BladeMAC: Radio Duty-Cycling in a Dynamic, Cyclical Channel

Mathew L. Wymore and Daji Qiao
Department of Electrical and Computer Engineering, Iowa State University, Ames, IA
{mlwymore, daji}@iastate.edu

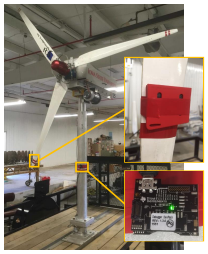
***Abstract*—As wind energy continues to expand to new frontiers in terms of the location, number, and size of wind turbines, the industry has begun to seek smarter operations and management solutions. Wireless sensing nodes could provide a low-cost platform to support a variety of applications designed to reduce the levelized cost of energy and increase the safety of wind turbines. However, a wireless sensor node deployed on a wind turbine blade would have an extremely limited energy supply. To combat this limitation, we present BladeMAC, a new MAC-layer protocol designed for sensor nodes deployed on rotating wind turbine blades. BladeMAC overcomes a unique cyclical channel problem to allow a sensor node attached to a rotating blade to opportunistically and efficiently offload its data to a sink node attached to the turbine tower. We have implemented and evaluated BladeMAC using Contiki OS and the Cooja simulation tool. We present results showing that BladeMAC effectively deals with the cyclical channel problem at a wide range of data arrival intervals, and that BladeMAC is insensitive to rotation speed and rotation speed fluctuations.**

I. Introduction

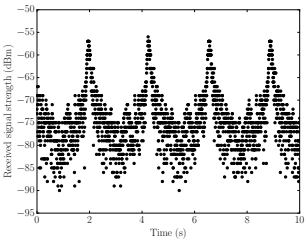
Renewable energy, and particularly wind energy, has seen massive growth in the past few decades [1] due to factors such as decreasing costs, government policies and incentives, and increased concern about fossil fuels and their effects on our planet. Wind turbines are continually growing in size, and 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. On the other hand, small-size wireless sensor nodes attached to blades could provide a low-cost, flexible data collection platform. However, the challenge of powering these nodes over the multi-decade lifespan of a wind turbine must first be overcome. Our research takes on this challenge.

Emerging energy-harvesting techniques can help address this challenge, but our approach comes from the opposite, but complementary, direction: reducing energy consumption of the nodes. Since the wireless radio hardware of a node consumes a relatively large amount of power, 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 wireless sensor network (WSN) research, and is necessary even with energy-harvesting, due to the dynamic and unreliable nature of the available energy resources.



(a) The lab turbine, with nodes attached to the tower and one of the blades. The node on the blade is encased in a rubber enclosure.



(b) An RSS trace gathered using this setup. Each point is an RSS sample taken from a packet sent at a transmission power of −21 dBm.

Fig. 1. A SpectraQuest (www.spectraquest.com) laboratory-scale wind turbine with 1.4 m blades was used to observe the cyclical channel phenomenon. The sensor nodes are TI CC2650-based Sensortags (www.ti.com/sensortag).

One method for deploying a WSN to monitor wind turbine blades could be to place a base station, or *sink*, in the hub at the center of the blades, and allow all sensing nodes, or *sources*, to communicate directly with the sink. Another option could be a multi-hop approach that uses relay nodes deployed along the length of the blade. However, neither of these traditional approaches are suited for the extremely low energy usage required for this application.

We therefore propose a novel *opportunistic single-hop* approach. Instead of placing the sink at the hub, we attach it to the tower. A source attached to a blade can then offload its data in a single hop as the blade rotates past the tower. The short distance between the nodes at this time allows for reliable, low-power transmission. 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. We observed this cyclical channel using a laboratory-scale wind turbine and TI 2650-based Sensortags (www.ti.com/sensortag), as shown in Fig. 1.

On our lab turbine, the nodes stay within range of each other throughout the rotation. But on a utility-scale turbine with 40 m blades or longer, the link between the nodes may be broken during the upper part of the rotation. This creates a unique scenario that existing duty-cycled MAC protocols are not designed to handle efficiently: instead of the traditional two-way rendezvous between the sender and receiver, the protocol must arrange a *three-way rendezvous* between the

978-1-4673-8999-0/17/$31.00 ©2017 IEEE

sender, the receiver, and the channel cycle. To address this challenge, we present BladeMAC.

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 detail our implementation of BladeMAC in Contiki OS [4], our experimental methods, and our evaluation results. Finally, we present our conclusions in Section VI.

