Lightweight Neighborhood Cardinality Estimation in Dynamic Wireless Networks

Marco Cattani* Marco Zuniga Andreas Loukas Koen Langendoen
Embedded Software Group
Delft University of Technology, The Netherlands
{*m.cattani, m.a.zunigazamalloa, a.loukas, k.g.langendoen*}*@tudelft.nl*

***Abstract*—We address the problem of estimating the neighborhood cardinality of nodes in dynamic wireless networks. Different from previous studies, we consider networks with high densities (a hundred neighbors per node) and where *all* nodes estimate cardinality concurrently. Performing concurrent estimations on dense mobile networks is hard; we need estimators that are not only accurate, but also fast, asynchronous (due to mobility) and lightweight (due to concurrency and high density). To cope with these requirements, we propose *Estreme*, a neighborhood cardinality estimator with extremely low overhead that leverages the rendezvous time of low-power medium access control (MAC) protocols. We implemented *Estreme* on the Contiki OS and show a significant improvement over the state-of-the-art. With Estreme, 100 nodes can concurrently estimate their neighborhood cardinality with an error of ≈10%. State-of-the-art solutions provide a similar accuracy, but on networks consisting of a few tens of nodes and where only a fraction of nodes estimate the cardinality concurrently.**

***Keywords*—Wireless Communications; Modeling; Performance Evaluation.**

I. Introduction

Knowing the neighborhood cardinality in wireless networks is an essential building block of many adaptive algorithms, such as resource allocation [1] and random-access control [2]. Cardinality estimation is also a valuable tool on itself. It can be used to monitor the surrounding environment, transforming the radio device into a smart sensor [3]. Our work is indeed part of a larger project related to public safety. The project's goal is to provide coin-size devices to attendees in open-air large-scale festivals and issue alerts when the crowd density crosses dangerous thresholds. In this type of applications, all or most of the devices need to periodically estimate their surrounding density, which can reach levels of hundreds of nodes.

Neighborhood cardinality estimation is a broad and active research area, but for the purposes of our work, we are interested in studies that perform empirical evaluations on dynamic networks, i.e. networks with frequent topology changes due to fluctuations in link quality and node movement. Within this scope, the state-of-the-art achieves an accuracy between 3% and 35% for 25 smartphones, relying on audio signals [4]. Other studies, using bluetooth signals in smartphones [3] and radio signals in sensor nodes [5], achieve comparable results for similar settings. Nevertheless, only a fraction of the nodes perform the estimation process.

*Marco Cattani was supported by the Dutch national program COMMIT/

We advance the state-of-the-art in two ways. First, we move from scales of a few tens of neighboring nodes to one hundred nodes. Second, we allow all nodes to perform the estimation concurrently. Solving this novel estimation problem is challenging. Network dynamics require estimations that are fast and asynchronous. These latter characteristics limit the number of samples (i.e., information) that can be collected, which in turn decreases accuracy. On the other hand, higher density and concurrency pose an extra burden on the sampling process and necessitate an efficient use of bandwidth.

To cope with these challenges we propose *Estreme*, a low-overhead cardinality estimator that is robust to mobility and supports multiple estimators running simultaneously in an asynchronous manner. The key idea behind Estreme is simple: in networks where all nodes perform periodic but random events within a given period, the time difference between two consecutive events (rendezvous time) captures the density of the neighborhood. The shorter the rendezvous time, the higher the density, and vice-versa.

Contributions. Our main contributions are:

(i) In Section II, we model the rendezvous time using order statistics and derive a neighborhood cardinality estimator. The model permits us to (a) provide four rules that are necessary and sufficient for using Estreme in a wide family of communication protocols, (b) gain insights into the performance of the estimator and (c) derive bounds for the estimation error.

(ii) In Section III, we implement Estreme on top of a low-power listening protocol. Even though Estreme is conceptually simple, implementing it on real nodes rises a number of challenges: rendezvousing with the right events, measuring the rendezvous time accurately and exploiting spatial/temporal correlations in the sampling process. We thoroughly analyze these challenges and provide viable solutions.

(iii) In Section IV, we extensively evaluate Estreme on a testbed consisting of 100 wireless nodes performing concurrent estimations. Our implementation achieves an estimation error of ≈ 10% with an overhead of just 4 bytes per estimation and a duty cycle of ≈ 3%. Moreover, Estreme provides a good trade-off between agility and precision. For example, when the neighborhood size changes abruptly from 30 to 60 nodes, the estimated cardinality of nodes converges within one minute to the 10% error range.

II. ESTREME

The ideas presented in this section form the basis of our approach for estimating the neighborhood cardinality. First, we derive an estimator based on order statistics and discuss its applicability to communication protocols (Section II-A). We then analyze the impact of timing errors on the estimation accuracy (Section II-B).

A. Mechanism

The neighborhood set V_u of a node u in a wireless network consists of all nodes v in the radio vicinity of u. Our objective is to compute $n_u = |V_u|$ i.e., its cardinality. In the following, we use n instead of n_u when u becomes implicit.

Cardinality estimator. We assume that, within a given period t_w, all nodes in the network perform an event in a random and desynchronized manner, as in Figure 1. Considering a random point in time, the time sequence of the subsequent events can be modeled as a set of independent random variables following a uniform distribution ($X_1 \dots X_n$). This random point in time represents the moment when a node, called *initiator*, wishes to estimate the cardinality of its neighborhood. The rendezvous time with the first k neighbors (events) captures the cardinality. Intuitively, the longer it takes to rendezvous, the lower the cardinality (because the distribution of random events during t_w is sparser).

The rendezvous time $T_r(k)$ with the first k neighbors is a random variable and can be modeled using order statistics. The density function of $T_r(k)$ is known to follow the beta distribution [6]

$$T_r(k) \sim \text{Beta}(\alpha, \beta), \tag{1}$$

and in our scenario $\alpha = k$ and $\beta = n + 1 - k$. Considering the period t_w, the expected value of the rendezvous time, i.e., the expected time it takes to observe k events is

$$\mathbb{E}[T_r(k)] = t_w \frac{\alpha}{\alpha + \beta} = t_w \frac{k}{n+1}. \tag{2}$$

Inverting the expectation, we obtain a simple-to-compute estimator for the neighborhood cardinality n based on the average of the observed rendezvous times $\bar{t}_r$.

$$\hat{n} = t_w \frac{k}{\bar{t}_r} - 1. \tag{3}$$

In our work we consider $k = 1$, that is, we estimate the cardinality by using the average rendezvous time with the first event only. As we will describe later, $k = 1$ is chosen because it minimizes the amount of bandwidth used to collect a sample. As mentioned before, an efficient use of bandwidth is a central requirement to cope with a high number of concurrent estimations.

It is important to note that in theory, for $k = 1$ the expectation of the estimator diverges [7], because as $n \to \infty$, $t_r \to 0$ and $\hat{n} \to \infty$. In practice, t_r remains positive. However, for very large neighborhoods, as $n \to \infty$, Estreme could use $k \geq 2$. For these values of k the expectation of the estimator becomes $n/(k-1)$ [8].

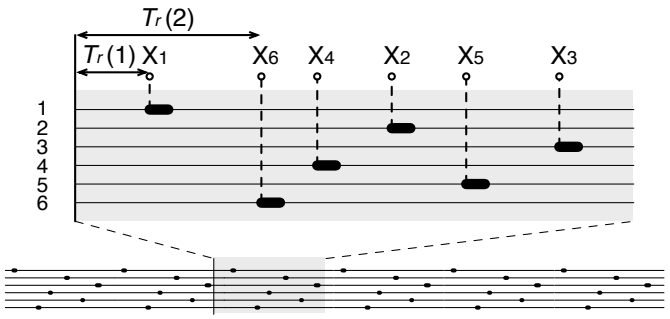


Fig. 1. Modeling the rendezvous times with the first k neighbors using the Beta random variable $T_r(k)$. The random variables $X_1 \dots X_n$ model the wake-up times of nodes.

