Understanding RFID Counting Protocols*

Binbin Chen
Advanced Digital Sciences Center
binbin.chen@adsc.com.sg

Ziling Zhou [†]
National University of Singapore
zhouzl@comp.nus.edu.sg

Haifeng Yu
National University of Singapore
haifeng@comp.nus.edu.sg

December 2013

Abstract

Counting the number of RFID tags, or RFID counting*, is needed by a wide array of important wireless applications. Motivated by its paramount practical importance, researchers have developed an impressive arsenal of techniques to improve the performance of RFID counting (i.e., to reduce the time needed to do the counting). This paper aims to gain deeper and fundamental insights in this subject to facilitate future research on this topic.*

As our central thesis, we find out that the overlooked key design aspect for RFID counting protocols to achieve near-optimal performance is a conceptual separation of a protocol into two phases. The first phase uses small overhead to obtain a rough estimate, and the second phase uses the rough estimate to further achieve an accuracy target. Our thesis also indicates that other performance-enhancing techniques or ideas proposed in the literature are only of secondary importance. Guided by our central thesis, we manage to design near-optimal protocols that are more efficient than existing ones and simultaneously simpler than most of them.

1 Introduction

Radio-frequency identification (RFID) technology uses RFID tags and RFID readers (or simply called *tags* and *readers*) to monitor objects in physical world. A tag is a low-cost microchip that can be attached to an object. It can store some information (including a unique ID) and can communicate with a reader through wireless channel. Over the past decade, RFID technology has enjoyed significant growth. With more than 3 billion tags sold in 2012, RFID technology has by now impacted applications ranging from inventory control, supply chain management, to people tracking. A common basic functionality needed by many of these applications is *RFID counting* — to count the number of tags and thus the number of tagged objects in a certain physical area [16]. For example:

- Wal-Mart [2] puts tags on individual clothes. Here RFID counting provides information about sales trend and speeds up the restocking process.
- Purdue Pharma [4] has tagged millions of its tablet bottles. Here RFID counting ensures the right amount of its products are passing through its manufacturing, packaging, and shipping process.
- Many events (e.g., TechEd [1] and Bonnaroo festival [3]) distribute RFID wristbands to their visitors. Here RFID counting helps reveal the number of people around.

Often in such scenarios, it is desirable to simply count or just estimate the number of tags without explicitly identifying individual tags. This helps to significantly reduce the processing time, preserve people's privacy, and avoid the cost incurred for handling a large amount of unnecessary information. In addition to its direct utility, RFID counting can also serve as a preprocessing step and help other tasks. For example, even if one were still to identify individual tags, knowing the rough number of tags can make the identification process much more efficient [11, 19]. As another example, one can use RFID counting to help find popular categories in a large collection of tags [18].

In this paper, we will consider two common versions of RFID counting problem. The first *single-set RFID counting* problem is simply to count the number of tags in a given physical area, using a single stationary reader whose radio range covers that entire area. In the second *multiple-set RFID counting* problem, the reader's radio range cannot cover the whole area. Instead, the (single) reader becomes mobile and sequentially visits a number of locations, so that the union of the coverages at these locations can cover the whole physical area. Note that the coverage at different locations may overlap and hence double counting needs to be avoided.

In both versions of the problem, a key performance metric is the amount of time needed to count or estimate the total number of tags, which will be the focus of this work. Since exact results are often not necessary for many applications (e.g., for the earlier example application scenarios) and since the overhead of exact counting is fundamentally high,[1] as in most prior efforts [8, 11, 12, 15, 17, 23, 24], we will focus on approximate counting.

[†] The conference version of this work appears in MobiCom'13.

[†] This work was partly done when Ziling Zhou was a research intern at Advanced Digital Sciences Center.

[1] As implied by our formal lower bound results in Section 3.

Previous efforts. Given the paramount practical importance of RFID counting, there have been a steady stream of recent research efforts on efficient RFID counting. To reduce the overhead (time) needed to count (i.e., to improve the *performance*), these efforts have developed an impressive arsenal of novel techniques, such as probabilistic framed ALOHA [11], multi-resolution probing [12], lottery frame protocol [15], first non-empty slot based estimation [8], probabilistic estimating tree [23], average run based estimation [17], and zero-one estimator [24].

While these efforts all aim at reducing the overhead of RFID counting, they often approach the problem from rather different perspectives without being guided by a central principle. This has led to ad hoc research outcomes where different researchers view different aspects of RFID counting protocols as key. For example, some researchers focus on using novel statistical quantities to estimate the count [8, 15, 17], some researchers put more emphasis on obtaining optimal trade-offs among different protocol parameters [12, 17], while others resort to gradually refining the parameters via an adaptive iterative process [8, 11].

The fundamentals of the RFID counting problem get easily buried among all these research outcomes — At this point, it is far from clear whether all these techniques are equally important or whether one technique plays the dominant role. Such a lack of deep understanding hinders future research on RFID counting — if we would like to advance the state of the art, should we combine all these techniques despite the resulting complexity? Or should we focus on improving one of them and ignore others?

Our goal. Given such a lack of fundamental understanding into the RFID counting problem, this paper aims to gain deeper insights to facilitate future research. Specifically, we aim to answer the following three key questions, none of which have been posed or answered in prior efforts:

- *Question 1:* Given the long list of protocols in the literature, how much room is there for further improvement?
- *Question 2:* What are the key aspects that determine an RFID counting protocol's performance? What are the techniques that are only of secondary importance?
- *Question 3:* Guided by the answers to the earlier two questions, can we easily design simple protocols that outperform existing ones?

Our results. Our main contributions are precisely the answers to these three questions:

- *Answer 1: Lower bounds.* To determine how much improvement is still possible, we obtain strong lower bounds on the overhead of RFID counting, by leveraging a recent breakthrough result on communication complexity [5]. Our lower bounds show that it is *impossible* for a single-set RFID counting protocol to use only $o(\frac{1}{\epsilon^2 \log \frac{1}{\epsilon}} + \log\log n)$ time slots for all inputs. Here n is the number of tags to count, and ϵ is the relative error on the final output of the protocol (since we are considering approximate counting). In each *time slot*, the reader may broadcast $O(1)$ bits to the tags, and all the tags combined can send back $O(1)$ bits to the reader. A similar lower bound is obtained for multiple-set RFID counting.

  We then compare these lower bounds with the asymptotic overhead of existing protocols. Such comparison readily reveals that:

  - For single-set RFID counting, some existing protocols' performance is already asymptotically close to optimal. Improvements are still possible though one should not expect huge improvements.
  - For multiple-set RFID counting, existing protocols' performance is further away from optimal. Larger improvements hence seem still possible.

- *Answer 2: The overlooked key design aspect for approaching optimal performance.* We identify that a key design aspect for single-set RFID counting protocols to approach optimal performance is to have two conceptual phases: The first phase uses roughly $\Theta(\log\log n)$ slots to obtain a rough estimate with constant (e.g., 0.5) relative error, and the second phase uses roughly $\Theta(\frac{1}{\epsilon^2 \log \frac{1}{\epsilon}})$ slots to eventually obtain a final estimate with the desired relative error of ϵ. Our thesis further indicates that many other performance-enhancing techniques or ideas proposed in the literature are only of secondary importance. We also generalize this answer to multiple-set RFID counting protocols.

  It is worth noting that our answer to this question is quite surprising because prior efforts [8, 11, 12, 15, 17, 23, 24] often view various other aspects of RFID counting protocols as key, and have overlooked this two-phase aspect. Those efforts also attribute their performance improvements to various clever techniques on those aspects (e.g., the use of novel statistical quantities to do the estimation, the use of complex optimization techniques to tune various parameters, and the use of iterative process to refine the estimation). Our answer implies that all those design aspects are perhaps less important than originally thought.

  As direct evidence to support our claim, this paper carefully examines the source of performance gains in some existing RFID counting protocols. For example, some recent protocols [8, 17] attribute their performance improvements over prior protocols to the use of the novel statistical quantities to do the estimation. Quite surprisingly, in our experiments, we find that these novel quantity does not necessarily improve the performance of these protocols: Replacing these novel quantities with some old quantity from some earlier protocol [12] either improves the protocols' performance or provides comparable performance in our experiments. We further show that the source of performance gains in these protocols is their two-phase design, despite that such a two-phase design was not considered as the key.

- *Answer 3: Simple & more efficient RFID counting protocols.* Guided by our answers to the earlier two questions, we set out to search for more efficient RFID counting

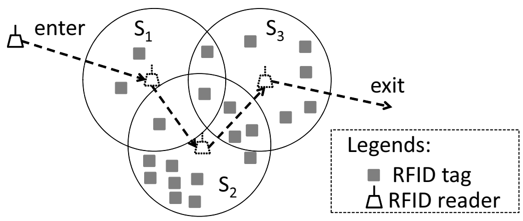

Figure 1: A multiple-set RFID counting example: A mobile reader sequentially visits three locations.

protocols while keeping our design as simple as possible. We manage to design such protocols by simply putting together a few basic building blocks (with some rather minor adaptations) from the literature. We do *not* claim novelty on these building blocks – instead, we aim to show that simply putting them together in a *proper manner* as guided by our earlier answers is already sufficient to outperform existing protocols. This serves as an ultimate validation of the utility of our earlier findings.

Specifically, our RFID counting protocols are significantly simpler than most existing protocols — for example, we do not need iterative refinement or to solve optimization problems to tune parameters. Despite the simplicity, our experiments show that our single-set (multiple-set) RFID counting protocol is around 100% (500%) faster than the best existing single-set (multiple-set) RFID counting protocol. Furthermore, our protocols are *near-optimal* and are within a small $O(\log \frac{1}{\epsilon})$ factor from the lower bounds, for both single-set and multiple-set RFID counting.

Roadmap. The next section formalizes the RFID counting problem. Section 3 proves lower bounds on the overhead of single-set and multiple-set RFID counting. Section 4 reviews major existing RFID counting protocols. Section 5 presents our thesis on the overlooked key design aspect of RFID counting. Section 6 provides direct and immediate evidence to support our thesis by examining the source of performance gain of some recent protocols. Section 7 demonstrates the utility of our insights by applying them to construct new protocols that are both simple and more efficient. Section 8 and Section 9 discuss variant models and related work. We conclude in Section 10.

2 Problem Formulation

This section formalizes the RFID counting problem. We define the overhead of RFID counting protocols mainly for later studying the asymptotic lower bound on the problem and the asymptotic upper bound achieved by the protocols. Hence our formulation here will only be concerned with asymptotic overhead.

Single-set and multiple-set RFID counting. In the *single-set RFID counting* problem, the reader covers a certain physical area. Let S denote the set of tags in that area, and let $n = |S|$. The goal of the counting protocol is to produce an estimate $\hat{n}$ for n, so that $Pr(|\hat{n} - n| \leq \epsilon n) \geq 1 - \delta$, with the probability taken over the random coin flips done by the randomized protocol. Here ϵ and δ captures the target estimation quality, and should be specified by the end user. We also refer to ϵ as the *relative error* of $\hat{n}$. We call $\hat{n}$ as having (ϵ, δ) *estimation quality* and call $\hat{n}$ itself as an (ϵ, δ) estimate.

In the *multiple-set RFID counting* problem (Figure 1), a mobile reader sequentially visits k locations exactly once.[2] At location i, the reader's radio range covers a set S_i of tags. Let $n_i = |S_i|$. The goal of the counting protocol is to produce an (ϵ, δ)-approximation $\hat{n}$ for n, where $n = |S_1 \cup S_2 \cup ... \cup S_k|$. Usually $n \neq n_1 + n_2 + ... + n_k$ since the S_i's may overlap. Note that such formulation of the multiple-set RFID counting problem implicitly but fully captures more general application scenarios. For example, it also captures the setting where a static reader takes a sequence of snapshots of mobile tags and then counts the total number of tags.

Since we aim for (ϵ, δ) estimation quality, the RFID counting protocols are essentially Monte Carlo randomized algorithms [14]. In our reasoning on asymptotic overhead, we will adopt the following standard way of treating δ in Monte Carlo algorithms [14]: We will only require the protocol to achieve constant δ (e.g., 0.2). It is well-known that to achieve a smaller δ, one can repeat the protocol $O(\log \frac{1}{\delta})$ times and then take the median of the $O(\log \frac{1}{\delta})$ outputs as the final output. A constant δ helps simplify our discussion.

Abstracting RFID counting protocols. In an RFID counting protocol, the reader communicates with tags in synchronized time slots. In Section 1, we explained that in each time slot the reader and the tags can exchange $O(1)$ bits. Without loss of generality, from now on, we will assume that in each time slot the reader may send $O(1)$ bits to the tags, while all the tags collectively can either send a single bit of "1" or send nothing.[3] Such treatment is without loss of generality because our formalization here is only for reasoning about asymptotic overhead — one can easily use $O(1)$ slots to send $O(1)$ bits. We say that a tag *responds* in a slot iff it sends back a "1" bit. If there exists at least one tag responding in a slot, the slot becomes *non-empty*. Otherwise the slot is *empty*.

Now consider a given slot. Since the tags are distributed, each tag will need to unilaterally determine whether it will respond, based on its id, random numbers generated locally, and its current state (since the protocol may be stateful), and the bits received from the reader. For our formal reasoning later, it will be convenient to imagine that in each slot the reader conceptually specifies a boolean *predicate function* f. A tag responds in the slot iff it satisfies the predicate f. Note that the RFID counting protocol may be stateful — this is captured by allowing the function f to take the local state (i.e., local variables) of the tag as an input as well.

Measure of goodness. Our measure of goodness (or *performance*) of an RFID counting protocol is the amount of time it

[2] Some researchers (e.g., [17]) consider a *simpler* variant of the problem by assuming *parallel* access to all sets through multiple readers. Section 8 discusses this simpler variant.

[3] Some protocols (e.g., [11]) assume that the reader can further distinguish whether a single tag or multiple tags send a bit. We will cover this extended model in Section 8.

needs. When studying asymptotic behavior, this is the same as the total number of slots used by the protocol. Hence we define the *asymptotic overhead* of a protocol to be $O(x)$, if for all inputs, it needs $O(x)$ slots on expectation for achieving the accuracy target. Here the expectation is taken over the coin flips done by the randomized protocol.

3 Lower Bounds on The Overhead of RFID Counting Protocols

We first consider single-set RFID counting, and then generalize to multiple-set RFID counting.

Single-set Counting: Lower bound as a function of ϵ. We will use a standard *reduction* approach to obtain our novel lower bound on the overhead of an RFID protocol. For readers not familiar with reduction, following is a quick explanation. To prove that a problem $\mathcal{A}$ (in our case, the RFID counting problem) is hard and hence to obtain a lower bound for the complexity of any protocol that solves $\mathcal{A}$, a common approach (called *reduction*) in complexity research is to establish a connection with another hard problem $\mathcal{B}$. Namely, one first shows that any protocol for solving $\mathcal{A}$ can be used, as a black box sub-procedure, to solve $\mathcal{B}$. Next since $\mathcal{B}$ is hard, any protocol for solving $\mathcal{B}$ must incur large overhead. This in turn can be translated back to reason about the hardness of $\mathcal{A}$.

The key step/challenge in reduction is to choose a proper $\mathcal{B}$ and then to show how to construct a protocol for solving $\mathcal{B}$, given any protocol for solving $\mathcal{A}$. We choose the *Hamming Distance Estimation* (HDE) problem as the hard problem $\mathcal{B}$. HDE is a two-party communication complexity problem, where the two parties Alice and Bob are given m-bit strings x and y as input respectively. They would like to estimate the hamming distance between x and y, with (ϵ, δ) estimation quality, while minimizing the number of bits they need to exchange. A recent breakthrough result by Chakrabarti and Regev [5] implies that even for a constant δ, solving the HDE problem requires $\Omega(\frac{1}{\epsilon^2})$ bits of communication between Alice and Bob for $\epsilon \geq 1/\sqrt{m}$.

With HDE as problem $\mathcal{B}$, our goal now is to design a protocol for solving HDE, using any given RFID counting protocol $\mathcal{P}$ as a building block. To do so, Alice and Bob will locally *simulate* an execution of $\mathcal{P}$. Specifically, they will simulate n RFID tags, with IDs from 1 through n. We want tag i to be *present* and be included in the RFID counting result iff $x[i] \neq y[i]$. All other tags j where $x[j] = y[j]$ should be *absent* and will not be included in the count. This will make the RFID count to exactly equal the hamming distance between x and y, hence solving the HDE problem once we know the count.

Now to properly simulate the execution of $\mathcal{P}$ with those present tags, Alice/Bob needs to determine which slots in the simulated execution of $\mathcal{P}$ are empty. Doing so enables Alice/Bob to simulate the responses received in all these slots and feed those into $\mathcal{P}$ to obtain the final count. For each slot, we will show that Alice and Bob can determine whether it is empty by only exchanging $O(\log \frac{1}{\epsilon})$ bits. Consider the first slot. $\mathcal{P}$ must have specified a predicate f for the first slot. Alice/Bob can thus locally determine the set of tags (e.g., tag 2, 6, and 7) that satisfy f. Next Alice computes a short fingerprint of size $O(\log \frac{1}{\epsilon})$ for the (potentially long) string $x[2]x[6]x[7]$ and sends to Bob. Bob similarly computes the fingerprint over $y[2]y[6]y[7]$ and compares the two fingerprints. For now assume no fingerprint collisions (collisions will be properly addressed in our proof). Then the two fingerprints differ iff $x[2] \neq y[2]$ or $x[6] \neq y[6]$ or $x[7] \neq y[7]$, which in turn is equivalent to tag 2 or tag 6 or tag 7 being present, and also equivalent to the first slot being non-empty. Alice and Bob now have successfully determined whether the first slot is empty or not. Emptiness of later slots can be sequentially determined in a similar way.

Formalizing the above intuition will lead to the following theorem, whose proof is in the Appendix A.1:

Theorem 1. *No single-set RFID counting protocol can output an $(\epsilon, 0.2)$ estimate with $o(\frac{1}{\epsilon^2 \log \frac{1}{\epsilon}})$ overhead, for $\epsilon \in [1/\sqrt{n}, 0.5]$.*

Single-set Counting: Lower bound as a function of n. One naturally expects that the number of slots needed by an RFID counting protocol will increase with n as well. For example, to approximate every possible tag count between 1 to n within a relative error of 0.5, a deterministic RFID counting protocol needs to be ready to output $\Omega(\log n)$ different values, with *at least* one in each of ranges $[1, 2]$, $[4, 8]$, $[16, 32]$, ... These $\Omega(\log n)$ different values require at least $\Omega(\log \log n)$ bits (i.e., slots used by the RFID counting protocol) to encode. To extend this argument to randomized RFID counting protocols with (ϵ, δ) guarantee, we leverage Yao's well-known minimax principle [21] on the complexity of randomized algorithms. Doing so will eventually yield a similar $\log \log$ lower bound (see the Appendix A.1 for full proof):

Theorem 2. *No single-set RFID counting protocol can output an $(\epsilon, 0.2)$ estimate with $o(\log \log n)$ overhead, for $\epsilon \leq 0.5$.*

Single-set Counting: Putting everything together.

