Low-Power and Topology-Free Data Transfer Protocol with Synchronous Packet Transmissions

Jongsoo Jeong, Jongjun Park, Hoon Jeong, JongArm Jun, Chieh-Jan Mike Liang†, and JeongGil Ko
Electronics and Telecommunications Research Institute
† Microsoft Research
Email: jeonggil.ko@etri.re.kr

***Abstract*—Tightly synchronizing transmissions of the same packet from different sources theoretically results in constructive interference. Exploiting this property potentially speeds up network-wide packet propagation with minimal latencies. Our empirical results suggest the timing constraints can be relaxed in the real world, especially for radios using lower frequencies such as the IEEE 802.15.4 radios at 900 MHz. Based on these observations we propose PEASST, a topology-free protocol that leverages synchronized transmissions to lower the cost of end-to-end data transfers, and enables multiple traffic flows. In addition, PEASST integrates a receiver-initiated duty-cycling mechanism to further reduce node energy consumption. Results from both our Matlab-based simulations and indoor testbed reveal that PEASST can achieve a packet delivery latency matching the current state-of-the-art schemes that also leverages synchronized transmissions. In addition, PEASST reduces the radio duty-cycling by three-fold. Furthermore, comparisons with a multi-hop routing protocol shows that PEASST effectively reduces the per-packet control overhead. This translates to a ∼10% higher packet delivery performance with a duty cycle of less than half.**

I. Introduction

Over the past several years, wireless sensor networks (WSN) have enabled diverse monitoring applications in domains ranging from environment to health care. We observe that, due to the energy constraints of battery-operated nodes and data requirements, the traffic patterns in many of these monitoring deployments are event-based [4], [5], [10], rather than performing continuous and high-frequency data collection. In addition, while some of them require periodic status messages from each node to the gateway, the interval is quite infrequent (at the granularity of hours or days [17]). Two high-level design goals are shared by event-based monitoring applications: long network life time, and minimal human intervention. The former is driven by the fact that WSN deployments can be at locations without readily stable power sources, and the latter ensures that human effort does not linearly (or worse) grow with the network size. To this end, the community has identified both *dynamic topology maintenance* and *energy efficiency* as two challenges.

Typically, topology maintenance implements a process of having nodes learn their neighborhood by listening to periodic presence beacons from neighbors. While this primitive of neighborhood discovery is agnostic to the topology type, the drawback is the control overhead in beaconing and information exchange, especially in cases of high network dynamics. While it is possible to build on-demand routing paths without maintaining any network knowledge, the route discovery phase can take a long time to complete. A recent work, Glossy, by Ferrari et al. suggested that synchronized transmissions (from constructive interference) can speed up flooding a packet in the network [8]. The PEASST protocol, proposed in this work, leverages this rapid flooding to implement practical topology-free routing in WSNs. The challenges to make this approach practical are the tight timing constraints for the concurrent senders, and the lack of considerations for the energy metric.

Being a major energy consumer on a WSN node, the radio usage has been a focus in reducing the energy consumption. While some WSN applications can lower the amount of outgoing traffic with data compression, another approach is duty-cycling the radio to minimize the idle-listening times. An example is the receiver-initiated link layer, or Low-Power Probing (LPP), in which a receiver triggers communications by first advertising its presence. As later sections point out, radio duty-cycling introduces design challenges in rapid flooding, especially the uncertainty of which nodes should stay awake for the flooding.

Our contributions lie in a practical point-to-point data transfer protocol with synchronized transmissions for minimal topology maintenance, and asynchronous duty-cycling. First, we demonstrate that an IEEE 802.15.4-based platform at 900 MHz provides more relaxed constraints to encourage constructive interference, as compared to the platforms in previous work; thus, increasing room for practically realizing constructive interference-based systems. Second, we describe our PEASST protocol that combines both constructive inference-based synchronized packet transmissions and receiver initiated duty-cycling techniques. Finally, empirical results on a real-world testbed show an improvement of 52% in terms of radio duty-cycle over multihop routing protocol-based networks.

This paper is structured as follows. In Section II, we start our technical discussions by introducing preliminary analysis and experiments on the challenges of realizing synchronized packet transmissions on low-cost, low-power wireless platforms. Next, based on these observations we introduce our PEASST protocol for achieving effective synchronized packet transmissions in a contention-based wireless channel environment in Section III. We perform evaluations on the

978-1-4799-4657-0/14/$31.00 © 2014 IEEE

performance of PEASST in Section IV and discuss about some related work in Section V. Finally, we conclude the paper and bring up some topics for future work in Section VI.

II. Challenges in Realizing Synchronized Packet Transmissions with Constructive Interference

As synchronized packet transmissions have been the building block for several previous work in the wireless community, this section starts with an overview followed by empirical analysis to motivate our design decisions.

During a transmission, the radio transceiver modulates data bits into analog waveforms. With signals from concurrent transmitters, the receiver essentially observes wave interference, or the net effect of all waves. If the interference overpowers the intended signal, then the receiver will fail to decode the incoming signal with high probability. Interestingly, if all concurrent signals have the same wave form with matching phase, then the receiver will observe a single amplified signal. This phenomenon of synchronized packet transmissions is called *constructive interference*. Unfortunately, the constraint of matching wave phase (within a chip rate, T_c) imposes overhead. For example, on 2.4 GHz 802.15.4 radios, two concurrent waves cannot differ by more than 0.5 μsec in the transmission timing (e.g., 2 Mchips/sec), which motivates the need for frequent network-wide time synchronization to achieve a bound within a single chip rate [8]. As we will explain later, this constraint of 0.5 μsec can be relaxed with changes in the frequency-band of the radio.

Next, we present a set of data-driven observations for a hardware platform with relaxed time constraints for achieving successful synchronized packet transmissions with constructive interference.

A. Experiment Hardware Setup

We select the Atmel RF212 RFIC, a 900 MHz 802.15.4 radio supporting BPSK modulation [1]. First, due to the longer chip rates at 900 MHz, the time synchronization bound is relaxed by a factor of three in theory (i.e., $\sim$1.66 μsec vs. 0.5 μsec at 2.4 GHz). Second, our hardware configuration has a longer communication range due to the lower frequency of the 900 MHz band and higher receiver sensitivity from using BPSK (-108 dBm vs. -101 dBm on O-QPSK). This translates to deploying fewer relay nodes in the same geographical region, and reduces the cumulative "de-synchronization" as packets travel over many hops.