Applicability of Estreme. Because of its simplicity, the proposed estimator is applicable to any network protocol that abides to three requirements: font=

- R1. Periodically (every t_w), nodes generate independent and random events.
- R2. The event of a node is observable by all neighbors.
- R3. Nodes can accurately measure inter-event periods.

Requirement 1 guarantees that, given a random moment in time, we can model the occurrence of the first event using a uniform random variable. Requirement 2 ensures that an *initiator* node appropriately identifies the first k events. That is, it does not mistake a later event for an earlier one. For example, due to collisions, earlier events can be lost and Estreme would underestimate the cardinality. Requirement 3 ensures that the rendezvous time is measured accurately. Otherwise an overestimation (shorter rendezvous than factual) or underestimation (longer rendezvous than factual) of the cardinality would occur.

B. Timing inaccuracies

As we will observe in the next section, measuring accurately the rendezvous time is challenging because several delays are introduced. Some of these delays can be measured and overcome, but others are difficult to track, such as computation and transmission delays. Analyzing the effects of these delays is central to understanding the performance of Estreme in real scenarios. In this section we derive a bound for the cardinality error caused by a positive delay ϵ in the rendezvous time.

Proposition 1. *Given a timing error ϵ in the rendezvous time t_r, the expected cardinality error is*

$$\mathbb{E}[e_n] = \frac{\hat{n}_\epsilon - n}{n} = \Theta\left(-\frac{\rho}{1+\rho}\right),$$

where $\rho = \epsilon(n+1)/t_w$, and $\hat{n}_\epsilon$ is the expected value of the estimated cardinality considering ϵ.

Proof: The proof consists of two steps. First, we derive $\hat{n}_\epsilon$ based on an expected rendezvous time delayed by ϵ. We then propose a bound on the resulting error e_n and show that it is tight. Let us start with the derivation of e_n. Substituting the expression of $\hat{n}_\epsilon$ derived from (2) and (3) into the definition of $\mathbb{E}[e_n]$, we obtain

$$\mathbb{E}[e_n] = -\frac{\epsilon(n+1)^2}{n\,(t_w + \epsilon(n+1))}. \quad (4)$$

Observe that for $\epsilon > 0$, the above is negative; any positive delay causes the underestimation of n. Unfortunately, even though exact, Equation (4) is not intuitive. It is easy to see that, the simpler and more insightful expression

$$\Phi = -\frac{\epsilon(n+1)^2}{(n+1)\,(t_w + \epsilon(n+1))} = -\frac{\epsilon(n+1)/t_w}{1 + \epsilon(n+1)/t_w} \quad (5)$$

tightly bounds $\mathbb{E}[e_n]$. It is sufficient to show that, for each n, positive k_1, k_2 exist such that

$$k_1\Phi \leq \mathbb{E}[e_n] \leq k_2\Phi, \text{ for all } n \geq 1.$$

Indeed, the above inequality holds for

$$k_1 \leq (n+1)/n \quad \text{and} \quad k_2 \geq (n+1)/n.$$

Setting $\rho = \epsilon(n+1)/t_w$ in Equation (5) concludes our proof. ■

Proposition 1 leads to two observations for practical implementations of Estreme. *First, given a fixed measurement error, longer periods t_w are preferable in terms of estimation error.* As $\rho \to 0$ the estimation error tends to 0^-. This can be caused both by ϵ being very small (this is obvious, since a small measurement error implies small estimation errors), but also by long periods t_w and small neighborhood sizes. While we cannot choose the number of neighbors, we should prefer longer periods t_w to shorter ones. *Second, to run Estreme, a platform should be able to measure the rendezvous time with sub-millisecond accuracy.* If we apply the Proposition to a neighborhood of 100 nodes with a period $t_w = 1000\,ms$, the estimation error introduced by a measurement delay $\epsilon = 1\,ms$ is

$$\Theta(-\frac{0.101}{1+0.101} = -0.09).$$

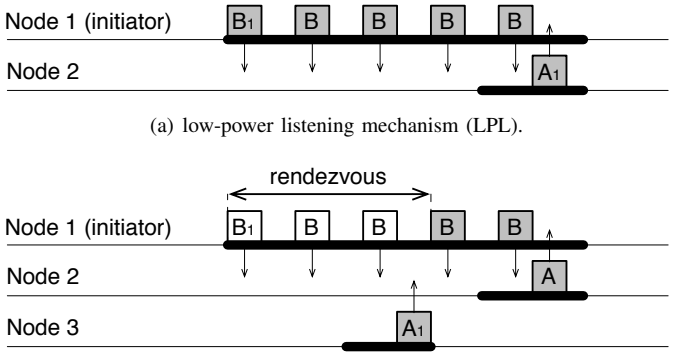
That is, with a measurement error of just 1 ms, the estimation error is 9%. Obtaining an accurate estimation of the rendezvous time at sub-millisecond accuracy is therefore central for the correct operation of Estreme.

III. Estreme in LPL

Estreme is a general framework that can be implemented over many communication protocols. In our work, we build Estreme on top of a generic low-power listening MAC protocol in Contiki OS (X-MAC [9]). We first describe a *Naive* implementation of Estreme on LPL and highlight its limitations (Section III-A). Then, we analyze these limitations and provide solutions for them (Sections III-B, III-C, and III-D).

A. Naive implementation

In LPL, nodes wake up periodically and with a fixed frequency t_w. When a node, called *initiator*, wants to communicate (see Figure 2(a)), it sends a strobe of beacons (B) until the intended neighbor (node 2) wakes up. The wake-up of nodes is the observable event required by Estreme. To adhere to the uniform and random distribution of requirement 1, we slightly modified the code. Instead of letting nodes wake up deterministically at every t_w, we introduce a uniform and random delay in the range $[-t_w/2, t_w/2]$. With this modification, the probability density function of the rendezvous times adheres to the Beta distribution required by Estreme to perform the estimation. Note, however, that the random delay introduced into the LPL's wake-up times can lead to jitter, which may cause compatibility issues with other protocol designs.



(a) low-power listening mechanism (LPL).

(b) Estreme mechanism applied to LPL unicast

Fig. 2. Estreme mechanism applied to LPL

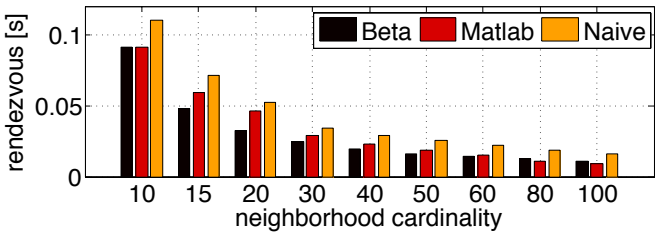
Anycast, unicast, broadcast. LPL has two basic communication primitives: unicast and broadcast. With broadcast, the strobe of beacons lasts for the entire duration of the period t_w. With unicast, the strobe of beacons ends as soon as the intended receiver is found.

With unicast and broadcast, the rendezvous time with the *first* neighbor to wake up can be obtained via a slight modification of the LPL beaconing mechanism. During the strobe of beacons, the initiator announces that the first wake-up has not yet been observed by setting a flag in its beacons (white packets in Figure 2(b)). When a neighbor receives a flagged beacon (node 3), it sends an acknowledgement (A1). After receiving "A1", the initiator clears the flag and continues its strobe normally until the intended destination is found (in the case of unicast), or until the wake-up period expires (in the case of broadcast).