Corollary 3. *No single-set RFID counting protocol can output an $(\epsilon, 0.2)$ estimate with $o(\frac{1}{\epsilon^2 \log \frac{1}{\epsilon}} + \log \log n)$ overhead, for $\epsilon \in [1/\sqrt{n}, 0.5]$.*

This corollary also implies the difficulty of exact counting: Exact counting is no easier that approximate counting with $\epsilon = \frac{1}{\sqrt{n}}$, where $o(\frac{n}{\log n})$ overhead is already impossible.

Multiple-set Counting: Lower bounds. Recall that in multiple-set RFID counting, the RFID reader sequentially sees a sequence of (potentially overlapping) sets S_1, S_2, ..., S_k. The goal is to estimate the size of the union of all these sets. We can show that in the worst case, to estimate the size of the union, it is actually *necessary* for the protocol to estimate with similar accuracy the size (n_i) of each individual set S_i. Formalize such intuition, together with our single-set RFID counting lower bound, would lead to the following theorem, whose proof is in the Appendix A.2:

Theorem 4. *No multiple-set RFID counting protocol can output an $(\epsilon, 0.2)$ estimate with $o(\sum_{i=1}^{k}(\frac{1}{\epsilon^2 \log \frac{1}{\epsilon}} + \log \log n_i))$ overhead, for $\epsilon \in [1/\sqrt{\min\{n_1, n_2, \ldots, n_k\}}, 0.25]$.*

Key implications of our lower bounds — how much room is there for further improving RFID counting protocols? As we will show in Section 5, in terms of asymptotic overhead, the best existing single-set RFID counting protocol incurs an overhead of $O(\frac{1}{\epsilon^2} + \log\log n)$. This is already close to our lower bound. Improvements may still be possible though one should not expect huge improvements. For multiple-set RFID counting, the best existing protocol incurs an overhead of $O(\frac{k}{\epsilon^2}\log\log(\sum_{i=1}^{k} n_i))$. It exhibits a larger gap from our lower bound — in particular, this overhead is multiplicative while our lower bound is additive. Hence significant asymptotic improvement seems still possible.

4 Review of The Main Ideas in Existing Protocols

This section concisely reviews major RFID counting protocols in the literature (Table 1). This serves to set up the stage for our later discussion on which design aspects are key for RFID counting protocols. For each protocol, we will highlight which design aspects are believed by the original authors as the key aspects of that protocol. Throughout this section, we use $\tilde{n}$ to denote a rough estimate on n (e.g., with constant relative error), and $\hat{n}$ to denote the final estimation on n with ϵ relative error.

These protocols adopt some common concepts. Each of these protocols is comprised of a sequence of *trials*, where each trial is a sequence of slots. At the beginning of a trial, the reader sends a command to the tags. This causes the tags to initialize their local state machines and potentially load new random numbers. Next in each slot within that trial, a tag will respond or not respond based on the command, its local state, and its random number. For all existing protocols, a tag does not carry state across trial boundary. Due to the processing needed at the beginning of a trial, in certain physical implementations of RFID systems, a trial may incur an additional *per-trial overhead*. If there is indeed such overhead, this extra overhead will be in addition to the time needed for all the slots in that trial [6].

The number of slots in a trial is called the *length* of the trial. Recall that a slot is either empty or non-empty, depending on whether there is at least one tag responding in that slot. A non-empty slot is called a *collision* slot iff at least two tags respond in that slot.

One simple way of running a trial, as adopted by multiple protocols, is to start a trial of length l and let each tag *participate* in that trial with a certain probability p, with total np tags participating on expectation. Here we say a tag *participates* in such a trial iff it chooses a uniformly random slot within that trial and then responds in that slot, and we call such a trial a *balls-and-bins trial*. The value of n can then be estimated from various statistical quantities on the status of the slots. A basic principle, which will help us understand these protocols, is that usually we want to use a p value such that np is on the same order as l. This ensures that we see a healthy mixture of empty and non-empty slots in the trials, maximizing the amount of information carried about n. Besides such balls-and-bins trials, existing protocols have also developed alternative ways to use the slots of a trial, as will be described later in the corresponding protocols.

Unified probabilistic estimation (UPE) [11]. In UPE, all trials are balls-and-bins trials with the same length (e.g., 30). In the first trial, all tags participate. Depending on the number of empty slots observed in this trial, the protocol will branch into several different execution paths. We will focus on the most important execution path, which corresponds to large n and where the protocol observes no empty slots in the first trial. In such a case, the protocol proceeds sequentially to the second trial, the third trial, and so on, with each tag participating with $p = 0.1^{i-1}$ probability in the ith trial. This process stops once the protocol sees an empty slot in a trial. The protocol then generates a rough estimate $\tilde{n}$ based on the current p and the number of collision slots in the current trial (i.e., the trial with at least one empty slot), and the first phase ends. In each trial of the second phase, the protocol uses the rough estimate $\tilde{n}$ so far to calculate an optimal p, and has each tag participate with probability p. Next using the new information received in this trial, the protocol amends $\tilde{n}$. This iterative process continues until the protocol believes that the estimation accuracy of $\tilde{n}$ is high enough.

The authors [11] attribute UPE's performance to the proper use of randomization, i.e., carefully choosing the probability for tags to participate in trials (called *probabilistic framed ALOHA* scheme), and the unified use of empty slots and collision slots to do the estimation. The basic idea of randomization has been inherited by virtually all follow-up research on the problem. Despite that UPE does have a rough estimation phase followed by an accurate estimation phase, this two-phase design is not mentioned as a key aspect of UPE by the authors. Multiple later protocols, including the authors' own follow-up work [12], abandon this two-phase approach.

Enhanced zero based estimator (EZB) [12]. EZB partitions the entire domain for the possible values of n into logarithmic number of narrow ranges: $[1, r)$, $[r, r^2)$, $[r^2, r^3)$, ... Here r is some parameter to be explained later. Each of these narrow ranges has the property that the max of the range is at most r times larger than the min. EZB works on each range sequentially and independently. For each range, EZB uses a certain number of balls-and-bins trials with a certain length. In each such trial, tags participate with some probability p. Here the number of trials and trial length are the same for all ranges, while the value of p depends on the range. Finally for each range, EZB uses the number of empty slots in the trials, together with the probability p, to estimate n. EZB then combines all estimates from all ranges to obtain the final output. EZB uses various involved optimization techniques to choose the optimal values for the various parameters such as r and p. Intuitively, EZB works because the count n must be in one of these ranges. Since each range is narrow, one can pick a single p value such that for any value x within that range, xp is on the same order as the length of the trial. This enables n to be properly estimated, as long as n is in that range.

The authors [12] attribute EZB's performance gain to its unique narrow range design (called *multi-resolution probing*) and the various parameter optimization techniques.

Protocol	Venue	Key source of performance gains, as believed by the authors
UPE [11]	MobiCom'06	i) proper randomization; ii) use of empty and collision slots for estimation
EZB [12]	INFOCOM'07	i) multi-resolution probing; ii) various parameter optimization techniques
LOF [15]	PerCom'08 / TPDS'11	small length of the trials
(Enhanced) FNEB [8]	INFOCOM'10	use of the indices of the first non-empty slots for estimation
PET [23]	ICDCS'11 / TMC'12	use of the binary search to find the index of the last nonempty slot
ART [17]	MobiCom'12	use of the average run length of non-empty slots for estimation
ZOE [24]	INFOCOM'13	i) each trial has a single slot; ii) two-phase design

Table 1: Major Existing RFID Counting Protocols

First non-empty slots based estimator (FNEB) and enhanced FNEB [8]. Enhanced FNEB has two phases, while FNEB is exactly the same as the second phase of enhanced FNEB, so we only review enhanced FNEB. A trial in enhanced FNEB is similar to a balls-and-bins trial as it lets each tag uniformly randomly choose an integer from the range of 1 to l'. Here l' is some parameter to be explained later. Different from a balls-and-bins trial, a trial here does not use l' slots to sequentially scan the whole range. Instead, it does so only for the first few slots. If any of them is non-empty (i.e., its index is chosen by some tag), the trial ends immediately and returns the index of that slot. Otherwise, the trial continues with a binary search to find the smallest integer j that has been chosen by at least one tag. Imagine the protocol uses a balls-and-bins trial, the jth slot would be the first non-empty slot it sees. Therefore j is still called the index of the first non-empty slot here.

To start, enhanced FNEB requires the user to input an upper bound on n. The protocol determines the l' used in its first trial by solving an optimization problem parameterized with this upper bound. The protocol then uses the index of the first non-empty slot in its first trial to generate a rough estimate $\tilde{n}$. Intuitively, this index carries information about n since for a given l', the larger the value of n, the smaller this index will likely be. Next the protocol determines the l' used in its second trial by solving the same optimization problem, this time parameterized with the rough estimate $\tilde{n}$. The second trial then proceeds in the same way as the first trial, and amends $\tilde{n}$. This iterative process continues until the protocol believes that the estimation quality of $\tilde{n}$ is good enough. Next the protocol moves on to the second phase where all trials use the same value of l', which is obtained by solving the optimization problem again but parameterized using the $\tilde{n}$ from the first phase. The protocol then combines the first non-empty slot information from all of its second-phase trials to produce a final estimate.

The authors [8] consider their use of the first non-empty slots as the key improvement of (enhanced) FNEB over prior protocols. This design enables (enhanced) FNEB to end a trial as soon as it finds the index of the first non-empty slot. Despite that enhanced FNEB has two phases, these two phases are introduced by the authors only as an "enhancement" instead of a key design aspect.

Lottery frame protocol (LOF) [15]. LOF consists of multiple independent trials. For each trial, a tag randomly chooses a slot according to a geometric distribution where the ith slot is chosen with $\frac{1}{2^i}$ probability. A tag then responds in its chosen slot. LOF finds the index j of the first empty slot by sequentially going through the slots. A trial ends immediately and returns j when the protocol sees the first empty slot. The value of j carries useful information about n: On expectation, $\frac{n}{2^i}$ tags respond in the ith slot, and j tends to take a value around $\log(n)$. Finally, LOF combines the information obtained from all of its trials to produce a final estimate.

The authors [15] attribute LOF's improvement over prior protocols to its small trial length.

Probabilistic estimating tree (PET) [23]. Similar to LOF, PET does a sequence of independent *trials*, where in each trial each tag randomly chooses a positive integer i according to the same geometric distribution as LOF. But instead of determining the j in LOF, PET finds the maximum j' such that there exists some tag choosing j'. The intuition why such j' carries useful information about n is similar to j as in LOF. In addition, PET (implicitly) requires an upper bound x on the maximum j'. These two changes enable PET to perform a more efficient binary search on the slot index range of $[1, x]$ to find the maximum j', instead of sequentially going through the slots. In the first slot of this binary search, PET asks all tags whose chosen integer falls within $[x/2, x]$ to respond. If the slot is empty (non-empty respectively), PET can then focus on the range of $[1, x/2]$ ($[x/2, x]$ respectively) in the next slot.

The authors [23] attribute PET's improvement over prior protocols to the efficient way of using binary search to determine the maximum j'.

Average run based tag estimation (ART) [17]. The first trial in ART is roughly the same as a trial in LOF. ART uses this trial to obtain a rough estimate $\tilde{n}$ on n. The quality of this rough estimate is low since different from LOF which uses many trials to estimate, ART only uses a single trial. All the following trials are balls-and-bins trials, where each tag participates independently with a certain probability p. The length of these trials and the p used in these trials are all the same. ART then observes which slots in each trial are non-empty. Next it calculates the average *run length* of non-empty slots (i.e., the average length of sequences of consecutive non-empty slots), and uses such information to generate a final estimate. Such average run length carries information about n since the larger the value of n, the more non-empty slots, and the larger the average run length. The total number of trials, the length of the trials, and the probability p used in ART are determined by solving an involved optimization problem with the rough estimate $\tilde{n}$ being an input parameter.

The authors [17] attribute ART's improvement over prior protocols to its novel use of run length to do the estimation. While ART does have two phases (with the first phase having a single trial), the authors neither emphasize this aspect nor attribute ART's performance gain to this aspect.

Zero-One Estimator (ZOE) [24]. ZOE is independent of and concurrent with our work. ZOE has two explicit phases, where the first phase gets a rough estimate $\tilde{n}$ and the second one obtains the final estimate. As a key design decision, each trial in ZOE has a single slot, so we directly describe slots here. In its first phase, ZOE aims to find a j such that if all tags participate in a slot with a probability of $1/2^j$, the probability of the slot being empty is around $1/e$. To find such a j efficiently, ZOE (implicitly) requires an upper bound x on the number of tags so that it can does a binary search over $[0, \log x]$. Each step of the binary search uses a constant number of slots. In each such slot, the tags respond with probability of $1/2^i$ where i is the current value tested in the binary search. The protocol then observes the fraction of empty slots, and determines how to continue the binary search. With a suitable j found by the first phase, ZOE's second phase uses a certain number of slots where the tags participate in each slot with probability of $1/2^j$. The number of slots needed in the second phase is determined by the required estimation quality. ZOE eventually estimates n from the fraction of empty slots observed in the second phase.

The authors [24] attribute ZOE's improvement over prior protocols to the following two design aspects: i) each trial having only a single slot so that this slot can potentially collect information from all tags, and ii) having two explicit phases. While this concurrent work of ZOE does emphasize the importance of its two-phase design, the thesis identified in this paper is still not discovered in ZOE: ZOE believes that its unique design of each trial having a single slot is also key to ZOE's performance. Our thesis, on the other hand, suggests that the two-phase design is the key while other aspects are only secondary. Guided by our thesis, a protocol designer would not be overly concerned with sticking to ZOE's idea of having a single slot in each trial. Section 7.3 will show that *not* having a single slot in each trial, as in our protocol, enables us to get better performance in our experiments.

5 Which Design Aspects Are Key?

So far we have reviewed major RFID counting protocols in the literature, each with its own unique techniques. Given such a myriad of interesting techniques, which techniques are the actual dominant factors for good performance? Which techniques are less important? If one would like to outperform the state-of-the-art, which existing technique should one builds upon? To answer these questions, we aim to identify the key aspects of efficient RFID counting protocols.

While experimental study can help reveal about which aspects in these protocols are more important than others, we notice that what we are looking for could very well be buried deep under the vast amount of experimental data. Thus we start by first systematically investigating and comparing the asymptotic overhead of these protocols, with respect to the n and ϵ. Interestingly, as we will soon see, such a simple investigation already sheds much light onto the question.

UPE [11]	–
EZB [12]	$O(\frac{1}{\epsilon^2} \log n)$
LOF [15]	$O(\frac{1}{\epsilon^2} \log n)$
FNEB [8]	$O(\frac{1}{\epsilon^2} \log n)$
Enhanced FNEB [8]	$O(\frac{1}{\epsilon^2} + \log n)$
PET [23]	$O(\frac{1}{\epsilon^2} \log \log n)$
ART [17]	$O(\frac{1}{\epsilon^2} + \log n)$
ZOE [24]	$O(\frac{1}{\epsilon^2} + \log \log n)$

Table 2: Asymptotic Overhead of Single-Set Protocols

It is worth noting that such a systematic comparison of the asymptotic behavior has never been done before: The end-to-end performance of some protocols [11, 17] has not been formally analyzed, while the performance of other protocols [8, 12, 15, 23] has been analyzed and presented in a rather detailed form. These more precise but complex forms unfortunately prevent a direct comparison across the protocols and bury the key insights we are searching for.

Asymptotic overhead of single-set RFID counting protocols.
UPE [11] and ART [17] do not come with end-to-end overhead analysis. We find that the estimator used by UPE is biased, hence UPE cannot be used when ϵ is small. This is consistent with the findings by the original authors of UPE in their follow-up work [12] and will be validated by our experiments in Section 7.3. We have analyzed ART by ourselves, which shows that it uses $O(\log n)$ slots in the first phase and $O(\frac{1}{\epsilon^2})$ slots in the second phase. This implies a total overhead of $O(\frac{1}{\epsilon^2} + \log n)$. For space limitation, we leave the full analysis, which is straightforward and uses rather standard approaches, to the Appendix B.

The other existing protocols, i.e., EZB [12], (enhanced) FNEB [8], LOF [15], PET [23], and ZOE [24], all come with detailed analysis on the number of slots needed. Here all we do is to simplify their more precise results to asymptotic forms (with adaption to our formulation when necessary), for later comparison. More details about these protocols can be found in the Appendix B.

Table 2 summarizes the asymptotic overhead of these single-set RFID counting protocols. At this point, it is clear that the protocols have either additive overhead or multiplicative overhead. Additive overhead is obviously lower, and it comes from a conceptual separation of two phases in these protocols, with the first phase taking $O(\log n)$ or $O(\log \log n)$ slots and the second phase taking $O(\frac{1}{\epsilon^2})$ slots. The $\log n$ and $\log \log n$ term are about 16 and 4 respectively, for $n = 100,000$. (When n is small, almost all known protocols can complete fast, so further improvement is less interesting.) Unless the hidden constant in a multiplicative overhead protocol is comparably smaller, additive overhead protocol will be more efficient. Our experiments in Section 7 will show that this is indeed the case.

Bring our lower bound from Section 3 into the picture makes this key observation even clearer. There we proved that it is impossible to reduce the overhead of a single-set RFID counting protocol to $o(\frac{1}{\epsilon^2 \log \frac{1}{\epsilon}} + \log\log n)$. Now it is clear that $\Theta(\log\log n)$ slots are for the first phase, while the remaining $\Theta(\frac{1}{\epsilon^2 \log \frac{1}{\epsilon}})$ slots are for the second phase.

Our thesis. Our observations above lead us to conjecture the following thesis, which will be validated in the remainder of this paper:

> *The key design aspect for single-set RFID counting protocols to achieve near-optimal performance is to have two phases, where the first phase uses roughly $\Theta(\log\log n)$ slots to obtain a rough estimate with constant (e.g., 0.5) relative error, and the second phase uses roughly $\Theta(\frac{1}{\epsilon^2 \log \frac{1}{\epsilon}})$ slots to eventually obtain a final estimate with the desired relative error of ϵ. Furthermore, other techniques/ideas proposed in the literature are only of secondary importance.*

While this thesis is almost obvious from our discussion so far, somewhat surprisingly, it has never been identified by any of the previous efforts (including the concurrent work on ZOE [24]). Instead, existing protocols often overlook the two-phase design and often attribute their improvements to a diverse set of design aspects other that the two-phase design. Our thesis implies that all the following design aspects, emphasized by previous efforts, are far less important than originally thought:

- using various novel statistical quantities to do the estimation (such as using the average run length in ART [17] and using the index of the first non-empty slot in FNEB [8]);
- using an iterative process to refine the estimation over many iterations (such as in UPE [11] and enhanced FNEB [8]);
- using complex optimization techniques to tune various parameters (e.g., to trade off the trial length with the number of trials as in EZB [12], FNEB [8], and ART [17]);
- using a single slot in each trial as in ZOE [24].