For our feasibility studies, we set up a wired testbed of two transmitting nodes, a single receiver, and a trigger node. We configure the senders so that they are capable of sourcing their radio's clock using either the on-board external crystal oscillator or an external waveform generator. The trigger node informs the senders of the start of an experiment using connections to GPIO pins, which provides identical or different transmission times for each of the two senders. Once triggered, senders immediately transmit a pre-loaded packet.

[1]The IEEE 802.15.4-compliant chip rate for 900 MHz radios with BPSK is 600 Kchip/sec.

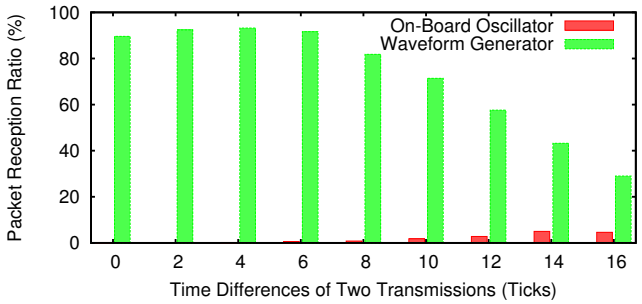

Fig. 1. Packet reception ratio for packets that were synchronously transmitted on our wired testbed. We vary the transmission time offset between the two transmitters to measure the resulting PRR and test for cases where the on-board oscillator is used to clock the radio and when an external waveform generator is used as the radio's common clock source. When using our RF212 radio, a single tick is $\sim$125 $n$sec and a single chip rate is 1.66 μsec (14 ticks).

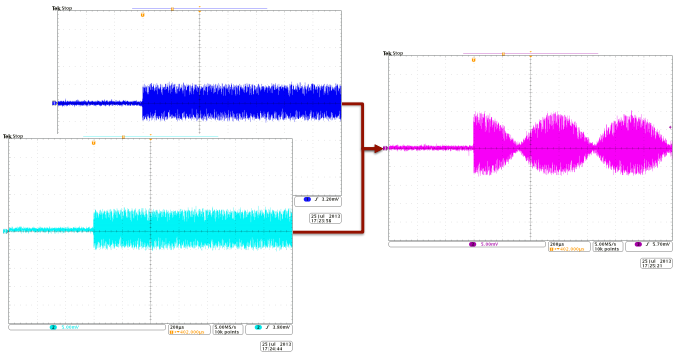

Fig. 2. The raw signals from the two transmitters (left) and the mixed signal of the two (right) captured using connections to an oscilloscope when using the on-board crystal oscillators as the transmitting radio's clock source.

B. Transmission Time Analysis

To quantify the timing constraints for achieving constructive interferences, we adjust the two senders' transmission timing parameters to introduce lags (Δ_{TX}) with an increment of 1 tick (or 125 $n$sec). We disable MAC-level features such as CCA checks and keep the received signal strength (RSS) the same for all packets.

Figure 1 plots the packet reception ratio (PRR) with respect to various Δ_{TX}. In theory, if two identical signals do not lag beyond 14 ticks (e.g., 1.66 μsec), the receiver should be able to successfully decode the signal. Surprisingly, unlike the observations from previous work [8], we observe a low PRR in spite of the tight synchronization of the two senders ($\Delta_{TX} < T_c$). Oscilloscope traces at the receiver indicates that the combined signal is not synchronized (c.f. Figure 2), even when the two senders are theoretically synchronized at the software scale. The shallow regions on the right of Figure 2 show two signals collide de-constructively. We conjectured that the reason behind this observation is the instability of the on-board crystal oscillators. To validate this, we repeat the experiment with the external waveform generator as the "common" clock source for both senders.

Figure 1 shows the improved PRR with the use of a

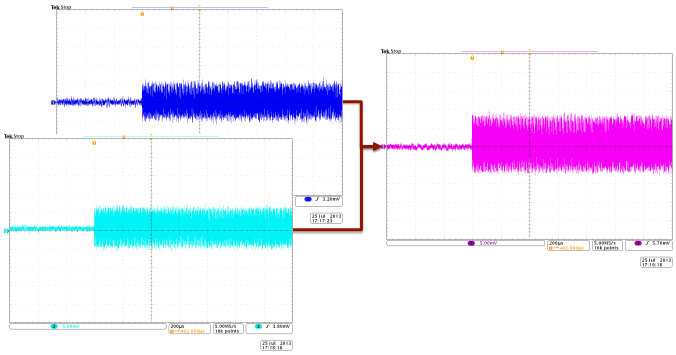

Fig. 3. The raw signals from the two transmitters (left) and the mixed signal of the two (right) captured using connections to an oscilloscope when using an external waveform generator as the transmitting radio's clock source.

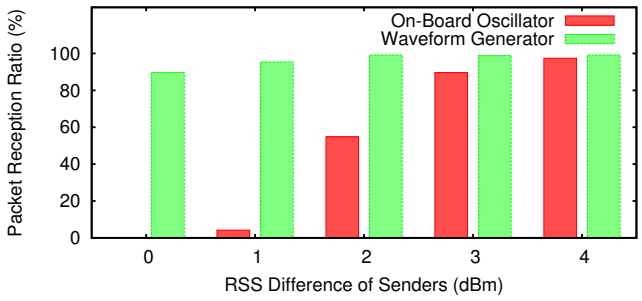

Fig. 4. Packet reception ratio of the packet transmitted from two different transmitters with Δ_{TX}=0 but with differences in reception power levels configured by connecting signal attenuators.

waveform generator. Furthermore, the resulting mixed signal shows a non-corrupted pattern as Figure 3. This suggests that using low-cost hardware components for designing low-power wireless platforms can be a reason that prevents the system from enjoying the benefits of making synchronized packet transmissions. In addition, Figure 1 also shows that the PRR degrades as the lag, T_c, increases. An interesting point is that the PRR drops to around 80% at $T_c = 10$ ticks (or 1 μsec), which is slightly lower than the theoretical upper bound of 14 ticks. While this is still a factor of two more relaxed than reported on the 2.4 GHz radio [8], we next show that the complex environment in real-world deployments actually help raise the PRR to an even higher level.

C. Received Signal Strength Analysis

Destructive interference happens when two concurrent signals are not synchronized in the analog wave forms, essentially if both have approximately the same signal strength. However, in the real-world deployments, the RSS of two signals (even transmitted at the same power level and distance) are likely to be different. This sub-section provides insights and analysis on how synchronized transmissions perform in the real-world radio environment.

