2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)

# RIVER-MAC: A Receiver-Initiated Asynchronously Duty-Cycled MAC Protocol for the Internet of Things

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

***Abstract*—This paper presents RIVER-MAC, a very efficient receiver-initiated asynchronously-duty-cycled medium access control (MAC) protocol for IoT devices. The key innovations of RIVER-MAC include (1) a *CCA-based rendezvous* to reduce idle listening for the sender node by an order of magnitude, and (2) a *beacon train-based collision resolution* scheme to reduce contention between receiver nodes, a previously-overlooked issue in receiver-initiated MAC protocol design. We have implemented RIVER-MAC in Contiki OS, and used extensive Cooja simulations to demonstrate its high performance compared to RI-MAC (a classic receiver-initiated protocol), as well as ContikiMAC (a state-of-the-art sender-initiated duty-cycled MAC protocol) in our tested scenarios. We also have used analytic studies to show that RIVER-MAC yields a comparable performance with a wakeup radio-based scheme, an emerging alternative to duty-cycled MAC protocols for IoT devices.**

***Keywords*-Internet of Things; Wireless Sensor Networks; MAC Protocols; Low-power Communications**

## I. 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 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 [1], a state-of-the-art sender initiated asynchronously duty-cycled MAC protocol, and RI-MAC [2], a classic receiver-initiated protocol. Inspired by these protocols' successes, and learning from their shortcomings, we here propose 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.

## II. Related Work

Energy efficiency through duty-cycling has been a popular research topic for WSNs for well over a decade, as summarized in surveys such as [3]. At the link layer, protocols such as B-MAC [4], X-MAC [5], and ContikiMAC [1] 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

978-1-7281-2607-4/19/$31.00 ©2019 IEEE
DOI 10.1109/COMPSAC.2019.00126

or acknowledges the packet. We discuss the shortcomings of these *sender-initiated* protocols in Section III.

*Receiver-initiated* protocols [2], [6]–[10] use *low-power probing* (LPP) to achieve similar results. In LPP, nodes periodically broadcast a probe packet, or *beacon*, to advertise their availability as a receiver. Senders simply listen for a beacon from their intended receiver. Example protocols include RI-MAC [2], a classic receiver-initiated protocol, and A-MAC [10], 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 III. Some work, such as [8], 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 [11], [12] and T-MAC [13]. 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 [14]. We acknowledge these developments as promising, but focus on flexible, single-channel protocols that forego the overhead and complexity of synchronization and scheduling.

An emerging alternative for traditionally duty-cycled MAC protocols is wakeup radios (WuR) [15], [16]. 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 VI, we analytically compare RIVER-MAC with a WuR platform to better understand which approach is more appropriate in different scenarios.

## III. 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 observe the shortcomings of receiver-initiated protocols, such as RI-MAC, that motivate us to propose a new protocol, RIVER-MAC.

### A. Shortcomings of Sender-Initiated Protocols

The ContikiMAC protocol [1] is a state-of-the-art sender-initiated protocol and, due to its inclusion in Contiki OS [17], 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. [8], [9].

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.

### B. 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 [2], a classic receiver-initiated protocol. We then explore its shortcomings, which motivate our design for an improved receiver-initiated protocol.

*1) Description of RI-MAC:* RI-MAC [2] is a receiver-initiated, asynchronously duty-cycled MAC protocol. In RI-MAC (Fig. 1), 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.

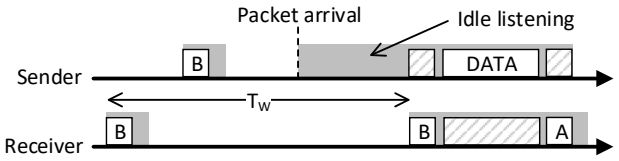


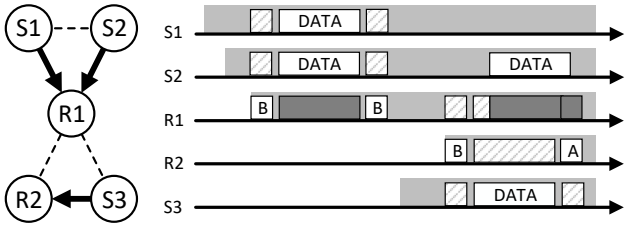

Figure 1. Overview of RI-MAC, with a timeline for a sender and a receiver. Both nodes broadcast beacon packets (B) at an interval $T_W$. Data packets (DATA) are acknowledged with acknowledgement beacons (A). Shaded areas indicate radio on-time, and hatched boxes indicate received packets.