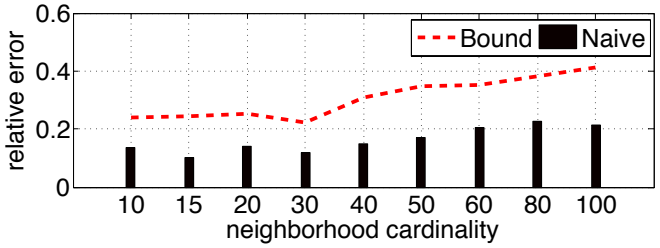
We implemented a third primitive: anycast, which communicates opportunistically with the first neighbor to wake up. Anycast has been gaining significant attention in the sensor network community due to its ability to reduce the delay and to increase the throughput of networks [10], [11].

From now on, our implementation focuses only on the anycast primitive for two reasons. First, anycast natively supports Estreme's requirements without posing any additional overhead (no need to strobe beacons after the first event is found). Second, anycast has been shown to be particularly suitable for dense, mobile networks [12]. Thanks to its very efficient use of bandwidth (much shorter rendezvous times compared to unicast and broadcast), Estreme is able to concurrently and periodically estimate neighborhood cardinalities of hundreds of nodes.

Implementation issues. At first glance, the rendezvous time is simple to measure: the initiator starts a timer before sending



(a) Rendezvous times



(b) Estimation error and bound

Fig. 3. Naive implementation of Estreme compared to the Beta model.

its first beacon (B1) and stops the timer upon receiving the acknowledgement (A1). We evaluated this *Naive* method on our testbed for various neighborhood sizes, from 10 to 100. The cardinality estimation is based on the mean rendezvous time ($\bar{t}_r$) of the last 50 samples observed by the initiator. The initiator sampled its neighborhood at a rate of 1 Hz.

Figure 3(a) compares the mean rendezvous times with (i) our analytical model ($\mathbb{E}[T_r]$ according to Equation 2), and (ii) Monte Carlo simulations performed in Matlab, which capture an ideal networking environment. Note that for all neighborhood cardinalities, Estreme over-measures the rendezvous time. Figure 3(b) shows that, as a consequence, the estimation error of the neighborhood cardinality (black bars) is significant, between 10% and 20%. Based on the *maximum* difference (ϵ) between the *Naive* measurements and the expected rendezvous time $\mathbb{E}[T_r]$ (Proposition 1), we observe that the estimation error could reach values between 20% and 40%.

The problem with the "*Naive*" implementation is that it does not adhere to Estreme's 2nd and 3rd requirements. Sometimes an initiator misses the first event, and mistakes subsequent events as being the first (non-compliance with requirement 2), and the calculation of the rendezvous time includes delays that are not part of the rendezvous itself (non-compliance with requirement 3). In the next sections we describe and solve these two problems, and explain a method to accelerate the estimation convergence of Estreme.

B. Correct observations

Since Estreme's estimations are based on the wakeup sequence of nodes, it is essential that such observations are correct. But collisions can affect this requirement, see Figure 4. If two or more neighbors wake up between two consecutive beacons, a collision will occur. Denoting t_b as the inter-beacon interval, the collision probability is

$$P(\text{collision}) = \sum_{j}\sum_{i=2}^{n} P\left(\begin{array}{c}\text{first } i \text{ nodes wake up at}\\ \text{the } j^{th} \text{ interval}\end{array}\right) = \sum_{j=1}^{\lfloor t_w/t_b \rfloor}\sum_{i=2}^{n}\binom{n}{i}\left(\frac{t_b}{t_w}\right)^{i}\left(1-\frac{t_b}{t_w}j\right)^{n-i}$$

where t_b/t_w is the probability that a node wakes up at each inter-beacon duration, and $1-(t_b/t_w)j$ is the probability that a node wakes up after the j^{th} interval. The likelihood of a collision is not high. For example, if n is 10, 50 and 100, the collision probability is 0.02, 0.11 and 0.22, respectively. Note that, even though this probability captures the collision of any number of nodes, most of the probability mass is concentrated on the case when only two nodes collide.

To reduce the chances of missing the first event(s), Estreme implements the following conflict resolution mechanism: if a node detects that its acknowledgment is lost (by receiving the beacon again), the node will retransmit its acknowledgement with probability p and go back to sleep with probability $(1-p)$. Denoting $n_c \geq 2$ as the number of colliding nodes, the conflict resolution mechanism leads to three outcomes:

(i) *starvation*, with probability $(1-p)^{n_c}$, all nodes go to sleep and the first event(s) is lost;

(ii) *completion*, with probability $n_c\, p\, (1-p)^{n_c-1}$, only one node remains awake and sends successfully the acknowledgement;

(iii) *contention*, more than one node remain awake and the contention process restarts with those nodes, with probability $1-(1-p+n_c\, p)(1-p)^{n_c-1}$.

Since we want to maximize the probability of *completions*,

$$\frac{\partial}{\partial p} n_c\, p\, (1-p)^{n_c-1} = 0 \implies p = \frac{1}{n_c}. \tag{6}$$

In a real scenario however Estreme will not know the number of contending nodes. In our implementation we settle for $p = 0.5$, because the higher p, the lower the probability of *starvation*, but according to Equation 6, $p \leq 0.5$. As an example, if $n = 100$ the probability that exactly two nodes collide and reach starvation is $P(\text{collision})P(\text{starvation}) = 0.19 \times 0.\overline{3} = 0.06$.

To avoid the (unlikely) possibility of infinite *contentions*, nodes will back off and go to sleep after a maximum number of unsuccessfully retransmissions.

Note that this conflict resolution mechanism introduces a delay in the measurements of the rendezvous time that badly affects Estreme's accuracy. In the next section, we provide a comprehensive solution to overcome not only this delay but many others.

C. Accurate measurements

Encountering the first node that wakes up is a necessary, but not sufficient, step to properly estimate the neighborhood cardinality. *The rendezvous time of nodes needs to be measured accurately*.

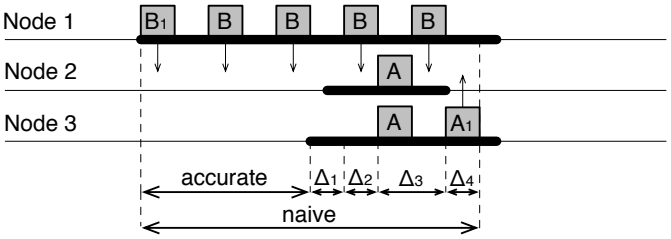


Fig. 4. Delays of the rendezvous measure.

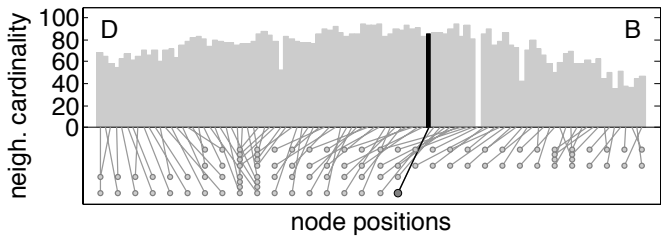


Fig. 5. Relation between the topology of the testbed (bottom) and neighborhood cardinalities (top). For specific experiments, only the results from a representative node (in black) are shown.

Overcoming delays. In principle, the measuring of the rendezvous time should stop at the moment when the first neighbor wakes up. But, as Figure 4 shows, the *Naive* mechanism includes different kinds of delays; namely, collisions ($\Delta 3$), the time required to transmit the radio packets ($\Delta 2$, $\Delta 4$), and the listening time between the actual wake-up and the moment when the beacon is sent ($\Delta 1$). This initial listening period $\Delta 1$ is a typical feature of LPL protocols such as X-MAC [9] and can be as long as the inter-beacon interval t_b.