We first configure the wired testbed to have a Δ_{TX} of 0, but with varying differences in the reception signal strength (Δ_{RSS}). We adjust the latter by connecting signal attenuators

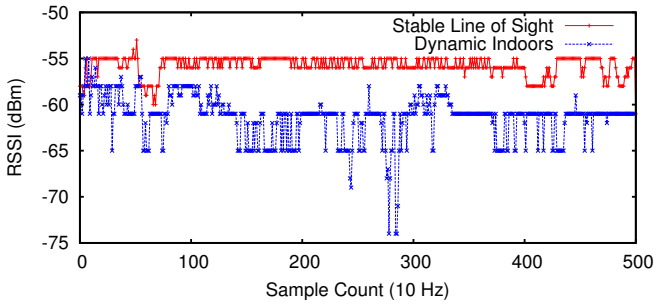

Fig. 5. Received signal strength collected over time in two different indoors environments.

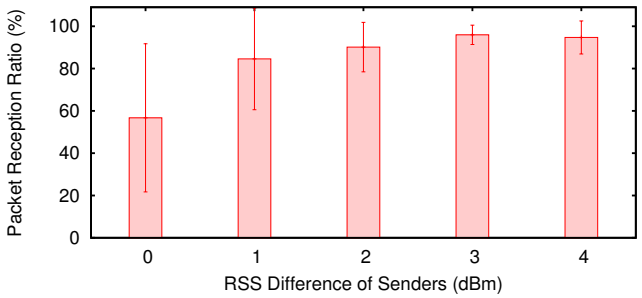

Fig. 6. Packet reception ratio plots for our wireless experiment with Δ_{TX}=0 and varying Δ_{RSS} collected in 9 different indoors environments.

to the sender's radio transceiver socket. Figure 4 plots the PRR with respect to each Δ_{RSS}. We observe that the PRR increases with Δ_{RSS} when using the relatively unstable on-board oscillators. Specifically, the improvement is as high as a factor of six with a mild RSS difference of 2 dBm. This result also suggests that even low-cost radio components can still achieve reasonable performance of synchronized transmissions.

Next, we verify the existence of RSS difference from multiple signals. We measured the RSS with a pair of nodes exchanging packets at 10 Hz, located 15 meters apart in various indoor environments. Figure 5 plots the RSS over time for both one relatively stable environment and one environment with furnitures and human movement. Results show that even in the stable environment with minimal human movement and line-of-sight (e.g., the best case scenario), the RSS continuously varies over time. Therefore, in a realistic scenario where various objects move around to complicate the channel environment, the signals sent from different radios should also show a high Δ_{RSS}. This observation confirms the intuition that the RSS difference from multiple signals are inevitable in indoor environments, and suggests the potential of synchronized transmissions on low-cost off-the-shelf radio components.

Finally, we quantify the observed PRR in a real indoor deployment setting. For this experiment we wirelessly trigger the senders (located at an equal distance at the receiver) to initiate packet transmissions using similar operations as the work by Ferrari et al. [8]. In Figure 6 we plot the mean PRR and standard deviation collected over nine different indoor

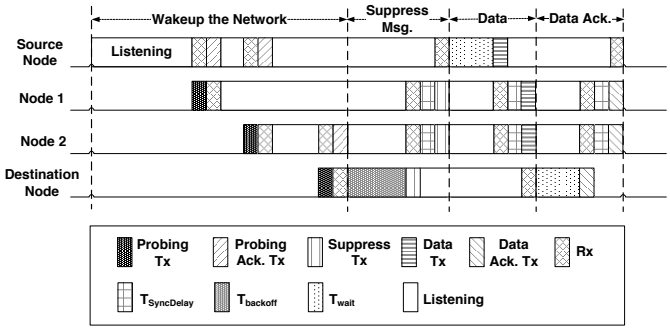

Fig. 7. Operational view of the PEASST protocol.

environments with respect to Δ_{RSS} (varied using attenuators) on the two senders. Since $\overline{RX\ TX_1} \approx \overline{RX\ TX_2}$, by configuring $\Delta_{RSS} = 0$ dBm, we are testing for the worst case scenario as observed in the results presented above. In reality, since nodes are commonly positioned "randomly", the node placement itself will naturally increase the chances of experiencing larger Δ_{RSS}. Notice that due to the channel dynamics, the experienced Δ_{RSS} increased to larger values than the preset values; thus, resulting in a high PRR for indoors environments compared to our wired tests.

Overall, these results suggest that ironically, having a complex channel environment, which increases the chances of experiencing large Δ_{RSS}, can be a more effective wireless channel environment in realizing synchronized packet transmissions with constructive interference.

III. PEASST: Point-to-Point Packet Exchange with Asynchronous Sleep and Synchronous Transmissions

Building on the results that synchronized packet transmissions are reasonably achievable in the real world, we design a point-to-point transfer protocol that performs asynchronous radio duty-cycling and at the same time achieve the benefits of synchronous packet transmissions. The rest of this section describes our PEASST (Point-to-Point Packet Exchange with Asynchronous Sleep and Synchronous Transmissions) protocol.

PEASST sits between the applications and radio driver to replace the network stack. Through a set of object-based APIs, applications pass the pointer and the size of the object to deliver, along with the destination node address.

The data transmission process in PEASST has three phases (c.f. Figure 7): (1) selective network-wide wake up, (2) rapid-flood based packet delivery, and (3) sleep phase. PEASST addresses the following major challenges to be practical for real-world monitoring deployments. First, in contrast to Glossy where all nodes in the network need to participate in the flooding session, PEASST aims to reduce this number to improve the network energy efficiency. Second, PEASST supports multiple point-to-point traffic flows in the network by minimizing the contention among concurrent flood sessions.

A. Network Wakeup Phase

There are two categories of network-wide duty-cycling mechanisms: sender-initiated and receiver-initiated. PEASST implements the network-wide wakeup with receiver-initiated Lower-Power Probing (LPP) [14]. Compared to the sender-initiated Low-Power Listing (LPL) [2], LPP is known to perform more efficiently as the network scales up [14]. This characteristic enables a wide range of large-scale monitoring networks. In addition, LPP has a lower overhead and contention, as LPL relies on continuously transmitting packets to wake up neighboring nodes.

