Design and Evaluation of a Versatile and Efficient Receiver-Initiated Link Layer for Low-Power Wireless

Prabal Dutta[†], Stephen Dawson-Haggerty[‡], Yin Chen[⋆], Chieh-Jan Mike Liang[⋆], and Andreas Terzis[⋆]

[†]Computer Science & Eng. Division
University of Michigan
Ann Arbor, MI 48109
prabal@eecs.umich.edu

[‡]Computer Science Division
University of California, Berkeley
Berkeley, CA 94720
stevedh@cs.berkeley.edu

[⋆]Computer Science Department
Johns Hopkins University
Baltimore, MD 21218
{yinchen,cliang4,terzis}@cs.jhu.edu

Abstract

We present A-MAC, a receiver-initiated link layer for low-power wireless networks that supports several services under a unified architecture, and does so more efficiently and scalably than prior approaches. A-MAC's versatility stems from layering unicast, broadcast, wakeup, pollcast, and discovery above a single, flexible synchronization primitive. A-MAC's efficiency stems from optimizing this primitive and with it the most consequential decision that a low-power link makes: whether to stay awake or go to sleep after probing the channel. Today's receiver-initiated protocols require more time and energy to make this decision, and they exhibit worse judgment as well, leading to many false positives and negatives, and lower packet delivery ratios. A-MAC begins to make this decision quickly, and decides more conclusively and correctly in both the negative and affirmative. A-MAC's scalability comes from reserving one channel for the initial handshake and different channels for data transfer. Our results show that: (i) a unified implementation is possible; (ii) A-MAC's idle listening power increases by just 1.12× under interference, compared to 17.3× for LPL and 54.7× for RI-MAC; (iii) A-MAC offers high single-hop delivery ratios, even with multiple contending senders; (iv) network wakeup is faster and far more channel efficient than LPL; and (v) collection routing performance exceeds the state-of-the-art.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms

Design, Experimentation, Performance, Standardization

Keywords

Link protocols, MAC protocols, wireless sensor networks

SenSys'10, November 3–5, 2010, Zurich, Switzerland.

1 Introduction

A receiver-initiated link layer is one in which the receiver triggers communications by first transmitting a probe. Receiver-initiated protocols have experienced a renewed interest because they offer many benefits over sender-initiated protocols for low-power wireless: they [17, 33] handle hidden terminals better than sender-initiated ones [30, 38, 39]; their low-power probing (LPP) mechanism ([28]) supports asynchronous communications but avoids the long preambles of sender-initiated low-power listening (LPL, [20, 30]) which run afoul of regulatory standards [2]; they support extremely low duty cycles [28] or high data rates [33]; and they support many low-power services including wakeup [12], discovery [11], broadcast [32], anycast [13], and pollcast [9].

Despite these many benefits, receiver-initiated protocols face a number of drawbacks as well. Their fundamental synchronization primitive – the probe – costs more than channel sampling, which means that baseline power draw is higher than sender-initiated protocols. Their frequent probe transmissions can congest the channel and delay data communications, which affects their scalability under even light traffic loads. Their use of incompatible probe semantics for different services makes concurrent use of those services difficult: some probes use hardware acknowledgments [13, 28] while others do not [9, 33]; some probes include only receiver-specific data [28, 33] while others may also include sender-specific data [9, 13]; and some probes include contention windows [13, 33] while others do not [9, 28]. These differences raise the question of whether it is possible to design a general-purpose, yet efficient, receiver-initiated link layer.

In this paper, we present A-MAC, a new receiver-initiated link layer that shows it is possible to support multiple services under a unified architecture, and to do so more efficiently and scalably than prior designs. Thus, we narrow the gap between sender- and receiver-initiated approaches to low-power wireless. A-MAC uses the backcast synchronization primitive – a probe/ack frame exchange – to determine quickly, robustly, and in constant time whether inbound traffic is pending [13]. All other services are multiplexed above the primitive or piggybacked on the probe. To minimize contention between probe and data traffic, A-MAC (optionally) uses one or more secondary channels to complete data transfer after the initial probe, allowing A-MAC to scale with density and load [22, 25]. Section 3 presents the A-MAC design.

Lacking proper hardware support, A-MAC achieves its high performance by dynamically reassigning hardware addresses, making use of hardware address recognition, and leveraging hardware acknowledgment collisions. While these mechanisms misappropriate addresses, violate standards, and abuse acknowledgments, the underlying ideas are more principled, and we believe they highlight new directions for radio hardware and link protocols. The techniques allow us to implement unicast, broadcast, wakeup, and pollcast using today's off-the-shelf radios within a unified framework that exposes a standard TinyOS `ActiveMessage` interface, allowing drop-in use with many existing codebases. Our description omits asynchronous neighbor discovery due to space constraints, but supporting discovery is a matter of systematically scheduling the probe and listen times [11]. Section 4 details our prototype implementation, and the various mechanisms we employ, to demonstrate the value of hardware support for a receiver-initiated link layer.

Sections 5 and 6 explore A-MAC's microbenchmarks and macrobenchmarks, respectively. Key microbenchmarks include evaluating the robustness of the fundamental synchronization mechanism (including effects of path delays, path loss, and neighborhood density). We also provide energy microbenchmarks for A-MAC's probe, receive, transmit, and idle listening energy costs, and we present how these figures translate to average current across a range of probe and data periods. We show that A-MAC's idle listening power increases by just 1.12× in the presence of interference, compared to 17.3× for LPL and 54.7× for a recent receiver-initiated MAC, RI-MAC [33]. Our macrobenchmarks show that A-MAC offers higher single-hop delivery ratios with multiple contending senders than RI-MAC as well. We also show that network wakeup is nearly twice as fast as LPL and uses vastly fewer transmissions, making A-MAC far more channel efficient. Finally, we show that collection routing with CTP [18] over A-MAC outperforms the state-of-the-art.

The A-MAC design faces a number of obvious limitations, however. Timing critical operations require low-level hardware support, which is only partly provided today, hampering broader use. Some of the design choices violate current standards (like acknowledging broadcast frames), but our work shows there are significant gains to be won by doing so. Since communications is receiver-initiated, the basic primitive is a probe, which means baseline channel usage scales with node density rather than data rate. For low or medium density networks, this is not an issue, but for higher density networks, it could affect latency. Although using one probe channel and (optional) secondary channels for data transfer helps significantly, very high neighborhood densities might also require coordinating probe transmissions [8], which we do not explore in this paper.

2 Related Work

Since radio communications dominate node-level energy consumption, it is not surprising that a wide range of MAC protocols have been proposed for low-power wireless networks. Low-power links provide a range of service abstractions, allowing nodes to synchronize with peers, contend for the channel, discover neighbors, and transfer data.

Depending on which end of a communication link initiates a transfer, a MAC can be classified as either *sender-initiated* or *receiver-initiated*. Among the sender-initiated protocols, LPL/B-MAC [20, 30], Hui's MAC [22], SCP [39], S-MAC [38], T-MAC [37], and X-MAC [5], represent canonical design points, and Flash [26] represents a link layer flooding protocol. Among the receiver-initiated protocols, PTIP [15], RI-MAC [33], LPP/Koala[28], Pollcast [9], Backcast [13], and ADB [32] offer a range of both conventional and more exotic communication abstractions. The rest of this section compares the abstractions they provide and the low-power synchronization mechanisms they employ.

B-MAC, S-MAC, T-MAC, X-MAC, and SCP all offer unicast and broadcast. RI-MAC offers just unicast but ADB essentially extends RI-MAC to offer a broadcast service. The Koala system uses low-power probing (LPP) to offer a receiver-initiated, asynchronous network wakeup. The Flash flooding protocol uses low-power listening (LPL) to offer sender-initiated wakeup. Pollcast offers single-hop collaborative feedback, which allows a node to pose true/false predicates to neighbors. Backcast offers an optimized acknowledged anycast service that can implement Pollcast and LPP. A-MAC offers all of these service abstractions – unicast, broadcast, flood, wakeup, and pollcast, layered above backcast and within a unified link layer architecture.

Low-power wireless protocols must synchronize their communications either explicitly by scheduling communication windows or implicitly by sampling or probing for pending traffic. S-MAC, T-MAC, and SCP all schedule communication windows: S-MAC uses fixed windows, T-MAC adjusts the window size to match the traffic load, and SCP adjusts the window size to account for clock drift.

B-MAC, X-MAC, and Hui's MAC employ channel sampling techniques to detect pending traffic. B-MAC sends long preambles which receivers detect with channel sampling. X-MAC senders transmit "packetized preambles" and listen for a receiver-generated acknowledgment between packets, which reduces expected channel occupancy. Hui's MAC employs a packetized preamble as well but transmits preamble "chirps" which contain rendezvous time and channel data. As an optimization, neighbor sleep schedules are also cached. PTIP, RI-MAC, and Pollcast all employ probing by transmitting probe packets. Both LPL-based sampling and LPP-based probing are vulnerable to false positives (waking up when no traffic is pending) or false negatives (prematurely falling asleep when traffic is pending) [4].

A-MAC transmits a probe as well, but uses explicit hardware-generated acknowledgments as part of its synchronization mechanism. The use of a probe/ack frame exchange allows A-MAC to determine quickly, robustly, and in constant time whether inbound traffic is pending. This mechanism, called backcast [13], runs over 802.15.4 radios using O-QPSK modulation [23], but similar schemes have been shown to work for OFDM modulation as well [10]. A-MAC also caches neighbor probe times, reducing radio on time. Finally, A-MAC includes multichannel rendezvous information on the first transmitted probe, which reduces congestion and increases capacity through spectrum reuse. A-MAC essentially integrates several earlier optimizations.

