Bird-MAC: Energy-Efficient MAC for Quasi-Periodic IoT Applications by Avoiding Early Wake-up

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

Abstract—We propose a new MAC protocol for IoT applications, called *Bird-MAC,* which is highly energy efficient in the applications where IoT sensors report monitoring status in a quasi-periodic manner, 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 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 a 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. The experiment on our testbed using a 26 node testbed at an underground parking lot of our office building to monitor its structural health, shows that energy consumption is reduced by about up to 45% over existing sensor MAC protocols. We also confirm the applicability of Bird-MAC in a challenging and realistic scenario through the experiment on Yeongjong Grand Bridge in South Korea.

Index Terms—IoT, Wireless Sensor Network, MAC protocol.

1 Introduction

In this paper, we consider the sensing applications where battery-powered sensors monitor and *quasi-periodically* send their status to sinks but the monitoring period is highly long. Such applications seem to cover a non-negligible portion of IoT applications, such as 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 [2]. In those applications, it is also 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 [3][1], whose period is even longer than that typically considered in literature (from an order of hours to a few days) [3]–[6]. 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 MAC protocols based on duty cycling, energy consumption is due to the following combination of two cost sources: (i) *(communication) 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., [11], [12], [18], 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., [7], however, have large synchronization cost due to frequent signaling for synchronization, but small coordination cost.

Thus, an energy-efficient IoT MAC protocol should be designed so as to choose a good trade-off between those two cost sources and minimize its energy waste. We claim that despite a large array of existing IoT MAC protocols (see Section 1.1), a large room for saving energy still exists, if MAC is smartly designed in a *customized* manner considering quasi-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 quasi-periodicity in environmental monitoring. The key design features are summarized in what follows:

(a) ***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 as 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

The authors are with the Department of Electrical Engineering, KAIST, South Korea (e-mails: {dwkim,jhjung,ypkoo}@lanada.kaist.ac.kr, yiyung@kaist.edu.)
Part of this work was presented at IEEE SECON 2017, San Diego [1].

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 1: Summary of Related Work

Protocols	Sync/Async	TX/RX-initiated	Early wake-up
S-MAC [7], T-MAC [8], DS-MAC [9], D-MAC [10]	Sync	TX	O
B-MAC [11], X-MAC [12], TICER [13], SpeckMAC [14]	Async	TX	O
SCP-MAC [15], WiseMAC [16], Dozer [17]	Partial	TX	O
RI-MAC [18], A-MAC [19], RCMAC [20]	Async	RX	O
PW-MAC [21], AS-MAC [22]	Partial	RX	O
LB-MAC [23]	Async	TX-RX	O
This paper: Bird-MAC	Partial	TX-RX	X

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 quasi-periodic data generation, sensing data's availability is predictable. This feature of Bird-MAC 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.

(b) ***Balance between synchronization and coordination costs.*** As mentioned earlier, asynchronous and purely synchronous protocols are not the good candidates for quasi-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 4).

(c) ***Evaluation: Simulation and real experiments.*** We evaluate the performance of Bird-MAC by implementing it on top of Contiki OS [24], which enables us to carry out both simulations and real experiments with the same code. 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 two testbeds. The first testbed is equipped with 16 Z1 and 10 MICAz motes in an underground parking lot of our office building. We also apply Bird-MAC on 16 Z1 motes deployed on Yeongjong Grand Bridge in South Korea. 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 [25].

Bird-MAC is a protocol that is tailored to IoT applications with periodic traffic pattern in order to save a large amount of energy, i.e., sacrificing generality to some degree for energy efficiency. However, as demonstrated in Section 5, Bird-MAC works well for *quasi-periodic* applications with a certain amount of traffic randomness being allowed ($1\sim2\%$ of the average period in our simulations, corresponding to ±30 mins variance at maximum with the average period of 48 hours). We believe that designing this type of customized protocol such as Bird-MAC is of value when a non-negligible amount of energy gain is guaranteed in IoT applications.

1.1 Related Work and Organization

There exists an extensive array of research on sensor MAC protocols, where we try 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 1 for the key difference of Bird-MAC from existing protocols).

(a): Synchronous protocols [7]–[10] synchronize nodes' clock "very frequently" so as not to need additional coordination, whereas asynchronous ones do not synchronize time of sensor nodes. B-MAC [11], X-MAC [12] and RI-MAC [18] 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. RX which listens to channel for a short time periodically can detect the preamble signal from TX and they can communicate with each other. 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 [13], [14], [18], [19], [23], [26]. In partially synchronous protocols, nodes synchronize explicitly by exchanging synchronization control messages [15], [17], or implicitly in which the nodes roughly predict the wake up time of intended RX based on a simple scheduling information [16], [21], [22]. 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 1. 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 [23] 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 [23] 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 between purely synchronous and asynchronous and behave under

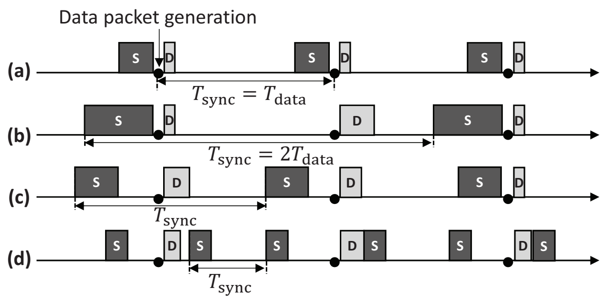

Data packet generation
(a)
(b)
(c)
(d)
S
D
$T_{\text{sync}} = T_{\text{data}}$
$T_{\text{sync}} = 2T_{\text{data}}$
T_{sync}
T_{sync}

Fig. 1: Four example configurations of sync ('S') and data ('D') phases. The width of each block is proportional to the average energy cost for communication coordination and message exchange.

a good tradeoff point, whereas LB-MAC is designed to operate in an asynchronous manner and parameterize between purely TX-initiated (e.g., [12]) and purely RX-initiated (e.g., [18]). 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 sources to save energy in periodic status monitoring. This paper is an extended version of our preliminary work [1], where (i) more extensive simulations and realistic experimental results and (ii) the protocol and analytical details are added, providing useful insight into our protocol development.

Organization. The rest of this paper is organized as follows: In Section 2, we first overview the framework of Bird-MAC, followed by the design details of Bird-MAC in 3. In Section 4, we present how the protocol parameters are chosen via a rigorous mathematical analysis, and then provide performance evaluation results in Section 5. Finally, we conclude in Section 6.