In the example of Figure 7, the source node (N_{TX}) and the destination node (N_{dest}) are two hops apart, and node 1 and 2 (N_1 and N_2) are within a single-hop range of both N_{TX} and N_{dest}. Before N_{TX} sends data packets to N_{dest}, it first initiates the network wakeup following the typical LPP operations. Specifically, it waits for the periodic wakeup of LPP nodes signaled by the presence beacon. Following the beacon, the node radio will leave the radio on for a pre-defined time $T_{LPPWait}$, to wait for neighboring nodes to send pending packets destined to it. In PEASST, the sender uses this time to transmit a MSG_{awake} to extend the receivers' radio ON time. MSG_{awake} also includes the destination node ID and size (e.g., packet count) of the data object to be sent in this session (C_{TX}). In Figure 7, N_{TX} wakes up N_1 followed by N_2. Then, N_2 wakes up N_{dest} (located one hop away) with a MSG_{awake} as well.

Wakeup Suppression. After N_{dest} receives MSG_{awake}, there is no need to wake up additional nodes in the network, as the set of intermediate nodes ($N_{intermediate}$) between N_{TX} and N_{dest} are already up.

N_{dest} can initiate the wakeup suppression by broadcasting $MSG_{suppress}$ after a short delay of $T_{backoff}$. This delay ensures that all probes and MSG_{awake} transmissions are at least two hops away from N_{dest} to minimize the interference for $MSG_{suppress}$. Starting from the wakeup suppression phase, the nodes in the network (or the subset that are associated with the data flow from N_{TX} to N_{dest}) has their radios on; thus, are ready to perform synchronized packet transmissions. Specifically, once nodes N_1 and N_2 receive $MSG_{suppress}$ from N_{dest}, which will happen simultaneously for both nodes since they are at the same hop, N_1 and N_2 will process the packet internally and forward the same message, $MSG_{suppress}$, in a synchronized manner after $T_{SyncDelay}$ ticks of the start frame delimiter (SFD) detected for the $MSG_{suppress}$ packet. This will allow synchronized packet transmissions and the packet will eventually reach N_{TX}.

B. Data Exchange Phase

When $MSG_{suppress}$ reaches N_{TX}, N_{TX} implicitly takes this as an indication that N_{dest} and $N_{intermediate}$ nodes are ready to participate in the synchronized transmission-based rapid-flood session. After a short wait of T_{wait} (to avoid collisions with ongoing $MSG_{suppress}$ transmissions), N_{TX} initiates its packets (a total of C_{TX} packets) with an interval of $I_{C_{TX}}$ to nodes in its one hop (e.g., N_1 and N_2 in Figure 7).

After receiving an incoming packet, all receivers wait for a fixed interval before broadcasting to encourage synchronized transmissions.

Since nodes record the meta data of packets they have forwarded, they can avoid re-broadcasting a particular packet. This prevents the "echo looping" problem.

C. Sleep Phase

PEASST implements two different schemes to instruct the network to return to low-power sleep state after the data objects have been transmitted.

Implicit Termination. As all nodes learn the data object size from MSG_{awake} (e.g., C_{TX}), they (i.e., N_{TX}, N_{dest}, and $N_{intermediate}$) can enter the LPP-based duty-cycling after the last packet is relayed. The benefit of this method is the simplicity, but the drawback is that packet losses can prevent nodes from entering the proper state. For such cases, nodes exit the data exchange phase after a timeout, TO_{TX}. We will discuss further into how this timeout is configured in the next section.

Explicit Termination. After N_{dest} receives all C_{TX} messages from N_{TX}, it sends an acknowledgment (MSG_{ACK}). Otherwise, N_{dest} sends an negative acknowledgment (MSG_{NACK}) after a timeout. After $N_{intermediate}$ forward either MSG_{ACK} or MSG_{NACK}, they terminate the data exchange phase and return to LPP-based duty-cycling. While this method requires an additional transmission phase for MSG_{ACK} or MSG_{NACK}, this explicit behavior can reduce the nodes' wait time (TO_{TX}) in networks with high packet loss rates.

D. Configuring the Timeouts

Since packets in PEASST operate in a collision-capable environment, for PEASST to operate smoothly between different phases, configuring effective timeout values become an important issue. Specifically, we consider two major timeout parameters: TO_{wakeup} and TO_{TX}.

Initiated at N_{TX}, TO_{wakeup} represents the time from N_{TX}'s initial wakeup (due to its intent to send data) until $MSG_{suppress}$ is received. If N_{TX} knows the number of hops to N_{dest}, computing TO_{wakeup} would be trivial as the LPP wakeup interval I_{wakeup} is pre-defined. However, since the global topology is not available a priori, we take an adaptive approach in computing TO_{wakeup} as Eq. 1.

$$TO_{wakeup} = \alpha \times TO_{wakeup} + (1 - \alpha) \times TO_{wakeupRep} \quad (1)$$

Here, TO_{wakeup} is initialized to $TO_{wakeup} = Maximum\ Hops\ (H_{max}) \times I_{wakeup}$. α is weighting factor, and $TO_{wakeupRep}$ is computed as follows.

$$TO_{wakeupRep} = H_{reported} \times I_{wakeup}, \quad (2)$$

where $H_{reported}$ is the reported hop count to N_{dest} collected from the previous transmission. Specifically, in MSG_{awake}, N_{TX} includes a counter for each hop node to increase, and N_{dest} includes $H_{reported}$ in $MSG_{suppress}$. While starting at a rough state, the exponentially weighted moving average (EWMA) allows TO_{wakeup} to reach a near-optimal value.

Once nodes on the path from N_{TX} to N_{dest} have their radios on, nodes need to keep track of an additional timeout that concerns the duration of the data transmission and acknowledgment phases. Specifically, a timer is initiated when $MSG_{suppress}$ is received and turns off the radio if the re-sleeping phase (e.g., using either of the implicit of explicit methods) does not complete within TO_{TX}. Since at this point $H_{reported}$ is known, TO_{TX_h} for nodes that are h hops from N_{TX} is computed as follows, in the case of the explicit termination scheme with acknowledgment.

$$\begin{aligned} TO_{TX_h} = (H_{total} \cdot C_{TX} + H_{total} - h) \times T_{SyncDelay} \\ + T_{backoff} + (C_{TX} - 1) \cdot I_{C_{TX}} + \epsilon, \end{aligned} \quad (3)$$

ϵ is a predefined buffering time to account for unexpected delays. On the other hand, when using implicit termination, the equation for computing the timeout becomes even simpler.

$$\begin{aligned} TO_{TX_h} = (h \cdot C_{TX}) \times T_{SyncDelay} \\ + T_{backoff} + (C_{TX} - 1) \cdot I_{C_{TX}} + \epsilon. \end{aligned} \quad (4)$$

By waiting for TO_{TX_h} after the reception of $MSG_{suppress}$ at N_{TX}, each node participating in the flow is guaranteed to stay awake throughout the data exchange process. This property allows synchronized transmissions to work effectively while minimizing the radio ON time. We will later discuss the parameter configurations used for our analysis and experiments in Section IV.