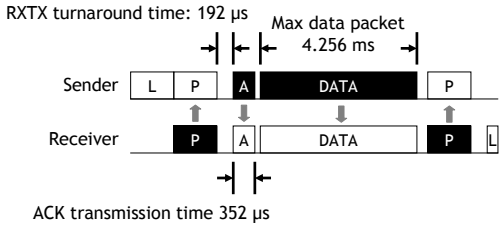


Figure 1. A-MAC communications timing and flow. A sender listens (L) for a receiver's probe (P) which it auto-acks (A) precisely 192 μs later. The sender subsequently transmits a data frame (DATA) after a short but random interval, perhaps on a different channel, which the receiver acknowledges with a second probe and then listens briefly for an auto-ack before returning to sleep.

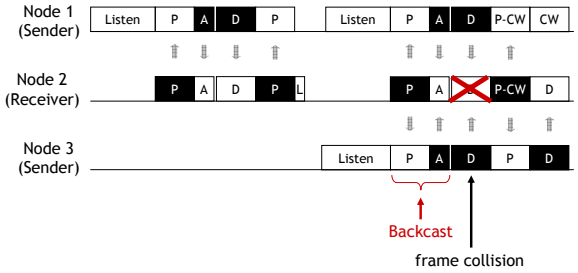


Figure 2. A contention-free transfer (left) and a collision (right). Although the auto-ack frames collide, they do so non-destructively, so the receiver correctly decodes their superposition as a valid frame. Hence, the receiver concludes that traffic is pending, so it retransmits a probe with an explicit contention window, which Node 3 wins.

3 A-MAC Design Overview

This section presents the design of A-MAC, a receiver-initiated link layer for low-power wireless networks that supports several services under a unified architecture. We ground our discussion in the context of the IEEE 802.15.4 standard. The basic A-MAC design requires a sender to first listen for a probe frame from the intended receiver, then acknowledge the frame using the 802.15.4 standard's support for hardware automatic acknowledgments (*auto-ack* or *HACK*), then pause for a short, random delay, and finally transmit the data frame if the channel is clear.

Figure 1 shows the critical time constants of an optimized A-MAC communication over 802.15.4. In this figure, the probe, labeled **P**, is a standard data frame transmitted by the receiver with the acknowledgment request bit set. The sender, upon receiving this probe frame, generates an auto-ack, labeled **A**. The 802.15.4 standard stipulates that the auto-ack must be generated precisely 12 symbol periods (192 μs) after the end of **P**. The auto-ack frame is 11 bytes long[1] and requires 352 μs to transmit. A sender transmits a **DATA** frame with a short, random delay after the auto-ack **A**, potentially on a different channel as stipulated in the probe. A second probe acknowledges the data frame. If the second probe does not trigger an auto-ack, the receiver goes to sleep.

This design choice – to use an auto-ack – departs from prior work in which receiver-initiated MACs simply send a data frame in response to a probe [17, 33]. This decision is motivated by the observation that *the most consequential decision that a low-power MAC makes after polling the channel is whether to stay awake or go back to sleep*. Since this decision must be made on the order of one hundred thousand times or more per day in a typical low-power MAC, being indecisive or incorrect can get very costly very quickly. If the MAC decides traffic is pending when none exists – *a false positive* – then the radio will remain on, wasting energy. If the MAC decides no traffic is pending when some is – *a false negative* – then the sender's energy is wasted, communication latency increases, and packet goodput drops.

[1] A hardware auto-ack or HACK frame includes: preamble (4), start-of-frame delimiter (1), length (1), frame control (2), sequence number (1), frame check sequence (2).

Clearly, making a good decision about whether to stay awake or go to sleep is a critical one, but it is not an easy one for many reasons. First, external *interference* (e.g., 802.11 network) might be mistaken for legitimate radio activity. Second, a receiver might *overhear* a partial packet sent to a different node, and stay awake until it can conclude that the packet is destined elsewhere. Third, hidden terminals might cause packets from multiple senders to *collide* at the receiver. Note that it might not be possible for the receiver to differentiate collisions from interference, forcing the radio to stay awake for shorter than required or longer than desired.

Our design reliably and efficiently balances these conflicting needs by using backcast, a link layer primitive that allows a node to probe all of its neighbors in parallel and robustly distinguish the case of zero replies (indicating no pending traffic) from the case of one or more replies (indicating pending traffic) [13]. In the former case, the MAC can turn off the radio quickly[2] and return to a sleep state. In the latter case, the MAC would leave the radio on to receive the auto-ack frame and any additional data frames. Note that all senders with pending traffic for a particular receiver concurrently transmit an auto-ack, as Figure 2 shows. Although these auto-acks collide, they do so non-destructively with high probability. Therefore, the receiver can decode their superposition as a valid frame and conclude that traffic is pending. In the case of a data frame collision, the receiver retransmits the probe with a larger contention window.

All other link layer services are implemented above the backcast synchronization primitive using a combination of hardware auto-acks and judicious frame filtering. Unicast, in principle, could be implemented by auto-ack-ing frames based on the probe source address. Broadcast and wakeup could be implemented by auto-ack-ing all probes which have the ACK request bit set in the 802.15.4 frame control field. Pollcast could be implemented by including a predicate in the probe itself, which is quickly evaluated by the sender and if found true, then auto-acked. Unfortunately, the needed hardware support is lacking in modern radios, requiring some creative contortions, which we describe next.

[2] Since the radio would not signal a start-of-frame (SFD) event.

4 Implementation Details

Section 3 presents a conceptual, clean-slate design for the A-MAC link layer. Unfortunately, modern radios lack the hardware and software support needed to optimally implement the A-MAC design. To work around the limitations of current hardware, we implement a version of A-MAC that misappropriates addresses, violates standards, and abuses acknowledgments. However, the goal of our work is to demonstrate the power and performance benefits of the design; the underlying ideas are more principled than the hacks we employ to accomplish this goal. We hope this work highlights new directions for radio hardware and link protocols.

4.1 Software, Hardware, and Radio Platform

A-MAC is implemented in TinyOS 2.1 [21] and runs on the Berkeley TelosB [31] and Epic [14] motes. The backcast synchronization primitive of A-MAC also runs on the Crossbow Iris [7] mote, but we did not implement the rest of A-MAC on the Iris mote because the radio-processor interface is more limited, due to fewer handshake lines, than the TelosB and Epic platforms, which offer better A-MAC performance due to a more efficient processor-radio interface.

The TelosB and Epic platforms are based on the TI CC2420 radio [34] while the Iris uses the Atmel AT86RF230 radio [3]. Both the CC2420 and the AT86RF230 radios are 802.15.4 standards-compliant and they inter-operate at a 250 kbps data rate. Therefore, they both support backcast using offset quadrature phase shift keying (O-QPSK) modulation with half-sine pulse shaping [19] used in the 802.15.4 standard [23]. This modulation technique employs continuous-phase frequency shift keying and is also known as minimum shift keying (MSK) [29].

4.2 Backcast-Based Synchronization

We implement the backcast synchronization primitive using the hardware automatic acknowledgments (auto-acks) available in all 802.15.4 standards-compliant radios. The scheme works as follows on the CC2420 radio. A receiver transmits a frame to a unicast, multicast, or broadcast address. Nodes with pending traffic for the receiver temporarily set their radio's local hardware address to the particular destination address transmitted in the probe frame by the receiver (this address is a special value, specific to the service, and described later in this section). All nodes that match the destination address transmitted in the probe frame respond with identical acknowledgment frames that are automatically generated by their radio hardware. Receiving an auto-ack signals to the receiver that inbound traffic is pending.

More generally, the 802.15.4 MAC defines a frame control field (FCF) that includes an acknowledgment request flag. On the CC2420, when configured for automatic acknowledgments, an auto-ack frame is transmitted after an incoming frame meets three conditions: it (i) has the acknowledgment request flag set, (ii) is accepted by the radio's address recognition hardware, and (iii) contains a valid CRC. Acknowledgments are transmitted without performing clear channel assessment, so their timing is not delayed due to interference [23, 34].

4.3 Unicast Communications

In typical receiver-initiated unicast communications, a sender first listens for a probe frame and then transmits a data frame in response to the probe. The sender may jitter the data transmission with a small, random delay to avoid collisions when multiple senders are contending. Protocol processing overhead can introduce additional delays in generating the data frame (unless it is preloaded into the radio's transmit buffer): the sender must receive the probe, copy it from the radio to the processor memory, signal an interrupt, dispatch the frame to the link layer, determine if the frame is indeed a probe from the intended receiver, and if so, then possibly jitter the transmission, and finally copy the data frame into the radio's transmit buffer and issue a transmit command. Meanwhile, the receiver must wait patiently with its radio turned on, wasting precious energy and remaining susceptible to false positives from external interference.

The A-MAC unicast design diverges from traditional receiver-initiated designs by first acknowledging the probe with a fast and deterministic radio-generated frame (a backcast frame exchange [13]), and only then sending the data frame. This approach has many benefits. First, the receiver only has to wait marginally longer than the radio's RX/TX turnaround time before concluding that no inbound traffic is present, saving considerable energy on every probe. In the IEEE 802.15.4 standard, a turnaround occurs in 192 μs, nearly 20 times faster than the 3.75 ms beacon-data turnaround time that RI-MAC requires with its software-based protocol processing [33]. Second, our approach distinguishes between collisions and interference, whereas RI-MAC cannot. In RI-MAC, as with LPL channel samples, interference leads to extended listening. With a backcast-based approach, interference is easily distinguished from an auto-ack superposition since the former appears as just channel energy while the latter results in a valid frame reception. Therefore, A-MAC is far less susceptible to interference-based false alarms than either LPL or RI-MAC.

To implement unicast, we use two key features of 802.15.4-compliant radios: hardware-based address filtering and hardware-generated auto-acks. The critical design question is *what source and destination addresses should be used in the probe frame?* One option is to send the probe to the broadcast address requesting an auto-ack. Under this scheme, a node with pending traffic for *any* destination enables auto-acks for broadcast frames.

