2019

Low-energy sensor network protocols and application to smart wind turbines

Mathew Lee Wymore
Iowa State University

Low-energy sensor network protocols and application to smart wind turbines

by

Mathew L. Wymore

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Co-majors: Wind Energy Science, Engineering, and Policy; Computer Engineering

Program of Study Committee:
Daji Qiao, Co-major Professor
Halil Ceylan, Co-major Professor
Ahmed Kamal
Lizhi Wang
Wensheng Zhang

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this dissertation. The Graduate College will ensure this dissertation is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2019

DEDICATION

Dedicated to my parents, Joyce and Lee; to my wife, Megan; and to Munster, the cat. Your constant love, support, and patience have kept me going.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to thank my co-major professors: Dr. Daji Qiao, whose endless hours of mentoring and constant stream of opportunities got me here, and Dr. Halil Ceylan, whose support was unwavering. I would also like to thank Dr. Jim McCalley and the WESEP program, and acknowledge the NSF IGERT fellowship that jump-started my graduate career.

ABSTRACT

The Internet of Things (IoT) has shown promise as an enabling technology for a wide variety of applications, from smart homes to infrastructure monitoring and management. However, a number of challenges remain before the grand vision of an everything-sensed, everything-connected world can be achieved. One of these challenges is the energy problem: how can embedded, networked sensor devices be sustainably powered over the lifetime of an application? To that end, this dissertation focuses on reducing energy consumption of communication protocols in wireless sensor networks and the IoT. The motivating application is wind energy infrastructure monitoring and management, or "smart" wind turbines. A variety of approaches to low-energy protocol design are studied. The result is a family of low-energy communication protocols, including one specifically designed for nodes deployed on wind turbine blades. This dissertation also presents background information on monitoring and management of wind turbines, and a vision of how the proposed protocols could be integrated and deployed to enable smart wind turbine applications.

CHAPTER 1. OVERVIEW

1.1 Introduction

1.1.1 Climate Change and Wind Energy

Climate change is a manmade crisis of previously-unexperienced, global proportions [1]. One of the primary emitters of the greenhouse gases that contribute to climate change is the electricity generation industry, as seen in Fig. 1.1. However, as the figure shows, the electricity sector also has the greatest potential for emissions reductions in typical mitigation scenarios. These reductions would largely come from a transition to clean, renewable energy sources. One such source that is technologically ready for deployment on a massive scale is wind energy. Even putting environmental benefits aside, wind energy is now cost-competitive with fossil fuels in many markets [2].

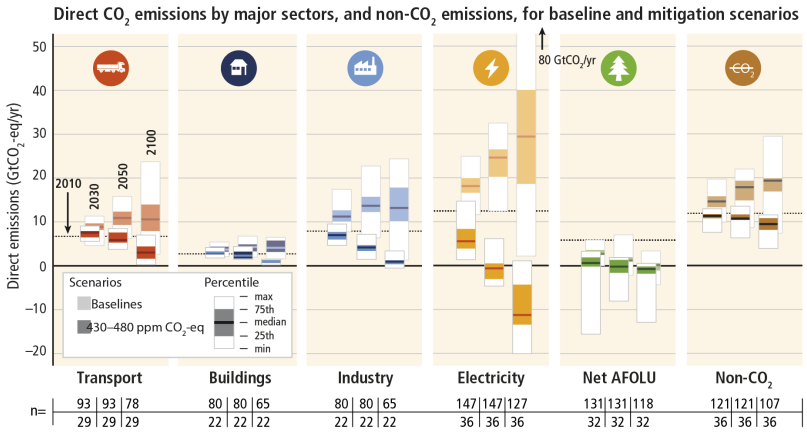


Figure 1.1: Emissions contributions by sector. The electricity generation sector contributes the most emissions, both currently and in projected baseline futures. However, electricity generation is also poised to make the largest reductions in mitigation scenarios. From [1].

These and additional factors have led to a rapid increase in the installed capacity of wind energy over the past decade and a half, as shown in Fig. 1.2. However, looking ahead, the wind industry faces a number of changes and challenges, particularly in the U.S. For example, the U.S. Congress

has agreed to phase out the wind Production Tax Credit (PTC) [3], which will stabilize policy for the wind industry but also remove the industry's most significant subsidy. To sustain growth and the cost advantage currently enjoyed by wind, the industry will need to further decrease costs. Some other challenges to the wind industry in the U.S. include the dismemberment of the Clean Power Plan [4] and the U.S.'s plan to withdraw from the Paris climate accord [5].

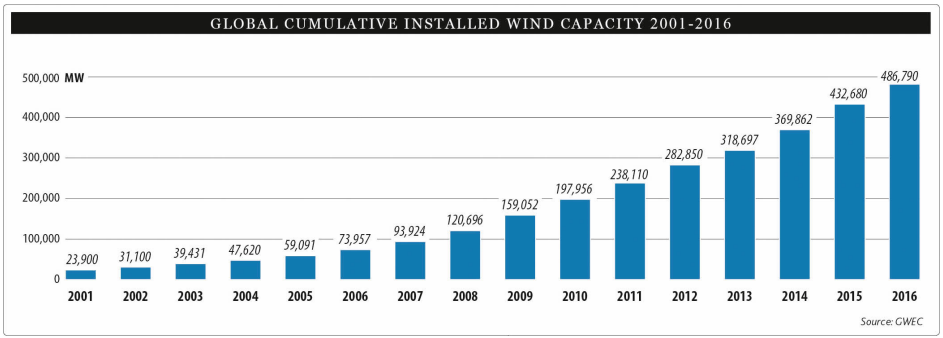


Figure 1.2: Global cumulative wind energy installed capacity over time. From the Global Wind Energy Council (www.gwec.net).

1.1.2 Smart Wind Turbines

One challenge for wind plant operators moving forward will be to keep operations and maintenance (O&M) costs in check. Since each wind turbine is a separate and complete machine, and these machines are spread across the countryside (or offshore), planning wind plant maintenance is a task that becomes increasingly more challenging as the plant increases in capacity. The wind industry has turned to advanced, data-driven techniques to tackle this challenge, with data potentially coming from a wide variety of advanced monitoring systems [6].

At a high level, adoption of these systems and techniques is a simple matter of economics: if a technique lowers the levelized cost of energy (LCOE, a measure of the total cost of energy over the lifetime of a project), the industry will use it to stay competitive. However, the cost-benefit analysis of data-driven O&M techniques is not straightforward. Costs include the capital cost and installation cost for the monitoring equipment, as well as the ongoing costs of interpreting the data and maintaining the system. Benefits depend on the goal of the monitoring system and can be

complex to analyze. For example, for a condition monitoring (CM) system, the benefits depend on the type of failures and repair costs for the component being monitored, the failure rate of the component, and the capabilities of the system in detecting impending failures, in terms of probability of detection and time between detection and failure.

The cost-benefit tradeoff clearly improves if the benefits increase, but a complementary approach is to reduce the costs of the systems used to gather and process the data. Also, reducing hardware costs can mitigate "sticker shock," which can dissuade investment in monitoring systems regardless of the long-term economics. For example, a survey that we conducted of wind plant operators found that, with no context on benefits or even which components were being monitored, the majority of respondents indicated that a cost of less than 1% of the turbine cost is appropriate for an advanced monitoring system.

Wireless sensor networks (WSNs) have the potential to provide a low-cost platform for data collection on a variety of wind turbine components, particularly the large structural components such as the wind turbine blades, tower, and foundation. However, some technical challenges unique to wind turbines must be overcome before WSNs can be a fully viable platform. Wind turbines are unique physical structures, and traditional WSN deployment approaches must be reconsidered for some components, particularly the wind turbine blades. Additionally, some of the traditional challenges of WSNs, particularly energy consumption, are amplified in a wind turbine deployment scenario. For example, nodes in WSNs are typically battery-powered, but battery replacement in a wind energy scenario is prohibitively expensive, since wind turbines are geographically dispersed and cannot be accessed without shutdown (and potentially specialized workers such as rope-access technicians).

This dissertation presents work that addresses these challenges. First, the remainder of this chapter presents an overview of WSNs. Chapter 2 presents an overview of our motivating application, monitoring and management of wind energy infrastructure. The remaining chapters describe a family of low-energy protocols inspired by this application.

1.2 Wireless Sensor Networks

WSNs are a type of computer network composed of small, self-contained nodes (Fig. 1.3) that communicate wirelessly, often in multi-hop topologies. Since they typically run on batteries, sometimes with energy harvesting, WSN nodes are energy-constrained. WSNs are classified by the Internet Engineering Task Force (IETF) as Low-power and Lossy Networks (LLNs) [7]. The lack of energy and computing resources, along with the unreliable nature of the wireless links, are the key challenges and characteristics of WSNs.

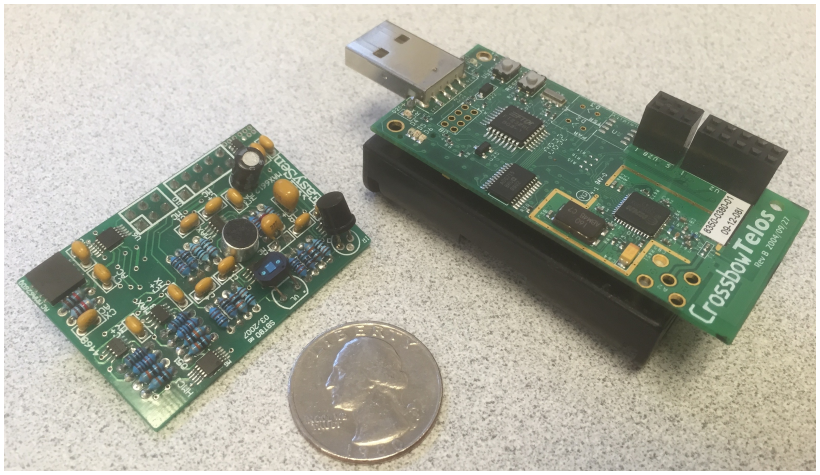


Figure 1.3: Examples of traditional WSN hardware. A Crossbow TelosB mote and an EasySen sensor board, shown with a U.S. quarter for size comparison.

WSNs are related to the emerging Internet of Things (IoT) concept [8], and IoT networks may use similar hardware and protocols as WSNs. The IoT can be thought of as an expansion of the scope and scale of WSNs, along with a tighter integration with the existing Internet. WSNs have found application in SHM [9], environmental and wildlife monitoring [10], disaster recovery scenarios [11], smart buildings and cities [12], and more. Many of these types of applications have been made possible by the advent of sensors based on micro-electromechanical systems (MEMS), which essentially allow for the integration of mechanical sensors into small chips and circuit boards. As more of these types of sensors become available, WSNs may find an even wider range of uses, particularly when integrated into the IoT.

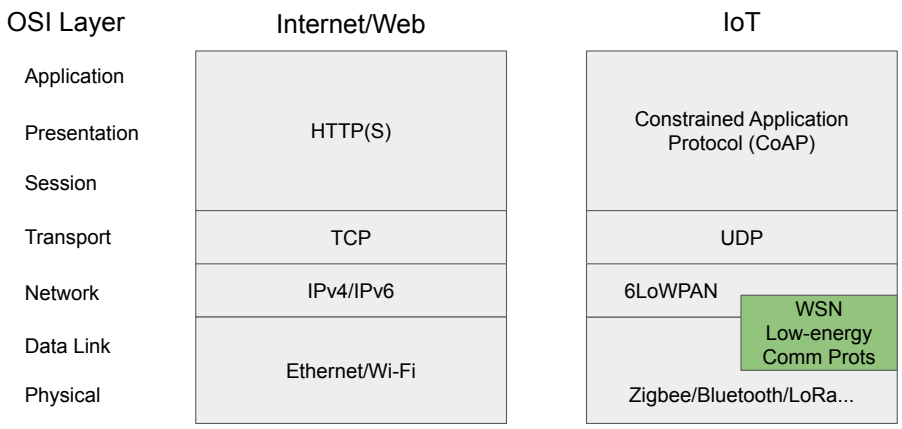


Figure 1.4: Potential protocol stack for the IoT, as compared to a more traditional web stack. The green box indicates where low-energy WSN protocols can find application in the IoT.

Like all modern computer networks, the IoT can be organized into layers. An overview of possible layers for the IoT, as related to the classic OSI seven-layer model, is shown in Fig. 1.4. The green box in the figure indicates where low-energy WSN protocols can fit into an IoT stack. The main layers of interest are the physical layer, the link layer, the network layer, and the application layer. These layers are described in the remaining parts of this section.

1.2.1 Physical Layer

The physical layer is comprised of the wireless medium, the antenna and wireless radio hardware, and the modulation techniques used to translate the binary data into a radio wave. In practice, WSNs have traditionally used the IEEE 802.15.4 standard's physical layer [13], which provides (among other options) a 250 kbps data rate in the 2.4 GHz band using direct sequence spread spectrum (DSSS) and offset quadrature phase-shift keying (OQPSK).

1.2.2 Link Layer

The link layer resides immediately above the physical layer and is responsible for establishing and maintaining the link between neighboring nodes. Because of energy constraints, link layers in WSNs usually *duty-cycle* the radio [14], meaning they turn the radio on and off as needed. The

reasoning behind duty-cycling is that the radio hardware consumes a significant amount of power, often to the point where it dominates the overall energy consumption of the node. Turning off the radio when it is not in use saves energy and can greatly extend the lifetime of the node. However, for a sender and receiver to communicate, the radios of both nodes must be on. Therefore, the link layer must arrange a *rendezvous*, or a time when communication can happen, between the sender and receiver.

Because of the technical challenge and the potential energy savings of duty cycling, link layer protocols have been been a staple WSN research topic for well over a decade [14]. Link layers may generally be categorized as synchronous or asynchronous, depending on how they arrange the rendezvous between sender and receiver. Synchronous protocols set a schedule for when nodes should listen to the channel and when nodes should transmit. A state-of-the-art synchronous protocol is IEEE 802.15.4e's Time Slotted Channel Hopping (TSCH) protocol [15], which is designed to achieve high reliability and low latency, particularly in industrial environments. However, synchronous protocols suffer from problems with overhead: setting a schedule for nodes is complicated (even 802.15.4e does not specify how to schedule for TSCH), and the level of clock synchronization between nodes that is required for a tightly scheduled protocol is not trivial for WSN hardware, which traditionally exhibits significant clock drift. Overall, synchronous link layers are currently best suited for applications that need to send data frequently, and with hard delay constraints.

Asynchronous link layers, on the other hand, do not adopt a strict schedule, and do not rely on clock synchronization. Instead, asynchronous link layers use the wireless medium to arrange a rendezvous on demand. The exact method can be classified as either *sender-initiated* or *receiver-initiated*. In a sender-initiated protocol, the sender occupies the channel by emitting a jamming signal or by repeatedly transmitting a packet. The receiver periodically listens to the channel, and if it hears a sender, it engages in communication. This is also called *low-powered listening*. Some notable sender-initiated protocols include B-MAC [16], X-MAC [17], and ContikiMAC [18].

In a receiver-initiated protocol, the sender silently listens to the channel. The receiver periodically transmits a short "probe" packet that advertises its presence, and the sender responds to

the packet. This is also called *low-power probing*. While probing is less efficient than low-power listening, receiver-initiated protocols tend to be more efficient than sender-initiated protocols in higher-traffic scenarios because they occupy the channel less. An example of a well-known receiver-initiated protocol is RI-MAC [19]. Much of the research presented here is built on the concepts of asynchronously duty-cycled receiver-initiated link layers, which are discussed in more detail in Chapter 3.

Some research logically separates the link layer into the media access control (MAC) protocol and the radio duty-cycling (RDC) protocol. This is the approach of Contiki OS [20], an open-source real-time operating system favored by WSN researchers. As our work uses Contiki for evaluations, we adopt the Contiki model, explained in greater detail as follows:

MAC The MAC protocol is generally responsible for managing the wireless radio's access to the channel. The IEEE 802.15.4 standard defines a MAC protocol that may be used in WSNs. However, if duty-cycling is enabled, the precise timing of channel access is handled by the RDC protocol. The MAC protocol instead takes on responsibilities such as queuing packets and handling collisions and retransmissions.

RDC The RDC protocol is responsible for duty-cycling, or turning on and off, the wireless radio hardware. Therefore, the RDC protocol is primarily responsible for arranging the rendezvous between the sender and receiver. Using this link layer model, many classic WSN "MAC" protocols are actually RDC protocols, including those mentioned above.

However, we note that most RDC protocols are called MAC protocols in the literature, and we use the two interchangeably here.

1.2.3 Network Layer

The network layer is responsible for organizing the nodes into a topology. In WSNs, this often means that the network layer is responsible for multi-hop routing. From a local perspective, then, the network layer is responsible for determining the next hop for a packet, given the packet's

destination. WSNs can borrow multi-hop routing protocols, such as AODV [21] or OLSR [22], from the more general realm of ad-hoc wireless networks. However, WSNs have traditionally catered to applications with predictable traffic patterns, and more efficient protocols for these applications have been developed. In particular, WSNs tend to cater toward data collection, in which many *source* nodes generate data packets and send them to a single *sink* node, which interfaces the WSN with an outside system. This traffic pattern is known as "many-to-one."

The classic many-to-one routing protocol for WSNs is called the Collection Tree Protocol (CTP) [23]. As shown in Fig. 1.5, CTP builds a tree with the destination node (the sink) at the root. Nodes forward data packets to their parents in the tree; in this way, all data eventually reaches the root. The tree is built using a *routing metric*, which is a measure of the "distance" between a node and the destination. In some "geographical" routing protocols, the routing metric corresponds to physical distance, but in most protocols, the routing metric is based on some other model. Classic examples are hop count and the expected transmissions to reach the destination (ETX). The value of a routing metric for a particular node is calculated recursively based on its chosen *forwarder* (parent in the data collection tree), and the forwarder is chosen to minimize the metric value. For example, if p_{ij} is the probability of success of a TX attempt from i to j, then node i's ETX can be calculated as follows:

$$ETX_i = \min_{j \in \Gamma_i} \left(ETX_j + \frac{1}{p_{ij}} \right), \tag{1.1}$$

where Γ_i is the set of i's neighbors. As its forwarder, node i chooses the neighbor $j \in \Gamma_i$ that provides ETX_i. The destination's routing metric value is zero, and other nodes repeatedly calculate and propagate their own routing metric values as new information arrives. In a stable state, the network forms a routing metric *gradient*, and packets flow along the gradient to the destination.

The concepts of CTP have been expanded and formalized in the IETF's Routing for Low-Power and Lossy Networks (RPL) protocol [7]. RPL is capable of operating as part of a 6LoWPAN stack, and in this way, the distinction between WSN networking and IoT networking is blurred.

In Chapter 4, we present an opportunistic data collection scheme that builds on CTP/RPL. This scheme allows the link layer and the network layer to work together to dynamically choose the next

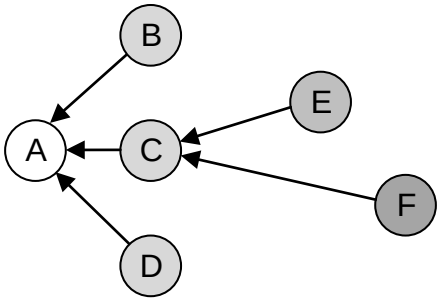


Figure 1.5: Example data collection topology using CTP. Node A is the destination, or sink, and all other nodes are sources. The shading gradient represents the routing metric values of the nodes.

hop, and is thus referred to as a *cross-layer* protocol. The use of cross-layer techniques has gained traction in recent years in order to make further gains in the efficiency of WSN communications.

1.2.4 Application Layer

The application layer is the highest layer in the protocol stack. In a WSN, the application layer is generally responsible for scheduling and performing sensor readings, any pre-processing of the sensor data, and initiating transmission of the data. While the work presented in the following chapters does not cover the application layer, it is worth noting that a number of application-layer techniques for saving energy have emerged [24]. For example, a source node can build and distribute a model for its expected sensor readings. Then, instead of reporting every reading, the source node only reports readings that deviate significantly from the model. This type of scheme can largely reduce data transmissions, thereby saving energy on communication.

1.2.5 Summary

WSNs are a relatively young field, even for computer networks. Much progress has been made, but more technical challenges must be surmounted before reliable, long-lived WSNs can be ubiquitously implemented. Still, the potential of WSNs to act as a low-cost platform for data collection has made them an attractive area of research, particularly given the trend toward data-driven applications and solutions for many critical systems—including wind energy.

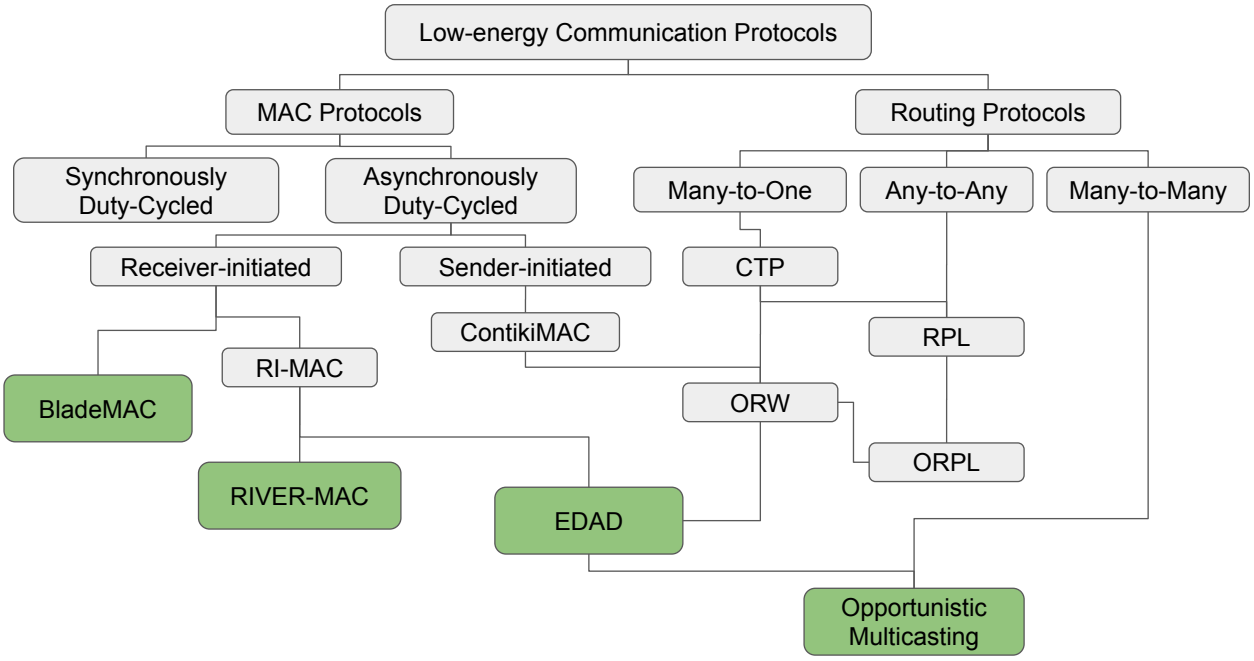


Figure 1.6: A classification tree of approaches to low-energy protocol design in WSNs, with our contributions highlighted in green.

1.3 Contributions and Approaches

The primary contribution of this dissertation is a family of low-energy wireless communication protocols. These protocols have been designed with application to wind energy infrastructure monitoring and management in mind. This motivating application is discussed in Chapter 2 (see also [6]). The protocols presented here are RIVER-MAC, EDAD [25], an opportunistic multicast scheme [26] based on EDAD, and BladeMAC [27, 28]. As shown in Fig. 1.6, these protocols can be categorized by their various approaches to the problem of reducing energy consumption of wireless communication protocols. At a high level, those approaches are as follows:

Link layer improvements This approach, exemplified by the RIVER-MAC protocol presented in Chapter 3, focuses on making a general-purpose link layer as efficient as possible. Protocols created with this approach are widely-applicable but may not be optimal for all scenarios, due to their generic nature.

Cross-layer optimization This approach involves the use of information from multiple layers. The EDAD protocol and its multicast extension, presented in Chapter 4, merge information from the routing and link layers to allow for very low-energy opportunistic communication. We have also extended this approach to multicasting, an important communication pattern not typically supported by low-energy protocols, in Chapter 5.

Application-specific design This approach involves tailoring the communication protocol to a specific system. As an example, a wind turbine is a very unique structure with unique physical characteristics. The BladeMAC protocol, presented in Chapter 6, is designed specifically for offloading data from nodes deployed on wind turbine blades. Application-specific protocols can potentially provide optimal performance for their target systems, but likely find little use otherwise.

In Chapter 7, we present a high-level vision of a smart wind turbine enabled by our family of low-energy protocols. We also sketch out a vision for the future of sensor network technology.

CHAPTER 2. MOTIVATING APPLICATION

2.1 Introduction

The wind energy industry has grown quickly since the early 2000s. Global wind capacity reached close to 370 GW by the end of 2014, with China alone installing over 23 GW in 2014 [29]. Wind penetration has increased as well: in 2014, the U.S. state of Iowa generated over 28% of its electricity from wind power [30], and in 2013, wind supplied up to 68.5% of Spain's demand coverage [31]. Additionally, the European Wind Energy Association (EWEA) is tracking plans for over 98 GW of offshore capacity [32], and offshore wind is expected to continue to grow.

Some of wind energy's growth is due to increased public awareness of impending climate issues; another factor is the decreasing cost and increasing output of wind energy systems. Yet another driver of growth has been government policy, such as the adoption of renewable portfolio standards (RPSs) by a majority of U.S. states, requiring utilities to obtain a portion of the electricity they supply from renewable sources. The American Wind Energy Association (AWEA) expects utilities to choose wind to fulfill over 40% of all RPS requirements put in place as of 2013 [33]. With these factors contributing to increased demand, more cost-effective solutions for wind energy are sought to meet this demand and make wind even more attractive.

An emerging approach to reducing wind energy costs is to make wind turbines smarter. Sensing systems deployed on each turbine can collect data that could be used for a variety of purposes. A number of such systems for wind turbines already exist and provide benefits such as online health monitoring (HM) and load mitigation. Many other uses for such systems can be imagined, such as providing data to turbine control processes to maximize wind farm efficiency. Because of this, HM and other data collection systems are expected to play an important role in wind energy's future.

In fact, as time goes on, HM of wind turbines becomes more important and more attractive. Turbines have rapidly grown in physical size, and correspondingly, in power generation. This

means that each turbine is becoming a greater source of revenue, and mitigating downtime is becoming more critical. Additionally, larger components are generally more expensive to maintain and replace. More and more wind farms are being sited offshore, where remote monitoring is particularly useful. And finally, wind energy is becoming a larger part of the world's electrical generation portfolio, so wind turbines are going to be relied upon for consistent operation more and more. Therefore, increased interest in systems for smart wind turbines is expected. This chapter is intended to provide a contextualized, practical introduction to these systems.

Existing surveys of HM for wind turbines tend to focus on monitoring techniques [34, 35, 36, 37, 38, 39], such as sensing technologies, with minimal attention given to the systems perspective. Existing lists of systems [40, 41], while useful, are not intended to provide much context about monitoring, and also do not cover academic and research systems. Finally, existing surveys tend to either focus on particular turbine components, or provide little context about for which component a technology is suited. The goal of this chapter is to provide a comprehensive, component-by-component survey of existing HM systems for wind turbines. Contrasting with existing surveys, the primary focus is at the system level. Specific sensing technologies and data processing techniques are discussed to the extent required to contextualize the systems and discuss future directions, but for more details on sensing technologies and analysis techniques, readers are encouraged to refer to other sources. This chapter is also intended to provide context and high-level background on HM and the challenges faced by industry and researchers applying HM to wind turbines.

The remainder of this chapter is organized as follows. The next section provides background information on HM. Section 2.3 presents a discussion of the motivations for monitoring wind turbines and an overview of the challenges and current state. Sections 2.4 to 2.7 examine challenges, existing systems, and the future for HM of turbine nacelles, foundations, towers, and blades, respectively. Finally, the discussion is concluded in Section 2.8.

2.2 Health Monitoring

This section presents background information about HM in general. As discussed in Section 2.3, the potential uses of a turbine's sensing system are not limited to traditional HM. However, the bulk of existing systems are designed for HM-related purposes, so HM is discussed here to provide context.

The definitions of HM and related topics are not consistent in the literature. In this chapter, HM refers to the process of using a sensing system to detect damage to an object, with damage being defined as a change in the object's properties that adversely affects current or future performance [42]. This chapter defines condition monitoring (CM) as HM applied to machinery, and structural health monitoring (SHM) as HM applied to structures. This work distinguishes HM from nondestructive evaluation and testing (NDE/T) by further defining HM as online, continuous monitoring using a permanent or semi-permanent sensing system installation. By this definition, an HM system could use NDE/T technologies for damage detection. The use of vibrations for damage detection will be referred to as vibration-based damage detection (VBDD). Both CM and SHM commonly employ VBDD, though other methods do exist and will be discussed.

2.2.1 Condition Monitoring

In this chapter, CM is defined as HM for rotating or reciprocating machinery. VBDD is a key technique for components such as drive shafts, bearings, gearboxes, and generators. One proven monitoring method is to use sensors such as piezoelectric accelerometers, attached to the component or its casing, to obtain dynamic characteristics such as vibration velocity [43]. For example, Fig. 2.1 shows instrumentation of a bearing in a wind turbine gearbox. If the dynamic characteristic being measured changes significantly from the reference, or undamaged, state, then damage is present. Industries with a history of CM may have databases of vibration signatures for machinery that can be used to diagnose the specific problem associated with a particular change.

Another common CM method comes from tribology, the study of moving surfaces interacting with each other. This includes analysis of lubrication and of wear. Oil samples can be analyzed to

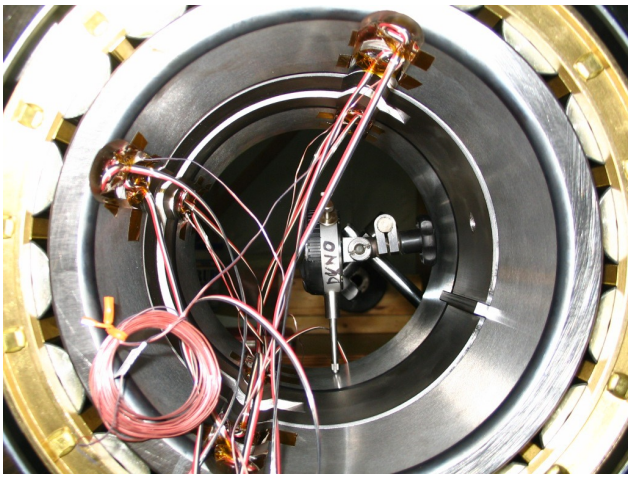


Figure 2.1: Instrumentation of a gearbox bearing. Photo by Jeroen van Dam, NREL 19680.

determine viscosity, levels of contaminants such as water, coolant, or fuel, and size, shape, composition, and count of solid particles. Traditionally, detailed analysis requires expensive laboratory equipment, and a regular sampling schedule is required [44]. However, recent advances in sensing technology have made more and more oil analysis tools available for continuous monitoring [45].

Oil monitoring can serve two purposes. The first is to measure the quality of the oil. This allows optimization of the oil changing schedule and prevents damage from operation with poor quality oil. The second purpose of oil monitoring is to measure wear on the machinery. For instance, the presence of large particles, an excessive amount of particles, or particles of a particular shape can indicate impending failure or abnormal wear conditions [44]. This type of detailed analysis is still typically performed offline, in a laboratory [46].

Other CM techniques include motor current signature analysis (MCSA) and acoustic emissions (AE). MCSA is analogous to vibration analysis, but monitors the current fluctuations in the electrical motor driving the machinery instead of the physical vibrations of the machinery. In contrast, AE detects damage by measuring the elastic waves emitted from a rapid release of strain energy in a material, which happens when the material undergoes stress or when damage occurs. While AE is considered more sensitive than vibration techniques, the disadvantage of AE is that, due to

attenuation of the elastic waves, AE sensors need to be close to the point of stress or damage in order to be effective. This is often not practical in moving machinery [47].

Overall, CM is a well-developed field. CM systems are regularly used by the transportation and manufacturing industries, among others [48]. Online oil monitoring is used in applications ranging from automobiles to electrical transformers [49]. A typical CM system uses wired analogue sensors connected to some type of data acquisition unit (DAU), so future advancements in CM will likely include miniaturization, digitization, and modularization of data collection systems. Also, CM data still often requires an expert to interpret, so ease of use is another area for improvement.

2.2.2 Structural Health Monitoring

SHM is here defined as HM applied to a structure in order to monitor its integrity or estimate its lifetime. Rytter [50] outlines four levels of damage detection for an SHM method:

1. Detection - the method indicates damage might be present
2. Localization - the method indicates where the damage likely is located
3. Assessment - the method estimates the severity of the damage
4. Consequence - the method evaluates structure safety, given the damage

While a system of the fourth level is ideal, systems of the lower levels are generally simpler and easier to implement.

Farrar and Worden [42] and Worden et al. [51] discuss damage in terms of length-scales and time-scales. The length-scale classifies the damage in terms of physical scope. For instance, the damage may be only at the material level, as in a natural defect, or the damage may be at the structural level, as in a failed wind turbine blade like that shown in Fig. 2.2. An SHM system for a wind turbine should be able to detect damage at the component level, meaning that damage requiring a repair should be detected well before component failure occurs.

The time-scale of damage refers to the length of time over which the damage takes place. For a wind turbine, long time-scale damage includes fatigue, the weathering of the protective coatings

Figure 2.2: Structural failure in a wind turbine blade. Photo by Mike Lascut, used with permission.

of the blades and tower, the corrosion of the tower, and the erosion and delamination of the blades. Short time-scale damage includes lightning strikes, impact damage, extreme winds, and earthquakes. A comprehensive SHM system for wind turbines would be able to detect all of these types of damage, though a variety of detection methods may be required.

SHM's roots are in NDE/T. Early studies of dynamic properties for damage detection examined modulus and damping. In 1978, Adams et al. [52] identified natural frequencies as a property of a structure that could be measured anywhere on the structure but still provide damage localization and characterization. Since then, VBDD has developed as a prominent method of damage identification for structures because it detects damage on a global scale, requiring relatively few sensors and no prior knowledge of the damage location. The fundamental idea of VBDD in SHM is that, because of a structure's material properties, it responds a particular way to outside forces, such as wind. As the structure takes damage, its material and dynamic properties change, so its response to forces also changes. VBDD SHM systems are designed to detect these changes in order to characterize the damage [53].

VBDD techniques have expanded to include numerous methods in addition to natural frequencies, such as mode shapes, modal strain energy, residual force vectors, and statistical methods. Carden and Fanning [54] performed a comprehensive literature review of VBDD SHM methods in 2004 and concluded that no one method had been invented to identify any given type of damage to

any given structure. Furthermore, they noted disagreement in the literature as to the effectiveness of various methods, and noted a lack of tests performed in the field. These observations appear to still hold.

In the search for a better method, a number of non-VBDD methods have been developed, especially for use with composite materials such as concrete and fiber reinforced polymers (FRPs) like those used in wind turbine blades. One such method is AE, but as in CM, AE only detects local damage and thus requires sensors close to the point of damage, which may not be feasible on a large structure if the location of damage is not known beforehand. Infrared thermography techniques work well in laboratory settings, but most require the target material to be actively heated, so online monitoring of large structures using infrared is not yet practical. Many other NDE/T techniques suffer from similar setbacks.

For a simpler approach, moisture and temperature profiles of a composite material can indicate health and inform lifetime estimates, as these factors contribute to cracking and delamination. Also, numerous types of sensors, such as piezoresistive strain gauges and fiber Bragg grating (FBG) fiber optic sensors, can be used to simply measure strain on a structure. This type of instrumentation can identify stress hotspots and, with the ability to trend over time, can be used to infer the stability of a structure's health or make lifetime estimates through fatigue analysis. Another straightforward method is to monitor the width of an already-existing crack, looking for changes over time [55, 56].

In recent years, a number of enabling technologies for SHM have advanced considerably. For instance, computer chips that contain both a microprocessor and a wireless communication unit now cost only a few dollars. This type of integration and miniaturization allows for the creation of cost-effective and powerful wireless sensor networks (WSNs) [57, 58] consisting of self-contained nodes that can be powered with energy scavenged from the environment. Additionally, sensors that can easily interface with such systems, such as sensors based on microelectromechanical systems (MEMS), are becoming more accurate and more prevalent. These sensors are manufactured in mass quantities at low prices using technologies adapted from semiconductor fabrication [59]. With hardware of this caliber available, SHM appears poised to make the transition from scattered

research applications to mainstream industry, especially in the monitoring of critical infrastructure such as wind turbines.

In addition to cost and the choice of sensors, SHM presents many more challenges, such as data processing, analysis, and management. Also, SHM is an interdisciplinary field, requiring a range of expertise. For example, the complete design of an SHM system for a wind turbine blade might require an NDT/E expert to understand what needs to be monitored, an electrical engineer to design a sensor, an aerospace engineer to determine where the sensors can be placed, a computer engineer to network the sensors together, a statistician to process the acquired data, and a civil engineer or wind energy expert to make recommendations based on the results.

But these challenges are surmountable. Even in its current form, SHM is being used for a variety of purposes worldwide, from monitoring restoration projects to evaluating critical infrastructure like roads and bridges. In Athens, Greece, the structural integrity of the Parthenon's west facade is being monitored while the monument is being renovated. The monitoring is performed by a fiber optic sensing system designed by CRD Group [60], shown in Fig. 2.3. In the UK, government mandates have encouraged automated SHM systems to evolve from traditional monitoring techniques for structures such as dams and nuclear power plants [61]. In South Korea, an extensive wireless SHM system was deployed to a large bridge and was used to successfully identify characteristics such as mode shapes of the bridge deck [62, 9]. SHM applications like these are expected to become more widespread as the technology becomes more affordable and the systems become more advanced.

2.3 Health Monitoring of Wind Turbines

This section presents background on HM of wind turbines, including the motivations for HM of wind turbines and an overview of turbine monitoring in general. Specific systems for monitoring the various components of turbines are discussed in later sections.

Figure 2.3: Fiber optic sensors are being used for SHM of the Parthenon in Athens, Greece, during renovation. Photo by Lee A. Wymore, used with permission from CRD Group.

2.3.1 Motivation

HM of wind turbines has a wide variety of motivations because the data collected by an HM system could have many uses. Clearly, in order for an HM system for wind turbines to be viable, the benefits it provides must outweigh the costs of implementing and maintaining the system, as well as storing, processing, and evaluating the data. However, the benefits of an HM system are complex and can be hard to quantify. Below, these benefits are discussed in terms of maintenance, wind farm management, and other motivations.

2.3.1.1 Maintenance

HM can potentially reduce ongoing operations and maintenance (O&M) costs for each outfitted turbine. In 2011, the National Renewable Energy Laboratory (NREL) estimated that annual O&M cost per MW was around $17,000 for a typical onshore wind project and $46,000 for an offshore project [63]. Unlike traditional power plants, these costs are spread out over many distinct but similar structures (the turbines), and over a relatively large geographic area (the wind farm).

Additionally, turbines are often located in remote areas, such as offshore, requiring time, money, and planning to visit for inspection or maintenance.

Lower maintenance costs are a traditional benefit of HM because HM allows for more efficient maintenance practices. Current maintenance practices for wind turbines are generally a combination of two strategies [64], known throughout the literature by many names, but referred to here as:

1. Scheduled maintenance (SM) - periodically performing a set of prescribed maintenance tasks, such as replacing certain parts and changing gearbox oil.

2. Reactive maintenance (RM) - fixing or replacing components when they fail.

HM enables an alternative, predictive strategy, often referred to as condition-based maintenance (CBM) [65, 66], in which the operator performs maintenance when a component shows signs of damage or impending failure. This strategy reduces the amount of maintenance required and ensures that the maintenance performed is worthwhile. This reduces O&M costs both in terms of labor and materials, and also improves inventory management.

Additionally, if an HM system provides early warning that the component is failing, the replacement can be secured and the maintenance can be performed at convenience. This reduces downtime and increases production. The notion of convenience can also directly reduce costs, especially for offshore sites. NREL estimates that, for a 500 MW offshore wind farm, O&M costs can be cut by over $20 million per year simply by reducing weather-induced delay for maintenance crews to access turbines [64].

The O&M benefits of HM are frequently cited for the drivetrain, generator, bearings, and other mechanical and electrical components of a turbine, but less so for the major structural components. While structural failures of wind turbines are rare, they do occur. The Caithness Windfarm Information Forum (CWIF), an organization dedicated to halting the spread of wind turbines in the Caithness area of Scotland, tracks all publicly known wind turbine accidents. In 2013, CWIF recorded 29 incidents of blade failure, defined as failure which "results in either whole blades or pieces of blades being thrown from the turbine." A total of 280 total instances of blade

failure have been noted since the 1990s. CWIF also lists 145 instances of structural failure dating back to the 1980s, with structural failure defined as "major component failure under conditions which components should be designed to withstand" [67].

