Low Power Counting via Collaborative Wireless Communications

Wenjie Zeng, Anish Arora, and Kannan Srinivasan
Computer Science and Engineering
The Ohio State University
Columbus, OH, USA
{zengw, anish, kannan}@cse.ohio-state.edu

ABSTRACT

Metrics that aggregate the state of neighboring nodes are frequently used in wireless sensor networks. In this paper, we present two primitives that exploit simultaneous communications in 802.15.4 radios to enable a polling node to calculate with low power the number (or set) of its neighbors where some state predicate of interest holds. In both primitives, the poller assigns transmission powers and response lengths to its respective neighbors for their simultaneous response to each of its poll requests. The two primitives adopt complementary schemes for power assignment such that the Received Signal Strength Indicator (RSSI) of the respective signal from each neighbor is significantly different from that of all others in one primitive and nearly equivalent to that of the others in the other. The first primitive, LinearPoll, suits sparse networks and consumes energy that is linear in the size of its neighborhood, whereas the second primitive, LogPoll, suits dense networks and consumes constant energy. Compared to the state-of-the-art solutions that use multiple sub-carriers, our primitives are simpler and more compute-efficient while provide estimation with comparable quality. Compared to single-carrier solutions, our primitives achieve comparable quality at less than half the energy cost or richer information at comparable energy cost. They are also compatible with other radio physical layers. Based on our implementation for CC2420 radios on the TelosB platform, we evaluate the primitives in different wireless environments and neighborhood topologies to study their performance, the tradeoff between their estimation accuracy and energy cost, and methods for tuning their critical parameters, and we compare them with baseline counting protocols.

Categories and Subject Descriptors

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

IPSN'13, April 8–11, 2013, Philadelphia, Pennsylvania, USA.

Keywords

Counting; Sensor Network; Energy Efficiency

1. INTRODUCTION

A common pattern in sensor network applications is the collection of up-to-date neighborhood metrics to perform local or distributed decision making. We may abstract this pattern in terms of a local state predicate of interest P over the sensor values or other local variables at each neighboring node. For a collector node i whose neighboring set of nodes is N_i, let P_i be the set of nodes in N_i where the predicate P holds. The pattern typically involves counting the number of nodes in P_i at i. When this count is dynamic, its efficient ongoing collection becomes material to system performance.

Examples of this pattern are found at various layers of the network. In medium access control, for instance, scheduling depends on the number of contending sensors; P in this case is whether the node has a non-empty queue. By estimating $|P_i|$, CSMA-based MAC protocols [10, 12] choose a contention window and a backoff window that optimizes efficiency and latency. Likewise, TDMA protocols [3, 15] avoid wasting time slots that are reserved for inactive neighbors. At the application layer, detection and alerting for some event of interest P is made tolerant to individual sensor false alarms by redundant collection of P_i and checking that the count of event witnesses is nontrivial. For example, in a volcano monitoring deployment [16], because the real-time seismic data of individual nodes is high in volume and low in accuracy, it is infeasible for a poling node to trust a single neighbor or to constantly download the raw data from all neighbors. Instead, a polling node only collects threshold-based events (predicates) from its neighbors based on their local processing. The bulk download of the raw data is only initiated when the number of detected events exceeds some threshold. When seismic activities are rare, the polling node spends most of its energy collecting local predicates of its neighbors rather than downloading the actual data. In such cases, employing our primitives and collect the local predicates concurrently would substantially reduce the energy consumption of the polling nodes.

In this paper, we address how neighbors collaboratively and efficiently communicate their P to a polling node i. We propose two energy efficient and complementary communication primitives that estimate P_i with different asymptotic energy costs and precisions. Rather than relying on explicit data communication, both primitives require each neighbor to immediately acknowledge a poll from i if P holds locally, using an assigned transmission power and response length.

For sparse networks, our first primitive, LinearPoll, estimates the set P_i with a cost that is linear in $|N_i|$; for dense networks, our second primitive, LogPoll, estimates $\log |P_i|$ with a constant cost. Polling nodes can choose between (or combine) the two primitives based on their neighborhood topology and application requirements. Both primitives allow users to control tradeoffs between accuracy and cost. In both, i exploits the received signal strength —as indicated by the RSSI— of the superposed signal from the responding neighbors.

Why and how we use RSSI? Although low-power radios are resource constrained, they provide access to several physical layer measures. In the latest 802.15.4 standard [1], the physical layer is required to provide three basic parameters: Energy Detection (ED), Link Quality Indicator (LQI), and Clear Channel Assessment (CCA). ED represents the RSSI detected; LQI represents the quality of a received packet, and CCA is a binary indicator based on ED. Because LQI requires successful packet reception, and hence does not work for concurrent responses, and CCA is derived from ED, we construct our primitives based on the ED metric. The actual mapping between ED and RSSI is platform specific, but nevertheless bijective [4]. For ease of exposition, we will henceforth use RSSI to represent both the signal strength (in dBm) and the integer ED values provided by the radio.

A key idea in LinearPoll is to ensure that the RSSI from each neighbor is sufficiently different from each other. It is then possible to select different response lengths for each neighbor such that measuring the drops in signal strength of the superposed responses at the poller lets it count (and in fact identify) each neighbor that responds to a poll. Conversely, a key idea in LogPoll is to ensure that both the response RSSI and the response length from each neighbor are nearly the same. As is analytically and empirically shown in this paper, the RSSI of the superposed signal (in dB) increases log-linearly with the number of responding neighbors. It is therefore possible to devise a local scheme for the poller to count the number of responding neighbors in a logarithmic scale.

As neighbors respond with the assigned power and length, the poller expects RSSI sample sequences of different structures from the two primitives: while the poller expects a cascading RSSI sequence in LinearPoll, it expects a fixed-length response of stable RSSI whose value depends on the number of responders in LogPoll. The estimation accuracy depends on how well we ensure that the measured RSSI sample sequence matches the expected sequence even in the presence of superposition of response signals, i.e., irrespective of uncertainty in the locations of the neighbors and reaction time differences in the simultaneous responses, the received signal strength of the superposed response at the poller has bounded variability with respect to its expected value.

Two factors are critical for us to ensure that the RSSI sample sequence matches what is expected: the respective response RSSI from individual neighbors and how these responses superpose. If these two factors change arbitrarily and cannot be predicted, the RSSI sequence will also become unpredictable and diverge from what is expected. In general, the RSSI of a response from a given neighbor depends on path loss, fading resulted from shadowing and multipath, and external interference. Fortunately, as we target static networks, i.e., networks consisting of static nodes in a stationary physical environment, the response RSSI can be reliably predicted because path loss remains constant and fading changes slowly as long as the poller can effectively detect external interference and reject contaminated samples. More discussion on detecting interference is in Section 5.1; for non-static networks, mobility of both the sensor nodes and the physical environment results in unpredictable fading effects, which renders accurate estimation of response RSSI rather difficult. In such cases, our primitives may not apply. Systematic evaluation of the proposed primitives in a non-static network is beyond the scope of this paper. In the rest of the paper, we focus our discussion on static networks.

The other critical factor, i.e., the superposition of concurrent responses, is also statistically predictable. 802.15.4 uses Direct Sequence Spread Spectrum (DSSS) and maps each symbol to a pseudo-random chip sequence, which is then modulated with O-QPSK. For the superposed signals to constructively add or destructively add, these signals should be identical and be perfectly synchronized at the center (carrier) frequency. Such a high-precision synchronization is highly unlikely, even impossible, when all the nodes simply respond to a poll, albeit simultaneously. Thus, the superposition has signals with random phases and add up with a high probability to their mean, which ensures that it is unlikely for any RSSI sample to differ from its expected value by a large amount.