To obtain an accurate measurement, we need to subtract these delays from the naive measurements. In our implementation, this is done through the receiver (node 3), by starting a timer once the receiver wakes up and piggybacking the elapsed time on the acknowledgments ($\Delta 1$ + $\Delta 2$ + $\Delta 3$). Assuming a fixed bit rate and a fixed acknowledgement size, both reasonable assumptions, $\Delta 4$ can be computed off-line as a constant and be systematically subtracted by the initiator after the acknowledgement is received. In our case $\Delta 4$ = 1.1 ms.

Accurate timing in Contiki OS. In Contiki, the clock library allows measurements with a maximum precision of 2.83 ms. This coarse-grained measure is not suitable for our estimation purposes. To ensure the maximum possible accuracy, we measured the rendezvous times using Contiki's real-time module (*rtimer*). Its precision depends on the processor's clock frequency (32 KHz in our devices) and allows to measure time-ranges with sub-millisecond accuracy.

D. Improving the estimation process

As any other estimator based on order statistics, Estreme is bound to the law of large numbers: the larger the number of samples, the closer the mean gets to the expected value and the more accurate the estimation. Unfortunately, in mobile networks nodes cannot collect many samples because they only have a limited amount of time to capture the current status of their neighborhoods. The central question is hence, *how can we facilitate a fast gathering of samples?* Estreme's design tackles this problem in the temporal (T-Estreme) and spatial (S-Estreme) domains.

Averaging samples in time. Intuitively, given a certain period T, we would like nodes to gather as many samples as possible during that period. The key characteristic to achieve this goal is an efficient use of bandwidth, that is, to spend as little time as possible gathering each sample. By using anycast, *Estreme reduces the use of bandwidth to the minimum time required to observe an event*. To process these samples, T-Estreme utilizes a simple moving window average (MWA) filter, where the last w samples of the rendezvous time are averaged to estimate the cardinality. We tried other robust statistical methods based on the median and on alpha-trimmer filters, but there was not much difference with regards to MWA (because requirements 2 and 3 already remove most outliers). Like most MWA estimations, the tradeoff in the time domain is between accuracy and adaptability: the bigger w, the more time it takes to adapt to changes in cardinality.

Averaging samples in space and time. Different from most cardinality estimators in the literature, Estreme is designed from inception to allow *all* nodes to perform concurrent estimations. This characteristic is not only good because some practical applications require it, but perhaps more importantly because it can quadratically increase the convergence of the estimation. In S-Estreme, besides sending the delay information described in Section III-C, a receiver also sends the average of its own time window. *Every node is hence able to process w^2 samples in the same time required to locally collect w samples*. Note that in S-Estreme, it is preferable to choose w based on the expected cardinality of the neighborhood n. With $w > n$, some nodes' samples will be counted multiple times. With $w < n$, on the other hand, only part of the neighbors' samples will be taken into account by the estimator. While in our experiments w is kept constant, it is also possible to dynamically adapt w's size based on the cardinality estimation provided by T-Estreme. It is important to highlight that while in most cases S-Estreme will significantly improve the estimation process, in case of drastic changes in network density, errors can be introduced due to the spatial smoothing that is inherent to S-Estreme. This phenomenon is analyzed in Section IV.

IV. Evaluation

To evaluate Estreme, we ran an exhaustive set of experiments on a testbed consisting of 100 sensor nodes equipped with an MSP430 processor and a CC1101 radio. The testbed is deployed in the ceiling of our offices and covers an entire floor of the building. As the testbed was operated throughout the week, Estreme was subject to all kinds of environmental conditions (e.g., human activity during office hours and temperature swings during the weekend) inducing quite a variety of link dynamics, see [13].

To test network dynamics in a more controlled manner, we follow a twofold approach: (i) we systematically turn on/off various nodes in the network, and (ii) we provide a few people with nodes and have them walk around our floor and building to test transitions between areas with different node densities.

TABLE I. DEFAULT PARAMETER SETTINGS.

Symbol	Parameter	Value
t_w	wake-up period	1 second
t_s	sampling period	1 second
w	window size	50 samples
n	network size	100 nodes

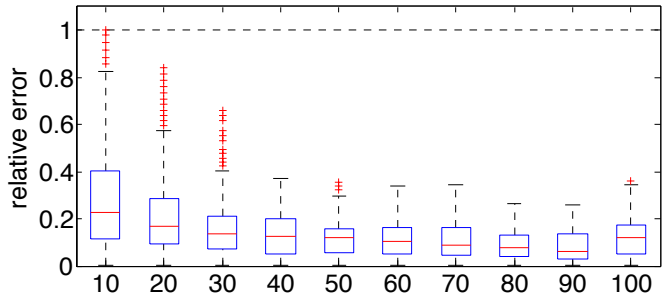
Ground truth: In testbed experiments, it is difficult to define the ground truth of the neighborhood cardinality. This occurs because the quality of links is highly variable, and cardinality changes significantly over time. This high variability is not particular to our testbed. In [14], the authors report that in the Twist testbed, the cardinality of some nodes oscillate between 12 and 17 in periods of 30 minutes.

For each experiment, we compute the ground truth cardinality of a node as the number of neighbors that have been *observed* at least once during the duration of an experimental run. A node is *observed* if it successfully exchanges at least one acknowledgement with the corresponding initiator. Note that the ground truth cardinality is an upper bound that is seldom reached. The fact that a node is observed during a period t_w does not imply that it will be observed in the next period(s). Hence, all the estimation errors reported in this section are *worst case errors*.

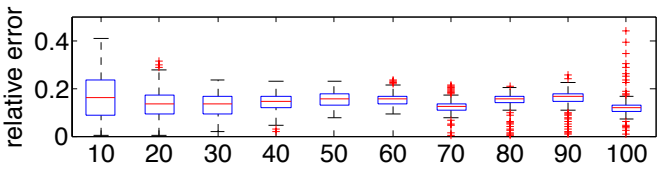
Figure 5 shows the network topology (bottom part) together with the neighborhood cardinality (upper part) of each node in a *typical* experiment. Due to the shape of the building (narrow and long), when the maximum power is used (+10dBm), the central nodes communicate with most of the network, while the nodes on the two far-ends reach approximately half of the other nodes. This setup creates a network with a minimum diameter of 2 and offers a wide range of cardinalities, approximately from 35 to 85 neighbors. These network cardinalities allow us to understand the behavior of Estreme in case of *un-even densities*. Also, to provide a more accurate comparison of different methods, in some cases we use a representative node with low cardinality variability. This representative node is marked black in Figure 5.

Metrics and parameters. To evaluate the accuracy of Estreme, we computed the *relative error* of the estimations as $|(\hat{n}-n)/n|$, where $\hat{n}$ is the estimated neighborhood size and n is the "ground truth" neighborhood cardinality. With regards to the parameters, unless stated otherwise, our experiments use the default settings listed in Table I.

Window size exploration. We evaluated the performance of T-Estreme with different window sizes (from 10 to 100) and different network sizes (10, 50,and 100 nodes). Figure 6(a) shows the results for a central node (the black node in Figure 5 in the full testbed (100 nodes)). T-Estreme reaches a plateau at $w = 50$. For $n = 10$ and $n = 50$ (results not shown) we obtained similar results. Since bigger windows will reduce the estimator's agility to adapt to changes in cardinality, we chose $w = 50$ as the default value. As we will show later, the reason why $w = 50$ holds for all n is a natural consequence of the fact that the rendezvous samples follow a gaussian distribution



(a) Exploring different window sizes (w) for T-Estreme



(b) Exploring different window sizes (w) for S-Estreme.