2 Overview and 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 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, see e.g., [27]. 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 MAC protocols for IoT sensors 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 synchronization 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

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

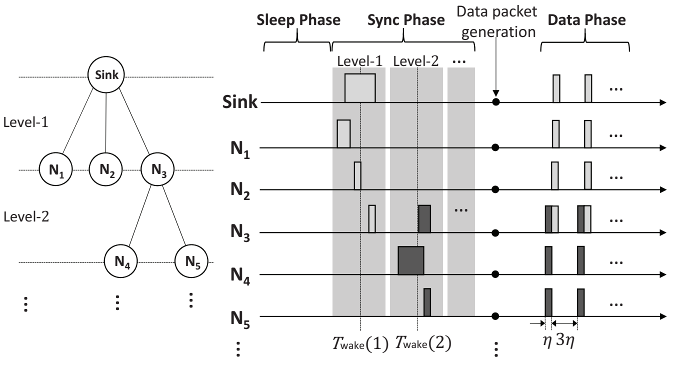


Fig. 2: Framework of Bird-MAC. Nodes are scheduled based on its level, but due to their clock drifts they wake up at different times within the maximum clock drift (shaded areas). Different colors means different levels coming from a routing tree.

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 average energy consumption in the corresponding phase, which includes the cost due to coordination and message exchange. 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.,

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

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 3) and the rigorous mathematical analysis (see Section 4).

Bird-MAC is not coupled with any specific synchronization protocol, and thus a popular pair-wise synchronization scheme, e.g., see [28], 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. In performing the operations in sync and data phases, we employ a pipelined wake-up scheduling that permits each node at different levels to wake up at different times. In sync phase, we schedule the levels in a top-down fashion, so that all nodes are able to synchronize their clocks to the reference one within one sync phase by synchronizing their clocks to that of their parent sequentially. For example, in sync phase of Fig. 2, N_3 is activated in level-1 slot to synchronize its clock to its parent, and also wakes up in level-2 slot to act as a parent. During data phase, scheduling the levels in a bottom-up manner enables 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 in achieving small delay (see

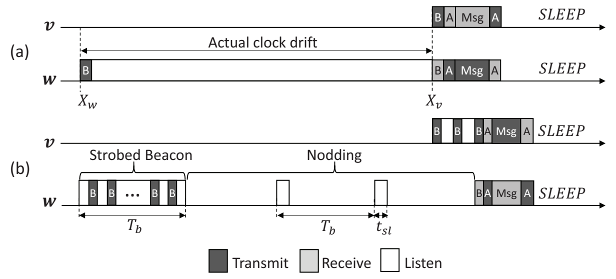


Fig. 3: Two key concepts for energy-efficient coordination in Bird-MAC: (a) late-bird initiated and (b) early-bird nodding.

Section 5).[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. In the figure, the non-filled and filled blocks represent the activated times of level-1 and level-2 nodes, respectively. These blocks in sync phase include coordination and message exchange, whereas those in data phase only include message exchange, because there is a negligible clock drift thanks to alignment. 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 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 3.2 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.

3 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 3.1 and 3.2, respectively.

3.1 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

3. This pipelined scheduling was proposed in prior work [10], [29], [30], but our contribution lies in how to adopt such an idea with the right parameters in our design.

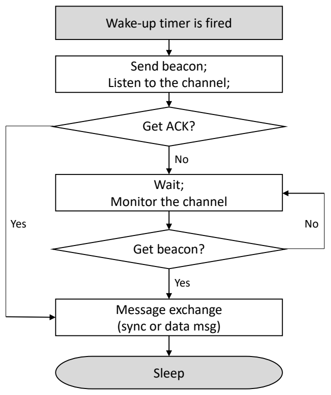


Fig. 4: Flow chart of late-bird initiated coordination.

a random amount of clock drift at each node is generated) and perform data transfers among them. In Bird-MAC, coordination is used for both sync and data message exchange, and the proposed coordination schemes can be applied to both sync and data phases.

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*).

3.1.1 Late-bird Initiated

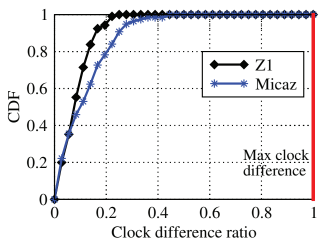
The basic concept of the late-bird initiated coordination is described in Fig. 4. In a communication pair of nodes v and w,

(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).

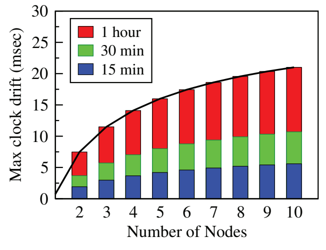
Note that v can be either TX or RX. As an example in Fig. 3(a), w is the early bird and thus transmits a beacon, but no ACK, because v, which is the late bird, is still dormant. Thus w waits for v's beacon to be sent some time later, and returns ACK for v's beacon to start communication.

We comment that this coordination scheme is similar to that of [23]. 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

4. There may exist interference among different subtrees, which will be discussed in Section 3.2.



(a)



(b)

Fig. 5: 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.

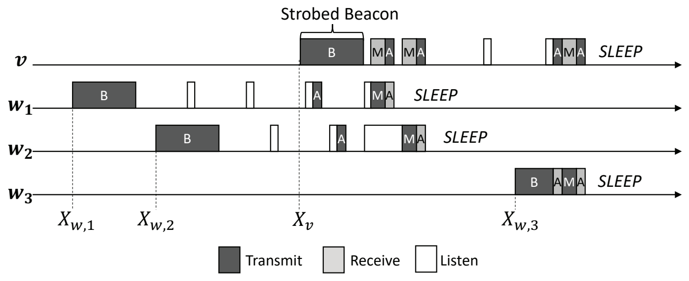


Fig. 6: Coordination aggregation in one parent and three children.

early-bird nodding and supporting coordination aggregation where multiple TX's transmissions are handled by one coordination, as explained in Section 3.1.3.

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. 5(a) and that in [31].

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

3.1.2 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 to 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 4 for mathematical derivation). In our design, we choose T_b to be:

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

where c is the parameter representing the distribution of clock drift which is determined by the variance of clock drift (see Section 4 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 3.2.2). 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 4.

3.1.3 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

5. Similar beacon strobing was applied in asynchronous protocols, e.g., [12], 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 3.1.3 for details).

is available from the routing protocol) rather than individually coordinates with each child.

Fig. 6 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. 6) 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 3.2). 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.