Last but not least, an important feature of our primitives lies in the light-weight initialization. Response power and length assignment for both primitives solely depends on the path loss between neighbors and the poller. For a given static neighborhood, since path loss is constant, both primitives need only to assign response power and length once based on a one-time measurement of the path loss.

Contributions of this paper. We present two complementary low-power counting schemes that work reliably in the presence of measurement errors, environment changes, and varying external noise and interference sources. Also, we show how and when to use these counting schemes by extensively studying their performance under varying conditions. Moreover, we discuss the general applicability of these schemes to any wireless platform (or physical layer). We identify the critical parameters for the performance of the primitives, and identify hardware induced constraints in the context of our implementation for the 802.15.4 CC2420 radio on the TelosB platform. We evaluate performance and tradeoff between cost and estimation accuracy for both primitives in two different sensor network testbeds. We show that the primitives are resilient to variation in radio sensitivity, neighborhood topology, and background environment.

The main findings are summarized as follows:

- Both primitives can be initialized either with piggyback traffic or dedicated control traffic. In the worst case where control traffic has to be sent, the control overhead is no more than 0.2%.
- When the assigned output powers ensure that response RSSI's are sufficiently differentiated, LinearPoll achieves an accuracy of at least 98% when there is sporadic interference and an accuracy of 95% even when malicious interference attacks 10% of the time.
- LogPoll achieves an accuracy of 99% (averaged over different numbers of responders) when all neighbors can adapt to a similar response RSSI and remains over

95% even when 10% of the neighbors fail to honor their power assignment.

- In LogPoll, interestingly, estimation accuracy increases with the number of responders .

Outline of the paper. Section 2 defines the system model, which in constructing the one-time measurement for predicting response RSSI used by both primitives. Based on accurate predictions of response RSSI from neighbors, Sections 3 and 4 respectively present the design, implementation, and empirical evaluation of our two primitives. Section 5 discusses their constraints and proposes extensions to relax these constraints. Section 6 recaps related work on communication primitives and protocols that exploit controlled transmission collisions. Section 7 discusses our future work and makes concluding remarks.

2. SYSTEM MODEL

Consider a node i in a static network whose neighborhood is the set of nodes N_i. Node i polls its neighborhood, and its neighbors respond immediately if and only if the predicate P holds locally. Each neighbor j responds using any one of the powers m_j from a set M, and the poller samples the RSSI of the response. Let $R(i,j,m_j)$ be the value of an RSSI sample that i obtains when a single neighbor j, $j \in N_i$, responds at power m_j.

Although $R(i,j,m_j)$ varies over time due to changes in the background noise, its variation is usually within a small bound unless interference occurs [4, 14]. This allows poller i to first construct an estimation function of $R(i,j,m_j)$, denoted as $\widehat{R}(i,j,m_j)$, based on a set of learning samples, and to later on use $\widehat{R}(i,j,m_j)$ as a prediction of the response RSSI from j at any given power $m_j \in M$. We have the following requirement over such predictions:

Requirement I (Bounded prediction error):
There exists a measurable σ_i for each poller i such that $\forall\, m_j \in M, \forall\, j \in N_i$:

$$\widehat{R}(i,j,m_j) - \sigma_i \le R(i,j,m_j) \le \widehat{R}(i,j,m_j) + \sigma_i \ . \qquad (1)$$

In other words, based on the learning samples, poller i can predict the response RSSI with bounded error σ_i from any given neighbor at any power. For ease of exposition, we introduce the following symbols to represent the bounds of $R(i,j,m_j)$

$$R_l(i,j,m_j) = \widehat{R}(i,j,m_j) - \sigma_i$$
$$R_u(i,j,m_j) = \widehat{R}(i,j,m_j) + \sigma_i \ ,$$

and abbreviate the number of nodes in N_i as N.

When interference occurs, some samples from a response can violate Requirement I. Nevertheless, each primitive implements special mechanisms to detect interference, which we will detail in Sections 3 and 4.

Initialization for RSSI Prediction. We assume that when neighbor j responds, the RSSI obtained at node i depends linearly on power m_j:

$$R(i,j,m_j) = a_{ij} \cdot m_j + b_{ij} \ , \qquad (2)$$

where a_{ij} and b_{ij} are coefficients that we estimate. Further, b_{ij} depends linearly on the additive background noise ρ and the time-invariant path loss over link (i,j), denoted as $L(i,j)$, as:

$$b_{ij} = \rho - L(i,j) \ . \qquad (3)$$

In the context of this paper, path loss $L(i,j)$ should be interpreted as the effective path loss from node j to node i that accounts for not only the free space signal attenuation, but also the shadowing and multipath effects.

For node i to predict $R(i,j,m_j)$ given m_j, it first estimates a_{ij} and b_{ij}. Poller i can acquire, and later on update, a set of learning samples (each of which is a triplet) $\{(R^k(i,j,m_j^k), m_j^k, \rho^k)\}$ by either snooping or directly receiving packets from node j, where $R^k(i,j,m_j^k)$ is the measured RSSI, m_j^k is the response power, and ρ^k is the background noise in the k^{th} learning sample. ρ^k is computed as the mean background noise measured before and after the sample[1]. Given a number of learning samples, the estimates of a_{ij} and b_{ij}, denoted as $\widehat{a}_{ij}$ and $\widehat{b}_{ij}$, are computed using least-squares approximation. In the meanwhile, we use the average background noise as its estimate $\widehat{\rho}$. With $\widehat{a}$, $\widehat{b}$, and $\widehat{\rho}$, i can predict $R(i,j,m_j)$ and $L(i,j)$ as:

$$\widehat{R}(i,j,m_j) = \widehat{a}_{ij} \cdot m_j + \widehat{b}_{ij} \qquad (4)$$
$$\widehat{L}(i,j) = \widehat{b}_{ij} - \widehat{\rho} \ . \qquad (5)$$

Without loss of generality, we henceforth label the neighbors according to their path loss such that

$$\widehat{L}(i,j) \le \widehat{L}(i,j+1),\ j = 1..N-1 \ .$$

It is easy to see that a neighbor with a higher node index has a higher path loss to the poller and can adapt within a lower range of response RSSI.

In the next two sections, we assume that the one-time measurement for RSSI prediction described above has been completed and that the $\{L(i,j)\}$ are amenable to the respective power assignment requirements imposed by the primitives. We will discuss techniques for combining the two primitives to deal with more general neighborhood topologies in Section 5.

3. LINEARPOLL

LinearPoll allows a poller to estimate the nodes in P_i. In this section, we will first formally analyze the preliminaries of LinearPoll and then present its detailed design, implementation and evaluation.

3.1 Preliminaries

The poller in LinearPoll estimates P_i by distinguishing different signals in the superposed response from its neighbors. The poller respectively assigns to neighbors response power and length such that the response RSSI of a neighbor is inversely related to its response length. Thus, the poller can estimate the set P_i by analyzing the drops in RSSI as it measures the superposed response signal. We formalize the inverse relationship between response RSSI and length as two requirements:

Requirement II (Sufficient difference, Δ, in RSSIs):
The power assignment $\{m_j\}$ is chosen such that for some

[1]Neighbor j can efficiently send its output power m to the poller by attaching to its packets a small control flag whose size depends on the number of available power levels.

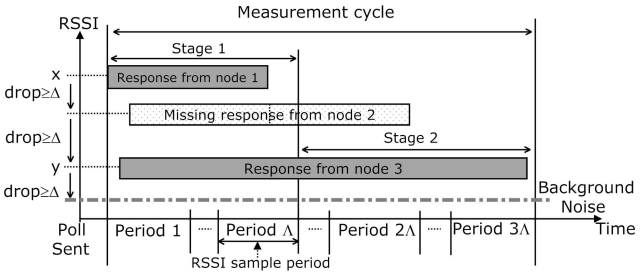

Figure 1: LinearPoll overview where 2 out of 3 neighbors have responded.