Fig. 6. Exploring the error and variance of Estreme for different sizes of the sampling windows.

for all n. When both time and spatial information is taken into account, the cardinality can be estimated as follows:

$$\hat{n} = \hat{n}_T\,\alpha + \hat{n}_S\,(1-\alpha),$$

where $\hat{n}_T$ is the cardinality estimated by T-Estreme, i.e. considering only the node's own samples, and $\hat{n}_S$ is the cardinality estimated by S-Estreme, i.e. considering only the node's neighbors averages. In our work, we only report the extremes. When T-Estreme is used $\alpha = 1$, when S-Estreme is used $\alpha = 0$. Clearly, both estimations, temporal and spatial, can be combined by fine-tuning α according to the needs of the application. Figure 6(b) shows the results for S-Estreme. Due to the gains in spatial correlation, S-Estreme with $w = 10$ provides similar results to T-Estreme with $w = 50$.

Note that for network sizes beyond 70, the performance of S-Estreme decreases. This phenomena is due to the smoothing behavior caused by spatial averaging and it is evaluated in more detail later in this section.

Comparing with the state of the art: To evaluate the performance of Estreme, we implemented a *baseline* estimator combining the ideas of three studies [5], [15], [16]. The specific difference of Estreme with each one of these studies is described in the related work section (Section V). The baseline estimator works as follows. The initiator broadcasts a request and the subsequent time is divided into 10 time slots. At each slot m, nodes jam the channel with probability $\frac{1}{2^m}$. The idea of jamming the channel is taken from [5]. Sending raw radio signal strengths (jamming), instead of sending packets, permits estimations in the order of a few ms (like Estreme). The idea of using time slots is borrowed from [16]. This time-slotted method permits coarse-grained but fast estimations of large cardinalities (which is also the focus of Estreme). Note that the baseline method requires all nodes to be awake at the moment of the estimation, while our method is asynchronous and requires only two active nodes at a time.

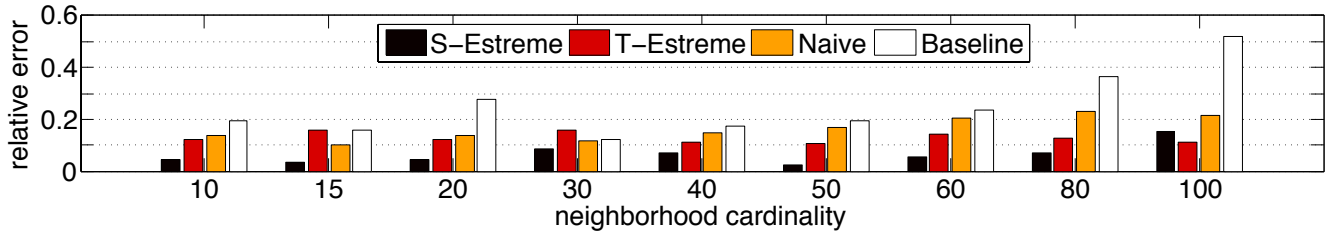


Fig. 7. Estreme's relative error compared to the baseline.

A. Results

This section tackles two key aspects of Estreme: (i) the ability to provide accurate cardinality estimations for large neighborhood sizes while performing concurrent estimations, and (ii) the agility to adjust to network dynamics.

Insight 1: *In the temporal domain, T-Estreme provides an accuracy that is similar to those of current solutions for low cardinalities, but at high cardinalities the performance of T-Estreme is many times better*. Figure 7 depicts the accuracy for various neighborhood sizes and various methods. For each tuple <method, neighborhood size>, the experiment run for an hour. For T-Estreme, S-Estreme and Naive, the estimation process runs concurrently on all nodes. For Baseline, the estimation runs only on the *representative* node (the black node in Figure 5), because running Baseline on all nodes requires synchronizing their requests, otherwise collisions occur. At low cardinalities (under 40), there is no clear gain for T-Estreme. Even further, while Baseline takes ≈10 ms for each estimation, T-Estreme takes on average $1000/n$ ms, e.g. 100 ms for 10 nodes. Under these conditions, Baseline would have sufficient bandwidth to perform concurrent communications as well (in spite of requiring an extra synchronization step). At higher cardinalities however, the accuracy of T-Estreme remains around 10%, while Baseline's performance deteriorates significantly. At $n = 100$, Baseline and Estreme spend a similar amount of time on each estimation. The reason for the poor performance of Baseline is its coarse-grained logarithmic approach. The estimation could be made more accurate by following a linear approach, as in [5], but this would require a high level of synchronization and longer times, which may not be permissible in dynamic settings.

An important characteristic of T-Estreme is that its accuracy remains rather stable for various cardinalities, between 10% and 15%, and the error bound validates this trend, see Figure 8(a) . None of the other estimators shown in Figure 7 share this characteristic. Intuitively the error should increase as n increases because 1 ms of error at $n = 10$ should matter less than the same error at $n = 100$. This rather stable behavior occurs because when n increases, not only the expected rendezvous time decreases linearly but so does the range of errors, see Figure 8(b). These trends cancel each other out. At this point it is also important to notice that the distribution of the rendezvous samples follows a Gaussian-like distribution. As explained before, it is due to this reason that $w = 50$ is a good window size for all n, because 50 random instances allow (in probability) sampling of most of the mass (range) of a Gaussian.

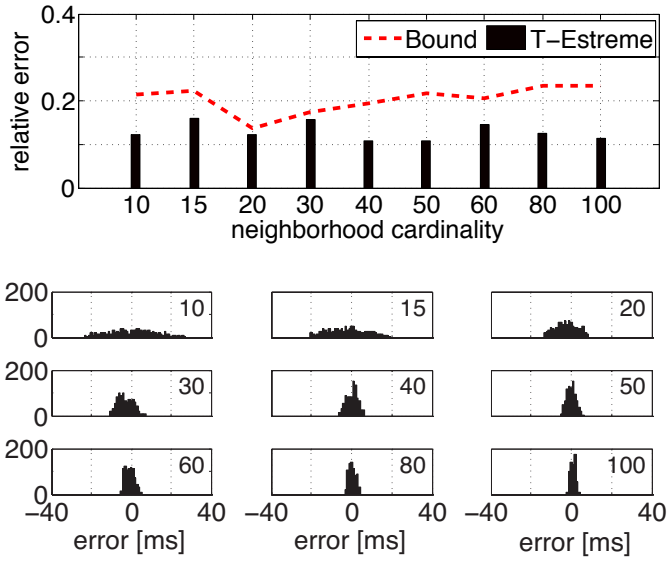
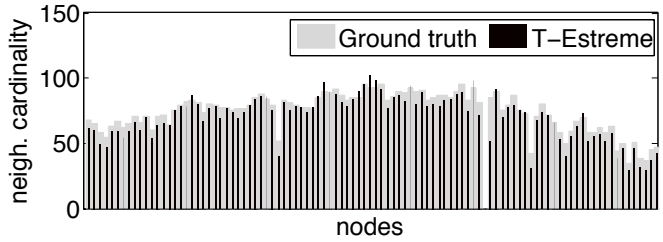


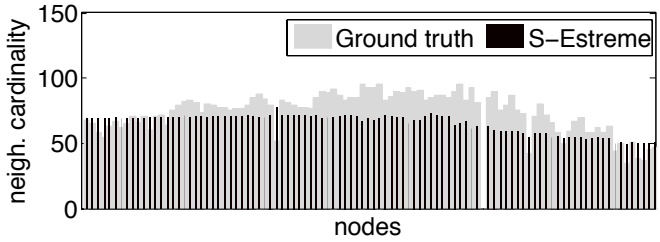
Fig. 8. Estimation accuracy (averaged) of T-Estreme across different cardinalities (top) with underlying distributions at the node level (below).

