An Opportunistic MAC Protocol for Energy-Efficient Wireless Communication in a Dynamic, Cyclical Channel

Mathew L. Wymore, *Student Member, IEEE*, and Daji Qiao, *Senior Member, IEEE*

***Abstract*—As wind energy continues to expand to new frontiers in terms of the location, number, and size of wind turbines, the industry has begun to seek smarter operations and management solutions. Low-cost wireless sensing nodes could be used to support data-driven techniques for optimizing production and reducing maintenance costs, among other benefits. Wireless instrumentation makes particular sense for wind turbine blades; however, traditional wireless sensor network deployment approaches are not suitable for blades, due to physical constraints and the resulting extremely limited energy supply. Alternatively, a sensor node attached to a rotating blade could opportunistically and efficiently offload its data to a sink node, attached to the turbine tower, as the blade passes the tower. This approach results in a channel with a cyclical signal strength pattern that existing medium access control (MAC) protocols are not designed to handle. We thus present BladeMAC, a MAC-layer protocol that efficiently handles the cyclical channel problem by dynamically identifying duty-cycling opportunities based on received signal strength. In this paper, we describe the details of BladeMAC's design and our implementation and evaluation in Contiki OS and the Cooja simulation tool. We also discuss practical considerations for our deployment scenario.**

***Index Terms*—Radio communication, wind energy, networks, communication channels, energy conservation.**

I. Introduction

RENEWABLE energy, and particularly wind energy, has seen massive growth in the past few decades [1] due to factors such as decreasing costs, new government policies and incentives, and increased concern about fossil fuels and their effects on our planet. Wind turbines are continually growing in size, and they are being deployed in larger numbers and in increasingly remote locations, such as offshore. The challenge of operating and maintaining large fleets of wind turbines is being met more and more by digital, data-driven methods (e.g., [2]). These methods rely in part on sensors deployed on the wind turbine. Wind turbine blades are particularly difficult to instrument, and current methods are costly [3]. For example, fiber optic sensors can be embedded into the composite material of the blades, but this requires expensive hardware and integration into the blade manufacturing process. Non-contact optical sensors can also be used [4], but these methods are too expensive for continuous online monitoring of an entire wind farm.

Alternatively, small-size wireless sensor nodes attached directly to blades could provide a low-cost, flexible data collection platform. Current applications for such a platform include structural health monitoring with an acoustic emissions wireless sensor network (WSN) [5] or a smart sensing skin [6] for wind turbine blades. Section VII-A contains discussion on more potential applications. However, for these applications to become reality, the challenge of powering WSN nodes deployed on wind turbine blades over the multi-decade lifespan of a wind turbine must first be overcome.

Our research addresses this challenge by reducing energy consumption of the nodes. The wireless radio hardware of a node typically dominates the power consumption, so we seek to reduce the amount of time the radio is on, or to decrease the *radio duty cycle* of the nodes. This is a hallmark task of WSN research, which we note is even more critical in a wind turbine blade deployment, because nodes attached to a wind turbine blade must be small, lightweight, and self-sustaining. These nodes cannot have large batteries or energy harvesting devices, and they cannot be regularly accessed for maintenance.

In a data collection WSN, the nodes that produce data are called *sources*, and the node that receives the data is called the *sink*. One possible approach to our scenario would be to place the sink in the hub at the center of the blades, as in Figs. 1a and 1b. A source deployed at the end of a blade could attempt to communicate with the sink in a single hop, as shown in Fig. 1a. However, the single-hop distance could be 50 m or more on modern wind turbines [7], which is not within the communication range of typical radio hardware that meets our size and power constraints [8]. Therefore, the single-hop approach is not practical for our scenario.

Another option is to use relay nodes arranged in a multi-hop configuration along the blade, as shown in Fig. 1b. The source then only needs to transmit data over a short distance to the next relay node. However, since the relay nodes are also

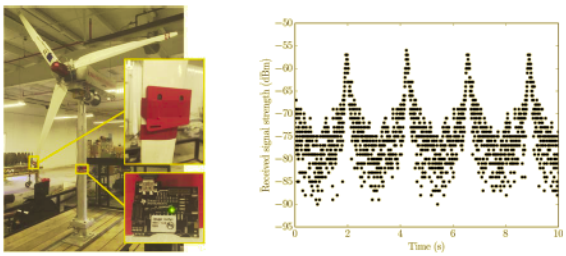
Manuscript received July 11, 2017; revised November 10, 2017 and January 17, 2018; accepted January 24, 2018. Date of publication February 2, 2018; date of current version May 17, 2018. This work was supported by the U.S. National Science Foundation under Grant 1730275, and also Grant 1069283, which supports the activities of the Integrative Graduate Education and Research Traineeship in Wind Energy Science, Engineering and Policy at Iowa State University. Financial support for the meteorological tower construction was provided from the National Science Foundation Iowa EPSCoR Grant 1101284. The associate editor coordinating the review of this paper and approving it for publication was E. Ayanoglu. *(Corresponding author: Mathew L. Wymore.)*

The authors are with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50011 USA (e-mail: mlwymore@iastate.edu; daji@iastate.edu).

Digital Object Identifier 10.1109/TGCN.2018.2801723

(a) Single-hop. (b) Multi-hop. (c) Opportunistic.

Fig. 1. Three approaches to WSN deployment on wind turbine blades. BladeMAC uses the approach shown in (c).



(a) Lab-scale wind turbine. (b) RSS trace.

Fig. 2. A SpectraQuest [10] laboratory-scale wind turbine with 1.4 m blades, shown in (a), was used to observe the cyclical channel phenomenon. The sensor nodes shown in insets are TI CC2650-based SensorTags [9]. The node attached to the blade is encased in a rubber enclosure. An RSS trace gathered using this setup is shown in (b). Each point is an RSS sample taken from a packet sent at a transmission power of −21 dBm while the turbine blades were rotating.

attached to the wind turbine blade, they have the same energy constraints as the source node. While the source node can save energy by reducing the frequency at which it collects and transmits data, the relay nodes must be regularly available to forward data. This communication overhead of relaying makes a multi-hop configuration unsustainable in our scenario.

We therefore propose an alternative approach. Instead of placing the sink at the hub, we attach it to the tower, relaxing its size and energy constraints. A source attached to a blade can then offload its data in an *opportunistic single hop* as the blade rotates past the tower, as shown in Fig. 1c. The short distance between the nodes at this time allows for reliable, low-power transmission. Moreover, this approach is promising in terms of meeting the practical deployment constraints of the system. However, this approach faces a *cyclical channel* problem, in which the received signal strength (RSS) of the link between the nodes varies continually in a periodic pattern as the blades rotate. We observed this cyclical channel using a laboratory-scale wind turbine and Texas Instruments (TI) 2650-based SensorTags [9], as shown in Fig. 2.

On our lab turbine, the nodes stay within range of each other throughout the rotation. But on a large utility-scale turbine, the source will be out of the communication range of the sink for much of the rotation. This uncertainty in the medium creates the need for a specialized medium access control (MAC) protocol. In traditional networking, MAC protocols allow multiple users to access the same channel. Duty-cycled MAC protocols must also arrange a *rendezvous*, a time when both the sender and receiver's radios are on and communication can happen. Normally, a rendezvous is two-way, meaning it involves only the sender and receiver. However, our cyclical channel scenario introduces a new constraint: the rendezvous must occur while the nodes are in communication range, i.e., during a part of the channel cycle when communication is possible. We call this a *three-way rendezvous*, between the sender, receiver, and channel cycle.

Existing duty-cycled MAC protocols are not designed to handle this three-way rendezvous efficiently. For example, when the source attempts to connect with the sink in our scenario, the connection may fail either because the sink's radio is not on, or because the nodes are not in communication range; in the latter case, existing duty-cycled MAC protocols will waste energy while continuing to attempt the connection. An additional challenge in our scenario is that the rotation speed, and thus the channel cycle's period, will change continually as the wind speed varies. These observations motivate the proposal of our application-specific duty-cycled MAC protocol, which we call BladeMAC [11].

In the following section, we examine related work. In Section III, we present possible baseline solutions for our problem scenario. The shortcomings of these baseline solutions motivate the design of BladeMAC, presented in Section IV. In Section V, we describe BladeMAC's method for estimating the length of time available for communication in each channel cycle. In Section VI, we detail our implementation of BladeMAC in Contiki OS [12], our experimental methods, and our evaluation results. We discuss possible applications and a variety of practical considerations for BladeMAC in Section VII. Finally, we conclude the paper in Section VIII.

II. Related Work

Duty-cycled MAC protocols have been a popular research topic among researchers attempting to minimize the energy consumption of WSN nodes [13]. These protocols can be either *synchronous/scheduled* or *asynchronous*. In synchronous/scheduled protocols such as T-MAC [14] or the time-slotted channel hopping (TSCH) protocol of IEEE 802.15.4e [15], nodes track the wakeup schedules of their neighbors and thus know when to wake to listen for packets. These protocols are not suited for our cyclical channel problem because scheduled wakeups would need to consider the channel's "schedule," which, due to rotation speed fluctuations, would be constantly changing.