$\Delta > 2\sigma_i$ *for all* $j \in N_i$

$$R_l(i, j, m_j) - \rho_u \geq \Delta \,, \tag{6}$$

$$R_l(i, j, m_j) - R_u(i, j+1, m_{j+1}) \geq \Delta \,. \tag{7}$$

Essentially, (6) ensures that even the weakest response is above the upper bound of the noise floor. (7) ensures that lowest RSSI from the response of j is still greater than the highest RSSI from the response of $j+1$, which ensures that the poller sees a drop of at least Δ in the RSSI sequence whenever a response of higher strength terminates.

In addition to the unique response RSSI from each neighbor, for the poller to discriminate two component signals, we also let them differ in response length. Define $C(t)$ to be the number of RSSI samples the poller can obtain from a response packet of t bytes. Also define T and Λ to be the maximum packet size supported by the radio and the minimum number of RSSI samples the poller needs to identify a neighbor. We introduce the final requirement for LinearPoll:

Requirement III (Sufficient difference, Λ, in response lengths): Λ *and the response length assignment* $\{t_j\}$ *are chosen such that for* $j = 1, \ldots, N-1$

$$C(t_{j+1}) - C(t_j) = \Lambda \tag{8}$$

$$C(T) \geq N\Lambda \,. \tag{9}$$

The exact values of Δ and Λ depend not only on the accuracy and efficiency requirement for LinearPoll but also on the radio platform being used. In the rest of the section, we first outline the LinearPoll protocol and then detail its individual modules. Based on our implementation of LinearPoll on the TelosB platform, we discuss the platform specific constraints on choosing Δ and Λ in Section 3.3. We conclude with an experimental evaluation of the performance of LinearPoll.

3.2 LinearPoll Design

The correctness of LinearPoll depends on Requirements I, II, and III being met. While the first two requirements depend on both the wireless environment and the radio hardware, the last one depends only on the radio.

Fig. 1 illustrates one particular round of LinearPoll in which two of the three neighbors respond to a poll. In our example, neighbor 2 has not responded and its missing response is represented by a dotted line in contrast to the two gray bars that represent the responses from neighbor 1 and 3. Here the entire duration of the superposed response constitutes a *measurement cycle*. Each measurement cycle consists of multiple *stages*, in which each stage uniquely corresponds to a responder. Each stage lasts a multiple of Λ *RSSI sample periods*, in which one RSSI value is obtained at the end of each period. For each stage, one *measure* is computed based on the samples in that stage.

Each round of LinearPoll consists of a one-time step followed by two routine steps:

Step 1 (One-time-only: Assign Power and Length): To each neighbor j, i assigns a response power and length, according to Algorithm 1 to be described in Section 3.2.1. As seen in Fig. 1, the assingment pattern is that a stronger response has a shorter length. The power assignment ensures that the response RSSI from any two neighbors are at least Δ dB apart and the number of samples a response lasts is a multiple of Λ.

Step 2 (Measure RSSI): Poller i broadcasts its poll and immediately acquires RSSI samples at the highest possible rate supported by the radio.

Step 3 (Estimate P_i): Given the RSSI samples, node i groups them into stages, computes the measure of each stage, and estimates P_i according to Algorithm 2. As seen in Fig. 1, every time the response of a neighbor terminates, a drop in RSSI can be detected. In our example, given the two drops before the RSSI decreases to the noise floor, along with the timing of these drops, the poller can identify the stages that correspond to neighbor 1 and 3. Further details are discussed in Section 3.2.3. □

Once initialized, LinearPoll can skip the first two steps until Requirement I no longer holds, i.e., until the estimate of $\widehat{R}(i, j, m)$becomes invalid viz-a-viz (1) in Requirement I. As one illustration of the rate at which the first two steps may be needed, we note that in our experiments, LinearPoll produced highly accurate estimates of P_i even if the same assignments were used for 15 minutes.

3.2.1 Response Power and Length Assignment

Algorithm 1: Power and length assignment

```
1 m_N ← argmin_{m∈M} (R_l(i, N, m) − ρ_u ≥ Δ)
2 for j ← N − 1 to 1 do
3     for k ← 1 to |M| do
4         if R_l(i, j, M_k) − R_u(i, j + 1, m_{j+1}) ≥ Δ then
5             m_j ← M_k
6             t_j ← C^{−1}(Λ × (N − j))
```

Algorithm 1 computes the assignment $\{m_j, t_j\}$ for each neighbor j using a simple rule: the smaller the estimated path loss for j, the higher the assigned power and shorter the assigned response length.

Line 1 assigns to node N the minimum power such that $R_l(i, N, m_N)$ is at least $(\rho_u + \Delta)$, which ensures (6) for all nodes. For the remaining nodes, from node $N-1$ down to node 1 of decreasing path loss (Line 2), it searches for the minimum power for neighbor j that is at least Δ dB above the one just assigned to neighbor $j+1$. To find such a power, it searches for the lowest power that satisfies (7) (Line 2, 3, and 4) and assigns length that satisfies (8) accordingly (Line 6). Note that C^{-1} is the inverse function of C, which returns the response length in bytes in order to obtain the given number of samples from that response. Thus, Algorithm 1 ensures that both Requirement II and III hold.

Two critical parameters need to be chosen carefully with respect to the specific wireless environment and radio plat-

form to which LinearPoll is applied. Δ controls the minimum gap in RSSI between any two responses, while Λ controls the number of samples per stage. Below, we discuss how Δ and Λ are chosen for the testbeds where we ran our experiments.

Choosing Δ . Polling nodes measure σ_i to choose Δ accordingly. In our experiments, the weakest response is about 10 dB above the noise floor. In terms of absolute value, variation in noise will not create any visible change in the response RSSI. However, because CC2420 reports RSSI in 1 dB resolution, the slightest change in received signal strength can lead to a change of 1 dB due to rounding. For instance, we observed in our experiments that for any given neighbor j, $R(i,j,m_j)$ almost never differ by more than 1 dB from $\widehat{R}(i,j,m_j)$. It follows from (1) that σ_i is 0.5 dB and thus we thus set Δ to 2 according to Requirement II.

Choosing Λ . 802.15.4 specifies that an RSSI sample period lasts 8 symbols, which introduces a complication: the boundary of a response need not be aligned with the boundary of a sample period. A *transitional sample* is created between two stages when a sample period contains some signal from the earlier stage and some from the later stage. As a result, its RSSI is lower than that of the earlier stage but above that of the later one. As a stage can have transitional samples at both of ends, it must last at least 3 samples for its measure (the median) to reflect the actual response RSSI.

3.2.2 Interference Detection

Before P_i can be estimated, we need to verify that the RSSI samples obtained in the measurement cycle are not contaminated by interference. Just like in error detection codes, such as Cyclic Redundancy Check (CRC), where a receiver rejects a received packet if its bits leads to a incorrect CRC, LinearPoll rejects a measurement cycle if the layout of the stages does not match what can be generated from the power and length assignment.

LinearPoll utilizes two mechanisms to detect interference. First, it exploits the fact that the response length is known *a priori*. Given an assignment, RSSI drops occur only at a fixed set of samples in the measurement cycle. Although software and hardware jitters can vary the exact moments of drops, the variation is seldom more one sample. Thus, the poller can reject a measurement cycle if unexpected drops are observed. Second, measurements should be monotonically decreasing and they should be around the expected response RSSI determined by the power assignment algorithm.

A measurement cycle is rejected if it fails either of these tests. Otherwise, LinearPoll proceeds to estimate P_i based on the verified measurement cycle.

3.2.3 Algorithm for Estimating P_i

Right after the poller broadcasts a poll, it samples RSSI at the highest available rate. Let $\{r_j\}$ be the list of samples in a measurement cycle. A measurement cycle starts with the first sample that is at least Δ dB above than ρ_u and ends with the first sample that drops back below ρ_u.