Insight 2: *In the spatial domain, S-Estreme outperforms current solutions at low and high cardinalities, but in scenarios with un-even deployments the estimation accuracy is compromised due to spatial averaging effects*. The ability of S-Estreme to collect data in a quadratic manner allows for a remarkable accuracy in a short period of time. Figure 7 shows that for most cardinalities S-Estreme has an error under 5%. The main exception occurs at $n = 100$, and it is due to the spatial averaging of un-even densities. Figures 9(a) and (b) capture the spatial averaging effect. In T-Estreme, each node has a precise view of its own neighborhood, as shown by the accurate mapping between the individual node estimations and the ground truth (Figure 9(a)). In S-Estreme however, by averaging the views of its neighbors, a node with a low cardinality –with regards to its neighbors– will overestimate its density, and vice versa (Figure 9(b)). Recall that we are showing the extremes of the parameter α (0 and 1), an intermediate α can be used to balance the benefits of T-Estreme and S-Estreme. It is also important to highlight that, while in principle spatial correlations can be applied to any estimator that exchanges packets with its neighbors, in mobile scenarios this spatial correlation requires a very efficient use of bandwidth (to be fast and to allow concurrency), which is one of the key characteristics of Estreme.

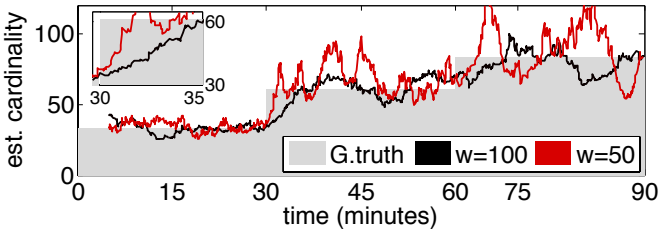
Insight 3: *Under network dynamics, Estreme adapts to sudden cardinality changes in a few minutes.* We run a series of



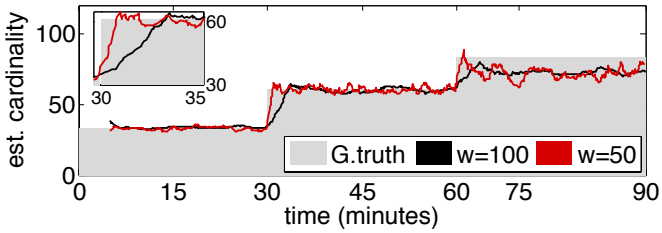
(a) T-Estreme in a static scenario with un-even cardinalities.



(b) S-Estreme in a static scenario with un-even cardinalities.



(c) T-Estreme with cardinality that grows over time (30 nodes every 30 minutes).



(d) S-Estreme with cardinality that grows over time (30 nodes / 30 minutes).

Fig. 9. Estreme estimations in networks with spatial (top) and temporal (bottom) variations in neighborhood cardinality.

90-minute experiments in which the neighborhood cardinality grows. Starting from 30 nodes, every 30 minutes, 30 nodes are added. Figure 9(c) shows the estimations of the *representative* node for T-Estreme. With twice the window size $w : 50 \rightarrow 100$, the estimator has a lower variance (more accurate), but it takes three times longer (five minutes) to adapt to the change in the neighborhood cardinality (zoom-in on the top left corner of Figure 9(c)).

S-Estreme, on the other hand, achieves a more accurate and faster convergence with $w = 50$, see Figure 9(d). In less than one minute it is able to adapt to the new cardinality. On the last jump however ($n : 60 \rightarrow 90$), S-Estreme suffers from the spatial averaging effect: the *representative* node starts receiving averages from nodes at the far-ends of the testbed (lower density), resulting in an underestimation of the neighborhood size. Note that a 1-minute convergence implies that a person at walking speed (1 m/s) covers approximately 60 meters while sampling the current neighborhood. Assuming a device with a transmission range of 50 m, Estreme should be able to cope with the dynamics of practical environments.

As a final experiment, we equipped 3 colleagues with a sensor node and asked them to move according to a predefined path. The experiment lasted 50 minutes. In the first 15 minutes, we asked our colleagues to have a chat at the ground floor of the building (location A, no testbed coverage). After taking the elevator, they reached the 9th floor where the testbed is located and moved towards one far-end of the testbed where the cardinality is lower (location B, cf. Figure 5). After standing there for 15 minutes, they were asked to move slowly to the other end of the floor (location D). The slow movement (in section C) was required to get an accurate measurement of the ground truth: at each step we required approximately 10 s to get a snapshot of the cardinality of the testbed node that was closest to the mobile node.

Figure 10 shows the estimated neighborhood cardinality of one of the mobile nodes. This figure highlights the tradeoff

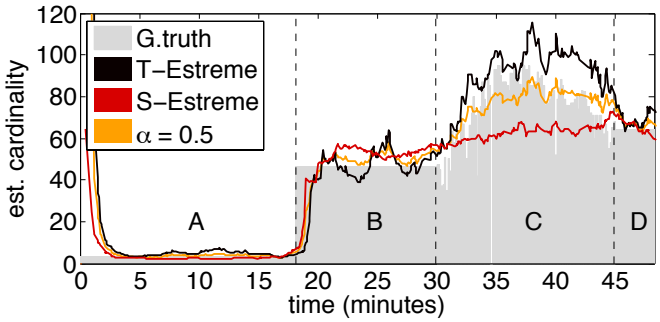


Fig. 10. Estreme's estimations in a 50-minute experiment with a group of 3 mobile nodes.

of T-Estreme and S-Estreme. If a quick estimation is required, S-Estreme is the best solution. On the other hand, if a more accurate, but longer, measurement is needed T-Estreme should be used. By considering both estimators is possible to combine the agility of S-Estreme with the accuracy of T-Estreme (see the trade-off estimator in Figure 10 with $\alpha = 0.5$).

Energy efficiency and bandwidth utilization. Estreme is not only a simple method, its overhead is very limited. T-Estreme only needs to piggyback the extra-delays incurred by the receiver (2 bytes). S-Estreme adds another 2 bytes to convey the current cardinality average of the receiver. In our implementation, the final data overhead is thus 4 bytes per estimation. This low overhead together with the short rendezvousing of anycast, not only leads to a very efficient utilization of bandwidth, but also to a very low duty cycle (energy consumption). Figure 11 shows the average duty cycle of nodes running Estreme on our testbed. The overhead of Estreme is quite limited, the average duty cycle is between 5% and 2%, and decreases with growing neighborhoods (as the time spent on rendezvousing decreases).

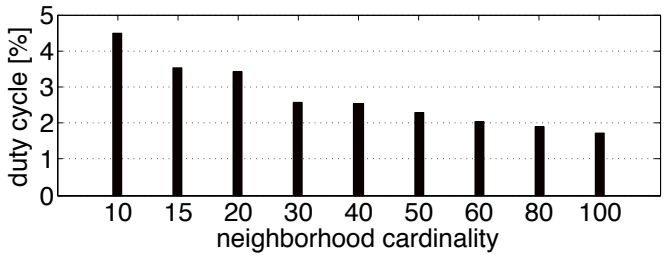


Fig. 11. Average duty cycle of nodes with different cardinalities.

Exploring the parameter space. Throughout our experiments, we used some default values for the parameters of both the underlying MAC and Estreme itself, namely, the wake-up period (t_w) and the sampling period (t_s). These parameters influence the performance of Estreme and, hence, it is important to assess them more thoroughly. We tested Estreme with two different wake-up periods, 0.5 and 1 second, and with five different sampling periods, 0.5, 1, 2, 5 and 10 seconds. Each $<t_s, t_w>$ tuple was run for an hour using 100 nodes. Tables II and III show the result of our experiments. The metrics of interest are the relative error and the relative sampling rate. The relative sampling rate captures the percentage of sampling requests that are successfully completed.

Relative error. As predicted by our model, increasing t_w decreases the error. Changing the sampling period, instead, does not seem to have a clear consequence on the estimation error, except for very low t_s and t_w, which may indicate some channel saturation.