In asynchronous protocols, rendezvous are not scheduled and occur only as needed. These protocols are generally characterized as either sender-initiated or receiver-initiated. In sender-initiated protocols, such as B-MAC [16], X-MAC [17] and ContikiMAC [18], the sender indicates its desire to send by jamming or strobing the channel until the receiver wakes and responds. In receiver-initiated protocols such as RI-MAC [19], the sender *idly listens* until the receiver announces its availability with a beacon packet, resulting in less channel occupation. Asynchronous protocols exhibit the flexible, dynamic behavior required for our cyclical channel problem; however, existing protocols are not designed to efficiently arrange a three-way rendezvous between the sender, receiver, and channel. For example, if we directly apply RI-MAC to our scenario, the receiver may not always send a beacon during the part of the channel cycle when the link is available,

resulting in excessive idle listening or dropped packets for the sender. We have designed BladeMAC to solve problems such as this.

As part of our solution, BladeMAC uses the RSS and RSS trend to predict future RSS. This takes advantage of the cyber-physical characteristics of the system. Cyber-physical systems [20] have been a popular research topic in recent years. To the best of our knowledge, BladeMAC is the first research to consider the physical rotation of a wind turbine's rotating blades in designing a monitoring system for those blades. Problems similar to our cyclical channel problem have received limited attention in the past, such as for energy-saving in body area sensor networks [21]. Wireless communications in a rotating environment have also been studied in the context of tire pressure monitoring systems [22], which, due to the relaxed deployment constraints, can use standard protocols [23]. Researchers have also explored saving energy through planning (re)transmissions based on channel state and statistics [24], [25]. Our problem is distinct from these problems because our channel is cyclical and the nodes are out of communication range for much of the cycle.

III. Evolution of BladeMAC

In this section, we first present our formal problem statement. Then we invent two baseline solutions for this problem. These solutions represent naive adaptations of existing duty-cycled MAC protocols to the cyclical channel problem. The shortcomings of these solutions motivate the design of BladeMAC and further illustrate our problem and why BladeMAC is needed.

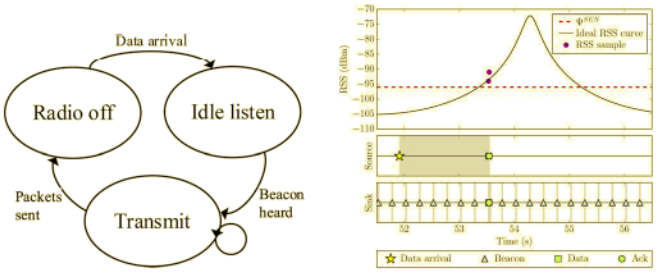
A. Problem Statement

Given a source node deployed on a rotating wind turbine blade and a sink node deployed on the wind turbine tower, our problem is to minimize the energy consumption of the source node (E_{SOURCE}), subject to the constraints of the energy budget of the sink and a one-rotation delay bound (that is, the source should transmit data the next time it passes the tower). Formally, assuming the energy consumption of the node is dominated by the radio hardware, our problem is:

$$\begin{aligned} \min \;& E_{SOURCE} = P_{TX}T_{TX} + P_{RX}T_{RX} + P_{IDLE}T_{IDLE}, \\ s.t.\;& P_{SINK} \le P_{SINK}^{MAX}, \quad D \le T_{\Omega}, \end{aligned} \tag{1}$$

where $P_{TX}/P_{RX}/P_{IDLE}$ is the source's transmit/receive/idle listening power, and $T_{TX}/T_{RX}/T_{IDLE}$ is the source's time spent transmitting/receiving/idle listening. P_{SINK} is the sink's total power consumption, P_{SINK}^{MAX} is the sink's maximum sustainable power consumption based on its energy budget, D is communication delay, and T_{Ω} is the rotation period.

We satisfy the constraints of (1) through protocol design. Assuming P_{TX}, P_{RX}, and P_{IDLE} are equal (essentially true for WSN hardware such as [9]), the minimization in (1) is equivalent to minimizing the radio on-time percentage (duty cycle) of the source node. Because of this, we design to minimize duty cycle and use duty cycle as our evaluation metric in Section VI. An ideal protocol will be able to effectively perform this minimization regardless of factors external to the protocol stack, such as data arrival interval and blade rotation speed.



(a) Diagram of the source's state machine in CC-MAC.

(b) Example trace of CC-MAC, taken from Cooja simulation.

Fig. 3. Overview of CC-MAC. In (b), an RSS plot is shown with an ideal RSS curve, drawn using a log-distance path loss model. Ψ^{SEN} is the sensitivity threshold. Dots show the source's RSS samples gathered from received packets. Event timelines are shown for the source and sink, and the shaded areas represent radio on-time.

B. CC-MAC

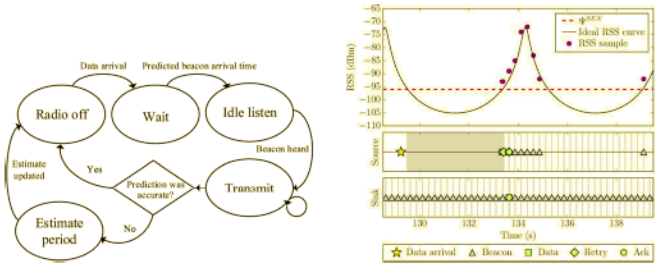
Our first baseline solution is a receiver-initiated scheme, inspired by RI-MAC [19], that we call Cyclical Channel MAC (CC-MAC). A receiver-initiated MAC is chosen because it prevents nodes from occupying the channel when a link is not even available, allowing other nodes to communicate. We note that, for a receiver-initiated MAC, the minimization in (1) can loosely be reduced to the minimization of T_{IDLE}, since T_{TX} and T_{RX} will be similar across protocols. Therefore, the minimization of idle listening serves as the key design goal.

In CC-MAC, the sink sends beacon packets at a fixed interval T_B, which is chosen to be small enough to guarantee that at least one beacon per cycle is sent when the RSS of the link is above the receive sensitivity threshold Ψ^{SEN}. This fixed beaconing interval must also be chosen large enough to satisfy the sink energy budget constraint in (1).

A state machine for the source in CC-MAC is shown in Fig. 3a, with a sample trace of CC-MAC's operation shown in Fig. 3b. The source *sleeps*, meaning it keeps its radio off, until it has data to send (*data arrival*). It then wakes and listens until it hears a beacon, at which point it engages in an exchange of data and acknowledgement (Ack) packets with the sink. Since the source hears and responds to the first beacon possible, and the beacon interval is chosen to be small enough that a beacon can be heard every rotation, CC-MAC satisfies the delay constraint of (1).

We emphasize that, in CC-MAC, the source and sink have separate behaviors. The source does not periodically wake and beacon, meaning that it does not expend energy on communications unless it has data to send. If the sink needs to communicate with the source, it can piggyback command and control information on beacon or Ack packets.

CC-MAC works well enough for infrequent data arrival, but more frequent data leads to a high amount of energy wasted idly listening. Additionally, since CC-MAC transmits data when any beacon is received, an excessive number of retries may be required, as the link may be in a weak state. The pros and cons of CC-MAC are summarized as follows.



(a) Diagram of the source's state machine in CPCC-MAC.

(b) Example trace of CPCC-MAC, taken from Cooja simulation.

Fig. 4. Overview of CPCC-MAC.

- *Pros:* (a) CC-MAC is simple; (b) CC-MAC is sufficient if the interval between data arrivals is very long.
- *Cons:* (a) CC-MAC is sensitive to the data arrival interval; (b) CC-MAC may transmit data while the channel is still in poor condition, requiring excessive retries.

C. CPCC-MAC

To improve on CC-MAC when data arrival is frequent, we now outline a version of CC-MAC with *cycle prediction*, which we call CPCC-MAC (Fig. 4). In CPCC-MAC, the source tracks the cycle period and phase, allowing it to predict when the channel will be present. The source then wakes at this predicted time, as shown in Fig. 4a, and idly listens until a beacon is heard. If the cycle prediction is accurate, the idle listening time is very short. If the prediction is not accurate (the beacon arrives more than T_B after the predicted time), this results in CPCC-MAC redoing the period estimate, as shown in the state diagram.

However, estimating the period is non-trivial. For example, if a data transaction occurs at $t = 0$ and $t = 20$, the period could be any integer division of 20 s, depending on how many cycles elapsed between transactions. The period range may be known, but since the period may be changing even during the estimation process, the problem of period estimation remains difficult. CPCC-MAC therefore performs period estimation by time-stamping *consecutive* cycles. To do this, CPCC-MAC performs an initial data exchange as in CC-MAC. After the exchange, the source wakes up and listens at the known interval T_B until it has heard a beacon in the next cycle. The time elapsed between this beacon and the previous data exchange is used as the period estimate.

