Approximate Neighbor Counting in Radio Networks

Calvin Newport[1]
Georgetown University, Washington, D.C., United States
cnewport@cs.georgetown.edu

Chaodong Zheng[2]
State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
chaodong@nju.edu.cn

Abstract

For many distributed algorithms, neighborhood size is an important parameter. In radio networks, however, obtaining this information can be difficult due to ad hoc deployments and communication that occurs on a collision-prone shared channel. This paper conducts a comprehensive survey of the approximate neighbor counting problem, which requires nodes to obtain a constant factor approximation of the size of their network neighborhood. We produce new lower and upper bounds for three main variations of this problem in the radio network model: (a) the network is single-hop and every node must obtain an estimate of its neighborhood size; (b) the network is multi-hop and only a designated node must obtain an estimate of its neighborhood size; and (c) the network is multi-hop and every node must obtain an estimate of its neighborhood size. In studying these problem variations, we consider solutions with and without collision detection, and with both constant and high success probability. Some of our results are extensions of existing strategies, while others require technical innovations. We argue this collection of results provides insight into the nature of this well-motivated problem (including how it differs from related symmetry breaking tasks in radio networks), and provides a useful toolbox for algorithm designers tackling higher level problems that might benefit from neighborhood size estimates.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Radio networks, neighborhood size estimation, approximate counting

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2018.26

Related Version A full version of the paper is available at [20], `https://arxiv.org/abs/1811.03278`.

1 Introduction

Many distributed algorithms assume nodes have advance knowledge of their neighborhood, allowing them to take steps that depend, for example, on gathering information from every neighbor (e.g., [16]), or flipping a coin weighted with their neighborhood size (e.g., [1]).

In standard wired network models, obtaining this neighbor information is often trivial (e.g., as in the LOCAL or CONGEST models). In radio networks, by contrast, this information might be harder to obtain. Specifically, because nodes in these networks are often deployed in an ad hoc manner, and subsequently communicate only on a contended shared channel, we

[1] Supported by NSF award 7773087.
[2] Supported by National Key R&D Program of China 2018YFB1003200, and NSFC 61702255.

cannot expect that they possess advance knowledge of their neighborhood. In fact, learning this information might require non-trivial feats of contention management.

Some distributed algorithms for radio networks depend on nodes possessing an estimate of their neighborhood size (e.g., [9, 10]), while other algorithms could be significantly simplified if this information was available (e.g., [22, 18, 13]). Though it is generally assumed that calculating these size estimates should not take *too* long in most settings, this problem has escaped the more systematic scrutiny applied to related tasks like contention resolution.

In this paper, we work toward filling in more of this knowledge gap. We conduct a comprehensive survey of lower and upper bounds for the approximate neighbor counting problem in the radio network model under different combinations of common assumptions for this setting. Some of our results require only extensions of existing strategies, while many others require non-trivial technical innovations.

Combined, this collection of results provides two important contributions to the study of distributed algorithms for radio networks. First, it supports a deeper understanding of the well-motivated neighbor counting problem, highlighting both its similarities and differences to related low-level radio network tasks. Second, the collection acts as a useful toolbox for algorithm designers tackling higher level problems.

Result summary. The radio network model we study describes the underlying network topology with an undirected connected graph $G = (V, E)$, with the $n = |V|$ vertices corresponding to the radio devices (usually called *nodes* in this paper), and the edges in E describing which node pairs are within communication range. For every node $u \in V$, n_u describes the number of neighbors of u in G. We sometimes call this parameter the *neighbor count* of u. In single-hop networks (i.e., G is a clique), all nodes have the same neighbor count, while in multi-hop networks these counts can differ.

The *approximate neighbor counting* problem requires nodes to calculate constant factor estimates of their neighbor counts. We study the variant where every node must obtain this estimate (e.g., during network initialization), and the variant where only a designated node must obtain this estimate (e.g., when neighborhood of a node changes). We study these variants in single-hop and multi-hop networks, and consider solutions with and without collision detection. We study both lower and upper bounds for randomized solutions. When relevant, we look at both results that hold with constant and high probability.

Our results are summarized in Table 1. Notice that we do not study both designated and all nodes counting in single-hop networks, as in this setting all nodes have the same neighbor count, making these two cases essentially identical (e.g., a designated node in a single-hop network can simply announce its count, transforming the solution to an all nodes counting solution). We also do not study constant probability solutions for all nodes counting in multi-hop networks. This follows because in the multi-hop setting the success probability applies to each individual node. A constant success probability, therefore, implies that a constant fraction of the nodes are expected to generate inaccurate neighbor counts – a result that is too weak in most scenarios. In the single-hop setting, by contrast, the success probability refers to the probability that *all* nodes generate good counts.

Also notice that two upper bounds are given for multi-hop all nodes counting without collision detection: $O(\lg^2 n_u)$ and $O(\lg^3 N)$. The first bound describes an algorithm that generates good neighbor counts but never terminates (specifically, each node must keep participating to help neighbors that are still counting). The second bound does terminate, but requires an upper bound N on the maximum possible network size. This is the only algorithm we study that requires this information to work properly.

Table 1 Summary of all approximate neighbor counting results proved in this paper. In the above table, "CD" and "no-CD" denote "with collision detection" and "without collision detection" respectively, while "high probability" is expressed with respect to the parameter in the bound. The N and N_Δ terms describe upper bounds on the maximum neighbor count in single-hop and multi-hop networks, respectively. Our lower bounds are expressed with respect to these maximum sizes, while our upper bounds are expressed with respect to the actual network sizes, with the exception of the $O(\lg^3 N)$ bound for multi-hop all nodes counting without collision detection. This is the only algorithm we study that requires knowledge of network statistics to work properly.

		with constant probability		with high probability	
		no-CD	CD	no-CD	CD
all nodes in single-hop	lower bound	$\Omega(\lg N)$	$\Omega(\lg \lg N)$	$\Omega(\lg^2 N)$	$\Omega(\lg N)$
	upper bound	$O(\lg n)$	$O(\lg \lg n)$	$O(\lg^2 n)$	$O(\lg n)$
designated node in multi-hop	lower bound	$\Omega(\lg N_\Delta)$	$\Omega(\lg \lg N_\Delta)$	$\Omega(\lg^2 N_\Delta)$	$\Omega(\lg N_\Delta)$
	upper bound	$O(\lg n_w)$	$O(\lg \lg n_w)$	$O(\lg^2 n_w)$	$O(\lg n_w)$
all nodes in multi-hop	lower bound	–	–	$\Omega(\lg^2 N_\Delta)$	$\Omega(\lg N_\Delta)$
	upper bound	–	–	$O(\lg^2 n_u)$, $O(\lg^3 N)$	$O(\lg^2 n_u)$

Discussion. For all but one cases that we have lower bounds, they match our upper bounds. For the single-hop results, these bounds also match the relevant bounds from the related single-hop contention resolution problem (c.f., [19]). In fact, most of the lower bounds in this single-hop setting follow by reduction from contention resolution. That is, we show that if you can solve approximate neighbor counting fast, then you can also solve contention resolution fast – allowing existing lower bounds from the latter to carry over to the former.