Generalizing to multiple-set RFID counting protocols. We naturally generalize our thesis to the multiple-set setting: There the protocol should have two phases at each location i for $1 \leq i \leq k$, where the first phase uses roughly $\Theta(\log\log n_i)$ slots to obtain a rough estimate, and the second phase uses roughly $\Theta(\frac{1}{\epsilon^2 \log \frac{1}{\epsilon}})$ slots.

Existing multiple-set RFID counting protocols[4] (EZB, FNEB[5], LOF[6], and PET) all focus on other aspects of the protocol instead of having the above two phases, and incur multiplicative overhead. Specifically, EZB, LOF, and FNEB all incur $O(\frac{k}{\epsilon^2}\log(\sum_{i=1}^k n_i))$ overhead, while PET incurs $O(\frac{k}{\epsilon^2}\log\log(\sum_{i=1}^k n_i))$ overhead.
Such multiplicative overhead contrasts sharply with the additive overhead of our new SRC$_M$ protocol (Section 7.2), which has applied our thesis on the two-phase design. Hence in the multiple-set setting, these previous efforts have not even implicitly applied our thesis.

6 Source of Performance Gain — Two Case Studies

An ultimate way of validating our thesis is to see whether applying such a design principle enables new protocols that are significantly better than existing ones. We will do so later in Section 7. This section instead aims to provide direct and immediate evidence to support our thesis, by carefully examining the source of performance gains in existing protocols. We will focus on two recent protocols, ART [17] and enhanced FNEB [8], as two prominent examples. As reviewed in Section 4, ART uses the average run length of non-empty slots as a *gauge* for estimation and attributes its performance gain over prior protocols to this unique gauge. Similarly, the authors of enhanced FNEB [8] consider the novel use of the first non-empty slots as a *gauge* being the key source of performance gain.

We will show that quite surprisingly, in our experiments, these two novel gauges do not necessarily improve the performance of ART and enhanced FNEB: Replacing these two novel gauges with a simple gauge (i.e., the number of empty slots in balls-and-bins trials) from the earlier EZB protocol [12] either improves the performance or provides comparable performance in our experiments. We further show that the actual source of performance gains in these two protocols is their (implicit) two-phase design, despite that such a two-phase design was not considered as the key.

6.1 Source of Performance Gain in ART

For all experimental results presented in this section, we use $n = 100,000$ and a constant $\delta = 0.2$ unless otherwise mentioned — we have performed extensive experiments under other settings (e.g., with smaller n) and observe similar trends (see Appendix F). Our evaluation in this subsection adopts the same setting as the original ART paper [17]. Specifically, we assume that each slot takes 0.3ms, and each trial incurs an additional overhead of 1ms.

ART outperforms EZB. To identify the source of performance gain in ART [17], for clarity, we focus on ART's performance gain when compared with a specific prior protocol EZB [12]. As a sanity check, we first perform experiments to see whether ART indeed outperforms EZB, as claimed in [17]. Figure 2 summarizes our experimental results, showing the amount of time needed for ART and EZB to achieve a certain target relative error ϵ. Consistent with [17], we observe that ART significantly outperforms EZB — more than 200% faster.

ART's novel gauge and ART's performance. Next we proceed to test whether this performance gain comes from ART's novel run length based gauge. To do so, we keep everything else unmodified in ART except that we replace ART's novel

[4] Other protocols are not for the multiple-set RFID counting problem. Among those, ART only works for a simpler variant of the multiple-set problem (see Section 8).

[5] Enhanced FNEB no longer works in the multiple-set problem.

[6] Here LOF requires an upper bound x on the number of tags, and can no longer end a trial when it sees the first empty slot.

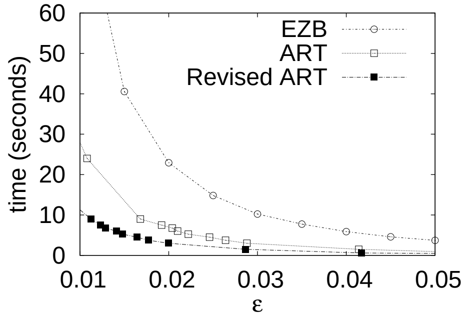

Figure 2: Time needed to achieve relative error ϵ under $\delta = 0.2$.

	ART	Revised ART
With 2000 time slots:		
Var($gauge$)	0.080	4.8
Var($\hat{n}$)	12 $\times 10^6$	8.0 $\times 10^6$
With 4000 time slots:		
Var($gauge$)	0.045	2.4
Var($\hat{n}$)	7.0 $\times 10^6$	3.9 $\times 10^6$

Table 3: Variances of gauges and estimates under ART and revised ART.

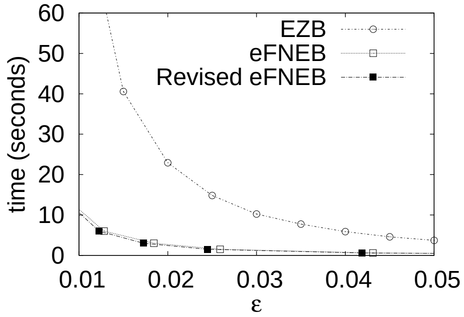

Figure 3: Time needed to achieve relative error ϵ under $\delta = 0.2$.

run length based gauge with the old gauge in EZB. This old gauge in EZB is based on the number of empty slots. We call this protocol as the revised ART. If the run length based gauge were indeed the source of ART's performance gain, the revised ART should perform significantly worse than ART. Quite surprisingly, as shown in Figure 2, the revised ART actually outperforms the original ART.

Resolving the contradiction. To resolve such contradiction with the claims from [17] that ART's novel gauge is the source of performance gain, we trace back and examine the reasoning in that work. There the authors [17] compare the variance of ART's average run length based gauge with the variance of other old gauges, including the gauge in EZB (and hence the gauge in revised ART). They show that the variance of ART's gauge is smaller, leading to the conclusion that ART's gauge is the source of performance gain. Again as a sanity check, we examine the variance of ART's gauge and EZB's gauge as observed in our experiments. Consistent with [17], we also observe that ART's gauge has smaller variance (Table 3). On the other hand, however, we find that smaller variance of a gauge does not necessarily translate to better accuracy of the final estimate. Table 3 also presents the variance of the final estimate as generated by ART and revised ART (which uses EZB's gauge). Despite ART's gauge has smaller variance than EZB's, the variance of ART's final estimate is actually larger than that of revised ART's final estimate. Note that this is consistent with the better performance of revised ART as we observed in Figure 2.

The fundamental reason behind these results is that in order for the final estimate to have better accuracy, the gauge needs to not only have small variance but also be *sensitive* to the count. In other words, under different number (n) of tags, the value of the gauge should ideally be very different. This ensures that we can easily differentiate different n even if the value of the gauge is a bit off from its expectation. In fact, if we were not concerned with such sensitivity, it would be trivial to design a gauge with zero variance: We simply let the value of the gauge always be a constant regardless of what n is. Clearly such gauge cannot be used to accurately estimate n. Hence the reason that the variance of ART's final estimate is larger is exactly that ART's gauge is less sensitive than EZB's. Intuitively, such insensitivity can even be partly observed from the fact that under practical parameters, the value of ART's gauge has a smaller domain that EZB's.

The actual source of performance gain. It will shed much light onto the problem if we view the revised ART protocol from a different perspective. Namely, one can alternatively view the revised ART protocol as a variant of the EZB protocol — the only main difference between these two is that EZB does not have a rough estimate from a first phase. Thus EZB has to divide the possible domain for n into $O(\log n)$ narrow ranges and process them sequentially. In comparison, revised ART has a rough estimate from its first phase to identify the correct range to process.

Hence one can view revised ART as adding a first phase to EZB. This implies that the performance gain of revised ART over EZB comes from having two phases as suggested by our thesis. In turn, this is also the source of the performance gain in ART.

6.2 Source of Performance Gain in Enhanced FNEB

Using the same approach as above, we continue to examine the source of the performance gain of enhanced FNEB [8] over EZB. Here our evaluation adopts the same setting as [8], where each slot still takes 0.3ms (as in [17]) but there is no per-trial overhead. As shown in Figure 3, our experiments first confirm that enhanced FNEB significantly outperforms EZB. To test whether this performance gain comes from FNEB's unique first non-empty slot gauge, we revise the enhanced FNEB by using EZB's gauge in its second phase while keeping all other design in enhanced FNEB unchanged. Our revised version of enhanced FNEB provides comparable performance as the enhanced FNEB (specifically, our revised protocol outperforms enhanced FNEB slightly by around 6%), showing that FNEB's novel gauge does not necessarily improve its performance.

The original authors of (enhanced) FNEB [8] attribute the performance gain to their novel gauge, because they believe that such gauge enables (enhanced) FNEB to end a trial as soon as it sees the first non-empty slot and thus reduces the number of slots per trial. While this is obviously true, the total overhead of a protocol also depends on the number of trials needed. For example, when $\delta = 0.2$, to achieve $\epsilon = 0.01$, enhanced FNEB uses on average around 2 slots per trial but it needs to invoke around $16,000$ trials. To achieve the same estimation quality with EZB's gauge, each trial uses 242 slots and only around 120 trials are needed. Hence the total number of slots needed by EZB's gauge is comparable to that needed by FNEB's gauge.

Exactly as in the case of revised ART, here one can alternatively view the revised version of enhanced FNEB as adding a first phase to EZB. This directly leads to our conclusion that the actual source of the performance gain in enhanced FNEB is having two phases as suggested in our thesis.

7 Designing Better RFID Counting Protocols

Guided by our thesis in Section 5, this section aims to design new RFID counting protocols that are more efficient than existing ones and also simultaneously simpler than most of them. We will design our protocols by simply putting together various basic building blocks in the literature. We do *not* claim novelty on these building blocks – instead, we aim to show that simply putting them together in a *proper manner* as guided by our thesis is already sufficient to outperform existing protocols. This serves as an ultimate validation of the utility of our thesis.

7.1 SRC$_S$: Our Simple RFID Counting Protocol for Single-Set

For single-set RFID counting, our thesis suggests that the protocol should have two conceptual phases, the first one does a rough estimation, while the second one generates the final estimate. When designing these two phases, we will use as simple building blocks as possible. This is because: i) more complex designs tend to have larger hidden constants, and ii) our thesis indicates that other performance tricks only have minor effects in further improving performance.

Our SRC$_S$ protocol. Algorithm 1 summarizes the main steps of our SRC$_S$ protocol. The first phase of our SRC$_S$ protocol is exactly the same as the simple LOF protocol [15] as reviewed in Section 4. Recall that LOF does a sequence of independent trials with each trial using $O(\log n)$ slots. For $\delta = 0.2$, our protocol invokes LOF to do 10 trials, using total $O(\log n)$ slots. It then uses LOF's output as the rough estimate $\tilde{n}$. By LOF's analysis [15], $\tilde{n}$'s relative error is below 0.5 with at least $\frac{9}{10}$ probability. Given such a $\tilde{n}$, the second phase of SRC$_S$ (as we will soon describe) guarantees to output an estimate $\hat{n}$ of relative error below ϵ with probability of $\frac{8}{9}$. Combining the guarantees from these two phases ensures that $\hat{n}$'s relative error is below ϵ with probability of $\frac{9}{10} \times \frac{8}{9}$, which corresponds to $\delta = 0.2$. To achieve a δ smaller than 0.2, one can sequentially invoke m (m being some odd integer) independent instances of Algorithm 1 and then take the median of their outputs as the final output. Asymptotically, it is well-known that $m = O(\log \frac{1}{\delta})$ suffices [14]. Obtaining a concrete value of m for a certain target δ is not hard: Each instance of Algorithm 1 has $1 - 0.2 = 0.8$ probability to generate a "good" result with at most ϵ relative error. For the median to have at most ϵ relative error, it suffices to have at least $(m + 1)/2$ good results among the m results. With all instances being independent, we simply pick the smallest m such that $\sum_{i=(m+1)/2}^{m} \binom{m}{i} \times 0.8^i \times 0.2^{m-i} \geq 1 - \delta$. Since m is usually small (e.g., m only needs to be 41 even for $\delta = 10^{-5}$), the value of m can be trivially determined via brute-force calculation.

Algorithm 1 Our SRC$_S$ protocol (for $\delta = 0.2$)

```
Invoke LOF with 10 trials to get $\tilde{n}$;
Start a balls-and-bins trial of length $l$, and let each tag *participate in the trial* with probability $p = \min\{1, 1.6l/\tilde{n}\}$;
Count the number of empty slots $z$ in the trial;
Output $\ln(z/l)/\ln(1 - p/l)$.
```

The second phase of SRC$_S$ simply consists of a single trial with l slots, and each tag participates in this trial (i.e., responds in a uniformly random slot in the trial) independently with probability p. We will explain the two parameters l and p later. The expected fraction of empty slots in this trial will thus be $(1 - p/l)^n$. Our protocol determines the observed number of empty slots in this trial, denoted by z. Obviously, z directly carries information about n. The protocol finally generates the final estimate $\hat{n}$ by solving the equation $(1 - p/l)^{\hat{n}} = z/l$, which leads to $\hat{n} = \ln(z/l)/\ln(1 - p/l)$. The second phase of our protocol is rather similar to subprocedures used in UPE [11] and EZB [12]. The only (minor) difference is that we further simplify the design and use a single trial instead of doing multiple trials. This simplification actually also slightly improves our performance: By putting all slots into the same trial, whether a slot is empty becomes negatively correlated with each other. Such negative correlation makes the total number of empty slots concentrate better near its expected value.

The parameter l is uniquely determined by the target relative error of ϵ, and there are two ways to do so. The first approach is to set $l = \frac{65}{(1-0.04^{\epsilon})^2}$, which is $O(\frac{1}{\epsilon^2})$ (see the proof of Theorem 5 in the Appendix D.3, where we have proved that such l is sufficiently large). The second approach is to directly construct a numerical lookup table. This lookup table is constructed by running the algorithm under a wide range of n values, and then observing the l needed to achieve a certain ϵ. See Appendix C for a sample table. Between the two approaches, since mathematical analysis is often a loose approximation, in practice, using a lookup table usually offers superior performance. The parameter p is set to be $\min\{1, 1.6l/\tilde{n}\}$, so that the expected number of tags responding is on the same order as l. The constant 1.6 here provides the best estimation performance (see analysis in [11, 12]).

The following theorem summarizes the end-to-end guarantee of our SRC$_S$ protocol, whose proof is in the Appendix D.2:

Theorem 5. *Our SRC$_S$ protocol outputs an $(\epsilon, 0.2)$ estimate with $O(\frac{1}{\epsilon^2} + \log n)$ overhead.*

Incurring $O(\log\log n)$ slots in the first phase. The first phase of the design above incurs $O(\log n)$ slots. It is possible to use only $O(\log\log n)$ slots by using a revised version of PET protocol [23] instead. As reviewed in Section 4, PET does a sequence of independent trials. In each trial, each tag randomly chooses a positive integer according to a geometric distribution. Given a proper upper bound x (from the end user) on n, PET uses a binary search over $[1, \log x]$ to find the maximum j' such that there exists some tag choosing j'. Hence the number of slots incurred in PET for each trial is $O(\log \log x)$. It is possible for x to be much larger than n, in which case this will still not give us $O(\log \log n)$ complexity. To always have $O(\log \log n)$ complexity, we slightly modify PET so that the user does not input x: In each trial before the binary search, the protocol uses some extra slots. In the ith extra slot, tags that have chosen an integer larger than or equal to 2^{i-1} will respond. This process stops once the protocol observes an empty slot. Let the corresponding i in this empty slot be y. Next the protocol does a binary search as before, except that now the binary search is done over $[1, 2^{y-1}]$ instead of $[1, \log x]$. This binary search will take another y slots at most. It can be easily shown that $y = O(\log \log n)$ on expectation. Hence the total overhead will be $O(\log \log n)$ slots. See Appendix C for more details and the pseudo-code.

Under practical settings, however, the overhead for the second phase usually dominates and such improvement will be negligible. But we will need this revised PET later in our multi-set protocol.

7.2 SRC$_M$: Our Simple RFID Counting Protocol for Multiple-Set

For multiple-set counting, our thesis suggests that the protocol should have two conceptual phases at each location i for $1 \leq i \leq k$. We will focus on achieving the two phases in a simple way.

Protocol intuition. Recall that SRC$_S$ conceptually works by throwing np (on expectation) balls uniformly randomly into l bins. The value of n can then be inferred from the fraction of empty bins. We would like to design SRC$_M$ in a similarly simple way, i.e., by throwing np balls (on expectation) into l bins, where n is the total number of tags in all sets (if there is no overlapping between sets, $n = n_1 + n_2 + \ldots + n_k$). The value of l can still be determined by ϵ and our Theorem 6 later shows that $l = O(1/\epsilon^2)$. Imagine for now that magically, we can also properly set p to be $\min\{1, 1.6l/\tilde{n}\}$, where $\tilde{n}$ is a rough estimate for n with constant relative error. With such value for p, the problem becomes trivial: At each location, the protocol simply does a balls-and-bins trial with participation probability of p, so that on expectation there are np balls in total. The protocol records the outcome at each location and merges these results for producing a final estimate. The merging is done by considering a bin occupied as long as it is occupied in any of the k locations. Note that this already takes care of potential overlaps between the k sets – as long as we use the same random seed when doing these experiments, the same tag will always be hashed into the same bin, even if it appears in multiple sets.

So far we have assumed that the protocol can properly set p. However in the multiple-set setting, the protocol sees S_1, S_2, ..., S_k sequentially and it is not possible to obtain $\tilde{n}$ until the last location. Observe however that at location i, the protocol can easily get a rough estimate $\tilde{n}'_i$ for the size of $S_1 \cup S_2 \ldots \cup S_i$ (by merging all the first phase results up to location i). Define $p_i = \min\{1, 1.6l/\tilde{n}'_i\}$ and we obviously have $p_i \geq p$ (note that $p_k = p$). Next note that these values of p and p_i do not need to be accurate, since the rough estimate is rough in the first place. Hence let us assume, without loss of generality, that they are both in the form of $1/2^x$ for some integer x. If not, we simply round them to the nearest value with such a form. When the reader finishes the first phase for the ith set, it knows p_i but not p. Conceptually for set S_i, the protocol will do the balls-and-bins trial with participation probabilities p_i, $\frac{p_i}{2}$, $\frac{p_i}{4}$, $\frac{p_i}{8}$, ..., and so on. This ensures that one of the participation probability will equal p, regardless of what p is. After processing all sets, we can then decide the proper value of p and use the combined result for the corresponding trial to obtain the final estimate.