Both blade and structural failure are costly. GCube, a renewable energy insurance provider, reported average claims of $240,000 for blade damage and $1.3 million for foundation damage in 2012 [68]. Gearbox, generator, and transformer damage claims were similarly priced. HM could allow damage leading to failures like these to be identified and repaired before failure occurs [69].

From a more general cost/benefit perspective, for HM to see widespread adoption, the economic benefits of a system must at least recover the costs of installing and maintaining that system. A number of studies have examined these economics. For example, Nilsson and Bertling [70] found that converting 47% of RM events into CBM events would recover the cost of an HM system. Besnard and Bertling [71] found that HM was more economical for wind turbine maintenance than offline NDE or visual inspections when crack initiation rates were high and time to failure was short, assuming regular visual or NDE inspections. Van Dam and Bond [72] estimated that an HM system could recover its costs, purely from CBM benefits, up to 70% of the time.

Finally, maintenance-related benefits of HM can only be realized if the HM system is reliable. Any false positive from the HM system will result in waste from the investigation and potential replacement of a part that is still in good condition. On the other hand, a false negative could allow damage that would have been detected in regular inspections to worsen to the point of component failure. If the operator cannot assume the HM system reliably identifies all faults, then regular inspections or part replacements still need to be performed, and the HM system becomes less attractive.

2.3.1.2 Management

HM of wind turbines could also have a number of benefits in terms of wind farm management, such as increasing power production. For instance, HM data could be used to provide input and feedback for a control system used to dynamically pitch the blades, maximizing individual turbine

output. Similar systems are already being used to mitigate blade stresses [73, 74, 75]. Or, data could be shared among turbines to combat wake effects or provide advance notice of weather conditions, optimizing generation for the entire wind farm.

HM could also increase generation by minimizing downtime; if a component fails with no warning, the turbine may not be able to generate electricity for the time it takes to acquire a replacement, hire the needed transportation and installation equipment, schedule a favorable maintenance window, and perform the maintenance. But if the HM system provides advance warning, the turbine could be operated up until the maintenance window. Studies indicate that decreased downtime could have large economic benefits; for example, Nilsson and Bertling [70] found that a 0.43% increase in availability would recover the costs of an HM system.

HM could also be used to quickly verify structural safety [42] and estimate remaining lifetime. This could be useful for bringing turbines back online after an earthquake, lightning strike, or storm. Additionally, this type of information could guide the process of turbine retirement and refurbishment, allowing a turbine or parts of a turbine to safely continue operating beyond their original life expectancy. This data could also allow a turbine to be automatically shut down if stresses exceed a threshold, both preventing failure and helping to mitigate public safety concerns regarding wind turbines. Data about ice formation on blades could be used for a similar purpose [76].

As an example of how an HM system could help with management and with maintenance, Fig. 2.4 shows a turbine that dramatically failed in a severe wind storm after a blade had recently been repaired. Ideally, an HM system would have been able to both verify if the blade was functioning properly after the repairs, and automatically shut down the turbine and feather the blades during the storm to prevent this failure. This figure also illustrates how a single failure in a wind turbine can cause a complete system failure, making the case for HM even stronger.

Figure 2.4: Wind turbine failure from high winds shortly after a blade was repaired. Photo by Green Mountain Power, NREL 16713.

2.3.1.3 Other

The data gathered from an HM system could have a variety of other uses. Strain and dynamic response data could be used to identify structural weaknesses and inform future designs, which could lead to greater reliability or less conservative designs that use less material. An HM system could also be used to monitor environmental conditions that a turbine depends upon, such as permafrost in Alaska and similar locations.

Finally, the type of data gathered from a monitoring system should not necessarily be restricted to traditional HM data. For example, a monitoring system could collect data to help researchers better understand wind farm wake effects, a serious source of power losses [77]. Or, a wildlife monitoring system could trigger a reduction in turbine speed at the approach of a large body of birds or bats, further improving the environmental-friendliness of turbines [78].

2.3.2 Overview of Turbine Monitoring

HM for wind turbines is a developing field. While commercial products for monitoring of some components exist [40], and all components have at least had experimental systems tested, the wind

energy industry has not yet embraced HM on a widespread basis. Still, related research, such as the dynamic characterization testing shown in Fig. 2.5, has been ongoing for years [79].

Figure 2.5: Dynamic characterization testing of a turbine in 1999. The tower is struck with a hammer and the resulting vibrations are measured with accelerometers. Photo by David Parsons, NREL 07388.

As previously discussed, any HM system needs a positive return on investment to see widespread adoption. For wind energy, this poses a particular challenge because of the nature of a wind farm. A turbine may not be as large as a large bridge or skyscraper, but there could be hundreds or even thousands of them to monitor in a wind farm [80]. This could provide advantages such as design reuse and mass manufacturing, but such such advantages are not likely to overcome the current high cost of many kinds of sensing systems that could be used for SHM of wind turbines. Thus, improvements to the basic sensing technology are still important.

Furthermore, mass deployment of a monitoring system requires that installation, setup, and maintenance of the system be simple and inexpensive. Ideally, these tasks could be performed by a technician with minimal training. To maximize the potential uses of acquired data, either sensor placement needs to be standardized so that readings between turbines are comparable, or the system needs to be designed flexibly, such that differences in placement of the sensors between turbines are not a concern.

Another factor for wind turbine monitoring systems is lifetime. Most wind turbines are designed to last at least 20 years, and recent research suggests that they will [81]. A monitoring system would

be expected to last just as long, or longer, as monitoring could be used to extend the lifetime of a turbine if the system is able to judge the turbine still structurally sound. Thus, a monitoring system must be robust and long-lived. It must be maintainable, but given the potential scale of a deployment, it should not require regular maintenance, such as manual data collection or replacement of batteries. Both low maintenance and long life will help increase a system's cost-effectiveness.

A final challenge for wind turbine HM is adoption. As Mobley [44] describes, justifying the installation of any monitoring system is difficult; quantifying its benefits after installation is even harder. Currently, adoption decisions must be made based on a limited pool of data. To aid in the adoption of HM in the industry, research in monitoring for wind turbines should aim not only to increase the attractiveness of HM systems, but also to increase the amount of cost/benefit data available for decision-makers interested in implementing a monitoring system.

These are some of the general challenges of designing an HM system for wind turbines. In the following sections, specific challenges and the current state of monitoring for each major component of wind turbines, as depicted in Fig. 2.6, will be discussed. The discussion assumes an upwind horizontal axis wind turbine (HAWT), the most common type of turbine in utility-scale wind today. This type of turbine is used both onshore and offshore, and the challenges of both will be addressed. While this discussion focuses on megawatt-scale turbines, due to their role in utility generation, many of the challenges and techniques are applicable to smaller turbines as well.

2.4 Nacelle

The nacelle, positioned on top of the tower, houses the mechanical and electrical equipment used for generating electricity. A cutaway view of a nacelle is shown in Fig. 2.7. The rotor blade assembly's hub attaches to the nacelle, and the nacelle yaws, or rotates around the axis of the tower, to point the blade assembly into the wind. Inside the nacelle, the drivetrain connects the blade assembly to the generator, often utilizing a gearbox to scale the rotational speed of the shaft connected to the blade assembly up to the rotational speed required by the generator. Other major

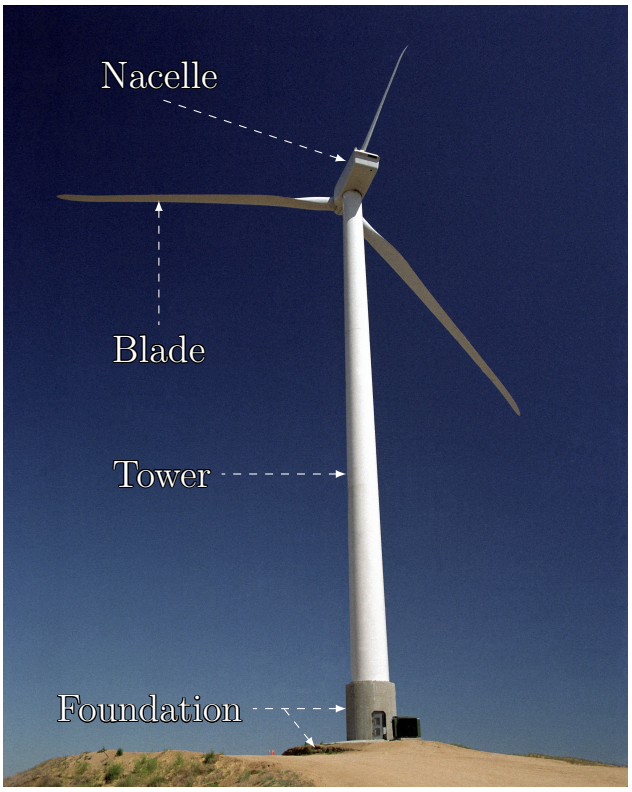


Figure 2.6: Major components of a HAWT, as they are discussed here. Adapted from a photo by Warren Gretz, NREL 11148.

components housed in the nacelle are the yaw control system, which keeps the turbine pointed into the wind, and the bearings for the yaw system and for the main drive shaft. In general, the components housed in the nacelle can be monitored with well-developed CM techniques similar to other industries, as discussed in Section 2.2.1. The power converter used to condition the generated electricity should also be monitored, and may be housed in the nacelle or at ground level.

The gearbox is a component of particular concern. Gearboxes have been identified as a leading contributor to turbine O&M costs, and gearboxes in general have not been meeting their design lifetimes, to the point where NREL has established a special program, the Gearbox Reliability Collaborative, to address the problem [82]. Eric Bechhoefer from Renewable NRG Systems illustrates the ideal case for how CM can help: an uptower repair for a gearbox that was identified as needing repair via CM might cost $50K, whereas the fix if the gearbox had instead operated to

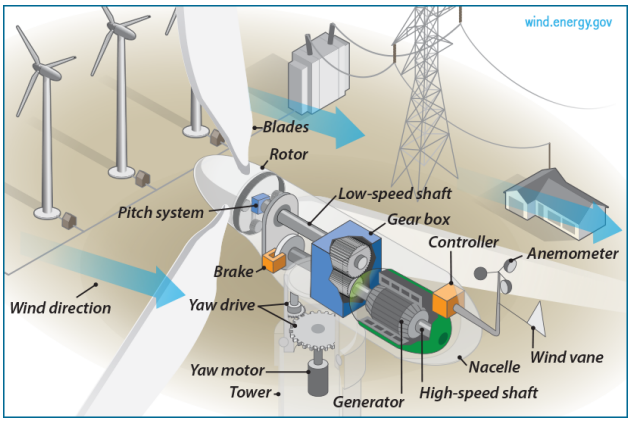


Figure 2.7: Cutaway illustration of a wind turbine nacelle. Image from the Office of Energy Efficiency and Renewable Energy, U.S. Department of Energy.

failure could be as much as $400K, including $250K for the new gearbox and $150K for the crane to swap the new gearbox for the old [83]. Fig. 2.8 shows a crane being used during a gearbox installation, which includes first bringing down the blade assembly. Direct-drive turbines with no gearbox are also available, but the permanent magnet generators used in these designs tend to be more expensive than the induction machines used in geared systems.

2.4.1 Existing Systems

Monitoring of the components housed in the nacelle is the most well-developed and market-penetrating type of wind turbine monitoring. For instance, supervisory control and data acquisition (SCADA) systems that are built into turbines for the purpose of controlling electricity generation are increasingly also used for basic CM purposes, often through software packages that analyze the data already being collected. For example, changes in electrical output of the generator can indicate impending bearing failure in the drivetrain [84, 85] or asymmetries and eccentricities due to other mechanical failures in the generator [86]. As the data required for this type of analysis is already being gathered by SCADA systems, these methods have potential as an inexpensive solution for CM [87]. For a more complete discussion of commercial SCADA systems and analysis packages, see [41].

Figure 2.8: Installing a gearbox. Photo by Jeroen van Dam, NREL 19257.

Another approach to CM is to deploy a system that uses sensors such as accelerometers to collect data more directly related to HM. More and more, this sort of commercial CM system is being integrated into the SCADA system [88], either borrowing data from the SCADA system, or allowing CM results to be integrated into the SCADA displays. Ideally, CM and SCADA systems will continue to merge over time, eventually resulting in one comprehensive system that uses collected data to both provide control feedback and perform HM.

Most commercial CM systems for wind turbines utilize some form of vibration monitoring, similar to established methods in other industries. One notable difference and challenge for wind turbines is the variable speed operation of many types of turbines. While traditional fast Fourier transform (FFT) methods have been developed for machinery that runs at constant speed, the operational speed of a large wind turbine typically depends on the speed of the wind, so data from

two points in time may not be directly comparable. Some commercial CM products use other techniques such as AE and MCSA, discussed in Section 2.2.1, and many also implement basic oil monitoring. An extensive list of commercial systems for CM of wind turbines is given in [40]; a few of these products are discussed here only to illustrate the capabilities of a typical system, and not to evaluate the systems in relation to each other.

An example of a commercially available product is GE's Bentley Nevada ADAPT Wind system [89], which GE claims has saved customers up to $3,000 per turbine per year for new units. The system focuses on monitoring of the gearbox, generator, and main bearing, all using accelerometers connected by wires to a processing unit housed in the nacelle. The ADAPT system also uses the turbine's control system as an input, and the ADAPT system can be integrated into the turbine's SCADA system. Tower sway and gearbox oil condition can also be monitored with the ADAPT system.

Many manufacturers have similar CM systems. Turbine manufacturer Siemens has a system [90] which they specifically note can automatically shut down a turbine, if needed [91]. CM systems for wind turbines are also available from third parties, such as Renewable NRG Systems' TurbinePhD [92]. Some systems have even taken a step past HM and into automated maintenance; for example, SKF's WindCon online CM system can inform SKF's WindLub system when bearings are low on lubrication, and the WindLub system will automatically deliver grease to the bearings [93].

2.4.2 Future

As discussed, nacelle HM is relatively well-developed and commercialized. One potential area for improvement is more advanced oil analysis. In 2011, Hamilton and Quail [46] reviewed the potential use of various oil analysis technologies for online wind turbine gearbox monitoring. They recommended a combination of techniques be used to achieve sufficient accuracy and diagnostic detail while utilizing cheaper and smaller-scale technologies. For monitoring oil quality, they recommended a combination of fluorescence spectroscopy, Fourier transform infrared spectroscopy (FTIR), solid state viscometers, and photoacoustic spectroscopy. For analyzing wear particles,

they recommended ferrography, particle counting, fluorescence spectroscopy, and a simple electrical constant sensor. They believed these technologies could be integrated into one sensing system.

Currently, online oil monitoring is mostly used to detect changes in oil quality or particle count, and offline analysis of a spot sample is used to diagnose the cause of the change [94, 49]. The oil monitoring capabilities of the commercial CM systems listed in [40] are essentially limited to trending changes. However, technologies such as FTIR and linear variable filter-based mini spectrometers have high potential for more detailed online diagnosis in the future [88].

Other improvements to nacelle HM could include wireless monitoring systems. In addition to reducing cable clutter, wireless solutions allow for easier installation and use in locations where cabled solutions are not feasible. This technology is available and ready for the transition to industry. For instance, as far back as 2006, BP, Intel, Rockwell, and Crossbow explored the use of a WSN for CM in the engine room of an oil tanker. They found the wireless performance to be satisfactory and their trial system to be robust in the harsh conditions of the ship [95]. However, sustainably powering WSN nodes is a challenge and an area of ongoing research.

Another area for continued research is cost-effective sensor technology. Since the attractiveness and extensiveness of a monitoring system is limited by its price, the development and testing of cheaper sensors for such systems is imperative. For instance, accelerometers based on MEMS can be a small fraction of the price of conventional accelerometers, but these accelerometers currently suffer from noise and resolution constraints. Initial research has shown that MEMS accelerometers have potential for CM of machinery [96, 59], but the use of such technology has yet to be proven in the field.

2.5 Foundation

Onshore wind turbine foundations are typically steel rebar-reinforced concrete, as shown in Fig. 2.9. A number of different foundations and foundation concepts exist for offshore turbines. For both onshore and offshore turbines, the choice of foundation depends on the location and

Figure 2.9: A typical onshore wind turbine foundation, before backfilling. Photo from the Bureau of Land Management, US Department of the Interior.

the environmental conditions. For instance, soil quality and strength affect the size and shape of onshore foundations, whereas water depth is a deciding factor for offshore turbines.

Therefore, a challenge of foundation monitoring is the wide spread of foundation types, even for a given turbine model. Another challenge is that, for onshore turbines, a monitoring system likely needs to be installed during construction. Turbine foundations are often poured by local concrete companies, and installing a sensor system during construction may require extra coordination between these companies and technicians. Finally, for both onshore and offshore turbines, durability and accessibility of sensors is a concern. The long-term survivability of sensors embedded into concrete has yet to be proven [97], though some sensors, such as a MEMS cantilever-based temperature and humidity sensor, have shown promise [98]. Should such a sensor fail, accessing it for maintenance is impractical. Offshore foundations present similar challenges, as sensors may be immersed in salt water, requiring special packaging and diver-based maintenance.

Offshore turbine foundation monitoring systems can build on the experience of monitoring for other offshore platforms, such as offshore oil rigs. However, turbine structures tend to be less massive than these platforms, while the nature of a turbine guarantees the structure will experience large forces from the wind, as well as the waves. Turbine structures are thus designed more for dynamics than bearing capacity [99], and this difference affects the goal and methods of a monitoring system.

2.5.1 Existing Systems

In general, turbine foundation HM is currently limited to research activities and spot applications. Offshore foundation monitoring has received more attention than onshore monitoring, likely due to the more challenging conditions, the more complex structures, and the large number of questions still surrounding the design and deployment of offshore turbines. One example of offshore turbine foundation SHM began in 2010, when a specific failure mode in offshore monopile foundations was detected. The grout between the pile foundations and the tower's transition piece was failing, causing the tower to slip downward by more than 25 mm until caught by supporting brackets used during construction [100].

Strainstall Monitoring installed SHM systems that consist of strain gauges, displacement sensors, and accelerometers. Stresses on the brackets are monitored and fatigue life of critical areas is calculated. Additionally, natural frequencies of the tower are tracked, since a change in tower frequencies could indicate a foundation problem, and some of the systems use inclinometers to track the angle of the tower as well. The sensors are wired to a DAU, and in most cases, data is transferred offsite over a broadband connection. However, in areas where such a connection is not available, hot swap hard drives are collected periodically from the DAU. While originally installed as a reaction to a specific failure mode, these systems have been successful enough that Strainstall has been requested to install them on new turbines as well [101].

An extensive monitoring system has been installed at the Belwind offshore wind farm in the North Sea. The system is an Offshore Wind Infrastructure Lab (OWI-Lab) project, in conjunction with BruWind, Zensor Corrosion Control, and Vrije Universiteit Brussel [102]. The system includes corrosion monitoring for the foundation using sensors from Zensor [103], as well as load and displacement monitoring for grouted connections and dynamic monitoring for the global structure.

Brincker and Ibsen [104] instrumented a Vestas 3 MW turbine with accelerometers in order to assist a finite element model (FEM) in performing fatigue analysis on a new offshore "bucket" foundation design, with the goal of aiding the design of the foundation and its installation process. The accelerometers used were on the order of a thousand dollars each. This study did not include

any submerged sensors, but the type of data used in this study has been gathered underwater in the past using an FBG-based system [105].

In general, in the harsh and noisy environment of offshore foundations, simple solutions that monitor a particular failure mode may be more immediately practical than global monitoring solutions. For instance, for a foundation where posts are inserted into pre-piled jackets in the seabed, the performance of the grouted connection between the post and the jacket is a concern [99]. Instrumenting such a connection with a subsea extensometer is feasible with current technology, and the data would be easy to interpret using a simple threshold-based trigger.

Another potential failure mode of offshore foundations is excessive scouring, where the water erodes the seabed around the foundation's pile to the point of structural instability. Strainstall has a scour monitoring system that tracks the progress of a magnetic collar down the length of a metal tube embedded in the seabed. As the soil is eroded, the collar slips down further, triggering an alert whenever it passes one of the sensors distributed along the length of the tube [106]. Michalis et al. [107] proposed a similar solution, but with no moving parts. A probe with sensors distributed along its length is embedded into the seabed. The sensors have two steel rings that, if buried, have soil between them. If the sensor is uncovered, the capacitance between the rings changes, indicating scour progress.

Onshore foundation monitoring has received minimal attention, though failures do occur. While these failures are often because of installation errors such as improperly tightened tower anchor bolts [108] or improperly settled concrete [109], a monitoring scheme could provide early warning of these issues. Anchor bolt monitoring could be useful to reduce manual inspections, as research has suggested that bolt tension should be checked once every two years [110]. Additionally, foundation monitoring could contribute to a quick evaluation of structural integrity after an earthquake or high-wind conditions, as well as provide peace of mind regarding aging turbines still in service.

Onshore foundation monitoring systems that have been deployed have mostly focused on research of loads and fatigue for design purposes, not SHM. Renewable Energy Systems (RES) Americas and NREL [110], in conjunction with Canary Systems [111], instrumented a gravity base foun-

dation for an 80 m tower. The sensors included earth pressure cells to measure contact pressure between the foundation and the soil, strain gauges to measure strain on the foundation's reinforcement steel, and bolt load cells to measure the tension of the anchor bolts. The sensors were wired into a DAU housed in the tower, and the data was sent to an offsite server over a cellular connection. An example result from the study was that the vertical pedestal reinforcement steel experienced significant load, validating its presence as a critical part of the foundation.

2.5.2 Future

As further foundation failure modes are identified, solutions specific to these modes can be developed. For example, Currie et al. [112] have begun an SHM study for onshore foundations. They describe a common failure mode for embedded can-type foundations, where the protective coating between the tower and foundation cracks, allowing moisture to enter and erode concrete around the steel can embedded into the foundation. This results in potential vertical movement, up to tens of millimeters, of the tower. The authors proposed a simple monitoring solution to detect this vertical movement at the foundation level, using either infrared, linear variable differential transformer, or Hall-effect sensors. While this solution is straightforward, this particular failure mode may also be detectable by accelerometers attached to the tower for tower monitoring.

Failure mode-based monitoring can be useful, but more general concrete monitoring could also be helpful, especially for onshore foundations. Concrete monitoring systems have been designed and tested for other types of civil infrastructure, such as roads, bridges, and tunnels. Such monitoring has been estimated to provide large reductions in the operating cost of these structures [113]. Onshore turbine foundation monitoring has thus far been largely strain and vibration sensing, but concrete monitoring systems may also measure the temperature or moisture profile of the concrete. These factors contribute to effects such as creep, shrinkage, and deformation [114]. Monitoring of these factors is typically used to expedite the construction process, and commercial systems for temporary temperature and moisture monitoring exist [115].

Another possible method for concrete monitoring is to use AE to track crack propagation [116, 117, 118]. Combining AE with temperature and moisture sensing can not only allow for monitoring of the structure, but lead to better understanding of the structure's response to its environment [119]. The freeze-thaw cycle, for example, is known to contribute to concrete degradation, as is contact with seawater [120].

A concern particular to reinforced concrete is corrosion of the steel reinforcement bars. Polder et. al [121] demonstrated that early detection of corrosion could provide large cost savings. Early detection can be accomplished with a variety of embedded sensors [120]. Since corrosion is an electrochemical process, embedded reference electrodes can detect corrosion via changes in electrochemical potential. Corrosion risk can be monitored with the well-proven "anode ladder system," which tracks the progress of corrosive conditions using dummy steel bars embedded at various depths in the concrete. These conditions can also be tracked using moisture and chloride sensors.

In terms of wireless advancements, both offshore and onshore foundations present challenges. Underwater sensor networks that use acoustic signals for communication are a current research topic [122]. For onshore foundations, the challenge is concrete. Radio frequency (RF) signals such as those used by WSNs experience fast attenuation when traveling through concrete [123]. This means that high transmission power is required to successfully transmit an RF signal through concrete, and a sensor node embedded in concrete cannot be easily accessed for a battery change.

In an attempt to circumvent this issue, Ong et al. [124] developed a passive wireless moisture sensor suitable for embedding into concrete. The sensor is based on an inductor-capacitor resonant circuit whose resonant frequency can be remotely queried using a loop antenna [125]. As moisture increases, the dielectric constant of the capacitor's insulator increases, changing the capacitance and the resonant frequency of the circuit. This type of sensor is practical for embedding into concrete because it is robust and requires no power source. However, the required querying unit and the limited querying range make continuous online monitoring with this type of sensor challenging. Radio-frequency identification (RFID) sensors of a similar nature have also been tested [126], but suffer from the same challenges for continuous monitoring.

2.6 Tower

The wind turbine tower supports the nacelle and blade assembly and allows access to the better wind resources typically found at taller heights. Together with the foundation, the tower must bear not only the weight of the upper turbine components, but also the horizontal forces of the wind on the turbine blades and the rotational forces of the nacelle yawing into the wind. Clearly, the tower is a crucial part of the turbine structure; failure of the tower results in total failure of the turbine. HM can help prevent such failure.

Currently, the most predominant type of wind turbine tower is the tubular steel tower, made of large steel rings welded together into long sections, shown in Fig. 2.10, which are bolted together during assembly. The World Steel Association claimed in 2012 that about 90% of all turbine towers were tubular steel [127]. These towers are typically around 80 m to 100 m tall, but, in order to produce more power, the trend is toward even taller towers. This has led to alternative tower designs, such as Iowa State University's Hexcrete tower concept [128]. Operators installing GE's 2.75 MW wind turbines currently have a choice of a tubular steel tower of up to 110 m, a 139 m steel/concrete hybrid tower, or a 139 m space frame tower [129].

This variety of designs could lead to a variety of HM systems. The space frame tower concept resembles the steel lattice towers of the early utility-scale wind energy years, but provides an enclosed internal space and is assembled with special bolts that do not require the ongoing maintenance that plagued earlier lattice designs [130]. Thus, in many ways, monitoring the space frame tower should be similar to monitoring a tubular steel tower. Similarly, monitoring a concrete/steel hybrid should be similar to a combination of tubular steel monitoring and concrete foundation monitoring. Therefore, the remainder of this section will focus on tubular steel towers, the dominant product in the market today.

A steel tubular tower is relatively easy to globally monitor using VBDD, but cost is a concern. Sensors can be placed around the weatherproof interior of the tower with minimal trouble. However, the vibrations of such a large structure are very low frequency, with first natural frequencies below 1 Hz [131]. Traditionally, more expensive sensors are required to obtain accurate measurements

Figure 2.10: On-site assembly of steel wind turbine tower sections. Photo by Dennis Schroeder, NREL 20843.

at such low frequencies. Once such measurements are acquired, damage can be diagnosed using a variety of methods, as noted in Section 2.2.2. Thus far, systems have tended toward classic methods such as trending natural frequencies or modal parameters, sometimes with the aid of an FEM. Strain and tilt measurements can also be used for fatigue analysis and lifetime prediction.

2.6.1 Existing Systems

Aside from tower sway monitoring built into some turbine CM systems, tower HM has seen little commercialization. However, a number of research systems have been created. One of the most complete turbine SHM systems was a tower monitoring system designed and implemented

by Smarsly et al. [132]. The system was installed on a 65 m turbine tower in Germany in 2009 and had been operating continuously for several years at the time of writing of [132]. The sensor array includes nine accelerometers, six inductive displacement transducers, temperature sensors for compensation, and an ultrasonic anemometer mounted on a separate mast next to the tower. Wired DAUs feed the data from the sensors into the server located inside the turbine tower, which also receives data from the turbine's SCADA system. The server periodically sends the data over a DSL connection to an offsite server, where the data is converted and stored in SQL tables. From here, the data is available for remote access via a database connection or a web interface.

The monitoring software extracts modal parameters from the measurement data and uses them to update an FEM. Artificial damage is introduced into the FEM, and the resulting response profile is recorded in a damage catalog that can be used for quick diagnosis in the event that real damage occurs. The stochastic loading data of the structure can also be used for computing failure probabilities and thus lifetime estimates [133]. Finally, the system also features extensive self-monitoring and fault detection to make it as reliable and self-maintaining as possible [134].

Rolfes et al. proposed the Integral SHM-System [135], which compares dynamic stress to vibrational velocity to detect stiffness reduction in a tower. Measurements are used to produce a validated FEM that is used to diagnose damage location and severity. Swartz et al. [136] tested the instrumentation system, composed of eight accelerometers and two strain gauges, on two turbine towers in Germany. They were successfully able to identify the first three modes of a 78 m Vestas tower. The accelerometers used currently cost hundreds of dollars each. The sensors were connected to DAUs that communicated wirelessly with a central unit.

The OWI-Lab monitoring project uses a similar array of accelerometers to perform advanced operational modal analysis (OMA) on an offshore turbine tower. Modal parameters are automatically extracted from the data and tracked over time [137].

Brincker and Ibsen's 2003 foundation study [104] also included tower instrumentation. The gathered acceleration measurements were integrated twice to find displacements, which were used to identify mode shapes. These were then used in a finite element analysis, and the resulting model,

to be used in fatigue analysis, was able to calculate displacements and stresses anywhere in the structure. Similarly, Bang et al. [138] mounted FBG sensors on the inside shell of a 70 m tower and used the online strain measurements to find mode shapes of the tower.

Guo et al. [139] analyzed existing tower vibration data from a single up-tower accelerometer using the nonlinear state estimation technique (NSET), a nonparametric modeling technique that uses past observations to predict future observations, based on the input conditions. If the model prediction deviates significantly from measured values, damage could be present. They found that the model predicted vibrations well, and an offline application of the method was able to detect an anomaly in the data shortly before the turbine was shut down to fix an asymmetrical blade condition. This result illustrates an interesting opportunity in wind turbine HM: the components in turbine structures are so tightly coupled that monitoring one component can reveal problems in another. However, this tight coupling is also a challenge, as it can introduce non-random noise into measurements.

2.6.2 Future

The largest obstacle for widespread turbine tower monitoring is likely the cost/benefit tradeoff. Traditional monitoring systems are expensive, and tower failure is rare and often caused by the foundation or the blades. Therefore, tower monitoring is currently motivated by dynamic characterization as often as damage detection. However, the possibility of cross-component monitoring, such as that discussed regarding [139] in the previous section, may make tower HM more attractive, especially with decreasing sensing system costs.

Some non-traditional sensing systems have been used for towers, and an innovative deployment of these systems could provide continuous HM. Chen and He [140] were able to measure the first natural frequency of a tower using a microwave radar system from over 1 km away from the tower. While price may prevent a permanent monitoring system using this technology from being feasible, the technology does have the advantage of being able to monitor several turbines at the same time, as long as they are all within its panoramic line of sight. Other similar remote monitoring

technologies, such as light detection and ranging (LIDAR) systems, have also been proposed for turbine monitoring [141].

Few studies have used non-VBDD methods for tower SHM. Benedetti et al. [142] proposed using perturbations in the local strain field as a damage indicator and validated the method on an FEM of a turbine tower. While straightforward, this method does not provide the convenient global damage detection of VBDD, requiring strain sensors in the area immediately surrounding the damage. A wireless deployment could make this technique more feasible.

Since the towers are steel, electromagnetic or magnetostrictive testing could be a possibility, but these methods currently lack feasibility as continuous monitoring solutions. Optical methods such as image processing could also be useful for detecting surface damage, such as to the tower's anti-corrosive coating. These methods have been explored for turbine blades, but little research has been done into monitoring surface damage of towers, again likely due to the cost/benefit tradeoff.

2.7 Blades

A wind turbine's blades are responsible for transferring the horizontal force of the wind into the rotational force that drives the turbine's generator. Modern megawatt-scale wind turbine blades are typically over 40 m long and made of layers of glass fiber reinforced polymer (GFRP) material bonded together using a resin. GFRP offers a cost-effective solution to the weight and strength tradeoff that blade designers must balance. Longer blades may use carbon fiber reinforced polymer (CFRP) to increase strength and reduce weight, but the cost of CFRP can be prohibitive. Balsa wood adds extra bending stiffness to the shell of the blade [143].

Blade manufacturing is largely still a manual process, as shown in Fig. 2.11. The blade structure consists of three parts: two outer shell halves, and an inner shear web. The shear web creates an I-beam inside the blade, allowing for more strength with less material and weight. The blade's cross-section shape, or airfoil, creates the lift force on the blade that produces torque, turning the shaft in the nacelle and generating electricity. The blades connect to the hub with bolts that screw into the root section of the blade. The root itself is a cylindrical structure made of layers

of GFRP. A gel coat over the outside of the blade provides protection from minor impacts and moisture, which can cause delamination, the separation of the layers of FRP. Within the hub, a pitch-control system rotates each blade to change its angle-of-attack to the wind, either to increase energy capture, reduce load, or provide aerodynamic braking [144].

Figure 2.11: Manufacturing of a wind turbine blade. Two outer shell halves can be seen. Photo from SNL.

Blade HM is particularly challenging because the aerodynamics of the blades are critical for efficient power production, so an SHM system must not interfere with these aerodynamics. Additionally, blades rotate and experience other stochastic loads, such as wind and yaw forces, which creates a noisy sensing environment. An SHM system may monitor various blade characteristics, but distinguishing between damage and changes in the environment can be especially difficult for blades. Fluctuations in temperature, weather, and wind speeds can all cause false positive damage alarms. Additionally, a blade monitoring system's hardware is likely to be exposed to harsh environmental events, such as wind gusts and lightning strikes. Finally, cost presents a particularly large obstacle for blade SHM, because the technologies best suited for monitoring blades tend to be expensive.

2.7.1 Existing Systems

Commercial blade monitoring systems have mostly been used for load monitoring applications. As early as 2005, an article [145] discussed two different commercial systems for blades. The two systems, developed by Smart Fibres Ltd. and LM Wind Power (then LM Glasfiber), use FBG sensors to measure the strain in the blade. Though marketed as load monitoring systems, these systems also possess the potential to detect some types of blade damage before failure. Additionally, the LM claims the system can reduce loads by 10-30%, potentially increasing blade lifetime.

The 2014 report by Crabtree et al. [40] lists three commercially available fiber optic systems for blades. Fiber optic sensors are capable of detecting temperature, displacement, acceleration, and strain [146]. Manufacturers can install the sensors during manufacturing, which could provide quality-control during the manufacturing process. Alternatively, wind plant operators can retrofit sensors onto existing blades. Many operators consider fiber optic systems expensive, but the combined purposes of load monitoring, damage detection, and quality control may make these systems more attractive as blades become larger and more expensive.

The Crabtree report also lists several VBDD systems, including one composed of accelerometers that are bonded directly to the blade exterior and connected by wires to a DAU in the hub that communicates wirelessly with a receiver in the nacelle [147]. Similar sensors can also be placed inside blades. These sensors can monitor the motion, path, and vibrations of the blades during operation. Changes in any of the monitored characteristics can indicate damage in the blades or rotor hub. Yang et al. [148] note that VBDD systems can provide useful data at low cost, but are limited by the lack of specific information provided by the data. VBDD systems cannot give information concerning damage type or location. In addition, variability in the wind can cause features in the data that are unrelated to blade or hub conditions.

Due to cost and technical difficulty compared to laboratory settings, few field tests have been performed for SHM of blades. In one example, Schroeder et al. [149] documented the use of a fiber optic system in a blade over the course of two years. The system includes one temperature and six strain sensors. The sensors were attached to the interior of the blade and were capable of detecting

strain characteristics over the course of operation. Though the system was not designed for SHM, this type of strain data could be used for fault detection and active safety control. In another example, Blanch and Dutton [150] performed field tests with an AE system. The sensors were attached along the length of the outside of the blade. The tests were performed while the turbine was held static and the blade was artificially loaded. Due to this test method, the system is not practical for commercial use. Online tests were also performed, but the results were inconclusive.

Sandia National Laboratories (SNL) has a SMART Wind Turbine Rotor project [151] that uses sensors to provide feedback to an active aerodynamic control system in the blade. In a field test, the sensing array consisted of accelerometers, metal foil strain gauges, and fiber optic strain and temperature sensors, all of which were housed inside the blade. The blade contained several modifications to protect the sensors from electrostatic discharge, including copper mesh in the skin and a conductive gel coating for the outside of the blade. This protection was required not only for lightning, but also for the electric charge created from the friction of the blade moving through the air.

A research group in Greece has had success in field monitoring with an off-the-shelf AE system applied to a turbine blade [152]. Wired AE sensors connect to a wireless transmitter in the hub, which communicates with a wireless router at the bottom of the tower. This router then connects to the Ethernet network of the turbine, allowing the data to be transmitted over the Internet.

Contrary to the limited number of field tests, numerous tests have been done in laboratories. In a major example, SNL and NREL coordinated with other institutions on tests of a variety of SHM systems for a 9 m test blade [153].

- 30 strain gauges were installed on the gel-coat surface of the blade.
- 24 PAC Model R6I AE sensors were mounted over critical areas on the blade and interfaces inside the blade. The AE system noted significant AE events and then clearly followed the evolution of blade failure.

- NASA implemented an SHM system that included a macro-fiber composite (MFC) actuator and three sensors on one side of the blade and an MFC actuator and two sensors on the other side. The data collected was noisy, but trends in the data were able to identify cracking events.
- Purdue University implemented an array of high sensitivity triaxial accelerometers, low-frequency capacitive accelerometers, piezoelectric actuators, and force sensors distributed over the surface of the blade. The system was capable of detecting and identifying different types of damage.
- Virginia Tech implemented an impedance-based system that consisted of six self-sensing MFC actuators and an impedance analyzer. Though previous tests of this system on a different blade were promising, in this case, the VT test system detected no damage because the sensors were not located close to the failure.

In a different test, NREL, Sandia, and a blade manufacturer worked together to test a blade with bonded piezoceramic patches that both generate and receive stress waves [154]. The raw signals received from sensors near the final failure showed recognizable changes in stress wave parameters. Since this method requires generation of test waves, the interval between tests must be chosen carefully to ensure that damage is detected as early as possible.

Risø National Laboratory compiled test information concerning SHM using AE and fiber optic sensors. Several tests determined the detection and identification capability of the systems [155]. Optical methods such as photogrammetry and laser interferometry have also been tested [131], but these methods suffer from high cost and poor feasibility as continuous monitoring systems.

2.7.2 Future

HM applications outside of wind energy show the capability of alternative methods for monitoring composites like those used in wind turbine blades. Some examples include advanced online AE and active ultrasonic methods developed for composite pressure vessels, solid electrolyte electrochemical sensors for moisture determination in fiberglass and CFRP, and fiber optic sensors

using evanescent waves to measure moisture content demonstrated in flight tests on aircraft [156]. A piezoelectric-based embedded sensor/actuator network applied to an aircraft wing for structural integrity monitoring has also proven to be effective. The system employs an ultrasonic guided wave to detect, locate, and monitor damage [156]. Finally, the wind energy industry already uses ultrasonic testing techniques for blade evaluation, but not in an online SHM scenario [157].

Wireless sensors for blades have been proposed [158], and commercial blade monitoring systems frequently use wireless communications between the nacelle and the DAU housed in the hub. However, these systems have not addressed the most difficult challenges of using multihop WSNs with blade monitoring, such as finding a sustainable power source for sensor nodes, or a solution for placement of nodes such that they can be accessed for maintenance, yet do not interfere with blade aerodynamics or structural integrity.

2.8 Conclusions

HM of wind turbines is in its infancy, but growing strong. The various components of a turbine each present unique challenges to overcome, and the maturity of the HM systems available for the different components varies. CM systems for the mechanical and electrical components inside the nacelle are common, especially in new turbines, but SHM systems for the foundation, tower, and blades are mostly limited to experimental and research installations.

Future work in nacelle monitoring should focus on lowering the cost of the CM systems and improving the usability of the systems for operators. Work in foundation and tower monitoring should include identifying the failure modes of these components and investigating the cost/benefit tradeoffs of monitoring them, with offshore scenarios being a priority. Blade monitoring research should focus on decreasing system cost and improving feasibility for long-term, continuous monitoring, with field studies being preferable.

For wind turbines in general, most SHM-related field studies have focused on instrumentation, with few creating complete SHM solutions. Making the transition from a simple instrumentation system to a useful HM system is challenging, but necessary for the benefits of HM to be realized.

This transition requires multi-disciplinary work and knowledge, but should be a focus for future research.

One of the barriers between simple instrumentation and useful HM is the difficulty in collecting, storing, and processing the massive amount of data for a wind farm. Ongoing HM should not be labor intensive, nor should it require an expert, so whatever data processing methods are used should produce straightforward and easy-to-interpret results.