As shown in Fig. 1, when a node has a packet to send, it wakes and *idly listens* for its receiver to 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.

*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. 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) 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. 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 IV), making it more suitable for high-density IoT applications.




(a) Topology. (b) Timeline.

Figure 2. Example of excess collisions from contending receivers. In (a), arrows indicate communication flows, while a dashed line indicates nodes are within communication range of each other. In (b), darkly shaded boxes indicate 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).

## IV. 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 III-B1. 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. CCA-based Rendezvous

Our first major improvement is a *CCA-based rendezvous*. As shown in Fig. 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.

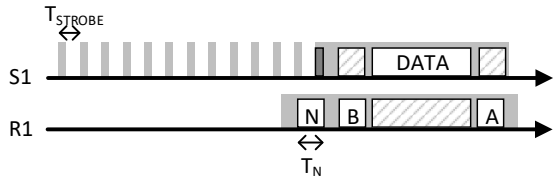



Figure 3. The CCA-based rendezvous of RIVER-MAC. Periodic wakeups now begin with an initial beacon (N).

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 ($T_{\text{STROBE}}$) must be no greater than the transmit time of the initial beacon packet ($T_N$). 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 IV-C1.

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.

### B. 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 III. Our goal is for an active receiver to reserve the channel resource so that additional receivers cannot use it. As shown in Fig. 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.

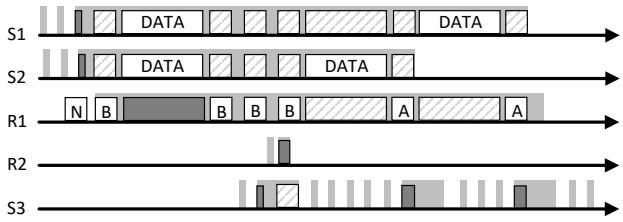



Figure 4. Illustration of the beacon train-based collision resolution. After R1 detects a collision (between S1 and S2), it repeatedly transmits beacons until the senders respond. This prevents R2 from accessing the channel while S1 and S2 are backing off.

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 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 $T_B$.

### C. Practical Considerations

*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 $T_{\text{STROBE}}$ 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 V-B, 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.

*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 results 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 must use another short timeout ($T_I$) to decide if an acknowledgement beacon is incoming. If not, the sender 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 does not know the acknowledgement was lost, and it continues normally. This results in the sender re-sending the same packet in the next wakeup interval, which can be resolved with 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) 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. 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. Otherwise, RIVER-MAC's delay (and energy) performance could be improved through an opportunistic cross-layer approach, as in [18], [19].

## V. Simulation Study

### A. Implementation and Setup

We have implemented RIVER-MAC in Contiki OS [17] 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 [18] to improve software acknowledgement (softack) support. RIVER-MAC's software architecture is shown in Fig. 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.

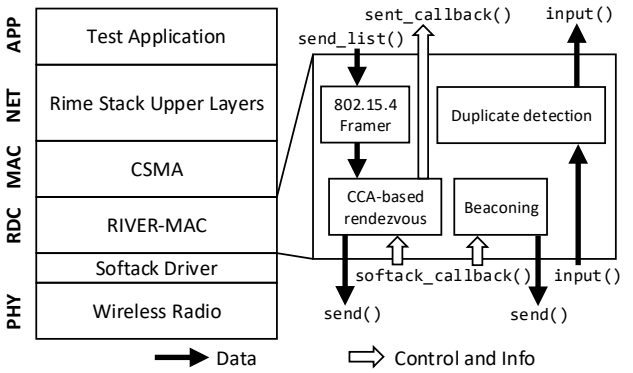



Figure 5. Software architecture of RIVER-MAC's Contiki implementation.

We evaluate our implementation through Cooja [20] simulations, using the Z1 platform, which is based on the MSP430 microcontroller and the Texas Instruments CC2420 802.15.4-compliant radio [21]. 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.

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.