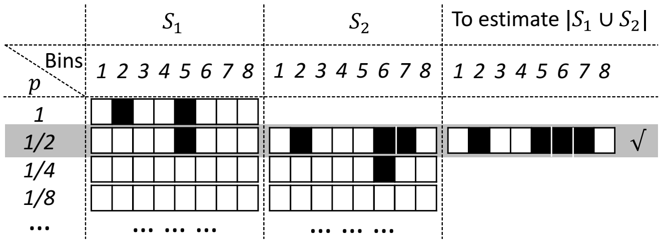

Figure 4: An example run of SRS$_M$ with two reader locations: Each column corresponds to a bin (note that the same bins appear at both locations), and each row corresponds to a participation probability. A filled (non-filled) rectangle means an occupied (non-occupied) bin. At a given location, once a bin becomes non-occupied at a certain participation probability, there is no need to further examine smaller probabilities for this bin. In this example, the second phase of SRS$_M$ starts at the participation probability of 1 and $\frac{1}{2}$ respectively at the first and the second location. SRC$_M$ eventually merges the outcomes at the participation probability of $\frac{1}{2}$ from the two locations to estimate $|S_1 \cup S_2|$.

Naively doing the above trials with the infinite sequence of participation probabilities will result in infinite overhead. One can easily make things correlated to avoid this: For each participation probability except the first one, a tag flips a fair coin and participate iff the coin flip result is head and the tag participated in the previous participation probability. This would mean that in this sequence, a tag will keep participation, and then stop participating after a certain probability. In turn, this means that for a given bin in this sequence of experiments, it will initially be occupied and then will never be occupied again after a certain participation probability (Figure 4). This enables the protocol to do the following: Instead of checking all bins for a given probability, the protocol iterates through the bins. For each bin, the protocol checks whether it is occupied, for all the probabilities in the sequence. Note that the protocol can stop once the bin becomes empty. The rough estimate from SRC$_M$'s first phase ensures that its second phase sees a constant number of balls in each bin on expectation. From the mean of geometric distributions, one can easily see that on average, it only needs to move down the sequence of participation probabilities $O(1)$ steps for a given bin to become empty. Hence the total number of slots needed is just $O(1) \cdot l = O(\frac{1}{\epsilon^2})$.[7]

Algorithm 2 Our SRC$_M$ protocol (for $\delta = 0.2$)

```
Each tag uniformly randomly chooses a bin out of $l$ bins, and chooses a positive integer $y$ according to a geometric distribution with mean of 2;
Initialize $A$ to an array of $l$ elements with values of $-1$. $A[j]$ will record the largest $y$ chosen by a tag in the $j$th bin;

for each set $S_i$ do
    Invoke revised PET with 30 trials, and merge its outcome with previous revised PET outcomes to get a rough estimate $\tilde{n}'_i$ for the size of $S_1 \cup S_2 \ldots \cup S_i$;
    Find an integer $x$ that minimizes $|1/2^x - \min\{1, 1.6l/\tilde{n}'_i\}|$;
    for $j = 1$ to $l$ do
        $h = x$;
        while true do
            Let all tags in the $j$th bin with $y \geq h$ respond;
            if (See a non-empty slot) then
                $A[j] = \max\{A[j], h\}$;
                $h = h + 1$;
            else
                Break;
            end if
        end while
    end for
end for

Consider the $x$ used for the last set and let $z$ be the number of elements in $A$ with value no less than $x$;
Output $\ln(z/l)/\ln(1 - 2^{-x}/l)$.
```

Our SRC$_M$ protocol. Our SRC$_M$ protocol implements the above intuitions. Algorithm 2 summarizes the main steps of our SRC$_M$ protocol (for $\delta = 0.2$). To achieve a δ smaller than 0.2, exactly as for SRC$_S$, one only needs to sequentially invoke multiple independent instances of Algorithm 2 and then take the median result. See Section 7.1 for how to determine the number of instances needed. For the parameter l, the only difference between SRC$_M$ and SRC$_S$ is that the participation probability used in SRC$_M$ needs to be rounded to the form of $1/2^x$. Taking this into account, we can either mathematically set $l = \frac{205}{(1-0.013^{\epsilon})^2}$, which is $O(\frac{1}{\epsilon^2})$ (see the proof of our Theorem 6 in the Appendix D.3), or find its value from a numerical lookup table. The lookup table is constructed by running the algorithm under a wide range of n values and then observing the l needed to achieve a certain ϵ. Note that l does not depend on the number of sets and how the sets overlap (see more detailed reasoning in the Appendix D.3), one only need to run the algorithm against a single set.

Given l, each tag determines which bin it will choose, and also the smallest participation probability for which it will still participate. At location i, our SRC$_M$ protocol has two phases. For a constant $\delta = 0.2$, the first phase invokes the revised PET protocol (with 30 trials), which was described at the end of Section 7.1. This incurs total $O(\log \log n_i)$ slots. SRC$_M$ then merges all the first phase results it sees so far to get a rough estimate $\tilde{n}'_i$ for the size of $S_1 \cup S_2 \ldots \cup S_i$. Such merging is possible since PET, and therefore the revised PET, is able to do multiple-set RFID counting. By PET's analysis [23], the relative error of $\tilde{n}'_i$ is below 0.5 with at least $\frac{9}{10}$ probability. The second phase now determines p_i based on $\tilde{n}'_i$ in exactly the same way as in our SRC$_S$ protocol. We then round p_i to the nearest $1/2^x$ for some integer x. The protocol then iterates through the l bins. For each bin, the protocol uses a sequence of slots, which corresponds to participation probabilities p_i, $\frac{p_i}{2}$, $\frac{p_i}{4}$, ... For each slot, those tags who select this bin and still participate at the current participation probability will respond. The protocol records all such information and stops once an empty slot is observed. It then proceeds to the next bin.

At the last (kth) location, SRC$_M$ can merge the first phase results from all the k sets to obtain a rough estimate for the size of the union of all k sets, and it can compute a proper participation probability p based on this rough estimate. By our design, SRC$_M$ must have collected the information regarding whether each bin is empty under p for every location. SRC$_M$ then combines such information by setting a bin to be empty iff it is empty in all sets (see Figure 4). Let z denote the number of empty bins in the combined l bins, SRC$_M$ generates the final estimate $\hat{n}$ by solving the equation $(1 - p/l)^{\hat{n}} = z/l$. See the Appendix D.3 for the proof for the following theorem about the end-to-end guarantee of our SRC$_M$ protocol:

Theorem 6. *Our SRC$_M$ protocol outputs an $(\epsilon, 0.2)$ estimate with $O(\sum_{i=1}^{k}(\frac{1}{\epsilon^2} + \log \log n_i))$ overhead.*

7.3 Evaluation Results

We conduct extensive simulations to compare the overhead of our protocols against all major existing protocols in the literature, including UPE, EZB, (enhanced) FNEB, LOF, PET, ART, and ZOE. As in Section 6, we consider a constant $\delta = 0.2$ to simplify our discussion — we observe similar trends under all other values of δ. When comparing the performance of these protocols, for each experiment, we first choose a time budget, then we simulate the protocols and observe their achieved relative error ϵ given such budget (i.e., overhead). This evaluation methodology is also taken by recent prior work [24]. An alternative evaluation methodology would be to compare the overhead of different protocols when achieving the same target ϵ. We do not take this method since for several two-phase protocols (e.g., [17,24]), when they mathematically decide the number of slots needed in their second phase, they assume a perfect estimate from their first phase. Since the estimate from the first phase is only a rough estimate, following their calculation will actually achieve a relative error that is somewhat larger than the target ϵ. This will make the comparison inconsistent across the protocols. Unless otherwise

[7] A less efficient design would be to iterate through the sequence of participation probabilities. For each probability, one checks all bins. The process stops if all bins are empty. Such a design would need on expectation $O(\frac{\log(1/\epsilon)}{\epsilon^2})$ slots.

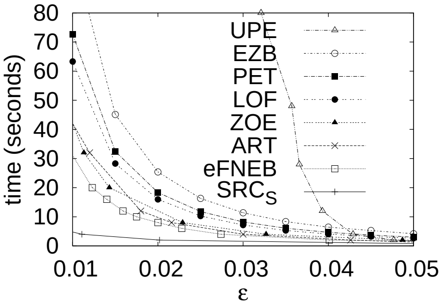

Figure 5: Overhead of single-set protocols ($n = 10,000$).

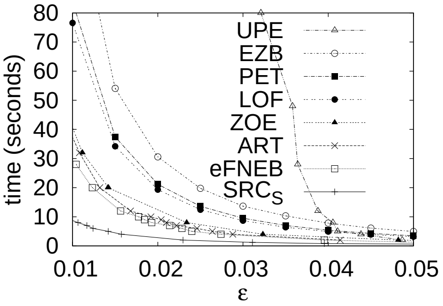

Figure 6: Overhead of single-set protocols ($n = 100,000$).

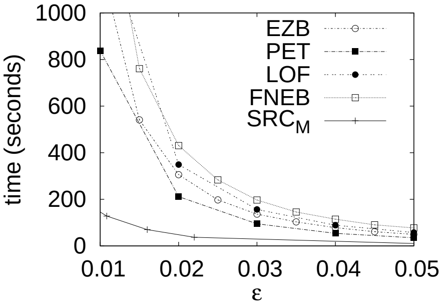

Figure 7: Overhead of multiple-set protocols ($n = 100,000$ and $k = 10$).

mentioned, all our experiments use the following parameters derived from EPCglobal C1G2 standard [6]: a slot in UPE takes 0.8ms[8], a slot in all other protocols takes 0.4ms, and for all protocols each trial incurs an extra overhead of 1ms.

Comparing SRC$_S$ with existing single-set protocols. Figure 5 and Figure 6 present the overhead of our SRC$_S$ protocol against the overhead of existing single-set RFID counting protocols, for tag count of $10,000$ and $100,000$ respectively. As shown in the figures, SRC$_S$ is significantly (more than 1000%) faster than EZB, PET, and LOF. This is because asymptotically SRC$_S$ incurs additive overhead while EZB, PET, and LOF all incur multiplicative overhead (see Section 5). SRC$_S$ is at least 100% faster than ART, ZOE, and enhanced FNEB (eFNEB for short in the figures) in all of our settings. The difference between SRC$_S$ and these three protocols is relatively moderate, since all of them incur additive overhead. For each of them: SRC$_S$ is faster than ART, partly because the novel gauge used by ART does not perform as well as the simpler gauge used by SRC$_S$ (see Section 6), and partly because the quality of the rough estimate in ART is overly low. SRC$_S$ is faster than ZOE for the following two reasons. First, recall that ZOE uses a single slot for each trial, while SRC$_S$ puts all its slots in the second phase into a single trial. Therefore for the second phase of SRC$_S$, whether a slot is empty becomes negatively correlated with each other. Such negative correlation makes the total number of empty slots concentrate better near its expected value and thus provides higher estimation equality, as compared to a design using independent slots like ZOE. Second, each slot in ZOE needs to incur per-trial overhead since each of them corresponds to an individual trial, while the per-trial overhead is incurred much less often in SRC$_S$. For enhanced FNEB, recall that each of its trials also only uses a small number of slots. Therefore, the same two reasons that explain why SRC$_S$ is faster than ZOE also apply here. In addition, the quality of the rough estimate in enhanced FNEB is also lower than desirable. Finally, our results show that UPE cannot support relative error $\epsilon < 0.03$ due to its biased estimator. This is consistent with the findings by the original authors [12].

As we see, the overhead difference between SRC$_S$ and some existing protocols is partly due to the existence of per-trial overhead. To understand how significant this factor is, we have further compared the protocols when there is no per-trial overhead. We find SRC$_S$ continues to have the lowest overhead among all protocols. For example, when $\epsilon = 0.01$, SRC$_S$ is 20% to 100% faster than the most efficient existing protocol, i.e., ZOE. See detailed results in the Appendix F.

Comparing SRC$_M$ with existing multiple-set protocols. Figure 7 presents the overhead of our SRC$_M$ protocol against existing multiple-set RFID counting protocols. We perform extensive experiments under different values of n and k, as well as different ways that the sets overlap with each other. Since they all show similar trends, Figure 7 presents a concrete setting, where a total of $n = 100,000$ tags (with index from 1 to $100,000$) are distributed over $k = 10$ overlapping sets. For $i = 1, \ldots, 9$, the ith set is comprised of $11,000$ tags with index from $(i-1) \times 10000+1$ to $i \times 10000+1000$. The last set is comprised of $10,000$ tags with index from 90001 to 100000. In this setting, our SRC$_M$ protocol is around 500% faster than the most efficient existing multiple-set protocol, i.e., PET. In particular, while all existing protocols require more than 10 minutes to provide an estimate with relative error ϵ of 0.01, our SRC$_M$ protocol can achieve the same estimation quality in 2 minutes. The significant difference between SRC$_M$ and existing multiple-set protocols is mainly because asymptotically all existing multiple-set protocols incur multiplicative overhead, while SRC$_M$ incurs additive overhead (see Section 5).

Same to the single-set experiments, the overhead of multiple-set protocols partly comes from the per-trial overhead. To understand the significance of this factor here, we again evaluate a setting without per-trial overhead. We find that SRC$_M$ continues to be 300% faster than the most efficient existing protocol (see the Appendix F for details).

8 Variant Models

This section discusses some variants of RFID counting problem.

A simpler variant of multiple-set problem. Some researchers (e.g., [17]) consider a simpler variant of the multiple-set RFID counting problem, where multiple readers jointly cover an area. These readers together count the total number of tags under their coverage. One can actually solve this simpler variant of our multiple-set problem using *any* single-set RFID counting protocol. Recall that a single-set protocol specifies a predicate for each slot. Roughly speaking,

[8] UPE requires a tag to send more bits in a slot to detect collision.

all readers send the same predicate to their sets. Together the readers return an empty slot to the single-set protocol iff every reader sees an empty slot. Note that this takes care of potential overlaps between sets, as long as a tag behaves identically for the same predicate from different readers.

Capability to detect collision. Some protocols (e.g., [11]) assume a reader can further detect collision, i.e., whether there are multiple tags responding in a non-empty slot. Though the reader becomes more capable in this variant model, we can still obtain a similarly strong lower bound result (with a small $\log \frac{1}{\epsilon}$ difference) as our original model. See the Appendix E for the proof.

Programmable tags vs. non-programmable tags. Same as many recent research efforts on RFID systems (e.g., [8, 15, 19, 23, 24]), our SRC$_S$ and SRC$_M$ protocol target programmable RFID tags that can run customized code. There have also been research work (e.g., [17]) that focuses on non-programmable RFID tags. These non-programmable tags can participate in a protocol only via a pre-determined way (e.g., only via framed slotted Aloha as specified in C1G2 [6]). We are currently working on adapting SRC$_S$ and SRC$_M$ to non-programmable tags. We already have initial designs for adapted SRC$_S$ and SRC$_M$, as well as promising preliminary results, though a full discussion into the subject is beyond the scope of this paper.

9 Related Work

Section 4 already reviewed major related RFID counting protocols [8, 11, 12, 15, 17, 23]. Same as this paper, these efforts all focus on improving the performance of RFID counting. There have also been efforts that optimize other metrics such as energy consumption [13]. In early days, researchers (e.g., [9, 22, 25]) focus on efficient identification of RFID tags. Obviously once all tags are identified, we will obtain an exact count of the tags. But the inherent $\Omega(n)$ complexity makes it impossible for large-scale RFID systems.

There are deep connections between RFID counting protocols and algorithms for counting the number of distinct elements in a data stream [10]. One can conceptually map a slot in RFID counting protocols to a memory bit in distinct element counting algorithms. Existing RFID counting protocols (including ours) have borrowed multiple ideas from distinct element counting algorithms (e.g., [7, 20]). These ideas include for example, the use of duplicate-insensitive statistical quantities to deal with the possible overlapping between sets in multiple-set RFID counting [8]. Furthermore, reduction from the Hamming Distance Estimation problem has also led to lower bounds on the memory space needed by distinct element counting algorithm [5]. Despite these deep connections, RFID counting and distinct element counting also have some fundamental differences. First, a memory bit in distinct element counting can be overwritten multiple times. A slot in RFID counting, however, can only be used once. Hence a distinct element counting technique that overwrites the same memory multiple times cannot be carried over directly to RFID counting. Second, RFID counting can have multiple passes/phases, while distinct element counting for data streams cannot.

10 Conclusion

In summary, we present three fundamental results about RFID counting protocols: We establish strong lower bounds for both the single-set and multiple-set problem. We show that the overlooked key aspect for RFID counting protocols is a conceptual separation of a protocol into two phases. Furthermore, other techniques/ideas proposed in the literature are only of secondary importance. Finally, we apply the obtained insights to design new protocols that are more efficient than existing ones and also simultaneously simpler than most of them. We hope that our results will help facilitate future research in this subject.

Acknowledgments

This work is partly supported by the research grant for the Human Sixth Sense Programme at the Advanced Digital Sciences Center from Singapore's Agency for Science, Technology and Research (A*STAR), and partly supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office.

References

[1] http://newzealand.msteched.com/.

[2] http://spectrum.ieee.org/riskfactor/computing/it/walmart-to-track-clothing-with-rfid-tags.

[3] http://www.exhibitionnews.co.uk/featuredetails/172/a-flick-of-the-wrist-rfid-wristbands-for-exhibitions.

[4] http://www.packaging-gateway.com/projects/purdue/.

[5] A. Chakrabarti and O. Regev. An optimal lower bound on the communication complexity of GAP-HAMMING-DISTANCE. In *STOC*, 2011.

[6] EPCglobal. EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz - 960 MHz Version 1.2.0. 2008.

[7] P. Flajolet and G. Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Science*, 31(2):182–209, 1985.

[8] H. Han, B. Sheng, C. C. Tan, Q. Li, W. Mao, and S. Lu. Counting RFID tags efficiently and anonymously. In *INFOCOM*, 2010.

[9] D. Hush and C. Wood. Analysis of tree algorithms for RFID arbitration. In *IEEE Symposium on Information Theory*, 1998.

[10] D. Kane, J. Nelson, and D. Woodruff. An optimal algorithm for the distinct elements problem. In *PODS*, 2010.

[11] M. Kodialam and T. Nandagopal. Fast and reliable estimation schemes in RFID systems. In *ACM MobiCom*, 2006.

[12] M. Kodialam, T. Nandagopal, and W. C. Lau. Anonymous tracking using RFID tags. In *INFOCOM*, 2007.

[13] T. Li, S. Wu, S. Chen, and M. Yang. Energy efficient algorithms for the RFID estimation problem. In *INFOCOM*, 2010.

[14] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[15] C. Qian, H. Ngan, Y. Liu, and L. Ni. Cardinality estimation for large-scale RFID systems. *IEEE Transactions on Parallel and Distributed Systems*, 22(9):1441–1454, September 2011.

[16] Y. Qiao, S. Chen, and T. Li. *RFID as an Infrastructure*. Springer, 2013.

[17] M. Shahzad and A. X. Liu. Every bit counts - fast and scalable RFID estimation. In *ACM MobiCom*, 2012.

[18] B. Sheng, C. Tan, Q. Li, and W. Mao. Finding popular categories for RFID tags. In *MobiHoc*, 2008.

[19] J. Wang, H. Hassanieh, D. Katabi, and P. Indyk. Efficient and reliable low-power backscatter networks. In *ACM SIGCOMM*, 2012.