II. Related Work

Duty-cycled MAC protocols have been a popular research topic among researchers attempting to minimize the energy consumption of WSN nodes. When a node's radio is off, it cannot receive packets; therefore, one challenge for duty-cycled MAC protocols is to efficiently arrange for a time when the sender and the receiver can communicate, known as a *rendezvous*. This rendezvous is arranged in either a *synchronous/scheduled* or an *asynchronous* manner. In synchronous/scheduled protocols such as T-MAC [5], 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 [6], X-MAC [7] and ContikiMAC [8], 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 [9], the sender silently listens until the receiver announces its availability with a beacon packet. Receiver-initiated protocols occupy the channel less, particularly when the ratio of senders to receivers is large. 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, a sender could need to wait for many channel cycles before the receiver wakes and beacons when the link is available. We have designed BladeMAC to solve this problem.

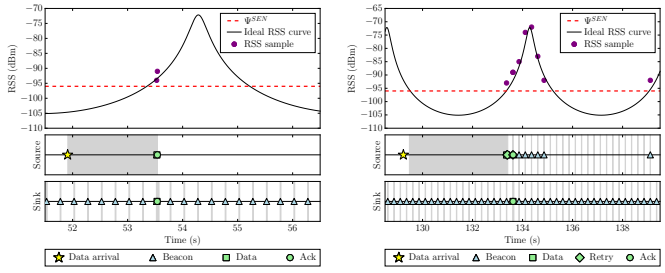
As part of our solution, BladeMAC uses the RSS and RSS trend to predict future RSS. This takes advantage of the cyber-physical characteristics of the system. Cyber-physical systems [10] have been a popular research topic in recent years. To the best of our knowledge, BladeMAC is the first research to 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 [11] and in Wi-Fi [12]. Our problem is distinct because the link between nodes may be broken entirely, the window for sending is very small, and the period may be constantly fluctuating.

III. Evolution of BladeMAC

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

A. CC-MAC

Our first baseline solution, illustrated in Fig. 2a, is a receiver-initiated scheme 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. 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}. The source sleeps, with its radio off, until it has data to send. 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. 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.



(a) CC-MAC (b) CPCC-MAC

Fig. 2. Example traces for CC-MAC and CPCC-MAC, taken from Cooja simulations. 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.

CC-MAC works well enough for infrequent data arrival, but more frequent data leads to a high percentage of time spent 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.

- **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.

B. 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. 2b). In CPCC-MAC, the source tracks the cycle period and phase, allowing it predict when the channel will be present. The source then wakes at this time and idly listens until a beacon is heard. If the cycle prediction is accurate, the idle listening time is very short.

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. We therefore suggest that CPCC-MAC perform 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.

For future data arrivals, the source uses the period estimate and the last data exchange timestamp to predict when it should wake. If the beacon arrives more than T_{B} after the predicted time, the period estimation process is repeated. For example, in Fig. 2b, 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. Thus, 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. 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 arrivals 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 difficult; (c) CPCC-MAC may introduce additional delay due to incorrect period estimation.

C. 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 readings and trends 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.

IV. BladeMAC

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 the source.

A. Overview

Fig. 3 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. BladeMAC makes use of the observation that the channel's cycle can be divided into different intervals based on RSS thresholds.

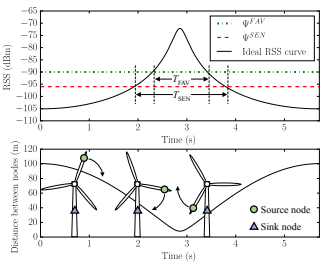


Fig. 3. Illustration of the cyclic channel caused by the rotation of the blades. We define the intervals T_{FAV} and T_{SEN} based on Ψ^{FAV} and Ψ^{SEN}.

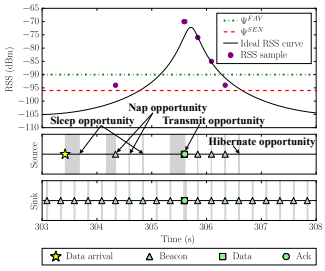


Fig. 4. Sample trace of BladeMAC's operation, from a Cooja simulation. The *opportunities* used by BladeMAC are labeled.

We here define two such intervals of the RSS curve for use in discussing BladeMAC. 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}. 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.