An example trace is shown in Fig. 4b. Data arrives around 129 s. The predicted wakeup time is 129.5 s, but this prediction is inaccurate, and the source idly listens for about four seconds (represented by the shaded area on the source's timeline) before a beacon finally arrives. The period estimation process is reinitiated after the Data/Ack exchange, at around 134 s. During this process, the source repeatedly wakes to listen for beacons, until it hears a beacon close to 139 s. This produces a period estimate of around $139 - 133.5 = 5.5$ s. To predict a future wakeup time, CPCC-MAC would use the period estimate to extrapolate out from 133.5 s, the timestamp of the most recent data exchange.

CPCC-MAC is effective if the period is stable and the period estimation is accurate. However, the period estimation will always be slightly inaccurate if the sink's beacon interval is not a divisor of the channel's period. Dropped beacons can cause more severe inaccuracies in the estimation. Additionally, in an application such as a wind turbine, the period is expected to change over time, because the blade rotation speed is related to the wind speed. An inaccurate or outdated period estimate leads to increased idle listening from poorly predicted wakeups and increased overhead from redoing the period estimate. Additionally, poor wakeup predictions can lead to increased delay, if the source wakes too late and misses a communication opportunity, violating the delay constraint in (1). Therefore, CPCC-MAC becomes less efficient as period changes become more frequent and larger in magnitude. The pros and cons of CPCC-MAC are summarized below.

- *Pros:* CPCC-MAC is efficient for frequent data arrival if the period estimation is accurate and the period is stable.
- *Cons:* (a) CPCC-MAC becomes less efficient as the period changes more rapidly and more frequently; (b) CPCC-MAC requires accurate period estimation, which is non-trivial; (c) CPCC-MAC may introduce additional delay due to incorrect period estimation.

IV. BladeMAC

From consideration of our baseline schemes CC-MAC and CPCC-MAC, and their shortcomings, we propose BladeMAC. Built on the foundations of our CC-MAC baseline scheme, BladeMAC defines a set of *opportunities* for either sleeping or transmitting. On each wakeup, BladeMAC uses RSS to identify which opportunity is applicable. Using this framework, BladeMAC makes intelligent decisions about when to sleep and when to transmit while waiting for the channel to become favorable. BladeMAC boasts the following features:

- BladeMAC is efficient at both low and high data arrival rates, alleviating the main drawback of CC-MAC.
- BladeMAC is robust to changes in the cycle period, alleviating the main drawback of CPCC-MAC.
- BladeMAC adapts to channel variation, making it robust to real-world conditions.

This section presents the details of BladeMAC's design. We first present an overview of BladeMAC's operation, then we separately detail the behaviors of the sink and source.

A. Overview

Fig. 5(a) illustrates our cyclical channel problem. As the wind turbine rotates, the distance between the source deployed on the blade and the sink deployed on the tower periodically increases and decreases. Mapping this distance to a theoretical RSS curve produces a cyclical channel. In each cycle, we define two intervals, based on RSS thresholds. The first interval is when RSS values are above a threshold Ψ^{FAV} that is empirically determined to correspond to a favorable channel. This interval is called the *favorable interval* and is of length T_{FAV}. A packet sent by the source during the favorable interval has a high probability to be received by the sink. Therefore, to minimize retransmissions and wasted energy, the source node should attempt to always send data during this interval.

The second interval of interest is when RSS values are above the receive sensitivity level of the radio hardware, Ψ^{SEN}.

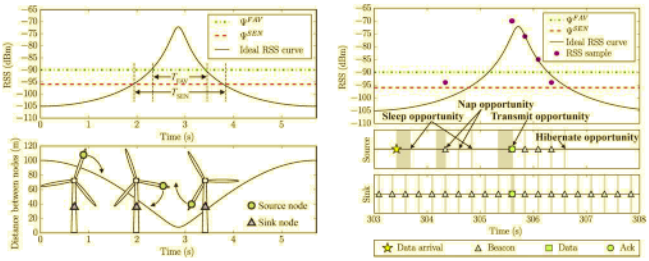


Fig. 5. Overview of BladeMAC. (a) Illustration of the cyclic channel caused by the rotation of the blades. We define the intervals T_{FAV} and T_{SEN} based on Ψ^{FAV}, Ψ^{SEN}, and the rotation speed. (b) Sample trace of BladeMAC's operation, from a Cooja simulation. The *opportunities* used by BladeMAC are labeled.

We call this interval the *sensitivity window* and denote its length T_{SEN}. The sensitivity window encompasses the favorable interval, and the two may vary in relative size with the conditions of the channel and the rotation speed.

In BladeMAC, the source uses RSS values from the beacon and Ack packets it receives to identify the appropriate *opportunity* for different possible actions. In some cases, the appropriate opportunity depends on the RSS *trend*, obtained from the RSS of consecutive packets. With these tools, BladeMAC dynamically makes decisions about when to transmit and when to sleep, allowing it to minimize the idle listening in (1) and also ensure that data is sent when the channel is favorable.

The following sections describe the opportunities and behavior of BladeMAC in detail, but for a high-level introduction, Fig. 5(b) shows an example trace of BladeMAC's operation. As in CC-MAC, the sink beacons at a fixed interval. In this example, the MAC layer of the source is given data to be sent (an event called *data arrival* and marked ☆) while the RSS is below Ψ^{SEN}, at around 303.5 s. The source listens for a period of time but does not hear a beacon packet, resulting in a *sleep opportunity*. At around 304.25 s, the source wakes and listens again. This time it receives a beacon packet (marked △) from the sink, but the packet's RSS is below Ψ^{FAV}, resulting in a *nap opportunity*. The source wakes for the next two beacons, but does not receive them due to packet loss in the still-weak channel. The first missed beacon is a nap opportunity, and the second is a sleep opportunity. Finally, the source receives a beacon with RSS above Ψ^{FAV} at around 305.5 s. In BladeMAC, this is called a *transmit opportunity*, and the source immediately engages in a Data/Ack exchange with the sink until all data is sent. Then, the source listens for additional beacons in order to estimate the sensitivity window size; details of this estimation process are provided in Section V. When the beacon at around 306.5 s is not heard, the source has a *hibernate opportunity*, allowing it to sleep until the next data arrival.

B. Sink Behavior

1) Overview: Due to the asymmetric nature of the source and sink in terms of purpose, physical constraints, and energy supply, the behavior of BladeMAC is different for the two nodes. The behavior of the sink node is straightforward. The sink spends most of its time inactive, with its radio off. Every interval T_{B}, the sink turns its radio on and broadcasts a beacon packet. The sink populates the beacon with the value of T_{B} to inform the source of the beacon interval length. After broadcasting a beacon, the sink listens for up to the *TX/RX turnaround time* plus the time required to send a beacon. This gives the source time to respond with a data packet. If the sink does not receive a data packet during this time, it sleeps until the beginning of the next beacon interval. If the sink does receive a data packet, it responds with an Ack packet and then listens for any additional data packets.

2) Beacon Interval: The choice of T_{B} is key for ensuring that both constraints in (1) are satisfied. T_{B} determines how often the sink must broadcast packets, and is thus the main factor in the sink's power consumption. Therefore, T_{B} must be large enough to satisfy the sink's energy budget. But to satisfy the delay constraint, the beacon interval must be small enough that a beacon can be received by the source in every rotation. Therefore, to strike a balance between energy efficiency for the source and duty cycling for the sink, we choose an upper bound of $T_{\text{B}} \leq 0.5T_{\text{FAV}}$, and we design the source's behavior such that the source cannot sleep through more than half of the favorable interval. With this design, the source is able to hear at least one beacon in each favorable interval, satisfying the delay constraint in (1). If the source cannot sustain a small enough beacon interval, its energy budget must be increased for BladeMAC to be effective.

3) Favorable Interval: The above choice of upper bound for T_{B} means that T_{FAV} must be known. A lower bound for T_{FAV} can be estimated for a particular deployment using the size of the turbine, a distance-based path loss model, and an empirical measure of the favorable threshold Ψ^{FAV} in the target environment. T_{FAV} is equal to the amount of time in each cycle for which the RSS is above Ψ^{FAV} (see Fig. 5(a)).

The favorable interval is also affected by sink height. In this work, we assume that the sink is attached to the tower at the same height as the lowest point of rotation of the source. In theory, positioning the sink higher than this point can result in a longer favorable interval; however, our analysis found that the difference is small (tens of milliseconds) for our scenario parameters. Therefore, we leave further examination of optimal sink height to future work.