E. Supporting Multiple Traffic Flows

The two mostly related previous work by Ferrari et al., either consider only network-wide flooding scenarios [8] or enforce network-wide transmission schedules to restrict multiple traffic flows from taking place at a given time [7]. Assuming the network has multiple traffic flows (e.g., $\mathbf{F}_1, \mathbf{F}_2, ..., \mathbf{F}_n$,) that start randomly at different nodes, PEASST supports multiple flows by *implicitly* scheduling the flows and minimizing the impact of contentions among traffic flows.

We note that, with selective wakeup, PEASST nodes fully activate their radios only if there is a chance that they are on the path from the source to the destination. This design helps constrain the coverage of a traffic flow to an area. In the PEASST design, there are two additional factors that can cause such interference. Particularly, the probe packets of LPP and MSG_{awake} packets are the two types of packets that are transmitted at any random times and are the ones that can interfere with the synchronously transmitted packets. Next, we describe methods to overcome these interference and other external challenges.

CCA and Random Backoff: We configure all LPP probes and MSG_{awake} packets to perform CCA checks prior to transmissions and set a backoff when the channel is sensed busy. Also, we configure the CCA threshold (the minimum energy level to detect and declare a busy channel) to be significantly lower (-95 dBm) compared to the default values

of -86 dBm for the Atmel RF212 operating with BPSK and -77 dBm for CC2420 operating with O-QPSK. This allows LPP probes and MSG_{awake} to effectively discover on-going transmissions and backoff when needed. We note that the first transmission attempt of the synchronous packet transmissions (e.g., data packets, $MSG_{suppress}$ and $MSG_{(N)ACK}$) also perform this CCA check to monitor the channel conditions before transmitting.

Dynamic Transmission Power for Prioritization: Despite the improved CCA sensitivity, there is still a chance for collisions due to hidden terminals. Based on the experiences from the testbed, we assign priority to each packet type by using the transmission power. Specifically, packets with higher priority are transmitted with relatively higher transmission power. In our current implementation, the synchronized transmissions during the data exchange phase have a transmission power of at least 5 dBm higher than those of LPP probes and MSG_{awake}. This provides the synchronized packets with a higher probability to reach a target destination node by (1) increasing the communication range; thus, minimizing the number of hops to travel and (2) by opportunistically exploiting the capture effect so that the stronger packets can be properly decoded despite the potentially negative packet collisions [13], [15].

Timeout Configurations: While the two methods above discuss about mostly prioritizing the synchronized flows, in a contention-based environment, other external factors can also disrupt the packets. To maintain network efficiency even under harsh environments we introduced a pair of timeout values in Section III-D. Here we introduce two additional "guard times" included in PEASST. The first is the term $T_{backoff}$, used in the text earlier, and the second is T_{wait}. To recap, $T_{backoff}$ is used to confirm that all probing messages are at least two hops away from N_{dest} before initiating $MSG_{suppress}$ so that chances of packet collision at the first hop is decreased. On the other hand, before initiating the data packets (at N_{TX}) and $MSG_{(N)ACK}$ (at N_{dest}) nodes wait for only T_{wait}, where $T_{wait} < T_{backoff}$ since at this phase the nodes know that the intermediate nodes are prepared to perform synchronized packet transmissions and no LPP probe or MSG_{awake} packets will be initiated from these nodes.

Duplicate Transmissions: Lastly, we configured PEASST so that after making an initial attempt to start a synchronized transmission flow, if the forwarding process of the packet cannot be overheard after $2 \times T_{SyncDelay}$, a retransmission is made. This is to actively increase the reliability when packets arrive corrupted at the receivers.

IV. Evaluations

We evaluate the performance of PEASST using both Matlab-based simulations and also using a 15 node indoor testbed. While focusing on the packet reception ratio (PRR), radio duty-cycle, and communication latency, we compare the performance of PEASST with LWB and the standardized IETF RPL routing protocol combined with the LPL low-power MAC. We present our experimental parameters in Table I.

Parameter	Default Value	**Parameter**	Default Value
I_{wakeup}	2 sec	$I_{C_{TX}}$	Variable
$T_{SyncDelay}$	375μsec	$T_{backoff}$	200 $m$sec
C_{TX}	1	$T_{LPPWait}$	20 $m$sec
Packet length	15 bytes		

TABLE I
SUMMARY OF BASE PARAMETERS USED IN OUR EVALUATIONS.

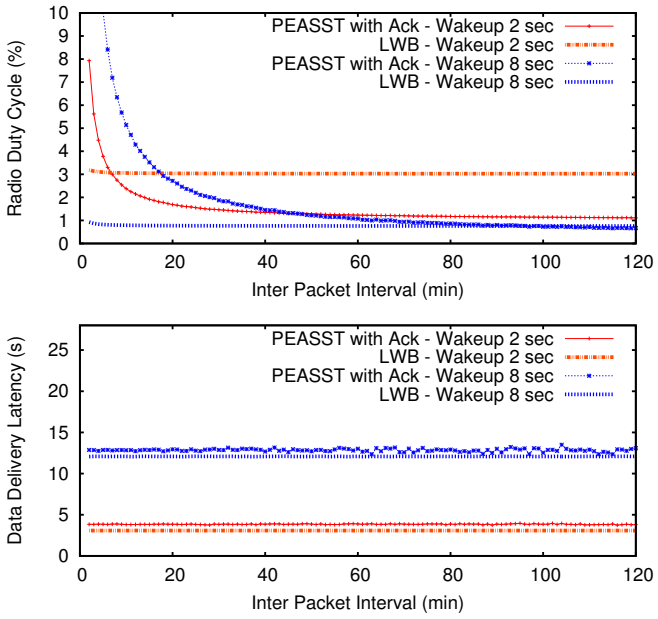

Fig. 8. Mean radio duty-cycle (top) and packet delivery latency (bottom) achieved in out Matlab simulation environment for PEASST and LWB with varying inter-packet intervals and different wakeup intervals. We use a linear topology of 6 nodes for our Matlab-based evaluations.