In BladeMAC, the source uses RSS values from the beacon and Ack packets it receives to identify the appropriate *opportunity* for the situation. In some cases, the appropriate opportunity for the source depends on the RSS *trend*. BladeMAC obtains the RSS trend by measuring the RSS of successive packets. With these tools, BladeMAC dynamically makes decisions about when to send data and when to sleep, allowing it to minimize idle listening and 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. 4 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 handed 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 (marked △), but with RSS below Ψ^{FAV}, resulting in a *nap opportunity*. The source wakes for the next two beacons, but does not hear 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 omitted due to space limitations. 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 conditions, 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 sends a beacon packet. The sink populates the beacon with the value of T_{B} to inform the source of the beacon interval length.

After sending a beacon, the sink listens for a period of time equal 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 it does receive a data packet, it responds with an Ack packet and then listens for any additional data packets.

2) Beacon Interval: A key issue for the sink is the determination of T_{B}. A smaller T_B is generally better for the source because it leads to less idle listening and a lower duty cycle. However, we assume the sink also needs to duty cycle its radio to some extent to satisfy its own energy budget. Therefore, to strike a balance between energy efficiency for the source and duty cycling for the sink, we choose an upper bound of $T_{\text{B}} \leq 0.5 T_{\text{FAV}}$, 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.

3) Favorable Interval: The above choice for T_{B} means that T_{FAV} must be known. Using empirical results and theoretical analysis, a lower bound for T_{FAV} can be estimated for a particular application.

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. 5. 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.

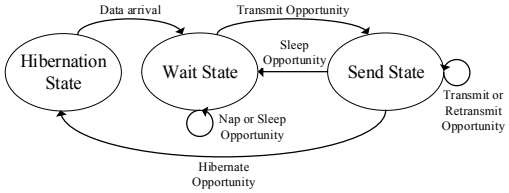


Fig. 5. Diagram of the source's state machine.

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 hearing 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*.

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 $0.5T_{\text{SEN}}$

• **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.

• **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.

• **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. In brief, to estimate T_{SEN}, the source examines the RSS from packets received during the sensitivity window. By comparing the timestamps of key packets, such as the first packet, the last packet, and the packet with the highest RSS, the source is able to obtain a conservative estimate of T_{SEN}, which it updates using a moving average. 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 speed changes. Due to space restrictions, further details of our T_{SEN} estimation process are omitted here.

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 to the point that the link disappears, which is signified by R consecutive transmission failures of a data packet. In our implementation of BladeMAC, we set $R = 3$.

V. EVALUATION

In this section, we present our evaluation methods and results for BladeMAC. We use simulations in Cooja [13], Contiki OS's [4] 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.

A. Method

We implemented BladeMAC by inserting it into Contiki's Rime network stack [14], as shown in Fig. 6. BladeMAC was implemented at the radio duty-cycling (RDC) layer, which is responsible for interfacing 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.

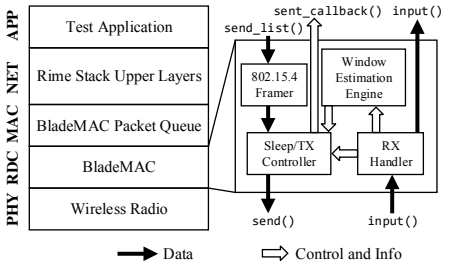


Fig. 6. BladeMAC as it fits into the Contiki network stack.

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[1] to simulate blade rotation with a 50 m radius [15], and we implemented the popular lognormal shadowing propagation model to obtain distance-based RSS. We used a path loss exponent of 3.0 and a standard deviation of 3 dB for the random component. To translate RSS into probability of reception, we set a static noise

[1]http://anrg.usc.edu/contiki/index.php/Mobility_of_Nodes_in_Cooja

floor of -100 dBm and used an empirically-derived function from TOSSIM [16] to obtain the probability of reception from the signal-to-noise ratio.

From this channel model, we obtained a favorable threshold of $\Psi^{\text{FAV}} = -90$ dBm and a beacon interval of $T_{\text{B}} = 250$ ms for our evaluation. These choices result in $T_{\text{B}} \approx T_{\text{FAV}}/4$, which satisfies our requirement that $T_{\text{B}} \leq T_{\text{FAV}}/2$. We used simulations to verify this requirement; 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 include duty cycle (a proxy for energy consumption), 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 shown here are averaged over at least 20 simulations per protocol, using the same random seeds for each protocol.