The single-hop upper bounds, however, required more than the simple application of existing contention resolution strategies. In contention resolution, for example, if you get lucky with your coin flips, and a node broadcasts alone earlier than expected, this is good news – you have solved the problem even faster! In neighbor counting, however, this "luck" might lead you to output an inaccurate size estimate. The analysis used for neighbor counting must bound the probabilities of these precocious symmetry breaking events.

Another complexity of neighbor counting (in single-hop networks) as compared to contention resolution is that *all* nodes must learn an estimate. This requires extra mechanisms to ensure that once some nodes learn a good estimate, this information is spread to all others. The most difficult single-hop case is the combination of high probability correctness and collision detection. To achieve an accurate estimate in an optimal $O(\lg n)$ rounds required the adaptation of a technique based on one-dimensional random walks [18, 3].

Obtaining lower bounds for the multi-hop designated node setting required technical innovations. In the single-hop setting, our lower bounds used reduction arguments that applied the contention resolution bounds from [19] as a black box. In the multi-hop designated node setting, by contrast, we were forced to open the black boxes and modify them to handle the issues specific to multi-hop topologies. For the particular case of collision detection and high probability, substantial new arguments were needed to transform the bound.

In the multi-hop all nodes setting, obtaining upper bounds also required techniques beyond standard symmetry breaking strategies, as each node may simultaneously participate in multiple estimation processes. Our collision detector algorithm for this case has nodes use detectable noise to notify neighbors that they are still counting. When collision detection

is not available, we consider two different approaches and hence present two algorithms. The first one returns an estimate for n_u in $O(\lg^2 n_u)$ rounds, which is correct with high probability in n_u. The second algorithm uses a "double counting" trick and takes longer time, but the returned estimate is correct with high probability in N.

Last but not least, we would like to clarify a point about our lower bound statements. As shown in Table 1, our lower bounds are expressed with respect to the maximum possible neighbor counts (e.g., N and N_Δ), whereas, to obtain the strongest possible results, our upper bounds are expressed with respect to the actual neighbor counts in the analyzed execution (e.g., n and n_u). The right way to interpret our lower bounds is that they claim in a setting where the number of participants comes from a set of N (or N_Δ) possible participants, there *exists* a subset of these participants for which the stated bound holds.

Our lower bound technique does not directly tell us anything about the *size* of the participant set that induces the slow performance. Given our matching upper bounds, however, we can conclude that the worst case participant sets for these algorithms must have a size close to the maximum bounds. Consider, for example, single-hop counting with no collision detection. The lower bound says that for each algorithm there exists a collection of no more than N participants that requires $\Omega(\lg N)$ rounds to generate a good count with constant probability. Our upper bound, on the other hand, guarantees a good count in $O(\lg n)$ rounds with constant probability, where n is the size of the participant set. It follows that when the lower bound is applied to our algorithm, the bad participant set must have a size that is polynomial in N (i.e., $n = \Theta(N^\gamma)$, for some constant $0 < \gamma \leq 1$), as otherwise the existence of both bounds is a logical contradiction.

2 Related Work

Algorithms to reduce contention and enable communication on shared channels date back to the early days of networking (c.f., [11, 6]), and remain an active area of study today. In the study of distributed algorithms for shared *radio* channels, many strategies explicitly execute approximate neighbor counting as a subroutine. For example, in their study of energy-efficient initialization with collision detection, Bordim et al. [2] propose a protocol that returns an estimate of n in the range $[n/(16 \lg n), 2n/\lg n]$ within $O(\lg^2 n)$ time, while requiring each node to be awake for at most $O(\lg n)$ rounds. Similarly, Gilbert et al. [10] use approximate neighbor counting as part of a neighbor discovery protocol in cognitive radio networks. It is also common for algorithms in this setting to simply assume these estimates are provided in advance. E.g., the often-used *decay* strategy introduced by Bar-Yehuda et al. [1], requires a bound on local neighborhood size to limit the estimates it tests.

As mentioned throughout this paper, neighbor counting is often closely related to contention resolution, which requires a single node to broadcast alone on the channel. Some common contention resolution strategies implicitly provide this approximation as a side-effect of their operation (e.g., [22, 18, 13]). At the same time, under some assumptions, a good estimate simplifies the problem of contention resolution. As we detail throughout this paper, however, this relationship is not exact. Lower bounds for neighbor counting often require more intricate arguments than contention resolution, and in some cases, contention resolution algorithms require nontrivial extra analysis and mechanisms to provide counts. Teasing apart this intertwined relationship is one of the main contributions of this paper.

Others have directly studied approximate neighbor counting in radio networks. Jurdzinski et al. [12] develop an algorithm that provides a constant factor approximation of n within $O(\lg^{2+\delta} n)$ time without collision detection for arbitrary constant $\delta > 0$. Their algorithm

guarantees that no node participates in more than $O((\lg \lg n)^{\delta})$ rounds. (Our relevant algorithm only needs $O(\lg^2 n)$ rounds, but consumes more energy.) Caragiannis et al. [4] devise two constant-factor approximation algorithms: the first one requires collision detection and takes $O((\lg n) \cdot (\lg \lg n))$ time, while the second one works without collision detection and takes $O(\lg^2 n)$ time. (Our relevant algorithms only need $O(\lg n)$ rounds with collision detection, and perform as well as theirs without collision detection.) In [14, 15], the authors discuss how to approximate network size when adversaries are present.

Approximate neighbor counting has also been studied in the BEEPING model [7], which is similar to, but somewhat weaker than, the standard radio network model. In this setting, Chen et al. [5] conduct an excellent mini survey on recent works in RFID counting (e.g., [24, 21, 23, 5]). They conclude that a two-phase approach is the key to achieve efficient and accurate RFID counting. They also prove several lower bounds, one of which shows $\Omega(\lg \lg n)$ rounds are needed to obtain a constant factor approximation with constant probability. More recently, Brandes et al. [3] study how to efficiently estimate the size of a single-hop BEEPING network: they provide both lower and upper bounds for a parameterized approximation accuracy. Notice, the main objective of [5] and [3] differs from ours not just in the model, but in that they seek a $(1 + \epsilon)$ approximation of n for any $\epsilon > 0$ (ϵ can be non-constant). Nonetheless, they both use constant factor approximation as a key subroutine.

3 Model and Problem

We consider a synchronous radio network. We model the topology of this network with a connected undirected graph $G = (V, E)$, with the $n = |V|$ vertices corresponding to the radio devices (usually called *nodes* in this paper), and the edges in E describing which node pairs are within communication range.

