol for multi-hop wireless sensor networks," in *Algorithmic Aspects of Wireless Sensor Networks*. Springer, 2004, pp. 18–31.

[16] N. Burri, P. Von Rickenbach, and R. Wattenhofer, "Dozer: ultra-low power data gathering in sensor networks," in *Proc. of IEEE IPSN*, 2007.

[17] Y. Sun and D. B. Johnson, "RI-MAC : A Receiver-Initiated Asynchronous Duty Cycle MAC Protocol for Dynamic Traffic Loads in Wireless Sensor Networks," in *Proc. of ACM SenSys*, 2008.

[18] P. Dutta *et al.*, "A-MAC: a versatile and efficient receiver-initiated link layer for low-power wireless," *ACM Transactions on Sensor Networks*, vol. 8, no. 4, p. 30, 2012.

[19] P. Huang, C. Wang, L. Xiao, and H. Chen, "RC-MAC: A receiver-centric medium access control protocol for wireless sensor networks," in *Proc. of IWQoS*, 2010.

[20] L. Tang, Y. Sun, O. Gurewitz, and D. B. Johnson, "PW-MAC: An energy-efficient predictive-wakeup MAC protocol for wireless sensor networks," in *Proc. of IEEE INFOCOM*, 2011.

[21] B. Jang *et al.*, "AS-MAC: an asynchronous scheduled mac protocol for wireless sensor networks," in *Proc. of IEEE MASS*, 2008.

[22] Y. Peng, Z. Li, W. Zhang, and D. Qiao, "Lb-mac: a lifetime-balanced mac protocol for sensor networks," in *International Conference on Wireless Algorithms, Systems, and Applications*, 2012.

[23] A. Dunkels *et al.*, "Contiki-a lightweight and flexible operating system for tiny networked sensors," in *Proc. of IEEE LCN*, 2004.

[24] "Technical report and source code," http://lanada.kaist.ac.kr/pub/birdmac/.

[25] S. Liu *et al.*, "CMAC: An energy-efficient MAC layer protocol using convergent packet forwarding for wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 5, no. 4, p. 29, 2009.

[26] T. Winter and P. Thubert, "Rpl: Ipv6 routing protocol for low-power and lossy networks," *IETF RFC6550*, 2012.

[27] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proc. of ACM SenSys*, 2003.

[28] S. Guo and T. He, "Robust multi-pipeline scheduling in low-duty-cycle wireless sensor networks," in *Proc. of IEEE INFOCOM*, 2012.

[29] S. Guha, C.-K. Chau, and P. Basu, "Green wave: Latency and capacity-efficient sleep scheduling for wireless networks," in *Proc. of IEEE INFOCOM*, 2010.

[30] M. Brzozowski, H. Salomon, and P. Langendoerfer, "On efficient clock drift prediction means and their applicability to ieee 802.15. 4," in *Proc. of IEEE/IFIP EUC*, 2010.

[31] G. Kamath, "Bounds on the expectation of the maximum of samples from a Gaussian," http://www.gautamkamath.com/writings/gaussian_max.pdf.