B. Rotation speed

We evaluated the protocols with static rotation speeds of 7–12 RPM [15], with a data arrival interval of 25 s. The results are shown in Fig. 7. For all metrics, CPCC-MAC shows significant sensitivity to rotation speed. This is due to an interaction of the rotation period and the data arrival interval that is aggravated by CPCC-MAC's reliance on period estimation. BladeMAC and CC-MAC are much less sensitive to rotation speed. BladeMAC's duty cycle, shown in Fig. 7a, 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.

Fig. 7b 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. 7c 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.

We choose 10.5 RPM as the default rotation speed for the rest of the evaluation because, at 10.5 RPM and a 25 s data arrival interval, BladeMAC and CPCC-MAC have similar performance in terms of duty cycle.

C. Data arrival interval

We evaluated the protocols at different data arrival intervals with a static rotation speed of 10.5 RPM. The results are plotted in Fig. 8. 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 25 s. Above this point, CPCC-MAC's period estimation is not accurate enough to be worth the overhead.

In general, we find that CPCC-MAC's performance worsens as the data arrival interval increases. CPCC-MAC shows predictable performance at lower data arrival intervals, but above 75 s, its performance begins to oscillate. CPCC-MAC's sensitivity to the ratio between rotation period and data arrival interval does not completely explain these oscillations. We believe these oscillations are evidence of another interaction, between the rotation period and the error in the period estimation process. If the data arrival interval is long enough that the wakeup prediction is off by an entire period, then the prediction may become relatively accurate again. The combination of these two interactions makes CPCC-MAC's performance difficult to predict.

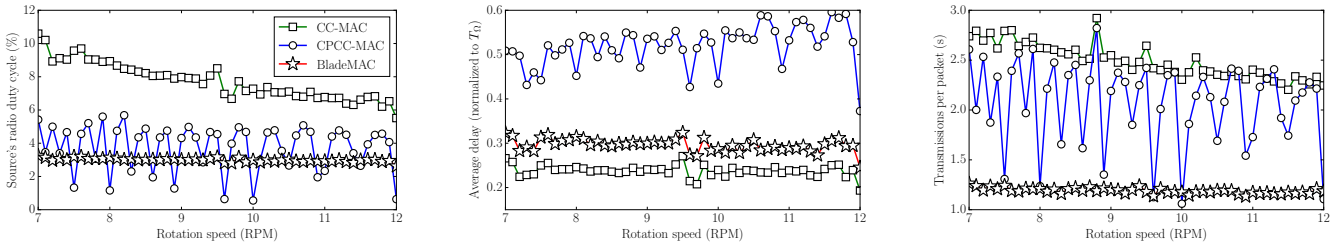
D. Dynamic rotation speed

To evaluate BladeMAC's resilience to changing rotation speeds, simulations were run with a dynamic rotation speed. The results are shown in Fig. 9. In these simulations, the rotation speed fluctuates between set points that are chosen uniformly from the given range, centered at 10.5 RPM. A new set point is reached every 10 s. The transition between set points is divided into discrete intermediate points with a resolution of 0.1 RPM. This scenario is intended to loosely mimic the effect of fluctuating wind speeds, but does not directly simulate all of the complex effects of the wind turbine's rotor blade system, such as inertia. More realistic dynamic scenarios are left to future work.

As expected, CPCC-MAC and BladeMAC have a similar duty cycle at zero variation (a constant speed of 10.5 RPM, which matches the previous figures). However, even the smallest amount of variation begins to tilt the scales in BladeMAC's favor. Above a range of ± 0.1 RPM, which is a variation of less than 2% of the rotational speed, BladeMAC performs better than CPCC-MAC. 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.

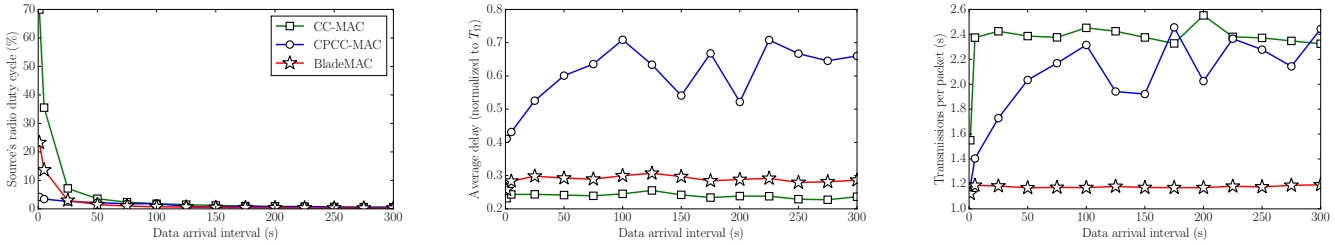
VI. Conclusion

We have presented BladeMAC, a radio duty-cycling MAC protocol designed specifically for wireless sensor nodes deployed on rotating wind turbine blades. BladeMAC addresses