The massive amounts of data generated can also be a powerful tool, and can aid in damage detection by statistical pattern recognition, a type of technique that some researchers consider to be the future of HM [51][42]. Cross-referencing data across wind turbines may lead to effective analysis methods, but most SHM systems are experimentally deployed on a single turbine. Future work in HM of wind turbines should explore the unique possibilities of processing data from hundreds of these highly similar structures. Additionally, research should focus on making HM systems easier to install and use, such that a wind farm-scale deployment would be feasible.

Finally, economic benefits are likely to provide the greatest motivation for HM adoption. Though the potential for cost-saving is widely acknowledged, analytical studies have shown mixed results for current technologies. Therefore, research in HM for wind turbines should also focus on reducing the cost of the HM system and its installation and maintenance, as well as on increasing the probability of detection of faults, so that the economics of HM systems are more attractive. Additionally, research should strive to produce case studies with cost/benefit data usable throughout the research community and industry.

CHAPTER 3. LOW-ENERGY LINK LAYER

3.1 Introduction

The Internet of Things (IoT) has widely been envisioned as a transformative computing paradigm, seeking to connect everything from everyday objects to industrial machines. One possible use of the IoT is to create large networks of tiny wireless sensors and actuators for advanced, high-fidelity monitoring, feedback, and control of natural and human-made environments. However, this application suffers from the classic drawback of wireless sensor networks (WSNs): computing devices require energy to operate.

Such devices are typically powered by batteries or supercapacitors, which may (or may not) be recharged, slowly and intermittently, by energy-harvesting systems. In order to operate for extended lengths of time—ideally, years—these devices must minimize their energy consumption. Because wireless radios consume a relatively significant amount of power on these embedded platforms, one of the ways this minimization is achieved is through low-energy networking, such as duty-cycled medium access control (MAC) protocols.

Duty-cycled MAC protocols have long been a subject of research in WSN and IoT applications. However, we argue that state-of-the-art duty-cycled MAC protocols do not satisfactorily meet the demands of the IoT. In particular, IoT MAC protocols should have the following characteristics.

- *High performing:* IoT MAC protocols should achieve low delay and high reliability while minimizing energy consumption, even under moderately-high traffic loads.
- *Resilient to interference:* IoT MAC protocols should be robust in dynamic and noisy environments, including scenarios with hidden nodes (senders that are not within range of each other but interfere at a receiver).

- *Polite:* IoT MAC protocols should have a small channel footprint in order to cause as little external interference as possible. They should occupy as few channels as possible, as little as possible.

- *Device-independent:* IoT MAC protocols should be easily ported to new devices, and devices should be easily interoperable using these protocols.

With these characteristics in mind, we have examined ContikiMAC [18], a state-of-the-art sender initiated asynchronously duty-cycled MAC protocol, and RI-MAC [19], a classic receiver-initiated protocol. We have drawn inspiration from these protocols' successes, and learned lessons from their shortcomings, to create RIVER-MAC, an asynchronously-duty-cycled MAC protocol with a Receiver-Initiated, (Very) Efficient Rendezvous. RIVER-MAC's rendezvous is clear channel assessment-based (*CCA-based*) and can reduce idle listening for senders by an order of magnitude. Additionally, RIVER-MAC uses a *beacon train-based collision resolution* scheme to reduce contention between receiver nodes, a scenario that has generally been overlooked in receiver-initiated protocol design. With these features, we believe RIVER-MAC is better able to meet the demands of the IoT.

3.2 Related Work

Energy efficiency through duty-cycling has been a constant research topic for WSNs for well over a decade, as summarized in surveys such as [159]. At the link layer, protocols such as B-MAC [16], X-MAC [17], and ContikiMAC [18] have achieved a progressively more efficient *asynchronous rendezvous* between sender and receiver, without the need for synchronization or explicit scheduling between nodes, using a technique called *low-power listening* (LPL). In LPL, nodes periodically wake and listen to or sample the channel for a short interval. A sender continually transmits a jamming signal or a packet train until its receiver responds or acknowledges the packet. Such protocols are also called *sender-initiated*, and we discuss their shortcomings in Section 3.3.

Receiver-initiated protocols [160, 161, 19, 162, 163, 164] use *low-power probing* (LPP) to achieve similar results. In LPP, nodes periodically broadcast a probe, or *beacon*, packet to advertise their availability as a receiver. Senders simply listen for a beacon from their intended receiver. Examples of receiver-initiated protocols include RI-MAC [19], the classic receiver-initiated protocol, and A-MAC [164], which uses hardware acknowledgements of broadcast packets, and also multiple channels, to increase efficiency. Receiver-initiated protocols in general suffer from excessive *idle listening*, as we discuss in Section 3.3. Some work, such as [162], has explored the use of pseudo-synchronous mechanisms to reduce this idle listening; in contrast, we propose a purely asynchronous mechanism.

Synchronously duty-cycled MAC protocols, in which nodes are synchronized so that transmissions can be scheduled, have also been studied. Early synchronous protocols include S-MAC [165, 166] and T-MAC [167]. More recently, the use of IEEE 802.15.4e's Time Slotted Channel Hopping (TSCH) protocol has become more practical for WSNs and the IoT through autonomous schedulers such as Orchestra [168]. We acknowledge this as a promising development, but our focus is on flexible, single-channel protocols that do not require the overhead and complexity of synchronization and scheduling.

An emerging alternative for traditionally duty-cycled MAC protocols is wakeup radios (WuR) [169, 170]. A WuR is a secondary, ultra-low-power radio that continually listens for a wakeup call from a wakeup transmitter. When the wakeup call is received, the main radio is awakened to transmit data. WuRs allow for on-demand transmission without a rendezvous, and they excel in low-traffic scenarios. However, they require additional hardware. In Section 3.6, we analytically compare RIVER-MAC with a WuR platform to better understand which approach is more appropriate in different scenarios.

3.3 Motivation and Observations

Our work is motivated in two parts. First, we identify shortcomings of sender-initiated protocols, such as ContikiMAC, that motivate us to seek receiver-initiated solutions. Second, we identify

shortcomings of receiver-initiated protocols, such as RI-MAC, that drive us to propose a new protocol.

3.3.1 Shortcomings of Sender-Initiated Protocols

The ContikiMAC protocol [18] is a state-of-the-art sender-initiated protocol and, due to its inclusion in Contiki OS[20], is a *de facto* standard for asynchronously duty-cycled MAC protocols. However, as in other sender-initiated protocols, ContikiMAC's rendezvous mechanism requires the sender to occupy the channel with a repeated packet train. This makes ContikiMAC *impolite*, because no other nodes can transmit while the sender occupies the channel. ContikiMAC does have a phase optimization mechanism that can mitigate this issue, but this is a general and pseudo-synchronous approach that can also be applied to receiver-initiated protocols, e.g. [162, 163].

The channel occupancy of sender-initiated protocols causes performance of these protocols to suffer, particularly in relatively high-traffic scenarios. If one sender occupies the channel while waiting for its receiver, all other senders within transmission range must wait until the sender has finished before they can send. To make matters worse, if two senders are hidden from each other, they may constantly interfere with one another without realizing it. In short, sender-initiated protocols are not suitable for high-density IoT applications because they are impolite and they are not resilient to interference, especially hidden nodes.

3.3.2 Shortcomings of Receiver-Initiated Protocols

In receiver-initiated protocols, senders do not occupy the channel while waiting for a receiver, but instead *politely* listen for the receiver to initiate the data transaction. This alleviates the main shortcoming of sender-initiated protocols by reducing channel occupancy and allowing the receiver to control collision resolution, improving performance in hidden-node scenarios. In this section, we first describe RI-MAC [19], a classic receiver-initiated protocol. We then explore its shortcomings, which motivate our design for an improved receiver-initiated protocol.

3.3.2.1 Description of RI-MAC

RI-MAC [19] is a receiver-initiated, asynchronously duty-cycled MAC protocol. In RI-MAC, nodes *sleep* (turn their radios off) when not engaged in communication. Periodically, nodes *wake* (turn their radios on) and advertise their presence as receivers by broadcasting a beacon packet and listening for a response. If no response arrives, the node goes back to sleep. Wakeups occur on average every T_W (the *wakeup interval*), with some amount of randomness to distribute the beacons in time.

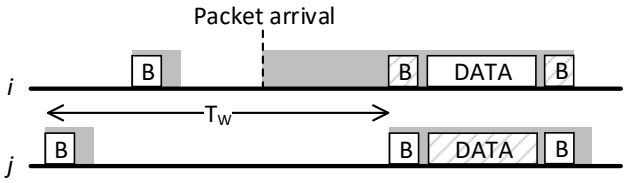


Figure 3.1: Overview of RI-MAC, showing a timeline for a sender and a receiver. Both nodes broadcast beacon packets (B) at an interval T_W. Data packets (DATA) are acknowledged with acknowledgement beacons (A). Shaded areas indicate radio on-time, and hatched boxes indicate received packets.

As shown in Fig. 3.1, when a node has a packet to send, it wakes and *idly listens* for its receiver to wake and send a beacon. When this occurs, the sender unicasts the data packet in response to the beacon. The receiver responds with an acknowledgement beacon that also advertises the receiver's availability to receive additional packets. In case of a collision at the receiver, this beacon may also include a backoff window value to prevent competing senders from colliding again.

3.3.2.2 Idle Listening

Our work is motivated by two practical problems with RI-MAC. First, in RI-MAC, *senders must idly listen for up to T_W or longer in order to receive a beacon from the intended receiver.* This idle listening (indicated in Fig. 3.1 as the shaded gray area between packet arrival and the beacon from the receiver) dominates the energy consumption of the wireless radio. For example, T_W may be on the order of hundreds or thousands of milliseconds (ms), while a data transaction may take less than 10 ms. The energy consumption from this much idle listening makes RI-MAC's

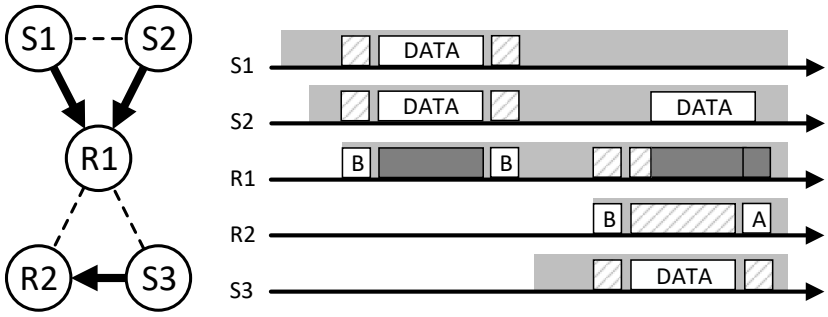
energy performance unsuitable for the demands of the IoT, particularly in moderate-to-high traffic scenarios.

3.3.2.3 Contention Between Receivers

The second problem we identify is that in RI-MAC, *excess collisions can occur when there is contention between receivers*. While RI-MAC has a backoff process to handle collisions between senders, it does not account for contention between receivers. Particularly, while senders are backing off for collision resolution, additional receivers may attempt to use the channel. This can cause more collisions that cannot be resolved by the normal process. As an example, see the topology and timeline in Fig. 3.2. In this figure, receiver R1 must resolve a collision between senders S1 and S2. While waiting for backoffs to expire, the channel is empty. This allows receiver R2 to beacon, effectively *stealing* the channel from R1. When S3 responds to R2's beacon, it can cause collisions at R1, because S3 is hidden from S1 and S2. Even if nodes are not hidden, collisions could still occur if S1 or S2 times out and transmits data at the same time as R2's beacon transmission. This scenario is often overlooked. To the best of our knowledge, RIVER-MAC is the first receiver-initiated protocol to address contention between receivers (as described in Section 3.4), making it more suitable for high-density IoT applications.

3.4 RIVER-MAC Design

In this section we introduce the design of RIVER-MAC, our receiver-initiated asynchronously duty-cycled MAC protocol motivated by the shortcomings discussed in the previous section. Our design is based on RI-MAC as described in Section 3.3.2.1. Specifically, we propose two major modifications to RI-MAC: a *CCA-based rendezvous* to reduce idle listening, and a *beacon train-based collision resolution* scheme to reduce contention between receivers.



(a) Topology. (b) Timeline.

Figure 3.2: Example of excess collisions from contending receivers. In (a), arrows indicate communication flows, while a dashed line between nodes indicates the nodes are within communication range of each other. In (b), darkly shaded boxes indicate packets that are not received due to collisions. S1 and S2 send a packet that collides at R1. While S1 and S2 back off, R2 beacons, effectively stealing the channel from R1 and resulting in another collision at R1, between S3 (responding to R2) and S2 (which has finished backing off).

3.4.1 CCA-based Rendezvous

Our first major improvement is a *CCA-based rendezvous*. As shown in Fig. 3.3, instead of idly listening for a beacon from its intended receiver, a sender strobes CCAs until it detects activity. CCA here refers to a short physical-layer check used to detect energy on the channel; any similar, short physical-layer channel check could be used. When activity is detected, the sender puts its radio in receive mode in order to receive the next packet. This approach is inspired by ContikiMAC's CCA-based receive check. We have adapted this mechanism into the CCA strobe, and then applied it to the sender side in order to reduce idle listening.

Indeed, as the sender's radio remains off between CCAs, this approach can reduce idle listening by an order of magnitude. The tradeoff is that the receiver must send two beacons: one for the sender to sense with a CCA (the *initial beacon*, marked "N"), and a second for the sender to actually receive (the *regular beacon*, marked "B"). Thus, the CCA-based rendezvous decreases the sender's load and increases the receiver's load.

In order to ensure that the sender's CCAs can detect the first beacon packet, the period of the CCA strobe (T_{STROBE}) must be no greater than the transmit time of the initial beacon packet

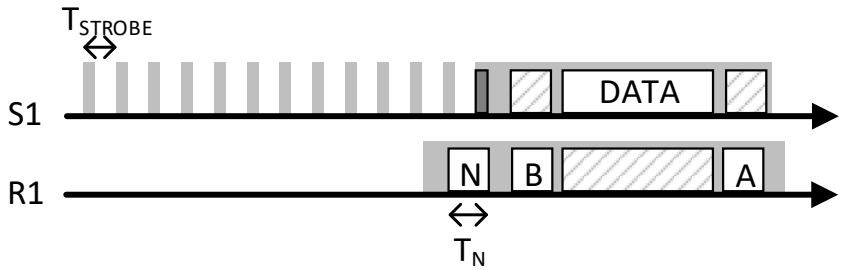


Figure 3.3: The CCA-based rendezvous of RIVER-MAC. Periodic wakeups now begin with an initial beacon (N).

(T_N). We note that T_N is a controllable parameter, to some extent, because the transmit time of the initial beacon can be increased by padding the packet with dummy data. In this way, T_N serves as a parameter that controls the tradeoff between sender energy reduction and receiver overhead during the CCA-based rendezvous. The choice of T_N is discussed in more detail in Section 3.4.3.1.

Finally, the sender cannot be sure that energy detected by its CCA is from an initial beacon, much less one from its intended receiver. Therefore, after the channel has cleared, if the sender does not hear a regular beacon from its intended receiver within the inter-packet interval (T_I) plus processing time, the sender times out and returns to its CCA strobe.

3.4.2 Beacon Train-based Collision Resolution

Our second major improvement is the use of *beacon train-based collision resolution*. This improvement addresses the problem with contention between receivers discussed in Section 3.3. Our goal is for an active receiver to reserve the channel resource so that additional receivers cannot use it. As shown in Fig. 3.4, this reservation is accomplished by having the active receiver transmit a train of regular beacons, instead of remaining silent, during the backoff portion of the sender collision resolution process. If a contending receiver wakes to beacon during this time, it will detect one of the beacon packets in the train from the active receiver and reschedule its own beacon transmission.

The use of a beacon train also affects how the backoff process works: instead of backing off for a random amount of time, a sender backs off for a random number of beacons in the train. The

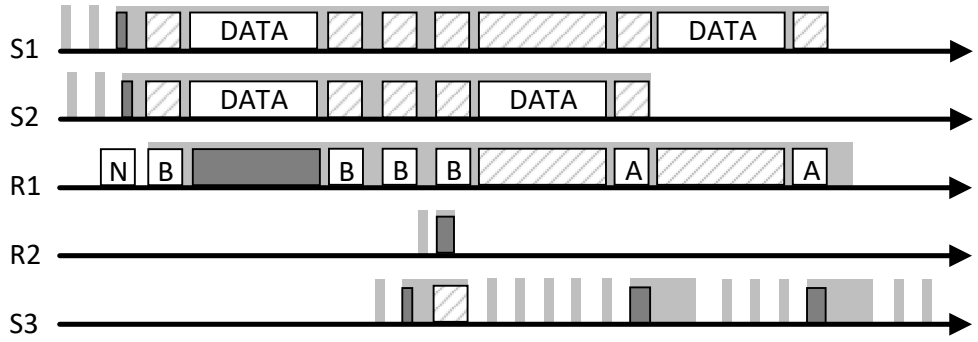


Figure 3.4: Illustration of the beacon train-based collision resolution. After R1 detects a collision (between S1 and S2), it repeatedly transmits beacons until the senders respond. This prevents R2 from accessing the channel while S1 and S2 are backing off.

beacon train is composed of k beacons (initially set to a minimum $k_{\min}$), and the receiver populates a field in each beacon with the number of beacons remaining in the train. When the sender receives the first backoff beacon, it chooses a random beacon in the range $[1 : k]$. When the sender receives that beacon, it responds with its data packet. If another collision happens, the receiver increases k (e.g. by doubling it), up to a maximum $k_{\max}$. The sender then randomly chooses another beacon to respond to. If the sender misses its chosen beacon, such as due to packet loss, it responds to the next beacon it hears.

In order to respect the reservation of the channel via the beacon train, we require a node to detect a clear channel for at least $T_I + T_B$, plus the CCA and processing time, before it can send an initial beacon. Waiting for T_I ensures that the node will detect a beacon train from a receiver within communication range. Waiting for an additional T_B (the transmit time of a regular beacon) also prevents the node from interrupting a data exchange between a hidden receiver and a sender within communication range. The clear channel detection can be efficiently accomplished using a CCA strobe, with at least one CCA occurring every T_B.

3.4.3 Practical Considerations

3.4.3.1 Choice of Initial Beacon Size

The size of the initial beacon packet (and the data rate of the radio) determines the transmit time T_N. As discussed above, this parameter constrains T_{STROBE} and effectively controls the tradeoff between the energy saved by the sender via reduced idle listening, and the additional energy overhead of the receiver due to having to transmit the initial beacon.

The choice of the initial beacon packet size is constrained by the hardware standards. For example, IEEE 802.15.4 specifies a maximum frame size of 127 bytes. At 250 kbps, this yields a maximum T_N of around 4 ms. The minimum T_N depends on the size of the packet header.

The optimal T_N (i.e. the value that minimizes energy consumption) depends on the traffic load of the node, and on T_W. A larger T_N leads to less energy spent idle listening when sending, but more energy spent on periodically transmitting the initial beacon. If the amount of idle listening is already small, either because T_W is small or because the node does not send very often, then the increased periodic overhead may outweigh the decreased idle listening.

Since the traffic load may vary over time or with different applications, we do not expect to be able to use the optimal T_N at all times. While we could design a scheme to dynamically optimize T_N, this would require coordination between neighbors about the current value of T_N, which would add overhead and complexity that we wish to avoid. Instead, we wish to choose a default initial beacon size that works well for a variety of scenarios. In Section 3.5.2, we simulate a data collection tree with moderately high traffic and different values of T_W, and we find that a larger initial beacon size (e.g. 100 bytes) generally provides the best energy performance.

3.4.3.2 Effects of Packet Loss

Here we describe the potential effects of packet loss on RIVER-MAC, based on the type of packet that is lost.

Initial Beacon: If an initial beacon is lost due to weak signal, the sender may not wake to hear the subsequent beacon. This would result in the sender waiting for the beacon from the receiver in

the next wakeup interval, as in RI-MAC. Since the sender is using a CCA strobe, the energy waste is much less than in RI-MAC.

Regular Beacon: If a regular beacon is lost, the sender (assuming it detected the initial beacon) is left hanging. However, according to the timeout previously described, the sender will shortly return to its CCA strobe and wait for the next beacon from its receiver.

Data Packet: If a data packet is lost due to weak signal, the receiver assumes no senders are active and goes back to sleep. Therefore, the sender uses another short timeout (T_I) to decide if an acknowledge beacon is incoming. If not, it returns to the CCA strobe. If a data packet is lost due to collision, the receiver will send a backoff beacon, as in RI-MAC.

Acknowledgement Beacon: If an acknowledgement beacon is lost, the effect on the sender is the same as a lost data packet, and the same rules apply. The receiver has no way to know the acknowledgement was lost, so it continues normally. This will result in the sender re-sending the same packet in the next wakeup interval, which can be resolved with simple MAC-layer duplicate detection on the receiver side.

Backoff Beacon: If the first backoff beacon is lost, the effect on the sender is the same as a lost acknowledgement beacon. If a later backoff beacon in the beacon train is lost, the sender can re-adjust itself to the backoff process simply by receiving any subsequent backoff beacon.

3.4.3.3 Tradeoffs

As with most any protocol, RIVER-MAC has a variety of tradeoffs to consider. As discussed earlier in this section, the choice of the initial beacon size allows for a tradeoff between increased receiver overhead and reduced energy for sending packets. More generally, RIVER-MAC itself is a protocol that spends additional energy on periodic overhead (the dual-beacon scheme) in exchange for better efficiency in sending packets (the CCA-based rendezvous). Therefore, while RIVER-MAC is designed to be efficient in many scenarios, it is particularly applicable for IoT applications in which nodes send packets at intervals on the order of seconds. When nodes send packets much less

often, i.e. on the order of minutes or hours, RIVER-MAC's tradeoff becomes less advantageous. This can be seen in the evaluation results presented later, such as Fig. 3.11.

An additional tradeoff inherent to the dual-beacon scheme of RIVER-MAC is that, since a wakeup is more costly, using a smaller T_W to decrease delay will be more expensive in terms of energy. Thus, RIVER-MAC may not be appropriate for applications that require delays of a few milliseconds or faster—such applications should likely use a scheduled, synchronous protocol instead. We do also note that RIVER-MAC's delay (and potentially energy) performance could be improved through the integration of opportunistic routing, as in [171].

3.5 Simulation Study

3.5.1 Implementation and Setup

We have implemented RIVER-MAC in Contiki OS [20] version 3.0. We implemented RIVER-MAC as a drop-in replacement for ContikiMAC, along with a radio driver update based on code originally written for ORPL [172] to improve software acknowledgement (softack) support. RIVER-MAC's software architecture is shown in Fig. 3.5. We have also implemented RI-MAC in a similar manner. Packet queuing and transmission scheduling is performed at the CSMA (carrier sense multiple-access) layer. Collision resolution is performed by the beaconing module. The *softack_callback()* function is called by the hardware interrupt handler when any packet (beacon or data) is received. This is used to quickly notify the CCA-based rendezvous/beaconing modules of a response to a sent data packet/beacon, reducing software delays and improving the code structure.

We evaluate our implementation through Cooja [173] simulations, using the Z1 platform, which is based on the MSP430 microcontroller and the Texas Instruments CC2420 802.15.4-compliant radio [174]. Cooja is a WSN simulator that is packaged with Contiki OS and emulates motes by running compiled code written for Contiki. We use Cooja's Unit Disk Graph Radio Medium (UDGM) as our channel model. This model creates a transmission range for each node and simulates interference from collisions. We use Contiki's Rime network stack and data packets with a payload of 28 bytes.

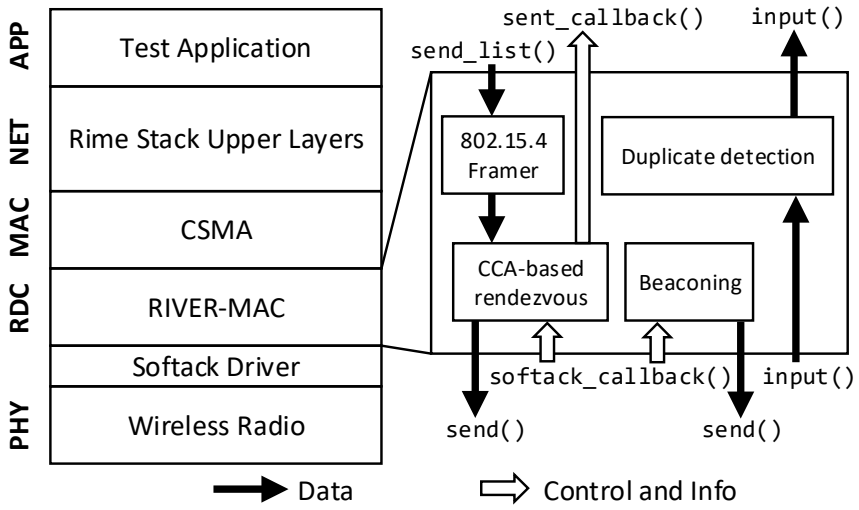


Figure 3.5: Software architecture of RIVER-MAC's Contiki implementation.

In our simulations, we compare RIVER-MAC to RI-MAC and ContikiMAC. All results, except CDFs, are averaged over at least 20 simulation trials, each composed of five simulated minutes. Our performance metrics are duty cycle (DC), packet delivery ratio (PDR), and delay.

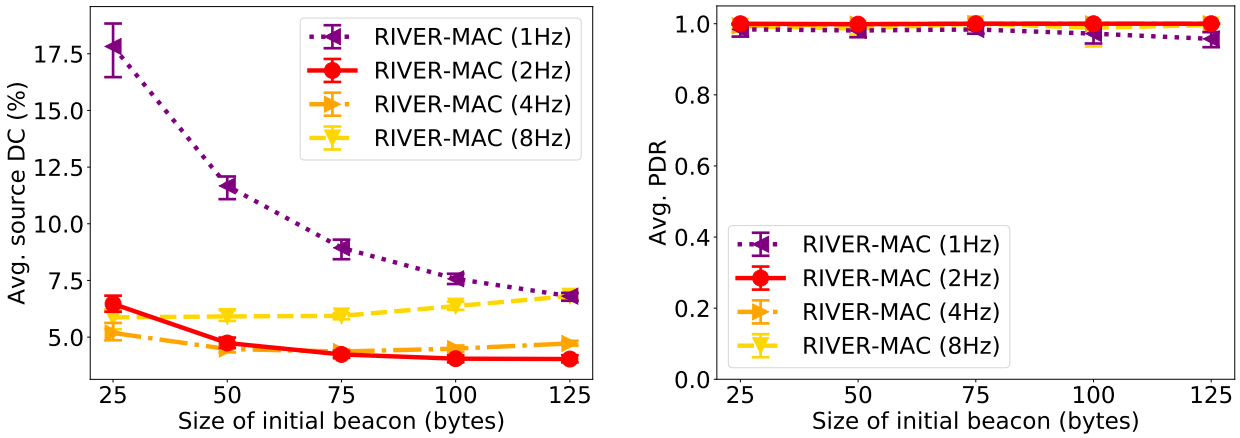
Duty cycle is the percentage of radio on-time and is a hardware-independent proxy for the energy consumption of the radio. We use the duty cycle calculated by Cooja, starting after the network has reached a steady state. Reported duty cycles are averaged first over the nodes in a simulation, then over all simulation trials. PDR is calculated as the number of packets received at the end point divided by the number of packets generated by sources. Simulations time out shortly after the final packet sent by all sources is expected to be received; in other words, all packets are given a chance to be delivered. The reported delay is application-level end-to-end delay. For all averaged metrics, we also plot the 0.05 to 0.95 quantile range as error bars. Due to their small size, many of these error bars are hidden by the plot markers.

3.5.2 RIVER-MAC Parameter Selection

We first need to choose an appropriate default initial beacon size, as described in Section 3.4.3.1. To do this, we simulate a 5x5 grid of nodes that form a data collection tree, with the sink in the

center. This setup is described in more detail in the Tree Network section below. All non-sink nodes are sources and generate data packets every 10 seconds, with a small amount of randomness.

Fig. 3.6 shows results for average source node duty cycle and PDR versus initial beacon size, with RIVER-MAC at a variety of beaconing rates, where the beaconing rate is defined as $1/T_W$. The best duty cycle performance for this traffic load is achieved at a beaconing rate of 2 Hz and an initial beacon size of 100 bytes. Initial beacon size does not affect PDR significantly, especially at beaconing rates of 2 Hz or above.



(a) Average source duty cycle.

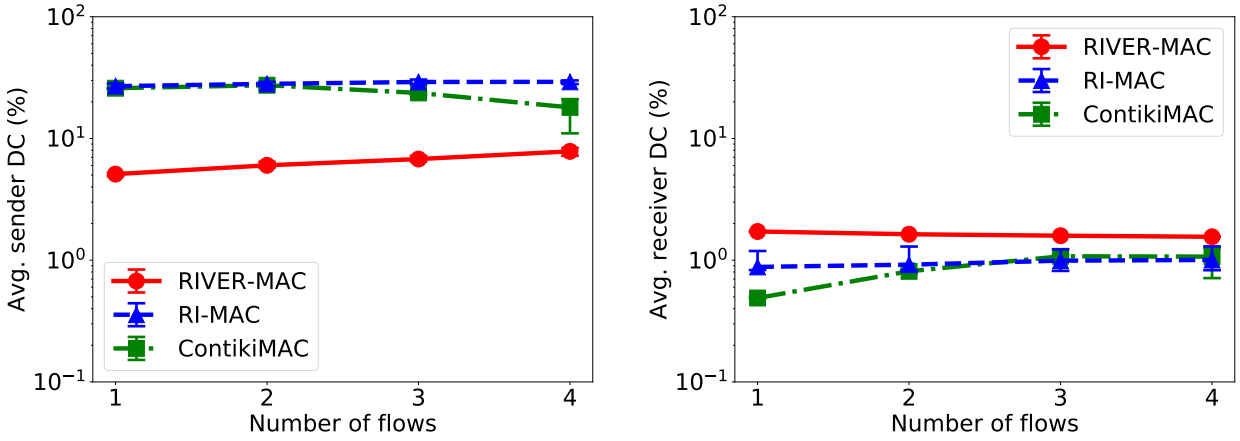
(b) Packet delivery ratio.

Figure 3.6: Results vs. initial beacon size.

Given these results, we choose 100 bytes as the default initial beacon size and 2 Hz as the beaconing rate. We also use 2 Hz as the beaconing rate for RI-MAC and the channel check rate for ContikiMAC in the remaining evaluations.

3.5.3 Clique Networks

We next evaluate in a clique network, in order to see performance under separate, contending traffic flows. In the clique network, all nodes are within communication range of each other. Each flow is composed of a different sender sending to a different receiver, resulting in a total number of nodes in the network equal to twice the number of flows. Each sender generates one packet every second, with a small amount of randomness. The results are shown in Figs. 3.7 and 3.8.



(a) Sender duty cycle.

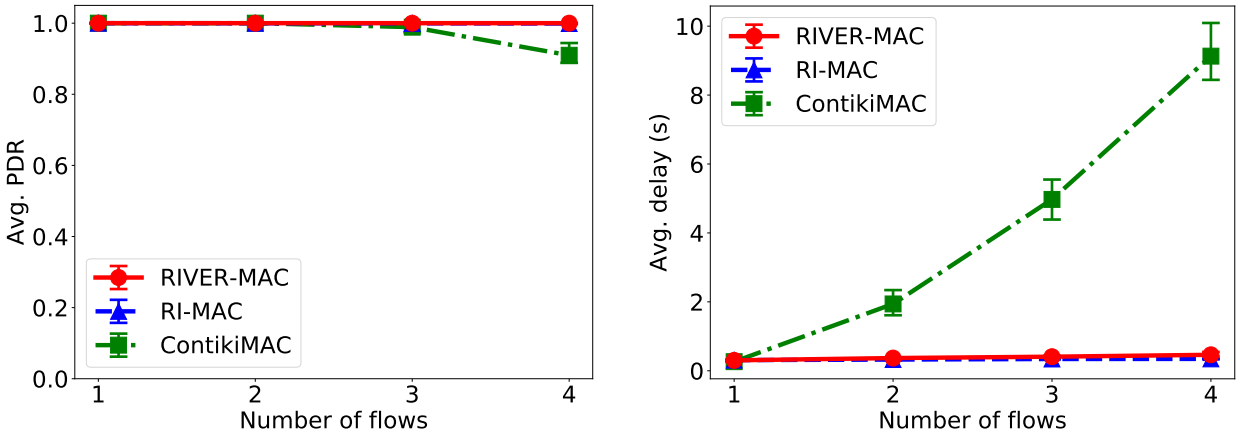
(b) Receiver duty cycle.

Figure 3.7: Duty cycle results for the clique networks, with the y-axes plotted in log scale. Each network contains twice as many nodes as flows.

In Fig. 3.7a, which shows the average duty cycle for senders, we immediately see the significant improvement in duty cycle that comes from RIVER-MAC's CCA-based rendezvous. At one flow, the average sender's duty cycle is less than 20% of RI-MAC's or ContikiMAC's. If we assume a platform with energy consumption dominated by the radio hardware, this translates to up to five times longer battery life for a sender with RIVER-MAC in this scenario.

The sender duty cycles of RI-MAC and ContikiMAC are just above 25%. This is expected, because a 2 Hz wakeup rate corresponds to 0.5 s between wakeups, and a sender waits for half of that on average. Since a packet is sent every second, this results in an average of 0.25 s of idle listening per second. Both RI-MAC and ContikiMAC have their radios on for this entire duration, resulting in a 25% duty cycle, whereas RIVER-MAC's CCA-based rendezvous breaks up this idle listening into short, efficient CCA pulses.

The tradeoff in using RIVER-MAC can be seen in Fig. 3.7b, which shows the average duty cycle for receivers. RIVER-MAC has the highest receiver duty cycle; however, in this scenario, RIVER-MAC's improvement for the sender outweighs the receiver's higher overhead. The balance of this tradeoff depends on the traffic rate, and we explore this balance more in the Tree Network section below.



(a) Packet delivery ratio.

(b) Delay.

Figure 3.8: Reliability and delay results for the clique networks.

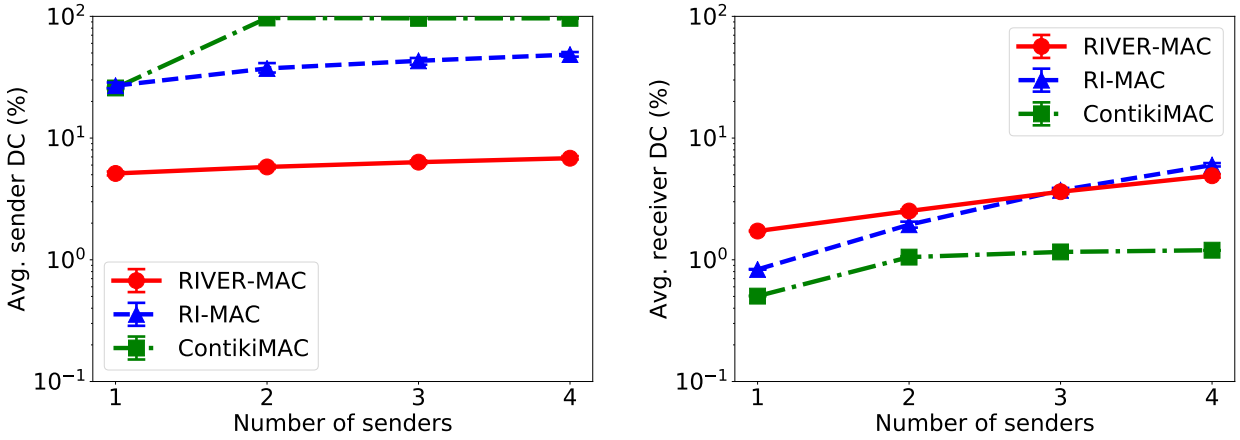
Fig. 3.7a also shows that, as the number of flows increases, ContikiMAC's average sender duty cycle decreases. However, this is not an indicator of better performance. Instead, this is due to dropped packets, as can be seen in Fig. 3.8a. In our simulations, packets are dropped after eight retries. Because of ContikiMAC's high channel occupancy, with more flows, senders are more likely to be unable to access the channel. This results in some packets being dropped without any radio activity from the sender, lowering the average duty cycle.

Finally, Fig. 3.8b shows average packet delay versus the number of flows. The polite rendezvous schemes of RIVER-MAC and RI-MAC allow them to both maintain low delay, regardless of the number of flows. But ContikiMAC experiences rising delay with more traffic, as the transmission scheduler (the CSMA layer in Contiki) increasingly backs off to try to find a time when the channel is free. For example, the average delay at four flows is over nine times greater for ContikiMAC than RIVER-MAC and RI-MAC.

3.5.4 Hidden-node Networks

We next evaluate in a small network with hidden nodes. The topology of the network is a single-hop star, with a varying number of senders and a single receiver in the center. All senders

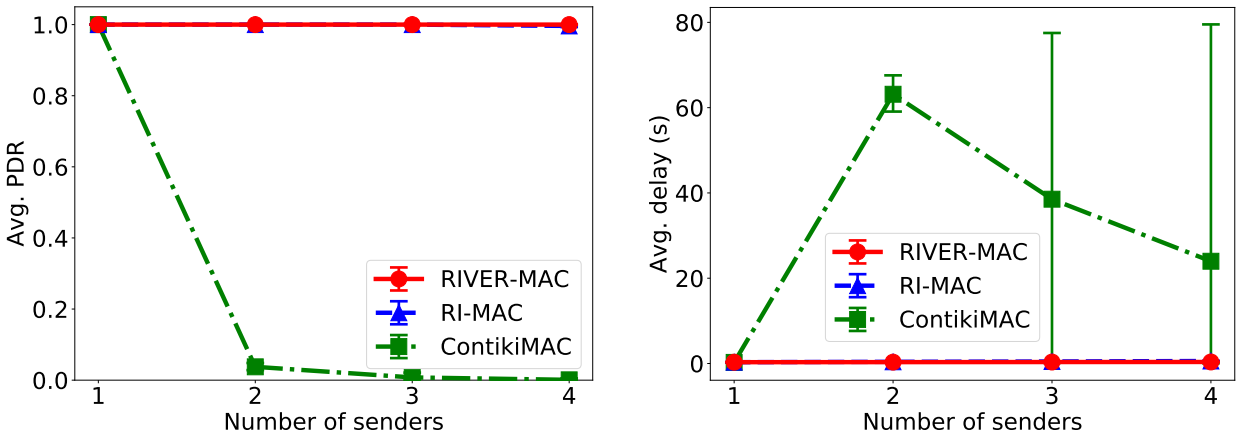
are hidden from each other, potentially resulting in collisions at the receiver. Other settings remain the same as the clique networks. The results are shown in Figs. 3.9 and 3.10.



(a) Sender duty cycle.

(b) Receiver duty cycle.

Figure 3.9: Duty cycle results for the star network, with the y-axes plotted in log scale. All senders are hidden from each other.



(a) Packet delivery ratio.

(b) Delay.

Figure 3.10: Reliability and delay results for the star network with hidden nodes.

RIVER-MAC performs well in this scenario, with a sender duty cycle (Fig. 3.9a) that is again much lower than both RI-MAC's and ContikiMAC's. Both RIVER-MAC's and RI-MAC's receiver duty cycles (Fig. 3.9b) increase with the number of senders, as the contention for the receiver results in the receiver having to resolve collisions, which consumes energy. RI-MAC's receiver duty

cycle grows faster than RIVER-MAC's, though, showing that RIVER-MAC's beacon train-based collision resolution is more efficient.

In contrast, ContikiMAC's performance in this scenario is strikingly poor. ContikiMAC's average sender duty cycle (Fig. 3.9a) grows to nearly 100% when the second sender is introduced, and the PDR (Fig. 3.10a) drops just as sharply. The explanation for this performance is simple: with two or more hidden nodes generating a packet each second, multiple nodes are nearly always sending. For ContikiMAC, this means multiple nodes are always transmitting a packet train, and these packet trains collide at the receiver. However, unlike in RIVER-MAC and RI-MAC, the receiver has no way to control or resolve these collisions. ContikiMAC could achieve better performance by "brute force," i.e., by using a higher channel check rate. Still, this scenario highlights that the lack of collision resolution is a fundamental problem for sender-initiated MAC protocols, with potentially disastrous results in the presence of hidden nodes.

ContikiMAC's poor performance in this scenario extends to its delay, as seen in Fig. 3.10b. The strange delay results for ContikiMAC are a side effect of the poor PDR. With two senders, a small number of packets are delivered, with high delay due to multiple backoffs. With three or four senders, ContikiMAC sometimes can only deliver the first packet attempted, with very low delay, before the channel becomes clogged. The end result is that some simulation runs have a very large average delay, while others have a very small average delay, creating the peculiar trend and the extremely large error bars seen for ContikiMAC in Fig. 3.10b.