For each node $u \in V$, we use Γ_u to denote the set of neighbors of u, and use $n_u = |\Gamma_u|$ to denote the number of neighbors of u. Let $n_\Delta = \max_{u \in V}\{n_u\}$. Our algorithms assume $n_u \geq 1$. That is, we do not confront the possibility of a node isolated from the rest of a multi-hop network, or a single-hop network consisting of only a single node (we see the so-called *loneliness detection* problem as an interesting but somewhat orthogonal challenge; e.g., [8]). For the ease of presentation, we assume n_u and n are always a power of two. This assumption does not affect the correctness or asymptotic time complexities of our results. We define N and N_Δ to be upper bounds on the maximum possible size of n and n_Δ, respectively. To obtain the strongest and most general possible results, our algorithms are *not* provided with knowledge of N and N_Δ, with the exception of an $O(\lg^3 N)$ time algorithm for multi-hop all nodes counting without collision detection.

We divide time into discrete and synchronous *slots* that we also sometimes call *rounds*. We assume all nodes start execution during the same slot. (The definition of "neighbor counting" becomes complicated once nodes can activate in different time slots.) These assumptions imply nodes have access to a global clock. We assume each node is equipped with a half-duplex radio transceiver. That is, in each time slot, each node can choose to broadcast or listen, but cannot do both. If a node chooses to broadcast, then it gets no feedback from the communication channel. If a node chooses to listen and no neighbors of it broadcasts, then the node hears nothing (i.e., silence). If a node chooses to listen and exactly one of its neighbors broadcasts, then the node receives the message from that neighbor. Finally, if a node chooses to listen and at least two of its neighbors broadcast, then the result depends on the availability of a *collision detection* mechanism: if collision detection is available, then the listening node hears noise; otherwise, the listening node hears nothing.

As a result, without collision detection, a listening node cannot tell whether there are no neighbors broadcasting or there are multiple neighbors broadcasting.

In this paper, we are interested in the *approximate neighbor counting* problem. This problem requires selected node(s) to obtain a constant factor approximation of their neighborhood size(s). In more detail, let constant $\tilde{c} \geq 1$ be the fixed approximation threshold for this problem. Each node u that produces an estimate $\hat{n}_u$ must satisfy $n_u \leq \hat{n}_u \leq \tilde{c} \cdot n_u$. We consider three variations of this problem that differ with respect to the allowable network topologies and requirements on which nodes produce an estimate. The first variant assumes G is single-hop and all nodes must produce an *identical* estimate. The second variant assumes G is multi-hop, but only a *single* designated node w must produce an estimate. The third variant is the same as the second, except that now *every* node must produce an estimate. We study randomized algorithms that are proved to be correct with a given probability p. In the single-hop variant, p describes the probability of the event in which all nodes generate a single good approximation. In the multi-hop variants, by contrast, p is the probability that an individual counting node generates a good approximation.

Throughout this paper, we cite the related *contention resolution* problem. In single-hop networks, the contention resolution problem is solved once some node broadcasts alone. Later in the paper, we consider a version of multi-hop contention resolution in which a single designated node must receive a message from a neighbor to solve the problem.

Finally, in the following, we say an event occurs *with high probability in parameter* k (or "w.h.p. in k") if it occurs with probability at least $1 - 1/k^{\gamma}$, for some constant $\gamma \geq 1$.

4 Lower Bounds

In this section, we presents our lower bounds for the approximate neighbor counting problem. We begin, in Section 4.1 by looking at lower bounds that can be proved by reducing from the contention resolution problem. That is, in that subsection, we prove lower bounds by arguing that solving neighbor counting fast implies an efficient algorithm to contention resolution, allowing the relevant contention resolution lower bounds to apply.

We employ this approach to derive bounds for constant probability and high probability counting with no collision detection in both single-hop and designated node multi-hop settings. We also apply this approach to derive bounds for constant probability counting with collision detection in these settings. We cannot, however, apply this approach to high probability counting with collision detection, as the reduction itself is too slow compared to the desired bounds. We note that for the single-hop arguments, we leverage existing contention resolution bounds from [19]. For the multi-hop arguments, however, we must first generalize the results from [19] to hold for the considered network topology.

In Section 4.2, we look at lower bounds for high probability approximate neighbor counting with collision detection in both single-hop and designated node multi-hop settings. Unlike in Section 4.1, we cannot deploy a reduction-based argument. We instead prove a new lower bound that directly argues a sufficiently accurate estimate requires the stated rounds.

Finally, in Section 4.3 we look at lower bounds for the remaining case of multi-hop all nodes counting. We establish these bounds by reduction from designated node multi-hop bounds, as solving all nodes counting trivially also solves designated node counting.

4.1 Lower Bounds via Reduction from Contention Resolution

We begin with our lower bound arguments that rely on reductions from contention resolution. For the single-hop scenario, we can reduce from single-hop contention resolution and apply existing lower bounds from [19]. (See full version of the paper [20] for details on contention

resolution lower bounds in single-hop networks.) For multi-hop designated node counting, however, we must first prove new contention resolution lower bounds.

In particular, consider the definition of multi-hop contention resolution in which there is a well-defined designated node w, and the goal is for exactly one of w's neighbors – which is a size n_w subset drawn from a size N_Δ universe – to broadcast alone in some time slot. At first glance, this problem might seem easier than single-hop contention resolution as we are provided with a designated node w that could coordinate its neighbors in their quest to break symmetry among themselves. We prove, however, that this is not the case: the lower bounds are the same as their single-hop counterparts. In more detail, we prove the following two lemmas by adapting the techniques from [19] to this new set of assumptions (see full version of the paper [20] for the omitted proofs of this section):

▶ **Lemma 1.** *Let $\mathcal{A}$ be an algorithm that solves contention resolution in $g(N_\Delta)$ time slots with probability p in multi-hop networks with no collision detection. It follows that: (a) if p is some constant, then $g(N_\Delta) \in \Omega(\lg N_\Delta)$; and (b) if $p \geq 1 - 1/N_\Delta$, then $g(N_\Delta) \in \Omega(\lg^2 N_\Delta)$.*

▶ **Lemma 2.** *Let $\mathcal{A}$ be an algorithm that solves contention resolution in $g(N_\Delta)$ time slots with probability p in multi-hop networks with collision detection. It follows that if p is some constant, then $g(N_\Delta) \in \Omega(\lg \lg N_\Delta)$.*

With the needed contention resolution lower bounds in hand, we turn our attention to reducing this problem to approximate neighbor counting. Take the single-hop scenario as an example, the basic idea behind the reduction is that once nodes have an estimate $\hat{n}$ of n, they can simply broadcast with probability $1/\hat{n}$ in each time slot. If this estimate is good, then in each time slot, they have a constant probability of isolating a broadcaster, thus solving contention resolution. Moreover, repeating this step multiple times increases the chance of success proportionally. Building on these basic observations, we prove the following:

▶ **Lemma 3.** *Assume there exists an algorithm $\mathcal{A}$ that solves approximate neighbor counting in $h(N)$ (or, $h(N_\Delta)$ in the multi-hop scenario) time slots with probability p. Then, there exists an algorithm $\mathcal{B}$ that solves contention resolution in $2(h(N) + k)$ (resp., $2h(N_\Delta) + k$ in the multi-hop scenario) time slots with probability at least $(1 - e^{-k/(4\tilde{c})}) \cdot p$. Here, $k \geq 1$ is an integer, and $\tilde{c} \geq 1$ is the constant defined in Section 3.*

Combining the reduction described in Lemma 3 with the single-hop lower bounds for contention resolution from [19] and the new multi-hop lower bounds proved above, we get the following lower bounds for approximate neighbor counting:

▶ **Theorem 4.** *In a single-hop radio network containing at most N nodes:*

- *When collision detection is not available, solving approximate neighbor counting with constant probability requires $\Omega(\lg N)$ time in the worst case; solving approximate neighbor counting with high probability in N requires $\Omega(\lg^2 N)$ time in the worst case.*
- *When collision detection is available, solving approximate neighbor counting with constant probability requires $\Omega(\lg \lg N)$ time in the worst case.*

In a multi-hop radio network in which the designated node has at most N_Δ neighbors:

- *When collision detection is not available, solving approximate neighbor counting with constant probability requires $\Omega(\lg N_\Delta)$ time in the worst case; solving approximate neighbor counting with high probability in N_Δ requires $\Omega(\lg^2 N_\Delta)$ time in the worst case.*
- *When collision detection is available, solving approximate neighbor counting with constant probability requires $\Omega(\lg \lg N_\Delta)$ time in the worst case.*

4.2 Custom Lower Bounds for High Probability and Collision Detection

At this point, for single-hop and designated node multi-hop variants of the approximate neighbor counting problem, the only lower bounds missing are the ones of ensuring high success probability with collision detection. As we detail in full version of the paper [20], our previous reduction-based approach no longer works in these scenarios. (Roughly speaking, the reduction itself takes at least as long as the lower bound we intend to prove.) Therefore, we must construct custom lower bounds for this problem and exact set of assumptions.

We start by proving the following combinatorial result:

▶ **Lemma 5.** *Let c and k be two positive integers such that $c \leq k$. Let $\mathcal{R}$ be the set containing all size c subsets from $[k] = \{1, 2, \cdots, k\}$. Let $\mathcal{H}$ be an arbitrary set of size less than $\lg(k/c)$ such that each element in $\mathcal{H}$ is a subset of $[k]$. Then, there exists some $R \in \mathcal{R}$ such that for each $H \in \mathcal{H}$, either $R \subseteq H$ or $R \cap H = \emptyset$.*

Intuitively, a set $\mathcal{H}$ can be interpreted as a *broadcast schedule* generated by an algorithm $\mathcal{A}$: a node labeled i broadcasts in slot j if and only if it is activated, and i is in the j^{th} set in $\mathcal{H}$. Given this interpretation, Lemma 5 suggests: for both the single-hop and the multi-hop designated node scenario, for any approximate neighbor counting algorithm $\mathcal{A}$, and for any broadcast schedule generated by $\mathcal{A}$ of length less than $\lg(k/c)$, there exists a set of c nodes (or a set of c neighbors of the designated node in the multi-hop scenario) such that if these c nodes are activated and execute $\mathcal{A}$, then during each of the first $\lg(k/c) - 1$ time slots, either none of them broadcast or all of them broadcast. This further implies, if only two of these c nodes are activated, then their view (of the first $\lg(k/c) - 1$ time slots of the execution) is indistinguishable from the case in which all of these c nodes are activated.

Now, imagine an adversary who samples a size c subset from $\mathcal{R}$ with uniform randomness, and then flips a fair coin to decide whether to activate all these c nodes, or just two of them. If the adversary happens to have chosen the set R proved to exist in Lemma 5, then by the end of slot $\lg(k/c) - 1$, algorithm $\mathcal{A}$ cannot distinguish between two and c nodes. Notice, if c is large compared to the approximation threshold $\tilde{c}$, then this difference matters: outputting two when the real count is c (or vice versa) is unacceptable. Thus, in such case, the algorithm gets the right answer with only probability $1/2$ – not enough for high success probability.

A complete and rigorous proof for the above intuition is actually quite involved, again see full paper [20] for more details. In the end, we obtain the following lower bounds:

▶ **Theorem 6.** *Assume collision detection is available, then:*

- *In a single-hop radio network containing at most N nodes, solving approximate neighbor counting with high probability in N requires $\Omega(\lg N)$ time in the worst case.*
- *In a multi-hop radio network in which the designated node has at most N_Δ neighbors, solving approximate neighbor counting with high probability in N_Δ requires $\Omega(\lg N_\Delta)$ time in the worst case.*

4.3 All Nodes Multi-Hop Lower Bounds

If an algorithm can solve multi-hop all nodes approximate neighbor counting, then clearly the same algorithm can be used to solve multi-hop designated node approximate neighbor counting, with same time complexity and success probability. Therefore, the lower bounds we previously proved for the latter variant naturally carries over to the former variant:

▶ **Theorem 7.** *In a multi-hop radio network containing at most N nodes:*

- *When collision detection is not available, solving approximate neighbor counting with high probability in N requires $\Omega(\lg^2 N)$ time in the worst case.*
- *When collision detection is available, solving approximate neighbor counting with high probability in N requires $\Omega(\lg N)$ time in the worst case.*

5 Upper Bounds

In this section, we describe and analyze several randomized algorithms that solve the approximate neighbor counting problem. We will begin with single-hop all nodes counting. Specifically, four algorithm are presented for this variant, each based on a different approach. Though most of the strategies used are previously known, extensions to design and analysis are often needed. We then introduce three algorithms for multi-hop all nodes counting, including one which is particularly interesting, as it uses a "double counting" trick that is not related to contention resolution at all to obtain high success probability. Finally, we briefly discuss solutions for multi-hop designated node counting, as most of these algorithm are simple variations of their counterparts for single-hop all nodes counting.

Due to space constraint, if not otherwise stated, complete proofs for lemma and theorem statements are provided in the appendix. Nonetheless, we will usually discuss the intuitions or high-level strategies for proving them.

5.1 Single-Hop Networks: No Collision Detection

Our algorithms often adopt a classical technique inspired by the contention resolution literature: "*guess and verify*". In more detail, take a *guess* about the count, and then *verify* its accuracy; if the guess is good enough then we are done, otherwise take another guess and repeat. This simple approach is versatile: depending on how the guesses are made and verified, many variations exist, resulting in efficient algorithms suitable for different settings.