A. PEASST vs. LWB: Latency and Duty-Cycle

The first step of our evaluation is a simulation-based evaluation of PEASST against the recently proposed low-power wireless bus (LWB) [7]. While we point the readers to [7] for details on LWB, on a high level, LWB exploits synchronized packet transmissions by configuring a global wakeup schedule (e.g., TDM-based) to isolate data flows. Specifically, all nodes wakeup to participate in the forwarding process and sleeps based on the TDM schedule that the gateway provides. The goal of this evaluation is to compare the performance of the two protocols under various traffic loads to observe what conditions each scheme is suitable for. We implement both schemes in MATLAB and present the results below.

In Figure 8 we configure the radio's wakeup interval I_{wakeup} to be identical for both PEASST and LWB (in [7] this term is named as the "communication round" interval) with varying inter-packet-intervals (IPIs) to compute the expected radio duty-cycle and packet delivery latency. We assume a line topology of six nodes where each node is connected to its direct neighbors only. Given that synchronized transmissions treat all n-hop neighbors equally when transmitting its packets (e.g., MAC-layer broadcasting-based protocol), we find this as

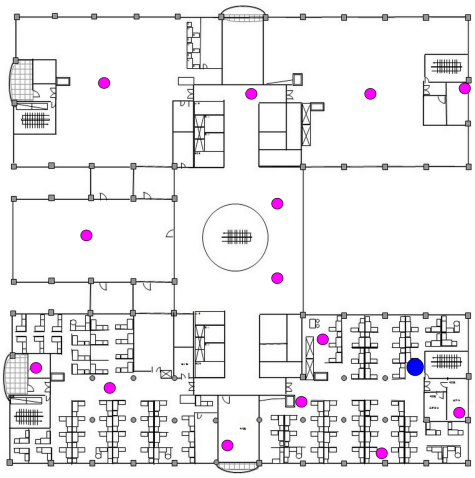

Fig. 9. Pictorial view of our indoors testbed. The testbed consists of 15 nodes equipped with Atmel RF212 radios operating on the 900 MHz-band deployed to form a maximum of 4 hops.

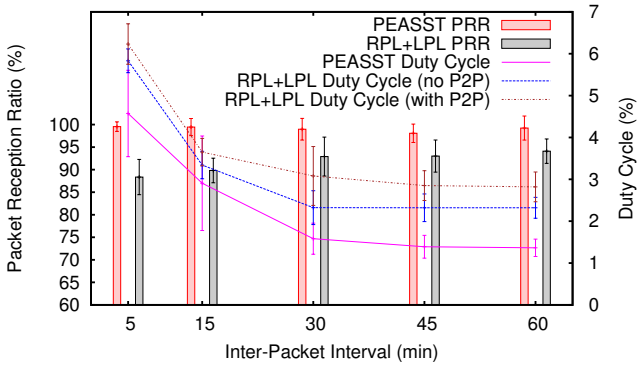

Fig. 10. Packet reception ratio and duty-cycle plots for PEASST and RPL+LPL with varying IPI. For RPL+LPL we plot two cases with and without routing support for point-to-point traffic.

a simple but reasonable topology to test the two protocols.

On the top of Figure 8 we plot the duty-cycle of PEASST and LWB configured with two different I_{wakeup}. We can notice that regardless of I_{wakeup} (but more prominent for lower I_{wakeup}), PEASST shows lower duty-cycles with increasing IPIs. This difference grows to as much as three-fold when IPI = 120 sec and I_{wakeup} = 2 sec. The main reason behind LWB's high duty-cycle is due to the fact that LWB requires all nodes to wake up to perform time synchronization and data forwarding at every I_{wakeup}. On the other hand, since PEASST nodes only wake up for a small $T_{LPPWake}$ at every I_{wakeup}, nodes can keep very low duty-cycles when the IPI is high. Nevertheless, as I_{wakeup} increases, we can notice that the radio duty-cycle of LWB decreases to a significantly low level as well. We point out that the latency and duty-cycle differences observed with and without the data acknowledgments (for data exchange phase termination) in PEASST was $<$0.5% in all cases.

Using the bottom of of Figure 8, where we plot the end-to-end latency (from the point where N_{TX} wakes up to send a packet until reception of MSG_{ACK} at N_{TX}) of PEASST and LWB, we can see that while increasing I_{wakeup} reduces LWB's duty-cycle, this comes at a price of trading off data delivery latency. Notice that for both protocols, the latency changes linearly with increasing I_{wakeup}. We also point out that despite the burden of having to wakeup intermediate nodes from LPP sleep mode, PEASST shows a latency performance that is only $\sim$1 second longer than LWB. Based on these observations, we can conclude that PEASST is a more suitable solution than LWB for applications with sparse and event-based traffic patterns, since it can deliver packets with similar latencies while achieving a much lower radio duty-cycle.

B. PEASST vs. RPL+LPL: PRR and Duty-Cycle

Using a sparsely deployed 15 node indoors testbed (see Fig. 9), we test the performance of PEASST (with acknowledgment-based termination) and compare it with a standardized routing protocol for low-power and lossy networks (LLNs), IETF RPL [18]. The RPL routing protocol, itself, does not have the capabilities of turning off the nodes' radios for conserving its limited battery resources. For this purpose, we integrate low-power listening (LPL) to the RPL routing protocol. Implemented in TinyOS, this implementation has been used in various previous work; thus, the validity of the implementation is proven [11], [12]. In this work we use OF0, the hop count metric as RPL's path selection metric, which is the default routing metric of the RPL standards. Using these two protocols we run experiments with varying IPI and present their respective PRR and duty-cycles in Figure 10. We vary IPI and configure nodes to send packets to a *single* destination node at each IPI.