Complexity and overhead. First, Bird-MAC requires the topology information around the neighborhood of each node, because each node determines two critical parameters, wake-up time T_{wake} and nodding interval T_b, which are computed based on the information such as the level in a routing tree and the number of children and siblings. After a routing path is constructed during the init phase, each node is able to obtain the required information by just one message exchange with its parent. Since the routing path construction is the common procedure for most protocols, this overhead is expected to be marginal in Bird-MAC. Once each node obtains this information, a simple local computation is required to calculate two parameters (see Section 4). The second overhead for operating Bird-MAC comes from message exchanges for synchronization and transmission of beacon signals for coordination. Note that we do not specify any particular protocol for synchronization, but provides a framework for synchronization, for which many pair-wise synchronization schemes in literature can be utilized in Bird-MAC. A typical example is the one proposed in [28] which only requires two-way message exchange between a pair of parent and child to synchronize their clocks. Also, the synchronization and coordination are common procedures in other partial synchronous schemes. Although there is an additional overhead in Bird-MAC for a transmission of one beacon signal of the earlier wake-up node, it takes a negligible portion of the activated time of the node. Thus, complexity and overhead of Bird-MAC are comparable with other existing protocols.

3.2 Contention and Reliability Control

In Bird-MAC, similar to conventional wireless system, controlling medium access to avoid and resolve media contention is necessary. In this subsection, we present unique challenges coming from Bird-MAC, and provide their solutions. In addition, we also present a design which significantly reduces contention so that reliability and energy efficiency are improved. These designs are tested by microbenchmark simulation results for a subtree of depth one with one parent and n children.

3.2.1 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. There are two types of collision in Bird-MAC. The first is collision between strobe beacons which occurs during sync phase, and the other is the collision between Msg packets during data phase. We focus on the contentions due to the strobed beacons, since, as we will discuss, there are two unique challenges ***Challenge 1*** and ***Challenge 2*** that do not arise in the conventional contention control.

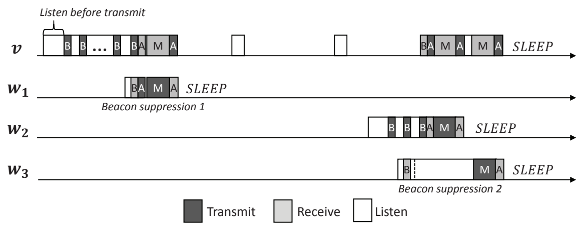


Fig. 7: Beacon suppression. Coordination of one RX and three TXs in Fig. 2. Nodes wake up in the sequence of Sink, N_1, N_2, and N_3.

Challenge 1: 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. 7. We set this sensing time to be 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.

Challenge 2: 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 back-off 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 an inter-beacon period to receive ACKs. If some unexpected signal is detected during the inter-beacon period, then the node considers it as a collision. Once collision is detected, it backs off a random amount of time and restarts transmitting beacon signal from the beginning. In CC2420 of Z1 mote, one back-off time slot is 0.32 msec and maximum slot size is 2^5-1, and the retry (for retransmission) count is limited by 7.

Contention control for Msg packets. 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, conventional CSMA with ACK can be used, where the rules for back-off and retransmission are the same as those of the strobed beacon.

Microbenchmark for contention control. To see the impact of our design of contention control, we compare the performance of the coordination of Bird-MAC with and without inter-beacon sensing by simulation as shown in Fig. 8(a). In each case, one parent and n children nodes coordinate and transfer data packets with 1 day data generation period. Applying inter-beacon sensing reduces the wake-up time of nodes up to 60% when there are 5 children nodes, and achieves 100% delivery ratio which is improved by less contention comparing to Bird-MAC without inter-beacon sensing. The contention control on coordination can save energy as well as enhance the reliability of coordination in sync phase. The contention control becomes more crucial as n grows, because more collision would occur when there are many children.

3.2.2 Beacon Suppression

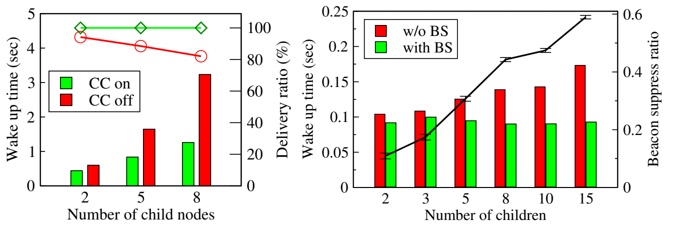
Further energy saving can be achieved if a certain portion of strobed beacons can be suppressed. To this end, we are able to use nodes' ability of overhearing the communications in their neighborhood. Thanks to our MAC-level sensing in the previous section, a node is capable of often decoding overheard 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. 7). Second, when w_3 overhears its sibling w_2's beacon, it keeps listening until the ongoing beacon is terminated. If the ongoing beacon ends with successful message exchange between w_2 and its parent, meaning that its parent is awake, 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. 7). On the other hand, if w_2's ongoing beacon ends without message exchange, inferring that the parent is in the sleep state, and thus w_3 starts nodding to wait for the parent's beacon without transmitting its own 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.

Microbenchmark for beacon suppression. We present a microbenchmark simulation result in Fig. 8(b), demonstrating the degree of beacon suppression with varying number of children n. This result shows that the *beacon suppression ratio*[6] grows with increasing n, reaching up to 60% when $n = 15$. It means that 60% of communications are coordinated without exchanging strobed beacon and this results in much less energy consumption. Thus, consumed energy can be reduced in proportion to *beacon suppression ratio* comparing to Bird-MAC with disabling beacon suppression, where we generate random clock drifts following the distribution of clock drift of Z1 mote (see Fig. 5(a)).

3.2.3 Handling Transmission Failures

Despite the contention control in Section 3.2.1, 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.

6. Total number of beacons with suppression divided by those without suppression.



(a) Impact of contention control (b) Impact of bescon suppression. Line graph represents beacon suppress ratio

Fig. 8: Micro benchmark for *contention control* and *beacon suppression.*

4 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 2) and T_b that corresponds to the nodding interval ((2) in Section 3.1.2). Our derivation is based on the following assumptions.

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 $E[\max_{1\le i\le n} X_i(\tau)] = c\tau\sqrt{\log n}$.[7] Second, we assume that the link is reliable, thereby no link-level transmission failure exists. Bird-MAC includes mechanisms such as pipeline scheduling and beacon suppression, which is highly likely to reduce the probability of collision. In addition to this, the costs for back-off and retransmission due to collisions are much smaller than coordination cost for extremely low data rate applications. 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 a sync phase is randomly chosen.