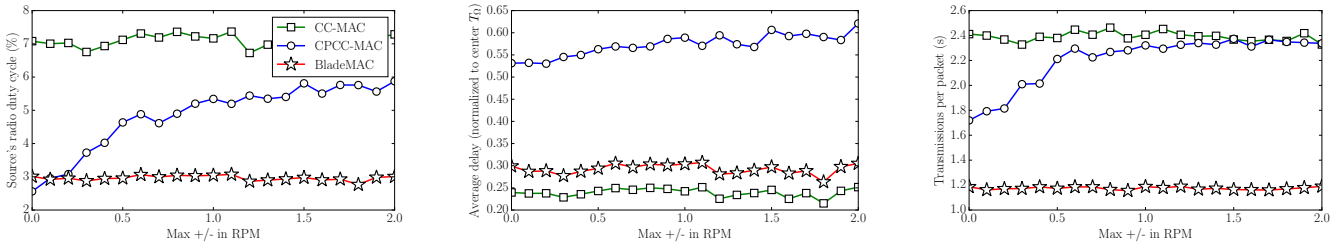
(a) Duty cycle. (b) Delay. (c) Transmissions per packet.

Fig. 7. Evaluation for different static rotation speeds, with a 25 s data arrival interval.



(a) Duty cycle. (b) Delay. (c) Transmissions per packet.

Fig. 8. Evaluation for different data arrival intervals, with a static 10.5 RPM rotation speed.



(a) Duty cycle. (b) Delay. (c) Transmissions per packet.

Fig. 9. Evaluation with a 25 s data arrival interval and dynamic rotation speed. The x-axis is the range of speeds allowed, in an interval centered on 10.5 RPM.

the cyclical channel problem created by this scenario in a way that has been shown to be effective regardless of rotation speed, data arrival interval, and rotation speed variation. Future work includes experiments on hardware, and extensions of BladeMAC to consider multiple sources and to take advantage of wakeup prediction when the rotation speed is stable.

Acknowledgements

The work reported in this paper was funded in part by U.S. National Science Foundation Grant No. 1069283, which supports the activities of the Integrative Graduate Education and Research Traineeship (IGERT) in Wind Energy Science, Engineering and Policy (WESEP) at Iowa State University.

References

[1] "Global wind report: annual market update 2015," Global Wind Energy Council, Tech. Rep., Apr. 2016.

[2] GE Renewable Energy, "Digital Wind Farm: the next evolution of wind energy," pp. 1–5, May 2016.

[3] M. L. Wymore, J. E. Van Dam, H. Ceylan, and D. Qiao, "A survey of health monitoring systems for wind turbines," *Renewable and Sustainable Energy Reviews*, vol. 52, no. C, pp. 976–990, Dec. 2015.

[4] A. Dunkels, B. Grönvall, and T. Voigt, "Contiki—a lightweight and flexible operating system for tiny networked sensors," in *Proc. IEEE LCN*, 2004.

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

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

[7] 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*, 2006.

[8] A. Dunkels, "The ContikiMAC radio duty cycling protocol," SICS, Tech. Rep. 5128, Jan. 2012.

[9] 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*, 2008.

[10] R. Rajkumar, I. Lee, L. Sha, and J. A. Stankovic, "Cyber-physical systems - the next computing revolution," in *Proc. Design Automation Conference*, 2010.

[11] P. Kindt, H. Jing, N. Peters, and S. Chakraborty, "ExPerio—exploiting periodicity for opportunistic energy-efficient data transmission," in *Proc. IEEE INFOCOM*, 2015.

[12] X. Chen, S. Jin, and D. Qiao, "M-PSM: Mobility-aware Power Save Mode for IEEE 802.11 WLANs," in *Proc. IEEE ICDCS*, 2011.

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

[14] A. Dunkels, F. Österlind, and Z. He, "An adaptive communication architecture for wireless sensor networks." in *Proc. ACM SenSys*, 2007.

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

[16] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: accurate and scalable simulation of entire TinyOS applications," in *Proc. ACM SenSys*, 2003.