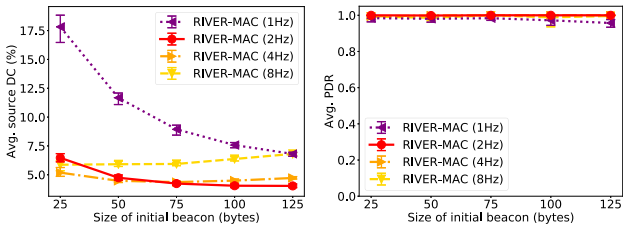
Duty cycle is the percentage of radio on-time and is a hardware-independent proxy for the energy consumption of the radio. We use the duty cycle calculated by Cooja, starting after the network has reached a steady state. Reported duty cycles are averaged first over the nodes in a simulation, then over 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.

### B. RIVER-MAC Parameter Selection

We first need to choose an appropriate default initial beacon size, as described in Section IV-C1. 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. 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 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.

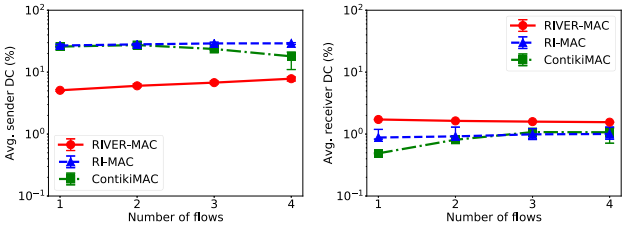
### C. Clique Networks

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 one sender sending to one 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. 7 and 8.

In Fig. 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.

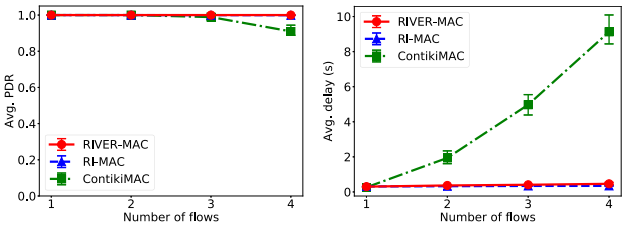
The sender duty cycles of RI-MAC and ContikiMAC are just above 25%. This is expected, because a 2 Hz wakeup rate corresponds to 0.5 s between wakeups, and a sender waits for half of that on average. Since a packet is sent every second, this results in an average of 0.25 s of idle listening per second. 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. 7b, which shows the average duty cycle for receivers. RIVER-




(a) Sender duty cycle. (b) Receiver duty cycle.

Figure 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.




(a) Packet delivery ratio. (b) Delay.

Figure 8. Reliability and delay results for the clique networks.

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.

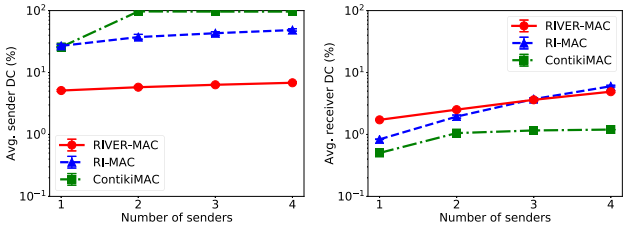
Fig. 7a also shows that, as the number of flows increases, ContikiMAC's average sender duty cycle decreases. However, this is not an indicator of better performance. Instead, this is due to dropped packets, as can be seen in Fig. 8a. In our simulations, packets are dropped after eight retries. Because 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. 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.

### D. Hidden-node Networks

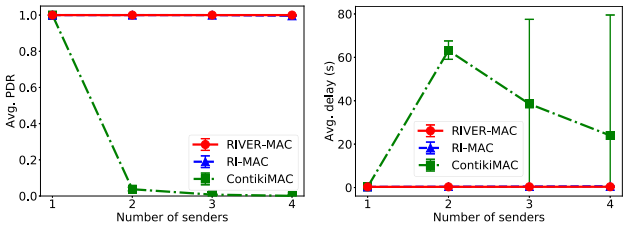
We next evaluate in a small network with hidden nodes. The topology is a single-hop star, with a varying number of senders and a single receiver in the center. All senders are hidden from each other, potentially resulting in collisions at

the receiver. Other settings remain the same as the clique networks. The results are shown in Figs. 9 and 10.




(a) Sender duty cycle. (b) Receiver duty cycle.

Figure 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 10. Reliability and delay results for the hidden-node network.

RIVER-MAC performs well in this scenario, with a sender duty cycle (Fig. 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. 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. 9a) grows to nearly 100% when the second sender is introduced, and the PDR (Fig. 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. 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. 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.