Relative sampling rate. Ideally, we would like every request from an initiator to be completed, but as t_s decreases this is not possible due to channel saturation issues. For the default setup in our experiments $<t_w = 1s, t_s = 1s>$ only approximately 30% of the requests are successful. This relative low rate occurs because, in Estreme, an initiator that observes and ongoing estimation goes back to sleep. As the channel saturates, more and more requests are silently canceled. It is important to highlight however that the fraction of requests that are successful provides the same rendezvous accuracy as of those requests that are performed on lightly-loaded channels. In general, the *relative sampling rate* can be raised by reducing t_w, but results in larger estimation errors, so is to be avoided.

TABLE II. RELATIVE ERROR FOR DIFFERENT t_w AND t_s.

Relative error ($\lvert\hat{n} - n\rvert / n$)					
t_w	Sampling period t_s (s)				
(s)	10	5	2	1	0.5
0.5	0.16	0.16	0.29	0.31	0.24
1	0.13	0.11	0.14	0.11	0.07

TABLE III. RELATIVE SAMPLING RATE FOR DIFFERENT t_w AND t_s.

Relative sampling rate (%)					
t_w	Sampling period t_s (s)				
(s)	10	5	2	1	0.5
0.5	93.4	76.9	58.7	40.2	23.9
1	81.6	73.3	49.3	31.8	17.7

V. RELATED WORK

We start by providing a comparison of Estreme within the (mobile) sensor network domain. We then broaden our scope and overview solutions proposed in RFID systems and mobile phone networks. For a concise overview of the related work see Table IV.

Mobile sensor networks. Although neighborhood cardinality estimation is an essential building block of various protocols, it has been little explored within the context of (mobile) sensor networks. We follow by comparing to the three overarching approaches:

(i) *Polling and group testing mechanisms are the closest to Estreme*. Their goal it to efficiently estimate the cardinality of the n neighbors that hold a given property. Different from Estreme, these mechanisms do not specifically target mobile networks and, in some cases, do not estimate the *exact* number of neighbors. Backcast [20] tests if $n > 0$, while [16] tries to asses if n crosses a certain threshold. If $n > 0$, Strawman [21] is able to efficiently identify one of the responders. Among the various polling mechanisms, the one that is the most related to our work is the study by Zeng *et. al.* [5], where the authors propose two different methods. LogPoll is able to estimate the logarithmic cardinality of a neighborhood, i.e., $\log n$, in just 4 ms, while LinearPoll is able to recognize all the identities of the responders at the cost of spending more energy and time. This remarkable performance comes at a high cost and relies on several assumptions. LogPoll and LinearPoll model and periodically calibrate the signal strength of neighbors, effectively limiting the applicability of these estimators to static networks with minor link dynamics. Additionally, LinearPoll can only count a limited number of neighbors, because it requires neighbors to send radio signal strengths that are Δ apart. LogPoll, on the other hand, can only estimate $\log n$, which gets increasingly coarse-grained as n grows.

(ii) *Cardinality estimation can also be provided through neighbor discovery mechanisms like Disco [20], WiFlock [22] and U-Connect [23].* Unfortunately, even with the help of acceleration techniques such as ACC [24], these mechanisms take in the order of 10 minutes or more, which is too slow to cope with the mobile networks addressed by Estreme.

(iii) *Last, the NetDetect [18] algorithm, while similar in spirit to Estreme, makes stronger assumptions and achieves lower estimation accuracy.* Similar to Estreme, NetDetect relies on the underlying distribution of packet transmissions to estimate neighborhood cardinality. Nevertheless, with the same default window as Estreme (50 samples), it estimates the cardinality of a 100-node network with a relative error of 0.25 (in simulations). Furthermore, unlike Estreme which is built on top of duty-cycling MAC protocols, NetDetect uses the Aloha protocol and assumes that the radio is always on.

RFID. An area in which cardinality estimation plays a central role is Radio-Frequency Identification (RFID). Different from sensor networks, RFID systems are designed to track and monitor thousands of goods in storage facilities. These systems

TABLE IV. COMPARISON OF THE VARIOUS METHODS TO ESTIMATE NEIGHBORHOOD CARDINALITY. IN THE "TECHNIQUE" COLUMN, "MEAS." STANDS FOR MEASUREMENTS AND "OBS." FOR OBSERVATIONS.

Work	Type	Evaluation	Mobile	Scale	Error	Technique	Estimator	Device	Concurrent
[4]	cardinality estimator	real world	yes	25	3-35%	Audio tones	LoF [17]	Cellphone	few
[3]	density estimator	real world	yes	50	20-70%	Bluetooth	Classifier	Cellphone	few
[5]	group testing	testbed	no	32	1%	RSSI meas.	Classifier	TelosB	no
[18]	cardinality estimator	simulation	yes	100	25%	Radio obs.	MLE est.	Sensor node	yes
[17]	cardinality estimator	simulation	yes	65K	10%	Data pkts	LoF	RFID	no
[19]	cardinality estimator	simulation	yes	10K	1%	Data pkts	Order stat.	RFID	no
Estreme	cardinality estimator	testbed	yes	100	10%	Radio obs.	Order stat.	TelosB	yes

assume a centralized initiator, called *reader*, and a set of cheap, passive devices called *tags*. To estimate the neighborhood cardinality (the number of tags), the reader starts a *response collection* process, which is usually based on a TDMA scheme. This process consists of multiple trials, each one having multiple slots. Within each trial, every tag selects a slot in which it sends a response to the reader. This slot is chosen based on the tag's state, a random number, and a command sent by the reader at the beginning of each trial.

Based on the number of observed responses, collisions, and empty slots the initiator is able to estimate the population size [17]. This process is repeated iteratively multiple times (trials) to improve the accuracy of the estimators. While usually the reader cannot terminate a trial until the last assigned slot is observed, in [19] the authors propose to use order statistics to drastically reduce the trials' duration (an initiator needs to observe only the first k responses). Compared to Estreme, this work is able to achieve an accuracy of 1% and scale to thousands of nodes. On the other hand, it has only been tested in simulations and requires two orders of magnitude more samples than Estreme.

Adapting such a mechanism (and others from the RFID community) to wireless sensor networks is possible but very challenging. For a correct observation, neighbors will need to be synchronized, so that responses in adjacent slots do not collide. Moreover, a global scheduling mechanism should be in place to guarantee that at any moment in time only one node in the neighborhood can act as a reader (initiator).

Recently, Chenet al. [25] discovered that having a 2-phase estimation (a rough one followed by a more refined one) is the key to improve the estimation accuracy, independently of the chosen technique. In light of this, we plan to extend Estreme with a second estimation phase that will use the IDs of the encountered neighbors to improve the estimation accuracy.

Mobile phones. Initial studies have used mobile phones to estimate the density of crowds. The most relevant work uses audio tones to count neighbor devices [4]. The main challenge involved is to successfully transmit data packets using low-quality speakers/mic-rophones, as well as to cope with the presence of environmental auditive noise. Thanks to the richer computational capabilities of mobile phones, techniques such as Fast Fourier Transform can be used to code the signal. Energy efficiency is also addressed, but on a different scale than Estreme. Here the comparison is against the typical consumption of a WiFi network interface. Finally, similar to Estreme, this system is able to support multiple, concurrent estimations. While in theory the method scales to hundreds of devices by using multiple frequencies, the system was only tested with two concurrent estimators.

An interesting alternative to cardinality estimation is proposed in [3]. The phones scan for discoverable bluetooth devices and, based on the number of unique identifiers discovered, an estimation of the crowd density is performed. Different from Estreme and the other related works, this work focuses on classifying the density into four classes instead of providing a cardinality estimation. Even though the estimation task is simpler, unrealistic assumptions on the distribution of bluetooth-enabled devices results in estimation errors that varies from 20 to 70%.