A standard approach to this guessing is to use a geometric sequence with common ratio two, which is usually called (exponential) *decay* [1]. This sequence leverages the fact that we only need a constant factor estimate to speed things up. Particularly, if the real count is n, then only $O(\lg n)$ iterations are needed before reaching an accurate estimate.

Once a guess is made, we need to verify its accuracy. To accomplish this, it is sufficient to let each participating node broadcast with a probability proportional to the reciprocal of the guess, and then observe the status of the channel. The intuition is simple: underestimate will result in collision and overestimate will result in silence; and we expect one node to broadcast alone – a distinguishable event – iff the estimate is accurate enough. The algorithms described below adapt this general approach to their specific constraints.

Constant probability of success. We now present Count-SH-noCD-Const. (In the algorithm's name, SH means "single-hop", noCD means "no collision detection", and Const means "success with constant probability".) This algorithm applies the most basic form of the "guess and verify" strategy. It provides a correct estimate with constant probability in $O(\lg n)$ time for single-hop radio networks, when collision detection is not available.

Count-SH-noCD-Const contains multiple iterations, each of which has two time slots. In the i^{th} iteration, nodes assume $n \approx 2^i$, and verify whether this estimate is accurate or not. More specifically, in the first time slot within the i^{th} iteration, each node will broadcast a beacon message with probability $1/2^i$ and listen otherwise. If a node decides to listen and hears a beacon message in the first time slot, then it will set its estimate to 2^{i+2} and terminate after this iteration. That is, if a single node u broadcasts alone in the first time

slot of iteration i, then all listening nodes – which is all nodes except u – will terminate by the end of this iteration, with 2^{i+2} being their estimate. Notice, we still need to inform u about this estimate, which is the very purpose of the second time slot within each iteration. More specifically, in the second time slot within the i^{th} iteration, for each node u, if it has heard a beacon message in the first time slot of this iteration, then it will broadcast a stop message with probability $1/(2^i - 1)$. Otherwise, if u has broadcast in the first time slot of this iteration, then it will listen in this second time slot. Moreover, it will terminate with its estimate set to 2^{i+2}, if it hears a stop message in this second time slot.

Despite its simplicity, proving the correctness of Count-SH-noCD-Const requires efforts beyond what would suffice for basic contention resolution. First, by carefully calculating and summing up the failure probabilities, we show no node will terminate during the first $\lg n - 3$ iterations, with at least constant probability. Then, we show during iteration i where $\lg n - 2 \leq i \leq \lg n$, either no node terminates or all nodes terminate, with at least constant probability. Finally, we prove that if all nodes are still active by iteration $\lg n$, then all of them will terminate by the end of it, again with at least constant probability. Complete analysis can be found in the full paper [20], here we present only the main theorem:

▶ **Theorem 8.** *The* Count-SH-noCD-Const *approximate neighbor counting algorithm ensures the following properties with constant probability when executed in a single-hop network with no collision detection: (a) all nodes terminate simultaneously within $O(\lg n)$ slots; and (b) all nodes obtain the same estimate of n, which is in the range $[n, 4n]$.*

High probability of success. Observe that in the aforementioned simplest form of "guess and verify", as the estimate increases, the probability that multiple nodes broadcast decreases, and the probability that no node broadcasts increases. A more interesting metric is the probability that a single node broadcasts alone: it first increases, and then decreases; not surprisingly, the peak value is reached when the estimate is the real count. These facts suggest, for each estimate $\hat{n}$, we could repeat the procedure of broadcasting with probability $1/\hat{n}$ multiple times, and use the *fraction* of noisy/silent/clear-message slots to determine the accuracy of the estimate. This method can provide stronger correctness guarantees, but complicates *termination detection* (i.e., when should a node stop), as different nodes may observe different fraction values. For example, if some nodes have already obtained a correct estimate but terminate too early, then remaining nodes might never get correct estimates, since there are fewer nodes remaining. The situation becomes more challenging when an upper bound of the real count is not available. To resolve this issue, sometimes, we have to carefully craft and embed a "consensus" mechanism.

Count-SH-noCD-High highlights our above discussion. This algorithm contains multiple iterations, each of which has three phases. In the i^{th} iteration, nodes assume $n \approx 2^i$. The first phase of each iteration i – which contains $\Theta(i)$ time slots – is used to verify the accuracy of the current estimate. In particular, in each slot within the first phase, each node will broadcast a beacon message with probability $1/2^i$, and listen otherwise. By the end of the first phase, each node will calculate the fraction of time slots (among all its listening slots in this phase) in which it has heard a beacon message. For each node, if at the end of the first phase of some iteration j, this fraction value has reached $1/2e$ for the first time since the start of execution, the node will set 2^j as its *private estimate* for n. Recall nodes might not obtain private estimates simultaneously, thus they cannot simply terminate and output private estimates as the final estimate. This is the place where the latter two phases come into play. More specifically, in the second phase, nodes that have already obtained private estimates will try to broadcast informed messages to signal other nodes to stop. In fact,

hearing an informed message is the only situation in which a node can safely terminate, even if the node has already obtained its private estimate. On the other hand, the third phase is used to deal with the case in which one single "unlucky" node successfully broadcasts an informed message during phase two (thus terminate all other nodes), but never gets the chance to successfully receive an informed message (thus cannot terminate along with other nodes). Complete description of Count-SH-noCD-High can be found in full paper [20].

To prove the correctness of Count-SH-noCD-High, we need to show: (a) nodes can correctly determine the accuracy of their estimates; and (b) all nodes terminate simultaneously and output identical estimate. Part (b) follows from our careful protocol design, as phase two and three in each iteration act like a mini "consensus" protocol, allowing nodes to agree on when to stop. Proving part (a), on the other hand, needs more effort. Recall we use the fraction of clear message slots to determine the accuracy of an estimate, and the expected fraction value should be identical for all nodes. However, due to random chances, the actual fraction value observed by each node might deviate from expectation. If we have an upper bound N of n, then by making the first phase to contain $\Theta(\lg N)$ slots, Chernoff bounds [17] will enforce the observed fraction value to be tightly concentrated around its expectation. In our case, N is not available, and we rely on more careful analysis. Specifically, during iterations one to $\lg(n/(a \ln n))$ where a is some sufficiently large constant, in each time slot in phase one, at least two nodes will broadcast (since the estimate is too small), thus no node will obtain private estimate in these iterations. Starting from iteration $\lg(n/(a \ln n))$, the length of phase one is long enough so that concentration inequalities will ensure the observed fraction value is close to its expectation. Building on these observations, we can eventually conclude the following theorem (again, see [20] for full analysis):

▶ **Theorem 9.** *The* Count-SH-noCD-High *approximate neighbor counting algorithm ensures the following properties with high probability in n when executed in a single-hop network with no collision detection: (a) all nodes terminate simultaneously within $O(\lg^2 n)$ slots; and (b) all nodes obtain the same estimate of n, which is in the range $[n, 4n]$.*

5.2 Single-Hop Networks: Collision Detection

Without collision detection, the feedback to a listening node is either silence or a message. Failing to receive a message, therefore, does not hint the nature of the failure: either no node is sending, or multiple nodes are sending. As a result, in the two previous algorithms, when nodes "guess" the count, they have to do it in a *linear* manner: start with a small estimate, and double if the guess is incorrect. With collision detection, by contrast, listening nodes can distinguish whether too few (i.e., zero) or too many (i.e., at least two) nodes are broadcasting. As first pointed out back in the 1980's [22], this extra power enables an exponential improvement over linear searching, since nodes can now perform a *binary search*.