Finally, we emphasize that in these scenarios, RIVER-MAC consistently achieves much better sender duty cycles than RI-MAC. RIVER-MAC does this while achieving competitive or better receiver duty cycles, and identical performance in terms of PDR and delay. In short, RIVER-MAC improves on RI-MAC significantly, with little to no drawback at this traffic load.

3.5.5 Tree Network

Finally, we simulate a 5×5 grid of nodes that form a multihop collection tree. The sink node is at the center of the grid. All other nodes are sources with varying data arrival intervals. The traffic

is moderately bursty, meaning that all sources generate packets within a few seconds of each other, regardless of the data interval. Packets are routed on the grid (i.e. not diagonally), though nodes on the diagonal are close enough to provide interference. Routes are static throughout a simulation trial and are randomly chosen at the start of the trial in a way that minimizes the number of hops to the sink. This means the nodes in the corners are four hops away from the sink. The results are shown in Figs. 3.11 to 3.13.

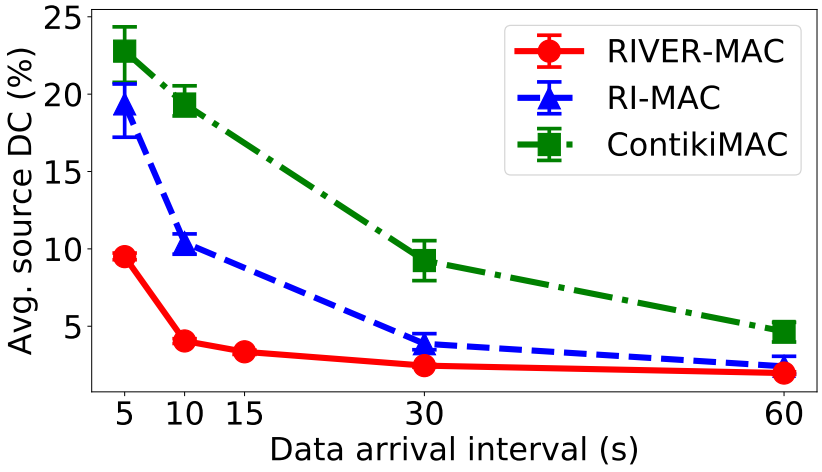
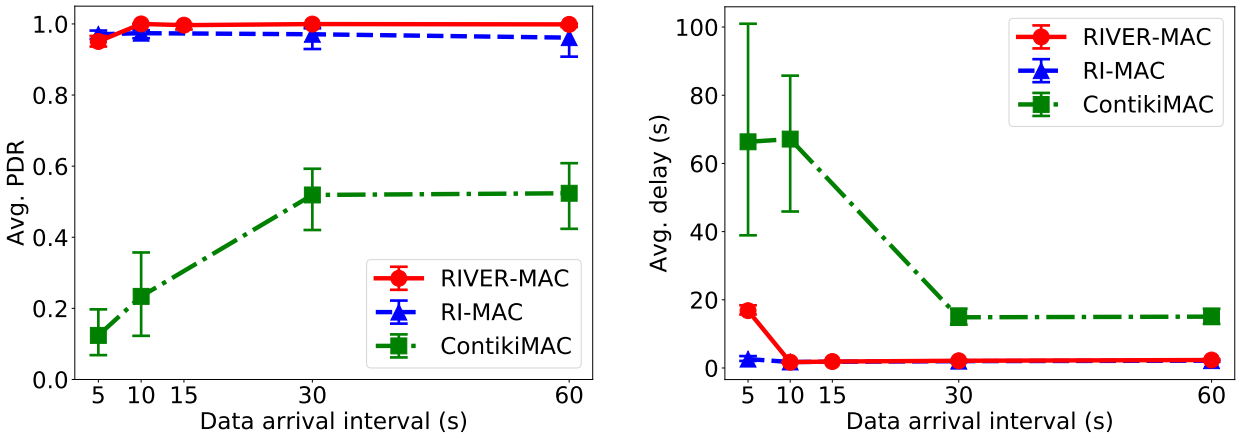


Figure 3.11: Average source duty cycle for the tree network.



(a) Packet delivery ratio.

(b) Delay.

Figure 3.12: Reliability and delay results for the tree network.

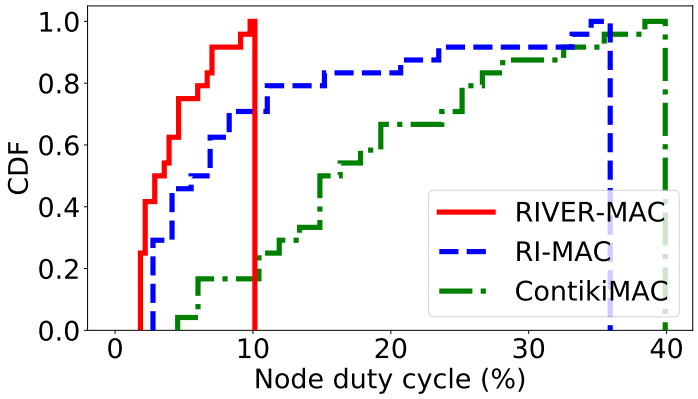


Figure 3.13: CDF of source node duty cycle for a single run of the tree network at a 10 s data interval.

RIVER-MAC performs consistently well in our tested range of data arrival intervals (5 s to 60 s). Fig. 3.11 shows the average duty cycle for all source nodes, which act as both senders and receivers in the tree topology. At a data arrival interval of 10 s, RIVER-MAC has less than one half the average duty cycle (2× the battery life) of RI-MAC and around one fifth the average duty cycle (5× the battery life) of ContikiMAC.

RI-MAC does show an advantage over RIVER-MAC in terms of PDR and delay at the 5 s data arrival interval, likely because its single-beacon wakeup occupies the channel less than RIVER-MAC's dual-beacon scheme. But again, RI-MAC pays for this advantage by consuming twice as much energy with the radio. As before, ContikiMAC shows an inability to handle hidden nodes, delivering only half as many packets as RIVER-MAC, even at low data rates (Fig. 3.12a), and showing high delay at small data intervals (Fig. 3.12b).

Finally, Fig. 3.13 shows a CDF of duty cycles for one simulation trial, revealing a much tighter distribution of duty cycles for RIVER-MAC than for RI-MAC and ContikiMAC. For example, with RIVER-MAC, all nodes in this simulation had an average duty cycle at or below 10%. Only around 70% of nodes achieved this mark with RI-MAC, and only around 20% with ContikiMAC. RIVER-MAC's tight distribution is due to its CCA-based rendezvous—the use of CCAs instead of continuous idle listening largely reduces the duty cycle impact of sending when compared to the

other protocols. If a node on the tree has to send more than other nodes (i.e. a "bottleneck" node), the impact on its energy consumption is much smaller when using RIVER-MAC.

3.6 Analytical Comparison to WuR

Wakeup radios (WuR) are an alternative to asynchronously duty-cycled MAC protocols. They can be used in similar applications. However, because of the large per-transmission overhead and the added cost and complexity of WuR hardware, they are not without drawbacks. In this section, we analyze the energy consumption of RIVER-MAC and present numerical results comparing it with WuRs at various traffic rates.

A Cooja extension for WuR, called WaCo [170], has recently been implemented. However, it has only been available for a short time, and it does not provide a straightforward method for comparing energy consumption of WuR schemes with that of traditional 802.15.4-based MACs. Therefore, we use analytical models in our comparisons instead.

3.6.1 Model of RIVER-MAC's Energy Consumption

In this section, we present a high-level model of the energy consumed by RIVER-MAC in a single *forwarding interval*. We define a forwarding interval T_F as the average time between two packets, plus the time to send a packet. We assume that one packet is sent in the forwarding interval, and that it is successfully transmitted on its first attempt. Due to the availability of current consumption data, we specifically model the average current consumption in a forwarding interval, $i_{\text{AVG}}^{\text{RIV}}$. To find the average current, we divide the charge consumption into three parts: sending the packet, receiving the packet, and overhead from periodically beaconing.

3.6.1.1 Sending Charge Consumption

The sending charge consumption Q_S^{RIV} is a function of the wakeup interval T_W and composed of the following parts:

- $Q_{\mathrm{CCA}}^{\mathrm{RIV}}$, the charge consumed in CCA checks, which is the time spent idle (assumed to be $T_W/2$, the expected value), times the fraction of the idle time spent doing one CCA (of length T_{CCA}) per initial beacon duration T_N, times the idle current i_{IL}.

$$Q_{\mathrm{CCA}}^{\mathrm{RIV}}(T_W) = \frac{T_W}{2}\frac{T_{\mathrm{CCA}}}{T_N} i_{\mathrm{IL}}. \tag{3.1}$$

- $Q_{\mathrm{BRX}}^{\mathrm{RIV}}$, the charge consumed while receiving beacons. We assume half of an initial beacon is received with reception current i_{RX}, plus one regular beacon of duration T_B. For brevity, we choose a T_B that accounts for miscellaneous factors such as the idle time between packets.

$$Q_{\mathrm{BRX}}^{\mathrm{RIV}} = \frac{T_N i_{\mathrm{RX}}}{2} + T_B i_{\mathrm{RX}}. \tag{3.2}$$

- $Q_{\mathrm{DTX}}^{\mathrm{RIV}}$, the charge consumed transmitting data of duration T_D, with transmit current i_{TX}, and receiving an acknowledgement beacon of duration T_A.

$$Q_{\mathrm{DTX}}^{\mathrm{RIV}} = T_D i_{\mathrm{TX}} + T_A i_{\mathrm{RX}}. \tag{3.3}$$

The total sending charge consumption is the sum of these three quantities:

$$Q_S^{\mathrm{RIV}}(T_W) = Q_{\mathrm{CCA}}^{\mathrm{RIV}}(T_W) + Q_{\mathrm{BRX}}^{\mathrm{RIV}} + Q_{\mathrm{DTX}}^{\mathrm{RIV}}. \tag{3.4}$$

3.6.1.2 Receiving Charge Consumption

The charge consumed while receiving, Q_R^{RIV}, is the charge consumed receiving a data packet and transmitting an acknowledgement beacon:

$$Q_R^{\mathrm{RIV}} = T_D i_{\mathrm{RX}} + T_A i_{\mathrm{TX}}. \tag{3.5}$$

3.6.1.3 Beaconing Overhead Charge Consumption

The beaconing overhead charge consumption Q_O^{RIV} is a function of T_F and T_W. The charge consumption for a single wakeup, Q_W^{RIV}, consists of the energy for the pre-beaconing CCA strobe

(one T_{CCA} per T_B) of length $T_I + T_B$ (where T_I is the inter-packet interval), the transmission of the initial beacon and the regular beacon, and the time spent listening for a response (T_L), as follows:

$$Q_W^{\mathrm{RIV}} = \frac{T_{\mathrm{CCA}}}{T_B}(T_I + T_B)i_{\mathrm{IL}} + (T_N + T_B)i_{\mathrm{TX}} + T_L i_{\mathrm{IL}}. \tag{3.6}$$

The number of wakeups in T_F is T_F/T_W, yielding a total beaconing charge consumption as follows:

$$Q_O^{\mathrm{RIV}}(T_F, T_W) = Q_W^{\mathrm{RIV}} \frac{T_F}{T_W}. \tag{3.7}$$

Combining all these calculations, the average current in a forwarding interval, $i_{\mathrm{AVG}}^{\mathrm{RIV}}$, is a function of T_W and the forwarding interval T_F. It is calculated as follows:

$$i_{\mathrm{AVG}}^{\mathrm{RIV}}(T_F, T_W) = \frac{Q_S^{\mathrm{RIV}}(T_W) + Q_R^{\mathrm{RIV}} + Q_O^{\mathrm{RIV}}(T_F, T_W)}{T_F}. \tag{3.8}$$

3.6.2 Model of WuR Energy Consumption

For a wakeup radio scheme, we model the transmitter-initiated WuR scheme from [169]. Except when transmitting or receiving, the WuR consumes its sleep current, which is very small. When the node is ready to send its packet, it must first issue a wakeup call (WuC), which can require high current. While the node is receiving a wakeup call, it also consumes more current than when idle. The wakeup call typically contains a node address and is sent at a low data rate, meaning it lasts for a relatively long duration. After the wakeup call, the node uses its main radio to send/receive the data packet and acknowledgement.

We again model average current consumption. The charge consumption from sending, Q_S^{WuR}, comes from the WuC of duration T_{WuC} sent with current i_{WTX}, and the data packet sent with the main radio, as follows:

$$Q_S^{\mathrm{WuR}} = T_{\mathrm{WuC}} i_{\mathrm{WTX}} + T_D i_{\mathrm{TX}}. \tag{3.9}$$

The charge consumption from receiving, Q_R^{WuR}, comes from the reception of the WuC with current i_{WRX} and the reception of the data packet on the main radio.

$$Q_R^{\mathrm{WuR}} = T_{\mathrm{WuC}} i_{\mathrm{WRX}} + T_D i_{\mathrm{RX}}. \tag{3.10}$$

The overhead charge consumption comes from listening on the WuR with sleep current i_{WS}. The amount of time spent listening during T_F is T_F minus the durations of the wakeup call and the data packet.

$$Q_O^{\mathrm{WuR}}(T_F) = (T_F - T_{\mathrm{WuC}} - T_D) i_{\mathrm{WS}}. \tag{3.11}$$

Finally, $i_{\mathrm{AVG}}^{\mathrm{WuR}}(T_F)$, the average current consumption in a forwarding interval T_F, is calculated as follows:

$$i_{\mathrm{AVG}}^{\mathrm{WuR}}(T_F) = \frac{Q_S^{\mathrm{WuR}} + Q_R^{\mathrm{WuR}} + Q_O^{\mathrm{WuR}}(T_F)}{T_F}. \tag{3.12}$$

3.6.3 Results

We parameterize our model for RIVER-MAC, as well as similar models for RI-MAC and ContikiMAC, with the current consumption values from the CC2420 datasheet [174] and timing values taken from Cooja. These parameters are shown in Table 3.1. Additionally, for ContikiMAC, we assume hardware acks of duration 0.3 ms. As before, we use $T_W = 500$ ms in our evaluation. We parameterize our WuR model with values reported for the platform SCM-WuR [169], summarized in Table 3.2.

Table 3.1: Parameter values for RIVER-MAC, RI-MAC, and ContikiMAC.

Parameter	Notation	Value
Regular beacon duration	T_B	1.0 ms
Initial beacon duration	T_N	3.2 ms
Ack beacon duration	T_A	1.0 ms
Data packet duration	T_D	2.5 ms
Listen time after beacon	T_L	0.5 ms
Inter-packet interval	T_I	1.5 ms
CCA check duration	T_{CCA}	0.38 ms
TX current	i_{TX}	17.4 mA
RX current	i_{RX}	18.8 mA
Idle current	i_{IL}	18.8 mA

In Fig. 3.14, we plot the average current consumption from our models versus the forwarding interval T_F. For verification, we can divide by the CC2420 receive current to roughly translate the

Table 3.2: Parameter values for WuR model.

Parameter	Notation	Value
WuC duration	T_{WuC}	12.2 ms
Data packet duration	T_D	2.5 ms
WuR sleep current	i_{WS}	3.5 μA
WuR RX current	i_{WRX}	8.0 μA
WuR TX current	i_{WTX}	152 mA
Main TX current	i_{TX}	17.4 mA
Main RX current	i_{RX}	18.8 mA

average current into equivalent duty cycle, yielding 4.9% at $T_F = 1$ s. This is in good agreement with our Cooja simulation results, i.e. Fig. 3.7a with one flow.

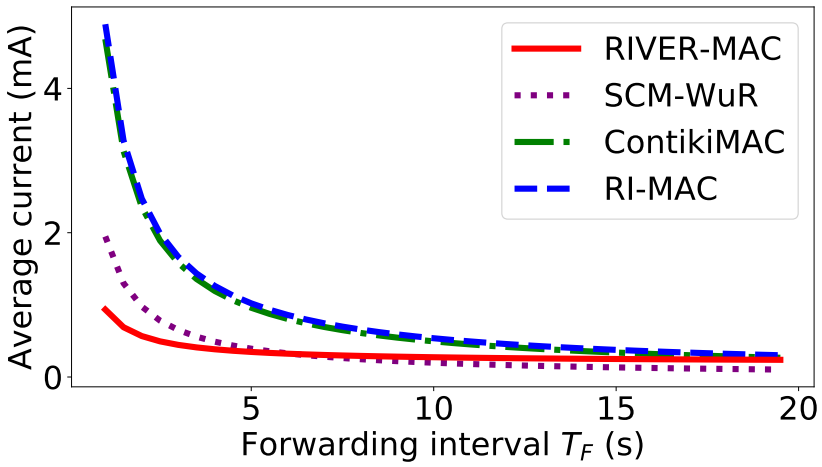


Figure 3.14: Analytical results for average current vs. forwarding interval.

We find that RIVER-MAC actually performs better than SCM-WuR at small forwarding intervals (< 6 s) because of SCM-WuR's costly wakeup call. Based on these results, we suggest that WuR is a more efficient solution at lower traffic rates, while RIVER-MAC is a better choice at moderate to high traffic rates. We also believe that RIVER-MAC's performance would scale better in more complex scenarios, such as the tree topology from Section 3.5, where interference and hidden nodes are factors.

Finally, we reiterate that any energy and cost savings from WuRs must be weighed against the added cost and complexity of the WuR hardware. In future work, we plan to further explore this tradeoff with simulation studies, e.g. utilizing the WaCo [170] tool.

3.7 Extensions and Future Directions

We here describe extensions and possible future directions for RIVER-MAC.

3.7.1 Burst Transmissions

If the application layer queues multiple packets at once, or if the wakeup interval of a node's receiver is longer than the node's data arrival interval, a node may have multiple packets for a receiver in its queue. A burst transmission allows the node to send some or all of these queued packets during a single rendezvous with a receiver.

Packet bursts seem an obvious feature for a duty-cycled MAC protocol; the sender ought to be able to queue up additional packets while waiting for the receiver to beacon. However, bursting is non-trivial to implement on a single-threaded microcontroller. For example, a duty-cycled MAC protocol that requires precise control of the radio may block at the sending process, meaning that the application layer is unable to queue additional packets. This is the case for protocols such as ContikiMAC and our baseline implementation of RIVER-MAC. Our first step toward burst transmissions in RIVER-MAC was thus to modify the CCA-based rendezvous code to periodically return control to the application layer (every 100 ms in our implementation).

Once the rendezvous is established, to activate a burst, the sender sets a bit in the IEEE 802.15.4 header that indicates to the receiver that additional packets are pending. The burst then consists of a straightforward repetition of the the data-ack sequence until the final packet is sent (without the pending bit set). During a backoff, the receiver pauses the downwards count of the beacon train in order to handle bursts, preventing contending senders from interrupting the burst.

3.7.2 Dynamic Duty Cycling

As is evident from the results in Section 3.5 and Section 3.6, the optimal wakeup interval T_W for a protocol such as RIVER-MAC depends on the traffic rate. This is because a larger T_W leads to an increase in periodic wakeup overhead, but also leads to an increase in the expected idle time of a sender waiting for a receiver to beacon (or acknowledge, in the case of ContikiMAC). Since RIVER-MAC's CCA-based rendezvous makes idle listening for the sender much less expensive, we expect RIVER-MAC to be less sensitive to T_W than RI-MAC. However, RIVER-MAC's increased beaconing overhead means T_W does still significantly affect RIVER-MAC's energy performance.

We also note an additional interaction between T_W and energy performance that is unique to receiver-initiated asynchronously-duty-cycled MACs. An increase in T_W for a receiver will result in more packets queued at senders waiting for the receiver. This will results in more collisions at the receiver. Because the MAC must use energy to resolve these collisions, an increase in T_W may also increase energy consumption due to collisions. We identified this issue by observing, in Cooja simulations, that a significant portion of RIVER-MAC's radio on-time is spent resolving collisions, particularly at high traffic rates.

The practice of adjusting T_W (or similar parameters) at run-time is called *dynamic duty cycling* in the literature. We believe that RIVER-MAC's performance could be further improved through the use of a dynamic duty cycling scheme. We have identified the following measurable quantities that could be useful in designing such a scheme:

Forwarding interval: we define the *forwarding interval* as the average time between when packets are forwarded, or the inverse of the traffic rate. We measure forwarding interval for a node as the average time between calls to the function that queues a packet at the MAC layer.

Number of senders: when a node beacons, it may receive no response (zero senders), one response (no collisions), or multiple responses (collisions). Zero senders suggests the receiver did not need to wake. Collisions suggests the receiver may need to wake more often.

Sender queue depth: this is an indicator of the relationship between T_W for a receiving node and the traffic rate for the sender. If the sender's queue depth is consistently high, the receiver may need to shorten T_W in order to prevent a queue full condition for the sender.

A framework to collect these quantities has been integrated into our implementation of RIVER-MAC. We leave the design of the dynamic duty cycling scheme that leverages these quantities to future work.

3.8 Conclusion

We have presented RIVER-MAC, a receiver-initiated, asynchronously duty-cycled MAC protocol for the IoT. RIVER-MAC uses an efficient CCA-based rendezvous, and beacon train-based collision resolution, to achieve good energy performance in a variety of scenarios, such as moderate to high traffic with hidden nodes. We have implemented RIVER-MAC in Contiki OS and evaluated it with Cooja and analytical models. Future work includes testing RIVER-MAC in large-scale testbeds, and exploration of opportunistic routing and dynamic duty-cycling in conjunction with RIVER-MAC.

CHAPTER 4. CROSS-LAYER DATA COLLECTION

4.1 Introduction

Chapter 6 presented a problem and solution unique to deployment of wireless sensor network (WSN) nodes on wind turbine blades. However, for other wind turbine components, general-purpose WSN data collection schemes are both sufficient and preferable—more protocols leads to more development and deployment effort and more points of failure. Therefore, we desire a single data collection scheme that can work for all applications other than blade monitoring, including drive train monitoring, tower monitoring, foundation monitoring, and even monitoring of the land around and between turbines. A traditional scheme such as CTP or RPL (discussed in Section 1.2) could hypothetically serve this function.

But while the nodes in this scenario would be much more accessible than those deployed on blades, the WSN still suffers from an accessibility problem: regularly changing batteries for dozens of nodes on hundreds of turbines is not reasonable. Energy harvesting modules can help, but add to the cost of the nodes. In the end, then, energy consumption is still a critical consideration, and reduction of energy consumption through more efficient protocols is still an attractive proposition.

One promising method of reducing energy consumption in asynchronously duty-cycled WSNs is a cross-layer technique called link-layer *anycast*, in which a packet only needs to be sent to one of a set of *potential forwarders*, neighbors that provide acceptable progress. Anycast has been shown to reduce energy consumption and delay and to increase reliability in duty-cycled networks by allowing nodes to dynamically take advantage of available links [175, 176, 177]. However, existing anycast data collection schemes do not specifically minimize energy consumption, which we consider to be the most important metric for our application.

To address this need, we present EDAD: Energy-Centric Data Collection with Anycast in Duty-Cycled WSNs. EDAD uses a new anycast routing metric called Expected Energy Consumed Along the Path (EEP) that, contrary to previous anycast routing metrics,

1. is designed to minimize energy, and

2. automatically adapts itself to network settings.

The next section investigates related work. EDAD's design, including EEP, is presented in Section 4.3. Section 4.4 presents simulation results that show EDAD reduces energy consumption compared to state-of-the-art alternatives, and Section 4.5 concludes this paper.

4.2 Related Work

Section 1.2 presented an overview of CTP, a classic WSN data collection protocol. In CTP, each node chooses its neighbor with the lowest routing metric value as its parent, forming a tree topology, as shown in Fig. 4.1a. When anycast is applied to data collection, this tree becomes a destination-oriented directed acyclic graph (DODAG) [178], as shown in Fig. 4.1b.

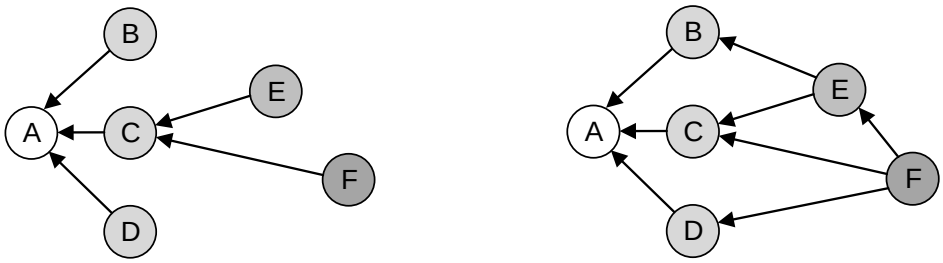


Figure 4.1: The effect of anycast on a data collection topology. (a) shows a traditional gradient-based tree topology, where node A is the sink. Each node chooses one parent as its next hop forwarder. (b) shows anycast applied to the same network, creating a DODAG topology in which each node can have multiple potential forwarders.

Anycast for WSNs has been analytically studied in papers such as [179] and [175], and in protocols such as GeRaF [180], CMAC [181], and A^2-MAC [182]. These studies do not define a practical anycast routing metric, with most of them relying on geographical positions of nodes

acquired by an external process. In contrast, EDAD presents EEP, a new anycast metric that is calculated using only information from neighboring nodes. [183] does propose an anycast routing metric, but for a synchronous MAC protocol, while EDAD's design utilizes an asynchronous MAC protocol and the associated benefits.

ORiNoCo [177], a receiver-initiated anycast data collection protocol, avoids the anycast metric problem by forwarding a packet to any neighbor that advertises a path within a difference Θ of the best known path. ORiNoCo relies on the arbitrary choice of Θ to adjust the protocol to a particular network, whereas EDAD's EEP automatically adapts itself to given network settings, such as the average amount of time between node wakeups. Also, EDAD maintains a neighbor table and an explicit forwarder set to allow EEP fine-grained control over potential forwarders, allowing it to make routing decisions based on energy considerations.

ORW [178, 184] and the follow-up ORPL protocol [172] use an anycast metric called Expected Duty Cycled Wakeups (EDC), an approximation of the expected number of wakeups required to deliver a packet to the sink [184]. The EDC calculation factors in both link qualities and the number of potential forwarders available to a node. However, the physical meaning of EDC is not intuitive, whereas EDAD's EEP metric uses a clearly defined unit of energy, allowing EEP to minimize energy consumption. Additionally, similar to ORiNoCo, EDC relies on an arbitrary tunable parameter w intended to reflect the general cost of forwarding in a network, though the authors do provide $w = 0.1$ as a reasonable default for most scenarios.

4.3 EDAD Design

An overview of EDAD's components is shown in Fig. 4.2. At each node i, a neighbor table is kept updated with information from all of i's neighbors, as detailed in Section 4.3.2.4. A set of forwarders F_i is selected from the table, using the process described in Section 4.3.1.4, such that the minimum EEP_i is achieved. Data packets are then dynamically sent to the first available forwarder in F_i.

This work focuses on a receiver-initiated implementation, but the concepts of EDAD and EEP can be applied to any link-layer protocol. The following sections explain the design of EEP, EEP forwarder set selection, anycast routing in EDAD, and some practical considerations, such as data packet retries, topology maintenance, and network initialization.

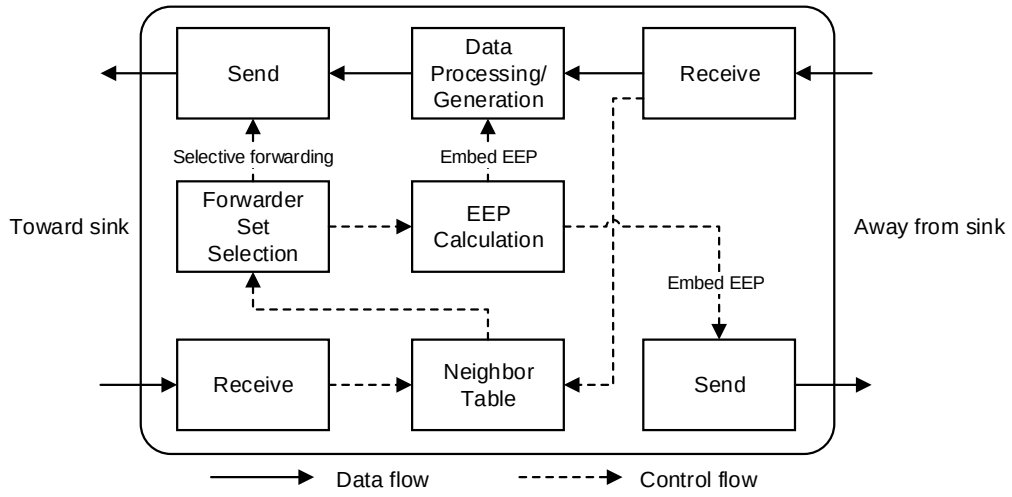


Figure 4.2: Overview of EDAD's components at a node i. A neighbor table tracks the EEPs of i's neighbors, as well as estimates of the quality of their links with i. F_i, i's forwarder set, is selected from the neighbor table such that the minimum EEP_i is achieved. EEP_i is embedded in every outgoing packet. Data packets are forwarded to the first node in F_i that is available.

4.3.1 EEP: Expected Energy Consumed Along the Path

EEP is a new routing metric that estimates the expected energy consumed by nodes along the path of a packet as they relay it to the sink. EEP is designed to minimize energy for anycast on duty-cycled, asynchronous link layer protocols. The following sections describe the link layer model and simplifying assumptions on which EEP is based, as well as the observations that lead to EEP's energy-centric design. Then, EEP's calculation and forwarder set selection process will be presented.

4.3.1.1 System model

EEP works with any duty-cycled, asynchronous link layer protocol. An example of a receiver-initiated version of such a protocol is shown in Fig. 4.3. Between beacons, node i's potential forwarders sleep for a random amount of time in the interval $[0.5T_W, 1.5T_W]$, where T_W is a network setting known as the *wakeup interval*, and the expected sleep time for a node is T_W. A node i with a packet to send wakes and actively listens for an amount of time X_{min} before any potential forwarder $j \in F_i$ wakes and broadcasts a beacon to advertise its availability as a receiver. Assuming the wakeups of i's potential forwarders are uniformly distributed on T_W, the expected value of X_{min} is well-known, as follows:

$$E[X_{min}] = \frac{T_W}{|F_i| + 1}, \tag{4.1}$$

where $|F_i|$ is the number of nodes in i's forwarder set.

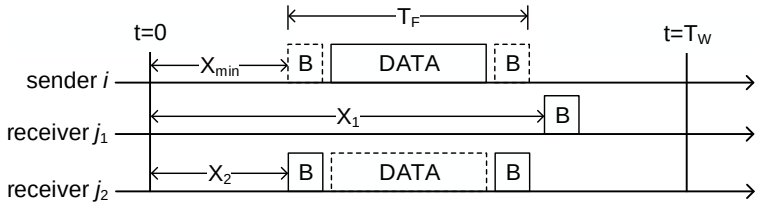


Figure 4.3: Basic example of anycast with two potential forwarders and a receiver-initiated link layer protocol. A sender i waits X_{min} before one of its forwarders, j_1 or j_2, wakes and sends a beacon. In this case, j_2 wakes first, so $X_{min} = X_2$, and i sends the data packet to j_2. Upon receipt, j_2 replies with an acknowledgement.

This model for X_{min} includes several simplifying assumptions. Because the sleep time for each node is random on an interval around T_W, X_{min} is actually expected to be larger than Eq. (4.1) suggests, due to a phenomenon resembling the *hitchhiker's paradox* [185]. However, as discussed in [177], Eq. (4.1) is a reasonable approximation and is used for its simplicity. Another assumption of Eq. (4.1) is that the beacon from the first $j \in F_i$ is successfully received by i. In reality, the beacon could be missed or garbled, meaning i would wait for the next j, which again suggests an expected wait longer than Eq. (4.1).

However, a counterbalancing assumption is that the wakeups of $j \in F_i$ are freshly uniformly distributed for each sender i in the path. Because the uniform distribution is not memoryless, this assumption pushes Eq. (4.1) toward overestimating expected wakeup delays. For example, suppose a node i can reach k either directly or through an intermediate node j. If j wakes first and receive's i's packet, then j likely does not need to wait as long as Eq. (4.1) suggests for k to wake, because part of the time until k wakes has already passed while the packet was queued at i and in transit to j.

Overall, these assumptions balance out well enough to use Eq. (4.1) as a simple model of the amount of time that a node with a packet to send has to wait until a forwarder wakes to receive it. This model provides part of the foundation for EEP calculation, described in Section 4.3.1.3.

4.3.1.2 Observations

EEP's design is based on the above model and the following observations of anycast routing in duty-cycled networks:

- Different forwarders provide different amounts of routing progress.
- Different forwarders require different numbers of expected transmission attempts before a successful transmission.
- A larger forwarder set for a node means less expected time for the node to wait for a forwarder to wake.
- A different forwarder set for a node may yield a different average routing progress per forwarder.

These observations reveal two tradeoffs in forwarder selection. The first is between end-to-end delay and the number of transmission attempts required to transmit the packet to the sink. Choosing only forwarders that provide low routing progress on reliable links will decrease the number of transmissions but increase the end-to-end delay. Choosing only forwarders with high

routing progress on unreliable links will decrease end-to-end delay but increase the number of transmissions.

The second tradeoff is in regard to the size of the forwarder set. Adding more forwarders can decrease one-hop delay, but if the forwarders do not provide enough routing progress, the overall routing cost can be increased. EEP is designed to balance these tradeoffs.

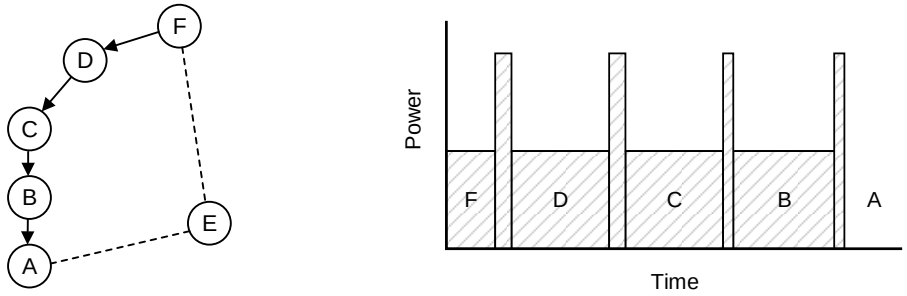
In order to do so, both delay and transmissions are translated into units of energy. A unit of energy is defined as the amount of energy consumed by a node that is active (sending, receiving, or listening) for the length of time T_F required to transmit a data frame, as shown in Fig. 4.3. Sending power and listening or receiving power are assumed to be equal, a reasonable simplification for WSN radio hardware [174]. Along the path of a packet, one energy unit is consumed for each T_F of delay because the node with the packet currently queued is always active. Two energy units are consumed per transmission because both the sender and receiver are active, and each transmission lasts T_F.

As shown in Fig. 4.4, this translation to energy allows the delay-transmissions tradeoff to be directly modeled. Fig. 4.4a shows an example of more delay over more hops on reliable links that require fewer transmissions, while Fig. 4.4b shows an example of less delay over fewer hops on unreliable links that require more transmissions. The shaded area is the energy consumed in each case. As discussed in Section 4.3.1.4, a forwarder set can be chosen that minimizes the expected shaded area, thus balancing the second tradeoff and minimizing energy consumed along the path of the packet.

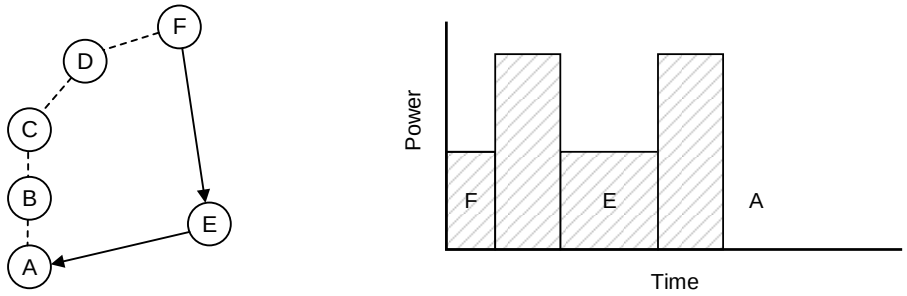
As a final observation, because a packet is likely to spend much more time sitting in a queue than in transmission, the energy consumed is strongly affected by the delay. Therefore, this energy minimization model also tends to minimize delay.

4.3.1.3 EEP calculation

Based on the above observations, EEP is calculated at each node i using the EEP of nodes $j \in F_i \subseteq N_i$, where F_i is i's forwarder set and N_i is i's set of communication neighbors. The EEP calculation also uses an estimate of the quality of the wireless link between i and j, measured as



(a) Case when a path with many reliable hops is chosen. The end-to-end delay is larger, but less transmission attempts are required.



(b) Case when a path with fewer hops, utilizing unreliable links, is chosen. The end-to-end delay is less, but many transmission attempts are required.

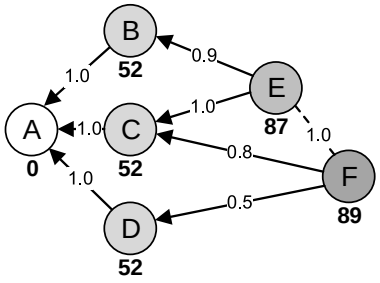
Figure 4.4: Illustration of the delay-transmissions tradeoff. The graphs show system-wide power usage over time as a packet is sent from F to A. The regions of lower power levels represent the times when a node along the path is listening, waiting for a forwarder to wake. The regions of higher power levels represent the time spent transmitting, when both sender and receiver are awake. The shaded area represents the total energy expended to route the packet from the source to the sink.

packet reception rate (PRR) and denoted as p_{ij}. The process for selecting F_i is detailed in the next section; for now, assume F_i is known. Then,

$$EEP_i = \frac{\sum_{j \in F_i} [EEP_j + 2/p_{ij}]}{|F_i|} + \frac{T_W/T_F}{|F_i| + 1}. \tag{4.2}$$

The unit of EEP is the energy unit defined in the previous section. An example topology using EEP is shown in Fig. 4.5, where nodes B through F send to the sink node A. A solid arrow from i to j indicates that j is in F_i. The resulting EEP of each node is written below it.

The EEP equation can be broken down into two distinct terms. The first term is the average expected energy expended along the multihop routes presented by the various $j \in F_i$. The energy for the route provided by a particular j is EEP_j, plus the expected number of transmission



F's neighbor table

j	EEP_j	p_{Fj}
C	52	0.8
D	52	0.5
E	87	1.0

Figure 4.5: Example of a topology using EEP when $T_W/T_F = 100$. Links are labeled with the p_{ij} of the link, and the EEP of each node is written below it. A solid arrow from i to j indicates that $j \in F_i$, while a dashed line indicates $j \notin F_i$. As shown in the table, even though F has a link to E, and E has a lower EEP than F, F does not include E in its forwarder set because adding E would increase EEP_F. In other words, with the given network settings, E does not provide enough routing progress for F to use it as a forwarder.

attempts required to reach j, multiplied by two because two energy units are consumed during each transmission. All $j \in F_i$ are assumed to have an equal probability of being the first to wake, so the average is obtained by dividing by $|F_i|$.

The second term of Eq. (4.2) is the expected energy consumed during the single-hop delay at i, meaning the time i spends actively waiting for some $j \in F_i$ to wake. Since one energy unit is consumed per T_F of activity, this quantity is Eq. (4.1) normalized to T_F.

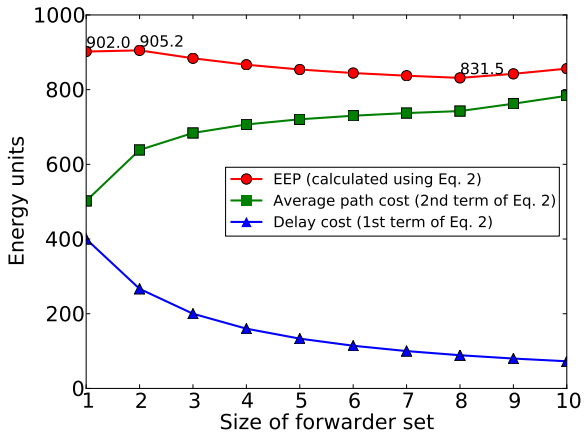
4.3.1.4 EEP forwarder set selection

The forwarder set selection component shown in Fig. 4.2 determines which $k \in N_i$ are included in F_i to produce the minimum EEP_i. This process works as follows. All neighbors $k \in N_i$ are sorted in ascending order by a sorting key $c_k = EEP_k + 2/p_{ik}$. One at a time, the neighbors are added to F_i and EEP_i is calculated. When all neighbors have been added in order, and EEP_i has been calculated with each additional neighbor, the set of neighbors that yields the minimum EEP_i is used as F_i. As an example, note how, in Fig. 4.5, node F does not include E in its forwarder set. Adding E would increase EEP_F, because the route provided by E would increase the expected average energy per route from $j \in F_i$, outweighing the decease in energy from the

lower one-hop delay expected with one additional forwarder. This is how the forwarder set selection process balances the forwarder set size tradeoff discussed in Section 4.3.1.2.