To estimate P_i, the poller needs to first discover a responder and then identify that responder. The poller discovers a responder whenever it sees a drop in measure that is no less than Δ. Further, the poller identifies the responder based on the drop's timing in $\{r_j\}$ or, equivalently, the point of termination of a stage.

Algorithm 2: P_i estimation

```
1 P̂_i ← {}
2 lastTrigger ← getMedian(r[1 : Λ])
3 for k ← Λ + 1;  k ≤ |r|;  k ← k + Λ do
4     curRss ← getMedian(r[k : min(|r|, k + Λ − 1)])
5     drop ← lastTrigger - curRss
6     if drop ≥ Δ then
7         responder ← findResponder({t_j}, k)
8         P̂_i ← P̂_i + {responder}
9         lastTrigger ← curRss
```

Algorithm 2 takes the assignment from Algorithm 1 as input and estimates P_i. $\widehat{P}_i$, the estimate of P_i, is initialized to an empty set (Line 1). *lastTrigger* represents the RSSI that triggers the last discovery and is initialized to the median of the samples[2] of the first stage (Line 2), where notation $r[a:b]$ represents the sub-array of $\{r_j\}$ from index a to b.

The number of samples that a stage lasts is a multiple of Λ, per (8). By dividing $\{r_j\}$ into sample groups of size Λ, we ensure that a stage arrives only at the boundaries of each sample group. Therefore, the `for` loop increments by Λ (Line 3). Line 4 computes the stage measure based on the samples from the current sample group. In Line 5 and 6, a new responder is discovered if the measure of the current stage is Δ lower than *lastTrigger*. After a new responder is discovered, we identify the responder via utility function `findResponder`() (Line 7). Given the possible response lengths $\{t_j\}$ provided by Algorithm 1, `findResponder`($\{\mathtt{t_j}\}, \mathtt{k}$) searches for the neighbor j whose response terminates closest to the k^{th} sample in $\{r_j\}$:

$$\texttt{findResponder}(\{t_j\}, k) = \underset{j \in N_i}{\operatorname{argmin}}\{|C(t_j) - k|\} \ .$$

Finally, we update $\widehat{P}$ and $lastTrigger$ accordingly.

3.3 Implementation Details

Here we describe our implementation of LinearPoll on the CC2420 radio-equipped TelosB platform, beginning with its response mechanism.

Since the responses are of different lengths, we cannot use the software or hardware acknowledgement features provided by CC2420. As a result, neighbors send explicit packets as response. To minimize jitter in transmission, neighbors preload the response packet before they receive the probe. Upon receiving the poll, responders send a strobe to the radio chip, which will then immediately starts the transmission.

The CC2420 radio samples signal strength at 62.5Khz and reports RSSI based on the mean value averaged over 8 symbol periods. Theoretically, we can obtain one RSSI measurement every $16\mu s$. However, given TelosB hardware limitations, we can only obtain a new RSSI reading every $113\mu s$. Given that the CC2420 radio is 250 Kbps, the number of samples $C(t)$ that we can obtain from a packet of t bytes is $C(t) \approx t/4$ samples.

[2]We empirically discovered that using the median of the sample group, as compared to using max, mean, and min, delivers better accuracy.

3.4 Experimental Evaluation of LinearPoll

We now describe our evaluatation of LinearPoll on the TelosB mote platform. After we describe the experiment setup, we introduce a baseline protocol for counting (identifying) and then empirically compare its performance with that of LinearPoll.

Experiment Setup. We conducted our evaluation on two 9-node arrays, one in Testbed 1 located in an office building and the other in Testbed 2 located in an industrial building shared by many occupants. LinearPoll was evaluated in both testbeds and the baseline protocol was evaluated in Testbed 2. We used channel 26 in Testbed 1 and channel 25 in Testbed 2.

While Testbed 1 is almost free-of-interference, Testbed 2 sees sporadic clusters of interference in channel 25 due to active 802.11 connections and other coexisting sensor networks. Even though the likelihood of seeing interference in a particular sample is small, the probability that a measurement cycle in LinearPoll sees interference is non-trivial as a cycle can span 40 samples. To quantify this, we collected a trace of measurements in Testbed 2 by acquiring a batch of 80 RSSI samples as fast as possible every 2 seconds before each experiment. Interference is identified whenever a sample is 3 dB above the noise floor, which accounts for 1.5% of all samples. The likelihood that there is at least one interference sample in 20, 40, and 60 consecutive samples is 3.2%, 5.3% and 7%, respectively. Therefore, the longer a measurement cycle lasts, the more likely it is affected by interference.

One node in each array is designated as the poller. In all the experiments, we set $\Delta = 2$. Each experiment proceeds in rounds and in each round the poller broadcasts a poll and estimates P_i. The poller broadcasts two polls per second and an experiment finishes when 2000 rounds are completed (16 minutes). We tested different values of Λ (2, 3, 4) and 2 experiments were run for each value. $\Lambda = 2$ was tested to verify the requirement of $\Lambda \geq 3$ as discussed in Section 3.2.1[3].

The set of responders was changed from round to round and all subsets of neighbors were evaluated.

In the beginning of every experiment, each neighbor sent to the poller 20 packets containing their power information for the poller to perform the one-time measurement for RSSI prediction as described in Section 2. *Step 1 in LinearPoll, i.e., power and length assignment, was done only once based on the measurements in the beginning of each experiment and neighbors used the same assignment until the experiment finished.* Note that the LinearPoll is expected to be initialized with regular data traffic. Also, an experiment is stopped after 16 minutes because of limited disk space to store RSSI samples rather than degraded accuracy. Even in the worst case where dedicated control traffic has to be used to re-calibrate LinearPoll every 16 minutes, the control overhead is less than 0.2% even for a neighborhood of 50 nodes as each back-to-pack control packet takes at most 2 ms to transmit on the CC2420 radio. The same remark also applies to LogPoll.

[3]Because a stage can last as short as 2 samples for $\Lambda = 2$, we used the mean, instead of the median, as its measure of a stage.

Λ		Testbed 1		Testbed 2	
		e_a	$e_r(\%)$	e_a	$e_r(\%)$
$\Lambda = 2$	$\lvert P_i \rvert$	0.8	34%	0.87	39%
	P_i	1.2	67%	1.65	71%
$\Lambda = 3$	$\lvert P_i \rvert$	0.044	4.4%	0.039	3.5%
	P_i	0.047	5%	0.043	4.5%
$\Lambda = 4$	$\lvert P_i \rvert$	0.011	1.1%	0.018	1.6%
	P_i	0.011	1.1%	0.018	1.6%
Baseline	$\lvert P_i \rvert$	N/A	N/A	0.06	3.9%
	P_i	N/A	N/A	0.065	4%

Table 1: Average error amount, e_a, and error rate e_r, for estimating P_i and $|P_i|$ with $\Delta = 2$ dB.

3.4.1 Baseline for Linear Counting

To evaluate the performance of LinearPoll, we implemented a baseline RSSI-based counting protocol according to the TDMA scheme described in [2]. In the baseline, each neighbor is assigned a unique time slot in which they respond to the poll. All neighbors respond at the highest power level to ensure detection. The poller can then identify the set of responders based on the slots that it detects energy above the noise floor[4].

3.4.2 Performance of LinearPoll

Table 1 summarizes the results. First let us define *error amount* and *error rate*. The error amount, denoted as e_a, for estimating P_i and $|P_i|$ is defined respectively as:

$$e_a = \begin{cases} |P_i \backslash \widehat{P}_i| + |\widehat{P}_i \backslash P_i| & \text{, for } P_i \\ ||P_i| - |\widehat{P}_i|| & \text{, for } |P_i| \end{cases}, \tag{10}$$

where $'\backslash'$ denotes the set complement operation; error rate, denoted as e_r, is the percentage of rounds whose e_a is greater than zero.