### *E. 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. 11 to 13.

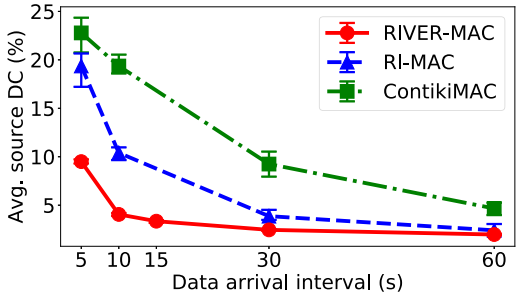
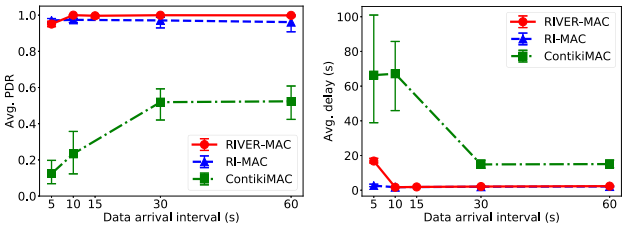



Figure 11. Average source duty cycle for tree network.




(a) Packet delivery ratio. (b) Delay.

Figure 12. Reliability and delay results for the tree network.

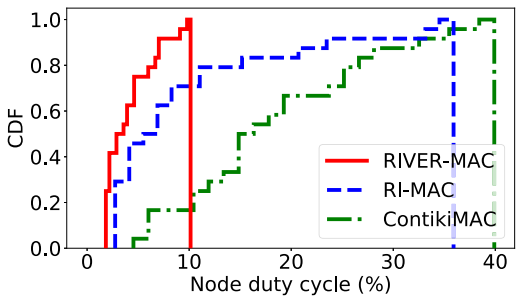



Figure 13. CDF of source node duty cycle for a single run of the tree network at a 10 s data interval.

RIVER-MAC performs consistently well in our tested range of data arrival intervals (5 s to 60 s). Fig. 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 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. 12a), and showing high delay at small data intervals (Fig. 12b).

Finally, Fig. 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.

## VI. 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 [16], has recently been implemented. However, WaCo does not provide a straightforward method for comparing energy consumption of WuR schemes with that of traditional duty-cycled MACs such as RIVER-MAC. Therefore, we instead use analytical models for our comparisons.

### A. 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 consecutive packets sent by a node, plus the time to send one of the packets. We assume that one packet is sent in the forwarding interval, and that it is successfully transmitted on its first attempt. Since current consumption data is readily available, we specifically model the average current consumption in a forwarding interval, $i^{\text{RIV}}_{\text{AVG}}$. To find the average current, we divide the charge consumption into three parts: sending the packet, receiving the packet, and overhead from periodically beaconing.

*1) Sending Charge Consumption:* The sending charge consumption $Q^{\text{RIV}}_S$ is a function of the wakeup interval $T_W$ and composed of the following parts:

- $Q^{\text{RIV}}_{\text{CCA}}$, 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 $T_{\text{CCA}}$) per initial beacon duration $T_N$, times the idle current $i_{\text{IL}}$.

$$Q^{\text{RIV}}_{\text{CCA}}(T_W) = \frac{T_W}{2}\frac{T_{\text{CCA}}}{T_N} i_{\text{IL}}. \tag{1}$$

- $Q^{\text{RIV}}_{\text{BRX}}$, the charge consumed while receiving beacons. We assume half of an initial beacon is received with reception current $i_{\text{RX}}$, plus one regular beacon of duration $T_B$. For brevity, we also include miscellaneous factors, such as the space between packets, in $T_B$.

$$Q^{\text{RIV}}_{\text{BRX}} = \frac{T_N i_{\text{RX}}}{2} + T_B i_{\text{RX}}. \tag{2}$$

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

$$Q^{\text{RIV}}_{\text{DTX}} = T_D i_{\text{TX}} + T_A i_{\text{RX}}. \tag{3}$$

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

$$Q^{\text{RIV}}_S(T_W) = Q^{\text{RIV}}_{\text{CCA}}(T_W) + Q^{\text{RIV}}_{\text{BRX}} + Q^{\text{RIV}}_{\text{DTX}}. \tag{4}$$