Constant probability of success. Here we leverage the aforementioned binary search strategy to return a constant factor estimate of n in $O(\lg \lg n)$ time, with at least some constant probability. Recall efficient binary search requires a rough upper bound of n as input. To this end, we first introduce an algorithm called EstUpper-SH: it can provide a polynomial upper bound of n within $O(\lg \lg n)$ time. At a high level, EstUpper-SH is doing a linear "guess and verify" search to estimate $\lg n$. (Notice, it is *not* estimating n.) This strategy, to the best of our knowledge, is first discussed by Willard in the seminal paper [22], and has later been used in other works (see, e.g., [5, 3]). Due to space constraint, detailed description and analysis of EstUpper-SH are provided in full version of the paper [20].

Once this estimate is obtained, we switch to the main logic of Count-SH-CD-Const. This algorithm contains multiple iterations, each of which has four time slots. In each iteration i, all nodes have a lower bound a_i and an upper bound b_i, and will test whether the median $m_i = \lfloor (a_i + b_i)/2 \rfloor$ is close to $\lg n$ or not. More specifically, in the first time slot in iteration i, each node will broadcast a beacon message with probability $1/2^{m_i}$, and listen otherwise. Listening nodes will use the channel status they observed to adjust a_i (or b_i), or terminate and output the final estimate. On the other hand, the other three time slots in each iteration allow nodes that have chosen to broadcast in the first time slot to learn the channel status too, with the help of the nodes that have chosen to listen in the first time slot. (See full paper [20] for complete description of Count-SH-CD-Const.)

To prove Count-SH-CD-Const can provide a correct estimate, we demonstrate that during one execution of Count-SH-CD-Const: (a) whenever m_i is too large or too small, all nodes can correctly detect this and adjust a_i or b_i accordingly; and (b) when m_i is a good estimate, all nodes can correctly detect this as well and stop execution. The full analysis can be found in the full version of the paper [20], here we state only the main theorem:

▶ **Theorem 10.** *The* Count-SH-CD-Const *approximate neighbor counting algorithm ensures the following properties when executed in a single-hop network with collision detection: (a) all nodes terminate simultaneously; and (b) with at least constant probability, all nodes obtain the same estimate of n in the range $[n, 4n]$ within $O(\lg \lg n)$ time slots.*

High probability of success. Our last algorithm for the single-hop scenario is Count-SH-CD-High. It significantly differs from the other algorithms studied so far in that it does *not* use a "guess and verify" strategy. Instead, it deploys a *random walk* to derive an estimate. The use of random walks for contention resolution was introduced by Nakano and Olariu [18], in the context of leader election in radio networks. It was later adopted by Brandes et al. [3] for solving approximate counting in beeping networks.

Prior to executing Count-SH-CD-High, nodes will first use $O(\lg \lg n)$ time slots to run EstUpper-SH to obtain a polynomial upper bound of n. Call this upper bound $\hat{N}$. All nodes then perform a random walk, the state space of which consists of potential estimates of n. More specifically, Count-SH-CD-High contains $\Theta(\lg \hat{N})$ iterations, each of which has three time slots. In each iteration, all nodes maintain a current estimate on n which is denoted by $\hat{n}$. (Initially, $\hat{n}$ is set to $\hat{N}$.) In the first slot in a iteration, each node will broadcast a beacon message with probability $1/\hat{n}$, and listen otherwise. If a node hears silence, it will decrease $\hat{n}$ by a factor of four; if a node hears noise, it will increase $\hat{n}$ by a factor of four; and if a node hears a beacon message, it will keep $\hat{n}$ unchanged. Similar to what we have done in Count-SH-CD-Const, in each iteration, the nodes that have chosen to listen in the first time slot will use the latter two slots to help nodes that have chosen to broadcast in the first time slot to learn the channel status of the first time slot. After these $\Theta(\lg \hat{N})$ iterations, all nodes will use $4\tilde{n}$ to be the final estimate of n, where $\tilde{n}$ is the most frequent estimate used by the nodes during the $\Theta(\lg \hat{N})$ iterations.

The high-level intuition of Count-SH-CD-High is: when the estimate is too large or too small, it will quickly shift towards correct estimates; and when the estimate is correct, it will remain unchanged. Therefore, the most frequent estimate will likely to be a correct one. Complete analysis is deferred to full paper [20], here we provide only the main theorem:

▶ **Theorem 11.** *The* Count-SH-CD-High *approximate neighbor counting algorithm ensures the following properties when executed in a single-hop network with collision detection: (a) all nodes terminate simultaneously; and (b) with high probability in n, all nodes obtain the same estimate of n in the range $[n, 64n]$ within $O(\lg n)$ time slots.*

5.3 Multi-Hop with All Nodes Counting: No Collision Detection

All nodes counting in a multi-hop network is challenging as different nodes may have significantly different number of neighbors. In this part, we present two algorithms that attempt to overcome this obstacle, the second of which is particularly interesting.

We begin with the first algorithm – called Count-All-noCD– which still relies on the linear "guess and verify" approach. However, it requires the upper bound N_Δ as an input parameter to enforce termination. Nonetheless, for each node, Count-All-noCD always returns an accurate estimate, even when knowledge of N_Δ is absent.