Notice in Figure 10 that the PRR for PEASST is $\sim$100% for all IPI while for RPL+LPL, the PRR is $\sim$90%. The main cause of the low PRR for RPL+LPL is the sparse deployment of the sensor nodes in our testbed. RPL will try to choose a (single) next hop node towards the destination and wait for that node to wake up when trying to make data transmissions. Under dynamic channel conditions and frequent link quality fluctuations, RPL will try to improve its routes. However, given a sparse topology, the number of nodes that RPL can select as forwarders are limited (both physically and with respect to RPL's routing policies). On the other hand, since PEASST is designed to be topology-free, nodes have the capability to select "any node(s)" as its data forwarder; thus, increase its probabilities of delivery the packet to the next hop successfully. Furthermore, when observing the results for the two protocols' duty-cycles, the differences in performance is more evident. Notice that we plot two different cases for RPL+LPL's duty-cycle. RPL can provide both collection and point-to-point routes, but the cost of providing point-to-point routes comes at a price of sending more frequent node connectivity reports (defined as destination advertisement objects in [18]). Nevertheless, since PEASST offers such point-to-point data delivery capabilities (e.g., not only data collection), we find this as a fair comparison. We

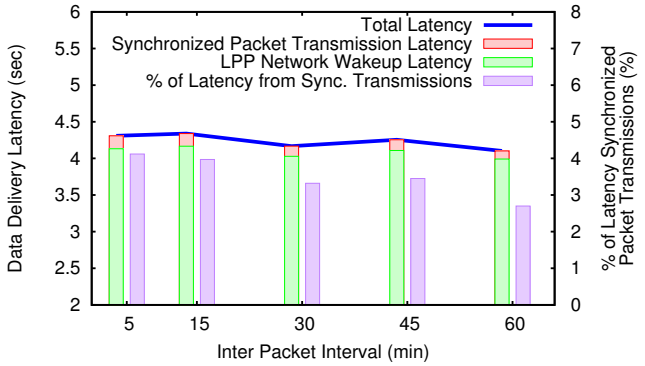

Fig. 11. Packet delivery latency breakdown and the percentage of latency taken up by the data exchange phase for PEASST.

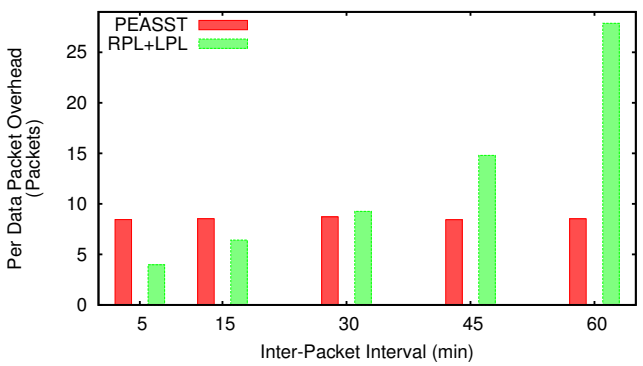

Fig. 12. Per data-packet overhead-packet count for PEASST and RPL+LPL.

test one case where this routing report is made at an interval of 30 minutes (e.g., with P2P) and another case where only data collection routes are maintained (e.g., no P2P).

Notice that the average duty-cycles of PEASST nodes are at least 13% lower than those of RPL+LPL nodes in any configuration. Specifically, despite not using P2P routes (e.g., relatively less routing overhead generated), the duty-cycles of PEASST is reduced by as much as 48% compared to RPL. Furthermore, when P2P routes are supported, PEASST reduces the duty cycle by as much as 52%. Looking deeper into the results for PEASST, while our two evaluation methods are not equally comparable (e.g., different topology, different channel environment, etc.), we point out that compared to the results in Figure 8, the duty-cycles we observe on the testbed are relatively high. Quantitatively speaking, for all IPIs that we tested, we were able to observe an average of 17% increase in duty-cycles compared to the simulation-based results in Figure 8. Based on deeper analysis on the packet traces, we were able to identify that in several cases MSG_{ACK} was not properly received; thus, forcing nodes to stay awake for TO_{TX}. Furthermore, the missed packets in our experiments with PEASST were mostly from N_{TX} not being able to find proper $N_{Intermediate}$. This naturally caused the nodes that were awake to continue its ON state for TO_{wakeup}. Furthermore, due to the increasing level of contention, the duty-cycles increase with decreasing IPI.

On the other hand, in terms of packet delivery latency we were able to observe a similar end-to-end latency compared to the plot in Figure 11. By breaking down the cause of the overall packet delivery latency, we can see that $>\sim$96% of the time was spent for waking up the LPP-sleeping nodes. Furthermore, due to similar reasons as the duty-cycles (e.g., unexpected packet loss), although the differences are less protrusive for the latency results, we can notice an increasing trend as the IPI decreases.

Lastly, we compare the control overhead to achieve data packet delivery for PEASST and RPL+LPL. We note that since LPL retransmits packets continuously until the next hop node wakes up to receive the packet, we count these transmissions as a single transmission for RPL+LPL. In Figure 12 we plot the average number of control packets issued for each data packet transmission. Notice that with a low IPI, the amount of relative control overhead (with respect to the number of data packet transmissions) is lower for RPL. However, we can see that as IPI increases, this overhead ratio increases quickly. This is a somewhat predicted behavior given that despite the lack of data traffic, proactive routing protocols will continuously try to maintain the optimal routes. On the other hand, since PEASST only wakes up nodes and issues control traffic when data traffic exists, the varying IPI has minimal effect on the per data packet control packet overhead. Overall, the experimental results from our testbed suggests that PEASST is a well-suitable protocol for applications with sparse traffic patterns.

V. Related Work

We now position our work among previous work related to PEASST. Specifically, we discuss about various previous work in the domain of asynchronous radio duty-cycling algorithms and schemes that utilize constructive interference for synchronized packet transmissions and capture effect.

Asynchronous Radio Duty-Cycling Schemes and Multi-hop Routing: Implementing asynchronous radio duty-cycling schemes minimize the wireless nodes' radio on-time by allowing nodes to turn off their radios when there is no traffic to serve in the network [2], [6], [14], [16], [19]. While these protocols are implemented under various paradigms, in this work we focus on the receiver-initiated radio duty-cycling paradigm. One example of such base-technology is LPP [14]. As discussed earlier, LPP shows effectiveness in waking up a large set of nodes and suits our purposes of minimizing the interference levels for "protecting" the synchronized packet transmissions. Thanks to this capability, similar concepts have been utilized in various previous work [6], [16]. We point out that these protocols focus on the MAC layer of the networking stack while PEASST implements data delivery capabilities on top of the receiver initiated wakeup paradigm. Nevertheless, these wakeup/MAC protocols can easily be connected to various routing engines to form a complete low-power networking suite (e.g., CTP [9], IETF RPL [18]). However, as Section IV-B shows, these routing protocols generate a large number of control messages to make the protocols less suitable

for applications with sparse, event-based traffic. One example of such a combination is Dozer which provides extremely low-power operations for data gathering applications [3]. While being an efficient protocol, Dozer is designed for data collection scenarios and extending it to many-to-many scenarios would require learning the sleeping schedules of all neighbor nodes and maintaining the routing tables to each destination.