*2) Receiving Charge Consumption:* The charge consumed while receiving, $Q^{\text{RIV}}_R$, is the charge consumed from receiving a data packet and from transmitting an acknowledgement beacon:

$$Q^{\text{RIV}}_R = T_D i_{\text{RX}} + T_A i_{\text{TX}}. \tag{5}$$

*3) Beaconing Overhead Charge Consumption:* The beaconing overhead charge consumption $Q_O^{\text{RIV}}$ is a function of $T_F$ and $T_W$. The charge consumption for a single wakeup, $Q_W^{\text{RIV}}$, consists of the energy for the pre-beaconing CCA strobe (one $T_{\text{CCA}}$ 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:

$$Q_W^{\text{RIV}} = \frac{T_{\text{CCA}}}{T_B}(T_I + T_B)i_{\text{IL}} + (T_N + T_B)i_{\text{TX}} + T_L i_{\text{IL}}. \quad (6)$$

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

$$Q_O^{\text{RIV}}(T_F, T_W) = Q_W^{\text{RIV}} \frac{T_F}{T_W}. \quad (7)$$

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

$$i_{\text{AVG}}^{\text{RIV}}(T_F, T_W) = \frac{Q_S^{\text{RIV}}(T_W) + Q_R^{\text{RIV}} + Q_O^{\text{RIV}}(T_F, T_W)}{T_F}. \quad (8)$$

### B. Model of WuR Energy Consumption

For a wakeup radio scheme, we model the transmitter-initiated WuR scheme from [15]. 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, $Q_S^{\text{WuR}}$, comes from the WuC of duration $T_{\text{WuC}}$ sent with current $i_{\text{WTX}}$, and the data packet sent with the main radio, as follows:

$$Q_S^{\text{WuR}} = T_{\text{WuC}} i_{\text{WTX}} + T_D i_{\text{TX}}. \quad (9)$$

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

$$Q_R^{\text{WuR}} = T_{\text{WuC}} i_{\text{WRX}} + T_D i_{\text{RX}}. \quad (10)$$

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

$$Q_O^{\text{WuR}}(T_F) = (T_F - T_{\text{WuC}} - T_D) i_{\text{WS}}. \quad (11)$$

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

$$i_{\text{AVG}}^{\text{WuR}}(T_F) = \frac{Q_S^{\text{WuR}} + Q_R^{\text{WuR}} + Q_O^{\text{WuR}}(T_F)}{T_F}. \quad (12)$$

### C. 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 [21] and timing values taken from Cooja. These parameters are shown in Table I. 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 [15], summarized in Table II.

Table I
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 | $T_{\text{CCA}}$ | 0.38 ms |
| TX current | $i_{\text{TX}}$ | 17.4 mA |
| RX current | $i_{\text{RX}}$ | 18.8 mA |
| Idle current | $i_{\text{IL}}$ | 18.8 mA |

Table II
PARAMETER VALUES FOR WUR MODEL, TAKEN FROM [15].

| Parameter | Notation | Value |
|---|---|---|
| WuC duration | $T_{\text{WuC}}$ | 12.2 ms |
| Data packet duration | $T_D$ | 2.5 ms |
| WuR sleep current | $i_{\text{WS}}$ | 3.5 $\mu$A |
| WuR RX current | $i_{\text{WRX}}$ | 8.0 $\mu$A |
| WuR TX current | $i_{\text{WTX}}$ | 152 mA |
| Main TX current | $i_{\text{TX}}$ | 17.4 mA |
| Main RX current | $i_{\text{RX}}$ | 18.8 mA |

In Fig. 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 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. 7a with one flow.

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 V, where interference and hidden nodes are factors.

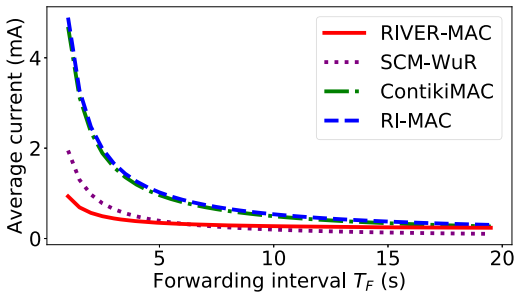



Figure 14. Analytical results for average current vs. forwarding interval.

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 [16] tool.

## VII. 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.

## Acknowledgements

Funded in part by U.S. National Science Foundation Grant No. 1730275.

## References

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

[2] Y. Sun, O. Gurewitz, and D. B. Johnson, "RI-MAC: a receiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks," in *Proc. ACM SenSys*, 2008.

[3] 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.

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

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

[6] 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.

[7] 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.

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

[9] 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.

[10] 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.

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

[12] ——, "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.

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

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

[15] 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.

[16] 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.

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

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

[19] 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.

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

[21] "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