[20] K. Whang, B. Vander-Zanden, and H. Taylor. A linear-time probabilistic counting algorithm for database applications. *ACM Transactions on Database Systems*, 15:208–229, June 1990.

[21] A. Yao. Probabilistic computations: Toward a unified measure of complexity. In *FOCS*, 1977.

[22] B. Zhen, M. Kobayashi, and M. Shimizu. Framed ALOHA for multiple RFID objects identification. *IEICE Transactions on Communications*, E88-B(3), March 2005.

[23] Y. Zheng and M. Li. PET: Probabilistic estimating tree for large-scale RFID estimation. *IEEE Transactions on Mobile Computing*, 11(11):1763–1774, November 2012.

[24] Y. Zheng and M. Li. ZOE: Fast cardinality estimation for large-scale rfid systems. In *INFOCOM*, 2013.

[25] F. Zhou, C. Chen, D. Jin, C. Huang, and H. Min. Evaluating and optimizing power consumption of anti-collision protocols for applications in RFID systems. In *ACM ISLPED*, 2004.

A Lower Bound Proofs (for Section 3)

A.1 Single-set Lower bound proof

To prove the lower bound for single-set RFID counting protocols in our Theorem 1, we show that given a protocol $\mathcal{P}$ that can output an $(\epsilon, 0.2)$ estimate, we can construct a communication protocol that solves the *Hamming Distance Estimation* (HDE) problem. Recall that HDE is a two-party communication complexity problem, where the two parties Alice and Bob are given m-bit strings x and y as input respectively. They would like to estimate the hamming distance between x and y, denoted by $\triangle(x, y)$, with (ϵ, δ) estimation quality. Then we can use the fact that the constructed HDE communication protocol must comply to the HDE lower bound to get our result.

Before proving Theorem 1, we firstly show the lower bound of the HDE communication complexity in the following lemma. Here we simply translate a recent breakthrough result by Chakrabarti and Regev [5], which considers a boolean version of the HDE problem called Gap Hamming Distance (GHD). In GHD (as compared to HDE), Alice and Bob are further given the promise that either $\triangle(x, y) \geq \frac{m}{2} + \sqrt{m}$ or $\triangle(x, y) \leq \frac{m}{2} - \sqrt{m}$. They should output 1 iff $\triangle(x, y)$ satisfies the first inequality. Chakrabarti and Regev [5] prove that no protocol can solve GHD by communicating $o(m)$ bits, even for randomized protocols that are allowed to err with some small constant probability (e.g., 1/3) on each input.

Lemma 7. *No protocol can solve HDE (parameterized with m, ϵ, and δ) by exchanging $o(m)$ bits, for $\epsilon < \frac{2}{\sqrt{m}}$ and $\delta = \frac{1}{3}$.*

Proof. We prove by contradiction. Assume there is a randomized protocol that solves HDE for $\epsilon < \frac{2}{\sqrt{m}}$ and $\delta = \frac{1}{3}$ while exchanging $o(m)$ bits, we show that this randomized protocol can be directly used to solve GHD. In particular, this protocol returns an estimate $\widehat{\triangle(x, y)}$ that satisfies $|\widehat{\triangle(x, y)} - \triangle(x, y)| \leq \epsilon \triangle(x, y)$ with probability $\geq \frac{2}{3}$. When this holds, if $\triangle(x, y) \leq \frac{m}{2} - \sqrt{m}$:

$$\widehat{\triangle(x, y)} \leq (1 + \epsilon) \triangle (x, y) \leq (1 + \epsilon)(\frac{m}{2} - \sqrt{m}) \qquad (1)$$

otherwise, $\triangle(x, y) \geq \frac{m}{2} + \sqrt{m}$, thus:

$$\widehat{\triangle(x, y)} \geq (1 - \epsilon) \triangle (x, y) \geq (1 - \epsilon)(\frac{m}{2} + \sqrt{m}) \qquad (2)$$

Since $\epsilon < \frac{2}{\sqrt{m}}$,

$$(1 + \epsilon)(\frac{m}{2} - \sqrt{m}) < (1 - \epsilon)(\frac{m}{2} + \sqrt{m}) \qquad (3)$$

Alice and Bob can solve the GHD problem (with err probability less than $\frac{1}{3}$) by picking a threshold between the two values in Equation (3) and outputting 1 *iff* $\widehat{\triangle(x, y)}$ is above the threshold. This contradicts with the GHD lower bound [5]. □

Proof for Theorem 1. Consider the HDE problem with $m = \lceil \frac{1}{\epsilon^2} \rceil$. Note that $m = \lceil \frac{1}{\epsilon^2} \rceil$ leads to $\epsilon < 2/\sqrt{m}$, thus lemma 7 applies here. Also, the condition $\epsilon \in [\frac{1}{\sqrt{n}}, 0.5]$ given in the theorem leads to $1 < m \leq n$.

Given any single-set RFID counting protocol $\mathcal{P}$ that can estimate up to n tags with $(\epsilon, 0.2)$ estimation quality, Alice and Bob can solve HDE by simulating $\mathcal{P}$ on an RFID counting problem input defined as follows: for $i = 1, 2, \ldots, m$, tag i is present and need to be included in the count iff the $x[i] \neq y[i]$. All other tags are absent and will not be included in the count. This will make the RFID count to exactly equal the hamming distance between x and y. Thus if Alice and Bob can simulate $\mathcal{P}$ and $\mathcal{P}$ returns a result with relative error no greater than ϵ, they can directly use the same result to solve the original HDE problem with relative error no greater than ϵ.

Now to properly simulate the execution of $\mathcal{P}$ with those present tags, Alice/Bob needs to determine which slots in the simulated execution of $\mathcal{P}$ are empty. Doing so enables Alice/Bob to simulate the responses received in all these slots and feed those into $\mathcal{P}$ to obtain the final count. For each slot, we will show that Alice and Bob can determine whether it is empty by only exchanging $O(\log \frac{1}{\epsilon})$ bits. Consider the first slot. $\mathcal{P}$ must have specified a predicate f for the first slot. Assume Alice and Bob have access to a shared random string needed to determine this predicate (we will release this assumption later). Alice/Bob can thus locally determine the set of tags that satisfy f, denoted by $\{i_1, i_2, \ldots, i_j\}$. Next Alice computes a short fingerprint of h bits for the (potentially long) string of $x[i_1]x[i_2]\ldots x[i_j]$ and sends it to Bob. Bob similarly computes the fingerprint over $y[i_1]y[i_2]\ldots y[i_j]$ and compares the two fingerprints. Bob uses one bit to inform Alice about the comparison result. We will discuss how to select the parameter h to properly address the fingerprint collision problem later. For now let us assume there is no fingerprint collisions. Then the two fingerprints differ iff for at least one index i that satisfies f, $x[i] \neq y[i]$. This in turn is equivalent to at least one tag being present, and also equivalent to the first slot being non-empty. Alice and Bob now have successfully determined whether the first slot is empty or not. Emptiness of later slots can be sequentially determined in a similar way.

Let $\mathcal{E}$ denote the event that Alice and Bob correctly simulate all T slots used by $\mathcal{P}$, and let $\overline{\mathcal{E}}$ denote the event otherwise. The simulated outcome for a slot becomes incorrect iff the hash function maps two different input bit vectors to the same value, which happens with probability $\frac{1}{2^h}$ for h-bit fingerprints. By union bound:

$$Pr(\overline{\mathcal{E}}) \leq \sum_{i=1}^{T} Pr(\text{the } i\text{th slot is incorrect}) \leq \frac{T}{2^h} \qquad (4)$$

Thus:

$$\begin{aligned} & Pr(\text{Alice/Bob solves HDE with relative error } \leq \epsilon) \\ \geq\ & Pr(\mathcal{P}\text{'s relative error} \leq \epsilon | \mathcal{E}) Pr(\mathcal{E}) \geq 0.8(1 - \frac{T}{2^h}) \end{aligned} \qquad (5)$$

Let $h = \lceil \log 6T \rceil$ ($\log$ in this paper means $\log_2$):

$$\begin{aligned} & Pr(\text{Alice/Bob solves HDE with relative error } \leq \epsilon) \\ \geq \quad & 0.8(1 - \frac{1}{6}) = \frac{2}{3} \end{aligned} \qquad (6)$$

Recall that Alice and Bob exchange $h + 1$ bits for each of the T slots used by $\mathcal{P}$. Also, to release the assumption that Alice and Bob have access to a shared random string, we apply the well known result that a shared random string protocol can be simulated by a private string protocol that uses an extra $O(\log m)$ bits when the input has $O(m)$ bits (i.e., the size of x and y for HDE). Thus, given an $(\epsilon, 0.2)$-approximate protocol $\mathcal{P}$, we can construct a protocol that solves HDE (for $\epsilon < \frac{2}{\sqrt{m}}$ and $\delta = \frac{1}{3}$) while requiring Alice and Bob to communicate $O(T(\lceil \log 6T \rceil + 1) + \log m) = O(T \log T + \log \frac{1}{\epsilon})$ bits. If there exists an RFID counting protocol $\mathcal{P}$ with an overhead of $o(\frac{m}{\log m})$, i.e., on expectation, $T = o(\frac{1}{\epsilon^2 \log \frac{1}{\epsilon}})$, Alice and Bob can construct a communication protocol that solves HDE while incurring a communication complexity of $o(\frac{1}{\epsilon^2 \log \frac{1}{\epsilon}} log(\frac{1}{\epsilon^2 \log \frac{1}{\epsilon}})) + O(log \frac{1}{\epsilon}) = o(m)$. This contradicts with Lemma 7.

Proof for Theorem 2. Our proof uses Yao's minimax principle [21], which states that distributional complexity provides a lower bound for randomized complexity even for the case that tolerates errors (see Theorem 3 in [21]). To invoke Yao's minimax principle for randomized RFID counting protocols that provide $(\epsilon, 0.2)$ estimation quality, we consider the expected cost of deterministic protocols that provide $(\epsilon, 0.4)$ estimation quality over the following input distribution.

Construct $\theta(\log n)$ different inputs, with the number of tags in the ith input being 4^i for $i = 1, \ldots, \lfloor \log_4 n \rfloor$. As $\epsilon \leq 0.5$, our construction ensures that given any two different inputs, they cannot be approximated within ϵ-relative error by the same value. Consider an input distribution such that each of these inputs appears with the equal probability. Against this distribution, any deterministic protocol that provides $(\epsilon, 0.4)$ estimation quality needs to provide proper estimation over at least 60% of possible inputs. Thus it needs to output $\theta(\log n)$ different values. If a deterministic protocol only uses $o(\log \log n)$ slots where each slot has only two possible outcomes, it can have at most $o(\log n)$ different outputs, which is not sufficient here. Therefore no deterministic protocol can has $o(\log \log n)$ overhead while providing the $(\epsilon, 0.4)$ estimation quality. From here, applying Yao's minimax principle leads to our result. □

A.2 Multiple-set lower bound proof

We leverage our single-set lower bound to reason about the lower bound for the multiple-set RFID counting problem. We cannot direct apply our single-set lower bound here, as a multiple-set RFID counting protocol is required to only estimate the union size of all sets, and it can potentially optimize its execution based on completed counting instances. Despite of these differences, we are able to construct worst-case scenarios to prove that the multiple-set lower bound can be expressed as a direct sum of the single-set lower bounds over individual sets.

Proof for Theorem 4. Consider any multiple-set protocol $\mathcal{P}$ that provides $(\epsilon, 0.2)$ estimation quality. Let $w_i = \sum_{j=1}^{i} n_j$ and let $\hat{w_i}$ denote the estimate for w_i. $\mathcal{P}$ needs to obtain an $(\epsilon, 0.2)$-approximate estimate $\hat{w_1}$ for w_1 (equivalently n_1) at the first location. This is needed in case it sees empty sets at all the following locations, i.e., $n_i = 0$ for $i \geq 2$, which will result in the union size being exactly n_1. Note that $\mathcal{P}$ cannot defer its estimation of n_1 to the moment when it visits more locations, since the reader does not revisit a previous location. Therefore, we can directly apply our single-set lower bound (Corollary 3) to show that no $\mathcal{P}$ can use $o(\frac{1}{\epsilon^2 \log \frac{1}{\epsilon}} + \log\log n_1)$ slots at its first location.

For the second location, let $n_2 \geq n_1$ and assume there is no overlapping between the two sets. Apply the same reasoning as earlier (i.e. considering the case that it sees empty sets at all the following locations), $\mathcal{P}$ needs to obtain $\hat{w_2}$ with $(\epsilon, 0.2)$ estimation quality for w_2 at the second location. Given an accurate n_1 and such a $\hat{w_2}$, $\hat{w_2} - n_1$ is an $(2\epsilon, 0.2)$-approximate estimate for n_2 since $n_2 \geq n_1$.

Let n_2 vary over the range from n_1 to n_1^2, we prove by contradiction that no $\mathcal{P}$ can use $o(\frac{1}{\epsilon^2 \log \frac{1}{\epsilon}} + \log\log n_1)$ slots at second location. Assume there is such a protocol $\mathcal{P}$, we will use it to construct a single-set RFID counting protocol $\mathcal{P}'$ that operates over the range of $[0, n_1]$ (with n_1 known) and provides $(2\epsilon, 0.2)$ estimation quality with only $o(\frac{1}{\epsilon^2 \log \frac{1}{\epsilon}} + \log\log n_1)$ overhead. In its first slot, $\mathcal{P}'$ asks all tags to respond. $\mathcal{P}'$ completes by outputting 0 immediately iff it sees an empty slot. Otherwise, the tag count must fall in the range of $[1, n_1]$. $\mathcal{P}'$ then asks every single tag to simulate n_1 independent virtual tags. This can be easily done by using specific forms of predicate function. The number of all the virtual tags then falls in the range of $[n_1, n_1^2]$. To simulate a virtual tag, a real tag responds in a slot iff the virtual tag responds. $\mathcal{P}'$ invokes the second counting instance of $\mathcal{P}$ over the virtual tags. By doing so, $\mathcal{P}'$ uses only $o(\frac{1}{\epsilon^2 \log \frac{1}{\epsilon}} + \log\log n_1)$ slots and $\mathcal{P}'$ can obtain a $(2\epsilon, 0.2)$-approximate estimation of the actual number of tags by outputting $\hat{w_2}/n_1 - 1$. Since $\epsilon \in [\frac{1}{\sqrt{\min_i n_i}}, 0.25]$, the existence of $\mathcal{P}'$ contradicts with our single-set lower bound as proved in Corollary 3. Thus, no $\mathcal{P}$ can incur only $o(\frac{1}{\epsilon^2 \log \frac{1}{\epsilon}} + \log\log n_1) = o(\frac{1}{\epsilon^2 \log \frac{1}{\epsilon}} + \log\log n_2)$ overhead at the second location.

For $i = 3, \ldots, k$, let n_i vary over the range from w_{i-1} to w_{i-1}^2. We can apply the same argument as at the second location to prove that no $\mathcal{P}$ can incur only $o(\frac{1}{\epsilon^2 \log \frac{1}{\epsilon}} + \log\log n_i)$ overhead at its ith location. Finally, by linearity of expectation, combining the overhead lower bounds of all the k locations leads to our final result. □

B Asymptotic Overhead of Existing RFID counting Protocols

This section discusses the asymptotic overhead of major existing RFID counting protocols (see Section 4). Among these protocols, UPE [11] and ART [17] do not come with end-to-end overhead analysis. Recall that the estimator used by UPE is biased, hence UPE cannot be used when ϵ is small. Therefore one cannot analyze UPE's overhead in an asymptotic manner with ϵ approaches 0. We will examine ART first, then briefly summarizes the analysis of other protocols.

B.1 Asymptotic Overhead of ART

Since ART [17] does not come with end-to-end overhead analysis, we analyze its asymptotic overhead by ourselves. Recall that ART has two phases and its key idea is to use the average run length of non-empty slots in its second phase to do the estimation. In addition, for its second phase, ART also solves involved optimization problems to determine the parameters and the final estimate. Instead of analyzing these optimization problems, we here analyze a simplified version of its second phase, which keeps the average-run-length key design, but uses closed-form formulas to determine the parameters and the final estimate (to be explained later). Since the original optimization techniques are meant to reduce overhead, our analysis result for the simplified version would provide an upper bound for the overhead of the original ART second phase. Specifically, we prove that the simplified version of ART achieves $O(\log n + \frac{1}{\epsilon^2})$ asymptotic overhead. Since this is close to our RFID counting lower bound (see Appendix A) and no protocol's overhead can be smaller than the lower bound, the asymptotic overhead of the original ART would be roughly $O(\log n + \frac{1}{\epsilon^2})$.

ART's first phase. Recall that ART's first phase is similar to a single trial of LOF. Specifically, for its ith ($i = 1, 2, \ldots$) time slot, each tag responds with probability $\frac{1}{2^{i-1}}$. Let x denote the index of the first empty slot, ART then calculates a rough estimate $\tilde{n} = 1.2897 \times 2^{x-2}$. Note that $x = 1$ iff $n = 0$, therefore in this special case, one can directly return 0 as the exact tag count. Lemma 8 summarizes the guarantee of ART's first phase.

Lemma 8. *ART's first phase outputs a rough estimate $\tilde{n}$ with $\Pr(\tilde{n} \in [0.16n, 10.4n]) > 0.95$ and incurs $O(\log n)$ overhead.*

Proof. Let q_i denote the probability that the ith slot is empty, $q_i = (1 - \frac{1}{2^{i-1}})^n$. We have: i) for all i, $q_i \leq e^{-\frac{n}{2^{i-1}}}$; and ii) for $i \geq 2$, $q_i \geq e^{-\frac{n}{2^{i-1}-1}}$. These can be derived easily by the fact that for all y, $e^y \geq 1 + y$ and with $y = -\frac{1}{2^{i-1}}$ and $y = \frac{1}{2^{i-1}-1}$ (for $i \geq 2$) respectively. Let r_i denote the probability that the index of the first empty slot is i, $r_i = (1-q_1)(1-q_2)\ldots(1-q_{i-1})q_i$. Therefore, for all i, $r_i \leq q_i$.