Let us first look at the impact of Λ. Overall, the accuracy improves as Λ increases, especially when Λ increases from 2 to 3. The high error rate for $\Lambda = 2$ is because of the difficulty in delineating RSSI stages, which verifies that *the minimum value of Λ is 3.* e_r drops substantially as Λ increases above 3 and reaches as low as 1.1% when $\Lambda = 4$.

Now we turn our attention to the cases where $\Lambda \geq 3$. The accuracy of P_i is almost identical to that of $|P_i|$, which indicates that finding the identity of an responder based on the length of its response is reliable. Moreover, we can see that e_r is almost identical to e_a, which indicates that *the number of nodes that LinearPoll overestimates or underestimates is almost never more than 1.*

Compared to the baseline, we see two advantages of using LinearPoll. Because there is no silent slot in LinearPoll, the poller can turn off its radio to save energy once the last responder finishes. In the baseline, however, the poller has to wait until the last slot. Assuming every subset of neighbors are equally likely to respond, the poller on average saves 50% of energy in LinearPoll. The second advantage of LinearPoll

[4]We did not rely on data communication in each slot for counting because this paper focuses on RSSI-based counting. Moreover, the small accuracy improvement of data communication incurs high energy cost due to limits on packet reception rate. The RSSI-based scheme requires a slot as short as 512 μs, whereas data communication requires a slot as long as 2ms for TelosB.

lies in its resilience to interference. While the poller in LinearPoll utilizes a priori knowledge about possible RSSI drops and their timings to reject interference, the baseline protocol identifies responders only by the presence of energy in their respective slots. Therefore, the baseline is more susceptible to interference. Our evaluation of the baseline protocol in Testbed 2 shows an error rate of 4%, which indicates that the baseline not only consumes more time and energy but also provides inferior accuracy as compared to the 1.6% error rate of LinearPoll.

4. LOGPOLL

LogPoll allows a poller to estimate the number of neighbors where predicate P holds on a logarithmic scale. In this section, we first analyze the basics of LogPoll and then present its detailed design, implementation and evaluation.

4.1 Preliminaries

The core idea is that the poller searches for some target RSSI Υ according to which each neighbor j adapts its power m_j such that $\widehat{R}(i, j, m_j) = \Upsilon$, i.e., the individual response RSSI from every neighbor is adjusted to be Υ. Such a power assignment, along with identical response lengths, ensures that the measure of the single stage of the superposed response from 2^u neighbors is roughly $\Upsilon + 3u$. Now, given a measure of r, the poller estimates $\log|P_i|$ by finding the $\widehat{u}$ that minimizes $|\Upsilon + 3\widehat{u} - r|$ and returns $\widehat{u}$ as the estimate of $\log|P_i|$.

To see why this sort of counting is accurate, consider how the received power of the superposed signal grows with the number of responses. For $2n$ signals that have the same frequency and the same amplitude A, their superposition has an amplitude of $2nA$ if their interference is entirely constructive, and an amplitude of 0 if their interference is entirely destructive. Because signal strength is proportional to the square of amplitude, the superposed signal from $2n$ sources is respectively 4x or 0x that of n signals in these extremal cases. The average case of the superposed signal from $2n$ sources has an expected strength that is twice, or 3 dB above, that of n sources. Hence, for 2^u responders that share the same response RSSI Υ, the superposed signal strength will be $3u$ dB above Υ.

More precisely, in addition to Requirement I, the following requirement is also needed for LogPoll to be accurate:

Requirement IV (Similar RSSIs, to Υ): *There exists a target RSSI Υ and a power assignment $\{m_j\}$ such that for $j = 1, 2, \ldots, N$:*

$$\widehat{R}(i, j, m_j) \in (\Upsilon - 1.5,\ \Upsilon + 1.5),$$

Proposition 1 implies that the RSSI of the superposed response increase by 3 dB whenever we double the number of responders. The following proposition provides the theoretical basis for the correctness of LogPoll. Let $s(u)$ be a node group of size 2^u. Let $\widehat{R}(i, s(u), \Upsilon)$ be the expected response RSSI at poller i of the superposed response from $s(u)$ whose target RSSI is Υ.

Proposition 1. *Given Requirement IV, the following holds for $u = 0, 1, 2, \ldots$ and $\forall\ s(u) \in N_i$:*

$$\widehat{R}(i, s(u), \Upsilon) \in (\Upsilon + 3u - 1.5,\ \Upsilon + 3u + 1.5) \qquad (11)$$

Proof. We prove by nested induction. For $u = 0$, (11) follows trivially from Requirement IV. For the base case of the induction, we consider $u = 1$, i.e., for $s(1) = 2$ responders, and prove the following:

$$\widehat{R}(i, s(1), \Upsilon) \in (\Upsilon + 3 - 1.5,\ \Upsilon + 3 + 1.5)$$

Let $A(x)$ be the peak amplitude of a signal whose power is x dB. Given two signals of the same frequency that respectively have amplitudes of $A(x_1)$ and $A(x_2)$, we can compute the instantaneous power of their superposed signal, denoted as $W(t, A(x_1), A(x_2))$, at time t as:

$$\begin{aligned} &W(t, A(x_1), A(x_2)) \\ =&Q\left(A(x_1)\sin(2\pi f \cdot t) + A(x_2)\sin(2\pi f \cdot t + \delta)\right)^2\ , \end{aligned}$$

where Q is some constant, f is the frequency of the signal, and δ is phase offset between the two signals. The maximal RSSI of the superposed imposed signal is achieved when both component signals have an RSSI of $(\Upsilon + 1.5)$, the maximal increase (in dB) from one responder to two is

$$10 \cdot \log \frac{E[W(t, A(\Upsilon + 1.5), A(\Upsilon + 1.5))]}{E[W(t, A(\Upsilon + 1.5), 0)]} < 3 \text{ dB}\ ,$$

hence

$$\widehat{R}(i, s(1), \Upsilon) < \Upsilon + 1.5 + 3\ .$$

We can likewise verify that the superposed signal reaches its minimal RSSI of $(\Upsilon + 1.5)$ when both component signals have an RSSI of $(\Upsilon - 1.5)$. Hence, we have proved (11) holds for $u = 1$.

Given that the two component signals have the same frequency f, the superposed signal will have the same frequency. If we consider the superposed signal of the two as one virtual signal whose target RSSI is $\Upsilon + 3$, Requirement IV inductively holds for the virtual signal. Now, for any $u > 1$, we can construct its proof by iterating the proof above from $u = 1$ as a superposed signal of 2^{u+1} component signals can be considered as one that consists of two virtual signals whose target RSSI is $\Upsilon + 3u$ and each of which is itself the superposed signal of 2^u component individual signals. The induction hypothesis thus holds. □

Next, we overview the LogPoll protocol, and describe how Υ is chosen and how the poller can estimate $\log|P_i|$ based on Proposition 1.

4.2 Design of LogPoll

Given the one-time measurement for response RSSI prediction, LogPoll follows the same three steps as LinearPoll, except that the algorithm for response power and length assignment used in step 1 and the algorithm for estimation used in step 3 are respectively replaced with the schemes we present in Section 4.2.1 and 4.2.3.

4.2.1 Response Power and Length Assignment

All neighbors are assigned the same response length. There exists a tradeoff between the response length (and hence energy cost) and accuracy which we will study empirically in Section 4.3. As for power assignment, the poller searches for the target RSSI Υ that all neighbors can adjust to within their power ranges. According to (4), the poller can compute for each neighbor j its power to adapt $\widehat{R}(i, j, m_j)$ to Υ.