7. In Gaussian distribution with zero mean, following bounds hold: $\frac{\sigma}{\pi \log 2}\sqrt{\log n} \le E[\max_{0\le i\le n}\{X_i\}] \le \sigma\sqrt{2}\sqrt{\log n}$ [32]. Thus, the distribution in **A1** can represent that of clock drift in Fig. 5. 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. 5(b)).

Decomposition of consumed energy. Let P_{on} (watt) be the amount of energy when in RX mode (i.e., sum of MCU and RF 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 2, 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.

4.1 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 3.1.2) and β is the beacon suppressed ratio (see Section 3.2.2). Theorem 4.1 states which choices of T_b and T_{sync} minimize the total energy consumption under what conditions.

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 is made:

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

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

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

Interpretation. A few interpretations are in order. First, T_{th} is about 1 minute in the subtree with one child in practice, and it is upper-bounded by 3 mins.[8] Thus, optimal choices in (3) of Theorem 4.1 works for a fairly large class of data sensing generation frequency. 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. Third, the effect of early-bird nodding lies in the fact that 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}. It means that infrequent synchronization minimizes the cost for sync phases, which provides an insight that it is recommended to align sync phase with data phase for energy efficiency. In addition to this, early-bird nodding also helps in coordinating communication in a more energy efficient manner, when data packets are generated with a highly long period.

8. 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).

4.2 Proof of Main Theorem

Assumptions. We first present the assumptions of our analysis.

A1. *Clock drift.* Let $X_i(\tau)$ denote the amount of clock drift for node i when time τ has elapsed. It follows a distribution which satisfies following features: (1) zero mean, (2) $E[\max_{1\le i\le n} X_i] = c\tau\sqrt{\log n}$. Using MMSE (Minimum Mean Square Error) fitting, we set c as 3.58×10^{-6} in Z1 mote, which fits well to the clock drfit measurement result (see Figure 5(b)).

A2. *Link-level reliability.* We assume that link is reliable, thereby no link-level transmission failures exist.

A3. *A subtree with* $1:n$. We focus on a subtree of depth one with one parent and n children for analytical tractability.

A4. *Random init.* In an unaligned case, the starting time of sync phase is randomly chosen.

4.2.1 Proof of Theorem 4.1

Our proof strategy is as follows. Recall that two design parameters T_{sync} and T_b affect E, thus $E = E(T_{\text{sync}}, T_b)$ for given T_{data}. We first minimize the coordination costs (i.e., $E_{\text{co\_sync}}, E_{\text{co\_data}}$) over T_b, and express them in terms of T_{sync}, both stated in Lemmas 4.1 and 4.2. Using the results of Lemmas, we find an optimal T_{sync} which minimizes E, in both *aligned* and *unaligned* cases, and conclude that the aligned case with $T_{\text{data}} = T_{\text{sync}}$ is optimal given T_{data}. Finally, we revisit Lemma 4.1 to find the optimal T_b based on the fact that the *aligned* case of $T_{\text{data}} = T_{\text{sync}}$ drives optimality.

Proof of Theorem 4.1. We start by presenting two key lemmas that provide the results of optimizing the coordination cost over T_b. To this end, we let τ denote the time elapsed since the last synchronization, which gives us random clock drift. Suppose that there are n children in the target subtree, and let $\bar{E}_{co}(\tau)$ denote the sum of $n+1$ nodes' expected consumed energy for one coordination.

Lemma 4.1. For given τ and n, the optimal nodding interval $T_b^*(\tau)$ and the optimal coordination cost $\bar{E}_{co}^*(\tau)$ are given by:

$$T_b^*(\tau) = 2\sqrt{\frac{c(n\sqrt{\log 2} + \sqrt{\log{(n+1)}})t_{sl}\tau}{(3n+4)(1-\beta)\gamma}},$$
$$\bar{E}_{co}^*(\tau) = \alpha P_{\text{on}}\sqrt{\tau}, \tag{5}$$

where,

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

From Lemma 4.1, the coordination cost can be expressed as $\alpha P_{\text{on}}\sqrt{\tau}$, based on which we are able to quantify $\bar{E}_{\text{co\_data}}$, i.e., the energy consumed for one coordination of n nodes in a data phase in following lemma.

Lemma 4.2. Given τ, T_{sync} and T_{data}, when the nodding interval T_b is chosen as in (5) of Lemma 4.1, $\bar{E}_{\text{co\_data}}$ is given by:

$$\bar{E}_{\text{co\_data}} = \begin{cases} \frac{2\alpha P_{\text{on}}}{3}\sqrt{T_{\text{sync}}} & \text{if unaligned,} \\ \frac{\alpha P_{\text{on}}}{k}\sum_{j=0}^{k-1}\sqrt{jT_{\text{data}}} & \text{if aligned with } T_{\text{sync}} = kT_{\text{data}}, \end{cases}$$

where k is some positive integer.

Using Lemmas 4.1 and 4.2, we can have: $E_{\text{co\_sync}} = \frac{\bar{E}^*_{co}(T_{\text{sync}})}{T_{\text{sync}}}$ and $E_{\text{co\_data}} = \frac{\bar{E}_{\text{co\_data}}}{T_{\text{data}}}$, which, if substituted into (4), we obtain E in both aligned and unaligned cases as stated in what follows:

(i) Aligned case with $T_{sync} = kT_{data}$:

$$E(k) = P_{\text{on}}\left(\frac{nt_s}{kT_{\text{data}}} + \frac{\alpha}{\sqrt{kT_{\text{data}}}} + \frac{nt_d}{T_{\text{data}}} + \frac{\alpha\sum_{j=0}^{k-1}\sqrt{j}}{k\sqrt{T_{\text{data}}}}\right) \quad (6)$$

(ii) Unaligned case:

$$\begin{aligned} E &= P_{\text{on}}\left(\frac{nt_s}{T_{\text{sync}}} + \frac{\alpha}{\sqrt{T_{\text{sync}}}} + \frac{nt_d}{T_{\text{data}}} + \frac{2\alpha\sqrt{T_{\text{sync}}}}{3T_{\text{data}}}\right) \\ &> P_{\text{on}}\left(\frac{2\sqrt{2}\alpha}{\sqrt{3T_{\text{data}}}} + \frac{nt_d}{T_{\text{data}}}\right) \quad (7) \end{aligned}$$

Note that $E = E(T_{\text{sync}})$ is a function of T_{sync}, and we now optimize E over T_{sync}. In the unaligned case, we find a lower bound of E as (7), and in the aligned case $E(k)$ is increasing over k when $T_{\text{data}} \geq (\frac{nt_s}{\alpha(\sqrt{2}-1)})^2$, indicating that $E(k)$ is minimized when $k = 1$. By comparing (7) and $E(1)$ in (6), we have that when $T_{\text{data}} \geq (\frac{nt_s}{\alpha(\sqrt{2}-1)})^2$, the aligned case with $k = 1$ always consumes less energy. Thus, $T_{\text{sync}} = T_{\text{data}}$ with aligning minimizes energy consumption, and the optimal energy consumption is expressed as:

$$E^* = \frac{P_{\text{on}}}{T_{\text{data}}}\left(\alpha\sqrt{T_{\text{data}}} + n(t_s + t_d)\right). \quad (8)$$

Since sync and data phases are aligned, from Lemma 4.1 the optimal nodding interval (used in sync phase) is derived as:

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

This completes the proof of Theorem 4.1.