Let x denote the random variable of the index of the first empty slot. Since $x = 1$ iff $n = 0$, which allows one to directly return 0 as an exact tag count, we discuss the case of $n > 0$ below. To analyze the distribution of x, we let $u = \lceil \log n \rceil$ (therefore $2^{u-1} < n \leq 2^u$). We have:

$$\begin{aligned}
& \Pr(x < u - 1) \\
= \ & \sum_{i=1}^{u-2} r_i \leq \sum_{i=1}^{u-2} q_i \\
\leq \ & \sum_{i=1}^{u-2} e^{-\frac{n}{2^{i-1}}} \leq \sum_{i=1}^{u-2} e^{-\frac{2^{u-1}}{2^{i-1}}} \\
\leq \ & e^{-2^{u-1}} + e^{-2^{u-2}} + \ldots + e^{-8} + e^{-4} \\
< \ & \frac{e^{-4}}{1 - e^{-4}} < 0.019
\end{aligned}$$

Next we want to bound $\Pr(x > u+4)$. For $n = 1$, $u = 0$, $\Pr(x > 4) = 1 \times \frac{1}{2} \times \frac{1}{4} \times \frac{1}{8} = 0.0156$. For $n > 1$ (therefore $u \geq 1$), we have:

$$\begin{aligned} & \Pr(x > u+4) \\ = \quad & \Pr(x > u+1)(1-q_{u+2})(1-q_{u+3})(1-q_{u+4}) \\ \leq \quad & (1-q_{u+2})(1-q_{u+3})(1-q_{u+4}) \\ \leq \quad & (1-e^{-\frac{n}{2^{u+1}-1}})(1-e^{-\frac{n}{2^{u+2}-1}})(1-e^{-\frac{n}{2^{u+3}-1}}) \\ \leq \quad & (1-e^{-\frac{2^u}{2^{u+1}-1}})(1-e^{-\frac{2^u}{2^{u+2}-1}})(1-e^{-\frac{2^u}{2^{u+3}-1}}) \\ \leq \quad & (1-e^{-\frac{2}{3}})(1-e^{-\frac{2}{7}})(1-e^{-\frac{2}{15}}) < 0.0152 \end{aligned}$$

Therefore, we have:

$$\begin{aligned} & \Pr(x \in [u-1, u+4]) \\ > \quad & 1 - max(0.0152, 0.0156) - 0.019 \\ > \quad & 0.95 \end{aligned}$$

When $x \in [u-1, u+4]$, $\tilde{n} = 1.2897 \times 2^{x-2} \in [1.2897 \times 2^{u-3}, 1.2897 \times 2^{u+2}] \subset [1.2897 \times \frac{n}{2^3}, 1.2897 \times n \times 2^3] \subset [0.16n, 10.4n]$. Therefore, $\Pr(\tilde{n} \in [0.16n, 10.4n]) > 0.95$.

To bound the expected number of slots in ART's first phase, note that for $i \geq u+4$ (therefore $2^{i-4} \geq 2^u \geq n$ and $i \geq 4$), $q_i \geq e^{-\frac{n}{2^{i-1}-1}} \geq e^{-\frac{2^{i-4}}{2^{i-1}-1}} \geq e^{-\frac{2^{4-4}}{2^{4-1}-1}} > 0.8$. Therefore, $\frac{r_{i+1} \times (i+1)}{r_i \times i} = \frac{q_{i+1}(1-q_i)}{q_i} \times \frac{i+1}{i} < \frac{1 \times (1-0.8)}{0.8} \times \frac{5}{4} < \frac{1}{2}$. We have: $E[x] = \sum_{i=1}^{\infty} r_i \times i = \sum_{i=1}^{u+3} r_i \times i + \sum_{i=u+4}^{\infty} r_i \times i < \Pr(x \leq u+3) \times (u+3) + \frac{r_{u+4} \times (u+4)}{1-\frac{1}{2}} < 1 \times (\log n + 1 + 3) + \frac{1 \times (\log n + 1 + 4)}{1 - \frac{1}{2}} = 3 \log n + 14 = O(\log n)$. □

A simplified second phase of ART. Before we proceed, we first present a fact that will be used here and in Appendix D.

Lemma 9. *With constant* $q \in (0,1)$, $\frac{1}{(1-q^\epsilon)^2} = O(\frac{1}{\epsilon^2})$ *for* $\epsilon > 0$.

Proof. Note the fact that for all y, $e^y \geq 1 + y$. Therefore for $y > 0$, $\frac{1}{1+y} \geq e^{-y}$, and $1 - e^{-y} \geq 1 - \frac{1}{1+y} = \frac{y}{1+y}$. Consider $y = \epsilon ln \frac{1}{q}$. $\frac{1}{(1-q^\epsilon)^2} = \frac{1}{(1-e^{-y})^2} \leq \frac{1}{(\frac{y}{1+y})^2} = (\frac{1}{y}+1)^2 = (\frac{1}{\epsilon ln \frac{1}{q}} + 1)^2 = O(\frac{1}{\epsilon^2})$. □

For the second phase of ART, we consider a simplified version that retains its key design of using average run length of non-empty slots, but uses closed-form formulas to determine parameters and final estimate (instead of solving involved optimization problems). Specifically, given the rough estimate $\tilde{n}$ from the first phase of ART, the simplified ART second phase uses multiple slots, and for each slot a tag independently responds with probability $p = \min\{\frac{1}{11}, \frac{1}{\tilde{n}}\}$. Therefore, the probability that a slot is empty is $q = (1-p)^n$. The simplified second phase terminates when it sees $\max\{\frac{4(1-q)}{(1-q^\epsilon)^2} | q \in (\frac{1}{1000}, \frac{10}{11})\}$ runs of non-empty slots or the number of time slots used exceeds $\max\{\frac{40}{q(1-q^\epsilon)^2} + 10 | q \in (\frac{1}{1000}, \frac{10}{11})\}$. In Theorem 12 we will prove this is suffice to output an $(\epsilon, 0.2)$ estimate. Let x_i denote the run-length of ith run of non-empty slots and m be the total number of runs before terminate. ART computes $x = \frac{\sum_{i=1}^{m} x_i}{m}$. Here x is the average run-length of non-empty slots. Our simplified version of ART then outputs $\hat{n} = -\frac{\ln x}{\ln(1-p)}$ as the final estimate. The following lemmas summarizes the guarantee of this simplified second phase of ART:

Lemma 10. *The second phase of the simplified ART incurs* $O(\frac{1}{\epsilon^2})$ *overhead.*

Proof. By our design, the simplified ART will use at most $\max\{\frac{40}{q(1-q^\epsilon)^2} + 10 | q \in (\frac{1}{1000}, \frac{10}{11})\}$ slots. By Lemma 9, this overhead is $O(\frac{1}{\epsilon^2})$. □

Lemma 11. *Given* $\tilde{n} \in [0.16n, 10.4n]$, *the simplified ART second phase outputs an* $(\epsilon, 0.15)$ *estimate.*

Proof. Since we are analyzing the asymptotic overhead, we consider the case where m, the number of runs of non-empty slots, is sufficiently large. Since the random variables x_i are independent from each other, and they all follow the same geometric distribution (with mean of $\frac{1}{q}$ and variance of $\frac{1-q}{q^2}$), by central limit theorem, we have: $x \sim \mathcal{N}(\frac{1}{q}, \frac{1-q}{mq^2})$, i.e., a normal distribution with mean of $E[x] = \frac{1}{q}$ and variance $\sigma^2 = \frac{1-q}{mq^2}$.

Given $\tilde{n} \in [0.16n, 10.4n]$, we first calculate the range of q, which is the probability that a slot is empty. Recall $q = (1-p)^n$ and $p = \min\{\frac{1}{11}, \frac{1}{\tilde{n}}\}$. If $p = \frac{1}{11}$, $q = (1-\frac{1}{11})^n \leq (1-\frac{1}{11})^1 = \frac{10}{11}$. Otherwise, $p = \frac{1}{\tilde{n}}$ and $q = (1-\frac{1}{\tilde{n}})^n \leq (1-\frac{1}{10.4n})^n < e^{-1/10.4} < \frac{10}{11}$. Combining these two cases, we have $q \leq \frac{10}{11}$. To bound q from below, we also consider two cases: if $n \leq 70$, since $p = \min\{\frac{1}{11}, \frac{1}{\tilde{n}}\} \leq \frac{1}{11}$, $q \geq (1-\frac{1}{11})^{70} > \frac{1}{1000}$. Otherwise $n > 70$, since $\tilde{n} \geq 0.16n > 11$, $\frac{1}{\tilde{n}} < \frac{1}{11}$ and $p = \frac{1}{\tilde{n}}$. In this case, $q = (1-\frac{1}{\tilde{n}})^n \geq (1-\frac{1}{0.16n})^n > (1-\frac{1}{0.16 \times 70})^{70} > \frac{1}{1000}$. Combining these two cases, we have $q > \frac{1}{1000}$. Therefore, $q \in (\frac{1}{1000}, \frac{10}{11})$.

Next we show that for any value of q, if the protocol sees $m = \frac{4(1-q)}{(1-q^\epsilon)^2}$ runs of non-empty slots, it is sufficient to output an $(\epsilon, 0.05)$ estimate of n. Recall that for $n > 0$, the final estimate is $\hat{n} = -\frac{\ln x}{\ln(1-p)}$, we now bound its tail distribution:

$$\begin{aligned} & \Pr(|\hat{n} - n| > \epsilon n) \\ = \quad & \Pr(\hat{n} > (1+\epsilon)n) + Pr(\hat{n} < (1-\epsilon)n) \\ = \quad & \Pr(-\frac{\ln x}{\ln(1-p)} > (1+\epsilon)n) \\ & + \Pr(-\frac{\ln x}{\ln(1-p)} < (1-\epsilon)n) \\ = \quad & \Pr(x > (\frac{1}{(1-p)^n})^{1+\epsilon}) + \Pr(x < (\frac{1}{(1-p)^n})^{1-\epsilon}) \\ = \quad & \Pr(x > (\frac{1}{q})^{1+\epsilon}) + \Pr(x < (\frac{1}{q})^{1-\epsilon}) \\ = \quad & \Pr(x > E[x]^{1+\epsilon}) + \Pr(x < E[x]^{1-\epsilon}) \\ = \quad & \Pr(x - E[x] > (E[x]^\epsilon - 1)E[x]) \\ & + \Pr(x - E[x] < (E[x]^{-\epsilon} - 1)E[x]) \\ < \quad & \Pr(x - E[x] > (1 - E[x]^{-\epsilon})E[x]) \\ & + \Pr(x - E[x] < (E[x]^{-\epsilon} - 1)E[x]) \\ = \quad & \Pr(|x - E[x]| > (1 - E[x]^{-\epsilon})E[x]) \end{aligned}$$

Let $\epsilon' = 1 - E[x]^{-\epsilon} = 1 - q^{\epsilon}$. To bound $\Pr(|x - E[x]| > \epsilon' E[x]) < 0.05$, we need $\Pr(|\frac{x-E[x]}{\sigma}| \le \frac{\epsilon' E[x]}{\sigma}) \ge 0.95$. Since $x \sim \mathcal{N}(E[x], \sigma^2)$, we need $\frac{\epsilon' E[x]}{\sigma} \ge \sqrt{2}erf^{-1}(0.95)$. Since $\sqrt{2}erf^{-1}(0.95) < 2$, setting $\sigma^2 = \frac{(\epsilon' E[x])^2}{4}$ suffices. Recall that $E[x] = \frac{1}{q}$ and $\sigma^2 = \frac{1-q}{mq^2}$, we have $m \ge \frac{4(1-q)}{\epsilon'^2} = \frac{4(1-q)}{(1-q^{\epsilon})^2}$. Since $q \in (\frac{1}{1000}, \frac{10}{11})$, we can conclude that $\max\{\frac{4(1-q)}{(1-q^{\epsilon})^2} | q \in (\frac{1}{1000}, \frac{10}{11})\}$ runs is sufficient to achieve an $(\epsilon, 0.05)$ estimate.

Let z denote the average run length of empty slots. By reasoning similar to x, we have $E[z] = \frac{1}{1-q}$. When there are m runs of non-empty slots, there are at most $m+1$ runs of empty slots, with the last run having a single empty slot to terminate the protocol.

Next we bound the probability that for any given q, the number of runs of non-empty slot is less than $\frac{4(1-q)}{(1-q^{\epsilon})^2}$ while the protocol has used up all the $\max\{\frac{40}{q(1-q^{\epsilon})^2} + 10 | q \in (\frac{1}{1000}, \frac{10}{11})\}$ slots. Let random variable l denote the number of slots needed by the protocol to see $\frac{4(1-q)}{(1-q^{\epsilon})^2}$ runs of non-empty slot. We have $E[l] \le \frac{4(1-q)}{(1-q^{\epsilon})^2} \times (E[x] + E[z]) + 1 = \frac{4(1-q)}{(1-q^{\epsilon})^2} \times (\frac{1}{q} + \frac{1}{1-q}) + 1 = \frac{4}{q(1-q^{\epsilon})^2} + 1 \le \max\{\frac{4}{q(1-q^{\epsilon})^2} + 1 | q \in (\frac{1}{1000}, \frac{10}{11})\}$. Let $\mathcal{L}$ denote the event that $l \le \max\{\frac{40}{q(1-q^{\epsilon})^2} + 10 | q \in (\frac{1}{1000}, \frac{10}{11})\}$ and $\bar{\mathcal{L}}$ denote the event that happens otherwise. By Markov inequality:

$$\begin{aligned} & \Pr(\bar{\mathcal{L}}) \\ = \; & \Pr(l > 10 \times \max\{\frac{4}{q(1-q^{\epsilon})^2} + 1 | q \in (\frac{1}{1000}, \frac{10}{11})\}) \\ < \; & \Pr(l > 10E[l]) < 0.1 \end{aligned}$$

By union bound, the probability that of $|\hat{n} - n| > \epsilon n$ is:

$$\begin{aligned} & \Pr(|\hat{n} - n| > \epsilon n) \\ = \; & \Pr((|\hat{n} - n| > \epsilon n) \bigcap \mathcal{L}) + \Pr((|\hat{n} - n| > \epsilon n) \bigcap \bar{\mathcal{L}}) \\ = \; & \Pr((|\hat{n} - n| > \epsilon n) | \mathcal{L}) \Pr(\mathcal{L}) + \\ & \Pr((|\hat{n} - n| > \epsilon n) | \bar{\mathcal{L}}) \Pr(\bar{\mathcal{L}}) \\ < \; & 0.05 \times 1 + 1 \times 0.1 = 0.15 \end{aligned}$$

□

Finally, combining Lemma 8, Lemma 10 and Lemma 11 leads to:

Theorem 12. *The simplified version of ART protocol outputs an* $(\epsilon, 0.2)$ *estimate with* $O(\frac{1}{\epsilon^2} + \log n)$ *slots.*

Proof. Let $\hat{n}$ be the final output of the simplified version of ART, the probability that $|\hat{n} - n| > \epsilon n$ is:

$$\begin{aligned} & \Pr(|\hat{n} - n| > \epsilon n) \\ = \; & \Pr((|\hat{n} - n| > \epsilon n) \bigcap (\tilde{n} \notin [0.16n, 10.4n])) \\ & + \Pr((|\hat{n} - n| > \epsilon n) \bigcap (\tilde{n} \in [0.16n, 10.4n])) \\ < \; & \Pr(\tilde{n} \notin [0.16n, 10.4n]) + \\ & \Pr((|\hat{n} - n| > \epsilon n) \mid (\tilde{n} \in [0.16n, 10.4n])) \times 1 \\ < \; & (1 - 0.95) + 0.15 = 0.2 \end{aligned}$$

Adding together the overhead of ART's first phase and its simplified second phase, the overall overhead of the simplified version of ART is $O(\frac{1}{\epsilon^2} + \log n)$. □

B.2 Asymptotic Overhead of Other Protocols

The other existing protocols, i.e., EZB [12], (enhanced) FNEB [8], LOF [15], PET [23], and ZOE [24], all come with detailed analysis on the number of slots needed. Here all we do is to simplify their more precise results to asymptotic forms (with adaption to our formulation when necessary).

EZB. Recall that EZB works on each of the $\Theta(\log n)$ narrow ranges. Each range needs $\frac{Z_\delta^2}{\epsilon^2 l} \frac{\frac{l}{5} - 1}{(\log \frac{l}{5})^2}$ trials [12], where Z_δ is a constant given constant δ. Omitting all constants (including the trial length l), EZB needs $O(\frac{1}{\epsilon^2} \log n)$ slots.

(Enhanced) FNEB. FNEB uses $\frac{c^2 e^{-n/l}(e^{n/l} - e^{-\epsilon n/l})^2}{(1 - e^{-\epsilon n/l})^2}$ trials [8], where c is a constant given a constant δ and n/l is also a constant. The total number of trials used by FNEB is hence $O(\frac{1}{\epsilon^2})$. In FNEB, if the actual number of tags is much smaller than the upper bound input by the user, FNEB needs to conduct a binary search incurring $O(\log n)$ slots in almost every trial. This results in an overhead of $O(\frac{1}{\epsilon^2} \log n)$. In enhanced FNEB, the binary search is likely to happen only in the first few trials (i.e., its first phase). The total number of slots hence is $O(\log n + \frac{1}{\epsilon^2})$.

LOF and PET. Both LOF and PET need to do $\max\{[\frac{-\sigma c}{log(1-\epsilon)}]^2, [\frac{-\sigma c}{\log(1+\epsilon)}]^2\}$ trials [15, 23], where σ is some constant in both cases and c is a constant given a constant δ. This corresponds to $O(\frac{1}{\epsilon^2})$ trials. Each trial takes (on expectation) $O(\log n)$ slots in LOF and $O(\log \log n)$ slots in PET. Hence LOF and PET needs $O(\frac{1}{\epsilon^2} \log n)$ and $O(\frac{1}{\epsilon^2} \log \log n)$ slots, respectively.

ZOE. ZOE first uses $O(\log \log n)$ slots to find a rough estimate. It then uses $[\frac{\sigma c}{e^{-\lambda}(1 - e^{-\epsilon\lambda})}]^2$ slots to eventually estimate n [24], which corresponds to $O(\frac{1}{\epsilon^2})$ slots with σ, c and λ all being constants. In total, ZOE needs $O(\frac{1}{\epsilon^2} + \log \log n)$ slots.

C Building Blocks of Our Protocols

This section discusses two building blocks of our protocols: (i) the lookup table for determining the number of slots needed in the second phase, and (ii) the revised version of PET.

C.1 Lookup table

ϵ	l for SRC_S	l for SRC_M
0.01	26575	28321
0.02	6638	6775
0.03	3009	3087
0.04	1674	1788
0.05	1075	1116

Table 4: Lookup Table for Determining l ($\delta = 0.2$).

For both SRC$_S$ and SRC$_M$, they need to decide the number of slots l needed in their second phase for achieving the

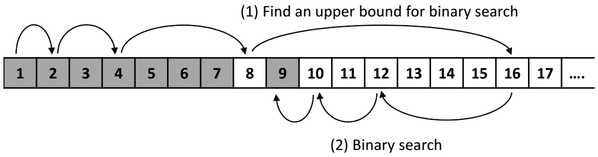

Figure 8: An example run of revised PET trial. An integer is shadowed if it is chosen by some tag, and the maximum integer chosen here is 9. As described in Algorithm 3, the protocol will firstly get an upper bound of all chosen integer (16 in this example), and then execute a binary search on the second half of the upper bound ($[8, 16)$ in this example), to find the maximum integer chosen.

required estimation quality. One way to determine the value of l is to construct a numerical lookup table by running the respective protocol under a wide range of n values, and then observing the l needed to achieve a certain relative error. For the same ϵ, the number of slots needed by SRC$_M$ is larger than SRC$_S$, since the participation probability used in SRC$_M$ needs to be rounded to the form of $1 = 2^x$.