Recall that M is the set of available output powers and that nodes are labeled in the order of increasing path loss. Based on (4) we define α as the highest expected RSSI that the neighbor with the largest path loss (node N) can reach, and β as the lowest expected RSSI that the neighbor with the lowest path loss (node 1) can reach:

$$\alpha = \widehat{R}(i, N, m_N) + \widehat{a}_{iN}(M_{max} - m_N),$$
$$\beta = \widehat{R}(i, 1, m_1) - \widehat{a}_{i1}(m_1 - M_{min}).$$

Further, we define $\omega(r, j)$ as the smallest gap that node j can adjust $\widehat{R}(i, j, m_j)$ to a given value r:

$$\omega(r, j) = \min_{m \in M} \left(|\widehat{R}(i, j, m) - r| \right).$$

Obviously, Υ does not exist if $\alpha < \beta$; If $\alpha \leq \beta$, we set Υ to the value that minimizes the sum of $\omega^2(\Upsilon, j)$ over all neighbors:

$$\Upsilon = \operatorname*{argmin}_{r \in [\beta,\ \alpha]} \left(\sum_{j \in N_i} \omega^2(r, j) \right).$$

Finally, the power assignment for j is computed as:

$$m_j = \operatorname*{argmin}_{m \in M} |\widehat{R}(i, j, m) - \Upsilon|$$

4.2.2 Interference Detection

Similar to LinearPoll, we first verify that the measurement cycle is not contaminated by interference before P_i is estimated. Because all responses in LogPoll are of the same length, the number of samples that can be obtained in a round is known a priori. Also, samples in a measurement cycle should share similar RSSI. The poller can thus reject a measurement when (i) the number of samples is different from its expected value by more than some threshold.[5] or (ii) there is inconsistency in RSSI values within a measurement cycle.

4.2.3 Algorithm to Estimate P

In LogPoll, as each measurement cycle contains a single stage, which eliminates transitional samples, we use the mean of the samples, rather than the median as in LinearPoll, as the measure of this single stage because it provides better resolution.

Now given a RSSI measure r, LogPoll estimates the number of responding neighbors with function $F_\Upsilon(r)$.

$$\log |P_i| = F_\Upsilon(r) = \operatorname*{argmin}_{u \in dom(\theta_\Upsilon)} \{|\theta_\Upsilon(u) - r|\} \qquad (12)$$

where θ_Υ is a discrete function that maps the number of responders in log scale to the expected strength of their superposed signal given a target RSSI of Υ. According to Proposition 1, we have

$$\theta_\Upsilon(u) = \Upsilon + 3u, \quad u = 0, 1, 2, \ldots \qquad (13)$$

In other words, node i uses the value u that makes $\theta_\Upsilon(u)$ closest to r as the estimate of $\log |P_i|$.

4.3 Implementation Details

Instead of detecting relative drops in RSSI in a measurement cycle as in LinearPoll, LogPoll estimates $\log |P_i|$ solely based on mean RSSI of the samples obtained. Hence, the accuracy of LogPoll depends heavily on how well the poller can measure the RSSI of the superposed response. Naturally, the more samples the poller can get from a measurement cycle, the more accurately the sample mean reflects the true RSSI of the response. We empirically observed that responses that are 32 bytes long provide almost the same accuracy as 128-byte long responses. Hence, neighbors use 32-byte responses in our implementation.

Also, we introduce a heuristic that can more accurately determine whether there is only one responder. Let us define the *peak-to-peak variability* of a measurement cycle to be the difference between the maximal sample and the minimal one. When there is only one responder, the peak-to-peak variability is seldom above 1 thanks to the absence of interference from other concurrent responses. When there are 2 or more responders with similar response RSSIs, the difference between the minimum and maximum in a measurement cycle is almost always greater than 1. Therefore, LogPoll exploits the variation in RSSI to help decide wether the number of responders is 1.

4.4 Experimental Evaluation of LogPoll

We now describe our evaluation of LogPoll on the TelosB mote platform: We begin with the experiment setup, then introduce a baseline for log scale counting, and finally empirically compare the baseline with LogPoll.

Experiment Setup. We tested LogPoll under different network conditions on three arrays in two testbeds. Two 63-node arrays, namely array A and B, are in Testbed 2 and a 43-node array, namely array C, is in Testbed 3. On each array, one node is designated as the poller. While array A represents an ideal neighborhood where all nodes satisfy Requirement IV and can choose a target RSSI above the external interference in Testbed 2, array B and array C represent different challenging network conditions. On array B, 10% of nodes do not meet Requirement IV due to the limited granularity in output power control on the TelosB platform; on array C, although the neighbors honor Requirement IV, they are far away from the poller and as a result they have to choose target RSSI lower than the external interference in Testbed 3.

Two experiment are conducted on each array and experiments on any given array share very similar results from run to run. In each experiment, we tested all groups of sizes 1, 2, 4, 8, 16, and 32 alternatively. The experiment finishes once each node group has been tested for 100 rounds.

In the beginning of every experiment, each neighbor sends to the poller 20 packets containing their power information for the poller to perform the one-time measurement for RSSI prediction as described in Section 2. *Step 1 in LogPoll, i.e., power and length assignment, is done only once based on the measurements in the beginning of each experiment and neighbors use the same assignment until an experiment finishes.*

4.4.1 Baseline for Log Scale Counting

To evaluate the performance of LogPoll, we implemented a baseline RSSI-based counting protocol based on a simplified group-testing as proposed in [2, 6]. Each round of the baseline takes multiple steps. In each step k, each responder has a probability of 2^{-k} to respond. A round finishes when no responder respond in a step. Based on the number of

[5]Empirically, we set the threshold to 1 for TelosB.

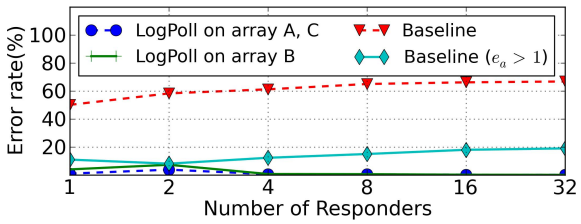

Figure 2: Comparison of error rates of LogPoll and the baseline protocol.

steps n in a round, the poller can estimate the number of responder in that round to be $\min(N, 2^n)$, where again N is the number of poller's neighbors. All neighbors respond at the highest power level to ensure detection.

4.4.2 LogPoll Performance

Fig. 2 shows the LogPoll results. For a set of responders P_i and its estimation $\log|\widehat{P}_i|$, error amount e_a is defined as $e_a = |\log|P_i| - \log|\widehat{P}_i||$ and error rate is the percentage of rounds where $e_a > 0$.

When Requirement IV is honored (as on array A and C), the accuracy of LogPoll for node groups of all sizes except for 2 is close to 100% regardless of the value of Υ; the accuracy for node groups of size 2, albeit worse, is still over 96%. The reason that the accuracy of node group of size 2 suffers is because of its above-average variance in received signal strength: as we will see in section 4.4.3, the RSSI measure for node groups of size 2 varies from its expected RSSI more than node groups of other sizes.

Although the LogPoll protocol is general and can handle a broad class of topologies and densities, the radio hardware can be a constraint. To study the impact of limited range in power levels, we repeated the experiment on array B where 10% of neighbors have a response RSSI that diverges 1.5 dB to 3 dB from Υ, which fails to meet Requirement IV. We call these 10% of nodes *bad neighbors*. The accuracy is most impacted when there are no more than 2 responders because the superposed RSSI of node groups of small sizes will diverge considerable from their expected value if one or more of their members are bad neighbors. However, when there are 4 responders or more, the existence of bad neighbors is unlikely to lead to incorrect estimations because we are counting in log scale. *LogPoll becomes more resilient to coarse granularity in output power control as the number of responders increases.*

The baseline protocols suffers in terms of both accuracy and energy cost. The baseline protocol makes an error greater than 0 at least 50% of the time (as shown by the red dash line) and an error greater than 1 at least 7% of the time (as shown by the cyan line). What's worse, its accuracy decreases the number of responders increases. In terms of energy cost, each round in the baseline takes approximately $\log|P_i|$ steps and each steps takes about 4ms. On the contrary, LogPoll takes a constant time of about 4 ms.