In more detail, Count-All-noCD contains $\lg N_\Delta$ iterations, and the i^{th} iteration contains $\Theta(i)$ time slots. In each slot in iteration i, each node will choose to be a broadcaster or a listener each with probability 1/2. If a node chooses to be a listener in a time slot, it will simply listen. Otherwise, if a node chooses to be a broadcaster, it will broadcast a beacon message with probability $1/2^i$, and do nothing otherwise. After an iteration i, for a node u, if for the first time since the beginning of protocol execution, it has heard beacon messages in at least 1/10 fraction of slots among the listening slots (within this iteration), then u will use 2^{i+3} as its estimate for n_u. Proving the correctness of Count-All-noCD borrows heavily from our analysis of Count-SH-noCD-High (see the full version of the paper [20] for more details), here we only state the main theorem:

▶ **Theorem 12.** *For each node u, the* Count-All-noCD *approximate neighbor counting algorithm ensures the following with high probability in n_u when executed in a multi-hop network with no collision detection: u will obtain an estimate of n_u in the range $[n_u, 4n_u]$ within $O(\lg^2 n_u)$ time. Moreover, u will terminate after $O(\lg^2 N_\Delta)$ time when N_Δ is known.*

Notice that in Count-All-noCD, for a node u, the high correctness guarantee is with respect to n_u. This implies, when n_u is some constant, the probability that the obtained estimate is desirable is also a constant. Sometimes, we may want *identical* and *high* correctness guarantees for *all* estimates, such as high probability in n. Our second algorithm – which is called Count-All-noCD-2– achieves this goal, at the cost of accessing N and demanding longer execution time. (Count-All-noCD only needs N_Δ to enforce termination, while Count-All-noCD-2 needs N to work properly.)

Careful readers might suspect Count-All-noCD-2 just extends the length of each iteration of Count-All-noCD to $\Theta(\lg N)$. Unfortunately, this simple modification is not sufficient: we still cannot change the fact that when a node u listens, the number of broadcasters among its neighbors is concentrated to $n_u/2$ only with high probability in n_u.

Instead, Count-All-noCD-2 takes a different approach, the core of which is a "double counting" trick. Specifically, Count-All-noCD-2 contains $L = \Theta(\lg N)$ iterations, each of which has $\Theta(\lg^2 N)$ time slots. At the beginning of each iteration, each node chooses to be a broadcaster or a listener each with probability 1/2. Then, by applying the "guess and verify" strategy, each listener will spend the $\Theta(\lg^2 N)$ time slots in this iteration to obtain a constant factor estimate on the number of neighboring broadcasters. When all $\Theta(\lg N)$ iterations are done, each node will sum the estimates it has obtained, divide it by $L/4$, and output the result as its estimate for the neighborhood size.

Due to space constraint, detailed description for each iteration is deferred to the full paper [20]. We only note here that the estimates obtained by the listeners are quite accurate:

▶ **Lemma 13.** *Consider an arbitrary iteration during the execution of* Count-All-noCD-2*, assume node u is a listener with m neighboring broadcasters. By the end of this iteration, node u will obtain an estimate of m in the range $[m, 4m]$, with high probability in N.*

We can now state and prove the guarantees provided by Count-All-noCD-2:

▶ **Theorem 14.** *The* Count-All-noCD-2 *approximate neighbor counting algorithm ensures the following properties when executed in a multi-hop network with no collision detection: (a) all nodes terminate after $O(\lg^3 N)$ time slots; and (b) with high probability in N, for each node u, the node will obtain an estimate of n_u in the range $[n_u, 5n_u]$.*

Proof sketch. Consider a node u and one of its neighbor v. Assume Count-All-noCD-2 contains $a \lg N$ iterations, where a is a sufficiently large constant. In expectation, in $(a/4) \cdot \lg N$ iterations, u will be listener and v will be broadcaster. Apply a Chernoff bound, we know u will be listener and v will be broadcaster in at least $(1-\delta) \cdot (a/4) \cdot \lg N$ iterations, and at most $(1+\delta) \cdot (a/4) \cdot \lg N$ iteration, w.h.p. in N. Here, $0 < \delta < 1$ is a small constant determined by a. Take a union bound over all $O(N)$ neighbors of u, we know this claim holds true for them as well. Therefore, if u were able to accurately count the number of broadcasting neighbors without any error in each listening iteration, the sum it will obtain would be in the range $[(1-\delta) \cdot (a/4) \cdot \lg N \cdot n_u, (1+\delta) \cdot (a/4) \cdot \lg N \cdot n_u]$, w.h.p. in N.

Now, due to Lemma 13, we know the actual sum of counts u will obtain is in the range $[(1-\delta) \cdot (a/4) \cdot \lg N \cdot n_u, 4 \cdot (1+\delta) \cdot (a/4) \cdot \lg N \cdot n_u]$, w.h.p. in N. As a result, according to our algorithm description, the final estimate u will obtain is in the range $[n_u, 5n_u]$, w.h.p. in N. Take a union bound over all nodes, the theorem is proved. ◀

5.4 Multi-Hop with All Nodes Counting: Collision Detection

In Count-All-noCD, we resolve the termination detection problem by accessing N_Δ. This allows nodes to run long enough so that they could be sure that everyone has a chance to learn what it needed to learn. With the addition of collision detection, however, the assumption that nodes know N_Δ can be removed for many network topologies. In particular, we can leverage the idea that neighbors of u that *have not* obtained estimates yet can use noise to *reliably* inform u that they wish u to continue. We call this algorithm Count-All-CD.

Count-All-CD contains multiple iterations, each of which has two parts. The first part of any iteration i is identical to iteration i of Count-All-noCD. The second part, on the other hand, helps nodes to determine when to stop. In particular, the second part of iteration i contains a single slot. For a node u, if it has not obtained an estimate of n_u by the end of the first part of iteration i yet, then in the second part, it will broadcast a continue message. On the other hand, if u has already obtained an estimate of n_u by the end of the first part of iteration i, it will simply listen in part two. Moreover, u will continue into the next iteration iff it hears continue or noise during part two. The guarantees provided by Count-All-CD are stated below, and the proof of it is provided in the full paper [20].

▶ **Theorem 15.** *The* Count-All-noCD *approximate neighbor counting algorithm ensures the following properties for each node u when executed in a multi-hop network with collision detection: (a) u will obtain an estimate of n_u in the range $[n_u, 4n_u]$ within $O(\lg^2 n_u)$ time slots, with high probability in n_u; and (b) if $\sum_{v \in \Gamma_u \cup \{u\}} (1/n_v) < 1$, then u will terminate within $(\max_{v \in \Gamma_u \cup \{u\}} \{\lg n_v\})^2$ time slots, with probability at least $1 - \sum_{v \in \Gamma_u \cup \{u\}} (1/n_v)$.*

A key point about the above termination bound is that it requires $\sum_{v \in \Gamma_u \cup \{u\}} (1/n_v) < 1$. (In fact, this constraint can be relaxed to $\sum_{v \in \Gamma_u \cup \{u\}} (1/n_v^\gamma) < 1$, for an arbitrarily chosen constant $\gamma \geq 1$.) If this is not the case (e.g., in a dense star network), the termination detection mechanism might not work properly. In that situation, the default dependence on N_Δ from the no collision detection case can be applied as a back-up.

5.5 Multi-Hop with Designated Node Counting

Compared to the all nodes counting variant, multi-hop neighbor counting with only the designated node is easier: the strategies we previously used for single-hop counting are still applicable, and the introduction of the designated node can actually make coordination easier. (In particular, this node can greatly simplify termination detection). Due to space constraint, we defer the upper bounds for this variant to the full version of the paper [20].