However, there are several problems with this approach, as follows. First, a sender will auto-ack *every* probe it receives, including probes from neighbors for which the sender has no pending traffic. This will cause all but one neighbor to stay awake unnecessarily and waste energy. We call this the *overreacting* problem. Second, the IEEE 802.15.4-2006 standard specifically prohibits this behavior: § 7.5.6.4, "... any frame that is broadcast shall be sent with its Acknowledgment Request subfield set to zero." Third, because this behavior is prohibited, it enjoys somewhat mixed radio support: while the CC2420 [34] radio and AT86RF230 [3] radio Rev A silicon both support broadcast auto-acks, the Rev B silicon "fixes" this standards non-compliance and does not auto-ack broadcast frames.

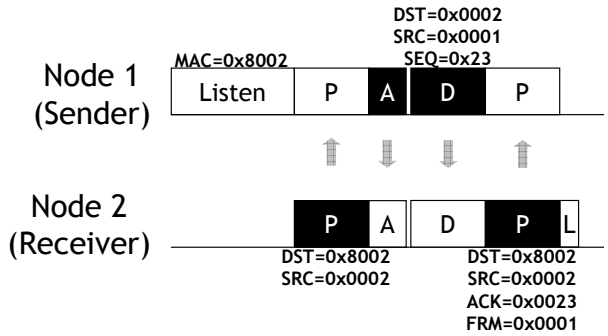


Figure 3. Example of an A-MAC unicast communication showing dynamic address changes and other frame fields.

We avoid the overreacting problem and design a standards-compliant unicast protocol as follows. When sender *S* has pending traffic for receiver *R*, *S* enables hardware address recognition, enables its hardware auto-acks, and *sets its hardware address* to *R*+0x8000.[3] Instead of sending a probe to the broadcast address, receiver *R* sends its probe to destination address *R*+0x8000 and requests an auto-ack. Sender *S* (as well as any other nodes with pending traffic to *R*) respond to the probe. If its probe is acknowledged, *R* remains awake to receive a frame while sender *S* does not succumb to the overreacting problem.

Figure 3, shows a sender (Node 1) with traffic pending for the receiver (Node 2). The sender turns on its radio, sets its hardware address to 0x8002, enables hardware auto-acks, and begins to listen. At some later time, the receiver wakes up and sends a probe with a source address of 0x0002 and a destination address of 0x8002, and requests an acknowledgment. When the sender receives the probe frame, its radio generates an auto-ack. Upon detecting the beginning of the auto-ack, the receiver decides that an auto-ack frame may be incoming, so it continues to listen for at least 352 μs (or possibly less if the data appear garbled) before turning off the radio. If a valid auto-ack is received, the receiver concludes there is pending traffic for it, and it remains awake to receive this data. At the same time, the sender transmits a data frame (after a short random delay comprising the contention window) with a source address of 0x0001, a destination address of 0x0002, and a locally-selected sequence number of 0x23, which is successfully received. The sender does *not* change its radio hardware address for this transmission. The receiver then prepares its next probe which explicitly acknowledges the preceding data frame by source address (0x0001) and sequence number (0x23). The sender turns off auto-acks if it has no further data pending (or repeats this process if it has more data), letting the receiver's second probe go unacknowledged, which allows the receiver to return to sleep after a brief wait.[4]

[3] We reserve addresses with the high-order bit set for such use.

[4] As an optimization, the receiver could acknowledge the sender's data frame, which the sender would use as a "hint" that its transmission was successful (since hardware auto-acks only have sequence numbers but not source or destination addresses). This optimization allows the sender to disable auto-acks prior to the receiver's next probe transmission, eliminating a race condition in which sender has to check the contents of the receiver's second probe to decide whether to acknowledge it. The sender still waits for the receiver's second probe to verify the hint by checking that the second probe's sequence number *and* source address match sender's previous frame. However, this approach is incompatible with broadcasting, as we describe in § 4.4.

4.4 Broadcast Communications

Broadcast is a fundamental operation used by a wide range of higher-layer services and applications. Neighbor discovery, routing updates, and data dissemination all depend on a robust broadcast service for operation. A-MAC's design of the broadcast service is identical to unicast communications with one important difference. A sender *S*, simply disables hardware address recognition altogether but keeps hardware auto-acks enabled. Of course, this requires that auto-acks be used exclusively for responding to probes (e.g., they cannot be used to acknowledge data).

When a higher-layer service needs to send a broadcast, it sets the destination address of the frame to the broadcast address, e.g., 0xFFFF, and submits the frame to A-MAC for delivery. When this frame is ready for transmission, A-MAC disables the hardware address recognition function of the radio for at least as long as the probe period of its neighbors (or the longest of its neighbors' probe periods, if different neighbors are operating with different periods). During this time, *S* will auto-ack *every* probe it receives, regardless of the probe's actual destination address, and proceed to send the data packet like in the unicast case. Although this design does not violate the 802.15.4 standard, it clearly abuses the standard in support of physical and link layer primitives that the standard was not originally designed to provide. Our goal is to show the feasibility of the A-MAC design using existing hardware, not that it is necessarily standards-compliant (although the latter is preferable, to allow it to be tested using off-the-shelf, standards-compliant hardware).

A common case that arises with this design is what to do if, while the broadcaster is listening for neighbors' probes, the broadcaster's own probe timer fires. Should it send the probe and then return to listening or should it forgo the probe and continue listening? The A-MAC design chooses the first approach: a probe is transmitted when the probe timer fires. Doing so avoids a scenario we call the *broadcast standoff* in which two or more nodes that attempt to broadcast a packet wait patiently for the other(s) to first transmit a probe. The A-MAC design avoids this situation, but it raises two further issues. First, the transmit and receive state machines within a node become more complex and cross-coupled. Second, while probing, a broadcaster may miss other neighbors' probes, thereby reducing broadcast reliability.

A potential issue with our design is that if hardware auto-acks are used to acknowledge data frames as well as probes, then a broadcaster would inadvertently acknowledge every single data frame it received, signaling that the data were successfully received when in fact it may not have actually been received. Our unicast implementation avoids this problem by reserving hardware auto-acks exclusively for acknowledging probes. Data frames are acknowledged by including the acknowledgment information in the next probe.

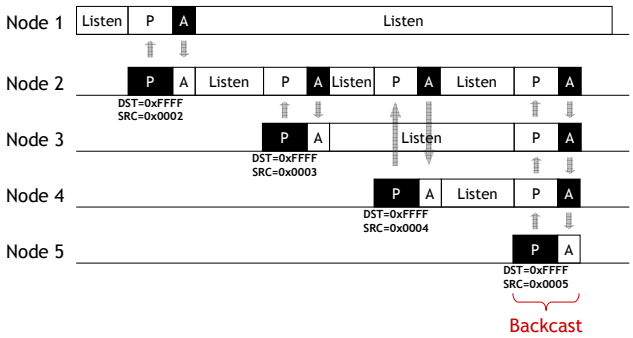


Figure 4. Asynchronous network wakeup with A-MAC. Although Nodes 2, 3, and 4 all ACK Node 5's query probe, the ACK collision is non-destructive, and Node 5 remains awake to communicate.

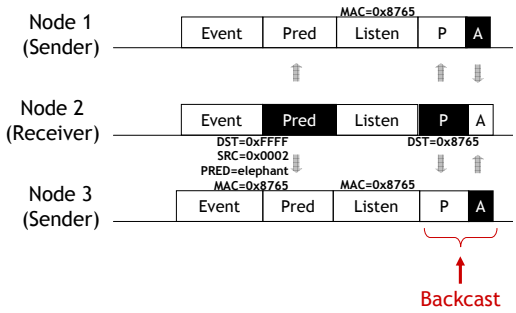


Figure 5. Pollcast implemented using the A-MAC architecture. All nodes observe an "elephant sighting" event. Node 2 wishes to corroborate this observation with its neighbors. It uses backcast to efficiently determine if any neighbor also observed this event.

4.5 Asynchronous Network Wakeup

Waking up a multihop network of duty cycled nodes is a fundamental problem in sensor networks. Applications as diverse as interactive data collection, exceptional event detection, and target tracking require nodes to wake up neighbors or even the entire network in response to an asynchronous event. In many such applications, nodes will remain asleep for long periods of time and so they are likely to lose synchronization. Ideally, the nodes would wake up only in response to external events or user queries, but would otherwise remain asleep. In the case of mobile sensors, nodes may only need to communicate when they have data to upload. However, it is still useful to be able to wake up a mobile node to issue it a command or query.

Several techniques have been proposed for asynchronous network wakeup in a low-power setting including various forms of flooding and dissemination, but these techniques have poor channel efficiency, exhibit logistic-like performance in that they start and end slowly, or are designed with the assumption that nodes are synchronized. As a result, none of these techniques are ideally suited to the low-power, asynchronous network wakeup problem. In this section, we discuss two approaches to designing a backcast-based wakeup service – one that can work with standards-compliant radios and one that cannot. They exhibit high channel efficiency, achieve the lower bound on wakeup time, and do not assume synchronization.

Figure 4 shows the first approach. In this figure, all nodes cease periodic communications like routing beacons and instead operate at a very low duty cycle. The nodes wake up infrequently, perhaps once every ten seconds or each minute, to check if any of their neighbors requires them to stay awake, by sending a probe to the broadcast address. Node 1 initiates an asynchronous network wakeup by configuring its radio to acknowledge all frames. After some time, Node 2 sends a probe. Node 1 auto-acks this probe and Node 2 stays awake. This process repeats with Node 2 waking up Node 3 and Node 4. However, when Node 5 wakes up, all of its neighbors – Nodes 2, 3, and 4 – are already awake and they all simultaneously auto-ack Node 5's probe, which Node 5 correctly decodes as a valid frame and hence remains awake.