4.4.3 Empirical Study of Proposition 1

LogPoll estimations are based on (13), whose correctness depends on Proposition 1. In this section, we empirically verify Proposition 1 and discuss why the performance of LogPoll is in general better as number of responders increases.

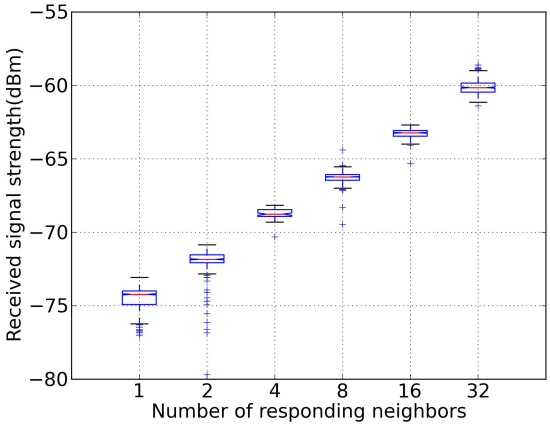

Figure 3: The distribution of RSSI measures for different numbers of responders on array A

According to Proposition 1, the average RSSI of the response 32 responders should be 15 dB above that of 1 responder. Our results show that the gaps in average RSSI between 32 responders and 1 responder is 14.8, 15.2 and 14.7 results on array A, B, and C respectively, which verifies our proposition.

Fig. 3 shows the distribution of the RSSI measures for node groups of different sizes for experiment on array A. Each vertical blue box shows the distribution of RSSI measures of a given node group size. The red line is the median and the edges of the blue boxes denote the 25th and 75th percentiles. The two black whiskers cover a range of $[\mu - 2.7z,\ \mu + 2.7z]$, where μ and z are the mean and standard deviation of the data. For normally distributed data, the two whiskers have a coverage of 99.2%. Outliers are plotted individually using '+'. As we can see, the variance of the RSSI of the superposed signal decreases as the number of responders increases.

We observe that *the variance in response RSSI decreases decreases as the number of responders increases.* Due to the limited granularity in output power control, the mean RSSI for a single node can be 1.5 dB away from Υ and thus it can be as much as 3 dB apart from that of another neighbor. As a result, the variance for small groups is high, which explains why the accuracy for groups of size 2 or less suffers most, as shown in Fig. 2. As $|P_i|$ increases, the RSSI of the superposed response converges to $\Upsilon \cdot |P_i|$, which reduces the variance. Results on arrays B and C show similar trends of diminishing variance as the number of responders increase.

5. CONSTRAINTS AND EXTENSIONS

5.1 Handling Malicious Interference

Although non-trivial external interference was present in all of our LinearPoll and LogPoll experiments on Testbed 2, it did not noticeably degrade performance because of two reasons: (i) it is unlikely that the RSSI of the random interference resembles the RSSI of a valid responder; (ii) it is unlikely that the interference to start and end within within the polling window of 4 ms.

In this section, we introduce a malicious interferer F which attempts to make its interference resemble the response of a valid neighborin terms of timing, duration and RSSI. F not only transmits at the same time as the responders for a duration no longer than the longest valid response, it also adapts its power m_F such and $R(i, j, m_j) \approx R(i, F, m_F)$ for some $j \in N_i$.

We configured F to inject interference every 10 rounds and evaluated LinearPoll on Testbed 2 again with Λ set to 4. Although F interferes 10% of the time, the error rate increases only from 1.6% to 4.4% and error amount increases from 0.018 to 0.05.

In LogPoll, since the RSSI from I is similar to that of others and we are estimating P_i in log scale, we find that interference only leads to errors when $P_i \leq 4$. Also, F succeeds to create error only when its length is close to that of others. In a similar setting as LinearPoll above, we repeated LogPoll on array A. The error rate for two responders or less increases from 3% to 5%. Due to its very limited impact on accuracy, we skip further empirical study of the impact of smart interference on LogPoll.

5.2 Handling Topology Constraints

The limited range and granularity of the output power provided by radio hardware lead to constraints in topology. Given the RSSI measurement precision of the CC2420 radio, LinearPoll requires that the response RSSI's from any two neighbors be separated by at least 3 dB. In a topology benign to LinearPoll, neighbors have fairly different path losses and as a result the separation in response RSSI can be fulfilled even if the radio provides very limited output power levels; In the worst case where neighbors share very similar path losses, LinearPoll has to solely rely on the radio to provide the RSSI differentiation. For example, with the CC2420 radio, only 6 out of 8 output powers are at least 3 dB apart. If more than 6 neighbors share very similar path losses, LinearPoll should adopt the grouping technique that we will shortly introduce. Opposite to LinearPoll, LogPoll requires all response RSSI's to be close to some poller-dependent target RSSI Υ. In the worst case where neighbors have very different path losses, LogPoll would also have to rely on the radio to fulfill its requirement.

In general, users should take advantage of the complementary use cases of the two primitives and choose according to application needs and the radio hardware capabilities. In challenging cases where the precision of LinearPoll or the scalability of LogPoll is desired but their corresponding requirement cannot be satisfied given the specific topology, one can resort to the *vertical* or *horizontal* grouping.

In vertical grouping, we divide the neighborhood into multiple groups wherein members of each group have diverse path losses; in horizontal grouping, neighbors with similar path losses are grouped together.

Now, if the precision of LinearPoll is desired but Requirement II is not met, vertical grouping can be applied to ensure Requirement II holds in each group. If the scalability of LogPoll but Requirement IV cannot be satisfied, horizontal grouping can be applied to ensure Requirement IV holds in each group.

The information needed by either grouping scheme is already provided by the path loss estimation discussed in Section 2. Also, decomposing neighbors into groups incur minimal energy overhead on the poller, i.e., one polling message for each group, and no additional cost to the responding neighbors. We delegate evaluation of grouping to future work.

5.3 Beyond 802.15.4 Radios

For accurate counting, both primitives require that the superposition of component signals results in predicable signal strength. Were there to be highly constructive or destructive interference between component signals, the signal strength variability would increase (and the predictability would decrease). Randomness in the component signals helps avoid highly constructive or destructive superposition; this randomness could result from software jitter during response transmission but also by scripting the response to use random bits and by leveraging the physical layer properties of different radio technologies. Depending on the modulation scheme, the random bits in the packet will be modulated into random phases, amplitudes, or frequencies. As long as the poller averages RSSI samples over a period in which multiple bits are transmitted, it can still obtain a a good assessment of the mean RSSI of the superimposed signal. Thus, although we have only validated our primitives for a 802.15.4 radio, we argue that they are applicable to radios that follow other physical layers standards.

6. RELATED WORK

In-network processing approaches [9,11,17] typically focus on node-based function computation to reduce the amount of communication needed. In contrast, the processing in this paper exploits collaborative communication to reduce the overall complexity of (counter) function computation on the nodes.

In [2], the performance of three response collection methods — polling, group testing, and TDMA — is compared. As both polling and group testing require multiple rounds of communication for a single estimation, these methods are less efficient than LinearPoll and LogPoll. The TDMA approach, in which each neighbor is assigned a unique slot during which it sends back response, is closest to LinearPoll in terms of precision and energy cost. However, a poller using the TDMA approach cannot terminate its measurement cycle until the last assigned slot, whereas a poller in *LinearPoll* can terminate its measurement cycle as soon as the last responder finishes its response. On average, LinearPoll is 2x as efficient as the TDMA approach.