We note that one interesting algorithm in this variant is Count-Desig-noCD-Const, which achieves constant success probability without collision detection. Count-Desig-noCD-Const differs from its single-hop counterpart (i.e., Count-SH-noCD-Const) in that it uses fraction of clear message slots to determine the accuracy of nodes' estimate. The primary reason we develop this algorithm is that the success probability of Count-SH-noCD-Const is *fixed*. In contrast, in Count-Desig-noCD-Const, by tweaking the running time (up to some constant factor), the success probability – despite being a constant – can be adjusted accordingly.

6 Discussion

We see at least two problems that worth further exploration. First, how do termination requirements affect the complexity of the problem? This is particularly interesting in the multi-hop all nodes counting scenario, when knowledge of N_Δ or N is not available. Count-All-noCD shows lower bound can be achieved at the cost of no termination, but how much time must we spend if termination needs to be enforced, or is simply impossible without knowing N_Δ or N? Another open problem concerns the gap between the lower and upper bounds, in the multi-hop all nodes counting scenario with collision detection. On the one hand, the lower bound might be loose as it is a simple carry over, ignoring the possibility that all nodes counting could be fundamentally harder than designated node counting. Yet on the other hand, we have not found a way to leverage collision detection to reduce algorithm runtime (e.g., it seems hard to run multiple instances of binary search in parallel). Currently, our best guess is that *both* the lower bound and the upper bound are not tight.

References

1. Reuven Bar-Yehuda, Oded Goldreich, and Alon Itai. On the Time-complexity of Broadcast in Radio Networks: An Exponential Gap Between Determinism and Randomization. In *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing*, PODC '87, pages 98–108. ACM, 1987.
2. Jacir Luiz Bordim, JiangTao Cui, Tatsuya Hayashi, Koji Nakano, and Stephan Olariu. Energy-Efficient Initialization Protocols for Ad-hoc Radio Networks. In *International Symposium on Algorithms and Computation*, ISAAC '99, pages 215–224. Springer, 1999.
3. Philipp Brandes, Marcin Kardas, Marek Klonowski, Dominik Pajak, and Roger Wattenhofer. Fast Size Approximation of a Radio Network in Beeping Model. *Theoretical Computer Science*, In Press, 2017. doi:10.1016/j.tcs.2017.05.022.
4. Ioannis Caragiannis, Clemente Galdi, and Christos Kaklamanis. Basic Computations in Wireless Networks. In *International Symposium on Algorithms and Computation*, ISAAC '05, pages 533–542. Springer, 2005.
5. Binbin Chen, Ziling Zhou, and Haifeng Yu. Understanding RFID Counting Protocols. In *Proceedings of the 19th Annual International Conference on Mobile Computing and Networking*, MobiCom '13, pages 291–302. ACM, 2013.
6. Israel Cidon and Moshe Sidi. Conflict Multiplicity Estimation and Batch Resolution Algorithms. *IEEE Transactions on Information Theory*, 34(1):101–110, 1988.

7 Alejandro Cornejo and Fabian Kuhn. Deploying Wireless Networks with Beeps. In *International Symposium on Distributed Computing*, DISC '10, pages 148–162. Springer, 2010.

8 Mohsen Ghaffari, Nancy Lynch, and Srikanth Sastry. Leader Election Using Loneliness Detection. *Distributed Computing*, 25(6):427–450, 2012.

9 Seth Gilbert, Valerie King, Seth Pettie, Ely Porat, Jared Saia, and Maxwell Young. (Near) Optimal Resource-competitive Broadcast with Jamming. In *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '14, pages 257–266. ACM, 2014.

10 Seth Gilbert, Fabian Kuhn, and Chaodong Zheng. Communication Primitives in Cognitive Radio Networks. In *Proceedings of the 2017 ACM Symposium on Principles of Distributed Computing*, PODC '17, pages 23–32. ACM, 2017.

11 Albert G. Greenberg, Philippe Flajolet, and Richard E. Ladner. Estimating the Multiplicities of Conflicts to Speed Their Resolution in Multiple Access Channels. *Journal of the ACM*, 34(2):289–325, 1987.

12 Tomasz Jurdziński, Mirosław Kutyłowski, and Jan Zatopiański. Energy-Efficient Size Approximation of Radio Networks with No Collision Detection. In *International Computing and Combinatorics Conference*, COCOON '02, pages 279–289. Springer, 2002.

13 Tomasz Jurdzinski and Grzegorz Stachowiak. Probabilistic Algorithms for the Wake-Up Problem in Single-Hop Radio Networks. *Theory of Computing Systems*, 38(3):347–367, 2005.

14 Jedrzej Kabarowski, Mirosław Kutyłowski, and Wojciech Rutkowski. Adversary Immune Size Approximation of Single-Hop Radio Networks. In *International Conference on Theory and Applications of Models of Computation*, pages 148–158. Springer, 2006.

15 Marek Klonowski and Kamil Wolny. Immune Size Approximation Algorithms in Ad Hoc Radio Network. In *European Conference on Wireless Sensor Networks*, pages 33–48. Springer, 2012.

16 Michael Luby. A Simple Parallel Algorithm for the Maximal Independent Set Problem. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, STOC '85, pages 1–10. ACM, 1985.

17 Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

18 Koji Nakan and Stephan Olari. Uniform Leader Election Protocols for Radio Networks. *IEEE Transactions on Parallel and Distributed Systems*, 13(5):516–526, 2002.

19 Calvin Newport. Radio Network Lower Bounds Made Easy. In *Proceedings of the 28th International Symposium on Distributed Computing*, DISC '14, pages 258–272. Springer, 2014.

20 Calvin Newport and Chaodong Zheng. Approximate Neighbor Counting in Radio Networks, 2018. `arXiv:1811.03278`.

21 Muhammad Shahzad and Alex X. Liu. Every Bit Counts: Fast and Scalable RFID Estimation. In *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, MobiCom '12, pages 365–376. ACM, 2012.

22 Dan E. Willard. Log-logarithmic Selection Resolution Protocols in a Multiple Access Channel. *SIAM Journal on Computing*, 15(2):468–477, 1986.

23 Yuanqing Zheng and Mo Li. ZOE: Fast cardinality estimation for large-scale RFID systems. In *Proceedings of the 32nd IEEE International Conference on Computer Communications*, INFOCOM '13, pages 908–916. IEEE, 2013.

24 Yuanqing Zheng, Mo Li, and Chen Qian. PET: Probabilistic Estimating Tree for Large-Scale RFID Estimation. In *Proceedings of the 31st International Conference on Distributed Computing Systems*, ICDCS '11, pages 37–46. IEEE, 2011.