Transmitting to the broadcast address with the acknowledgment request bit set does not comply with the 802.15.4 standard (and hence only works with the CC2420). One way to sidestep the issue is to send the probe to a reserved wakeup address rather than the broadcast address. This leads to a wakeup phase, in which a node first performs wakeup for one cycle, and then engages in normal communications. This approach may be preferred since it also disentangles broadcasts and floods from wakeup, and is standards-compliant.

One problem common to both designs is that if a node misses the acknowledgments to its specially-addressed probes during the network wakeup phase, then the node will remain asleep after the wakeup phase since its neighbors will no longer acknowledge specially-addressed probes. This problem, too, can be avoided by using a special wakeup address and increasing the length of the probe frame so that the processor has enough time to: (i) read the address from the radio's receive FIFO while the rest of the probe is being received (i.e., pipelining the read and reception), (ii) check if the address matches the special wakeup address, and (iii) instruct the radio to auto-ack the frame within the tight time window needed to generate a timely auto-ack. We do not explore this idea any further in the context of network wakeup, but we do return to it in a more general form in the context of Pollcast.

4.6 Pollcast Neighborhood Queries

Demirbas et al. recently proposed *pollcast*, a two-phase primitive in which a node broadcasts a poll about the existence of a node-level predicate P and then all nodes for which P holds reply simultaneously [9]. The poller detects one or more positive replies by sampling its radio's Clear Channel Assessment (CCA) signal which indicates whether the received signal strength exceeds a threshold. While pollcast offers a novel approach to quickly calculate predicates, the proposed mechanism has some drawbacks, as their work acknowledges: simultaneous pollcasts within a two-hop neighborhood causes false positives (as would external interference). Selecting the CCA threshold presents a tuning challenge since setting it too low causes false positives but setting it too high causes false negatives.

A-MAC provides a more robust architecture for implementing pollcast by mapping the original two-frame, query/response to a three frame operation. First, a single frame transmission containing the predicate to be evaluated is sent to the broadcast address, received by all neighbors, and evaluated. Next, a short time later, a probe is transmitted to a special address (contained in the first transmission). Finally, the probe is acknowledged by all nodes for which the predicate evaluated true.

Figure 5 illustrates an example in which all nodes observe an event. Node 2 wishes to corroborate an "elephant sighting" event with its neighbors so it transmits a predicate describing the event, including a locally-generate ephemeral identifier. The destination address of the predicate is `0xFFFF` (broadcast), the source address is `0x0002`, the predicate is `'elephant'`, and the ephemeral identifier is `0x8765`. Node 2 then waits for some time to allow Nodes 1 and 3 to receive and evaluate the predicate. Node 2 then sends a probe destined to the ephemeral identifier `0x8765`. Since both Node 1 and Node 3 observed the same event, they both auto-ack the probe, indicating the predicate was true. Note that although the predicate is sent to the broadcast address, it does not need to be automatically acknowledged, so this approach is compatible with 802.15.4 and A-MAC's unicast and broadcast.

A drawback with this approach to pollcast is the need for *two* packet transmissions by the receiver: the first packet sends the predicate and the second packet sends the probe to the ephemeral identifier. Ideally, the predicate could be piggybacked onto to the probe, eliminating the separate predicate transmission and its associated delay. The two challenges with this approach include choosing the destination address of the probe and ensuring that the predicate can be evaluated quickly enough (by the processor) to generate a properly timed auto-ack. One option that we explore is to send the probe to the broadcast address, piggyback the predicate on the probe, and pad the probe with a large payload. This allows a node to detect the beginning of the probe, read and evaluate just the predicate while the rest of the packet is being received, and enable hardware auto-acks before frame reception completes. The pad bytes provide buffer time to evaluate the predicate before the 192 μs auto-ack timer fires.

4.7 Miscellaneous Details

In the current A-MAC implementation, each node chooses its own probe schedule without any local or global coordination. If two nodes pick identical schedules, we rely on capture and contention for short-term progress, and clock drift for long-term desynchronization. An improved implementation could use an explicit desynchronization protocol like DESYNC [8]. When two nodes communicate, the sender caches the receiver's probe period and phase. This allows the sender to minimize its radio on-time during subsequent communications by listening just before the expected probe transmission. The cache holds four entries and uses an LRU eviction policy. An alternate policy might consider usage frequency. Queued packets are transmitted round-robin for fairness, but this can result in head-of-line blocking. An EDF policy that orders pending packets by their receivers' probe times may be a better option, especially since current hardware can only auto-ack one receiver's probes at a time.

5 Backcast Evaluation

Backcast is a critical primitive upon which A-MAC rests, so we evaluate its reliability, efficiency, and performance under a range of conditions including carefully-controlled laboratory settings and more realistic indoor settings. Our results show that backcast works on two different radios from two different vendors, has a narrow range of failure cases, provides high energy- and channel-efficiency, and provides a strong foundation upon which to build the remaining link layer services.

5.1 Methodology

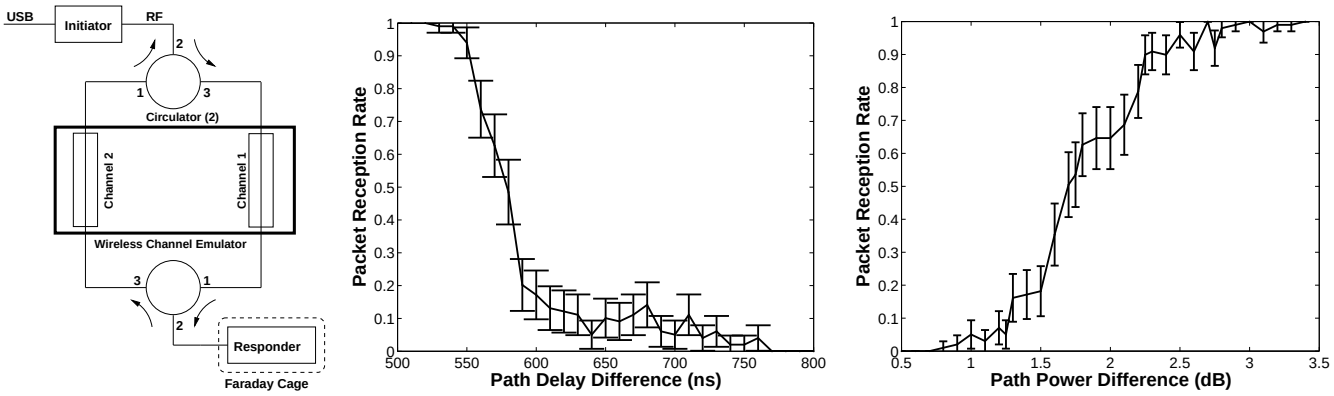
We use the Moteiv Tmote (Telos B) [31], Berkeley Epic [14], and Crossbow Iris [7] motes for these experiments. We find that the backcast performance of both radios is similar, so we only report detailed results for the CC2420 radio. In the experiments that follow, signal strength is measured by the radio over the first eight symbols of an acknowledgment (ACK) frame and reported as the received signal strength indicator (RSSI) in dBm. Signal quality (LQI) is also measured by the radio over the first eight symbols and is reported as a 7-bit unsigned integer that can be viewed as the average correlation value or chip error rate (values near 100 indicate an excellent link).

5.2 ACK Reception Robustness

We first explore how delay differences in the path length affect ACK reception rate. Figure 6(a) presents the setup for this experiment. Two nodes, an *initiator* and a *responder* (both Tmotes) are connected to each other through a pair of circulators and a wireless channel emulator. A circulator is essentially an RF splitter that provides a low-loss RF path between some terminals (1-to-2, 2-to-3, and 3-to-1) but a very high-loss path between other terminals (1-to-3, 2-to-1, and 3-to-2). Circulators are used to split a single bi-directional RF path into two unidirectional paths. We use the D3C2060 circulator from DiTom Microwave. A wireless channel emulator allows a complex RF environment, including attenuation, delay, fading, Doppler shift, and multipath, to be evaluated in a laboratory setting. We use the Spirent SR5500 wireless channel emulator in these experiments. The SR5500 allows each channel to be composed of several independent paths, each with its own delay and attenuation.

5.2.1 Effect of Path Delay Differences

To evaluate the effect of path delay difference on destructive intersymbol interference, the ACK channel from the responder to the initiator (Channel 2) is split into two equal loss paths inside the channel emulator. The delay in the second path is swept from 0 to 1 μs in 10 ns steps. For each delay step, the initiator transmits 100 packets to the hardware broadcast address, at 125 ms intervals, and logs the RSSI, LQI, and sequence number of the resulting acknowledgments. The results are shown in Figure 6(b) and indicate intersymbol interference becomes destructive between 500 and 600 ns, as expected. Note that a delay of 500 ns corresponds to a path delay *difference* of 150 m. Such path delay *differences* are rare in low-power wireless networks; links are rarely more than tens of meters, so such significant delay differences would result in different received signal strength values as well (unless transmission power control is used).



(a) Experimental Setup (b) Intersymbol Interference (c) Power Capture

Figure 6. Figure (a) shows the experimental setup. Figure (b) shows the onset of destructive inter-symbol interference. Packet reception rate falls sharply as the delay difference in two paths exceeds 0.5 μs. Figure (c) shows the effect of power capture. When two frames collide, the first frame to arrive will be decoded correctly if its receive power is 3 dB higher than the second frame.