VI. CONCLUSIONS

In this paper we addressed the issue of determining the neighborhood cardinality in dynamic wireless networks, where link quality fluctuations and node mobility ask for a robust and agile approach. An additional and important focus of our work is to handle high densities and concurrent estimations, requirements that are necessary in public safety applications such as crowd monitoring. Traditional approaches cannot meet these stringent requirements, so we developed Estreme: a cardinality estimator based on monitoring the inter-arrival times of (randomized) events raised by neighboring nodes.

To minimize channel usage, which is important in dense networks, Estreme leverages on the short rendezvous time offered by opportunistic anycast. To obtain high accuracy, Estreme employs a bag of tricks to account for various overheads, (transmission) latencies, collisions, and other factors that all distort the true rendezvous time with the first node to wake up. We derived a theoretical model underlying the rendezvous times, and used it to gain insight into the accuracy of Estreme as well as to derive bounds on the estimation error.

As a proof of concept, we implemented Estreme on the Contiki OS, and evaluated it on a testbed with node densities up to 100 nodes. Estreme achieves solid performance results with typical estimation errors below 10%, which compares favorably to state-of-the-art solutions. Estreme was also demonstrated to handle abrupt changes in density exemplified through an experiment involving few nodes moving through our testbed.

As part of our future work, we are currently planning a more ambitious deployment consisting of hundreds of nodes,

with a large fraction of them being mobile. We are also working on extending our analytical framework to provide bounds on delay and to characterize the operational regions of Estreme based on the channel saturation.

ACKNOWLEDGMENT

Marco Cattani was supported by the Dutch national program COMMIT/ (project P09 EWiDS). Andreas Loukas was supported by the Dutch Technology Foundation STW and the Technology Program of the Ministry of Economic Affairs, Agriculture and Innovation (D2S2 project). We would like to thank Venkat Iyer, Maarten van Steen and Spyros Voulgaris for the helpful discussions, the anonymous reviewers and our shepherd Tarek Abdelzaher for providing us detailed feedback on draft versions of this paper.

REFERENCES

[1] N. Burri, "Dozer: ultra-low power data gathering in sensor networks," in *6th Int. Simp. on Information Processing in Sensor Networks. (IPSN)*, 2007, pp. 450–459.

[2] A. Loukas, M. Woehrle, M. Zuniga, and K. Langendoen, "Fairness for All ; Rate Allocation for Mobile Wireless Networks," in *10th IEEE Int. Conf. on Mobile Ad-hoc and Sensor Systems (MASS)*, 2013.

[3] J. Weppner and P. Lukowicz, "Collaborative Crowd Density Estimation with Mobile Phones," in *9th ACM Conf. Embedded Networked Sensor Systems*, 2011.

[4] P. Kannan and S. Venkatagiri, "Low cost crowd counting using audio tones," in *10th ACM Conf. on Embedded Network Sensor Systems (Sensys)*, 2012, pp. 155–168.

[5] W. Zeng, A. Arora, and K. Srinivasan, "Low power counting via collaborative wireless communications," in *12th Int. Conf. on Information processing in sensor networks (IPSN)*. New York, New York, USA: ACM Press, 2013, pp. 43–54.

[6] N. Balakrishnan, E. Castillo, and J.-M. Sarabia, *Advances in Distribution Theory, Order Statistics, and Inference*, 2007.

[7] F. Giroire, "Order statistics and estimating cardinalities of massive data sets," *Discrete Applied Mathematics*, vol. 157, no. 2, pp. 406–427, Jan. 2009.

[8] P. Chassaing and L. Gerin, "Efficient estimation of the cardinality of large data sets," *arXiv preprint math/0701347*, no. 1, pp. 1–15, 2007.

[9] M. Buettner, G. Yee, and E. Anderson, "X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks," in *4th ACM conf. on Embedded networked sensor systems (Sensys)*, 2006, pp. 307–320.

[10] J. Kim, S. Member, X. Lin, N. B. Shroff, and P. Sinha, "Minimizing Delay and Maximizing Lifetime for Wireless Sensor Networks With Anycast," *IEEE/ACM Transactions on Networking (TON)*, vol. 18, no. 2, pp. 515–528, 2010.

[11] O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson, "Low Power , Low Delay : Opportunistic Routing meets Duty Cycling," in *11th Int. Conf. on Information Processing in Sensor Networks (IPSN)*, 2012, pp. 185–196.

[12] M. Cattani, M. Zuniga, W. Matthias, and K. Langendoen, "SOFA: Communication in Extreme Wireless Sensor Networks," in *11th European Conf. on Wireless Sensor Networks (EWSN)*, 2014.

[13] M. Woehrle, M. C. Bor, and K. G. Langendoen, "868 MHz: a noiseless environment, but no free lunch for protocol design," in *9th Int. Conf. on Networked Sensing Systems*, ser. INSS, no. Section VIII, 2012, pp. 1–8.

[14] M. Zuniga, C. Avin, and M. Hauswirth, "Querying Dynamic Wireless Sensor Networks with Non-Revisiting Random Walks," in *Wireless Sensor Networks*, 2010, pp. 49–64.

[15] M. Ammar and G. Rouskas, "On the performance of protocols for collecting responses over a multiple-access channel," in *10th Conf. of the IEEE Comp. and Comm. Societies (INFOCOM)*, 1991, pp. 1490–1499.

[16] M. Demirbas, S. Tasci, H. Gunes, and A. Rudra, "Singlehop Collaborative Feedback Primitives for Threshold Querying in Wireless Sensor Networks," in *Int. Parallel & Distributed Processing Symposium*. Ieee, May 2011, pp. 322–333.

[17] C. Qian, H. Ngan, Y. Liu, and L. Ni, "Cardinality Estimation for Large-Scale RFID Systems," *IEEE transactions on parallel and distributed systems*, vol. 22, no. 9, pp. 1441–1454, 2011.

[18] V. Iyer, A. Pruteanu, and S. Dulman, "NetDetect: Neighborhood Discovery in Wireless Networks Using Adaptive Beacons," in *5th Int. Conf. on Self-Adaptive and Self-Organizing Systems*, Oct. 2011, pp. 31–40.

[19] H. Han, B. Sheng, and C. Tan, "Counting RFID tags efficiently and anonymously," in *INFOCOM*, 2010, pp. 1–9.

[20] P. Dutta, R. Musaloiu-E, I. Stoica, and A. Terzis, "Wireless ACK collisions not considered harmful," in *In HotNets-VII The Seventh Workshop on Hot Topics in Networks*, 2008.

[21] F. Österlind, L. Mottola, T. Voigt, N. Tsiftes, and A. Dunkels, "Strawman : Resolving Collisions in Bursty Low-Power Wireless Networks," in *11th Int. Conf. on Information Processing in Sensor Networks (IPSN)*, 2012, pp. 161–172.

[22] A. Purohit, B. Priyantha, and J. Liu, "WiFlock : Collaborative Group Discovery and Maintenance in Mobile Sensor Networks," in *10th Int. Conf. on Information Processing in Sensor Networks (IPSN)*, 2011, pp. 37–48.

[23] A. Kandhalu, K. Lakshmanan, and R. Rajkumar, "U-connect: a low-latency energy-efficient asynchronous neighbor discovery protocol," in *9th Int. Conf. on Information Processing in Sensor Networks*, 2010, pp. 350–361.

[24] D. Zhang, T. He, Y. Liu, Y. Gu, and F. Ye, "Acc: generic on-demand accelerations for neighbor discovery in mobile applications," in *10th ACM Conf. on Embedded networked sensor systems (Sensys)*, 2012, pp. 169–182.

[25] B. Chen, Z. Zhou, and H. Yu, "Understanding RFID counting protocols," in *19th int. conf. on Mobile computing & networking*, 2013, pp. 291–302.