4.2.2 Proof of Lemmas

We now provide the proofs of Lemmas 4.1 and 4.2.

Proof of Lemma 4.1. For a given τ, the energy $\bar{E}_{co}(\tau)$ for a coordination of a subtree with a parent and n children can be decomposed into the energies consumed by a parent and n children, each denoted by $\bar{E}_{\text{co\_p}}$ and $\bar{E}_{\text{co\_c}}$, respectively, as follows:

$$\begin{aligned} \bar{E}_{\text{co}} &= n\bar{E}_{\text{co\_c}} + \bar{E}_{\text{co\_p}} \quad (9) \\ &= P_{\text{on}}\left(nE[T_{b\_c}] + n\frac{t_{sl}E[w_c]}{T_b} + E[T_{b\_p}] + \frac{t_{sl}E[w_p]}{T_b}\right), \end{aligned}$$

where T_{b_c} and T_{b_p} are actual transmission durations of strobed beacon for child and parent, respectively. While nodding, child and parent stay awake for $\frac{t_{sl}E[w_c]}{T_b}$ and $\frac{t_{sl}E[w_p]}{T_b}$, respectively, where w_c and w_p denote coordination waiting times, and thus $\frac{E[w_c]}{T_b}$ and $\frac{E[w_p]}{T_b}$ are the expected numbers of nodding in one coordination. Note that a node stays awake for the duration t_{sl}, when it wakes up during nodding. From this, the expected duration of transmitting strobed beacon ($E[T_{b\_c}]$, $E[T_{b\_p}]$) and coordination waiting time ($E[w_c]$, $E[w_p]$) can be obtained as follows:

- *Transmitting strobed beacon:* For a given T_b, the expected duration of actual beacon transmission is $E[T_{b\_p}] = (1-\beta)T_b$ for a parent and $E[T_{b\_c}] = \frac{3}{4}(1-\beta)T_b$ a child, respectively. Since strobed beacon is suppressed with the ratio of β, the node transmits beacon with probability $(1-\beta)$. The strobed beacon of a child can be shrinked down when it is a late bird (whose probability is 1/2). In this case $E[T_{b\_c}|\text{late bird}] = \frac{1}{2}(1-\beta)T_b$, whereas the parent always transmits a full length of a strobed beacon.
- *Waiting with nodding:* Let X_p and $X_{c,i}$ denote the wake up time of parent and child i, respectively, then $w_c = (X_p - X_c)^+$ and $w_p = \max_i\{(X_{c,i} - X_p)^+\}$. From the assumption **A1**, $E[w_c] = c\tau\sqrt{\log 2}$ and $E[w_p] = c\tau\sqrt{\log(n+1)}$.

Therefore, we obtain coordination cost for a child and a parent as stated in what follows:

$$\begin{aligned} \bar{E}_{\text{co\_c}} &= P_{\text{on}}\left(\frac{3}{4}(1-\beta)T_b + \frac{t_{sl}c\tau\sqrt{\log 2}}{T_b}\right), \\ \bar{E}_{\text{co\_p}} &= P_{\text{on}}\left((1-\beta)T_b + \frac{t_{sl}c\tau\sqrt{\log(n+1)}}{T_b}\right). \end{aligned}$$

Substituting $\bar{E}_{co\_c}$ and $\bar{E}_{co\_p}$ into (9), we obtain optimal nodding interval (T_b^*) and minimum energy consumption of a subtree ($\bar{E}^*_{co}$) by differentiating (9) with respect to T_b.

$$\begin{aligned} T_b^* &= 2\sqrt{\frac{c(n\sqrt{\log 2} + \sqrt{\log(n+1)})t_{sl}\tau}{(3n+4)(1-\beta)\gamma}}, \\ \bar{E}^*_{co} &= P_{on}\sqrt{c(n\sqrt{\log 2} + \sqrt{\log(n+1)})(3n+4)(1-\beta)\tau}. \end{aligned}$$

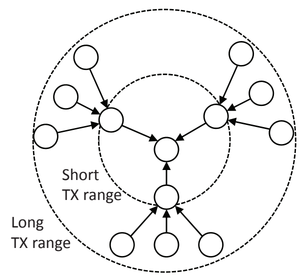
This completes the proof of Lemma 4.1.

Proof of Lemma 4.2. For a given τ, suppose that energy consumed for one coordination is $\bar{E}^*_{co}(\tau) = \alpha P_{\text{on}}\sqrt{\tau}$, from Lemma 4.1. For given T_{sync} and T_{data}, we derive expected consumed energy for one data coordination $E_{\text{co\_data}}$ in both *aligned* and *non-aligned* cases separately as follows.

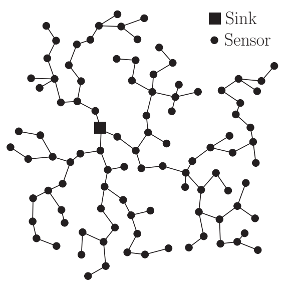
(i) Aligned case with $T_{sync} = kT_{data}$: When the data phase follows right after synch phase, the coordination cost is 0, otherwise there exists coordination cost for each data communication, i.e. coordination cost is $\alpha P_{\text{on}}\sqrt{jT_{\text{data}}}$ with probability $\frac{1}{k}$, where $0 \leq j \leq k-1$. Thus, $\bar{E}_{\text{co\_data}}$ is given by:

$$\bar{E}_{\text{co\_data}} = \frac{\alpha P_{\text{on}}}{k}\sum_{j=0}^{k-1}\sqrt{jT_{\text{data}}}.$$

(ii) Unaligned case: All nodes synchronize their clocks independent of data communication, thus τ varies when the nodes coordinate for data communication. By Assumption **A4**, τ is uniformly



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



(b) RANDOM: a large scale topology with 100 nodes

Fig. 9: Two types of topoloies used in simulation.

distributed in [0, T_{sync}] and coordination cost is $\alpha P_{\text{on}}\sqrt{\tau}$ for given τ. Thus, $\bar{E}_{\text{co\_data}}$ is given as:

$$\bar{E}_{\text{co\_data}} = \int_0^{T_{\text{sync}}} \frac{\alpha P_{\text{on}}\sqrt{\tau}}{T_{\text{sync}}} d\tau = \frac{2\alpha}{3}\sqrt{T_{\text{sync}}}.$$

This completes the proof of Lemma 4.2.