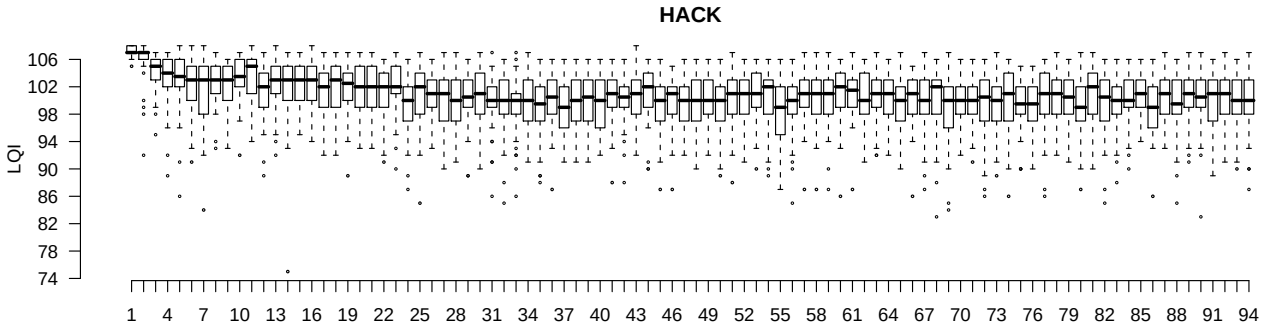


Figure 7. The effect on LQI as the number of concurrent ACKs increases from 0 to 94 in a typical indoor deployment setting. The median value of LQI falls quickly for the first six nodes and then falls slowly. Beyond approximately 30 nodes, the LQI values stabilize at approximately 100. The data suggest that even in the presence of a large number of ACK collisions, the receiver can successfully decode the ACK frame. Note the y-axis ranges from 74 to 106.

5.2.2 Effect of Path Loss

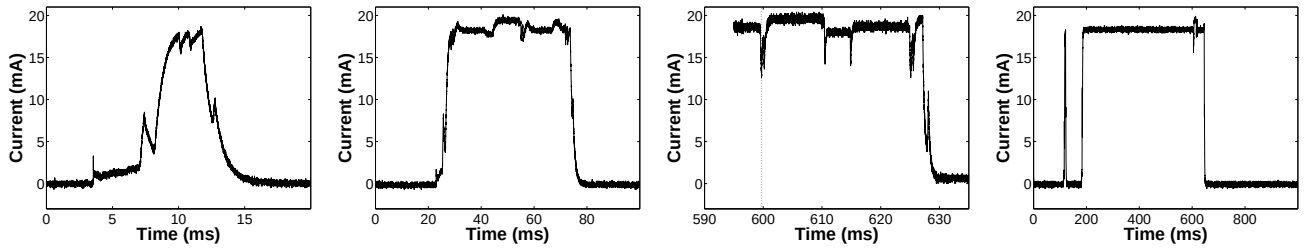
Power capture occurs when the received signal from one node is sufficiently stronger than the sum of the received signals from all other nodes [1]. To explore the effect of power capture on backcast performance, the second path component in Channel 2 is delayed by 8,00 ns (1/2 of the 802.15.4 symbol time). This base configuration ensures intersymbol interference and, assuming equal path loss, results in destructive interference and complete packet loss.

However, by adjusting the attenuation for the second path, from 0 to 3.5 dB, in 0.1 dB steps, the effect of power capture becomes evident. The initiator receives the superposition of two (identical) frames, delayed by 8,000 ns, over a range of SINR values. The results show that when the first frame arrives with 3 dB or higher power, it will be decoded consistently by the radio. The data also show a fairly linear transition region between approximately 1 dB and 2.5 dB. These figures establishes that power capture dominates (and explains) the backcast phenomenon when the strongest ACK's power exceeds the sum of the remaining ACKs by more than approximately 3 dB.

5.2.3 Large-Scale Performance

We now explore how backcast performs in a more realistic setting – a university testbed. The testbed consists of Telos B nodes and it is located in an office building with a typical RF environment. For this experiment, 94 nodes within radio range of an initiator are programmed to automatically acknowledge all probes. The 94 nodes are turned on, one after the other, and remain on for the rest of the experiment. After each node is turned on, 500 frames are transmitted at 125 ms intervals. This procedure generates a gradual increase in the number of auto-ack frame collisions. The LQI statistics are shown in Figure 7. The PRR is 100%.

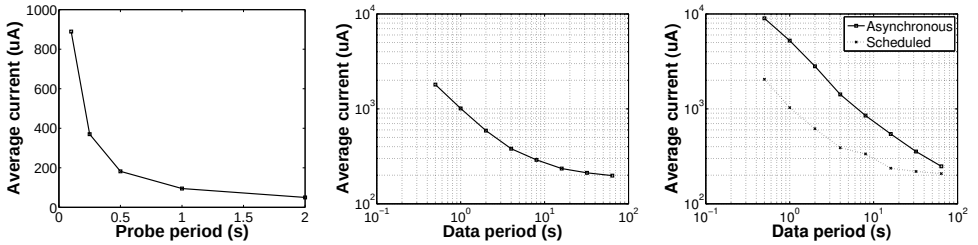
The results show that the median value of LQI falls quickly for the first six nodes and then falls slowly. Beyond approximately 30 nodes, the LQI values stabilize at approximately 100, although there are outliers. The data suggest that even in the presence of a large number of ACK collisions, the receiver can successfully decode ACK frames, even when no single ACK frame's power dominates. The ACK reception rate is nominally 100% (ACKs are received consistently, independent of the number of concurrent transmissions).



(a) Probe (b) Receive (Len=127 bytes) (c) Transmit (Len=127 bytes) (d) Idle (500 ms wait)

Primitive	Cost (μJ)
Probe	253
TX only	1578
RX only	2670
CCA Check	194

(e) Primitive energy costs



(f) Probe (g) Receive (h) Transmit

Figure 8. Link Power Model. Figures (a)-(c) show the Telos B mote's instantaneous current draw for representative asynchronous link primitives. Figure (d) shows the current draw when only listening (from 200 ms to 800 ms). Figure (e) shows the cost of each link primitive. Figures (f), (g), and (h) show the average current for probing, receiving, and transmitting, respectively, as a function of the probe period T_{probe} (f), and data period (g) and (h) with $T_{probe} = 0.5$ s.

The data suggest that both constructive and destructive interference of the carrier signal occur. This result is not surprising since the carrier signals are neither synchronized in phase nor frequency across these 94 nodes. Rather, they are generated locally by each node from a free-running crystal oscillator. The statistical superposition of an increasing number of signals does not lead to destructive interference, making backcast a robust synchronization primitive.

5.3 Energy Microbenchmarks

A-MAC services are built by combining a small set of link primitives including *probe*, *receive*, *transmit*, and *idle* (listening for a probe). Figures 8(a)-8(d) show the traces of these primitives as well as their energy costs. The vertical line in Figure 8(c) indicates the point at which the sender's radio signals that the probe's start-of-frame delimiter (SFD) event has occurred. These data are collected by capturing the voltage drop across a 10 Ω resistor in series with a 3 V power supply using a Tektronix TDS3014 digital storage oscilloscope. Figure 8(e) summarizes the energy cost of each basic primitive. In all cases, other than probes, we use the 802.15.4 link MTU frame size (127 byte payload).

Figure 8(f) shows how the average current due to probing cost scales with the probe period. Figure 8(g) shows how the receive cost scales with data rate. Figure 8(h) shows how the transmit cost scales with data rate for both *asynchronous* communications (when the sender does not know the receiver's probe schedule) and *synchronous* communications (when the sender knows the receiver's probe schedule). Figure 8 shows that A-MAC's link *primitives* are *more* expensive than in an optimized, commercial-grade LPL implementation approach [22], but under the critical assumption of no external interference. Section 5.4 explores what happens when this assumption is false.

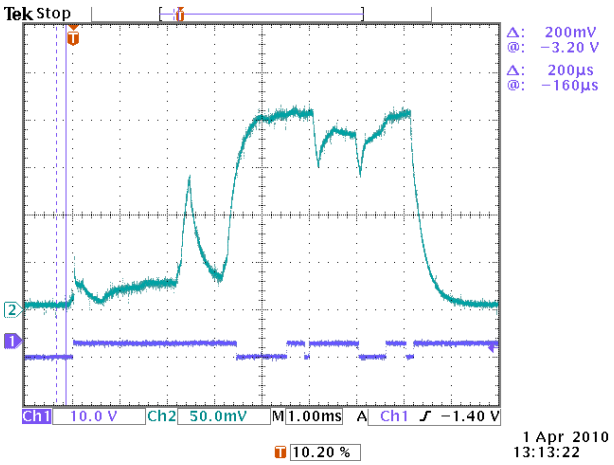
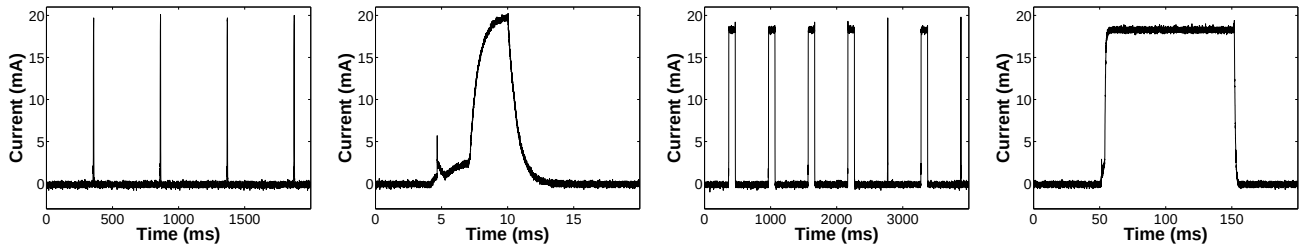


Figure 9. A-MAC probe states and their energy consumption. Transitions of the lower line indicate state changes. Total consumption is 263.56 μJ. Breakdown: (i) start (40.98 μJ); (ii) load probe (60.60 μJ); (iii) load done (22.7 μJ); (iv) probe alarm fired (re)send (6.31 μJ); (v) strobe and transmit (55.71 μJ); (vi) start ACK timer (29.86 μJ); (vii) send done ACK timeout (25.71 μJ); (viii) radio stop (8.78 μJ); and (ix) radio stopped (12.91 μJ).