C. Source Behavior

The source node's behavior is more complex than the sink. At a high level, the source's behavior is divided into three states: the *hibernation* state, the *wait* state, and the *send* state, as shown in Fig. 6. In the hibernation state, the MAC layer has no data to send, so the source remains inactive. When data arrives, the source enters the wait state. During the wait state, the source attempts to minimize idle listening while waiting for suitable channel conditions that will ensure reliable data transmission. When these conditions are detected, the source enters the send state and attempts to transmit all data packets before the channel degrades. If all data packets are sent, the source transitions to the hibernation state. But if the channel degrades before all the data can be sent, the source must revert to the wait state and wait for the next transmit opportunity. The states are described in detail below.

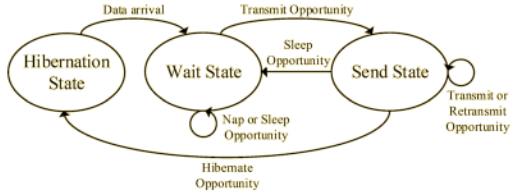


Fig. 6. Diagram of the source's state machine in BladeMAC.

TABLE I
SOURCE'S BEHAVIOR IN THE WAIT STATE

Opportunity	RSS (Ψ_b)	Trend (Ψ_b vs. Ψ_{b-1})	Action
Transmit	$\Psi_b \geq \Psi^{\text{FAV}}$	Any	Enter send state
	$\Psi_b < \Psi^{\text{FAV}}$	Descent ($\Psi_b < \Psi_{b-1}$)	
Nap	$\Psi_b < \Psi^{\text{FAV}}$	Ascent ($\Psi_b \geq \Psi_{b-1}$)	Nap for T_{B}
	$\Psi_b < \Psi^{\text{FAV}}$	$\Psi_{b-1} = \varnothing$	
	$\Psi_b = \varnothing$	$\Psi_{b-1} \neq \varnothing$	
Sleep	$\Psi_b = \varnothing$	$\Psi_{b-1} = \varnothing$	Sleep for $0.5T_{\text{SEN}}$

1) Hibernation State: In the hibernation state, the source keeps its radio off as long as its data queue is empty. Thus, the source remains completely inactive in the network until it has data to send. When data packets are added to the queue, the source enters the wait state. During the wait state, additional packets may be added to the queue.

2) Wait State: In the wait state, the source makes decisions about when to sleep and when to send based on the RSS of the packets it receives (or does not receive) from the sink. Specifically, when entering the wait state, the source wakes and listens for up to T_{B} for a beacon from the sink. If T_{B} is unknown, such as when the source is first turned on, it is set to ∞. When a beacon packet is received, T_{B} is extracted from it and stored for future use.

After either receiving a beacon b (with RSS Ψ_b) or listening for T_{B} and not receiving a beacon ($\Psi_b \triangleq \varnothing$), the source applies the rules summarized in Table I. Ψ_b and the current RSS trend (determined by comparing Ψ_b to Ψ_{b-1}) are used to match one of a number of cases. In the wait state, each case results in one of three *opportunities*: *transmit*, *nap*, or *sleep*.

a) Transmit opportunity: The source transitions into the send state. This opportunity occurs when the channel is favorable, meaning $\Psi_b \geq \Psi^{\text{FAV}}$, or when the channel is degrading, meaning $\Psi_b < \Psi_{b-1}$. In the latter case, this opportunity allows the source to transmit data, even though the channel is not favorable, in order to avoid having to wait for the next transmit opportunity, which would increase delay and energy use.

b) Nap opportunity: The source naps (turns its radio off for a short time) for T_{B} and remains in the wait state. This opportunity occurs when the link is present and the channel is not favorable but getting better, meaning $\Psi_b < \Psi^{\text{FAV}}$ and either $\Psi_b \geq \Psi_{b-1}$ or $\Psi_{b-1} = \varnothing$. Also, a nap opportunity occurs if no beacon is received this wakeup, but the previous beacon was received. This last case accounts for packet loss, acting as a "second chance" mechanism by allowing the source to listen for one more beacon when a beacon is missed unexpectedly.

c) Sleep opportunity: The source sleeps for an extended length of time and remains in the wait state. This opportunity occurs when the source listens for T_{B} but does not hear a beacon, and furthermore did not hear the previous beacon ($\Psi_b = \Psi_{b-1} = \varnothing$). This opportunity arises when the channel is too weak for communication. The length of time to sleep is chosen to be half of the sensitivity window ($0.5T_{\text{SEN}}$). This length guarantees that, even in the worst case where the source goes to sleep just before the first beacon transmission during the sensitivity window, it will still be able to hear the second beacon during the sensitivity window after it wakes up again.

The sleep opportunity requires the source to estimate T_{SEN}, a quantity that may change over time due to external conditions such as changing rotation speed. We present the details of this estimation process in Section V. We specifically design our estimation process to avoid overestimation of T_{SEN}, as overestimation can lead to missed transmit opportunities. Our simulations have shown that, as long as T_{SEN} is not overestimated, performance of BladeMAC remains robust to rotation speed changes. This contrasts sharply with the full period estimation of CPCC-MAC, which is highly sensitive to rotation speed changes.

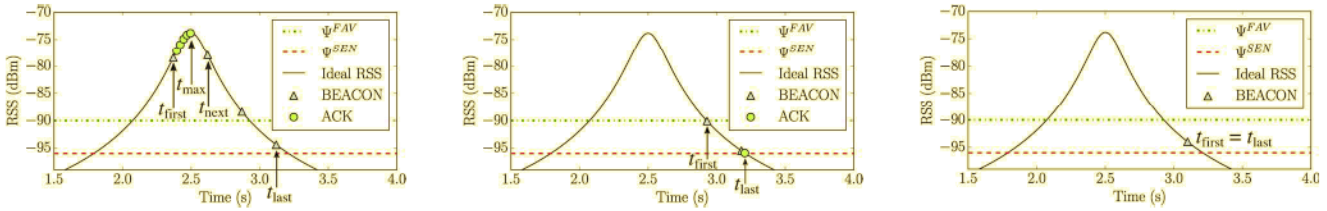
3) Send State: In the send state, the source engages in a Data/Ack exchange with the sink until either the data queue is empty, or the channel degrades and the link disappears, signified by R consecutive transmission failures of a data packet. In our implementation of BladeMAC, we set $R = 3$.

V. SENSITIVITY WINDOW ESTIMATION

As previously discussed, BladeMAC must estimate T_{SEN}, the size of the sensitivity window. BladeMAC uses this estimate to determine how long it can sleep when a sleep opportunity occurs. Therefore, accurately estimating T_{SEN} is key to minimizing BladeMAC's duty cycle. However, BladeMAC should not overestimate T_{SEN}, as this may cause the source to sleep through the sensitivity window, violating the delay constraint in (1). Therefore, we desire an estimate that approaches but does not exceed the actual size of the sensitivity window. To perform T_{SEN} estimation, the source records RSS readings for each packet it receives from the sink. After sending data, the source uses the RSS samples obtained during the sensitivity window to estimate T_{SEN}. However, window estimation is a non-trivial task. The number of acquired RSS samples varies, as does their distribution, and parametric methods such as linear or polynomial regression are unreliable. Instead, our approach is to find a tight lower bound for T_{SEN}, based on characteristics of the RSS samples collected.

A. Extra Beacons

To provide consistency in the RSS samples, BladeMAC allows the source to listen for extra beacons after a transmit opportunity. This behavior is shown in Fig. 5b. After the data is sent at around 305.5 s, the source wakes up every beacon interval to gather another RSS sample from a beacon. At around 306.5 s, the source does not hear an expected beacon, and has no data to send, so it enters the hibernation state. As shown in the figure, this behavior provides an RSS sample close to the edge of the sensitivity window. We denote the timestamp of this sample, the last received in this sensitivity window, as t_{last}. The consistency of this sample results in better T_{SEN} estimation. The wakeups required to gather this



(a) An example "peak" case, with a data group as the peak. $\hat{T}_{SEN} = \max(2 * (t_{last} - t_{next}), (t_{last} - t_{first}))$.

(b) An example "no peak" case. In this case, we use $\hat{T}_{SEN} = 2(t_{last} - t_{first})$.

(c) An example "single packet" case. In this case, we make no estimate and continue to use our previous $\hat{T}_{SEN}$.

Fig. 7. Illustrations of the sensitivity window estimation cases and methods. Each figure shows a set of typical RSS samples for that case. The time between some packets has been exaggerated for clarity.

sample are very short, so the cost is only slightly increased energy consumption, resulting in an overall system gain.