5 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 [24]. The source codes and all the test scripts are available in [25]. 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 suppression. Routing paths are made by RPL routing protocol [27] 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. 5). 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. 13(a). Following the analytical result in Section 4, we choose T_b and T_{sync} with dependence on the given topology and other hardware-dependent values. Other parameters such as the back-off 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.

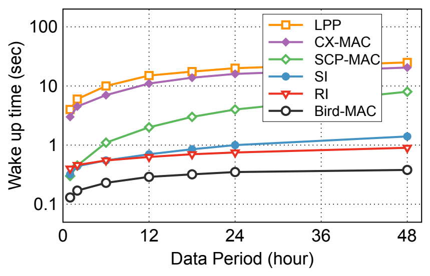
Tested protocols. In simulations, we compare Bird-MAC with other asynchronous and partially synchronous protocols. For asynchronous protocols, we test CXMAC and LPP in Contiki OS. CXMAC is a sender-initiated MAC protocol based on X-MAC [12], and LPP is a receiver-initiated MAC protocol similar to RI-MAC [18]. For partially synchronous protocols, we directly implement SCP-MAC [15] 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 [15] and PW-MAC [21] 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.

5.1 Results: Simulation

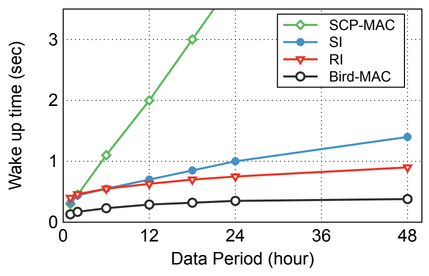
In simulations, we use two types of topologies: (i) B-ARY complete tree with $B = 2, 3, 4, 5$ with height 2 and (ii) random topologies with $5, 10, 20, 50, 100$ nodes, which we denote by RANDOM, as shown in Figs. 9(a) and 9(b). For RANDOM topologies, we run the RPL routing protocol [27] for constructing a routing tree, whereas in B-ARY, the routing tree is set to be the same as the original topology.

(a) *Energy efficiency:* Fig. 10(a) shows the energy efficiency of all tested protocols, where we particularly magnify the results of SCP-MAC, SI, RI, and Bird-MAC in Fig. 10(b) 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. For example, the energy consumption of Bird-MAC is 2.1% of CX-MAC which is typical asynchronous protocol when data period is 24 hours. It is because, as expected, asynchronous protocols are optimized for unpredictable data generations. Fig. 10(b) quantitatively illustrates the energy efficiency gain of the late-bird initiated rationale and the early-bird nodding of Bird-MAC. The gap between SCP-MAC and SI is mainly due to the early-bird nodding idea. RI consumes slightly less than SI, because SI causes more contention while transmitting a long preamble signal. The difference between RI and Bird-MAC comes from the late-bird initiated idea. All of these effect increase as T_{data} grows, and thus energy efficiency gain is 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 with 100 nodes, as shown in Fig. 10(c), 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.

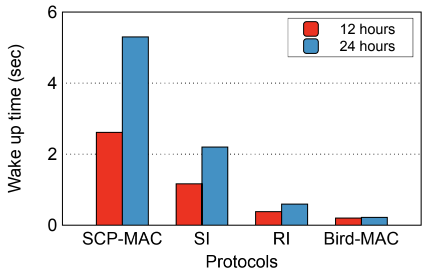
(b) *Impact of environmental changes:* First, Fig. 11(a) shows the results when $T_{\text{data}} = 1$ day, for a varying number of children in B-ARY, where $B = 2, 3, 4, 5$. As discussed in Section 3, depending on those changes, we appropriately set our protocol parameters and factors, e.g., nodding interval T_b and beacon suppression ratio β. We observe that Bird-MAC consumes 26 to 36% energy comparing to RI. Bird-MAC sustains its energy efficiency for varying number



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

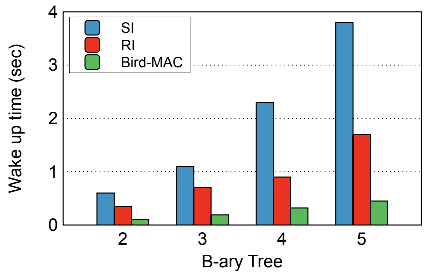


(b) Energy efficiency for varying T_{data} (SCP-MAC, SI, RI, Bird-MAC): `3-ARY`

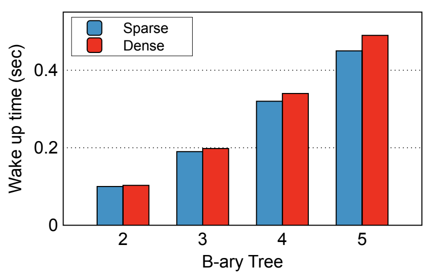


(c) Energy efficiency for `RANDOM` and $T_{\text{data}} =$ 12, 24 hours

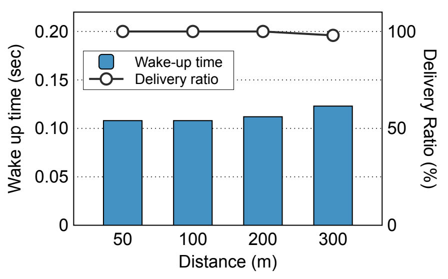
Fig. 10: Energy efficiency of Bird-MAC



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

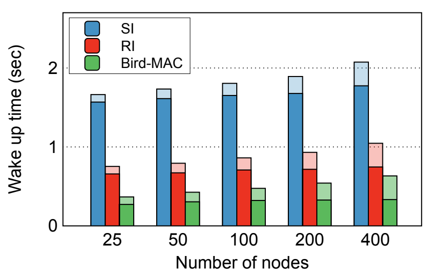


(b) Sparse and dense topology by varying TX ranges for B-ary trees (B=2, 3, 4, 5)

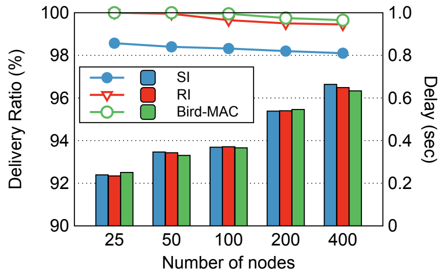


(c) Impact of link quality by varying inter-sensor distance.