Note that for SRC$_M$, since the final estimate is obtained by combining the results from all sets, it is equivalent to run the protocol directly against the union set. Hence when constructing the lookup table, one only needs to run SRC$_M$ against a single set and does not need to vary the number of sets as well as how the sets overlap.

Table 4 provides some sample values in the lookup table of SRC$_S$ and SRC$_M$.

C.2 Revised PET

Our SRC$_M$ protocol uses a revised version of PET for its first phase. Algorithm 3 describes the main steps of revised PET[9], and Figure 8 illustrates an example run of revised PET.

Lemma 13. *Each trial of revised PET incurs $O(\log \log n)$ overhead.*

Proof. In one trial of revised PET, each of n tags will choose an integer according to a geometric distribution with mean of 2. Let v be the maximum integer selected by all n tags. From analysis of PET [23] we know that $E[v] = O(\log n)$.

Since in each trial of revised PET, the protocol will firstly use j slots to make sure that $2^{j-2} \leq v < 2^{j-1}$, we have $j \leq \log(v) + 2$. Then revised PET will binary search on $[2^{j-2}, 2^{j-1}]$ to find the integer v, this uses $j - 2$ slots.

Therefore in each trial of revised PET will use $j + (j-2) \leq 2\log(v) + 2$ time slots.

Since $E[v] = O(\log n)$, on expectation, the number of time slots used by one trial of revised PET is $O(\log \log(n))$. □

Algorithm 3 revised PET algorithm (for $\epsilon = 0.5, \delta = 0.1$)

```
1:  Let all tags respond in the first slot: if it is empty, output
    $0$ and exit;
2:  for $i = 1$ to $30$ do
3:      Each tag randomly chooses a positive integer $u$ according
        to a geometric distribution with mean of $2$;
4:      $j = 1$;
5:      while true do
6:          Let all tags with $u \geq 2^{j-1}$ respond;
7:          if (See an empty slot) then
8:              Break;
9:          else
10:             $j = j + 1$;
11:         end if
12:     end while
13:     Binary search on $[2^{j-2}, 2^{j-1})$ to find the maximum in-
        teger $v_i$ that has been chosen by at least one tag;
14: end for
15: Output $\tilde{n} = 0.794 \times 2^{(\sum_{i=1}^{30} v_i)/30}$;
```

D Analysis of Our Protocols

In this section we first prove multiple technical lemmas for balls-and-bins trials. These prepare us to prove Theorem 5 (Section 7.1) and Theorem 6 (Section 7.2), which summarize the guarantees of our SRC$_S$ and SRC$_M$ protocol respectively.

D.1 Lemmas for balls-and-bins trials

Recall that for both SRC$_S$ and SRC$_M$, the building block of their second phase is the *balls-and-bins trial*, where n tags each independently and uniformly at random pick one slot from l slots, and respond in the chosen slot with probability p. Lemma 14 presents some simple yet useful properties about balls-and-bins trials.

Lemma 14. *Consider a balls-and-bins trial with n tags, l slots, and probability p for a tag to respond in its chosen slot. Let q denote the probability that a slot is empty and let z denote the number of empty slots in the trial. We have:*

(i) $q = (1 - \frac{p}{l})^n$;

(ii) $E[z] = lq = l(1 - \frac{p}{l})^n$;

(iii) $Var[z] = lq + l(l-1)(1 - \frac{2p}{l})^n - l^2q^2$;

(iv) $Var[z] \leq lq(1-q)$;

(v) if $p = 1$ *and* $n \leq l$, $Var[z] \leq lq - (l+n)q^2$.

Proof. (i) The probability that a tag responds in a given slot is $\frac{p}{l}$. Since a slot is empty iff none of the n tags responds in it, the probability that a slot is empty is $q = (1 - \frac{p}{l})^n$.

(ii) Let z_i be the indicator random variable for the event that the ith slot is empty. We have $z = \sum_{i=1}^{l} z_i$. From (i) above, $E[z_i] = q$ for all i. By the linearity of expectation, $E[z] = lq$.

(iii) Define z_i as above, we have $E[z_i^2] = E[z_i] = q$, and $E[z_iz_j] = Pr(z_i = 1, z_j = 1) = (1 - \frac{2p}{l})^n$. Hence $E[z^2] = lq + l(l-1)(1 - \frac{2p}{l})^n$, and $Var[z] = E[z^2] - E[z]^2 = lq + l(l-1)(1 - \frac{2p}{l})^n - l^2q^2$.

[9] The binary search in Line 13 of Algorithm 3 omits the range of $[1, 2^{j-2})$ because $v_i \geq 2^{j-2}$ (from Line 6).

(iv)

$$\begin{aligned} & Var[z] \\ = \ & E[z^2] - E[z]^2 = lq + l(l-1)(1-\frac{2p}{l})^n - l^2q^2 \\ \leq \ & lq + l(l-1)(1-\frac{2p}{l} + (\frac{p}{l})^2)^n - l^2q^2 \\ = \ & lq + l(l-1)q^2 - l^2q^2 = lq(1-q) \end{aligned}$$

(v) With $p = 1$ and $n \leq l$,

$$\begin{aligned} & Var[z] \\ = \ & lq + l(l-1)(1-\frac{2}{l})^n - l^2q^2 \\ = \ & lq + l(l-1)((1-\frac{1}{l})^2 - (\frac{1}{l})^2)^n - l^2q^2 \\ \leq \ & lq + l(l-1)((1-\frac{1}{l})^{2n} - n(1-\frac{1}{l})^{2n-2}(\frac{1}{l})^2 \\ & + \frac{n(n-1)}{2}(1-\frac{1}{l})^{2n-4}(\frac{1}{l})^4) - l^2q^2 \\ = \ & lq + l(l-1)q^2(1 - \frac{n}{(l-1)^2} + \frac{n(n-1)}{2(l-1)^4}) - l^2q^2 \\ = \ & lq - lq^2 - nq^2 - \frac{nq^2}{l-1} + \frac{ln(n-1)q^2}{2(l-1)^3} \\ = \ & lq - (l+n)q^2 - \frac{nq^2}{l-1}(1 - \frac{l(n-1)}{2(l-1)^2}) \\ \leq \ & lq - (l+n)q^2 \end{aligned}$$

The first inequation above uses the fact that $(x-y)^n \leq x^n - nx^{n-1}y + \frac{n(n-1)}{2}x^{n-2}y^2, for\ x, y > 0, n \geq 1$, which can be verified by a simple induction over n. □

Following the previous work on RFID counting [11,12], we make a normality assumption in our analysis below:

Assumption 1. The number of empty slots z in a balls-and-bins trial is distributed normally, i.e., $\frac{z-E[z]}{\sqrt{Var[z]}} \sim \mathcal{N}(0,1)$.

See [11, 12] for detailed discussion about the rationale behind this assumption.

Recall that both SRC$_S$ and SRC$_M$ output the final estimate as $\hat{n} = \frac{\ln(z/l)}{\ln(1-p/l)}$. Under Assumption 1, we now derive sufficient conditions for $\hat{n}$ to be an $(\epsilon, \frac{1}{9})$ estimate of n.

Lemma 15. *Consider a balls-and-bins trial with n tags, l slots, and probability p for a tag to respond in its chosen slot. Let z denote the number of empty slots in the trial and let $\hat{n} = \frac{\ln(z/l)}{\ln(1-p/l)}$. If $\frac{(1-(E[z]/l)^\epsilon)E[z]}{\sqrt{Var[z]}} > 1.6$, $Pr(|\hat{n} - n| > \epsilon n) < \frac{1}{9}$.*

Proof. Since $E[z] = l(1-\frac{p}{l})^n$ (Lemma 14), $n = \frac{\ln(E[z]/l)}{\ln(1-p/l)}$. We have:

$$\begin{aligned} & \Pr(|\hat{n} - n| > \epsilon n) \\ = \ & \Pr(\hat{n} > n(1+\epsilon)) + \Pr(\hat{n} < n(1-\epsilon)) \\ = \ & \Pr(\frac{\ln(z/l)}{\ln(1-p/l)} > \frac{\ln(E(z)/l)}{\ln(1-p/l)}(1+\epsilon)) \\ & + \Pr(\frac{\ln(z/l)}{\ln(1-p/l)} < \frac{\ln(E(z)/l)}{\ln(1-\frac{p}{l})}(1-\epsilon)) \\ = \ & \Pr(\frac{z}{l} < (\frac{E[z]}{l})^{1+\epsilon}) + \Pr(\frac{z}{l} > (\frac{E[z]}{l})^{1-\epsilon}) \\ = \ & \Pr(z < E[z](\frac{E[z]}{l})^\epsilon) + \Pr(z > E(z)(\frac{l}{E[z]})^\epsilon) \\ = \ & \Pr(z - E[z] < E[z]((\frac{E[z]}{l})^\epsilon - 1)) \\ & + \Pr(Z - E[z] > E[z]((\frac{l}{E[z]})^\epsilon - 1)) \\ < \ & \Pr(|Z - E[z]| > E[z](1 - (\frac{E[z]}{l})^\epsilon)) \\ = \ & \Pr(\frac{|z-E[z]|}{\sqrt{Var[z]}} > \frac{(1-(E[z]/l)^\epsilon)E[z]}{\sqrt{Var[z]}}) \end{aligned}$$

By Assumption 1, $\frac{z-E[z]}{\sqrt{Var[z]}} \sim \mathcal{N}(0,1)$. Also note that $\sqrt{2}erf^{-1}(\frac{8}{9}) < 1.6$. Therefore if $\frac{(1-(E[z]/l)^\epsilon)E[z]}{\sqrt{Var[z]}} > 1.6$, $\frac{(1-(E[z]/l)^\epsilon)E[z]}{\sqrt{Var[z]}} > \sqrt{2}erf^{-1}(\frac{8}{9})$. Hence $\Pr(\frac{|z-E[z]|}{\sqrt{Var[z]}} > \frac{(1-(E[z]/l)^\epsilon)E[z]}{\sqrt{Var[z]}}) < \Pr(\frac{|z-E[z]|}{\sqrt{Var[z]}} > \sqrt{2}erf^{-1}(\frac{8}{9})) = \frac{1}{9}$. Hence $\Pr(|\hat{n} - n| > \epsilon n) < \frac{1}{9}$. □

We now consider two specific settings where lemma 15 have more convenient forms. The corresponding results will be summarized in Lemma 16 and Lemma 17 below respectively.

Lemma 16. *Consider the setting in lemma 15 with the additional assumption that $\epsilon < 0.25$, $p = 1$, and $n \leq 0.6l$. If $l \geq \frac{25}{4\epsilon^2}$, $\Pr(|\hat{n} - n| > \epsilon n) < \frac{1}{9}$.*

Proof. We first derive some inequations useful for deriving the more convenient form in this setting.

i) Since $\epsilon < 0.25$, if $l \geq \frac{25}{4\epsilon^2}$, we have $l > 100$. Consider this together with the assumption that $n \leq 0.6l$, we have $q = (1-1/l)^n \geq (1-1/l)^{0.6l} > (1-1/100)^{0.6\times 100} > 0.547$.

ii) Trivially, we have $q = (1-1/l)^n > 1 - n/l$.

iii) By Lemma 14, $Var[z] < lq - (l+n)q^2$. With $q > 1 - n/l$, we have:

$$\begin{aligned} Var[z] \ & < \ lq - (l+n)q^2 \\ & = \ lq(1 - q(l+n)/l) \\ & < \ lq(1 - (1-n/l)(1+n/l) \\ & = \ lq(1 - (1 - (n/l)^2)) \\ & = \ n^2q/l \end{aligned}$$

With these inequations, we have:

$$
\begin{aligned}
& \frac{(1-(E[z]/l)^{\epsilon})E[z]}{\sqrt{Var[z]}} \\
> \; & \frac{(1-q^{\epsilon})lq}{\sqrt{n^2 q/l}} \\
= \; & (1-e^{-\ln(1/q)\epsilon}) \times \frac{l\sqrt{lq}}{n} \\
> \; & \frac{\ln(1/q)\epsilon}{1+\ln(1/q)\epsilon} \times \frac{l\sqrt{lq}}{n} \\
= \; & \frac{\epsilon n \ln(l/(l-1)) l \sqrt{lq}}{(1+\epsilon n \ln(l/(l-1)))n} \\
> \; & \frac{\epsilon\sqrt{lq}}{1+\epsilon n \ln(l/(l-1))} \\
> \; & \frac{\epsilon\sqrt{0.547l}}{1+0.25\times 0.6 \times l\ln(l/(l-1))} \\
> \; & \frac{\epsilon\sqrt{0.547l}}{1+0.25\times 0.6\times 100\ln(100/(100-1))} \\
> \; & 0.64\epsilon\sqrt{l}
\end{aligned}
$$

The second inequality above uses the fact that $1-e^{-x} > \frac{x}{1+x}$ for $x > 0$. The third inequality above is because $\ln(l/(l-1))l > 1$, which can be easily derived from the fact $e^y > 1+y$ for all $y \neq 0$ and let $y = -1/l$. The second-to-last inequality above uses the fact that for $l > 1$, $l\ln(l/(l-1))$ is a monotonically decreasing function of l and $l > 100$.

Therefore, if $l \geq \frac{25}{4\epsilon^2}$, $\frac{(1-(E[z]/l)^{\epsilon})E[z]}{\sqrt{Var[z]}} > 0.64\epsilon\sqrt{l} \geq 0.64\times\sqrt{\frac{25}{4}} = 1.6$. By lemma 15, $Pr(|\hat{n}-n| > \epsilon n) < \frac{1}{9}$. □

Lemma 17. *Consider the setting of Lemma 15 with the additional assumption that* $\epsilon < 0.25$ *and* $q \in [q_{min}, q_{max}]$*, where* q *denotes the probability that a slot is empty and* $[q_{min}, q_{max}] \subset (0, 0.6)$*. If* $l \geq max\{\frac{2.6\times(1-q_{min})}{q_{min}(1-q_{min}^{\epsilon})^2}, \frac{2.6\times(1-q_{max})}{q_{max}(1-q_{max}^{\epsilon})^2}\}$, $\Pr(|\hat{n}-n| > \epsilon n) < \frac{1}{9}$.

Proof. By Lemma 14, $E[z] = lq$ and $\sqrt{Var[z]} < \sqrt{lq(1-q)}$. Hence, $\frac{(1-(E[z]/l)^{\epsilon})E[z]}{\sqrt{Var[z]}} > \frac{(1-q^{\epsilon})lq}{\sqrt{lq(1-q)}} = \sqrt{\frac{(1-q^{\epsilon})^2 ql}{1-q}} = \sqrt{\frac{l}{g(q)}}$, where $g(q) = \frac{1-q}{(1-q^{\epsilon})^2 q}$. Therefore, if $l \geq 2.6 \times g(q)$ for all $q \in [q_{min}, q_{max}]$, $\frac{(1-(E[z]/l)^{\epsilon})E[z]}{\sqrt{Var[z]}} > \sqrt{\frac{l}{g(q)}} \geq \sqrt{2.6} > 1.6$. By lemma 15, $Pr(|\hat{n}-n| > \epsilon n) < \frac{1}{9}$.

To further simplify the form above, we now show that for $q \in [q_{min}, q_{max}]$ with $[q_{min}, q_{max}] \subset (0, 0.6)$, $g(q)$ obtains its maximum value at either q_{min} or q_{max}. To prove so, note that the derivative of $g(q)$ is $\frac{\mathrm{dg}}{\mathrm{dq}} = \frac{q^{\epsilon}(1+2\epsilon(1-q))-1}{q^2(1-q^{\epsilon})^3}$. Here the denumerator is always positive for $q \in (0, 0.6)$. We thus focus on its numerator. Let $h(q) = q^{\epsilon}(1+2\epsilon(1-q))-1$. We note that for $q \in [0.5, 0.6)$ and with $\epsilon < 0.25$, $q(1+2\epsilon(1-q))^{1/\epsilon} > 0.5\times(1+2\times 0.25\times(1-0.6))^{1/0.25} > 1$, therefore in this range, $h(q) = (q(1+2\epsilon(1-q))^{1/\epsilon})^{\epsilon} - 1 > 0$. We also note that the derivative of $h(q)$ over q is $\frac{\mathrm{dh}}{\mathrm{dq}} = \epsilon q^{\epsilon-1}((\epsilon+1)(1-2q)+\epsilon) > 0$ for $q \in (0, 0.5]$, hence in $[q_{min}, 0.5]$, $h(q)$ is monotonically increasing and can have at most one root q_r such that $h(q_r) = 0$. We consider two cases: i) if such a root q_r exists, $h(q)$ thus $\frac{\mathrm{dg}}{\mathrm{dq}}$ is negative for $q \in [q_{min}, q_r)$, and $h(q)$ thus $\frac{\mathrm{dg}}{\mathrm{dq}}$ is positive for $q \in (q_r, q_{max}]$. Hence in $[q_{min}, q_{max}]$, $g(q)$ is first monotonically decreasing then becomes monotonically increasing. The maximum value of $g(q)$ in $[q_{min}, q_{max}]$ is thus either $g(q_{min})$ or $g(q_{max})$. ii) If $h(q)$ has no root in $[q_{min}, 0.5]$, for $[q_{min}, q_{max}]$, $h(q)$ thus $\frac{\mathrm{dg}}{\mathrm{dq}}$ is always positive, hence $g(q)$ is monotonically increasing. Hence the maximum value of $g(q)$ is obtained at q_{max}. In both cases, setting $l = max(2.6\times g(q_{min}), 2.6\times g(q_{max}))$ would ensure $l \geq 2.6\times g(q)$ for all $q \in [q_{min}, q_{max}]$. □

D.2 Guarantee of Our SRC$_S$ protocol

This subsection proves Theorem 5 in our Section 7 regarding the guarantee of our SRC$_S$ protocol. See Algorithm 1 for the main steps of our SRC$_S$ protocol.

Recall that the first phase of SRC$_S$ uses LOF to obtain a rough estimate $\tilde{n}$. By the analysis of LOF [15], when the first phase of SRC$_S$ invokes LOF with 10 trials, the resulted rough estimate $\tilde{n}$ is a $(0.5, 0.1)$ estimate of n. Since each trial of LOF incurs $O(\log n)$ overhead, this first phase of SRC$_S$ uses $O(\log n)$ slots on expectation. Lemma 18 summarizes this guarantee:

Lemma 18. *The first phase of our SRC*$_S$ *outputs a* $(0.5, 0.1)$ *estimate with* $O(\log n)$ *overhead.*

Next we assume $\tilde{n} \in [0.5n, 1.5n]$ and summarize the guarantee of the second phase of SRC$_S$ under this assumption in Lemma 19:

Lemma 19. *If the second phase of SRC*$_S$ *uses* $l = \frac{65}{(1-0.04^{\epsilon})^2}$ *slots, and if it is given a rough estimate* $\tilde{n} \in [0.5n, 1.5n]$*, it outputs an* $(\epsilon, \frac{1}{9})$ *estimate for* $\epsilon < 0.25$*.*

Proof. Note that with $\epsilon < 0.25$, $l = \frac{65}{(1-0.04^{\epsilon})^2} > 200$. We consider two cases:

Case i: When $n \leq 0.6l$, recall that SRC$_S$ set $p = \min\{1, 1.6l/\tilde{n}\}$. Since $\tilde{n} \leq 1.5n \leq 1.5\times 0.6l = 0.9l$, we have $p = 1$. Also, $\frac{65}{(1-0.04^{\epsilon})^2} > \frac{65}{(ln(1/0.04)\epsilon)^2} > \frac{25}{4\epsilon^2}$. Hence, we can directly apply lemma 16 to get the estimation quality guarantee of $\hat{n}$, i.e., $\Pr(|\hat{n}-n| > \epsilon n) < \frac{1}{9}$.