B. Estimation Cases and Methods

The estimation of T_{SEN} is performed during the hibernation state following the sensitivity window in which RSS samples were gathered. The estimation only uses RSS samples since the prior hibernation state or sleep opportunity, guaranteeing that all samples were taken during the same sensitivity window. In addition to t_{last}, we define t_{first} as the timestamp of the first RSS sample in this sensitivity window, t_{max} as the timestamp of the RSS sample with the largest RSS value, and t_{next} as the timestamp of the beacon sample immediately following the t_{max} sample. We do not consider Acks for t_{next} because Ack packets are too closely spaced in time to reliably indicate trends in RSS. We use $\hat{T}_{SEN}$ to denote our estimation of the sensitivity window, and we define three different cases for our estimation method, detailed below.

1) Peak: This case, shown in Fig. 7a, occurs when the maximum RSS sample is greater than Ψ^{FAV}. In this case, we conservatively reason that the peak of the RSS curve must have occurred before t_{next}. We assume that the left side of the ideal RSS curve is the same as the right. Therefore, we choose $2 * (t_{last} - t_{next})$ as our estimate for the sensitivity window in this case. Additionally, as a safeguard against extreme amounts of RSS variability, we check the estimate against $(t_{last} - t_{first})$, a known lower bound for the sensitivity window. This lower bound is used if it is larger than the estimate from above, producing a final estimate as follows:

$$\hat{T}_{SEN} = \max(2 * (t_{last} - t_{next}), (t_{last} - t_{first})). \quad (2)$$

2) No Peak: This case, shown in Fig. 7b, occurs when no RSS samples are greater than Ψ^{FAV}. In this case, based on the defined behavior of the source and the fact that all RSS samples are from the same sensitivity window, we reason that all samples are on one side of the RSS peak. We therefore choose $\hat{T}_{SEN} = 2 * (t_{last} - t_{first})$.

3) Single Packet: This case, shown in Fig. 7c, occurs when the source has only one RSS sample. In this case, $t_{last} = t_{first}$ and no window estimation is possible, so the current estimate for T_{SEN} remains unchanged.

In BladeMAC, to estimate T_{SEN}, the source first identifies the case using the RSS samples collected, and then it calculates the estimate using the corresponding equation for $\hat{T}_{SEN}$.

C. Performance Evaluation

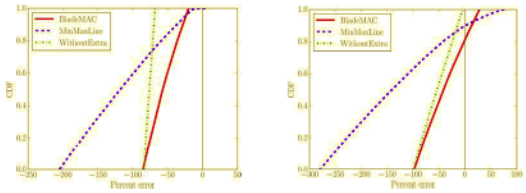
We used Python simulations to evaluate BladeMAC's T_{SEN} estimation method and the effectiveness of using extra beacons to obtain more RSS samples. We compare to a method we call MinMaxLine, which attempts to extrapolate the RSS trend, an approach that seems intuitive but which we have found to be unreliable. MinMaxLine draws a line (in the time-RSS plane) between t_{max} and the sample point with minimum RSS. The method then calculates the intersection of this line with Ψ^{SEN}, which we label t_{Ψ}^{SEN}. The window estimate for MinMaxLine is then $\hat{T}_{SEN} = 2 * (t_{\Psi}^{SEN} - t_{max})$.

We simulated BladeMAC's method with extra beacons (BladeMAC), MinMaxLine with extra beacons (MinMaxLine), and BladeMAC's method without extra beacons (WithoutExtra). In WithoutExtra, the estimation method is similar to BladeMAC's method outlined above, but the rule for the peak case is adapted to $\hat{T}_{SEN} = \max(2 * (t_{prev} - t_{first}), (t_{last} - t_{first}))$, where t_{prev} is the beacon sample preceding the t_{max} sample. In the simulations, RSS values are generated based on a log-normal shadowing model [26] with a standard deviation of σ. The distances between the nodes are calculated to simulate a node rotating at 12 RPM at radius $r = 50$ m, with a clearance of 8 m at the bottom of the rotation. These numbers are intended to represent a node attached to the end of a blade of a large wind turbine rotating at rated speed [7]. In each trial, we choose a random time for data arrival and apply BladeMAC's rules to collect RSS samples, which are used to calculate $\hat{T}_{SEN}$.

Fig. 8 plots CDFs (from 30,000 trials) of the percent error, which is calculated as $100 * (\hat{T}_{SEN} - T_{SEN})/T_{SEN}$. A vertical dividing line is drawn at zero percent error. This represents the ideal curve; negative percent error (to the left of the zero line) indicates an underestimate, and positive percent error (to the right of the zero line) indicates an overestimate. Since underestimating is better than overestimating, we prefer curves that approach the zero line from the left, but do not cross it.

Results are shown for channel parameters $\sigma = 0$ dB and $\sigma = 3$ dB. BladeMAC performs well with $\sigma = 0$ dB, with most estimates being relatively close to the actual value, and no overestimates. MinMaxLine performs worse, underestimating to a much larger extent. The effect of extra beacons is also noticeable, with BladeMAC generally coming much closer to the actual value than WithoutExtra.

The amount of channel variation clearly affects the results of all methods, but the effect is less noticeable for BladeMAC



(a) CDF with $\sigma = 0$ dB. (b) CDF with $\sigma = 3$ dB.

Fig. 8. Performance evaluation of the proposed sensitivity window estimation method. CDFs of the percent error are shown, with negative percent error indicating an underestimate and positive indicating an overestimate.

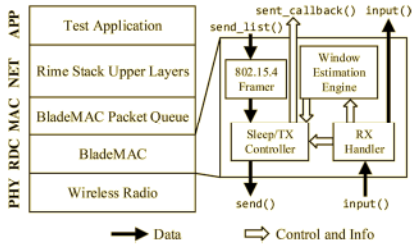


Fig. 9. BladeMAC as it fits into the Contiki network stack.

and WithoutExtra. At $\sigma = 3$ dB, the curves of all methods have reduced slopes, increasing the range of estimation errors and causing some amount of overestimation. BladeMAC overestimates more than WithoutExtra, since packets outside the theoretical sensitivity window are more likely to be heard with the extra beacons mechanism due to its deliberate attempt to collect a sample at the end of the window.

D. Summary

Overall, BladeMAC's estimation method generally provides a reasonable estimate of T_{SEN}. To smooth variations, BladeMAC uses a simple moving average to update $\hat{T}_{\text{SEN}}$. We note that, in addition to the variations in each estimation attempt, T_{SEN} itself may change over time because of fluctuating channel conditions due to external forces. Weather could effectively shift the ideal RSS curve up or down. A change in the channel cycle period would cause T_{SEN} to stretch or contract. However, any change in T_{SEN} is small compared to the change in the period, and unlikely to cause BladeMAC to miss a sensitivity window. BladeMAC is thus essentially agnostic to the cycle period, as we will demonstrate in Section VI using evaluation results.

VI. Evaluation

In this section, we present our evaluation methods and results for BladeMAC. We use simulations in Cooja [27], Contiki OS's [12] simulation tool, for our evaluation. We first discuss our implementation of BladeMAC and our simulation methods. Then, we evaluate BladeMAC against our CC-MAC and CPCC-MAC baseline protocols, at different rotation speeds, different data arrival intervals with a static rotation speed, and different amounts of rotation speed variation. We also present a day-long trace of BladeMAC's performance based on real wind speed data, and a trace of BladeMAC's window estimation process.

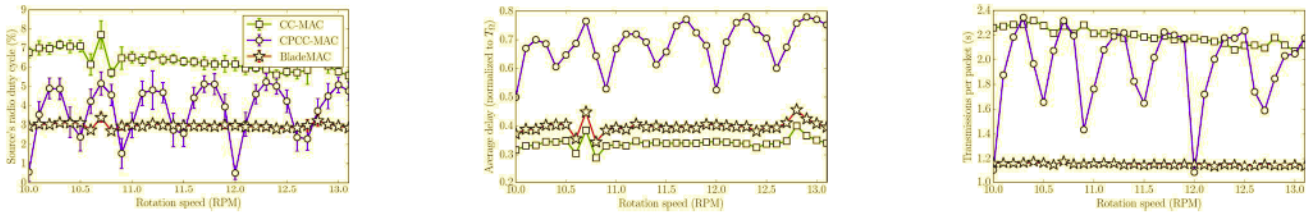
A. Method

We implemented BladeMAC by inserting it into Contiki's Rime network stack [28], as shown in Fig. 9. BladeMAC was implemented at the radio duty-cycling (RDC) layer, which interfaces with the radio hardware. BladeMAC uses Contiki's 802.15.4 packet framer to build 802.15.4-compliant packets. We implemented a packet queuing mechanism at the MAC layer, which interfaces the RDC layer with the upper layers. Using this framework, BladeMAC enters the wait state when the first packet arrives. If additional packets arrive while waiting for a transmit opportunity, these packets are added to the queue, and all queued packets are sent during the Data/Ack exchange. As comparison points for BladeMAC, we implemented our CC-MAC and CPCC-MAC baseline schemes, described in Section III, under the same framework.