Networking with Synchronized Packet Transmissions: The concept of utilizing synchronized packet transmissions with constructive collisions in low-power networks was initially presented in a work by Ferrari et al. [8]. In this work the Glossy protocol was presented to make efficient network-wide flooding possible. While Glossy acts as the basis of our study, Glossy does not provide support for node duty-cycling; thus, is not suitable for networks that target long lifetimes. As follow-up work the low-power bus (LWB) was proposed to realize point-to-point packet transmissions using Glossy [7]. However, to allocate the network for a single data flow at all times, an LWB network keeps a continuous network transmission schedule. This requires all nodes to wake up for long periods periodically; thus, as we show in Section IV-A, is less efficient than PEASST when dealing with event-based, sparse traffic generating applications.

Flooding with the Capture Effect: The capture effect explains the phenomenon where a receiver can hear packets that come in at relatively stronger signal strengths among other overlapping signals that can be overheard at the same time [1]. In WSNs capture effects are widely used to implement fast flooding protocols [13], [15]. While, the concept of synchronized transmissions used in this work and capture effect both deal with the overlap of transmitted signals, capture effect deals with "ignoring" weaker signals to decode the stronger ones while the synchronized transmissions in this work deal with "colliding constructively".

VI. Future Research Directions and Conclusions

In this work, we examine the feasibility of applying synchronized packet transmissions with constructive interference to low-cost, low-power wireless systems. Based on the results of our feasibility study, we design, implement and evaluate the PEASST protocol which provides point-to-point packet transmissions using a combination of an asynchronous radio duty-cycling scheme and synchronous packet transmissions. To the best of our knowledge, PEASST is the first protocol that attempts to apply synchronized packet transmissions to low-power networks where active channel contention factors exist. We evaluate PEASST against state-of-the-art low-power data transmission protocols and show that PEASST achieves reliable packet delivery with a duty-cycle of $\sim$1.3% and relatively low latency levels. We envision this work as a stepping stone in applying the concept of synchronized packet transmissions to various wireless channel environments. As a first step to realize this, we plan on incorporating different network wakeup schemes to PEASST as a way to further minimize the packet delivery latency and duty-cycles to a lower level. Furthermore, we plan on designing a real application system with PEASST as the main data transmission protocol.

Acknowledgements

This work is supported by "Development of Self-Powered Smart Sensor Node Platform for Smart and Green Building" [#10035570], and Basic Research Project (Development of an integrated early detection system of landslides based on real-time monitoring) of KIGAM, both funded by MSIP, Korea.

References

[1] A. Bottcher and M. Dippold. The capture effect in multiaccess communications-the rayleigh and landmobile satellite channels. *Communications, IEEE Transactions on*, 41(9):1364–1372, 1993.

[2] M. Buettner, G. Yee, E. Anderson, R. Han, and M. B. G. Yee. X-mac: A short preamble mac protocol for duty-cycled wireless sensor networks. In *Proceedings of SenSys 2006*, Nov. 2006.

[3] N. Burri, P. V. Rickenbach, and R. Wattenhofer. Dozer: ultra-low power data gathering in sensor networks. In *Proceedings of IPSN 2007*, Apr. 2007.

[4] R. Cardell-Oliver, K. Smettem, M. Kranz, and K. Mayer. Field testing a wireless sensor network for reactive environmental monitoring [soil moisture measurement]. In *Intelligent Sensors, Sensor Networks and Information Processing Conference, 2004. Proceedings of the 2004*, Dec. 2004.

[5] M. Chang and P. Bonnet. Meeting ecologists' requirements with adaptive data acquisition. In *Proceedings of SenSys 2010*, Nov. 2010.

[6] P. Dutta, S. Dawson-Haggerty, Y. Chen, C.-J. M. Liang, and A. Terzis. Design and evaluation of a versatile and efficient receiver-initiated link layer for low-power wireless. In *Proceedings of SenSys 2010*, Nov. 2010.

[7] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele. Low-Power Wireless Bus. In *Proceedings of SenSys 2012*, Nov. 2012.

[8] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh. Efficient Network Flooding and Time Synchronization with Glossy. In *Proceedings of IPSN 2011*, Apr. 2011.

[9] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection Tree Protocol. In *Proceedings of the 7^{th} ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 1–14, Nov 2009.

[10] J. Ko, J. Park, J. A. Jun, and N. Kim. Just send me the summary!: analyzing sensor data for accurate summary reports in indoor environments. In *Proceedings of SenSys 2012*, Nov. 2012.

[11] J. Ko, A. Terzis, S. Dawson-Haggerty, D. Culler, J. Hui, and P. Levis. Connecting Low-Power and Lossy Networks to the Internet. *Communications Magazine, IEEE*, 49(4):96 –101, April 2011.

[12] J. Ko, N. Tsiftes, A. Dunkels, and A. Terzis. Pragmatic low-power interoperability: Contikimac vs tinyos lpl. In *Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2012 9th Annual IEEE Communications Society Conference on*, pages 94–96, 2012.

[13] J. Lu and K. Whitehouse. Flash flooding: Exploiting the capture effect for rapid flooding in wireless sensor networks. In *Proceedings of IEEE INFOCOM 2009*, Apr. 2009.

[14] R. Musaloiu-E., C.-J. M. Liang, and A. Terzis. Koala: Ultra-low power data retrieval in wireless sensor networks. In *Proceedings of IPSN 2008*, Oct. 2008.

[15] D. Son, B. Krishnamachari, and J. Heidemann. Experimental study of concurrent transmission in wireless sensor networks. In *Proceedings of SenSys 2006*, Nov. 2006.

[16] 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 *Proceedings of SenSys 2008*, Nov. 2008.

[17] G. Werner-Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees, and M. Welsh. Deploying a Wireless Sensor Network on an Active Volcano. *IEEE Internet Computing, Special Issue on Data-Driven Applications in Sensor Networks*, Mar. 2006.

[18] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550 (Proposed Standard), Mar. 2012.

[19] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. In *Proceedings of IEEE INFOCOM 2002*, June 2002.