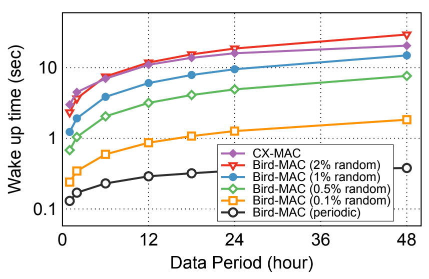
Fig. 11: Impact of environmental change



(a) Energy efficiency with varying number of nodes. Filled and non-filled areas are for sync and data phase, respectively.



(b) Delivery ratio (line graph) and delay (bar graph) for data transmission with varying the number of nodes.



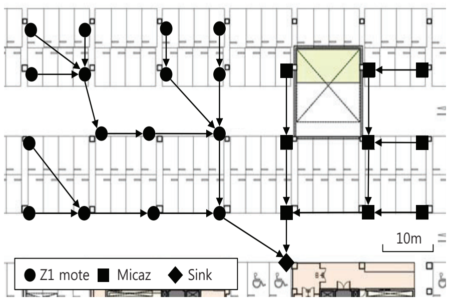
(c) Energy efficiency with varying degree of traffic periodicity

Fig. 12: Evaluation of Bird-MAC's in data phase (a, b). Impact of traffic periodicity on energy efficiency (c).

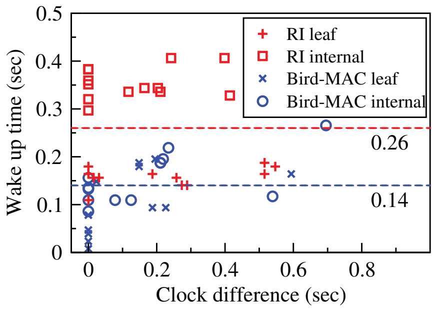
of children, and its gap from other protocols tends to increase thanks to beacon suppression (see Section 3.2.2). Fig. 11(b) compares the energy efficiency of sparse and dense topologies (see Fig. 9(a)), for a varying number of children in `B-ARY`, where B = 2, 3, 4, 5. We observe that the sensor node consumes less energy in sparse topology comparing to dense topology, but their difference is 5% when the number of children is 5 and it becomes smaller when the number of children is less than 5. Thus, choosing parameters considering only $1 : n$ subtree is reasonable. We also 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 3.2.3) is achieved. Fig. 11(c) shows that 100% delivery ratio is achieved up to 200 m distance, and about 99% delivery ratio is achieved for 300 m.

(c) *Scalability:* Fig. 12(a) shows the impact of the number of nodes on energy consumption. We vary the number of sensor nodes, which are deployed with the same density and generate data packet every $T_{\text{data}} = 1$ day. As discussed in Section 3.2, the consumed energy for synchronization is not affected by the number of nodes in the network, but by local environmental factors such as node density in the network. The results also show that the energy consumption for synchronization increases by only 18%, as the number of nodes grows from 25 to 400. In data phase, all the sensor nodes transmit packets whose size is 7 byte and the sink collects all generated packets by relaying, where as the number of nodes grows, the nodes relay more data packets from other nodes. We observe that the energy consumption in data phase increases, as expected, but only sub-linearly.

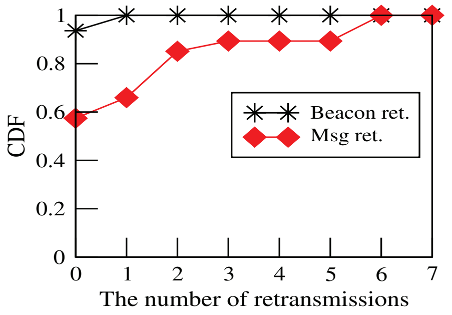
(d) *Data transmission performance:* Fig. 12(b) shows the performance of data transmission, where delivery ratio and delay are measured. The line graphs show that Bird-MAC and RI



(a) Sensor map in the underground parking lot.

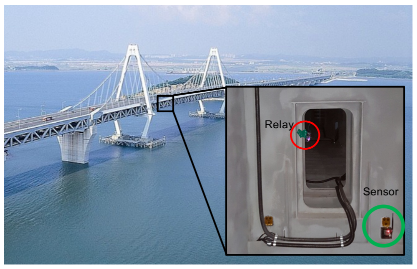


(b) Energy efficiency for `Z1` and `MICAz` motes

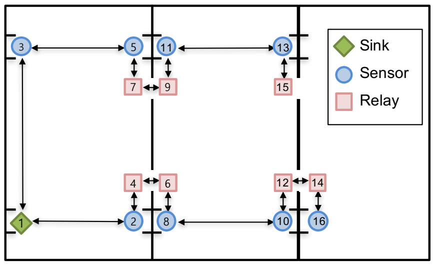


(c) The number of retransmissions of beacons and Msgs

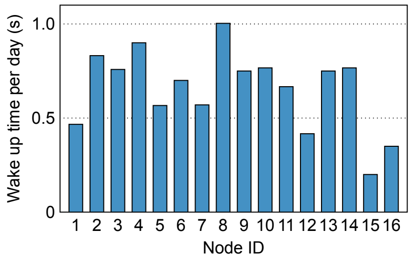
Fig. 13: Experiment result in the underground parking lot.



(a) Sensors are deployed in Yeongjong Grand Bridge



(b) Routing path in the bridge



(c) Energy consumption of sensor nodes

Fig. 14: Experiment setting and result of structural health monitoring application

achieve 100% delivery ratio in the small-scale case, and 99.5% delivery ratio for 400 nodes. Although imperfect, high delivery ratio is achieved thanks to a mechanism of failure handling using retransmission and recovery. Since large contention due to a long preamble signal causes the collision, SI performs much worse than others. Fig. 12(b) also shows the delay performance of Bird-MAC and others. Since the same pipeline scheduling mechanism in Section 2 is applied to all tested protocols, there is only a small difference in delay. A larger number of nodes generates the longer path, and thus the delivery and delay performances degrades as the number of nodes increases.