Figure 9 shows the A-MAC probe's power draw and associated state transitions. In our implementation, the *instrumented* probe consumes approximately 263 μJ. With radio hardware support, the following states would be eliminated: load probe, load done, probe alarm fired (re)send, and send done ACK timeout. This would save about 115 μJ and reduce the cost of the probe to approximately 148 μJ – less than three times the cost of an optimized LPL check [22].



(a) LPL sampling (no interference) (b) LPL sample detail (c) LPL sampling (w/ interference) (d) LPL overhearing detail

Figure 10. LPL preamble sampling techniques leave receivers susceptible to noisy wireless environments, such as those caused by 802.11 interference. Figures (a) and (b) show the macroscopic and microscopic behavior of the TinyOS 2.1 sampling algorithm when the channel is clear: the receiver immediately returns to sleep. Figures (c) and (d) show the macroscopic and microscopic behavior while a file transfer is in progress using a nearby 802.11 access point. Of the seven channel samples visible in this trace, five are unnecessarily lengthened due to channel noise.

5.4 Robustness to External Interference

A basic problem with LPL and LPP systems that employ RSSI to detect the presence of incoming traffic is that they suffer from many sources of false alarms including interference, overhearing, and collisions. Recent research has demonstrated the cost of external interference on the effective duty cycle of LPL protocols. The results show that significant differences can exist between the expected and actual duty cycles [4, 16]. We repeat similar experiments to quantify the effects of interference on MAC layer operation and energy consumption.

Table 1 shows the results of an experiment in which we measure the receiver's idle listening current in an office environment using three different synchronization schemes, TinyOS 2.1 LPL, RI-MAC LPP, and A-MAC LPP, under two different interference workloads (with and without a nearby 802.11 file transfer in progress). Although the TinyOS LPL technique performs better under ideal conditions, it degrades dramatically in the presence of interference, increasing average current draw by a factor of 17.3 compared to the idle listening case. The RI-MAC LPP technique performs even worse, exhibiting an increase in idle current by a factor of 54.7. A-MAC, in contrast, exhibits a nearly negligible 1.12× increase in current draw, demonstrating the backcast's resilience to false positives. For completeness, we also include reported figures for Hui's MAC [22] which uses 54 μJ per sample (54 μJ / (3 V × 0.5 s) = 36 μA). We estimate the power draw is doubled in the presence of external interference (and equals the reported overhearing cost).

Figure 10 illustrates in detail how the preamble sampling techniques used in LPL protocols leave receivers susceptible to noisy wireless environments, such as those caused by 802.11 interference during beaconing, file transfers, or audio/video streaming. Figure 10(a) shows the current draw over time when the channel is clear and Figure 10(b) shows the detailed current draw of one channel sample. Figure 10(c) shows the current draw of the same system while a file transfer is in progress using a nearby 802.11 access point. Of the seven channel samples in this trace, five are of extended length due to channel noise. Figure 10(d) shows the details of an extended sample.

Primitive Operation	w/o 802.11 interference	w/ 802.11 interference	Increase in Current
TinyOS LPL	175 μA	3,030 μA	17.3×
RI-MAC LPP	383 μA	12,576 μA	54.7×
A-MAC LPP	206 μA	230 μA	1.12×
Hui LPL	36 μA†	72 μA‡	2.0×‡

Table 1. The effect of interference on idle listening current. The average current draw of three different synchronization schemes under no-load conditions and a 500 ms check/probe interval. Results are the average of five samples, each one minute long. Although the LPL exhibits the lowest power under ideal conditions, both the TinyOS LPL and RI-MAC LPP exhibit dramatic power increases under interference while A-MAC's LPP mechanism shows a relatively negligible increase which shows A-MAC's low-power probing is resilient to false positives. Hui's LPL reported figure (†) is included for comparison and our *estimate* of its interference current is noted (‡).

The extended channel sample in Figure 10(d), termed "delay-after-receive-check," improves communications reliability. Shorter delays work under ideal circumstances but in noisy or congested environments, they lead to failed communications [27]. The 100 ms delay-after-receive-check, whenever channel energy is detected, substantially reduces LPL delivery failure (a false negative). A-MAC is largely immune to this problem because it uses an explicit probe rather than an implicit channel energy signal. We hypothesize the Hui's MAC is also more robust than the default TinyOS LPL due to its use of an explicit chirp, but lacking access to it, we could not verify this thesis. An open question is to further explore the complex relationship between duty cycles, delivery ratios, false positives, false negatives, and latency as channel sample time is adjusted after a "busy" channel assessment. These results show the challenge of predicting network lifetime based only on a model of the data workload, but without a good model of the environmental factors. Although overhearing and interference are well-known problems, these results suggest they deserve further study.

MAC	No. of Senders	Packet Delivery Ratio		
		Avg	Min	Max
RI-MAC	1	99.9%	—	—
	2	97.5%	97.3%	97.7%
	3	95.6%	95.0%	96.8%
	4	90.7%	90.3%	90.9%
A-MAC	1	99.9%	—	—
	2	99.3%	98.2%	100%
	3	99.3%	98.3%	99.5%
	4	98.5%	96.7%	99.5%

Table 2. Packet delivery ratios for 1 through 4 distinct senders transmitting to a single receiver. The packet interval on each sender is uniformly drawn from 0.5 to 1.5 s, so on average, it is 1 pkt/s from each sender. The receiver uses a T_{probe} =1 s. Senders attempt to send on each probe to stress the contention algorithms, for 1,000 packets. The largest difference between the maximum and minimum success rates for A-MAC is 2.8%, showing that A-MAC provides fairness under modest contention.

WL (#)	Tx:Rx (ratio)	Probe (ms)	Rate (pkt/s)	Pkts (#)	Time (s)	PDR (%)
3	1:1	512	48.4	8,010	164	100
3	1:1	128	71.9	8,056	112	100
3	2:2	128	78.7	5,982	76	99.2
			78.9	5,365	68	99.4
1	2:2	128	27.8	1,895	68	98.9
			58.9	3,535	60	99.9
6	2:2	128	74.7	5,975	80	99.2
			77.0	6,007	78	99.3
6	3:3	128	88.9	4,912	60	99.3
			80.7	4,834	60	99.3
			85.0	5,098	60	99.4
1	3:3	128	22.1	1,324	60	97.0
			30.8	1,845	60	98.3
			36.5	2,219	59	99.3

Table 3. A-MAC performance with multiple parallel unicast flows. Throughput and packet delivery ratio improve with additional channels. Even without the multichannel optimization, A-MAC can sustain multiple, parallel unicast flows located in the same collision domain.

6 Macrobenchmark Evaluation

Our evaluation thus far has focused on microbenchmarks comparing the time, energy, false positives, and false negatives of TinyOS LPL, RI-MAC, and A-MAC primitives. We now explore several macrobenchmarks to explore how low-level power and performance improvements translate to high-level performance for several link layer services. For these experiments, we use the standard TinyOS 2.1 distribution's default LPL MAC and the RI-MAC [33] source code, which was provided by its authors.

6.1 Multiple Contending Unicast Flows

It is well known that receiver-initiated MAC schemes handle contending flows and hidden terminals much better than low-power, sender-initiated ones [6, 17, 33]. We now evaluate how well A-MAC handles multiple contending flows. Table 2 shows between one and four senders contending to transmit to a single receiver for both RI-MAC and A-MAC. In this experiment, the receiver sends a probe, the senders may all auto-ack the probe concurrently, and then they contend for the channel. The receiver resends a probe after either each successful transmission or after receiving an auto-ack, but no data. The receiver sends up to a total of five probes before stopping. Each probe doubles the size of the contention window. The base contention window size is 20 jiffies (610 μs). Each node transmits 1,000 packets.

The data in Table 2 show that A-MAC matches RI-MAC's performance for a single transmitter but performs better than RI-MAC when additional senders begins to contend. The largest min-max difference is 2.8%, showing that even when four nodes are contending, A-MAC is fair.

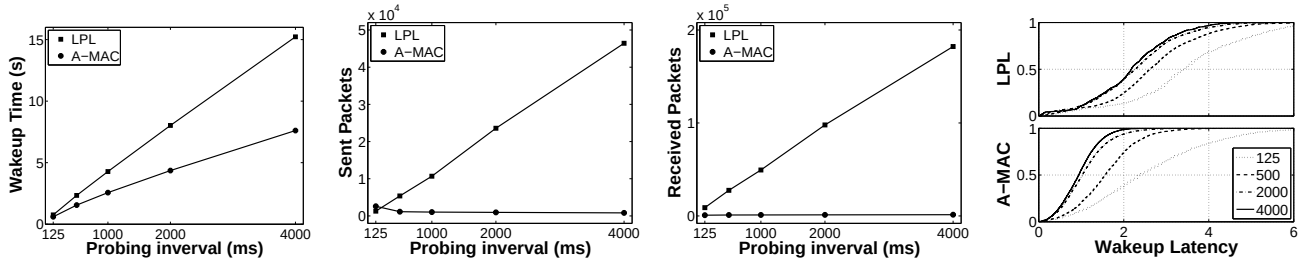
6.2 Multiple Parallel Unicast Flows

We now evaluate how well A-MAC supports multiple concurrent flows between distinct pairs of senders and receivers that are all located in a single collision domain. This experiment tests A-MAC's multichannel optimization in which the probe and acknowledgment are transmitted on a shared control channel, but data transfer may occur on a different channel as stipulated in the probe. Table 3 shows A-MAC throughput and packet delivery ratio as a function of the number of different whitelisted channels that are available for use, the number of sender:receiver pairs transferring data concurrently, and the receivers' probe interval.