Due to the availability of information, we chose to model the National Renewable Energy Laboratory (NREL) 5 MW reference wind turbine [7]. We used Cooja, Contiki OS's simulation tool, for our evaluations. To produce the cyclical channel phenomenon in Cooja, we used the Mobility Cooja plugin [29] to simulate blade rotation with a 50 m radius. We implemented the popular lognormal shadowing propagation model [26] to obtain distance-based RSS. We used a path loss exponent of 3.0 and a standard deviation of 3.0 dB for the random component, values that align with the measurements shown in Fig. 2b. We use the random component as a simple statistical method of encapsulating the Doppler effect (which is minimal in this case [26]), polarization response [22], and other random effects such as hardware noise. To translate RSS into probability of reception, we set a static noise floor of -100 dBm and used an empirically-derived function from TOSSIM [30] to obtain the packet reception rate (PRR) from the signal-to-noise ratio (SNR).

From the TOSSIM function, we obtained a favorable threshold of $\Psi^{\text{FAV}} = -90$ dBm, which yields very high (i.e., $> 99\%$) PRR with the -100 dBm noise floor. Using the lognormal shadowing propagation model from above, we calculated that T_{FAV} is around 850 ms for our simulated turbine rotating at 12.1 RPM, its rated speed [7]. We chose a beacon interval of $T_{\text{B}} = 250$ ms for our evaluation, resulting in $T_{\text{B}} \approx 0.3T_{\text{FAV}}$, which satisfies our requirement that $T_{\text{B}} \leq T_{\text{FAV}}/2$. We also used simulations to verify this upper bound requirement for the beacon interval; as expected, a beacon interval larger than $T_{\text{FAV}}/2$ produces a large drop in performance, and smaller beacon intervals lead to a lower duty cycle for the source. Therefore, in practice, the smallest beacon interval that is sustainable by the sink should be chosen; if the sink cannot sustain $T_{\text{B}} \leq T_{\text{FAV}}/2$, then its energy budget must be increased before BladeMAC can be effective.

For evaluation metrics, we use duty cycle as a parameter-independent proxy for energy consumption (see Section III-A for how energy consumption relates to duty cycle). We also evaluate communication delay and the number of transmissions per packet. In the simulation's application layer, the source queues a small data packet every data arrival interval, with a small amount of random variation. Each simulation runs until 250 packets are successfully sent. The aggregate results



(a) Average duty cycle of source. (b) Average delay per packet. (c) Transmissions per packet.

Fig. 10. Evaluation for different static rotation speeds, with a 28 s data arrival interval. Error bars in (a) show a 95% confidence interval.

shown here are averaged over at least 50 simulations with different random seeds. Error bars on the plots for duty cycle show 95% confidence intervals, calculated as ± 1.96 times the sample standard deviation. Error bars are not shown for delay because the delay naturally varies depending on the phase of the channel cycle when data arrives, so the confidence intervals are not as meaningful and make the plots difficult to read.

B. Rotation Speed

We evaluated the protocols with static rotation speeds of 10.0–13.1 RPM, a subset of the operating range of the modeled wind turbine [7], with a data arrival interval of 28 s. This interval was chosen because at 12.1 RPM, the rated speed of the wind turbine and the default rotation speed for our simulations, BladeMAC and CPCC-MAC have similar performance in terms of duty cycle. The results are shown in Fig. 10.

For all metrics, CPCC-MAC shows significant sensitivity to rotation speed, due to an interaction of the rotation period and the beacon interval that is aggravated by CPCC-MAC's reliance on period estimation. If the rotation period is an even multiple of the beacon interval, CPCC-MAC can acquire a perfect estimate of the rotation period; otherwise, its estimate will have some error, which results in varying levels of performance. BladeMAC and CC-MAC are much less sensitive to rotation speed. BladeMAC's duty cycle, shown in Fig. 10a, is nearly constant with rotation speed, and is generally the best of the three protocols. CC-MAC's duty cycle decreases with rotation speed because the quicker rotation means it listens for less time before a beacon is heard. BladeMAC has the least amount of variation across simulations, showing very tight confidence intervals (mostly hidden by the plot markers).

Fig. 10b shows delay normalized to the period of rotation. BladeMAC and CC-MAC display a constant trend with small fluctuations due to the rotation period's interaction with the data arrival interval. CC-MAC has the shortest delay because it always responds to the first possible beacon, while BladeMAC's slightly higher delay is the price of its much lower duty cycle. CPCC-MAC's normalized delay shows a slowly increasing trend with rotation speed. This is because CPCC-MAC's delay depends on the error of its period estimation process, and the same amount of error becomes relatively larger at a faster rotation speed.

Fig. 10c shows transmissions per packet, which is constant for BladeMAC, and is very low due to BladeMAC's deliberate attempts to send when the channel is strongest. For CC-MAC, transmissions per packet decreases with rotation speed because a faster speed means the channel spends less time in a transitional state, so a response to the first beacon is more likely to be heard. CPCC-MAC's transmissions per packet shows high sensitivity to rotation speed.

C. Data Arrival Interval

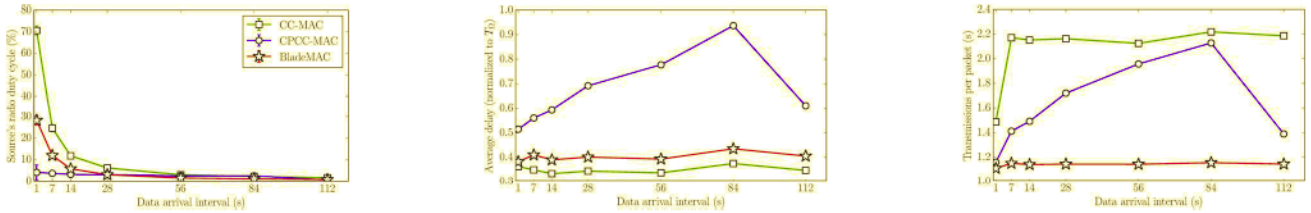
We evaluated the protocols at different data arrival intervals with a static rotation speed of 12.1 RPM. The results are plotted in Fig. 11. In terms of duty cycle, CC-MAC performs poorly at small data arrival intervals, where CPCC-MAC excels. BladeMAC performs the best for data arrival intervals larger than 28 s. Above this point, CPCC-MAC's period estimation is not accurate enough to be worth the overhead.

CPCC-MAC shows predictable performance at lower data arrival intervals, but above 56 s, its performance becomes unpredictable. This is due to an interaction between the rotation period, the period estimation error, and the data arrival interval. If the data arrival interval is long enough that the wakeup prediction is off by an entire period, then the prediction becomes relatively accurate again.

D. Dynamic Rotation Speed

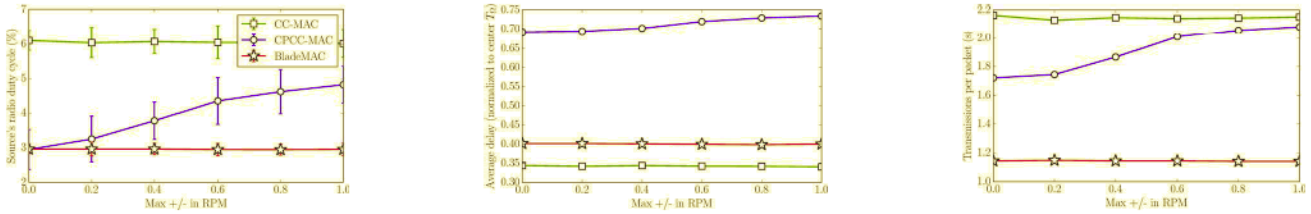
To evaluate BladeMAC's resilience to changing rotation speeds, simulations were run with a dynamic rotation speed that imitates the behavior of the modeled turbine running at rated speed in a turbulent wind flow. The results are shown in Fig. 12. To model the dynamic rotation speed, we first used NREL's FAST wind turbine simulation software [31] to obtain traces of rotation speed for the 5 MW reference turbine. A sample trace is shown in Fig. 13(a). We observed that the rotation speed is characterized by random oscillations around the rated speed. For our Cooja simulations, we generated rotation speed traces to mimic this behavior. In these traces, the rotation speed fluctuates between set points that are chosen uniformly from the range shown on the x-axis, centered at 12.1 RPM. A new set point is reached every 20 s. The transition between set points is divided into discrete intermediate points with a resolution of 0.01 RPM. We generated ten random traces for each fluctuation range, and ran at least ten simulations with different random seeds on each trace.

As expected, CPCC-MAC and BladeMAC have a similar duty cycle at zero variation (a constant speed of 12.1 RPM, which matches the previous figures). However, BladeMAC has the advantage with any amount of variation, even a range of ± 0.2 RPM, which is a variation of less than 2% of the



(a) Average duty cycle of source. (b) Average delay per packet. (c) Transmissions per packet.

Fig. 11. Evaluation for different data arrival intervals, with a static 12.1 RPM rotation speed. Error bars in (a) show a 95% confidence interval, but are too tight to be visible on most points.