(e) *Impact of degree of traffic periodicity:* Fig. 12(c) shows the energy efficiency of Bird-MAC when there exists randomness in traffic pattern. There is data period (T) and each node generates one packet in every data period. In periodic application, all nodes generate traffic at the same time. However, in random setting, the data is generated uniformly random within certain range (x% of T), and we call this as quasi-periodic application. As randomness (x) grows, the traffic pattern has strong randomness, so it requires longer wake up time for nodes to complete the communication in data phase, and it results in more energy consumption. When the data period is 2 hours, if the randomness is less than 2% (i.e., 72 sec), Bird-MAC is still more energy-efficient than CX-MAC which is designed for randomly generated traffic without the degradation of delay performance. The range of randomness where Bird-MAC is energy efficient, becomes 1% (i.e., 30 min) as data period grows upto 48 hours.

5.2 Results: Real Experiment

5.2.1 Experiment on Testbed

In the testbed deployed at an underground parking lot of our office building (see Fig. 13(a)), each node reports its sensing data to a sink every 12 hours. We run both Bird-MAC and *RI* (which showed the best performance in simulations) for 7 days for comparison. Fig. 13(b) 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 (0.18 sec on average) is relatively much smaller than the maximum possible clock difference (2.16 sec) which is the major source of RI's high energy consumption. Fig. 13(c) 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 3.2.1 is a good choice at least in our test environment.

5.2.2 Use Case: Bird-MAC on Structural Health Monitoring

A typical example of periodic monitoring application is Structural Health Monitoring (SHM). To see the applicability of Bird-MAC to SHM, we developed the sensor nodes [33], [34] and deployed them on Yeongjong Grand Bridge in South Korea. A sensor node is composed of three modules: (i) *sensing module*, (ii) *data module*, and (iii) *wireless communication module*. The sensing module generates ultrasonic wave and acquires the responses. The data module consists of a data logger which stores the data obtained by the sensing module into the on-board memory and a processor which conducts signal processing with a crack detection algorithm. The wireless communication module which is composed of MSP430 processor and CC2420 communication interface takes charge of wireless data transmission.

We apply Bird-MAC to the wireless communication module in order to collect the diagnosis result periodically, as follows. For most of the time, the sensor node will be in a sleep phase, where the wireless communication module is at low-power sleep mode, and the other modules are all powered down. We align the data and sync phase, as obtained from our analysis, so that right before data generation, all nodes synchronize their clocks to the reference time. Since there is large amount of clock drift for synchronization, the proposed coordination scheme is used in sync phase. After performing crack diagnosis, data is collected by the sink node through wireless channel during data phases.

As presented in Fig. 14(a), 16 sensor nodes are deployed inside the box girder under the bridge. The box girder consists of many rooms and sensor nodes located in different rooms can communicate only through small windows as shown in Fig. 14(a). Thus, we deploy some relay nodes near the windows and this results in the routing path as shown in Fig. 14(b). In this application, each node is required to report its sensing data to a sink every 24 hours. We run this system for two weeks and the sink node successfully collects the sensing data from all sensor nodes. Fig. 14(c) shows the active times of sensor nodes for each day, and the average wake up time for one day is 0.65 sec. In this experiment, we have found that the sensor nodes consume more energy than those from our simulations. This is because the mechanism of handling transmission failures is more often triggered due to bad channel conditions in this experiment. However, we confirm our protocol's applicability in a much more challenging, realistic scenario.

6 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. We proposed a way of communicating in an energy-efficient fashion, but there also exist other important issues such as security and time synchronization, which are beyond of the scope of this paper. We think that the ideas developed for handling those issues are orthogonally applied in Bird-MAC.

Acknowledge

This work was support by institute for Information & Communications Technology Promotion(IITP) grant funded by Korea government(MSIP) (2018-0-0072, Development of an ultra-Low power(500uW) long range radio for ultra-small IoT based on new Two-Tone OOK modulation scheme), and BK21 Plus project of the National Research Foundation of Korea Grant.

References

[1] D. Kim, J. Jung, Y. Koo, and Y. Yi, "Revisiting sensor mac for periodic monitoring: Why should transmitters be early birds?" in *Proc. of IEEE SECON*, 2017.

[2] 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.

[3] 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.

[4] 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.

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

[6] 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.

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

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

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

[10] 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.

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

[12] M. Buettner *et al.*, "X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks," in *Proc. of ACM SenSys*, 2006.

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

[14] 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.

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

[16] 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.

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

[18] 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.

[19] 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.

[20] P. Huang *et al.*, "RC-MAC: A receiver-centric medium access control protocol for wireless sensor networks," in *Proc. of IWQoS*, 2010.

[21] 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.

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

[23] 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.

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

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

[26] 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.

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

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

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

[30] 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.

[31] 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.
[32] G. Kamath, "Bounds on the expectation of the maximum of samples from a Gaussian," http://www.gautamkamath.com/writings/gaussian_max.pdf.
[33] H. Sohn *et al.*, "Self-sufficient and self-contained sensing for local monitoring of in-situ bridge structures," in *European Workshop On Structural Health Monitoring (EWSHM)*, 2016.
[34] P. Liu *et al.*, "Development of a "stick-and-detect" wireless sensor node for fatigue crack detection," *Structural Health Monitoring*, vol. 16, no. 2, pp. 153–163, 2017.

Daewoo Kim received his B.S. in the Department of Electrical Engineering from Yonsei University, South Korea in 2013. He is a doctoral student at the school of Electrical Engineering, KAIST since 2013. His research interests include sensor networks, network economics, fog computing, machine learning and deep reinforcement learning.

Jinhwan Jung received the B.S. degree in the School of Electrical Engineering from Korea University, South Korea in 2014, and he is a doctoral student in the school of Electrical Engineering, KAIST since 2014. His research interests include low power networks, IoT, and applied machine learning in networking.

Yoonpyo Koo received the B.S. degree in electrical engineering and computer science, and the M.S. degree in electrical engineering from KAIST, South Korea, in 2016 and 2019. His research interests include machine learning and drone simulation.

Yung Yi received the B.S. and M.S. degrees from the School of Computer Science and Engineering, Seoul National University, South Korea, in 1997 and 1999, respectively, and the Ph.D. degree from the Department of Electrical and Computer Engineering, The University of Texas at Austin, in 2006. From 2006 to 2008, he was a Post-Doctoral Research Associate with the Department of Electrical Engineering, Princeton University. He is currently an Associate Professor with the Department of Electrical Engineering, KAIST, South Korea. His current research interests include the design and analysis of computer networking and wireless communication systems, especially congestion control, scheduling, and interference management, with applications in wireless ad hoc networks, broadband access networks, economic aspects of communication networks, and green networking systems. He received the best paper awards at the IEEE SECON 2013 and the ACM MOBIHOC 2013, and the IEEE William R. Bennett Award in 2016.