In [5], Demirbas et al propose PollCast for a poller to estimate with constant cost whether $|P_i|$ is 0, 1, or greater than 1 based on Channel Clear Assessment (CCA). The poller detects no channel activity if $|P_i| = 0$, receives a response packet if $|P_i| = 1$, and detects collisions if $|P_i| \geq 1$. In comparison, LinearPoll not only counts but also identifies in a single round of communication.

In [6], *tcast* extends PollCast and estimates whether $|P_i|$ is greater than some threshold t at a cost of $O(2t \cdot (\log \frac{N_i}{2t}))$ in the worst case. In comparison, LogPoll estimates $\log |P_i|$ at a constant cost.

Dutta et al have proposed a simultaneous transmission scheme [7] that provides the same counting capability as LinearPoll. Their scheme identifies the set of responders, albeit it exploits the properties in Orthogonal Frequency Division Multiplexing (OFDM) modulation that is more suited for technologies with higher power radios such as IEEE 802.11 WLAN and WiMax. Although their scheme needs only a few

symbols of response, it is computationally more expensive than LinearPoll as it requires Fourier transformations. Cross platform performance is difficult to compare, that said, its reported accuracy numbers appear to be similar to those of LinearPoll. While SMACK was only evaluated on a 3-node array, our primitives has been evaluated at scale in different testbeds. Also, their proposed scheme is limited by the number of available subcarriers as each neighbor require a unique subcarrier. One can relax such limitation by extending SMACK to utilize our primitives to detect the existence of multiple responders in each sub-carrier.

Our primitives can be readily integrated into existing MAC protocol designs. A-MAC [8] is a receiver-initiated MAC protocol that allows the receiver to quickly determine whether there is any neighboring node with pending traffic. It exploits the constructive interference of identical hardware-generated acknowledgement packets, which allows the receiver to successfully receive a superimposed acknowledgement with high probability. In StrawMAN [13], senders contend for the medium by simultaneously responding to the receiver's probe with request packets of random lengths and the sender with the longest request packet wins the medium. In both MAC protocols, either LinearPoll or LogPoll can be integrated for the receiver to gain more information about its neighborhood at no extra energy cost.

The RSSI reading on sensor motes is not always accurate and sometimes require calibration. Chen et al showed that the RSSI reading on TelosB motes have a constant offset compared to a reference curve [4]. Although this is important for protocols whose correctness depends on RSSI values reported from different nodes, the RSSI readings in our work are measured and used locally by the poller. As a result, our primitives are robust with respect to the RSSI offset.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we investigated the problem of estimating neighborhood counts via collaborative communications. We showed that by carefully choosing the response packet length and output powers for neighbors, it is possible to encode information in their simultaneous responses to poll requests. We designed LinearPoll and LogPoll that allow the poller to collect the neighborhood metric at different precisions and asymptotic costs.

LinearPoll allows the poller to estimate P_i in a single round of communication whose duration is linear to the neighborhood size. We identified two parameters, namely the difference in response RSSI between neighbors and minimum duration of a response, that are critical for the performance of LinearPoll. We studied the constraints on choosing these parameters analytically and evaluated our implementation on the TelosB platform. Results show that LinearPoll achieves an accuracy of at least 98% for a neighborhood of 8 nodes in environments with sporadic interference and an accuracy of 95% even when malicious interference attacks 10% of the time.

LogPoll allows the poller to estimate $\log |P_i|$ at a constant cost. The accuracy of LogPoll the reliability of Proposition 1 which states signal strength from concurrent responses add up linearly. We analytically proved Proposition 1 and showed that it matches results on 3 different large node arrays in two testbeds. With our implementation of LogPoll on the TelosB motes, we showed that the accuracy of LogPoll is 99% (averaged over different numbers of responders) when all neighbors can adapt to a similar response RSSI and remains over 95% even when 10% of the neighbors fail to honor their power assignment.

In future work, it would be desirable to generalize our primitives for counting for P that are multivalued (instead of binary) predicates. A line of exploration would extend LogPoll as follows: since responders in LogPoll use fixed-length responses, it is feasible to encode more information using a variable response length. Let $v \in [V_{min}, V_{max}]$ be some variable of interest, which each neighbor j maintains locally as v_j. By dividing $[V_{min}, V_{max}]$ into bins, we can extend LogPoll by assigning to neighbor j a response length of $kC^{-1}(\Lambda)$ if v_j is in the k^{th} bin. The poller can then aggregate the distribution of v in the neighborhood by counting in log scale the number of neighbors in each bin.

8. REFERENCES

[1] IEEE Computer Society Standard for local and metropolitan area networks–part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs). IEEE, 2011.

[2] M. Ammar and G. Rouskas. On the performance of protocols for collecting responses over a multiple-access channel. In *Proceedings of the Tenth Annual Joint Conference of the IEEE Computer and Communications Societies. Networking in the 90s., IEEE*, pages 1490–1499, 1991.

[3] N. Burri, P. Von Rickenbach, and R. Wattenhofer. Dozer: Ultra-low power data gathering in sensor networks. In *2007 6th International Symposium on Information Processing in Sensor Networks*, pages 450–459, 2007.

[4] Y. Chen and A. Terzis. On the mechanisms and effects of calibrating RSSI measurements for 802.15. 4 radios. *Wireless Sensor Networks*, pages 256–271, 2010.

[5] M. Demirbas, O. Soysal, and M. Hussain. A singlehop collaborative feedback primitive for wireless sensor networks. In *IEEE INFOCOM The 27th Conference on Computer Communications*, pages 2047–2055, 2008.

[6] M. Demirbas, S. Tasci, H. Gunes, and A. Rudra. Singlehop collaborative feedback primitives for threshold querying in wireless sensor networks. In *Parallel and Distributed Processing Symposium*, pages 322–333, 2011.

[7] A. Dutta, D. Saha, D. Grunwald, and D. Sicker. Smack: a smart acknowledgment scheme for broadcast messages in wireless networks. In *ACM SIGCOMM Computer Communication Rev.*, volume 39, pages 15–26, 2009.

[8] 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 the 8th ACM Conference on Embedded Networked Sensor Systems*, pages 1–14, 2010.

[9] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *Networking, IEEE/ACM Trans. on*, 11(1):2–16, 2003.

[10] K. Jamieson, H. Balakrishnan, and Y. Tay. Sift: A MAC protocol for event-driven wireless sensor

networks. In *Wireless Sensor Networks*, pages 260–275. Springer, 2006.

[11] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TinyDB: an acquisitional query processing system for sensor networks. *ACM Trans. Database Syst.*, 30(1):122–173, 2005.

[12] V. Namboodiri and A. Keshavarzian. Alert: An adaptive low-latency event-driven mac protocol for wireless sensor networks. In *2008 International Conference on Information Processing in Sensor Networks*, pages 159–170, 2008.

[13] F. Österlind, N. Wirström, N. Tsiftes, N. Finne, T. Voigt, and A. Dunkels. Strawman: Making sudden traffic surges graceful in low-power wireless networks. In *Workshop on Hot Topics in Embedded Networked Sensors*, 2010.

[14] K. Srinivasan and P. Levis. RSSI is under appreciated. In *Proceedings of the Third Workshop on Embedded Networked Sensors*, 2006.

[15] T. Van Dam and K. Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, page 171, 2003.

[16] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh. Fidelity and yield in a volcano monitoring sensor network. In *OSDI '06: Proceedings of the 7th symposium on Operating systems design and implementation*, pages 381–396, Berkeley, CA, USA, 2006. USENIX Association.

[17] Y. Yao and J. Gehrke. The cougar approach to in-network query processing in sensor networks. In *SIGMOD Rec.*, volume 31, pages 9–18, 2002.