(a) Average duty cycle of source. (b) Average delay per packet. (c) Transmissions per packet.

Fig. 12. Evaluation with a 28 s data arrival interval and dynamic rotation speed. The x-axis is the range of speeds allowed, in an interval centered on 12.1 RPM. Error bars in (a) show a 95% confidence interval.

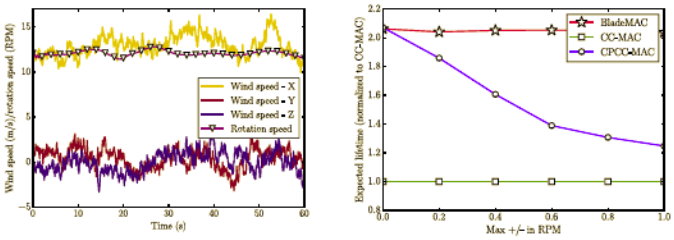


Fig. 13. Additional figures for dynamic rotation speed. (a) Sample trace of rotation speed from NREL's FAST wind turbine simulation software [31]. (b) Expected source node lifetime, normalized to CC-MAC.

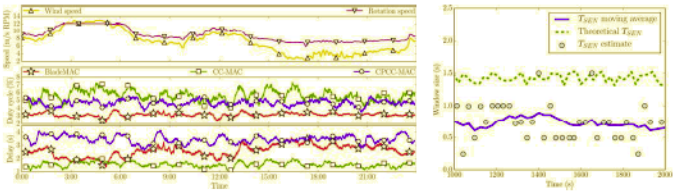


Fig. 14. Trace studies. (a) Trace of duty cycle and delay performance over a full day. Wind speeds are from real measurements at 80 m above the ground. Rotation speeds are from NREL FAST simulations. Duty cycle and delay are shown as a moving average with a 60-packet (about half an hour) window. (b) Trace of the sensitivity window estimation process of BladeMAC, with a data arrival interval of 28 s and a dynamic rotation speed of 12.1 ± 1.0 RPM.

rotational speed. Overall, BladeMAC is shown to be insensitive to rotation speed variation in all metrics, while CPCC-MAC's reliance on period estimation causes its performance to suffer in these dynamic scenarios. Since CC-MAC does not track the rotational speed in any way, its performance is consistent.

Since this evaluation scenario is the most realistic in practice, we provide Fig. 13(b) to contextualize the energy results. The figure shows expected energy lifetime of the source, normalized to the expected lifetime of the source for CC-MAC. This is a simple manipulation of the duty cycle, intended to illustrate the difference a few percentage points of duty cycle can make. We expect BladeMAC to have around twice the lifetime of CC-MAC, regardless of the variations in rotation speed. From another perspective, we expect that BladeMAC can sustain operations with about half the energy harvesting capabilities needed by CC-MAC.

E. Trace-Based Simulation

Fig. 14(a) shows a sample trace of performance of the protocols over the course of a day, showing the effect of the large-scale wind speed changes that happen over long periods of time. For this figure, we obtained one day's worth of 5-minute average wind speeds collected at a height of 80 m by a meteorological tower in Iowa. These wind speeds were fed into NREL's FAST wind turbine simulation software, producing the rotation speeds shown. These rotation speeds were used in the Cooja simulation, with the data arrival interval set to 28 s.

The figure shows that BladeMAC is the most robust to large-scale wind speed changes, maintaining the lowest duty cycle of the protocols. CPCC-MAC does not match BladeMAC even when the rotation speed is relatively (but not perfectly) steady between 3:00 and 6:00. Additionally, with CPCC-MAC's period estimation, the changes in rotation speed regularly result in having to wait for the following rotation to send a packet, which is reflected in CPCC-MAC's high delay. BladeMAC's delay is inversely related to the rotation speed, because a faster rotation speed means less time to wait for a favorable interval.

F. Estimation Process

Finally, in Fig. 14(b), we present a sample trace of the sensitivity window estimation process of BladeMAC. This trace was taken from a simulation with dynamic rotation speed with

± 1.0 RPM variation. The figure shows BladeMAC's sensitivity window estimation, including the individual estimates, the moving average that is used in practice, and the actual theoretical value of T_{SEN}, based on the channel model and the instantaneous rotation speed. As expected, most estimates are close to the theoretical value, and very few are larger than the theoretical value. The moving average estimate remains around 700 ms below the theoretical value. Therefore, to further improve the energy savings of BladeMAC, a less conservative T_{SEN} estimation process could be developed and used in practice.

VII. Discussion

A. Potential Applications of BladeMAC

BladeMAC is intended for use with any sensing application that can tolerate a small amount of communication delay (on the order of one period of rotation). BladeMAC is designed to be efficient for a wide range of data arrival intervals. Since BladeMAC imposes no communication overhead on the source node until data is ready to transmit, and since BladeMAC is designed to transmit packets in bursts, an application running over BladeMAC should aggregate as many data readings as possible before handing them off to the network stack.

BladeMAC's design is purposely decoupled from the type of sensor hardware on the source node and the type of data being gathered; however, we here offer some general ideas on uses for WSN nodes deployed on wind turbine blades.

1) Damage Detection: The source node's sensors could measure damage indicators, which could be tracked over time or compared to indicators from similar turbines in the wind farm. Example sensors include accelerometers (for vibration-based damage detection, e.g., [32]), strain sensors, or even a smart sensing skin [6]. Acoustic emissions monitoring of wind turbine blades with WSNs have also been proposed [5].

2) Advanced Control: The source node's sensors could provide feedback for an advanced control scheme, such as an independent blade pitching scheme designed to minimize stresses on the blades [33].

3) Blade Model Validation: The source node's sensors could measure quantities of interest to researchers, such as stress hotspots throughout the blade's rotation, that could be used to validate models used in blade design.

4) Ice Detection: The source node's sensors could provide early warning of ice buildup on the wind turbine's blades, activating control adjustments or a de-icing solution.

B. Practical Considerations for BladeMAC

1) Blade Pitch: Modern utility-scale wind turbines can adjust the pitch of their blades, which will change the orientation of the source node relative to the sink. This will have minimal effect on BladeMAC as long as the RSS readings from the channel still peak above Ψ^{FAV}.

2) Wind Turbine Yaw: Modern utility-scale wind turbines point themselves (yaw) into the wind during operation, meaning that the sink will not always be on the side of the tower directly facing the rotor blades. As with pitch, BladeMAC will still function as long as the RSS readings from the channel peak above Ψ^{FAV}. However, path loss through the hollow steel tower may be severe. This problem can be alleviated by deploying two sinks, on opposite sides of the tower.

3) Parked State: Wind turbines are sometimes "parked," or non-operational, due to low wind speeds, maintenance, or curtailment. During these times, the blades either do not rotate or only rotate idly. If the blades stop while the source and sink are not within communication range of each other, BladeMAC will not be able to function, and could waste energy attempting to communicate. A cyber-physical approach to this problem could be to allow the source node to use data from outside of the network stack, such as from an accelerometer, to detect this condition and suspend BladeMAC. If this approach is not possible, we propose an optional *recovery mode* for BladeMAC. BladeMAC enters recovery mode when it is attempting to send data but has not heard any beacons from the sink in $kT_{\Omega}^{\max}$, where $T_{\Omega}^{\max}$ is the turbine's maximum rotation period and k is a multiplier (e.g., $k = 3$) that represents how quickly BladeMAC resorts to recovery mode. In recovery mode, the source node 1) sets the sensitivity window estimate to a minimum, $\hat{T}_{\text{SEN}} = 2T_{\text{B}}$, to ensure that beacons are not being missed due to poor window estimation, and 2) begins an on/off cycle in which it first listens for beacons for $T_{\Omega}^{\max}$, using BladeMAC's normal rules, and then suspends BladeMAC's rules and sleeps for $mT_{\Omega}^{\max}$, before repeating the cycle.

Here, m is a multiplier that determines how aggressive the source is in trying to reestablish communication, with the tradeoff being that a smaller m leads to more energy consumption. One possible value for m is $m = \hat{T}_{\text{SEN}}'/2T_{\text{B}}$, where $\hat{T}_{\text{SEN}}'$ is the sensitivity window estimate the source had before entering recovery mode. This choice for m gives the source roughly the same energy consumption in recovery mode as it had during the wait state prior to entering recovery mode.

4) Channel Contention: Multiple source nodes on the same blade may need to communicate with the same sink. In this case, BladeMAC can be extended with a method for resolving contention. One option is retransmission with a backoff window, as in RI-MAC [19]. Since the nodes on the blades are not mobile relative to each other, the sink could also assign a permanent backoff time to each source.