This process guarantees the minimum EEP for a given N_i, without checking all possible permutations of F_i, because of the sorting of $k \in N_i$ by c_k. In more detail, the first term of Eq. (4.2) is the average c_j for $j \in F_i$. The second term of Eq. (4.2) decreases monotonically with the addition of more forwarders to F_i, regardless of the choice of forwarders added. Therefore, given an F_i and the set of remaining neighbors $k \in (N_i - F_i)$, the smallest EEP_i obtained from adding one more k to F_i will always be obtained by adding the k with the smallest c_k. Thus, only incremental F_i combinations, created by adding each k in order of c_k, need to be checked.

Some previous anycast works, such as [179], define a stopping condition for the forwarder set selection algorithm where the process can be stopped when the first ordered node that increases the routing metric value is found. In EEP, all incremental combinations, as described above, must be checked. As shown in Fig. 4.6, the EEP function is not necessarily convex, so a local minimum may occur, particularly when a group of forwarders have similar EEPs. However, even without an early stopping condition, forwarder selection for node i is efficient, with a computational complexity of $O(|N_i|)$.



j	c_j	EEP_i
1	502	902.0
2	775	905.2
3	775	884.0
4	775	866.8
5	776	853.9
6	778	844.5
7	780	837.3
8	780	831.5
9	920	842.3
10	974	856.2

Figure 4.6: Example of an N_i with a non-convex EEP function when $T_W/T_F = 800$. The graph shows EEP_i as nodes are added to F_i, and also the separate cost terms of the EEP calculation. The addition of the second forwarder to F_i increases EEP_i, causing a local minimum at $|F_i| = 1$, while $|F_i| = 8$ yields the global minimum and therefore the optimal forwarder set. The table shows each node's c_j and the value of EEP_i if all nodes up to j are included in F_i.

4.3.2 EDAD Operation

4.3.2.1 Framework

EDAD's overall framework is similar to that of other anycast data collection schemes. All nodes in the network may be data sources. One sink is assumed, though the concepts can be applied to a multiple-sink scenario. EEP is used as the routing metric. The sink has an EEP value of zero. All other nodes calculate their own EEP using a neighbor table, as shown in Fig. 4.2.

4.3.2.2 Anycast forwarding

In EDAD, all nodes maintain an explicit forwarder set, described in Section 4.3.1.4. A sender uses this set to choose the next hop receiver, which is the first $j \in F_i$ from which i hears a beacon. This dynamic choice of forwarder is what makes EDAD anycast and creates the DODAG topology shown in Fig. 4.1. Any beacon from a $j \notin F_i$ is ignored, even if the beacon advertises a low EEP. This is because the potential cost of sending to that node could outweigh the gain, either because the node does not provide enough progress or because the number of transmission attempts to transmit a packet to that node could be excessive. This is a tradeoff that EDAD allows EEP to balance via the forwarder set selection process.

4.3.2.3 Retries

EDAD maintains a separate retry counter for each $j \in F_i$. Sender i increments the corresponding counter whenever a data packet transmission to j fails, meaning that an acknowledgement from j is not received by i. If the counter reaches a maximum, then i stops retrying to j and waits for the next forwarder. Thus, EDAD retries a single data transmission to a particular node to try to utilize the available connection, but if the transmission is continually unsuccessful, EDAD takes advantage of its anycast nature and sends to a different forwarder instead of dropping the packet.

4.3.2.4 Maintenance

EEP calculation and forwarder set selection require that a node maintain an updated neighbor table, as shown in Fig. 4.2, to track its neighbors' EEPs and link quality estimates. To this end, all beacons and data packets contain the EEP of the transmitting node, allowing EDAD to handle new nodes and changes in EEP. When a new neighbor is discovered, or the EEP of a current neighbor changes, the neighbor table is updated and the forwarder set is rebuilt. This potentially results in a new EEP value, which is embedded in all outgoing beacons and data packets.

Because of its dynamic nature, anycast inherently mitigates node failure issues. In the extreme case, if i hears no beacons from any $j \in F_i$, all $j \in F_i$ are assumed to be at least temporarily unavailable, so the EEP for each j is set to infinity in i's neighbor table. This will in turn increase i's EEP, so upstream nodes will avoid forwarding to i until i reestablishes communication with its forwarders.

Failed transmission attempts have a similar, but slower, effect on i's forwarder set. Link qualities with forwarders can be estimated as a moving average of successes per number of attempts. A failed transmission attempt to j thus lowers p_{ij}. If enough transmissions to j fail, p_{ij} will become small enough that j will be excluded from i's forwarder set.

4.4 Evaluation

EDAD was implemented in the ns-2 network simulator with an underlying MAC protocol based on RI-MAC and extended to anycast. For comparison, single-parent ETX data collection (as in CTP), ORW, and ORiNoCo were also implemented. To directly compare EEP, EDAD's main contribution, to other metrics, these protocols were implemented on EDAD's framework; they all used the same MAC layer and retry scheme, but their own respective routing metrics. ORW was tested with $w = 0.1$ and $w = 1.0$, the extremes of the w range tested in the ORW papers, where $w = 0.1$ was found to be the best default configuration [184]. ORiNoCo was tested with $\Theta = 0.1$ and $\Theta = 0.9$, the extremes of the allowable range of Θ, according to [177]. ORW with $w = 0.1$ is denoted as ORW-0.1, and the other protocols are denoted similarly.

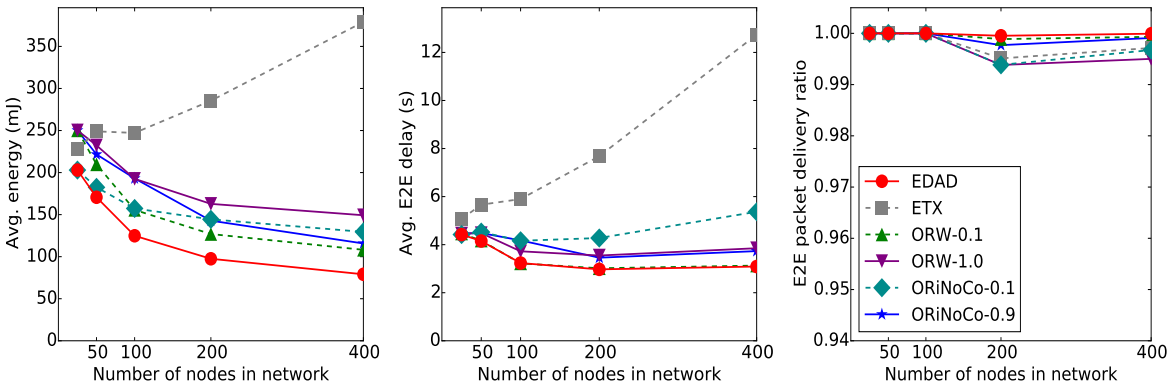
Nodes were distributed in a 250 x 250 m space in a grid plus variance pattern, as seen in Fig. 4.10. A fully random distribution was also tested, with similar results, so the grid plus variance pattern was used for clarity of examples. The sink was located in one corner and all other nodes generated packets according to a Poisson process with rate λ. Each link quality p_{ij} was calculated based on the distance between i and j using the shadowing propagation model, a fixed noise level of -97 dBm, and a well-known function for estimating packet reception rate based on signal-to-noise ratio [186]. The shadowing propagation model used a reference path loss of 61.4 dB at 2 m and a path loss exponent of 1.97, values that were experimentally determined for a real WSN deployment in [187]. No random variation was used in the propagation model; instead, 30 different node layouts were tested for one simulated hour each and averaged together for each data point.

4.4.1 General Performance

To test the general performance of EDAD relative to the other protocols, simulations were run at a variety of network densities, wakeup intervals, and data generation rates. The effect of each parameter is shown by fixing two of the three parameters at default values and varying the other. The default number of nodes in the network is $n = 100$, the default wakeup interval is $T_W = 2$ s, and the default data generation interval is $1/\lambda = 30$ s.

4.4.1.1 Network density

Performance under varying n, the number of nodes in the network, is shown in Fig. 4.7. Energy per packet for unicast ETX increases with density because, after a point, ETX's choice of forwarder remains similar, but the number of collisions increases as packets converge along similar routes to the sink, causing increased delay. However, for the anycast protocols, energy per packet decreases as n increases, because more nodes implies more forwarders, leading to lower delay and, due to the greater path diversity, fewer collisions. As expected given EEP's design goal, EDAD uses the least amount of energy, with a 25% decrease in energy per packet over ORW-0.1 when $n = 400$.



(a) Energy vs. n. (b) Delay vs. n. (c) Reliability vs. n.

Figure 4.7: Performance with different network densities.

ORiNoCo performs unpredictably over the range of densities because of the way it uses the arbitrary Θ to determine its forwarder set. For example, at low densities, ORiNoCo-0.1 performs better than ORiNoCo-0.9, while the opposite is true at higher densities. ORW performs more consistently, with $w = 0.1$ generally providing better performance than $w = 1.0$, an observation that holds through most of the results and agrees with [184]. All protocols except EDAD show a small decrease in E2E reliability above 100 nodes, with protocols with smaller forwarder sets suffering more. This is likely because a smaller forwarder set means less path diversity, resulting in more collisions as routes from multiple nodes tend to converge on the way to the sink. At higher densities, the reliability of EDAD is expected to decrease as well.

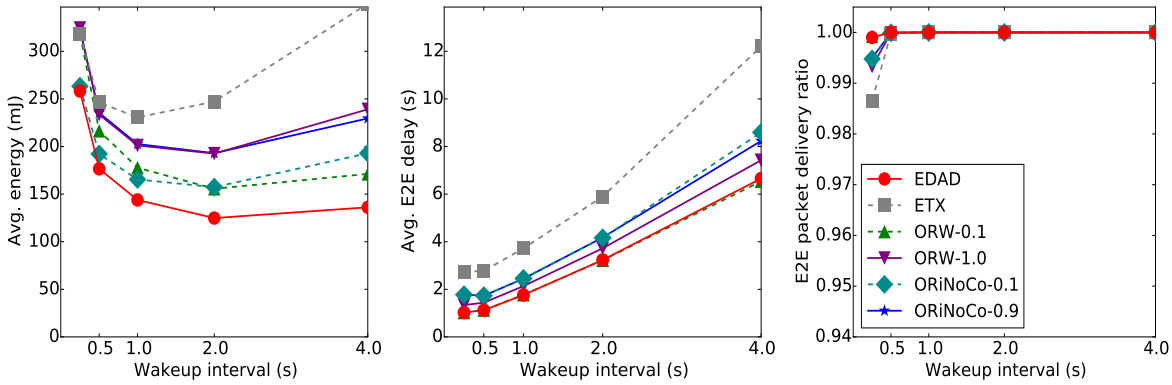
4.4.1.2 Wakeup interval

Fig. 4.8 shows average performance with different wakeup intervals. Most protocols reach an energy minimum at around $T_W = 2$ s. Below this, delay is shorter, implying less routing energy, but this decrease is outweighed by the energy of increased beaconing activity. EDAD again consumes the least energy per packet, around 20% less than ORW-0.1 at $T_W = 2$ s. EDAD also shows the slowest growth in energy per packet for $T_W > 2$ s.

These results also show how EDAD's automatic adjustment to T_W gives it an advantage over other protocols. At $T_W = 0.25$ s, ORW-0.1 shows energy performance similar to ORW-1.0, but as

T_W increases, ORW-0.1 uses less and less energy than ORW-1.0. Also, at $T_W = 0.25$ s, ORiNoCo-0.1 shows energy performance similar to EDAD, but as T_W increases, the separation between ORiNoCo-0.1 and EDAD grows. This behavior is because a larger T_W leads to a greater delay, implying more energy consumption. A larger forwarder set counters that delay. But at a smaller T_W, one-hop delay matters less, so only forwarders that provide greater progress should be used. Only EDAD balances this tradeoff in forwarder set size without parameter readjustment.

Above $T_W = 0.25$ s, all protocols show very high reliability. At $T_W = 0.25$ s, some packet loss occurs due to excessive collisions from beaconing activity.

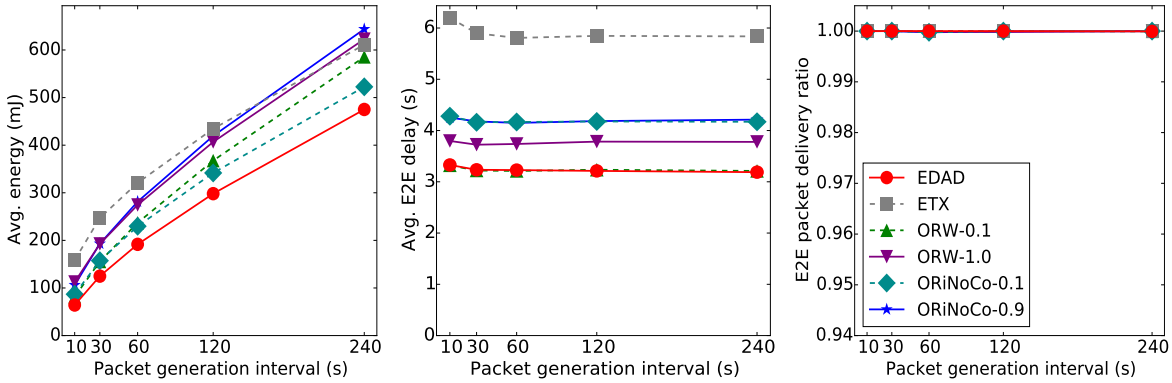


(a) Energy vs. T_W. (b) Delay vs. T_W. (c) Reliability vs. T_W.

Figure 4.8: Performance with different wakeup intervals.

4.4.1.3 Data generation rate

From Fig. 4.9, EDAD shows the least energy consumed per packet over the tested range of data generation intervals. In general, energy per packet increases as the data generation interval increases because when less packets are generated, relatively more of the network energy is consumed by beaconing activity. With data generation intervals above around 30 s, delay remains consistent. Below 30 s, collisions increase delay slightly. Reliability is high for all of the data generation intervals shown, but drops due to excessive collisions at higher data rates.



(a) Energy vs. $1/\lambda$. (b) Delay vs.$1/\lambda$. (c) Reliability vs. $1/\lambda$.

Figure 4.9: Performance with different data generation rates.

4.4.2 Detailed Behavior

4.4.2.1 Anycast route selection

Fig. 4.10 shows traces of two consecutive packets in a simulation using EDAD. At each hop, the packet can go to any one of a number of potential forwarders, resulting in widely different routes, even for back-to-back packets.

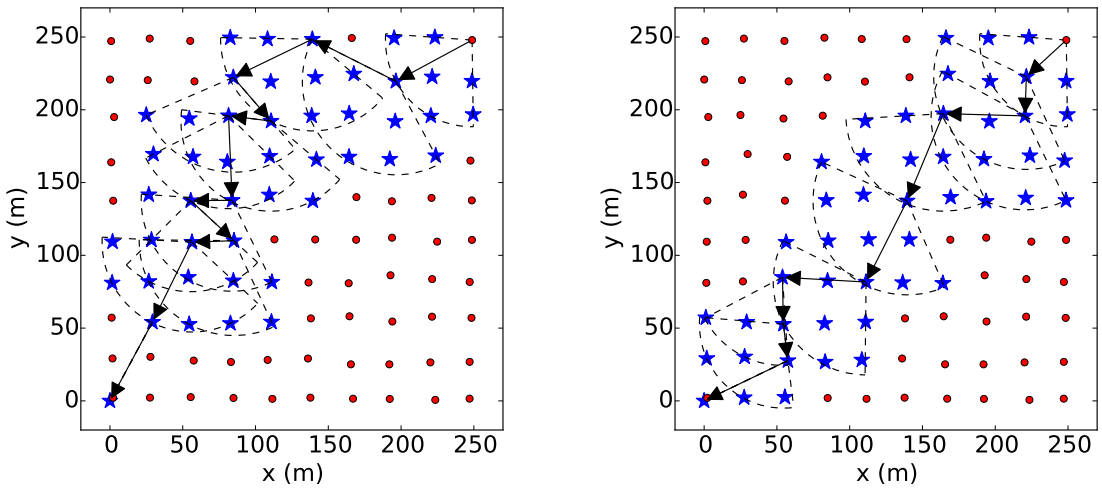
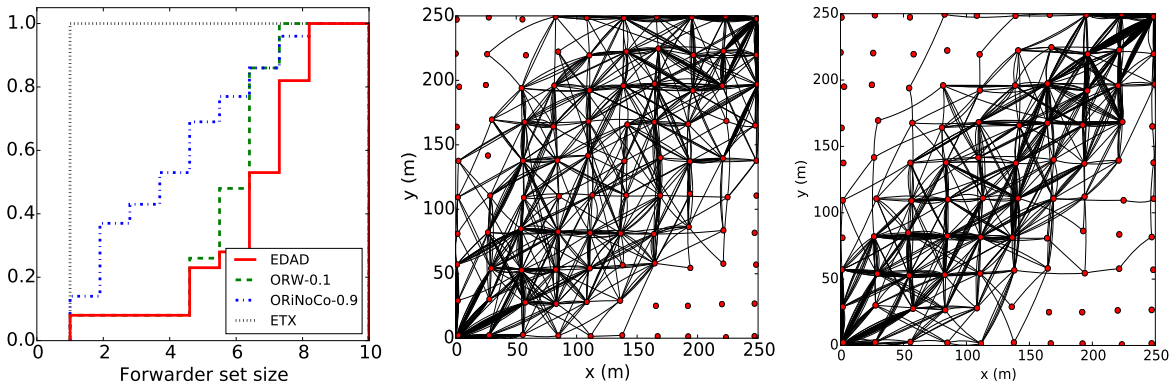


Figure 4.10: Back-to-back traces of packets from a simulation using EDAD. The source is in the upper-right corner and the sink is in the lower-left corner. Arrows indicate the route taken by the packet. For each hop, the forwarder set of the node is shown enclosed in a dashed circle segment, and the potential forwarders are marked with blue stars.

4.4.2.2 Path diversity

Fig. 4.11a shows a sample cumulative distribution function (CDF) for forwarder set sizes of four of the protocols with the default simulation settings. In this case, EDAD uses the most forwarders. Figs. 4.11b and 4.11c show traces of 100 packets in a particular topology for EDAD and ORW-0.1. EDAD shows a noticeably greater diversity in the paths taken by the packets, due to its larger forwarder set sizes. For this value of T_W, more forwarders leads to less energy consumption, so EDAD yields the lowest energy per packet, as seen at $n = 100$ in Fig. 4.7. As previously discussed, at a smaller T_W, the lowest energy may be achieved with fewer forwarders. Since EEP automatically adapts itself to T_W, for this case, EDAD's forwarder set would be smaller.



(a) $|F|$ (b) EDAD (c) ORW-0.1

Figure 4.11: Path diversity. (a) shows the CDF of the size of forwarder sets for a 100-node topology with default simulation settings. With these settings, EDAD selects the most forwarders. (b) and (c) show traces, for EDAD and ORW-0.1, of 100 packets sent diagonally across the simulated area. EDAD's path diversity spread is larger, leading to the energy savings seen for EDAD with these settings.

4.4.2.3 Delay-transmissions tradeoff

Fig. 4.12 shows traces of two consecutive packets from a simulation using EDAD, in a format similar to Fig. 4.4, to illustrate a real example of the tradeoff between shorter E2E delay with more time spent in transmission and longer delay with less time spent in transmission. EEP estimates

the shaded area in the figure, which allows the forwarder set selection process to balance the delay-transmissions tradeoff, and thus minimize energy, by minimizing the EEP for each node.

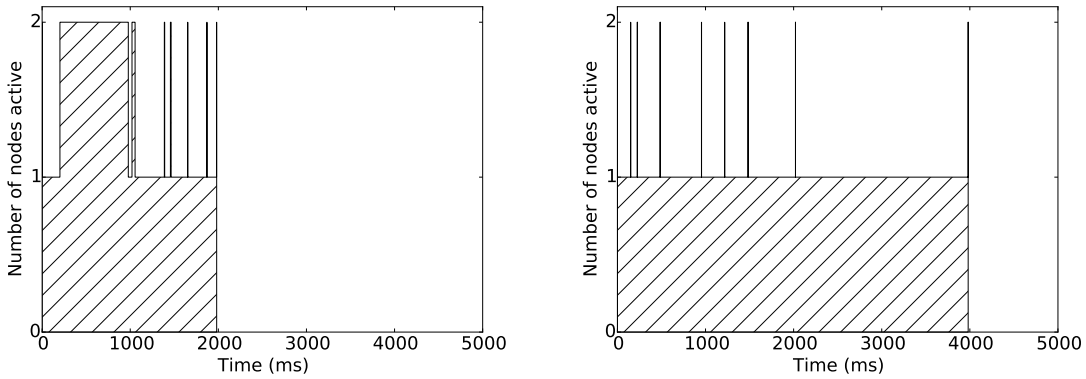


Figure 4.12: The delay-transmissions tradeoff in real traces of consecutive packets routed from the same source node in a network using EDAD. EEP estimates the shaded area.

4.5 Summary

EDAD has been shown to achieve its design goal of further reduction of energy consumption in general-purpose data-collection WSNs using anycast. This reduction comes primarily from EEP, a new anycast routing metric that judiciously selects the forwarder set that minimizes the expected energy consumed along the path of a packet. With EDAD, the lifetime and reliability of WSNs are increased, making WSNs more practical for real deployments in applications such as wind energy infrastructure monitoring and management.

CHAPTER 5. CROSS-LAYER MULTICASTING

5.1 Introduction

With the rise of data-driven applications and the Internet of Things, the use-cases of wireless sensor network (WSN) technologies are becoming clearer and more abundant. However, to cater to these uses, WSNs need to become more flexible, including in terms of traffic pattern support. A traditional data-collection WSN supports an efficient many-to-one traffic pattern, where many *source* nodes send data to a single *destination* node, typically the root of a data collection tree. Some of these data-collection protocols also support data dissemination from the root back down the tree (one-to-many). Also, some of these protocols can be adapted to allow sources to send to any one of multiple potential destinations (many-to-any). However, WSNs lack efficient protocols that support multicasting with any node as the source and any set of nodes as the destinations (many-to-many).

One of the challenges with designing a multicast protocol for a WSN is that the wireless radios in a WSN are often *duty-cycled*, or turned on and off, in order to satisfy energy constraints for nodes. But for a node to receive a packet, the radios of the sender and receiver must both be on. A key challenge with WSNs, then, is to arrange a rendezvous between a sender and its receiver. The longer a sender waits for the receiver to turn its radio on, the more energy the sender consumes. If the sender must wait for multiple receivers, as in multicast, it will consume even more energy. Also, if a single packet must be sent to multiple forwarders to reach all destinations, each of these copies of the packet will consume additional precious energy as they are forwarded through the network. The combination of these considerations makes efficient multicasting in WSNs challenging.

However, we believe that efficient multicasting in WSNs will only become more desirable. Consider an application model based on data *publishers* and data *subscribers* [188]. Publishers are source nodes that periodically produce data from sensor readings. Subscribers are destination

nodes that need this data to perform tasks. An example of this type of application could be an industrial monitoring and control network, where many control nodes need to be regularly updated on the conditions at other points in the network. Another example could be a smart building, where multiple actuators for lighting and HVAC may depend on readings from occupancy, temperature, and light sensors [189].

In recent years, the key metrics of energy, delay, and reliability of data collection have all been improved through the use of an *opportunistic* framework that allows nodes to dynamically make forwarding choices based on current network conditions and events. This gives the sender a large amount of flexibility in forwarding, allowing it to forward sooner and providing more options in case of failure. We propose to extend the concepts of opportunistic data collection to a full-fledged opportunistic multicast framework.

In our framework, detailed in Section 5.3, any node may directly and opportunistically forward a packet to any subset of potential destinations. Routes through the network are dynamically determined, along with decisions of when to send a copy of a packet to multiple forwarders (referred to hereafter as *splitting* the packet, for simplicity) so that it reaches all destinations efficiently. These processes are controlled by the *forwarder set selection* and *destination delegation* methods, also detailed in Section 5.3. We use simulations to evaluate our proposed framework in Section 5.4. We find that our framework provides significant reductions in energy and delay compared to current multicasting methods for duty-cycled WSNs.

5.2 Related Work

Energy efficiency through duty-cycling has been a constant research topic for WSNs for well over a decade, as summarized in surveys such as [159]. At the link layer, protocols such as B-MAC [16], X-MAC [17], and ContikiMAC [18] have achieved progressively more efficient rendezvous without the need for synchronization or explicit scheduling between nodes, using a technique called *low-power listening*. Receiver-initiated protocols such as RI-MAC [19] use *low-power probing* to achieve similar results. At the network layer, protocols such as CTP [190] and IETF's RPL [7] natively

support many-to-one data collection. More recently, *opportunistic* data collection for duty-cycled WSNs has been developed and tested in protocols such as ORW [176] and our own EDAD [25]. These opportunistic data collection protocols use similar frameworks and differ mainly in the routing metric used for forwarding decisions, and also in the choice of either a sender- or receiver-initiated link-layer protocol. We note that opportunistic data collection has also been referred to as "anycast" data collection in the past, because it uses link-layer anycasting; however, to avoid confusion with our goal of network-layer multicasting, we use the term "opportunistic" here.

We classify approaches to multicasting in WSNs into four categories: optimal multicast trees (Steiner trees), flooding, down-tree routing, and multi-tree routing. Schemes that build (pseudo-)optimal multicast trees, such as [191, 192, 193, 194], typically either do not consider duty cycling or rely on wakeup scheduling, and thus do not meet our energy efficiency and flexibility goals. In flooding schemes, such as IETF's MPL [195], each multicast packet is sent to every node in the domain. These schemes are inefficient at addressing smaller groups of nodes. A subset of protocols related to flooding, such as LWB [196] and Chaos [197], use the tightly synchronized transmissions of Glossy [198] to quickly disseminate packets to all participating nodes; however, these protocols are restrictive in terms of applications, scalability, and interoperability, making them unsuitable for our goal of flexible multicasting.

Down-tree routing is our name for a class of protocols that use an existing data collection tree structure [190] (or directed acyclic graph), rooted at the source, to support a one-to-many traffic pattern. To enable down-tree routing, each node stores the source addresses of the packets that it forwards up the tree in a *routing set*. The RPL standard defines a *storing mode* that builds these routing sets. Protocols such as SMRF [199] and BMRF [200] use RPL's storing mode to implement multicast. ORPL [171] implements an opportunistic version of down-tree routing that allows shortcuts across tree branches. REMI [201] builds clusters on top of the RPL tree and multicasts across clusters in addition to up and down the tree.

Even with these shortcuts, down-tree routing's reliance on the tree structure rooted at the source imposes limitations on its multicasting capabilities. In contrast, the *multi-tree routing* approach uses

multiple tree structures, one rooted at each destination. These protocols then aggregate branches of the separate trees to form "multi-source, multi-sink" trees [189]. Examples of these protocols include MUSTER [189] and FROMS [202]. Our proposed multicasting framework combines this approach with opportunistic forwarding, producing a highly dynamic, flexible, and efficient multicasting process.

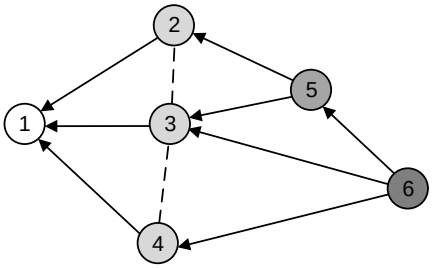


Figure 5.1: Single-sink opportunistic data collection with a routing metric gradient to node 1.

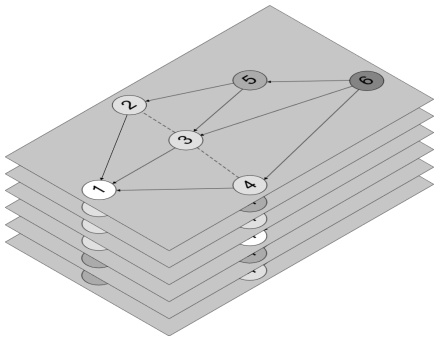


Figure 5.2: Opportunistic multicast framework based on gradients to each possible destination.

5.3 Opportunistic Multicast Framework

Our opportunistic multicast framework builds on the opportunistic data collection framework described in Chapter 4. Similar to [202], each node stores a routing metric value for each potential destination, creating a full-fledged *gradient stack*, shown in Fig. 5.2. The forwarding process for a multicast packet in our framework is shown in Fig. 5.3. The first step is *forwarder set selection*, which determines the next hops for the sender to consider. Once a forwarder in the forwarder set has awoken, the sender chooses which destinations will be handled by that forwarder, called *destination delegation*. A copy of the packet addressed to those destinations is sent to the forwarder, and the forwarding process repeats until all destinations have been delegated. The structure of the framework and the steps in the forwarding process are described in more detail in the following sections.

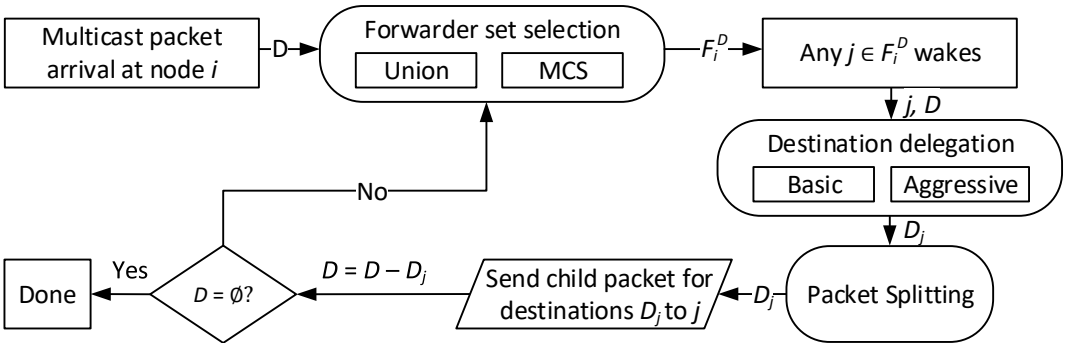


Figure 5.3: Overview of our multicast framework. In each iteration, a child packet is sent to the first forwarder in the forwarder set (selected via either the Union or the MCS method) to wake. The packet is addressed to some subset (selected via either the Basic or Aggressive method) of the destinations. The process repeats until all destinations have been delegated.

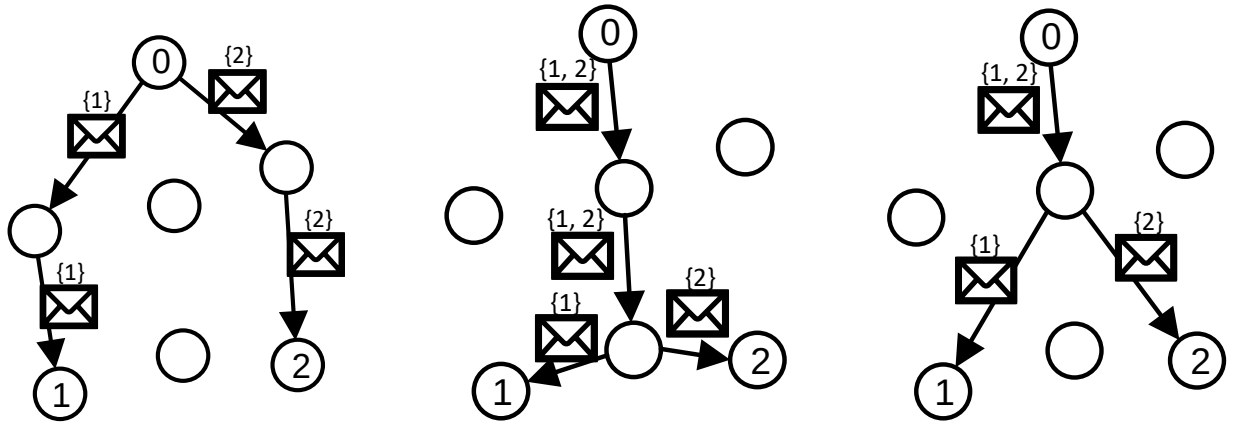
5.3.1 Gradient Stack

In our opportunistic multicast framework, each node stores a routing metric value for each potential destination, resulting in a routing metric gradient for each potential destination. We refer to these gradients as the *gradient stack*, conceptualized in Fig. 5.2. The gradient stack allows for highly dynamic, opportunistic multicasting, because each node has local information about how to reach each destination individually. This information can be dynamically merged, depending on the relevant destinations, to make multicasting decisions. The process of creating these gradients is the same as that of a traditional many-to-one gradient, with each potential destination acting as the root of a data collection DODAG.

5.3.2 Destination Grouping and Splitting

When a source generates a multicast packet, all of the multicast destinations are *grouped* into it. As shown in Fig. 5.4, as the packet is forwarded, it is *split* into child packets, each with a disjoint subset of the destinations, in order to reach all destinations. The choice of when to group and when to split destinations affects the efficiency of packet delivery and is a key consideration of our multicast framework. Splitting too early, as in Fig. 5.4a, creates excess copies of the packet that need to be separately forwarded through the network. Splitting too late, as in Fig. 5.4b, can

result in unnecessary hops. In our framework, the decisions for when to group and when to split are made by the forwarding process, described below.



(a) Splitting too early creates excess copies of the packet.

(b) Splitting too late creates unnecessary hops.

(c) Efficient splitting results in less network activity.

Figure 5.4: Example of how splitting behavior affects efficiency of packet delivery. In this example, node 0 has a multicast packet to deliver to nodes 1 and 2.

5.3.3 Forwarder Set Selection

As shown in Fig. 5.3, the first step of the forwarding process is forwarder set selection: given a multicast packet addressed to a set of destinations D, the sender must decide which neighbors to consider as forwarders. This set of neighbors is called the *forwarder set* and, for a node i, is denoted F_i^D. Each forwarder in the forwarder set provides enough "progress" to warrant sending to that forwarder. If a forwarder provides progress toward a particular destination d, we say that forwarder *covers* d. For a multicast packet, multiple forwarders may be required to cover all $d \in D$, and at least one node in the forwarder set must cover each destination.

Defining "progress" in a multicast sense is difficult. We first considered extending the design of the single-sink gradient to a multicast scenario by defining a meta-metric that combines the metrics for each $d \in D$ to produce one multicast gradient value. However, this approach would require knowledge of the meta-metric for each neighbor, and since the meta-metric would be different for each possible D, this approach is impractical. We therefore propose two simpler, heuristic-based

methods of choosing forwarders. These methods, Union and MCS, use only the information found in the gradient stack.

5.3.3.1 Union

The forwarder set $F_i{}^D$ for a node i, given a set of destinations D, is the union of the forwarder sets for each individual destination $d \in D$. The heuristic used here is that if a node is a good enough forwarder for at least one destination, then it is worth sending to that node.

5.3.3.2 MCS

The forwarder set is the union of all minimum covering sets (MCSs), where a minimum covering set is defined as a set with minimal members for which each $d \in D$ is covered by at least one member. This is a local attempt to minimize the number of packet splits. The union is taken to allow the wakeup order of the forwarders to determine the actual minimum covering set used.

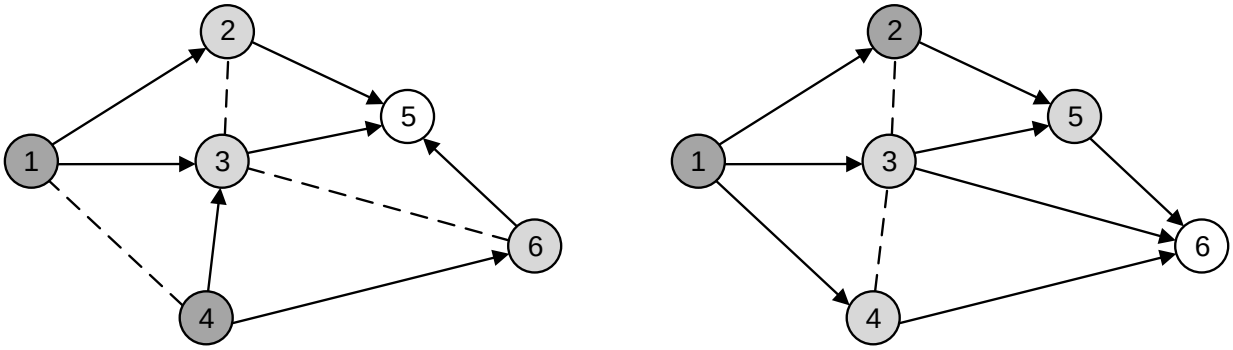
As an example, we use the gradients shown in Fig. 5.5 and consider a multicast packet with $D = \{5, 6\}$. Tables 5.1 and 5.2 show the forwarder sets created using Union and MCS, respectively. In this example, using Union, Node 1's forwarder set for destination 5 is $\{2, 3\}$, and for destination 6, $\{2, 3, 4\}$. The forwarder set $F_1^{\{5,6\}}$ is the union of these two sets, $\{2, 3, 4\}$. Using MCS, node 1's forwarder sets for destinations 5 and 6 both contain node 2, so $\{2\}$ is an MCS for node 1, because no sets exist that can cover both destinations with fewer nodes. Likewise, $\{3\}$ is also an MCS. The forwarder set $F_1^{\{5,6\}}$ is the union of these MCSs, $\{2, 3\}$.

Table 5.1: Forwarder sets selected for $D = \{5, 6\}$ using Union.

i	$F_i^{\{5\}}$	$F_i^{\{6\}}$	$F_i^{\{5,6\}}$
1	$\{2, 3\}$	$\{2, 3, 4\}$	$\{2, 3, 4\}$
2	$\{5\}$	$\{5\}$	$\{5\}$
3	$\{5\}$	$\{5, 6\}$	$\{5, 6\}$
4	$\{3, 6\}$	$\{6\}$	$\{3, 6\}$
5	$\{\}$	$\{6\}$	$\{6\}$
6	$\{5\}$	$\{\}$	$\{5\}$

Table 5.2: Forwarder sets selected for $D = \{5, 6\}$ using MCS.

i	MCSs	$F_i^{\{5,6\}}$
1	$\{2\}$, $\{3\}$	$\{2, 3\}$
2	$\{5\}$	$\{5\}$
3	$\{5\}$	$\{5\}$
4	$\{6\}$	$\{6\}$
5	$\{6\}$	$\{6\}$
6	$\{5\}$	$\{5\}$



(a) Gradient for destination 5.

(b) Gradient for destination 6.

Figure 5.5: Example network with gradients shown for destinations 5 and 6.

Union naturally creates larger forwarder sets than MCS. This gives Union an advantage in sparser networks, because in this case, adding even one more forwarder can significantly reduce the forwarding delay. In dense networks with many potential forwarders, MCS is penalized less for being more selective. Additionally, MCS's selectivity decreases the number of splits and the amount of traffic in the network. This analysis is supported by our simulations in Section 5.4.

5.3.4 Destination Delegation

In our opportunistic framework, the sender i sends a packet to the first forwarder $j \in {F_i}^D$ that wakes. As shown in Fig. 5.3, the next step is *destination delegation*, the process of choosing which destinations to group into the packet to j. We propose the following two methods:

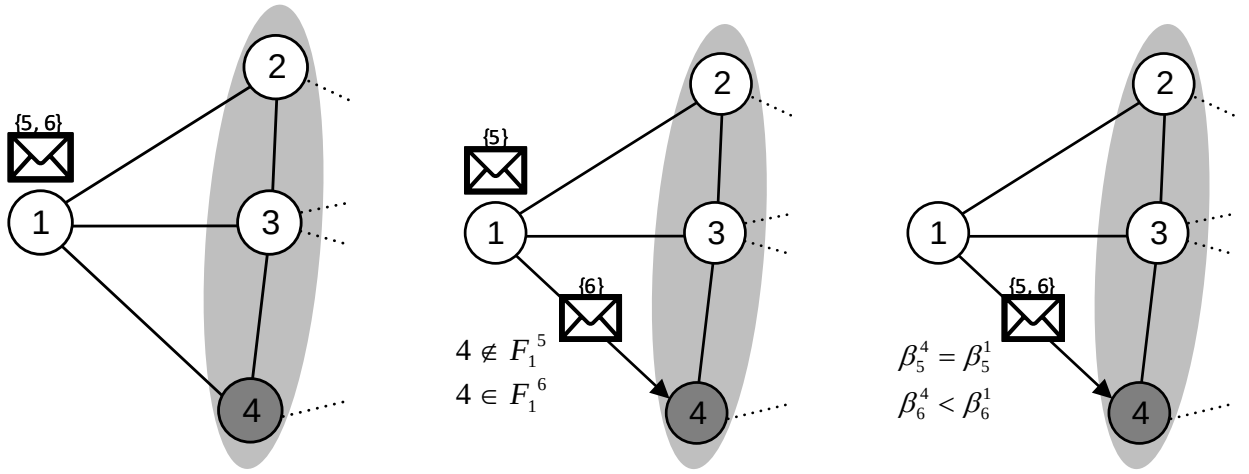
5.3.4.1 Basic

All destinations for which the forwarder provides *sufficient* progress, as determined by the gradient stack, are delegated. Formally, assume sender i has a multicast packet for a set of destinations D, and a forwarder $j \in {F_i}^D$ is available. Then D_j, the set of destinations delegated to j, is the set of all destinations d for which $j \in F_i^{\{d\}}$.