WL refers to the number of channels in the whitelist where WL=1 means all traffic happens on the control channel (25), WL=3 means channels 15, 21, and 24 are in the whitelist, and WL=6 means channels 11, 15, 20, 21, 24, and 26 are in the whitelist. Tx:Rx identifies the number of independent transmitter:receiver pairs concurrently transmitting. The Probe field specifies the probe interval. The throughput (Pkt/s), data size (#Pkts), transfer time (Time), and packet delivery ratio (PDR) are shown. The data show that throughput improves significantly with additional channels while the (already high) packet delivery ratio improves slightly with additional channels.

6.3 Asynchronous Network Wakeup

A network wakeup is a special case of flooding or dissemination in which the goal is to ensure that every node in the network receives a wakeup message. Prior work has shown that LPL-based flooding techniques can cause significant contention and can use the radio channel (a scarce resource) over an extended period of time to complete a flood [26]. Figure 11 explores how well the TinyOS 2.1 LPL and A-MAC wakeup implementations of asynchronous network wakeup compare. In this experiment, the LPL wakeup algorithm is a simple flood: the source of the flood repeatedly resends a wakeup packet for slightly longer than the sleep interval. Every node that receives the packet also retransmits it, after it detects a clear channel. The A-MAC flooding algorithm is a recursive broadcast without subsequent data packet transmissions. Since network wakeup is a special case of flood, this experiment also establishes A-MAC's broadcast performance.



(a) Wakeup Latency (b) Sent Packets (c) Received Packets (d) Normalized Wakeup Latency

Figure 11. Wakeup latency for LPL and A-MAC for several sleep periods (0.125 s, 0.5 s, 1 s, 2 s, 4 s). Figure (a) shows A-MAC wakes up the network in about 38% less time than LPL. Figures (b) and (c) show that far few packet transmissions and receptions are required by A-MAC. Figure (d) shows the CDF of wakeup latencies normalized by the probe interval. A-MAC wakes up the network faster, uses far fewer packets, and is far more channel efficient than LPL.

	LPL	A-MAC
Average Duty Cycle	6.36%	4.44%
Average Packet Delivery Ratio	95.1%	99.7%
Average Hop Count	7.34	4.85
Maximum Hop Count	14	13

Table 4. CTP performance over LPL and A-MAC. A-MAC offers higher packet delivery ratio, lower duty cycles, and lower average hop count. A-MAC performance meets or exceeds the widely-used TinyOS LPL link layer.

Figure 11 shows the wakeup times of 59 nodes in a multihop testbed across a range of sampling/probing intervals. Figure 11(a) shows that A-MAC wakes up the network about 38% faster than the default TinyOS LPL. Figures 11(b)-(c) show A-MAC transmits far fewer packets to do so, hence exhibiting dramatically better channel efficiency. Figure 11(d) shows the CDF of wakeup latencies. The better relative performance of longer probe intervals seems counter-intuitive, but it occurs because there is a lower probability of a node transmitting a probe when a neighbor is otherwise occupied as the probe interval length is increased.

6.4 Collection Tree Protocol Performance

We next explore how well the Collection Tree Protocol (CTP) [18] performs over the A-MAC link layer. CTP is the default collection routing protocol in TinyOS and it represents a canonical link layer client. Since A-MAC exports the standard TinyOS `ActiveMessage` layer, running CTP over A-MAC is largely a matter of changing configuration wirings. In the following two experiments, a network of 59 nodes run CTP and are programmed such that each node generates one data packet every 60 seconds. An experiment runs for one hour and the default TinyOS LPL implementation and A-MAC are tested in different experiments. Table 4 summarizes the results of the two experiments, run sequentially, and repeated twice, for a total of three trials.

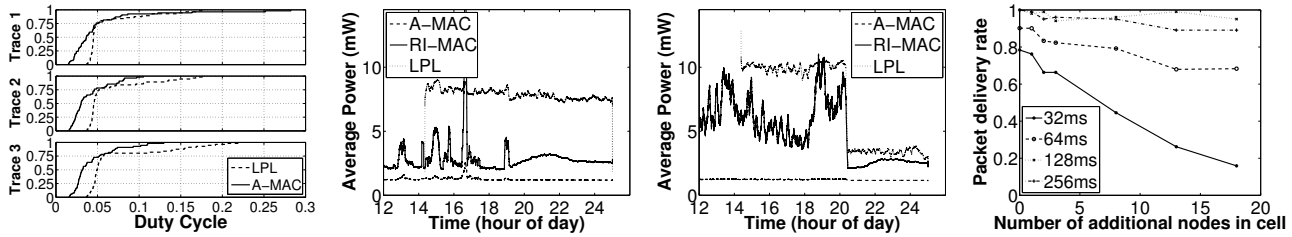
Figure 12(a) shows the CDF of per-node duty cycles when running CTP over LPL and A-MAC. The A-MAC and LPL experiments are run sequentially and the experimental pair is repeated three times. The A-MAC and LPL CDFs are largely self-correlated across the runs. A substantial fraction ($\approx 60\%$) of the nodes running A-MAC exhibit a strictly lower duty cycle than *any* LPL nodes. Every node running A-MAC exhibits a lower duty cycle, on a percentile basis, than the corresponding LPL node, except for 5% of the nodes in Trace 1. The results show that A-MAC satisfies realistic workloads, achieves lower duty cycles, offers higher packet delivery ratios, and provides greater channel efficiency.

6.5 Interference Vulnerability

In Section 5.4, we explored the effect of interference on the idle listening average current of TinyOS LPL, RI-MAC LPP, and A-MAC LPP in the presence and absence of nearby 802.11 file transfers. We now extend these experiments to explore the power draw of these three MAC layers on two different channels (802.15.4 channels 18 and 26) in a typical computer science department over the course of approximately 12 hours (noon to midnight). Figure 12(b) shows the average power draw on channel 18 and Figure 12(c) shows the average power on channel 26, all for a fixed probe interval. In all cases, A-MAC exhibits both the lowest and most stable power draw while LPL exhibits the highest power draw and RI-MAC exhibits the most volatile power draw.

6.6 Effect of Density on Packet Delivery

One major drawback of receiver-initiated protocols is that because of their periodic probing, their channel utilization scales with node density and probe frequency rather than strictly with traffic. We now explore the effect of node density and probe frequency on packet delivery rate. In this experiment, a single sender transmits 100 packets to a single receiver with an inter-packet interval of 500 ms. We vary the number of additional nodes that are in the same collision domain (between 0, 1, 2, 3, 8, 13, and 18) who simply transmit probes with varying probe periods (32 ms, 64 ms, 128 ms, and 256 ms). Probes are transmitted using clear channel assessment (CCA) enabled. Figure 12(d) shows the results of running these 28 experiments. We see that packet delivery ratio drops with both increasing density and decreasing probe intervals, as expected. This figure illustrates a major weakness of receiver-initiated protocols in general and A-MAC in particular. However, we do not coordinate the probe phases in this experiment, so the delivery ratios may improve if the probe times were better scheduled or evenly distributed, perhaps using a protocol like DESYNC [8].



(a) CDF of CTP Duty Cycles (b) Interference Effects (Ch. 18) (c) Interference Effects (Ch. 26) (d) Effect of Density

Figure 12. Macrobenchmarks and statistics for (a) collection routing duty cycles, (b) and (c) power draw vs. extant environmental interference (co-channel communications), and (d) effect of neighborhood on packet delivery ratios.

7 Discussion

In this section, we outline how future radio hardware could improve A-MAC performance and discuss some of the limitations that are fundamental to this design.

7.1 Future Hardware Support

A handful of radio enhancements could improve the performance and energy efficiency of both the backcast primitive and the link layer services multiplexed above it. The main bottlenecks in our current design occur from the limited processor-radio bandwidth. Since backcast-based communications requires multiple loads and unloads of the transmit and receive FIFOs, respectively, they are often the critical path operations that occur over a slow serial bus. If either hardware support for backcast existed inside the radio, or A-MAC was implemented in a processor with a memory-mapped radio [24, 35], the loads and unloads could be made more efficient. Based on Figure 9, we estimate that simple hardware support would reduce the probe energy cost from 263 μJ to 148 μJ, reducing idle listening power by 40%.

Under the current unicast design, a sender S sets it local address to R+`0x8000`, where R is the receiver's hardware address. As a result, S cannot concurrently acknowledge probes from a different receiver, R', for which it also has pending traffic. Richer support for hardware address recognition in the radio would allow a sender to multiplex listening for a probe. For example, a radio could filter for multiple source or destination addresses in parallel. Some radios, like the TI CC2520 [36], can already filter frames on up to twelve different source addresses but these frames must be sent to a unicast destination address (meaning some of the approaches outlined in this paper will not benefit). More flexible address recognition and auto-ack support would greatly reduce the processor burden and offer better efficiency than modern LPL protocols.

7.2 Limitations

There are two fundamental limitations to A-MAC. First, since A-MAC is a receiver-initiated protocol, the channel must be probed periodically. This makes A-MAC fundamentally less channel efficient under no-data conditions than sender-initiated protocols that listen quietly when no traffic is present. In other words, A-MAC channel usage scales with neighbor density and not necessarily with traffic. Hence, A-MAC may be incompatible with networks that have high node density, short communication latency, or low-probability of detection requirements.

The first two issues are partly addressed by using a different channel for the probes and auto-acks than for the actual data transmissions, since the initial probes can be sent on a control or pilot channel. The latter issue is more severe: A-MAC is fundamentally at odds with stealthy networks since nodes cannot just listen quietly.