5) Node Packaging: All nodes must be packaged to withstand severe weather, and may need special protection for lightning strikes. Source nodes must have an aerodynamic profile or be embedded into the composite material of the blade. Nodes could also be placed inside hollow blades, but this would limit access and energy harvesting options and increase path loss. Regardless of placement, source nodes would likely need to be bonded to the blades or potentially covered with an extra layer of blade material. The sink poses less of a challenge due to relaxed size requirements. Solar panels, or even micro wind turbines, could provide energy harvesting for the sink. Magnets could be used to attach the sink to a steel wind turbine tower, possibly using a drone for deployment and wireless battery charging.

VIII. Conclusion

We have presented BladeMAC, a radio duty-cycling MAC protocol designed specifically for wireless sensor nodes deployed on rotating wind turbine blades. BladeMAC addresses the cyclical channel problem created by this

scenario, and we showed through Cooja simulations that BladeMAC's approach is effective regardless of rotation speed, data arrival interval, and rotation speed variation. We have also discussed a variety of practical applications and considerations for BladeMAC.

BladeMAC represents a practical solution for a wind turbine blade deployment using conventional WSN hardware. Possible future work is to investigate alternative approaches using emerging technologies. For example, the source node could use a secondary, extremely low-power *wake-up radio* system [34] to respond to a wake-up call from the sink, instead of relying on idle listening and RSS samples. Or, the source node could use information from outside of the network stack, such as from an accelerometer, to track the rotation of the blade and time communications accordingly.

Acknowledgment

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] "Global wind report: Annual market update 2015," Glob. Wind Energy Council, Brussels, Belgium, Rep., Apr. 2016. [Online]. Available: http://www.gwec.net/wp-content/uploads/vip/GWEC-Global-Wind-2015-Report_April-2016_22_04.pdf

[2] *Digital Wind Farm: The Next Evolution of Wind Energy*, GE Renew. Energy, Fairfield, CT, USA, May 2016, pp. 1–5.

[3] M. L. Wymore, J. E. Van Dam, H. Ceylan, and D. Qiao, "A survey of health monitoring systems for wind turbines," *Renew. Sustain. Energy Rev.*, vol. 52, pp. 976–990, Dec. 2015.

[4] M. Ozbek and D. J. Rixen, "Operational modal analysis of a 2.5 MW wind turbine using optical measurement techniques and strain gauges," *Wind Energy*, vol. 16, no. 3, pp. 367–381, Apr. 2012.

[5] O. M. Bouzid, G. Y. Tian, K. Cumanan, and D. Moore, "Structural health monitoring of wind turbine blades: Acoustic source localization using wireless sensor networks," *J. Sensors*, vol. 2015, pp. 1–11, Dec. 2015.

[6] A. Downey, S. Laflamme, and F. Ubertini, "Experimental wind tunnel study of a smart sensing skin for condition evaluation of a wind turbine blade," *Smart Mater. Struct.*, vol. 26, no. 12, pp. 1–11, 2017.

[7] J. M. Jonkman, S. Butterfield, W. Musial, and G. Scott, "Definition of a 5-MW reference wind turbine for offshore system development," NREL, Golden, CO, USA, Rep. TP-500-38060, Feb. 2009.

[8] Z. A. Eu, H.-P. Tan, and W. K. G. Seah, "Wireless sensor networks powered by ambient energy harvesting: An empirical characterization," in *Proc. IEEE ICC*, May 2010, pp. 1–5.

[9] Texas Instruments. (2017). *The SensorTag Story*. Accessed: Nov. 3, 2017. [Online]. Available: http://www.ti.com/ww/en/wireless_connectivity/sensortag/

[10] (2017). *SpectraQuest, Inc*. Accessed: Nov. 3, 2017. [Online]. Available: http://spectraquest.com/

[11] M. L. Wymore and D. Qiao, "BladeMAC: Radio duty-cycling in a dynamic, cyclical channel," in *Proc. IEEE ICC*, Paris, France, 2017, pp. 1–7.

[12] A. Dunkels, B. Grönvall, and T. Voigt, "Contiki—A lightweight and flexible operating system for tiny networked sensors," in *Proc. IEEE LCN*, Tampa, FL, USA, 2004, pp. 455–462.

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

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

[15] T. Watteyne, M. Palattella, and L. Grieco, "Using IEEE 802.15.4e time-slotted channel hopping (TSCH) in the Internet of Things (IoT): Problem statement," Internet Eng. Task Force, Fremont, CA, USA, RFC 7554, May 2015, accessed: Nov. 3, 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc7554

[16] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proc. ACM SenSys*, Baltimore, MD, USA, 2004, pp. 95–107.

[17] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-MAC: A short preamble MAC protocol for duty-cycled wireless sensor networks," in *Proc. ACM SenSys*, Boulder, CO, USA, 2006, pp. 307–320.

[18] A. Dunkels, "The ContikiMAC radio duty cycling protocol," SICS, Stockholm, Sweden, Rep. 5128, Jan. 2012.

[19] Y. Sun, O. Gurewitz, and D. B. Johnson, "RI-MAC: A receiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks," in *Proc. ACM SenSys*, Raleigh, NC, USA, 2008, pp. 1–14.

[20] R. Rajkumar, I. Lee, L. Sha, and J. A. Stankovic, "Cyber-physical systems—The next computing revolution," in *Proc. Design Autom. Conf.*, Anaheim, CA, USA, 2010, pp. 731–736.

[21] P. Kindt, H. Jing, N. Peters, and S. Chakraborty, "ExPerio—Exploiting periodicity for opportunistic energy-efficient data transmission," in *Proc. IEEE INFOCOM*, 2015, pp. 82–90.

[22] Y. Leng *et al.*, "Study on electromagnetic wave propagation characteristics in rotating environments and its application in tire pressure monitoring," *IEEE Trans. Instrum. Meas.*, vol. 61, no. 6, pp. 1765–1777, Jun. 2012.

[23] R. Matsuzaki and A. Todoroki, "Wireless monitoring of automobile tires for intelligent tires," *Sensors*, vol. 8, no. 12, pp. 8123–8138, 2008.

[24] X. Chen, S. Jin, and D. Qiao, "M-PSM: Mobility-aware power save mode for IEEE 802.11 WLANs," in *Proc. IEEE ICDCS*, Minneapolis, MN, USA, 2011, pp. 77–86.

[25] P. Mukherjee, D. Mishra, and S. De, "Exploiting temporal correlation in wireless channel for energy-efficient communication," *IEEE Trans. Green Commun. Netw.*, vol. 1, no. 4, pp. 381–394, Dec. 2017.

[26] T. Rappaport, *Wireless Communications: Principles and Practice*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2001.

[27] F. Österlind, "A sensor network simulator for the Contiki OS," SICS, Stockholm, Sweden, Rep. T2006:05, Feb. 2006.

[28] A. Dunkels, F. Österlind, and Z. He, "An adaptive communication architecture for wireless sensor networks," in *Proc. ACM SenSys*, Sydney, NSW, Australia, 2007, pp. 335–349.

[29] P. Deshpande. (Nov. 2014). *Mobility of Nodes in Cooja*. Accessed: Nov. 3, 2017. [Online]. Available: http://anrg.usc.edu/contiki/index.php/Mobility_of_Nodes_in_Cooja

[30] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and scalable simulation of entire TinyOS applications," in *Proc. ACM SenSys*, Los Angeles, CA, USA, 2003, pp. 126–137.

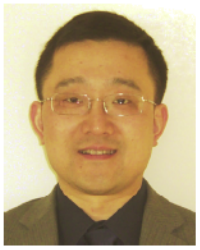
[31] National Renewable Energy Laboratory. (Jul. 2017). *NWTC Information Portal (FAST v8)*. Accessed: Nov. 5, 2017. [Online]. Available: https://nwtc.nrel.gov/FAST8

[32] W. Yang, Z. Lang, and W. Tian, "Condition monitoring and damage location of wind turbine blades by frequency response transmissibility analysis," *IEEE Trans. Ind. Electron.*, vol. 62, no. 10, pp. 6558–6564, Oct. 2015.

[33] E. A. Bossanyi, "Individual blade pitch control for load reduction," *Wind Energy*, vol. 6, no. 2, pp. 119–128, 2003.

[34] J. Oller *et al.*, "Has time come to switch from duty-cycled MAC protocols to wake-up radio for wireless sensor networks?" *IEEE/ACM Trans. Netw.*, vol. 24, no. 2, pp. 674–687, Apr. 2016.

Mathew L. Wymore (S'14) is currently pursuing the Ph.D. degree in wind energy science, engineering and policy with a co-major in computer engineering with Iowa State University. His research explores the use of wireless sensor networks for monitoring and management of wind energy infrastructure, with a focus on wireless communication protocols.



Daji Qiao (M'04–SM'17) received the Ph.D. degree in electrical engineering: systems from the University of Michigan, Ann Arbor, in 2004. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA, USA. His research interests include cyber security, wireless networking and mobile computing, sensor networks, and Internet of Things. He is a member of the ACM.