5.3.4.2 Aggressive

All destinations for which the forwarder provides *any* progress are delegated. Formally, if β_i^d is the routing metric value for destination d at node i, then D_j is the set of all destinations d for which $\beta_j^d \leq \beta_i^d$.

An example of the two delegation methods, using the topology of Fig. 5.5, is shown in Fig. 5.6. In this example, a packet for 5 and 6 has arrived at node 1. The forwarder set (using Union) is $F_1^{\{5,6\}} = \{2, 3, 4\}$, and node 4 wakes first. Using Basic delegation, the packet is split and only 6 is delegated to node 4. Node 1 must then start a second iteration of the cycle shown in Fig. 5.3, with $D = \{5\}$. But using Aggressive delegation, both destinations are delegated to node 4.



(a) Node 4 wakes first, and node 1 must determine D_4, the set of destinations to delegate to node 4.

(b) Basic delegation: $D_4 = \{6\}$ because node 4 is in 1's forwarder set for 6, but not for 5.

(c) Aggressive delegation: $D_4 = \{5, 6\}$ because 4's routing metric to both 5 and 6 is no larger than 1's.

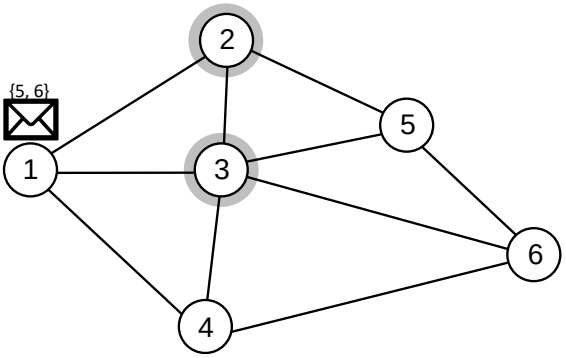
Figure 5.6: Example of delegation mechanisms. Node 1 has a packet with $D = \{5, 6\}$. The shaded area indicates $F_i^{\{5,6\}}$, selected using Union.

From this example, we can see the motivation for Aggressive delegation. Node 4 does not provide significant progress toward node 5 from node 1, but there is a chance that one of node 4's forwarders does provide good progress to both 5 and 6. If such a forwarder wakes next, then Aggressive delegation has both allowed node 1 to finish sending earlier and prevented an excess packet split from occurring. If not, the drawback is minimal, as the packet from 1 to 4 was being

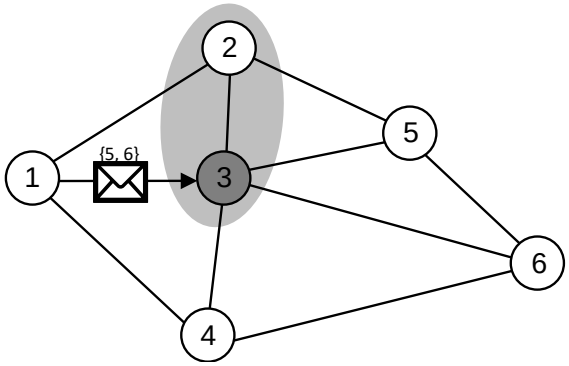
sent regardless, and the hop from node 1 to node 4 provides non-negative routing progress toward destination 5. Our simulation results have shown that Aggressive delegation is generally effective.

5.3.5 Summary and Example

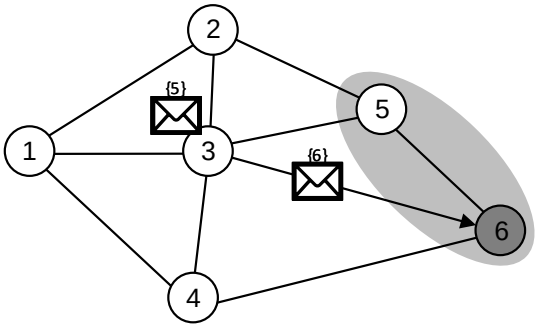
Returning to Fig. 5.3, after destination delegation, a child packet bound for D_j is sent to node j. D_j is then removed from D, and if D is now empty, node i is finished sending the packet. Otherwise, node i repeats the cycle, starting with forwarder set selection, using the updated D. We conclude the description of our opportunistic multicast framework with a small example, shown in Fig. 5.7. The gradient stack for this example is the same as Fig. 5.5.



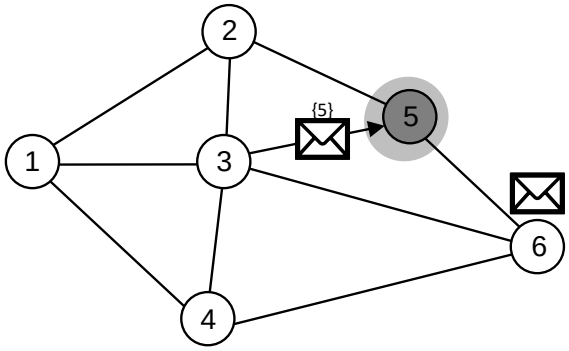
(a) A packet for $D = \{5, 6\}$ arrives at node 1, which is using MCS forwarder set selection. Both $\{2\}$ and $\{3\}$ are minimum covering sets.



(b) The forwarder set $F_1^{\{5,6\}}$ is the union of these sets. Node 3 wakes first and provides progress to 5 and 6, so node 1 delegates 5 and 6 to 3.



(c) Node 3 now has the packet with $D = \{5, 6\}$. The minimum covering set is $\{5, 6\}$. Node 6 wakes first, so node 3 splits the packet and sends to node 6 with $D_6 = \{6\}$.



(d) Node 3 still needs to deliver the packet to node 5, so it waits until node 5 wakes and then sends to node 5 with $D_5 = \{5\}$. All destinations have been reached.

Figure 5.7: Example of delivery of a packet using MCS forwarder set selection and Basic delegation.

5.4 Evaluation

We evaluated our multicast scheme with an event-based simulator written in Python. The simulator uses a distance-based path loss model to determine received signal strength, which is combined with a noise floor to obtain a signal-to-noise ratio (SNR). We then use an empirically-derived function from TOSSIM [186] to obtain packet reception ratio (PRR) from the SNR. The simulated link layer is an opportunistic version of RI-MAC [19]. We simulate basic channel contention and packet retransmissions. The default simulation settings are shown in Table 5.3. The frame duration settings were selected assuming a 250 kbps radio, as in IEEE 802.15.4, and packets of less than 100 bytes.

In each simulation topology, one source node is placed at each corner of the rectangular deployment, with a fifth source placed in the center. The other nodes, including the destinations, are randomly deployed in a connected topology. For each topology, each source node generates a total of 100 packets at a specified interval, with initial packets scheduled randomly. Each packet is addressed to the same set of destinations. This mimics a scenario in which the destinations are "subscribed" to data being generated by a group of sources. The results below are averaged over at least 50 topologies. All schemes use the same duty-cycled link layer. All tests are performed with the network at steady state.

Table 5.3: Simulation default settings.

Parameter	**Default**
Path loss exponent	3.0
Noise floor	-100 dBm
Deployment area	100x100 m
# nodes (n)	100
# sources (s)	5
# destinations (d)	10
Data interval (T_D)	60 s
Wakeup interval (T_W)	500 ms
Wakeup duration (T_B)	1 ms
Frame duration (T_F)	3 ms

We evaluate three combinations of our forwarder set selection methods and destination delegation methods: Union with Aggressive delegation (Union+Agg), Union with Basic delegation (Union+Basic), and MCS with Aggressive delegation (MCS+Agg). We use the EEP routing metric [25] for these protocols. We do not include results for MCS with Basic delegation, which performs worse than MCS+Agg.

We implemented a down-tree routing scheme, which we call DownTree-Opp, to compare to our framework. DownTree-Opp uses an opportunistic framework, similar to ORPL [171] but with EEP as the routing metric. We use EEP to focus the comparison on frameworks and not routing metrics; however, we observed nearly identical performance when using EDC, which is ORPL's routing metric. In our simulations, each source is the root of a data collection DODAG, a best-case scenario for DownTree-Opp.

Finally, we implemented a scheme we call Steiner, which represents an optimal multicast solution for a single-forwarder framework, such as that employed by FROMS [202]. This omniscient, centralized scheme exhaustively searches for the minimum-cost subgraph that connects each source with all destinations (the minimum Steiner tree on the graph). Packets are then routed on this tree. We use ETX as the cost metric.

5.4.1 Wakeup Interval Calibration

The energy performance strongly depends on the wakeup interval T_W and its relation to the traffic rate. Therefore, we first calibrate the network wakeup interval by testing a variety of wakeup intervals to find the optimal wakeup interval for our traffic settings. The average network duty cycle versus T_W is shown in Fig. 5.8. The duty cycle is calculated as the radio on-time divided by the total simulation time. If T_W is too small, excessive on-time is spent on wakeups. If T_W is too large, excessive on-time is spent on idle listening. From this figure, we see that the best T_W for our traffic load is around 500 ms, which we use for the remainder of our evaluations; however, we observed that trends in performance remain similar, regardless of wakeup interval optimality.

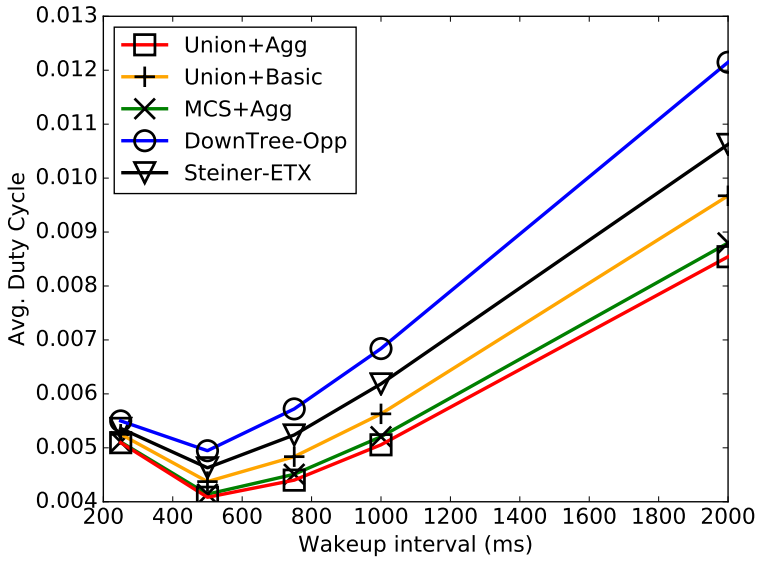


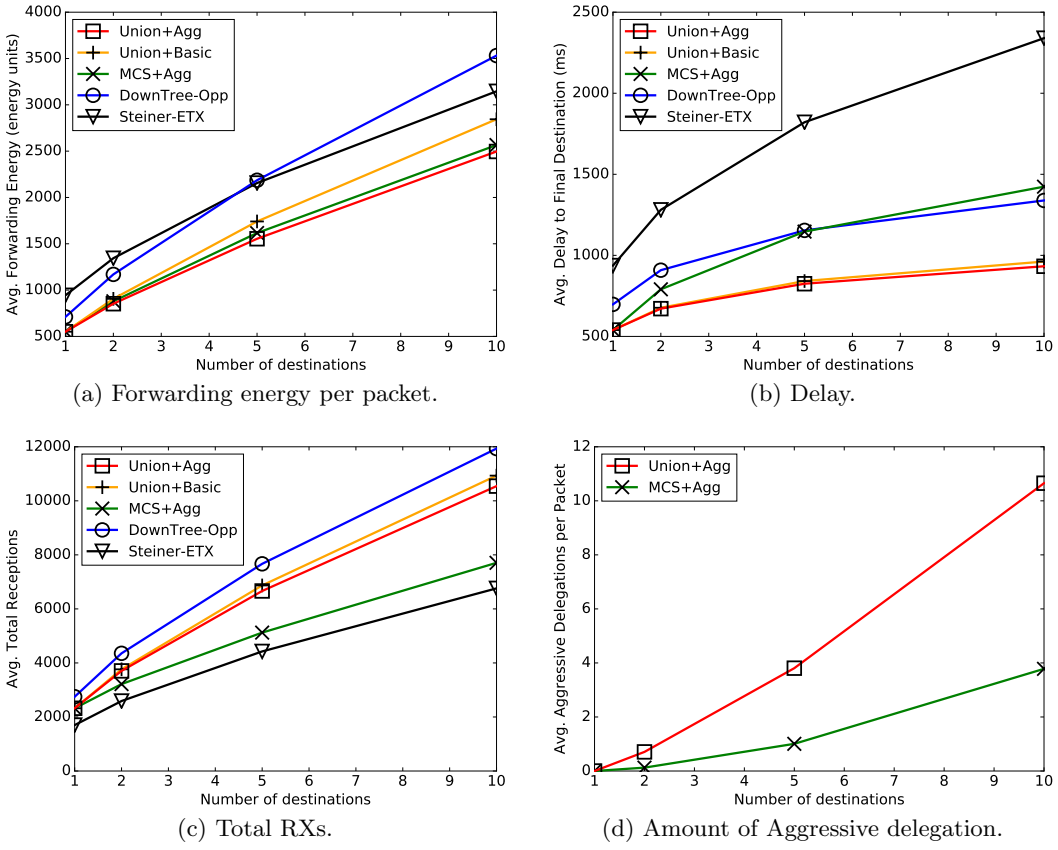
Figure 5.8: Average node duty cycle vs. wakeup interval.

5.4.2 Effect of the Number of Destinations

Fig. 5.9 shows the effect of the number of (randomly-deployed) destinations to which each source sends each packet. Fig. 5.9a shows the *forwarding energy*, which we define as the energy spent on idle listening, TXs, and RXs due to packet forwarding. This quantity does not include the baseline energy load of periodic wakeups. In short, forwarding energy represents the additional energy load imposed on the system from use of the multicast scheme. We measure forwarding energy in *energy units*, defined as the amount of energy consumed in one millisecond of radio on-time.

Forwarding energy per packet increases with the number of destinations for each packet, which is expected—more destinations means more work. Aggressive delegation helps energy performance for Union (and also MCS). Union+Agg uses the least energy for the numbers of destinations shown here. Steiner-ETX uniformly uses more energy more than Union+Agg, meaning that the benefits of Union+Agg's opportunistic framework outweigh the drawbacks of using non-optimal forwarders. Steiner-ETX does perform better than DownTree-Opp with five or more multicast destinations.

Fig. 5.9b shows delay, which we measure as the time required to deliver a packet from the source to all of the multicast destinations. Delay increases with the number of destinations, which



(a) Forwarding energy per packet.

(b) Delay.

(c) Total RXs.

(d) Amount of Aggressive delegation.

Figure 5.9: Evaluation for different numbers of destinations.

makes sense, because more destinations increases the chance that at least one of them will be far from each source. The Union forwarder set selection method provides the best delay performance, because Union chooses more forwarders, sending to the first forwarder good enough for any one destination, whereas MCS waits for a forwarder that is good for some number of destinations. The Aggressive delegation mechanism provides a small delay performance gain. DownTree-Opp's delay is competitive with MCS+Agg, but not Union. Steiner-ETX has the worst delay performance, because its adherence to an optimal tree structure means that senders must always wait for a particular forwarder to wake.

Comparing the delay and energy figures, the energy performance of Union and MCS appears surprisingly close. Fig. 5.9c provides a clue as to how MCS compensates for its lackluster delay,

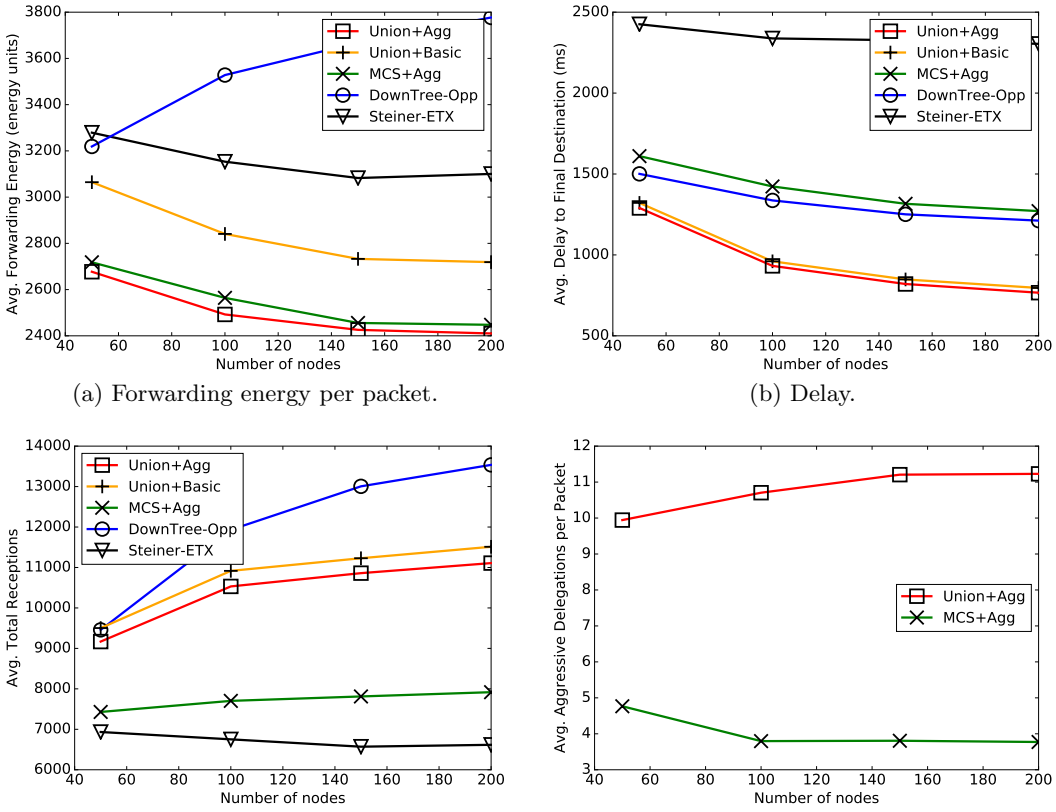
showing the average total number of successful RXs over the course of the simulations, which is directly related to the number of times each packet is split on the way to its destinations. MCS splits packets considerably less often than Union or DownTree-Opp. We therefore note that MCS may perform better than Union in a high-traffic scenario where congestion is a factor. Steiner-ETX splits even less than MCS, due to its optimal tree structure. The tradeoff for our proposed opportunistic framework's superior energy and delay performance, then, is higher amounts of network traffic.

Fig. 5.9d shows the number of destinations "aggressively" delegated, meaning those destinations assigned to a forwarder by Aggressive delegation that would not be assigned by Basic delegation. We see that Union makes more use of Aggressive delegation than MCS, which is one way Union compensates for its use of lower-quality forwarders.

5.4.3 Effect of the Network Density

Fig. 5.10 shows results for performance with different numbers of nodes in the fixed simulation area, measuring the effect of network density. Fig. 5.10a shows forwarding energy. Our proposed schemes again have better energy performance than DownTree-Opp and Steiner-ETX. For our schemes, forwarding energy decreases with the number of nodes due to the increased options for forwarder sets. Delay (Fig. 5.10b) also decreases, for the same reason.

For our randomly-deployed networks, fewer nodes often leads to irregular topologies. The effects of this irregularity can be seen in the figures, with a "regularity threshold" emerging around 100 nodes. For example, in Fig. 5.10c, the number of splits stays fairly steady above 100 nodes. Below this point, route options tend to be limited, forcing destinations to be grouped together for longer. Interestingly, this forced grouping actually helps the DownTree schemes in energy performance at 50 nodes. When more routes open up at 100 nodes, the DownTree schemes split more often. Splits in a DownTree scheme tend to be more costly because a packet from a source to a destination in a DownTree protocol must follow a route conforming to the existing data collection tree or DODAG structure that is rooted at the source. The splits therefore tend to happen earlier on the path of a



(a) Forwarding energy per packet.

(b) Delay.

(c) Total RXs.

(d) Amount of Aggressive delegation.

Figure 5.10: Evaluation for different numbers of nodes in a fixed area.

packet, where more of the tree branches overlap, which means each child packet must be forwarded farther after the split.

5.4.4 Performance Summary

For our testing scenario, Union+Agg is the most energy-efficient variation of our framework. Aggressive delegation uniformly outperforms Basic delegation. Union forwarder set selection produces less delay than MCS forwarder set selection, but MCS produces less network activity. Our framework outperforms a representative state-of-the-art protocol, DownTree-Opp (which is operating in a best-case scenario here), showing the advantage of the gradient stack approach. Finally, our opportunistic framework outperforms Steiner-ETX, a single-forwarder solution operating on an

optimal tree. We conclude that our dynamic and flexible approach is more suited to duty-cycled WSNs than rigid, "optimal" approaches.

5.4.5 Traces

To illustrate the dynamic nature of our multicast framework, we present two traces of Union+Agg in Fig. 5.11. All parameters, including routing metric values, are the same for these two traces, but the paths taken by the packets are different. This random-like behavior makes the framework robust, and also helps to balance the energy load between the nodes.

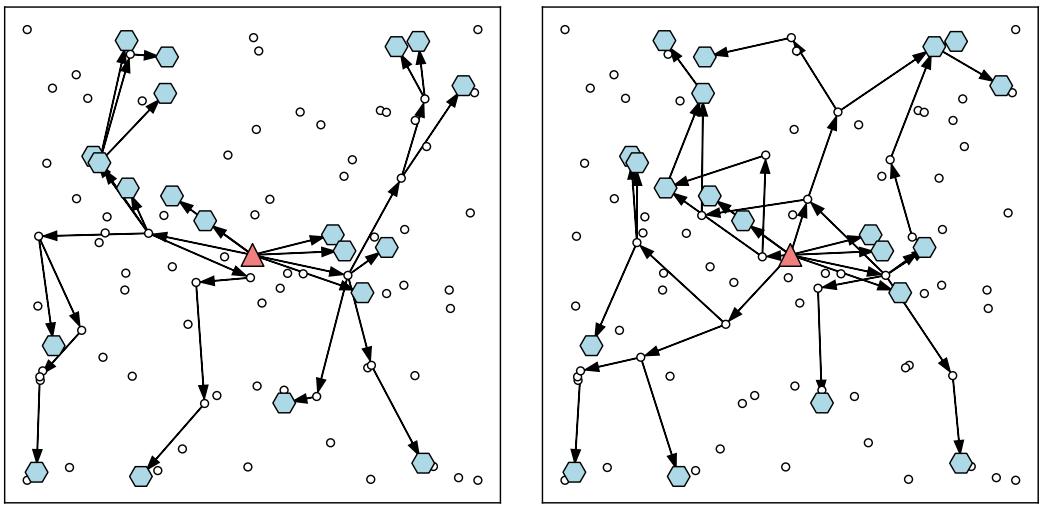


Figure 5.11: Two traces of the paths taken to deliver a single multicast packet with Union+Agg. The red triangle in the center is the source, and the blue hexagons are destinations. Arrows show packet transmissions.

5.5 Conclusion

We have proposed an opportunistic multicast framework for duty-cycled wireless sensor networks that provides flexible, efficient, and reliable multicasting. Our framework can be used to provide low-energy, low-delay multicasting for a variety of applications. Future work includes in-depth analysis of the overhead of our framework, methods for reducing memory consumption and network maintenance costs, and implementation of our framework in Contiki OS.

CHAPTER 6. APPLICATION-SPECIFIC PROTOCOL

6.1 Introduction

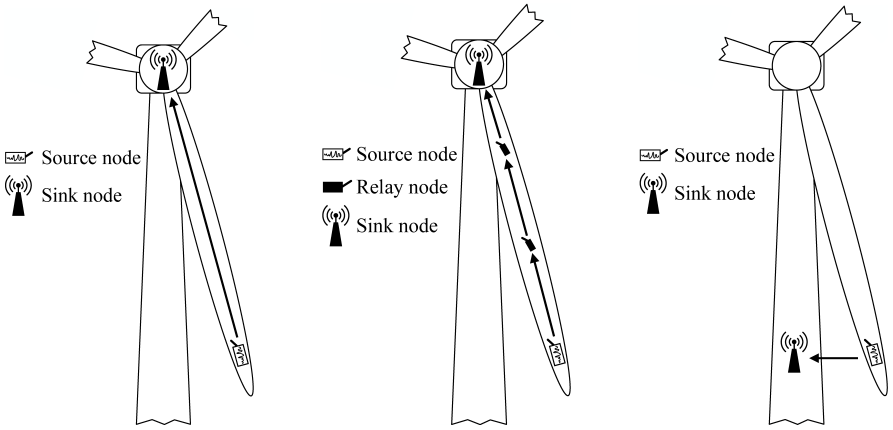
Renewable energy, and particularly wind energy, has seen massive growth in the past few decades [203] due to factors such as decreasing costs, new government policies and incentives, and increased concern about fossil fuels and their effects on our planet. Wind turbines are continually growing in size, 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. [204]). 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 [6]. 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 [131], 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) [205] or a smart sensing skin [206] for wind turbine blades. Section 6.7.1 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. 6.1a and 6.1b. A source deployed at the end of a blade could attempt to communicate with the sink in a single hop, as shown in Fig. 6.1a. However, the single-hop distance could be 50 m or more on modern wind turbines [207], which is not within the communication range of typical radio hardware that meets our size and power constraints [208]. Therefore, the single-hop approach is not practical for our scenario.



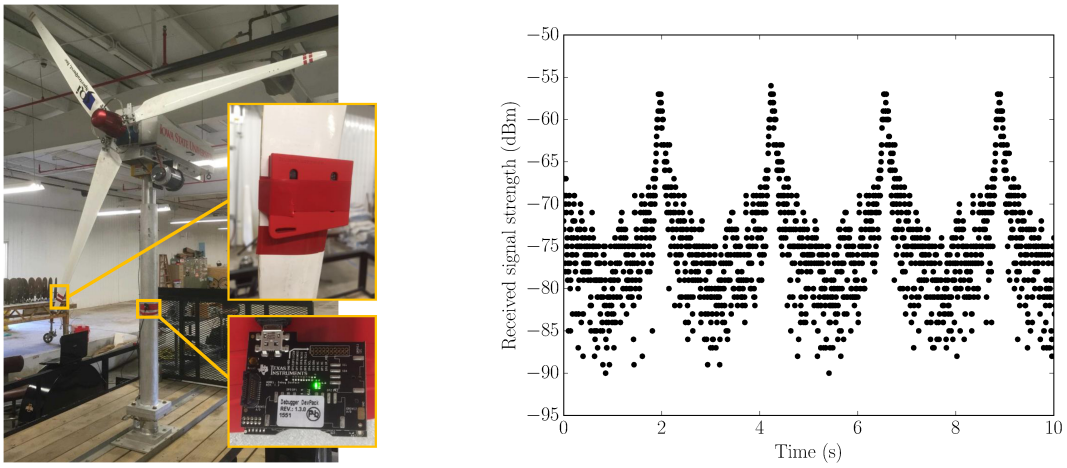
(a) Single-hop. (b) Multi-hop. (c) Opportunistic.

Figure 6.1: Three approaches to WSN deployment on wind turbine blades. BladeMAC uses the approach shown in (c).

Another option is to use relay nodes arranged in a multi-hop configuration along the blade, as shown in Fig. 6.1b. The source then only needs to transmit data over a short distance to the next relay node. However, since the relay nodes are also 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 avail-

able 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. 6.1c. 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 [209], as shown in Fig. 6.2.



(a) Lab-scale wind turbine.

(b) RSS trace.

Figure 6.2: A SpectraQuest [210] 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 [209]. 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.

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 [27].

In the following section, we examine related work. In Section 6.3, we present possible baseline solutions for our problem scenario. The shortcomings of these baseline solutions motivate the design of BladeMAC, presented in Section 6.4. In Section 6.5, we describe BladeMAC's method for estimating the length of time available for communication in each channel cycle. In Section 6.6, we detail our implementation of BladeMAC in Contiki OS [20], our experimental methods, and our evaluation results. We discuss possible applications and a variety of practical considerations for BladeMAC in Section 6.7. Finally, we conclude the paper in Section 6.8.

6.2 Related Work

Duty-cycled MAC protocols have been described in previous chapters; 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 [211] 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 [212]. Wireless communications in a rotating environment have also been studied in the context of tire pressure monitoring systems [213], which, due to the relaxed deployment constraints, can use standard protocols [214]. Researchers have also explored saving energy through planning (re)transmissions based on channel state and statistics [215, 216]. 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.

6.3 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.

6.3.1 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:

$$\min E_{SOURCE} = P_{TX}T_{TX} + P_{RX}T_{RX} + P_{IDLE}T_{IDLE},$$
$$s.t. \quad P_{SINK} \leq P_{SINK}^{MAX}, \quad D \leq T_{\Omega}, \quad (6.1)$$

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 (6.1) through protocol design. Assuming P_{TX}, P_{RX}, and P_{IDLE} are equal (essentially true for WSN hardware such as [209]), the minimization in (6.1) 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 6.6. 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.

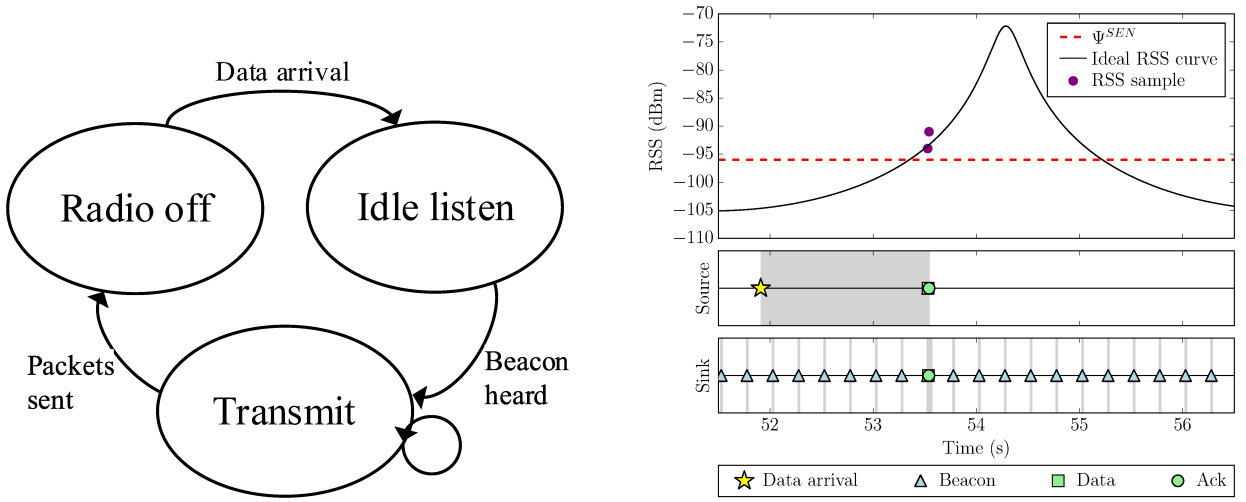
6.3.2 CC-MAC

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 (6.1) 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 (6.1).

A state machine for the source in CC-MAC is shown in Fig. 6.3a, with a sample trace of CC-MAC's operation shown in Fig. 6.3b. 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 (6.1).



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

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

Figure 6.3: 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.

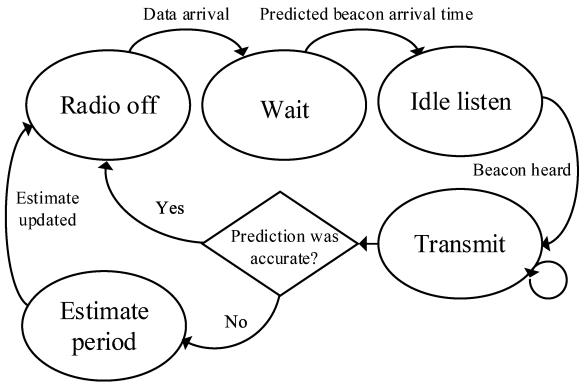
We emphasize that, in CC-MAC, the source and sink have separate behaviors. The source does not periodically wake and beacon, meaning that it does not expend energy on communications unless it has data to send. If the sink needs to communicate with the source, it can piggyback command and control information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

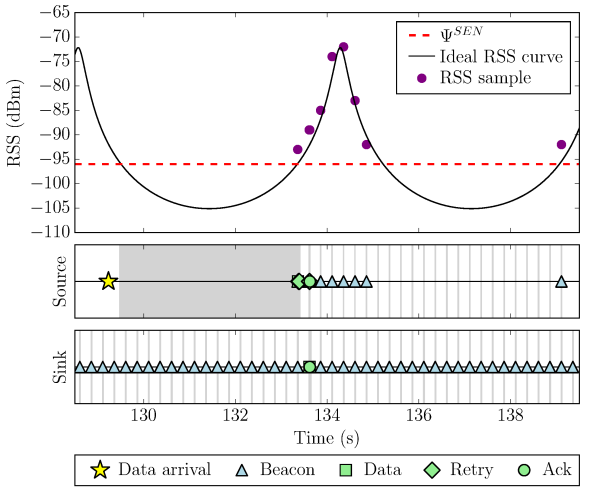
- **Pros:** (a) CC-MAC is simple; (b) CC-MAC is sufficient if the interval between data arrivals is very long.
- **Cons:** (a) CC-MAC is sensitive to the data arrival interval; (b) CC-MAC may transmit data while the channel is still in poor condition, requiring excessive retries.

6.3.3 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. 6.4). 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. 6.4a, 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.



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



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

Figure 6.4: Overview of CPCC-MAC.

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. 6.4b. 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 (6.1). 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.

6.4 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.

6.4.1 Overview

Fig. 6.5 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.

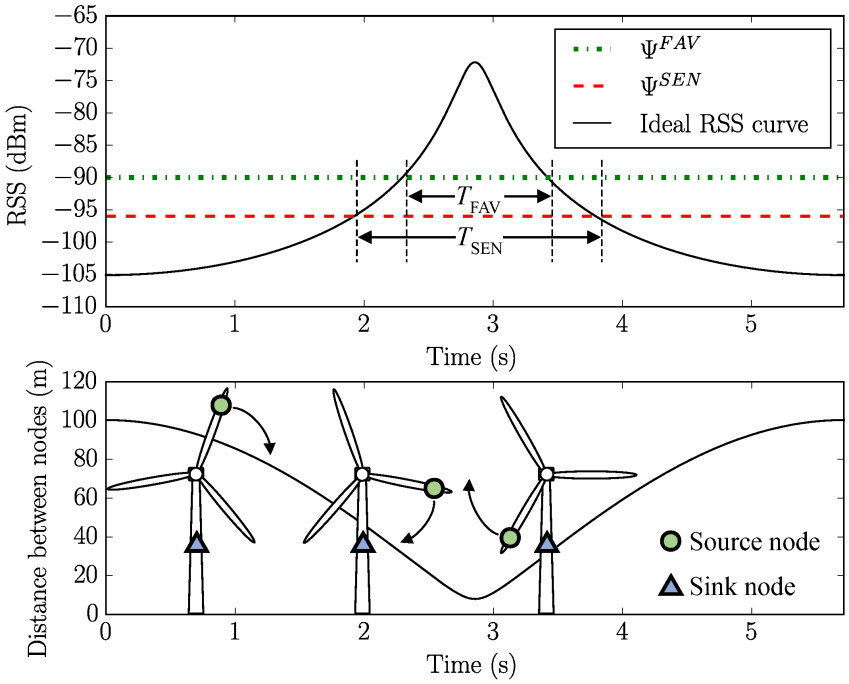


Figure 6.5: Illustration of the cyclic channel caused by the rotation of the blades. We define the intervals T_{FAV} and T_{SEN} based on Ψ^{FAV}, Ψ^{SEN}, and the rotation speed.

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 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 idle listening (see 6.1) 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. 6.6 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 6.5. 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.

6.4.2 Sink Behavior

6.4.2.1 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

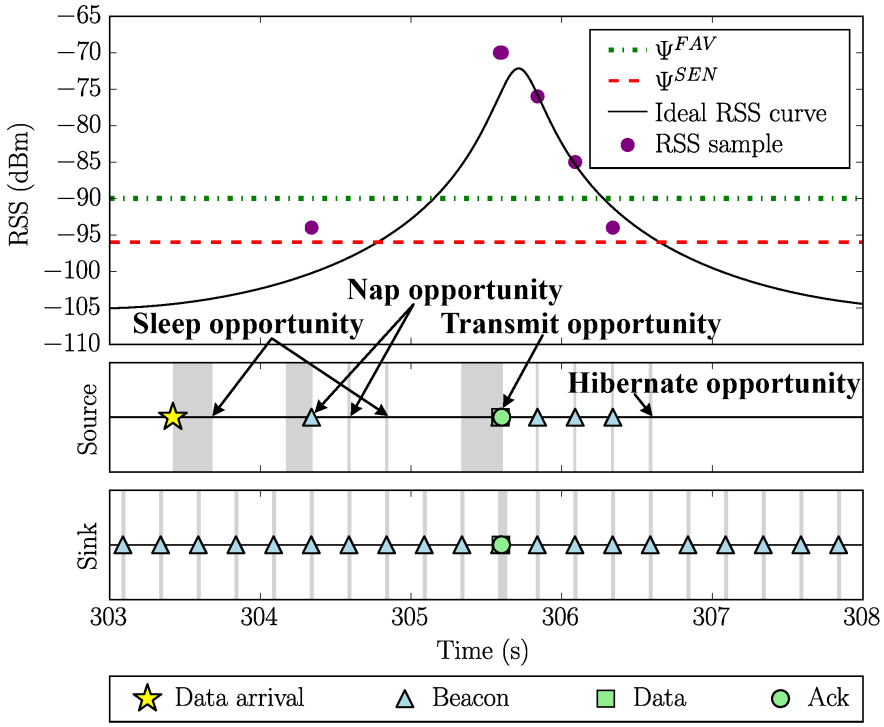


Figure 6.6: Sample trace of BladeMAC's operation, from a Cooja simulation. The *opportunities* used by BladeMAC are labeled.

does receive a data packet, it responds with an Ack packet and then listens for any additional data packets.

6.4.2.2 Beacon Interval

The choice of T_{B} is key for ensuring that both constraints in (6.1) 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 $T_{\mathrm{B}} \leq 0.5 T_{\mathrm{FAV}}$, 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 (6.1). If the source cannot sustain a small enough beacon interval, its energy budget must be increased for BladeMAC to be effective.

6.4.2.3 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. 6.5).

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.

6.4.3 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. 6.7. 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.

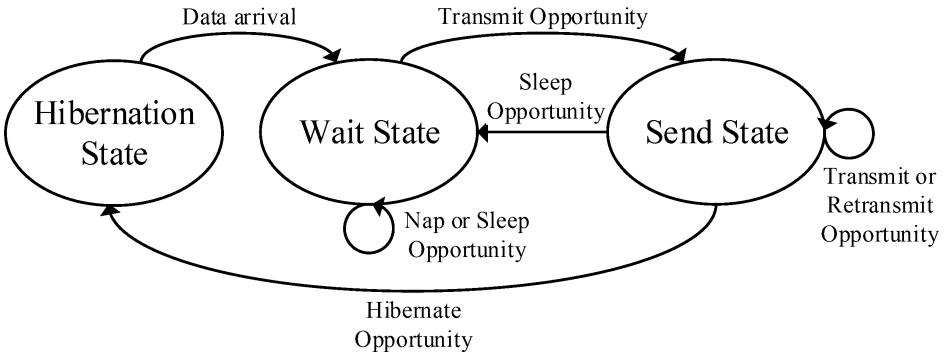


Figure 6.7: Diagram of the source's state machine in BladeMAC.

6.4.3.1 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.

6.4.3.2 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 6.1. 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*.