The second fundamental limitation with this approach is that A-MAC's primitive operation, a channel *probe*, is inherently more expensive than the channel *sample* primitive in sender-initiated protocols. Sending a probe frame and listening for an acknowledgment will always require more time than sampling the channel. However, the benefits of using backcast, namely its fixed energy cost, low false alarm rate, and efficient multiplexing ability, underscore a familiar theme in systems and networking research: optimal solutions that work well over a narrow range often perform more poorly over the diversity of workloads observed in practice.

8 Conclusion

Optimizing performance for a narrow range of operating conditions or isolated performance metrics is often relatively straightforward. For example, designing protocols that achieve low power or high throughput under ideal conditions is easy. It is more difficult to find general solutions that work well across a broad spectrum of workloads and externalities.

In this paper, we present A-MAC, the first receiver-initiated link layer that concurrently supports unicast, broadcast, wakeup, and pollcast services. Despite its generality, A-MAC achieves high channel efficiency, is resilient to a wide range of external interference and noise, offers high packet delivery ratios across a wide range of workloads including n-to-1 incast and multiple parallel flows, offers lower power than prior receiver-initiated protocols, and leverages multi-channel optimizations. We achieve these results using existing radios in novel ways, but we note that performance would improve with even a modicum of hardware support.

This work establishes that there is still plenty of room at the MAC layer to improve duty cycles, achieve predictable operation, offer high channel efficiency, and provide better support for bursty workloads. This work paves the way for new research in the design of radio hardware, MAC sublayer primitives, MAC-layer services, and performance studies to assess the utility and performance of this approach for emerging needs, like the 802.15.4(e) working group's search for a low-power, channel efficient, asynchronous link layer.

9 Acknowledgments

Special thanks to Răzvan Musăloiu-E. for help in developing and evaluating a preliminary A-MAC prototype, Steven Lanszisera for help with modulation schemes and radio receiver architectures, Rabin Patra for help with wireless channel emulation, Intel Labs Berkeley for allowing us access to their laboratory, the anonymous reviewers for their insightful feedback, and Alberto Cerpa for shepherding this paper. This material is based upon work partially supported by the National Science Foundation under grants #0964120, #0435454, #0454432, #0546648, #0834470, and #0627611, as well as a Microsoft Research Graduate Fellowship.

10 References

[1] J. Arnbak and W. van Blitterswijk. Capacity of slotted ALOHA in rayleigh-fading channels. *IEEE Journal on Selected Areas in Communications*, 5(2):261–269, Feb 1987.

[2] Association of Radio Industries and Businesses (ARIB). ARIB STD-T67: Telemeter, Telecontrol, and Data Transmission Radio Equipment for Specified Low-Power Radio Station, Version 1.1. ARIB STD-T67, 2005.

[3] Atmel. AT86RF230. Available at: `http://www.atmel.com/dyn/products/product_card.asp?part_id=3941`.

[4] C. A. Boano, T. Voigt, N. Tsiftes, L. Mottola, K. Roemer, and M. A. Zuniga. Making sensornet MAC protocols robust against interference. In *EWSN'10: Proceedings of the 7th European Conference on Wireless Sensor Networks*, Feb. 2010.

[5] M. Buettner, G. Yee, E. Anderson, and R. Han. X-MAC: A short preamble MAC protocol for duty-cycled wireless sensor networks. In *Sensys'06: Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, Nov. 2006.

[6] D. P. Connors and G. J. Pottie. Response Initiated Multiple Access (RIMA), a Medium Access Control protocol for satellite channels. In *GLOBECOM'00: Proceedings of the IEEE Global Telecommunications Conference*, 2000.

[7] Crossbow. Wireless Module - IRIS 2.4GHz. Available at: `http://www.xbow.com/Products/productdetails.aspx?sid=264`.

[8] J. Degesys, I. Rose, A. Patel, and R. Nagpal. DESYNC: self-organizing desynchronization and tdma on wireless sensor networks. In *IPSN '07: Proceedings of the 6th International Conference on Information Processing in Sensor Networks*, pages 11–20, Apr. 2007.

[9] M. Demirbas, O. Soysal, and M. Hussain. A singlehop collaborative feedback primitive for wireless sensor networks. In *INFOCOM'08: Proceedings of the 27th Conference on Computer Communications*, Apr. 2008.

[10] A. Dutta, D. Saha, D. Grunwald, and D. Sicker. SMACK: a SMart ACKnowledgment scheme for broadcast messages in wireless networks. In *SIGCOMM'09: Proceedings of the ACM Conference on Data Communication*, pages 15–26, Aug. 2009.

[11] P. Dutta and D. Culler. Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications. In *SenSys '08: Proceedings of the 6th International Conference on Embedded Networked Sensor Systems*, pages 71–84, Nov. 2008.

[12] P. Dutta, D. Culler, and S. Shenker. Procrastination Might Lead to a Longer and More Useful Life. In *HotNets-VI: Proceedings of the 6th Workshop on Hot Topics in Networks*, Nov. 2007.

[13] P. Dutta, R. Musăloiu-E., I. Stoica, and A. Terzis. Wireless ACK collisions not considered harmful. In *HotNets-VII: Proceedings of the 7th Workshop on Hot Topics in Networks*, Oct. 2008.

[14] P. Dutta, J. Taneja, J. Jeong, X. Jiang, and D. E. Culler. A building block approach to sensornet systems. In *SenSys'08: Proceedings of the 6th International Conference on Embedded Networked Sensor Systems*, pages 267–280, Nov. 2008.

[15] A. El-Hoiydi and J.-D. Decotignie. Low power downlink MAC protocols for infrastructure wireless sensor networks. *Mobile Networks and Applications*, 10(5):675–690, 2005.

[16] R. Fonseca, P. Dutta, P. Levis, and I. Stoica. Quanto: Tracking energy in networked embedded systems. In *OSDI'08: Proceedings of the 8th USENIX Symposium on Operating Systems Design and Implementation*, pages 323–338, Dec. 2008.

[17] J. J. Garcia-Luna-Aceves and A. Tzamaloukas. Reversing the collision-avoidance handshake in wireless networks. In *MobiCom '99: Proceedings of the 5th International Conference on Mobile Computing and Networking*, pages 120–131, Aug. 1999.

[18] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection Tree Protocol. In *SenSys'09: Proceedings of the 7th International Conference on Embedded Networked Sensor Systems*, pages 1–14, Nov. 2009.

[19] S. A. Gronomeyer and A. L. McBride. MSK and offset QPSK modulation. *IEEE Transactions on Communications*, 24(8), 1976.

[20] J. Hill and D. Culler. Mica: A Wireless Platform for Deeply Embedded Networks. *IEEE Micro*, 22(6):12–24, Nov. 2002.

[21] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for network sensors. In *ASPLOS-IX: Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, Nov. 2000.

[22] J. W. Hui and D. E. Culler. IP is dead, long live IP for wireless sensor networks. In *SenSys'08: Proceedings of the 6th International Conference on Embedded Networked Sensor Systems*, pages 15–28, Nov. 2008.

[23] IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks. Specific requirements – Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), May 2003.

[24] Jennic. Wireless Microcontrollers: JN5121 and JN513x. Available at `http://www.jennic.com/products/`, 2007.

[25] C.-J. M. Liang, R. Musaloiu-E, and A. Terzis. Typhoon: A reliable data dissemination protocol for wireless sensor networks. In *EWSN'08: Proccedings of the 5th European Conference on Sensor Networks*, Jan. 2008.

[26] J. Lu and K. Whitehouse. Flash flooding: Exploiting the capture effect for rapid flooding in wireless sensor networks. In *INFOCOM'09: Proceedings of the 28th Conference on Computer Communications*, Apr. 2009.

[27] D. Moss. Personal communications, 2010.

[28] R. Musăloiu-E., C.-J. Liang, and A. Terzis. Koala: Ultra-low power data retrieval in wireless sensor networks. In *IPSN'08: Proceedings of the 7th Intl. Conference on Information Processing in Sensor Networks*, Apr. 2008.

[29] S. Pasupathy. Minimum Shift Keying: A spectrally efficient modulation. *IEEE Communications Magazine*, 1979.

[30] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Sensys'04: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, Nov. 2004.

[31] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. In *IPSN'05: Proceedings of the 4th International Conference on Information Processing in Sensor Networks*, Apr. 2005.

[32] Y. Sun, O. Gurewitz, S. Du, L. Tang, and D. B. Johnson. ADB: an efficient multihop broadcast protocol based on asynchronous duty-cycling in wireless sensor networks. In *Sensys'09: Proceedings of the 7th International Conference on Embedded Networked Sensor Systems*, pages 43–56, Nov. 2009.

[33] 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 *SenSys'08: Proceedings of the 6thth International Conference on Embedded Networked Sensor Systems*, pages 1–14, Nov. 2008.

[34] Texas Instruments. CC2420: 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver. Available at `http://www.chipcon.com/files/CC2420_Data_Sheet_1_3.pdf`, 2006.

[35] Texas Instruments. CC2430: System-on-Chip Solution for 2.4 GHz IEEE 802.15.4 / ZigBee. Available at `http://www.ti.com/lit/gpn/cc2430`, 2007.

[36] Texas Instruments. CC2520: Second generation 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver. Available at `http://www.ti.com/lit/gpn/cc2520`, 2007.

[37] T. van Dam and K. Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *SenSys'03: Proceedings of the 1st Intl. Conference on Embedded Networked Sensor Systems*, pages 171–180, Nov. 2003.

[38] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *INFOCOM'02: Proceedings of the 21st Conference on Computer Communications*, June 2002.

[39] W. Ye, F. Silva, and J. Heidemann. Ultra-low duty cycle MAC with scheduled channel polling. In *SenSys '06: Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, pages 321–334, Nov. 2006.