Transmit opportunity The source transitions into the send state. This opportunity occurs when the channel is favorable, meaning $\Psi_b \geq \Psi^{\mathrm{FAV}}$, or when the channel is degrading, meaning

Table 6.1: 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 $0.5T_{\text{SEN}}$

$\Psi_b < \Psi_{b-1}$. 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. We present the details of this

estimation process in Section 6.5. 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.

6.4.3.3 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 $R = 3$.

6.5 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 (6.1). 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.

6.5.1 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. 6.6. 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 sample are very short, so the cost is only slightly increased energy consumption, resulting in an overall system gain.

6.5.2 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 $\hat{T}_{\text{SEN}}$ to denote our estimation of the sensitivity window, and we define three different cases for our estimation method, detailed below.

6.5.2.1 Peak

This case, shown in Fig. 6.8a, 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 $2 * (t_{\text{last}} - t_{\text{next}})$ 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 $(t_{\text{last}} - t_{\text{first}})$, 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:

$$\hat{T}_{\text{SEN}} = \max\left(2 * (t_{\text{last}} - t_{\text{next}}), (t_{\text{last}} - t_{\text{first}})\right). \tag{6.2}$$

6.5.2.2 No peak

This case, shown in Fig. 6.8b, 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 $\hat{T}_{\text{SEN}} = 2 * (t_{\text{last}} - t_{\text{first}})$.

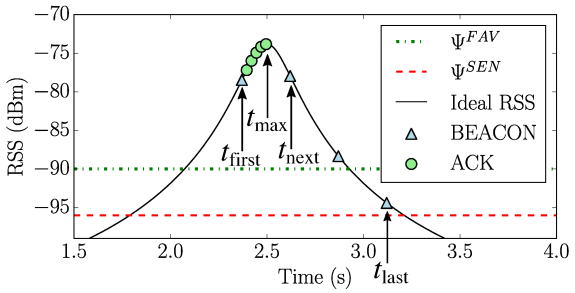
6.5.2.3 Single packet

This case, shown in Fig. 6.8c, occurs when the source has only one RSS sample. In this case, $t_{\text{last}} = t_{\text{first}}$ and no window estimation is possible, so the current estimate for T_{SEN} remains unchanged.

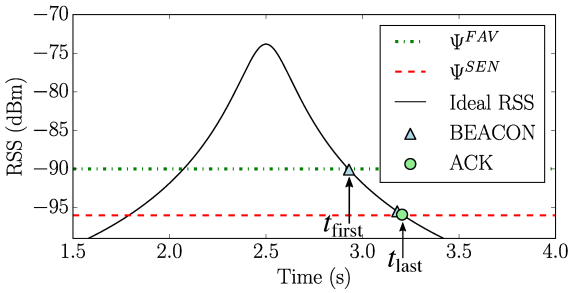
In BladeMAC, to estimate T_{SEN}, the source first identifies the case using the RSS samples collected, and then it calculates the estimate using the corresponding equation for $\hat{T}_{\text{SEN}}$.

6.5.3 Performance Evaluation

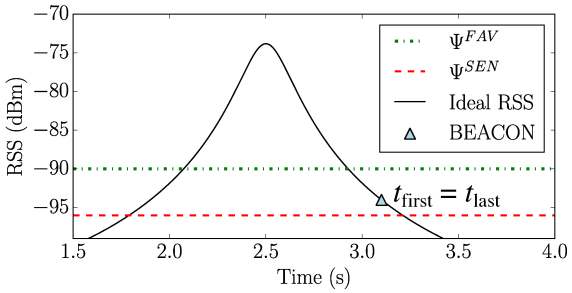
We used Python simulations to evaluate BladeMAC's T_{SEN} estimation method and the effectiveness of using extra beacons to obtain more RSS samples. We compare to a method we call MinMaxLine, which attempts to extrapolate the RSS trend, an approach that seems intuitive but which we have found to be unreliable. 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 $\hat{T}_{\text{SEN}} = 2 * (t_{\Psi}^{\text{SEN}} - t_{\text{max}})$.



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



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



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

Figure 6.8: 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.

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 $\hat{T}_{\text{SEN}} = \max\left(2 * (t_{\text{prev}} - t_{\text{first}}), (t_{\text{last}} - t_{\text{first}})\right)$, 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 [217] 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 [207]. 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 $\hat{T}_{\text{SEN}}$.

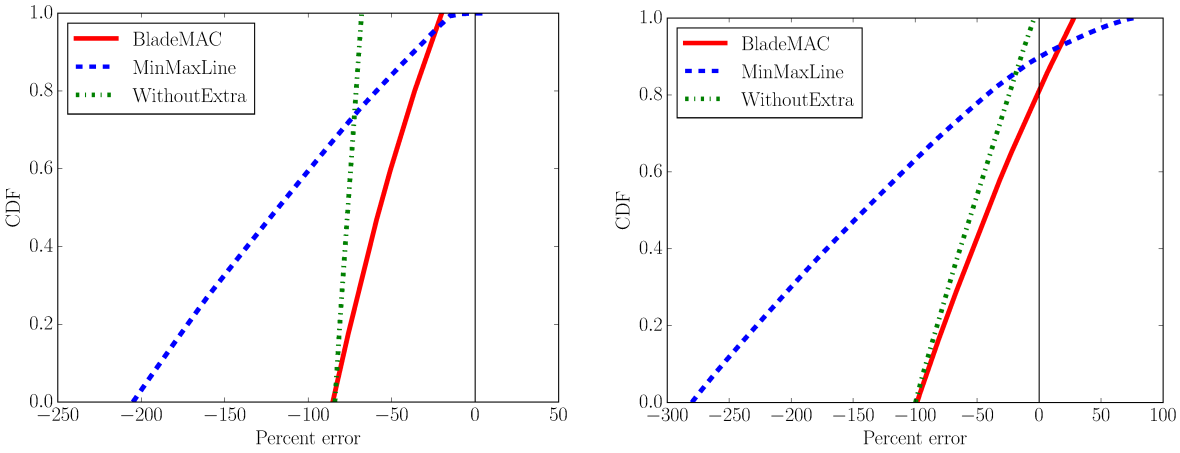
Fig. 6.9 plots CDFs (from 30,000 trials) of the percent error, which is calculated as $100 * \left(\hat{T}_{\text{SEN}} - T_{\text{SEN}}\right) / T_{\text{SEN}}$. A vertical dividing line is drawn at zero percent error. This represents the ideal curve; negative percent error (to the left of the zero 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 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.

6.5.4 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 6.6 using evaluation results.



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

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

Figure 6.9: 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.

6.6 Evaluation

In this section, we present our evaluation methods and results for BladeMAC. We use simulations in Cooja [173], Contiki OS's [20] 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.

6.6.1 Method

We implemented BladeMAC by inserting it into Contiki's Rime network stack [218], as shown in Fig. 6.10. 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 6.3, under the same framework.

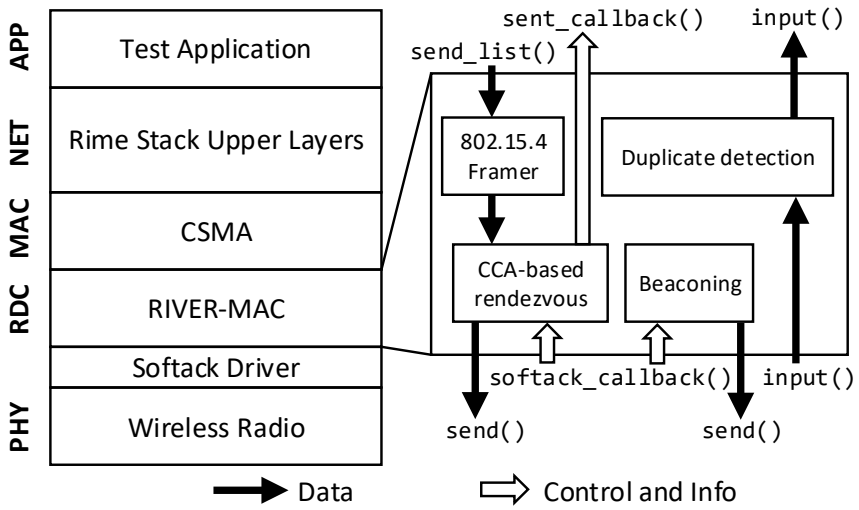


Figure 6.10: BladeMAC as it fits into the Contiki network stack.

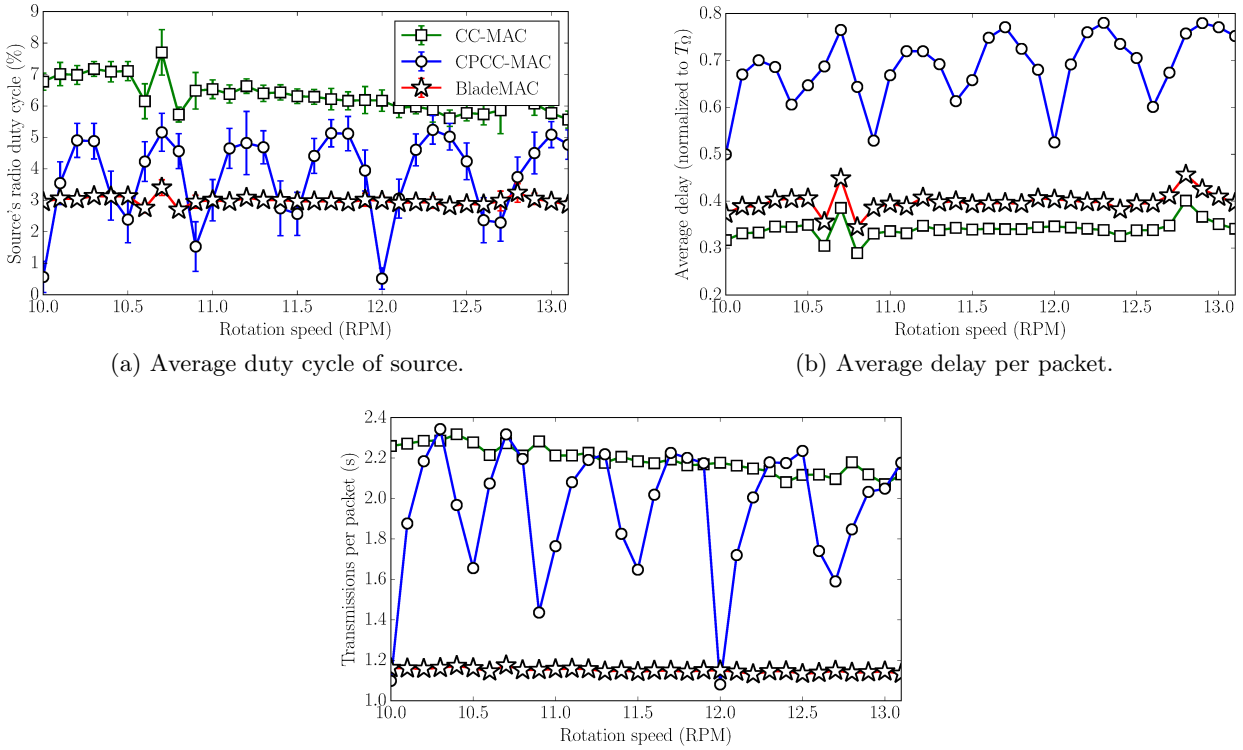
Due to the availability of information, we chose to model the National Renewable Energy Laboratory (NREL) 5 MW reference wind turbine [207]. We used Cooja, Contiki OS's simulation tool, for our evaluations. To produce the cyclical channel phenomenon in Cooja, we used the Mobility Cooja plugin [219] to simulate blade rotation with a 50 m radius. We implemented the popular lognormal shadowing propagation model [217] 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. 6.2b. We use the random component as a simple statistical method of encapsulating the Doppler effect (which is minimal in this case [217]), polarization response [213], 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 [186] to obtain the packet reception rate (PRR) from the signal-to-noise ratio (SNR).

From the TOSSIM function, we obtained a favorable threshold of $\Psi^{\mathrm{FAV}} = -90$ 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 [207]. We chose a beacon interval of $T_{\mathrm{B}} = 250$ ms for our evaluation, resulting in $T_{\mathrm{B}} \approx 0.3T_{\mathrm{FAV}}$, which satisfies our requirement that $T_{\mathrm{B}} \leq T_{\mathrm{FAV}}/2$. We also used simulations to verify this upper bound requirement for the beacon interval; as expected, a beacon interval larger than $T_{\mathrm{FAV}}/2$ 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 $T_{\mathrm{B}} \leq T_{\mathrm{FAV}}/2$, 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 6.3.1 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 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.

6.6.2 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 [207], 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. 6.11.



(a) Average duty cycle of source.

(b) Average delay per packet.

(c) Transmissions per packet.

Figure 6.11: Evaluation for different static rotation speeds, with a 28 s data arrival interval. Error bars in (a) show a 95% confidence interval.

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. 6.11a, 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).

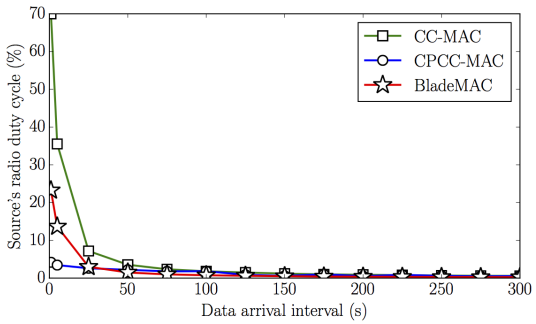
Fig. 6.11b shows delay normalized to the period of rotation. BladeMAC and CC-MAC display a constant trend with small fluctuations due to the rotation period's interaction with the data arrival interval. CC-MAC has the shortest delay because it always responds to the first possible beacon, while BladeMAC's 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. 6.11c 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.

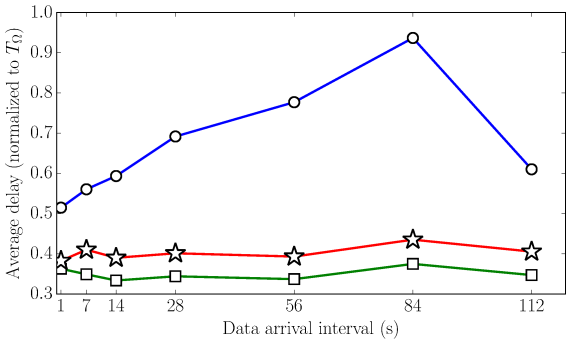
6.6.3 Data Arrival Interval

We evaluated the protocols at different data arrival intervals with a static rotation speed of 12.1 RPM. The results are plotted in Fig. 6.12. 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.

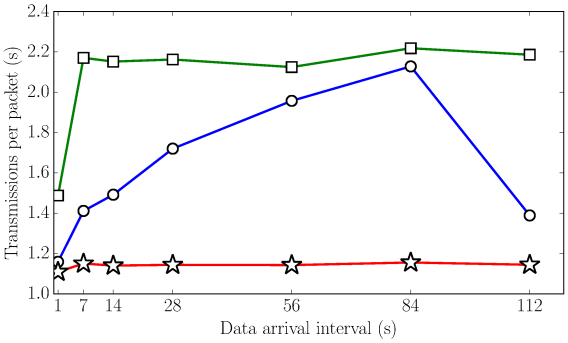
CPCC-MAC shows predictable performance at lower data arrival intervals, but above 56 s, its performance becomes unpredictable. This is due to an interaction between the rotation period, the period estimation error, and the data arrival interval. If the data arrival interval is long enough that the wakeup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



(a) Average duty cycle of source.



(b) Average delay per packet.



(c) Transmissions per packet.

Figure 6.12: 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.

6.6.4 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. 6.14. To model the dynamic rotation speed, we first used NREL's FAST wind turbine simulation software [220] to obtain traces of rotation speed for the 5 MW reference turbine. A sample trace is shown in Fig. 6.13. 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.

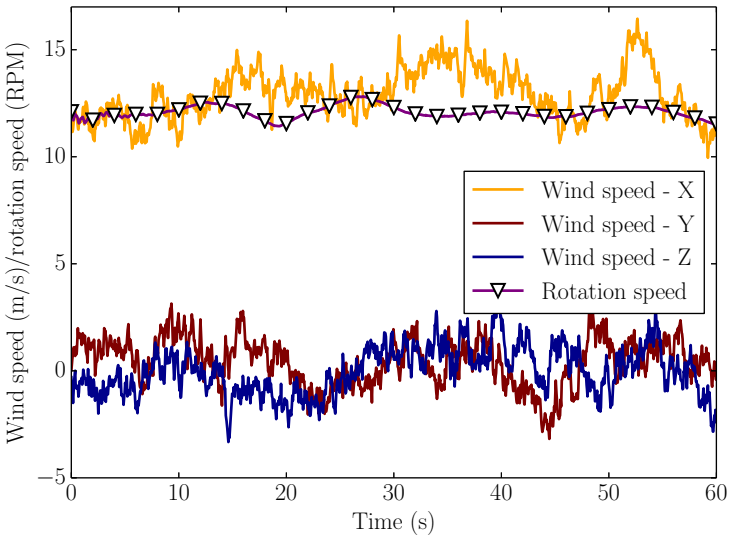
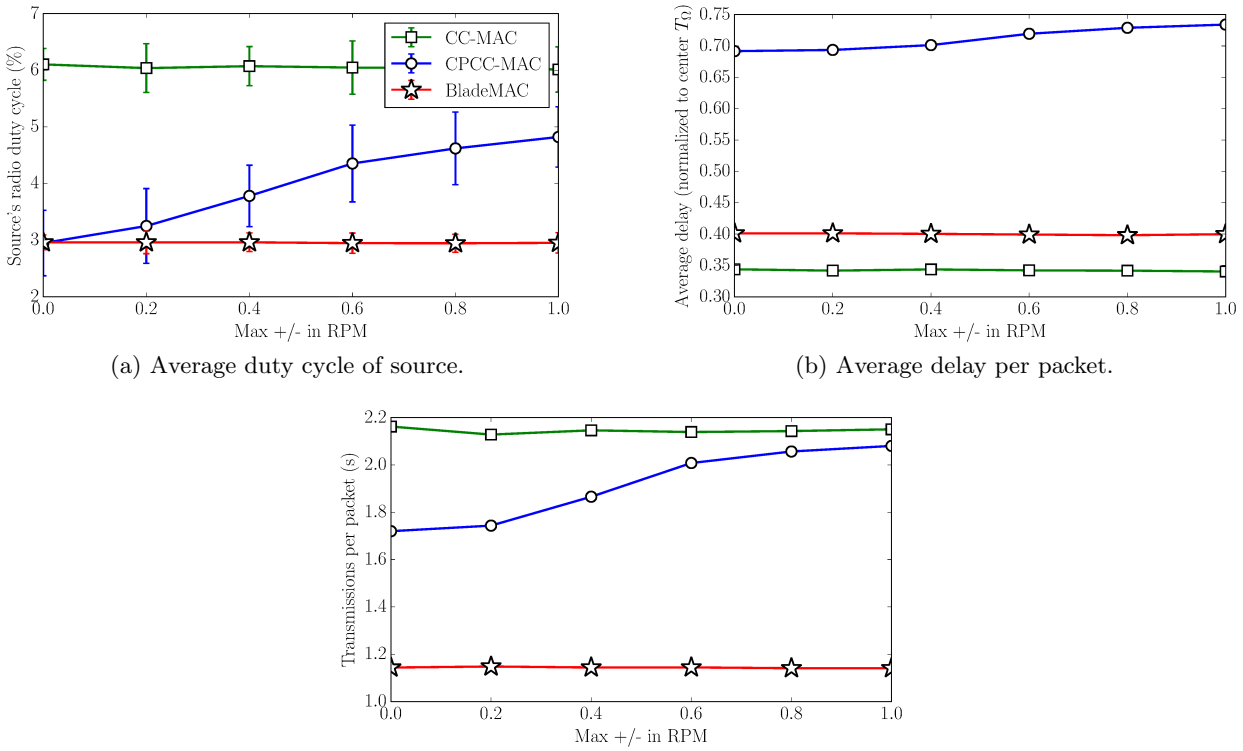


Figure 6.13: Sample trace of rotation speed from NREL's FAST wind turbine simulation software [220].

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 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. 6.15 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



(a) Average duty cycle of source.

(b) Average delay per packet.

(c) Transmissions per packet.

Figure 6.14: 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.

speed. From another perspective, we expect that BladeMAC can sustain operations with about half the energy harvesting capabilities needed by CC-MAC.

6.6.5 Trace-based Simulation

Fig. 6.16 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.

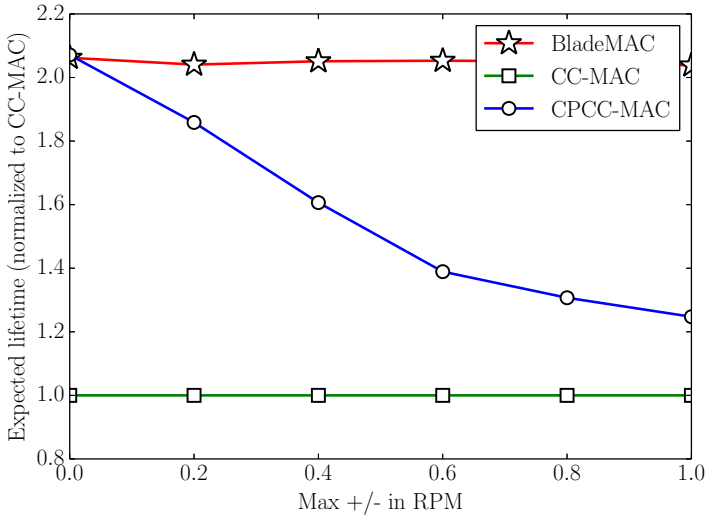


Figure 6.15: Expected source node lifetime, normalized to CC-MAC.

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.

6.6.6 Estimation Process

Finally, in Fig. 6.17, 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

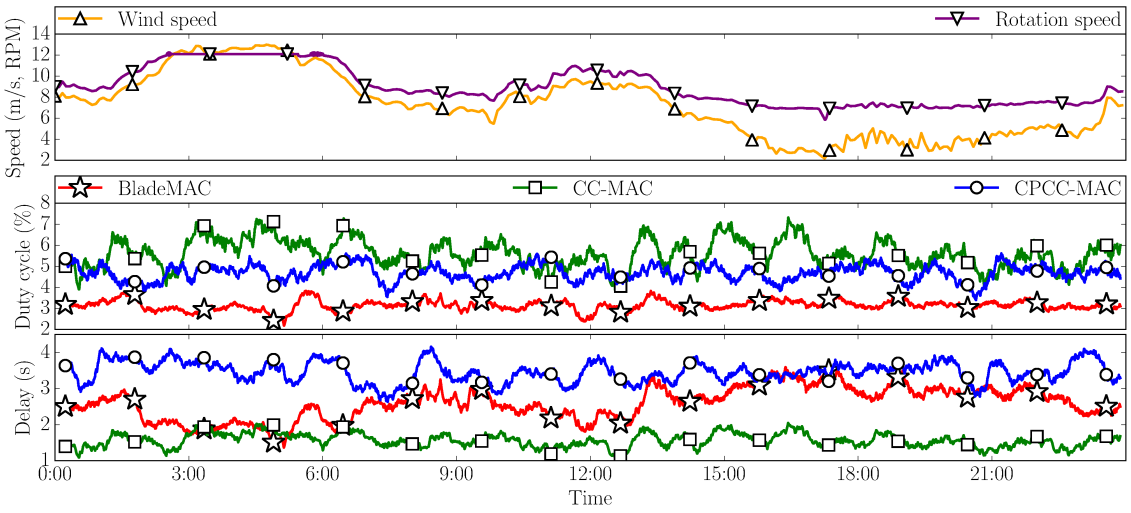


Figure 6.16: Trace of duty cycle and delay performance over a full day. Wind speeds are from real measurements at 80 m above the ground. Rotation speeds are from NREL FAST simulations. Duty cycle and delay are shown as a moving average with a 60-packet (about half an hour) window.

the energy savings of BladeMAC, a less conservative T_{SEN} estimation process could be developed and used in practice.

6.7 Discussion

6.7.1 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.

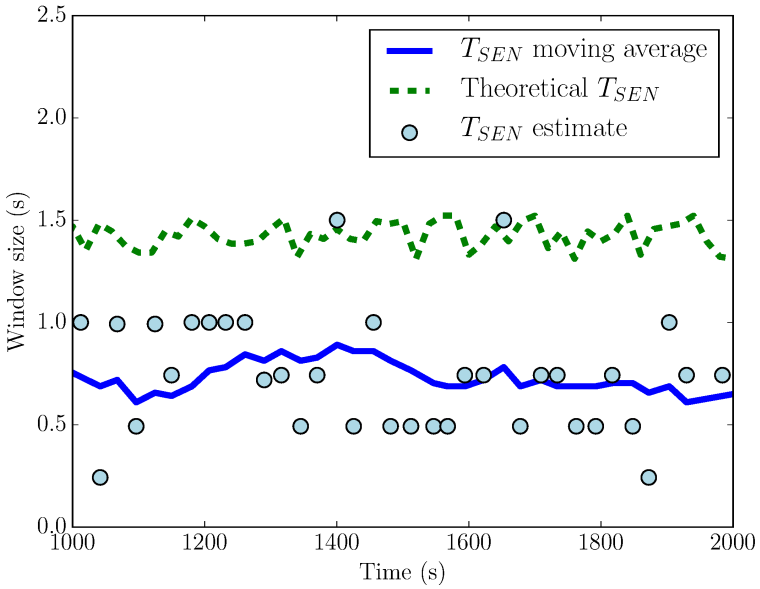


Figure 6.17: Trace of the sensitivity window estimation process of BladeMAC, with a data arrival interval of 28 s and a dynamic rotation speed of 12.1 ± 1.0 RPM.

6.7.1.1 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. [221]), strain sensors, or even a smart sensing skin [206]. Acoustic emissions monitoring of wind turbine blades with WSNs have also been proposed [205].

6.7.1.2 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 [222].

6.7.1.3 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.

6.7.1.4 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.

6.7.2 Practical Considerations for BladeMAC

6.7.2.1 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}.

6.7.2.2 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.

6.7.2.3 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.

6.7.2.4 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.

6.7.2.5 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.

6.8 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 [169] 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.

CHAPTER 7. FUTURE VISIONS

7.1 Smart Wind Turbine Vision

The family of protocols presented in this dissertation have been designed with wind energy in mind. This section describes how these protocols could, at a high level, be applied to a smart wind turbine. First, we define protocol stacks the nodes would use. Then, we describe where the nodes would be deployed.

7.1.1 Protocol Stacks

Different components of wind turbines have different requirements, deployment constraints, and potential end-user monitoring and management applications. We therefore envision our protocols being deployed differently for different components. To accomodate this, we introduce four different protocol stacks. As shown in Fig. 7.1, we define a standard node, a gateway node, a BladeMAC source, and a BladeMAC sink. We note that nodes in a smart wind turbine will also differ based on sensors installed, data generation rate, amount and type of on-board processing, and more; our goal with the protocol stacks is to present a communications framework that would transparently support these differences.

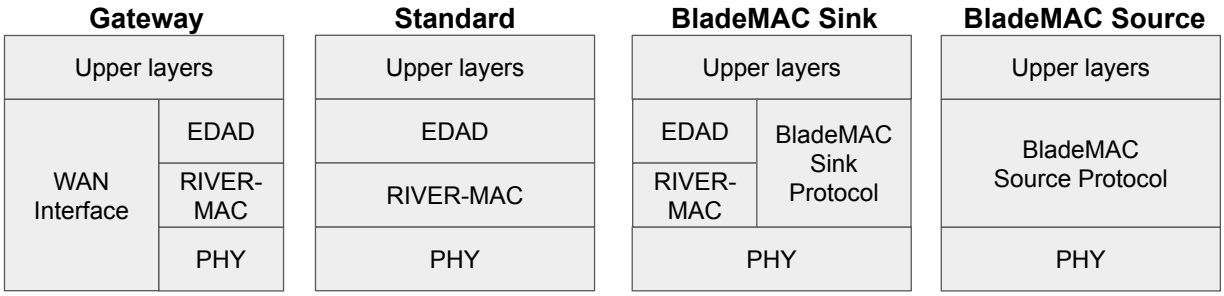


Figure 7.1: Our protocol stacks for use in a smart wind turbine.

7.1.1.1 Standard Node Stack

Our standard smart turbine node uses EDAD over RIVER-MAC for highly-efficient data collection. These nodes may also participate in opportunistic multicasting. Standard nodes may generate data, and forward data toward gateway nodes.

7.1.1.2 Gateway Node Stack

Gateway nodes are special versions of standard nodes that also include an interface to the wide-area network (WAN). The WAN interface may be wired (e.g. connected to an existing fiber line that carries SCADA data) or wireless (e.g. cellular or part of a wind farm-wide long range wireless network). Gateway nodes have a grid-connected power source and can be used as local processing centers, as well as interfaces to SCADA systems and other pre-existing monitoring systems.

7.1.1.3 BladeMAC Source Stack

BladeMAC source nodes, which are deployed on the wind turbine blades, use only the BladeMAC source protocol. These nodes are the most energy-constrained on a wind turbine, and the BladeMAC protocol was designed specifically for this scenario.

7.1.1.4 BladeMAC Sink Stack

BladeMAC sink nodes employ both the BladeMAC sink protocol and EDAD. The BladeMAC sink protocol is used to receive data from BladeMAC source nodes. EDAD over RIVER-MAC is used to forward this data toward gateway nodes. As described below, we believe these protocols may be integrated more tightly in the future.

7.1.2 Topology

In Fig. 7.2, we present a concept for the general topology of a smart wind turbine using our protocols. We additionally label the figure with some wind turbine monitoring and management applications that we believe can be enabled by this topology. The purpose is not to provide a

prescriptive vision of a smart turbine, but rather to illustrate the possibilities supported by our protocols.

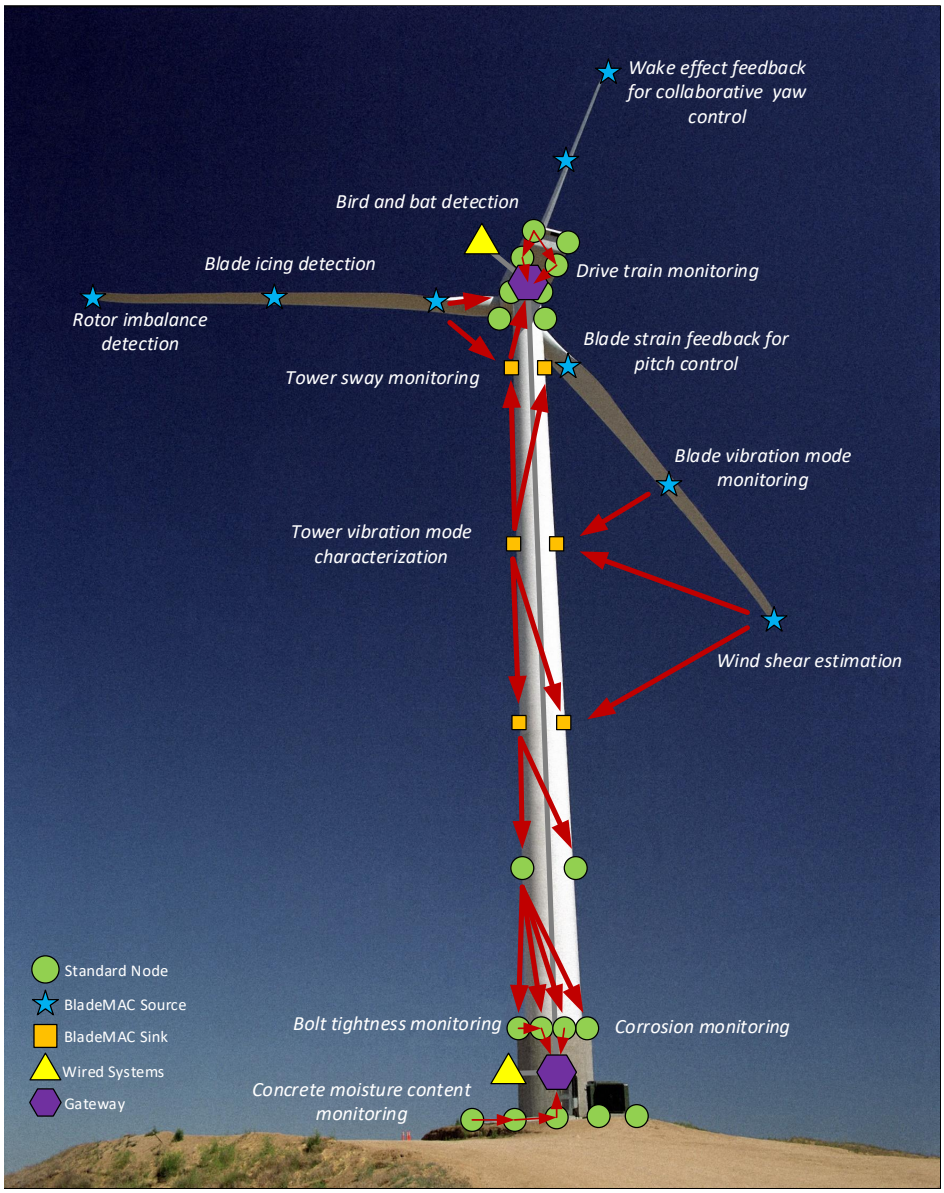


Figure 7.2: An envisioned smart wind turbine enabled by WSN technology.

Our proposed topology is based around two gateway nodes, one at the base of the tower and one in the nacelle. BladeMAC sources are deployed on the blades, and BladeMAC sinks are deployed at corresponding intervals on the tower. Multiple BladeMAC sinks are used at each height, e.g.

two on opposite sides of the tower, in order to account for yaw about the tower axis. Data packets are opportunistically sent from any node to any other node, depending on which gateway is closest.

7.1.3 Future Work

In order to complete this vision, a number of small but important extensions and integrations needs to be accomplished.

1. EDAD must be modified to work over RIVER-MAC instead of RI-MAC.
2. BladeMAC must be extended to opportunistically support multiple sinks and sources.
3. RIVER-MAC should be extended to natively support the BladeMAC sink protocol.

With these improvements in place, an extremely flexible, plug-and-play wireless monitoring architecture would be feasible.

At the application level, much work remains to be done in developing and assessing smart wind turbine applications. This includes cost-benefit analysis of the different applications and the various approaches to implementing them, including not only sensor networks, but also UAVs, non-contact optical methods, and more, as discussed in Chapter 2.

7.2 Sensor Network Vision

7.2.1 Low-energy Communication Protocols

We revisit our protocol tree from Chapter 1 in Fig. 7.3. Here, we look at the bigger picture of IoT and sensor network protocols in general. We classify protocols first by energy paradigm: those for grid-powered nodes, battery-powered nodes, or battery-less nodes.

The work presented in this dissertation falls into the battery-powered category (which may also make use of "macro" energy harvesting, i.e., relatively large-scale and reliable energy harvesting, such as a solar panel). This category encompasses the bulk of traditional WSN research and is based on existing technology. That is, the protocols presented in this dissertation could be deployed on existing hardware.

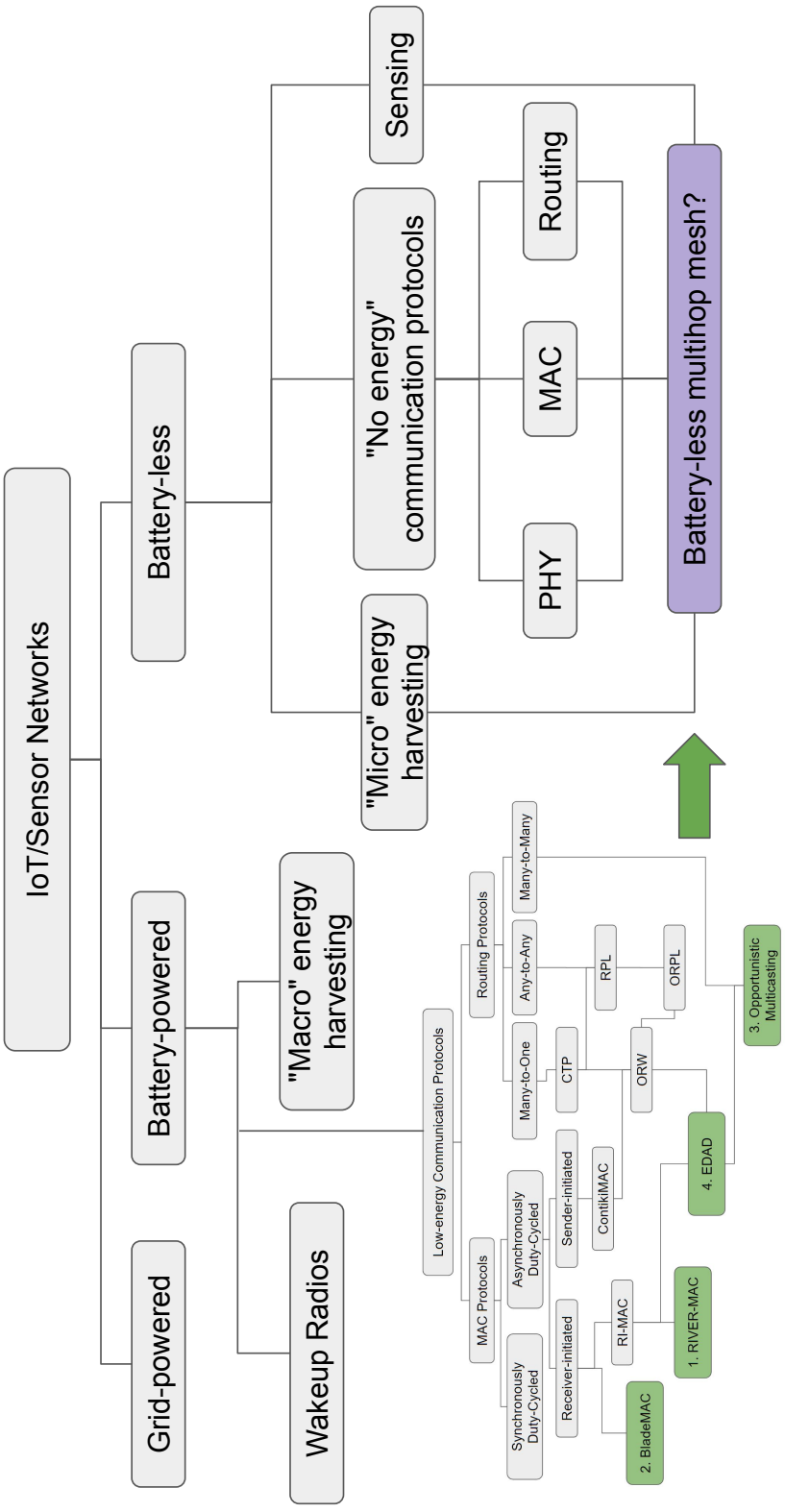


Figure 7.3: Research areas for IoT and sensor network protocols classified by energy paradigms.

In contrast, an emerging area of technology is "battery-less" IoT and sensor network nodes. These nodes have no permanent energy storage, and operate only intermittently. Examples include passive RFID tags (which require a high-powered reader) and nodes that temporarily store energy from "micro" energy harvesting (small-scale, unreliable harvesting from radio waves, vibrations, etc.) in capacitors.

This type of device presents many new challenges, including in terms of communication. Particularly, an open question is how these types of devices can participate in multihop mesh communication, which traditionally incurs overhead (e.g. regular wakeups) that is unsustainable in this paradigm. While our experience with low-energy communication protocols for sensor networks can be applied, we also anticipate the need for novel approaches that, for example, make use of all available information, both internal and external.

While the challenge of achieving battery-less multihop mesh is great, we believe the end results would be revolutionary. A battery-less multihop mesh would enable networks of tiny devices permanently embedded in materials and other inaccessible places. For example, these nodes could be mixed into concrete, allowing a bridge to "natively" monitor its own health throughout its lifetime. In short, this type of deployment would enable applications only dreamt of when WSN research began.

BIBLIOGRAPHY

[1] IPCC, "Climate change 2014: Synthesis report," Core Writing Team, R. K. Pachauri, and L. A. Meyer, Eds., Geneva, Switzerland, 2014, pp. 1–151, Contribution of Working Groups I, II and III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change.

[2] U.S. Energy Information Administration, "Levelized Cost and Levelized Avoided Cost of New Generation Resources in the Annual Energy Outlook 2017," Tech. Rep., 4 2017, accessed 21 August, 2017. [Online]. Available: https://www.eia.gov/outlooks/aeo/electricity_generation.php

[3] D. Bailey, "US approves five-year PTC phase out," *Windpower Monthly*, 12 2015, accessed 21 August, 2017. [Online]. Available: http://www.windpowermonthly.com/article/1377405/us-approves-five-year-ptc-phase

[4] T. Schlossberg, "What to Know About Trump?s Order to Dismantle the Clean Power Plan," *The New York Times*, 3 2017, accessed 21 August, 2017. [Online]. Available: https://www.nytimes.com/2017/03/27/science/what-to-know-about-trumps-order-to-dismantle-the-clean-power-plan.html

[5] M. D. Shear, "Trump Will Withdraw U.S. From Paris Climate Agreement," *The New York Times*, 6 2017, accessed 21 August, 2017. [Online]. Available: https://www.nytimes.com/2017/06/01/climate/trump-paris-climate-agreement.html

[6] 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.

[7] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. P. Vasseur, and R. Alexander. (2012, Mar.) RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. [Online]. Available: http://tools.ietf.org/html/rfc6550

[8] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X13000241

[9] H. Jo, S.-H. Sim, K. A. Mechitov, R. Kim, J. Li, P. Moinzadeh, B. F. J. Spencer, J.-W. Park, S. Cho, H.-J. Jung, C.-B. Yun, J. A. Rice, and T. Nagayama, "Hybrid wireless smart sensor network for full-scale structural health monitoring of a cable-stayed bridge," in *Proceedings of the SPIE*, M. Tomizuka, Ed., Univ. of Illinois at Urbana-Champaign, United States. SPIE, April 2011, pp. 1–15.

[10] B. Wei, M. Yang, Y. Shen, R. Rana, C. T. Chou, and W. Hu, "Real-time classification via sparse representation in acoustic sensor networks," in *ACM SenSys*, 2013.

[11] N. A. A. Aziz and K. A. Aziz, "Managing disaster with wireless sensor networks," in *13th International Conference on Advanced Communication Technology (ICACT2011)*, 2011, pp. 202–207.

[12] L. Sanchez, L. Muñoz, J. A. Galache, P. Sotres, J. R. Santana, V. Gutierrez, R. Ramdhany, A. Gluhak, S. Krco, E. Theodoridis *et al.*, "SmartSantander: IoT experimentation over a smart city testbed," *Computer Networks*, vol. 61, pp. 217–238, 2014.

[13] J. T. Adams, "An introduction to IEEE STD 802.15.4," in *IEEE Aerospace Conference*, 2007, pp. 1–8.

[14] I. Demirkol, C. Ersoy, and F. Alagoz, "MAC protocols for wireless sensor networks: a survey," *IEEE Communications Magazine*, vol. 44, no. 4, pp. 115–121, April 2006.

[15] "Using IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) in the Internet of Things (IoT): Problem statement," 2015, accessed 22 August, 2017. [Online]. Available: https://tools.ietf.org/html/rfc7554

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

[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 *ACM SenSys*, Oct. 2006.

[18] A. Dunkels, "The ContikiMAC radio duty cycling protocol," SICS, Tech. 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 *ACM SenSys*, Nov. 2008.

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

[21] E. M. Royer and C. E. Perkins, "Ad-hoc On-Demand Distance Vector Routing," in *IEEE Workshop on Mobile Computing Systems and Applications*, Feb. 1999, pp. 1–11.

[22] P. Jacquet, P. Mühlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot, "Optimized Link State Routing Protocol for Ad Hoc Networks," in *IEEE INMIC*, Dec. 2001, pp. 1–7.

[23] R. Fonseca, O. Gnawali, K. Jamieson, S. Kim, P. Levis, and A. Woo. The Collection Tree Protocol (CTP). [Online]. Available: http://www.tinyos.net/tinyos-2.x/doc/html/tep123.html

[24] V. Raghunathan, S. Ganeriwal, and M. Srivastava, "Emerging techniques for long lived wireless sensor networks," *IEEE Communications Magazine*, vol. 44, no. 4, pp. 108–114, April 2006.

[25] M. L. Wymore, Y. Peng, X. Zhang, and D. Qiao, "EDAD: energy-centric data collection with anycast in duty-cycled wireless sensor networks," in *Proc. IEEE WCNC*, 2015.

[26] M. L. Wymore and D. Qiao, "Opportunistic many-to-many multicasting in duty-cycled wireless sensor networks," in *2018 IEEE International Conference on Communications (ICC)*, May 2018, pp. 1–7.

[27] M. L. Wymore and D. Qiao, "BladeMAC: Radio duty-cycling in a dynamic, cyclical channel," in *Proc. IEEE ICC*, 2017.

[28] M. L. Wymore and D. Qiao, "An opportunistic mac protocol for energy-efficient wireless communication in a dynamic, cyclical channel," *IEEE Transactions on Green Communications and Networking*, vol. 2, no. 2, pp. 533–544, June 2018.

[29] "Global Wind Statistics 2014," Global Wind Energy Council, Tech. Rep., February 2015.

[30] "Iowa Wind Energy," American Wind Energy Association, Tech. Rep., 2015. [Online]. Available: http://awea.files.cms-plus.com/FileDownloads/pdfs/Iowa.pdf

[31] "Corporate Responsibility Report 2013: Towards a Sustainable Energy Future," Red Eléctrica Corporación, Tech. Rep., May 2014.

[32] "The European offshore wind industry - key trends and statistics 2014," European Wind Energy Association, Tech. Rep., January 2015.

[33] "AWEA State RPS Market Assessment 2013," American Wind Energy Association, Tech. Rep., September 2013.

[34] H. F. Zhou, H. Y. Dou, L. Z. Qin, Y. Chen, Y. Q. Ni, and J. M. Ko, "A review of full-scale structural testing of wind turbine blades," *Renewable and Sustainable Energy Reviews*, vol. 33, no. c, pp. 177–187, May 2014.

[35] P. Tchakoua, R. Wamkeue, M. Ouhrouche, F. Slaoui-Hasnaoui, T. A. Tameghe, and G. Ekemb, "Wind turbine condition monitoring: State-of-the-art review, new trends, and future challenges," *Energies*, vol. 7, no. 4, pp. 2595–2630, April 2014.

[36] B. Yang and D. Sun, "Testing, inspecting and monitoring technologies for wind turbine blades: A survey," *Renewable and Sustainable Energy Reviews*, vol. 22, no. C, pp. 515–526, June 2013.

[37] Y. Amirat, M. E. H. Benbouzid, E. Al Ahmar, B. Bensaker, and S. Turri, "A brief status on condition monitoring and fault diagnosis in wind energy conversion systems," *Renewable and Sustainable Energy Reviews*, vol. 13, no. 9, pp. 2629–2636, December 2009.

[38] Z. Hameed, Y. S. Hong, Y. M. Cho, S. H. Ahn, and C. K. Song, "Condition monitoring and fault detection of wind turbines and related algorithms: A review," *Renewable and Sustainable Energy Reviews*, vol. 13, no. 1, pp. 1–39, January 2009.

[39] C. C. Ciang, J.-R. Lee, and H.-j. Bang, "Structural health monitoring for a wind turbine system: a review of damage detection methods," *Measurement Science and Technology*, vol. 19, no. 1, p. 2001, December 2008.

[40] C. J. Crabtree, D. Zappalá, and P. J. Tavner, "Survey of commercially available condition monitoring systems for wind turbines." Durham University School of Engineering and Computing Sciences and the SUPERGEN Wind Energy Technologies Consortium, Tech. Rep., 2014.

[41] B. Chen, D. Zappalá, C. J. Crabtree, and P. J. Tavner, "Survey of commercially available SCADA data analysis tools for wind turbine health monitoring," Durham University School of Engineering and Computing Sciences, Tech. Rep., 2014.

[42] C. R. Farrar and K. Worden, "An introduction to structural health monitoring," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 365, no. 1851, pp. 303–315, February 2007.

[43] R. B. Randall, "State of the art in monitoring rotating machinery-part 1," *Sound and Vibration*, March 2004.

[44] R. K. Mobley, *An Introduction to Predictive Maintenance*. Butterworth-Heinemann, October 2002.

[45] "Used Oil Analysis: On-Line Sensors," 2014, accessed 9 June, 2014. [Online]. Available: http://www.kittiwake.com/online-sensors

[46] A. Hamilton and F. Quail, "Detailed state of the art review for the different online/inline oil analysis techniques in context of wind turbine gearboxes," *Journal of Tribology*, vol. 133, no. 4, pp. 1–18, 2011.

[47] D. Mba and R. Rao, "Development of acoustic emission technology for condition monitoring and diagnosis of rotating machines; Bearings, pumps, gearboxes, engines and rotating structures," *The Shock and Vibration Digest*, vol. 38, no. 1, pp. 3–16, 2006.

[48] G. Byrne, D. Dornfeld, I. Inasaki, G. Ketteler, W. König, and R. Teti, "Tool condition monitoring (TCM) — The status of research and industrial application," *CIRP Annals - Manufacturing Technology*, vol. 44, no. 2, pp. 541–567, January 1995.

[49] T. Cargol, "An overview of online oil monitoring technologies," in *Proceedings of the Fourth Annual Weidmann-ACTI Technical Conference*, 2005, pp. 1–6.

[50] A. Rytter, *Vibrational Based Inspection of Civil Engineering Structures*, ser. Fracture and Dynamics. Dept. of Building Technology and Structural Engineering, Aalborg University, 1993, Ph.D. thesis defended publicly at the University of Aalborg, April 20, 1993. 206 pp.

[51] K. Worden, C. R. Farrar, G. Manson, and G. Park, "The fundamental axioms of structural health monitoring," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 463, no. 2082, pp. 1639–1664, April 2007.

[52] R. D. Adams, P. Cawley, C. J. Pye, and B. J. Stone, "A vibration technique for non-destructively assessing the integrity of structures," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 20, no. 2, pp. 93–100, April 1978.

[53] C. R. Farrar, S. W. Doebling, and D. A. Nix, "Vibration-based structural damage identification," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 359, no. 1778, pp. 131–149, January 2001.

[54] E. P. Carden and P. Fanning, "Vibration based condition monitoring: A review," *Structural health monitoring*, vol. 3, no. 4, pp. 355–377, December 2004.

[55] E. Astreinidis and D. Egglezos, "Instrumented strain monitoring of the Acropolis Wall with optical fibre sensors: Comparison of the measurements to the analytical predictions," in *3rd Panhellenic Congress of Earthquake Engineering and Seismology*, Athens, Greece, November 2008, pp. 1–3.

[56] S. DelloRusso, G. Juneja, B. Gabby, and D. Dusenberry, "Monitoring and repair of the Milwaukee City Hall masonry tower," *Journal of Performance of Constructed Facilities*, vol. 22, no. 4, pp. 197–206, August 2008.

[57] J. P. Lynch, "An overview of wireless structural health monitoring for civil structures," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 365, no. 1851, pp. 345–372, February 2007.

[58] F. Federici, F. Graziosi, M. Faccio, A. Colarieti, V. Gattulli, M. Lepidi, and F. Potenza, "An Integrated Approach to the Design of Wireless Sensor Networks for Structural Health Monitoring," *International Journal of Distributed Sensor Networks*, vol. 2012, pp. 1–16, 2012.

[59] A. Albarbar, S. Mekid, A. Starr, and R. Pietruszkiewicz, "Suitability of MEMS accelerometers for condition monitoring: An experimental study," *Sensors*, vol. 8, no. 2, pp. 1–16, February 2008.

[60] "Parthenon west facade pillars instrumentation," 2014, accessed 22 June, 2014. [Online]. Available: http://www.crd.gr/en/projects/instrumentation-projects/parthenon-west-facade

[61] J. M. W. Brownjohn, "Structural health monitoring of civil infrastructure," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 365, no. 1851, pp. 589–622, February 2007.

[62] S. Jang, H. Jo, S. Cho, K. Mechitov, and J. A. Rice, "Structural health monitoring of a cable-stayed bridge using smart sensor technology: deployment and evaluation," *Smart Structures and Systems*, vol. 6, no. 5–6, pp. 439–459, 2010.

[63] S. Tegen, E. Lantz, M. Hand, B. Maples, A. Smith, and P. Schwabe, "2011 Cost of Wind Energy Review," National Renewable Energy Laboratory, Tech. Rep., March 2013.

[64] B. Maples, G. Saur, M. Hand, R. van de Pieterman, and T. Obdam, "Installation, Operation, and Maintenance Strategies to Reduce the Cost of Offshore Wind Energy," National Renewable Energy Laboratory, Tech. Rep., July 2013.

[65] R. Ahmad and S. Kamaruddin, "An overview of time-based and condition-based maintenance in industrial application," *Computers & Industrial Engineering*, vol. 63, no. 1, pp. 135–149, August 2012.

[66] A. K. S. Jardine, D. Lin, and D. Banjevic, "A review on machinery diagnostics and prognostics implementing condition-based maintenance," *Mechanical Systems and Signal Processing*, vol. 20, no. 7, pp. 1483–1510, October 2006.

[67] "Summary of Wind Turbine Accident data to 31 December 2013," Caithness Windfarm Information Forum, Tech. Rep., January 2014. [Online]. Available: www.caithnesswindfarms.co.uk

[68] "Top 5 Wind Energy Claims," GCube Insurance, Tech. Rep., July 2013.

[69] C. A. Walford, "Wind turbine reliability: Understanding and minimizing wind turbine operation and maintenance costs," Sandia National Laboratories, Tech. Rep., March 2006.

[70] J. Nilsson and L. Bertling, "Maintenance management of wind power systems using condition monitoring systems–Life cycle cost analysis for two case studies," *IEEE Transactions on Energy Conversion*, vol. 22, no. 1, pp. 223–229, March 2007.

[71] F. Besnard and L. Bertling, "An approach for condition-based maintenance optimization applied to wind turbine blades," *IEEE Transactions on Sustainable Energy*, vol. 1, no. 2, pp. 77–83, July 2010.

[72] J. Van Dam and L. J. Bond, "Economics of online structural health monitoring of wind turbines: Cost benefit analysis," in *AIP Conference Proceedings 1650*. AIP Publishing LLC, May 2015, pp. 899–908.

[73] "SmartScan: Optical blade loads monitoring for wind turbines," Smart Fibres, Tech. Rep., December 2010.

[74] L. W. M. M. Rademakers and T. W. Verbruggen, "Fibre optic blade load monitoring (FOBM)," Energy Research Centre of the Netherlands, Tech. Rep., March 2011.

[75] L. Rademakers, "Fibre optic load monitoring of wind turbines," Energy Research Centre of the Netherlands, Tech. Rep., November 2012.

[76] M. C. Homola, P. J. Nicklasson, and P. A. Sundsbø, "Ice sensors for wind turbines," *Cold Regions Science and Technology*, vol. 46, no. 2, pp. 125–131, November 2006.

[77] R. J. Barthelmie, S. T. Frandsen, M. N. Nielsen, S. C. Pryor, P. E. Rethore, and H. E. Jørgensen, "Modelling and measurements of power losses and turbulence intensity in wind turbine wakes at Middelgrunden offshore wind farm," *Wind Energy*, vol. 10, no. 6, pp. 517–528, 2007.

[78] R. May, O. Reitan, K. Bevanger, S. H. Lorentsen, and T. Nygård, "Mitigating wind-turbine induced avian mortality: Sensory, aerodynamic and cognitive constraints and options," *Renewable and Sustainable Energy Reviews*, vol. 42, no. C, pp. 170–181, February 2015.

[79] R. M. Osgood, "Dynamic Characterization Testing of Wind Turbines," National Renewable Energy Laboratory, Tech. Rep. NREL/TP-500-30070, May 2001.

[80] J. Watts, "Winds of change blow through China as spending on renewable energy soars," March 2012, accessed 14 May, 2015. [Online]. Available: http://www.theguardian.com/world/2012/mar/19/china-windfarms-renewable-energy

[81] I. Staffell and R. Green, "How does wind farm performance decline with age?" *Renewable Energy*, vol. 66, no. C, pp. 775–786, Jun. 2014.

[82] H. Link, J. Keller, Y. Guo, and B. McNiff, "Gearbox Reliability Collaborative Phase 3 Gearbox 2 Test Plan," National Renewable Energy Laboratory, Tech. Rep., April 2013.

[83] E. Bechhoefer, "Predicting gearbox health," *Wind Systems*, vol. 3, no. 31, pp. 1–4, March 2012.

[84] X. Gong and W. Qiao, "Bearing fault diagnosis for direct-drive wind turbines via current-demodulated signals," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 8, pp. 3419–3428, April 2013.

[85] S. J. Watson, B. J. Xiang, W. Yang, P. J. Tavner, and C. J. Crabtree, "Condition monitoring of the power output of wind turbine generators using wavelets," *IEEE Transactions on Energy Conversion*, vol. 25, no. 3, pp. 715–721, September 2010.

[86] Y. Amirat, V. Choqueuse, and M. Benbouzid, "Condition monitoring of wind turbines based on amplitude demodulation," in *IEEE Energy Conversion Congress and Exposition*. IEEE, September 2010, pp. 2417–2421.

[87] W. Yang, R. Court, and J. Jiang, "Wind turbine condition monitoring by the approach of SCADA data analysis," *Renewable Energy*, vol. 53, no. C, pp. 365–376, May 2013.

[88] B. R. Wiesent, D. G. Dorigo, M. Schardt, and A. W. Koch, "Gear oil condition monitoring for offshore wind turbines," February 2012, accessed 15 May, 2015. [Online]. Available: http://www.machinerylubrication.com/Read/28782/

[89] "Bently Nevada ADAPT Wind Condition Monitoring Solution," GE, Tech. Rep., August 2013. [Online]. Available: http://www.ge-mcs.com/download/monitoring/GEA18079B_AdaptWindBro_r3.pdf

[90] "Siemens G2 platform - 2.3-MW geared wind turbines," Siemens, Tech. Rep., February 2014. [Online]. Available: http://www.energy.siemens.com/us/pool/hq/power-generation/renewables/wind-power/platform%20brochures/G2-Onshore-Platform-brochures-English-Feb2014-WEB.pdf

[91] "Wind Turbine Technology - Key Technologies," 2014. [Online]. Available: http://www.energy.siemens.com/us/en/renewable-energy/wind-power/wind-turbine-technology/key-technologies.htm

[92] "TurbinePhD," Renewable NRG Systems, Tech. Rep., September 2013. [Online]. Available: http://www.renewablenrgsystems.com/FileLibrary/44e45948f45e4f7097da4207a8af4b83/TurbinePhD_rNRG_Web.pdf

[93] "Make the most out of your maintenance resources with SKF WindCon online condition monitoring system," SKF, Tech. Rep., November 2013. [Online]. Available: http://www.skf.com/binary/21-143837/SKF-Wind-Con-Bro-Update_6Nov13.pdf

[94] S. Sheng and P. S. Veers, "Wind turbine drivetrain condition monitoring—an overview," in *Mechanical Failures Prevention Group Applies Systems Health Management Conference*. Virginia Beach, VA: National Renewable Energy Laboratory, May 2011, pp. 1–19.

[95] H. Cassar, "Wireless condition monitoring," in *Intelligent Energy Conference and Exhibition*, BP. Amsterdam, The Netherlands: Society of Petroleum Engineers, 2006, pp. 1–11.

[96] S. Thanagasundram and F. Schlindwein, "Comparison of integrated micro-electrical-mechanical system and piezoelectric accelerometers for machine condition monitoring," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 220, no. 8, pp. 1135–1146, January 2006.

[97] H. Ceylan, K. Gopalakrishnan, S. Kim, P. C. Taylor, M. Prokudin, and A. F. Buss, "Highway infrastructure health monitoring using micro-electromechanical sensors and systems (MEMS)," *Journal of Civil Engineering and Management*, vol. 19, no. Sup1, pp. 188–201, 2013.

[98] A. Norris, M. Saafi, and P. Romine, "Temperature and moisture monitoring in concrete structures using embedded nanotechnology/microelectromechanical systems (MEMS) sensors," *Construction and Building Materials*, vol. 22, no. 2, pp. 111–120, February 2008.

[99] P. Sparrevik, "Monitoring offshore wind turbine foundations," in *Oceanology International*, London, England, March 2014, pp. 1–26.

[100] "Structural monitoring for offshore wind turbine foundations," Strainstall Monitoring, Tech. Rep., 2014.

[101] P. Faulkner, P. Cutter, and A. Owens, "Structural health monitoring systems in difficult environments—offshore wind turbines," *6th European Workshop on Structural Health Monitoring*, pp. 1–7, 2012.

[102] G. De Sitter, C. Devriendt, and P. J. Jordaens, "Dynamic monitoring of offshore wind farms," in *Study Day on Reliability and Efficiency in Renewable Energy Sources*, Bruges, Belgium, February 2013, pp. 1–33.

[103] "Continuous corrosion monitoring of monopile foundation structures," in *EWEA Annual Event*, Vienna, Austria, February 2013, pp. 1–17.

[104] L. B. Ibsen and R. Brincker, "Design of a new foundation for offshore wind turbines," in *Proceedings of The 22nd International Modal Analysis Conference (IMAC)*, Detroit, Michigan, 2004, pp. 1–8.

[105] J. Wernicke, S. Kuhnt, and R. Byars, "Structural monitoring system for offshore wind turbine foundation structures," in *European Wind Energy Conference Exhibition*, Athens, Greece, March 2006, pp. 1–7.

[106] "Wind turbine scour monitoring," 2014, accessed 28 June, 2014. [Online]. Available: http://www.strainstall.com/what-we-do/our-systems/structural-monitoring/wind-turbine-monitoring/wind-turbine-scour-monitoring/

[107] P. Michalis, M. Saafi, and M. Judd, "Capacitive sensors for offshore scour monitoring," *Proceedings of the ICE - Energy*, vol. 166, no. 4, pp. 189–197, November 2013.

[108] J. Yang, "Ming Yang turbine collapse kills one, injures three," September 2012, accessed 15 May, 2015. [Online]. Available: http://www.windpowermonthly.com/article/1149772/ming-yang-turbine-collapse-kills-one-injures-three

[109] "Nearly all 22 turbine foundations cracked," February 2011. [Online]. Available: http://ontario-wind-resistance.org/2011/02/03/nearly-all-22-turbine-foundations-cracked/

[110] J. B. Hassine, "Foundation condition monitoring," in *Wind Turbine Condition Monitoring Workshop*. RES Americas, October 2011, pp. 1–24.

[111] "Project Profile - Wind Turbine Foundation Monitoring," Canary Systems, Tech. Rep., October 2012.

[112] M. Currie, M. Saafi, C. Tachtatzis, and F. Quail, "Structural health monitoring for wind turbine foundations," *Proceedings of the ICE - Energy*, vol. 166, no. 4, pp. 162–169, November 2013.

[113] Y. Schiegg and L. Steiner, "Cost effectiveness and application of online monitoring in reinforced concrete structures," *Materials and Corrosion*, vol. 61, no. 6, pp. 490–493, June 2010.

[114] "Prediction of creep, shrinkage, and temperature effects in concrete structures," ACI Committee 209, Tech. Rep., 1982.

[115] "intelliRock System Overview," Engius, LLC, Tech. Rep., October 2005.

[116] K. L. Rens, T. J. Wipf, and F. W. Klaiber, "Review of nondestructive evaluation techniques of civil infrastructure," *Journal of Performance of Constructed Facilities*, vol. 11, no. 4, pp. 152–160, November 1997.

[117] T. Kishi, M. Ohtsu, and S. Yuyama, *Acoustic Emission—Beyond the Millennium*. Elsevier, 2000.

[118] A. Nair and C. S. Cai, "Acoustic emission monitoring of bridges: Review and case studies," *Engineering Structures*, vol. 32, no. 6, pp. 1704–1714, June 2010.

[119] C. U. Grosse and M. Krûger, "Bridge monitoring using wireless sensors and acoustic emission techniques," in *Proceedings, Inaugural International Conference of the ASCE Engineering Mechanics Institute*, 2008, pp. 1–7.

[120] L. Bertolini, B. Elsener, P. Pedeferri, E. Redaelli, and R. B. Polder, *Corrosion of Steel in Concrete*, ser. Prevention, Diagnosis, Repair. John Wiley & Sons, February 2013.

[121] R. B. Polder, W. Peelen, O. Klinghoffer, J. Eri, and J. Leggedoor, "Use of advanced corrosion monitoring for risk based management of concrete structures," *HERON*, vol. 52, no. 4, pp. 239–250, 2007.

[122] I. F. Akyildiz, D. Pompili, and T. Melodia, "Underwater acoustic sensor networks: research challenges," *Ad Hoc Networks*, vol. 3, no. 3, pp. 257–279, May 2005.

[123] C. D. Taylor, S. J. Gutierrez, S. L. Langdon, K. L. Murphy, and W. A. Walton III, "Measurement of RF propagation into concrete structures over the frequency range 100 MHZ to 3 GHz," in *Wireless Personal Communications*. Boston, MA: Springer US, January 1997, pp. 131–144.

[124] J. B. Ong, Z. You, J. Mills-Beale, E. L. Tan, B. D. Pereles, and K. G. Ong, "A Wireless, Passive Embedded Sensor for Real-Time Monitoring of Water Content in Civil Engineering Materials," *IEEE Sensors Journal*, vol. 8, no. 12, pp. 2053–2058, December 2008.

[125] K. Ghee Ong and C. A. Grimes, "A resonant printed-circuit sensor for remote query monitoring of environmental parameters," *Smart Materials and Structures*, vol. 9, no. 4, pp. 421–428, August 2000.

[126] W. Hansen and S. Surlaker, "Embedded wireless temperature monitoring systems for concrete quality control," University of Michigan, Ann Arbor, MI, Tech. Rep., June 2006.

[127] "Steel solutions in the green economy," World Steel Association, Tech. Rep., April 2012.

[128] M. Krapfl, "Energy Department supports Iowa State studies of concrete for taller wind turbine towers," September 2014, accessed 12 May, 2015. [Online]. Available: http://www.news.iastate.edu/news/2014/09/18/turbinetowers

[129] "Smarter and More Powerful: Introducing GE's 2.75-120," General Electric, Tech. Rep., April 2014.

[130] R. Davidson, "GE develops space frame tower," March 2014, accessed 24 June, 2014. [Online]. Available: http://www.windpowermonthly.com/article/1283941/ge-develops-space-frame-tower

[131] 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, Feb. 2012.

[132] K. Smarsly, D. Hartmann, and K. H. Law, "An integrated monitoring system for life-cycle management of wind turbines," *Smart Structures and Systems*, vol. 12, no. 2, pp. 209–233, January 2013.

[133] D. Hartmann, K. Smarsly, and K. H. Law, "Coupling sensor-based structural health monitoring with finite element model updating for probabilistic lifetime estimation of wind energy converter structures," in *The 8th International Workshop on Structural Health Monitoring 2011*, Stanford, CA, September 2011, pp. 1–8.

[134] K. Smarsly, K. H. Law, and D. Hartmann, "Structural health monitoring of wind turbines observed by autonomous software components – 2nd level monitoring," in *14th International Conference on Computing in Civil and Building Engineering*, Moscow, Russia, June 2012, pp. 1–8.

[135] R. Rolfes, S. Zerbst, G. Haake, J. Reetz, and J. P. Lynch, "Integral SHM-system for offshore wind turbines using smart wireless sensors," in *6th International Workshop on Structural Health Monitoring*, Stanford, CA, September 2007, pp. 1–8.

[136] R. A. Swartz, J. P. Lynch, B. Sweetman, R. Rolfes, and S. Zerbst, "Structural monitoring of wind turbines using wireless sensor networks," in *ESF-NSF Workshop on Sensor Networks for Civil Infrastructure Systems*, Cambridge, UK, April 2008, pp. 1–8.

[137] C. Devriendt, W. Weijtjens, M. El-Kafafy, and G. De Sitter, "Monitoring resonant frequencies and damping values of an offshore wind turbine in parked conditions," *IET Renewable Power Generation*, vol. 8, no. 4, pp. 433–441, May 2014.

[138] H.-j. Bang, S.-w. Ko, M.-s. Jang, and H.-i. Kim, "Shape estimation and health monitoring of wind turbine tower using a FBG sensor array," in *Instrumentation and Measurement Technology Conference (I2MTC), 2012 IEEE International*, 2012, pp. 496–500.

[139] P. Guo and D. Infield, "Wind turbine tower vibration modeling and monitoring by the Nonlinear State Estimation Technique (NSET)," *Energies*, vol. 5, no. 12, pp. 5279–5293, December 2012.

[140] A. J. Chen and G. J. He, "Wind-induced vibration analysis and remote monitoring test of wind turbine power tower," *Advanced Materials Research*, vol. 639-640, pp. 293–296, January 2013.

[141] P. Mitchell, A. Janssen, and B. Partov Poor, "Development of a high-speed wideband wavelength tunable infra-red laser source for real time wind turbine array sensing applications," *Integrated Photonics: Materials*, vol. 8069, pp. 1–12, May 2011.

[142] M. Benedetti, V. Fontanari, and D. Zonta, "Structural health monitoring of wind towers: remote damage detection using strain sensors," *Smart Materials and Structures*, vol. 20, no. 5, p. 055009, April 2011.

[143] O. Guillermin, "Developing composite wind blades that will stand the test of time," January 2011, accessed 30 June 2014. [Online]. Available: http://www.designnews.com/document.asp?doc_id=230009

[144] B. Lu, Y. Li, X. Wu, and Z. Yang, "A Review of recent advances in wind turbine condition monitoring and fault diagnosis," in *Power Electronics and Machines in Wind Applications*, Lincoln, NE, May 2009, pp. 1–7.

[145] "Managing the wind: Reducing kilowatt-hour costs with condition monitoring," *Refocus*, vol. 6, no. 3, pp. 48–51, May 2005.

[146] "Fiber Optic Sensors," 2014, accessed 30 June 2014. [Online]. Available: http://www.scaime.com/en/117/products-services/fiber-optic-sensors.html

[147] "BLADEcontrol®: Greater output—less risk," Bosch Rexroth, Tech. Rep., September 2014.

[148] W. Yang, P. J. Tavner, C. J. Crabtree, Y. Feng, and Y. Qiu, "Wind turbine condition monitoring: technical and commercial challenges," *Wind Energy*, 2012.

[149] K. Schroeder, W. Ecke, J. Apitz, E. Lembke, and G. Lenschow, "A fibre Bragg grating sensor system monitors operational load in a wind turbine rotor blade," *Measurement Science and Technology*, vol. 17, no. 5, pp. 1167–1172, April 2006.

[150] M. J. Blanch and A. G. Dutton, "Acoustic emission monitoring of field tests of an operating wind turbine," *Key Engineering Materials*, vol. 245-246, pp. 475–482, 2003.

[151] J. Berg, B. Resor, J. Paquette, and J. White, "SMART wind turbine rotor: Design and field test," Sandia National Laboratories, Tech. Rep. SAND2014-0681, January 2014.

[152] D. Papasalouros, N. Tsopelas, A. Anastasopoulos, D. Kourousis, D. J. Lekou, and F. Mouzakis, "Acoustic emission monitoring of composite blade of NM48/750 NEG-MICON wind turbine," *Journal of Acoustic Emission*, vol. 31, no. 1, pp. 36–49, 2013.

[153] M. A. Rumsey, J. Paquette, J. R. White, R. J. Werlink, A. G. Beattie, C. W. Pitchford, and J. van Dam, "Experimental results of structural health monitoring of wind turbine blades," in *46th AIAA aerospace sciences meeting and exhibit*. Reston, Virigina: American Institute of Aeronautics and Astronautics, 2008, pp. 1–14.

[154] M. J. Sundaresan, M. J. Schulz, and A. Ghoshal, "Structural health monitoring static test of a wind turbine blade," National Renewable Energy Laboratory, Tech. Rep., March 2002.

[155] B. F. Sørensen, L. Lading, P. Sendrup, M. McGugan, C. P. Debel, O. J. D. Kristensen, G. C. Larsen, A. M. Hansen, J. Rheinländer, J. Rusborg, and J. D. Vestergaard, "Fundamentals for remote structural health monitoring of wind turbine blades—a preproject," Risø National Laboratory, Tech. Rep. Risø-R-1336(EN), 2002.

[156] R. W. Hyers, J. G. Mcgowan, K. L. Sullivan, J. F. Manwell, and B. C. Syrett, "Condition monitoring and prognosis of utility scale wind turbines," *Energy Materials*, vol. 1, no. 3, pp. 187–203, September 2006.

[157] "Wind turbine blade inspections," 2014, accessed 10 October, 2014. [Online]. Available: http://www.upwindsolutions.com/repairs/blade-services/blade-inspection/

[158] G. Song, H. Li, B. Gajic, W. Zhou, P. Chen, and H. Gu, "Wind turbine blade health monitoring with piezoceramic-based wireless sensor network," *International Journal of Smart and Nano Materials*, vol. 4, no. 3, pp. 150–166, September 2013.

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

[160] E.-Y. Lin, J. M. Rabaey, and A. Wolisz, "Power-efficient rendez-vous schemes for dense wireless sensor networks," in *Proc. IEEE ICC*. IEEE, 2004.

[161] X. Fafoutis, A. D. Mauro, M. D. Vithanage, and N. Dragoni, "Receiver-initiated medium access control protocols for wireless sensor networks," *Computer Networks*, vol. 76, pp. 55–74, 2015.

[162] Y. Peng, Z. Li, D. Qiao, and W. Zhang, "Delay-bounded MAC with minimal idle listening for sensor networks," in *Proc. IEEE INFOCOM*, 2011.

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

[164] P. Dutta, S. Dawson-Haggerty, Y. Chen, C.-J. M. Liang, and A. Terzis, "A-MAC: A versatile and efficient receiver-initiated link layer for low-power wireless," *ACM Transactions on Sensor Networks (TOSN)*, vol. 8, no. 4, p. 30, 2012.

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

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

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

[168] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust mesh networks through autonomously scheduled TSCH," in *Proc. ACM SenSys*, 2015.

[169] J. Oller, I. Demirkol, J. Casademont, J. Paradells, G. U. Gamm, and L. Reindl, "Has time come to switch from duty-cycled MAC protocols to wake-up radio for wireless sensor networks?" *IEEE/ACM Transactions on Networking (ToN)*, vol. 24, no. 2, pp. 674–687, 2016.

[170] R. Piyare, T. Istomin, and A. L. Murphy, "WaCo: A wake-up radio Cooja extension for simulating ultra low power radios," in *Proc. ACM EWSN*, 2017.

[171] S. Duquennoy, O. Landsiedel, and T. Voigt, "Let the tree bloom: scalable opportunistic routing with ORPL," in *Proc. ACM SenSys*, 2013.

[172] ——, "Let the tree Bloom: scalable opportunistic routing with ORPL," in *ACM SenSys*, Nov. 2013.

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

[174] "2.4 GHz IEEE 802.15.4 / ZigBee-ready RF transceiver," Texas Instruments, Tech. Rep. SWRS041c, 2017. [Online]. Available: http://www.ti.com/lit/ds/symlink/cc2420.pdf

[175] G. Schaefer, F. Ingelrest, and M. Vetterli, "Potentials of opportunistic routing in energy-constrained wireless sensor networks," in *EWSN*, Feb. 2009.

[176] O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson, "Low power, low delay: Opportunistic routing meets duty cycling," in *Proc. ACM IPSN*, 2012.

[177] S. Unterschütz, C. Renner, and V. Turau, "Opportunistic, receiver-initiated data-collection protocol," in *EWSN*, Feb. 2012.

[178] O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson, "Low power, low delay: opportunistic routing meets duty cycling," in *IEEE IPSN*, Apr. 2012.

[179] J. Kim, X. Lin, N. B. Shroff, and P. Sinha, "Minimizing delay and maximizing lifetime for Wireless Sensor Networks With Anycast," *IEEE/ACM Transactions on Networking*, vol. 18, no. 2, pp. 515–528, 2009.

[180] M. Zorzi and R. R. Rao, "Geographic random forwarding (GeRaF) for ad hoc and sensor networks: Multihop performance," *IEEE Transactions on Mobile Computing*, vol. 2, no. 4, pp. 337–348, 2003.

[181] S. Liu, K.-W. Fan, and P. Sinha, "CMAC: An energy-efficient MAC layer protocol using convergent packet forwarding for wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 5, no. 4, pp. 1–34, Nov. 2009.

[182] H.-X. Tan and M. C. Chan, "A^2-MAC: An adaptive, anycast MAC protocol for wireless sensor networks," *IEEE WCNC*, Apr. 2010.

[183] Y. Gu and T. He, "Dynamic switching-based data forwarding for low-duty-cycle wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 10, no. 12, pp. 1741–1754, Dec. 2011.

[184] E. Ghadimi, O. Landsiedel, P. Soldati, S. Duquennoy, and M. Johansson, "Opportunistic routing in low duty-cycle wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 10, no. 4, pp. 1–39, Jun. 2014.

[185] H.-J. Bungartz, S. Zimmer, M. Buchholz, and D. Pflüger, *Modeling and simulation: an application-oriented introduction*. Berlin: Springer, 2014.

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

[187] Y. Chen and A. Terzis, "On the implications of the log-normal path loss model: an efficient method to deploy and move sensor motes," in *ACM SenSys*, Nov. 2011.

[188] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 2–16, Feb. 2003.

[189] L. Mottola and G. P. Picco, "MUSTER: adaptive energy-aware multisink routing in wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 10, no. 12, pp. 1694–1709, Dec 2011.

[190] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *Proc. ACM SenSys*, 2009.

[191] K. Han, Y. Liu, and J. Luo, "Duty-cycle-aware minimum-energy multicasting in wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 21, no. 3, pp. 910–923, Jun. 2013.

[192] K. Han, J. Luo, L. Xiang, M. Xiao, and L. Huang, "Achieving energy efficiency and reliability for data dissemination in duty-cycled WSNs," *IEEE/ACM Trans. Netw.*, vol. 23, no. 4, pp. 1041–1052, Aug. 2015.

[193] H. Gong, L. Fu, X. Fu, L. Zhao, K. Wang, and X. Wang, "Distributed multicast tree construction in wireless sensor networks," *IEEE Transactions on Information Theory*, vol. 63, no. 1, pp. 280–296, Jan 2017.

[194] Q. Chen, H. Gao, S. Cheng, X. Fang, Z. Cai, and J. Li, "Centralized and distributed delay-bounded scheduling algorithms for multicast in duty-cycled wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 6, pp. 3573–3586, Dec 2017.

[195] R. Kelsey and J. Hui, "Multicast protocol for low-power and lossy networks (MPL)," RFC 7731, Feb. 2016. [Online]. Available: https://rfc-editor.org/rfc/rfc7731.txt

[196] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele, "Low-power wireless bus," in *Proc. ACM SenSys*, 2012.

[197] O. Landsiedel, F. Ferrari, and M. Zimmerling, "Chaos: Versatile and efficient all-to-all data sharing and in-network processing at scale," in *Proc. ACM SenSys*, 2013.

[198] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with glossy," in *Proc. ACM/IEEE IPSN*, 2011.

[199] G. Oikonomou, I. Phillips, and T. Tryfonas, "IPv6 multicast forwarding in RPL-based wireless sensor networks," *Wireless Personal Communications*, vol. 73, no. 3, pp. 1089–1116, 2013.

[200] G. G. Lorente, B. Lemmens, M. Carlier, A. Braeken, and K. Steenhaut, "BMRF: bidirectional multicast RPL forwarding," *Ad Hoc Networks*, vol. 54, pp. 69–84, 2017.

[201] M. Conti, P. Kaliyar, and C. Lal, "REMI: a reliable and secure multicast routing protocol for IoT networks," in *Proc. ACM ARES*, 2017, pp. 1–8.

[202] A. Förster and A. L. Murphy, "FROMS: a failure tolerant and mobility enabled multicast routing paradigm with reinforcement learning for WSNs," *Ad Hoc Networks*, vol. 9, no. 5, pp. 940 – 965, 2011.

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

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

[205] 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," *Journal of Sensors*, vol. 2015, pp. 1–11, 2015.

[206] 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 Materials and Structures*, vol. 26, no. 12, pp. 1–11, 2017.

[207] 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.

[208] 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.

[209] Texas Instruments. (2017) The SensorTag story. Accessed: November 3, 2017. [Online]. Available: http://www.ti.com/ww/en/wireless_connectivity/sensortag/

[210] SpectraQuest, Inc. (2017) SpectraQuest, Inc. Accessed: November 3, 2017. [Online]. Available: http://spectraquest.com/

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

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

[213] Y. Leng, D. Wenfeng, S. Peng, X. Ge, G. J. Nga, and S. Liu, "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, June 2012.

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

[215] X. Chen, S. Jin, and D. Qiao, "M-PSM: Mobility-aware power save mode for IEEE 802.11 WLANs," in *Proc. IEEE ICDCS*, 2011.

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

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

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

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

[220] National Renewable Energy Laboratory. (2017, July) NWTC Information Portal (FAST v8). Accessed: November 5, 2017. [Online]. Available: https://nwtc.nrel.gov/FAST8

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

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