Case ii: When $n > 0.6l$, we distinguish two cases:

Case ii(a): If $p = \frac{1.6l}{\tilde{n}}$, this leads to $\tilde{n} \geq 1.6l > 320$, and $q = (1-p/l)^n = (1-1.6/\tilde{n})^n$. Since $\tilde{n} \leq 1.5n$, we have $q \leq (1-1.6/(1.5n))^n < e^{-16/15} < 0.35$. Also, since $\tilde{n} \geq 0.5n$, $n \leq 2\tilde{n}$, $q \geq (1-1.6/\tilde{n})^{2\tilde{n}} > (1-1.6/320)^{2\times 320} > 0.04$.

Case ii(b): Otherwise, $p = 1$, this leads to $\tilde{n} \leq 1.6l$, therefore $n \leq 2\tilde{n} \leq 3.2l$. $q = (1-1/l)^n \geq (1-1/l)^{3.2l} > (1-1/200)^{3.2\times 200} > 0.04$. Also, since $n > 0.6l$, we have $q = (1-1/l)^n < (1-1/l)^{0.6l} < e^{-0.6} < 0.55$.

Combining these two sub cases, we have $q \in [0.04, 0.55]$. Let $g(q) = \frac{1-q}{(1-q^{\epsilon})^2 q}$ (as in the proof of Lemma 17). Applying Lemma 17, setting $l = max(2.6\times g(0.04), 2.6\times g(0.55))$ ensures $\Pr(|\hat{n}-n| > \epsilon n) < \frac{1}{9}$. Note the fact that for all $y > 0$, $\frac{y}{1+y} < 1-e^{-y} < \frac{y}{1+y/2}$. Let $y = \ln(1/q)\epsilon$, we have $\frac{1}{\ln(1/q)\epsilon} + 0.5 < \frac{1}{1-q^{\epsilon}} < \frac{1}{\ln(1/q)\epsilon} + 1$.

Using above inequalities, we have $g(0.04) = \frac{1-0.04}{(1-0.04^{\epsilon})^2 \times 0.04} > 24 \times (\frac{1}{\ln(1/0.04)\epsilon} + 0.5)^2 > (1.52/\epsilon + 2.44)^2$; while $g(0.55) = \frac{1-0.55}{(1-0.55^{\epsilon})^2 \times 0.55} < 0.8182 \times (\frac{1}{\ln(1/0.55)\epsilon} + 1)^2 < (1.52/\epsilon + 0.905)^2$. Comparing them, we have $g(0.04) > g(0.55)$. Therefore, we only need to ensure $l > 2.6 \times g(0.04)$, which holds when we set $l = \frac{65}{(1-0.04^{\epsilon})^2}$. □

We are now ready to combine Lemma 18 and Lemma 19 to prove Theorem 5, which summarizes the guarantee of our SRC$_S$ protocol.

Proof for Theorem 5. For the overall estimation quality, the probability for the SRC$_S$ protocol to output a $\hat{n}$ that satisfies $|\hat{n} - n| \le \epsilon n$ can be bounded as below:

$$\begin{aligned} & \Pr(|\hat{n} - n| \le \epsilon n) \\ \ge \quad & \Pr((|\hat{n} - n| \le \epsilon n) \bigcap (|\tilde{n} - n| \le \frac{n}{2})) \\ = \quad & \Pr((|\hat{n} - n| \le \epsilon n) \mid (|\tilde{n} - n| \le \frac{n}{2})) \times \Pr(|\tilde{n} - n| \le \frac{n}{2}) \\ \ge \quad & (1 - \frac{1}{9}) \times (1 - 0.1) = 0.8 \end{aligned}$$

Therefore overall, $\hat{n}$ is an $(\epsilon, 0.2)$ estimate of n.

For the overhead, note that by Lemma 9, the number of slots in the second phase of SRC$_S$ $l = \frac{65}{(1-0.04^{\epsilon})^2} = O(\frac{1}{\epsilon^2})$. Combining this with the overhead of its first phase (lemma 18), SRC$_S$'s total overhead is $O(\frac{1}{\epsilon^2} + \log n)$. □

D.3 Guarantee of Our SRC$_M$ protocol

This subsection proves Theorem 6 in our Section 7 regarding the guarantee of our SRC$_M$ protocol. See Algorithm 2 for the main steps of our SRC$_M$ protocol.

Recall that our SRC$_M$ protocol sees k sets, i.e., S_1, S_2, ..., S_k, sequentially and then outputs an estimate $\hat{n}$ for the size of $S_1 \cup S_2 \ldots \cup S_k$.

To reason about the estimation quality of SRC$_M$, we consider the last location. Here SRC$_M$ invokes revised PET and merges the revised PET outcomes from all locations to generate a rough estimate $\tilde{n}$. Such merging is possible since PET works under multiple-set RFID counting setting, and the only difference between our revised PET and PET is that revised PET removes the need of a user-specified upper bound for n. By the analysis of PET [23], when SRC$_M$ invokes revised PET with 30 trials in each location, the rough estimate $\tilde{n}$ is a $(0.5, 0.1)$ estimate of n. Lemma 20 summarizes this guarantee.

Lemma 20. *The first phase of SRC$_M$ outputs a $(0.5, 0.1)$ estimate at the last location.*

Next we assume $\tilde{n} \in [0.5n, 1.5n]$ and summarize the guarantee of the second phase of SRC$_M$ under this assumption in Lemma 21:

Lemma 21. *If the second phase of SRC$_M$ uses $l = \frac{205}{(1-0.013^{\epsilon})^2}$ bins and if it is given a rough estimate $\tilde{n} \in [0.5n, 1.5n]$, it outputs an $(\epsilon, \frac{1}{9})$ estimate of n for $\epsilon < 0.25$.*

Proof. Recall that SRC$_M$ outputs its final estimate $\hat{n}$ by merging the balls-and-bins outcomes of all locations when tags participate at a certain probability p, where $p = \frac{1}{2^x}$ for an integer x that minimizes $|p - \min\{1, 1.6l/\tilde{n}\}|$. The design of SRC$_M$ ensures that the required information is available at all locations, since each location starts with a probability that is no smaller than p. After the merging, a bin is occupied iff there is at least one tag responds in the bin regardless in which sets the tags appear. Hence we can consider all n tags as if they appear together in a single set.

Note that with $\epsilon < 0.25$, $l = \frac{205}{(1-0.013^{\epsilon})^2} > 450$. We consider two cases:

Case i: When $n \le 0.6l$, since $\tilde{n} \le 1.5n \le 1.5 \times 0.6l = 0.9l$, we have $p = 1$. Since $\frac{205}{(1-0.013^{\epsilon})^2} > \frac{205}{(\epsilon \ln(1/0.013))^2} > \frac{25}{4\epsilon^2}$, we can directly apply Lemma 16 to get the estimation quality guarantee of $\hat{n}$, i.e., $\Pr(|\hat{n} - n| > \epsilon n) < \frac{1}{9}$.

Case ii: When $n > 0.6l$, we distinguish two cases:

Case ii(a): If $p = \frac{1}{2^x}, x \ge 1$, this leads to $0.75p \le \frac{1.6l}{\tilde{n}} \le 1.5p$, which is equivalent to $0.75\tilde{n} \le \frac{1.6l}{p} \le 1.5\tilde{n}$ and $\frac{16}{15\tilde{n}} \le \frac{p}{l} \le \frac{32}{15\tilde{n}}$. Hence, $\tilde{n} \ge \frac{1.6l}{1.5p} \ge \frac{1.6 \times 450}{1.5 \times 0.5} = 960$. Also, with $\tilde{n} \in [0.5n, 1.5n]$, we have $\frac{32}{45n} \le \frac{p}{l} \le \frac{64}{15n}$. Consider the probability that a slot is empty, i.e, $q = (1 - \frac{p}{l})^n$, we have $q \le (1 - \frac{32}{45n})^n \le e^{32/45} < 0.492$ and $q \ge (1 - \frac{32}{15\tilde{n}})^n \ge (1 - \frac{32}{15\tilde{n}})^{2\tilde{n}} > (1 - \frac{32}{15 \times 960})^{2 \times 960} > 0.013$. Therefore we have $q \in (0.013, 0.492)$.

Case ii(b): If $p = 1$, this leads to $\frac{1.6l}{\tilde{n}} \ge 0.75$, therefore $\tilde{n} \le \frac{32l}{15}$, and $n < 2\tilde{n} \le \frac{64l}{15}$. $q = (1 - 1/l)^n \ge (1 - 1/l)^{\frac{64l}{15}} > (1 - 1/450)^{\frac{64 \times 450}{15}} > 0.013$. Also, since $n > 0.6l$, we have $q = (1 - 1/l)^n < (1 - 1/l)^{0.6l} < e^{-0.6} < 0.55$.

Combining these two sub cases, we have $q \in [0.013, 0.55]$. Let $g(q) = \frac{1-q}{(1-q^{\epsilon})^2 q}$ (as in the proof of Lemma 17). Applying Lemma 17, setting $l = max(2.6 \times g(0.013), 2.6 \times g(0.55))$ ensures $\Pr(|\hat{n} - n| > \epsilon n) < \frac{1}{9}$. Note the $\frac{1}{\ln(1/q)\epsilon} + 0.5 < \frac{1}{1-q^{\epsilon}} < \frac{1}{\ln(1/q)\epsilon} + 1$ (see proof of Lemma 19), we have $g(0.013) = \frac{1-0.013}{(1-0.013^{\epsilon})^2 \times 0.013} > 75 \times (\frac{1}{\ln(1/0.013)\epsilon} + 0.5)^2 > (1.99/\epsilon + 4.33)^2$; while $g(0.55) = \frac{1-0.55}{(1-0.55^{\epsilon})^2 \times 0.55} < 0.8182 \times (\frac{1}{\ln(1/0.55)\epsilon} + 1)^2 < (1.52/\epsilon + 0.905)^2$. Comparing them, we have $g(0.013) > g(0.55)$. Therefore, we only need to ensure $l > 2.6 \times g(0.013)$, which holds when we set $l = \frac{205}{(1-0.013^{\epsilon})^2}$. □

Before we move on to prove Theorem 6, we need to reason about the overhead of the second phase of SRC$_M$. While $l = \frac{205}{(1-0.013^{\epsilon})^2}$ bins suffices, the overhead also depends on for each bin how many different probabilities SRC$_M$ needs to check before the bin becomes empty. This depends on the probability that the second phase SRC$_M$ starts with, which in turn depends on the rough estimate from the first phase of SRC$_M$. Lemma 22 summarizes the guarantee of the first phase of SRC$_M$, and Lemma 23 below summarizes the expected number of slots needed by the second phase of SRC$_M$ given a certain starting probability. Based on them, Lemma 24 summarizes the overhead of the second phase of SRC$_M$. Note that our discussion here applies to every location (not only the last one), though we omit the index of the location to simplify the notation. Specifically, we use $\tilde{n}$ to denote the first phase output of SRC$_M$, which is a rough estimate of n, the number of tags in the union of the sets the reader has seen so far.

Lemma 22. *At any location, the output $\tilde{n}$ of the first phase of SRC$_M$ satisfies $\Pr(\tilde{n} < \frac{0.794n}{2^i}]) < 30/e^{2^i}$ for $i \geq 1$, where n is the number of tags in the union of the sets the reader has seen so far.*

Proof. Recall that to provide a $(0.5, 0.1)$ rough estimate, the first phase of SRC$_M$ invokes revised PET with 30 trials. For the jth trial ($1 \leq j \leq 30$) of revised PET, each tag chooses a positive integer according to a geometric distribution with mean of 2. The protocol then finds the maximum integer x_j chosen by all tags. For a positive integer y, $\Pr(x_j < y) = (1 - \frac{1}{2^{y-1}})^n < e^{-n/2^{y-1}}$. The revised PET then outputs $\tilde{n} = 0.794 \times 2^{(\sum_{j=1}^{30} x_j)/30}$. We thus have:

$$\begin{aligned}
& \Pr(\tilde{n} < \frac{0.794n}{2^i}) \\
= & \Pr((\sum_{j=1}^{30} x_j)/30 < \log(n) - i) \\
< & \Pr(\bigcup_{j=1}^{30} (x_j < \log(n) - i)) \\
< & 30 \times e^{-n/2^{\log(n)-i}} \\
= & 30/e^{2^i}
\end{aligned}$$

□

Lemma 23. *At any location, the second phase of SRC$_M$ with a starting probability of p for a tag to respond uses less than $max\{3l, (log(pn/l) + 4)l\}$ slots on expectation, where n is the number of tags in the union of the sets the reader has seen so far.*

Proof. Suppose n_j is the number of tags that appear in the current location, obviously, $n_j \leq n$. Now consider a single bin. We want to find out on expectation how many different probabilities SRC$_M$ needs to check before the bin becomes unoccupied. Let w_i denote the indicator random variable for the event that it is occupied when a tag responds with probability $p/2^i$ (for $i = 0, 1, \ldots$). Note that $E[w_i] = \Pr(w_i = 1) = 1 - (1 - \frac{p}{2^i l})^{n_j} \leq (1 - \frac{p}{2^i l})^n < \frac{pn}{2^i l}$. Let w denote the random variable of the number of probabilities that SRC$_M$ needs to test before the bin becomes unoccupied. $w = 1 + \sum_{i=0}^{\infty} w_i$. By linearity of expectation: $E[w] = 1 + \sum_{i=0}^{\infty} E[w_i]$. Now we discuss the following cases:

Case i: If $pn/l < 1$, we have

$$\begin{aligned}
E[w] &= 1 + \sum_{i=0}^{\infty} E[w_i] \\
&< 1 + \sum_{i=0}^{\infty} \frac{pn}{2^i l} \\
&\leq 1 + \frac{pn}{l(1 - 1/2)} < 3
\end{aligned}$$

Case ii: Otherwise, $pn/l \geq 1$, we have:

$$\begin{aligned}
E[w] &= 1 + \sum_{i=0}^{\infty} E[w_i] \\
&= 1 + \sum_{i=0}^{\lfloor \log(pn/l) \rfloor} E[w_i] + \sum_{i=\lfloor \log(pn/l) \rfloor + 1}^{\infty} E[w_i] \\
&< 1 + (1 + \lfloor \log(pn/l) \rfloor) \times 1 + \sum_{\lfloor \log(pn/l) \rfloor + 1}^{\infty} \frac{pn}{2^i l} \\
&\leq 2 + \log(pn/l) + \frac{pn}{2^{log(pn/l)} l(1 - 1/2)} \\
&\leq 2 + \log(pn/l) + 2 = \log(pn/l) + 4
\end{aligned}$$

Therefore, we have $E[w] < max\{3, log(pn/l) + 4\}$. Again, by linearity of expectation, the expected number of slots needed for all the l bins is $l \times E[w] < max\{3l, (log(pn/l) + 4)l\}$. □

Lemma 24. *At each location, the second phase of SRC$_M$ uses $O(l)$ slots on expectation, where l is the number of bins.*

Proof. Let w denote the random variable of the number of slots used by the second phase of SRC$_M$.

Let $\mathcal{E}_0$ denote the event that the rough estimate from the first phase of SRC$_M$ satisfies $\tilde{n} > \frac{0.794n}{2}$. Since the starting probability of p in the second phase of SRC$_M$ is selected to minimize $|p - \frac{1.6l}{\tilde{n}}|$, we have $\frac{3p}{4} \leq \frac{1.6l}{\tilde{n}} \leq \frac{3p}{2}$, hence $p \leq \frac{1.6 \times 4l}{3\tilde{n}} < \frac{1.6 \times 4l}{3} \times \frac{2}{0.794n}$, which leads to $\frac{pn}{l} < \frac{2 \times 6.4}{0.794 \times 3} < 5.4$. By Lemma 23, $E[w|\mathcal{E}_0] < \max\{3l, (log(5.4) + 4)l\} < 6.5l$.

Let $\mathcal{E}_i$ ($i = 1, 2, \ldots$) denote the event that the rough estimate from the first phase of SRC$_M$ satisfies $\tilde{n} \in (\frac{0.794n}{2^{i+1}}, \frac{0.794n}{2^i}]$. Similar to the above analysis, we have $p \leq \frac{1.6 \times 4l}{3\tilde{n}} < \frac{1.6 \times 4l}{3} \times \frac{2^{i+1}}{0.794n}$, which leads to $\frac{pn}{l} < \frac{2^{i+1} \times 6.4}{0.794 \times 3} < 2.7 \times 2^{i+1}$. By Lemma 23, $E[w|\mathcal{E}_i] < \max\{3l, (log(2.7 \times 2^{i+1}) + 4)l\} < (i + 6.5)l$.

From Lemma 22 we know that for $i \geq 1$, $\Pr(\mathcal{E}_i) < \Pr(\tilde{n} < \frac{0.794n}{2^i}) < 30e^{-2^i}$. Further, for $i \geq 2$, we have $\frac{30e^{-2^{i+1}}(i+1+6.5)}{30e^{-2^i}(i+6.5)} = e^{-2^i}(1 + \frac{1}{i+6.5}) < 0.021$. We can now bound $E[w]$:

$$\begin{aligned}
E[w] &= E[w|\mathcal{E}_0]\Pr(\mathcal{E}_0) + \sum_{i=1}^{\infty} E[w|\mathcal{E}_i]\Pr(\mathcal{E}_i) \\
&< E[w|\mathcal{E}_0] \times 1 + E[w|\mathcal{E}_1] \times 1 + \sum_{i=2}^{\infty} E[w|\mathcal{E}_i]\Pr(\mathcal{E}_i) \\
&< 6.5l + (1 + 6.5)l + \frac{30e^{-2^2}(2 + 6.5)l}{1 - 0.021} \\
&< 19l = O(l)
\end{aligned}$$

□

Proof for Theorem 6. Combining results from Lemma 20 and Lemma 21, the end-to-end SRC$_M$ estimation quality guarantee can be computed as:

$$\begin{aligned} & \Pr(|\hat{n}-n| \le \epsilon n) \\ \ge\ & \Pr((|\hat{n}-n| \le \epsilon n) \bigcap (|\tilde{n}-n| \le \frac{n}{2})) \\ =\ & \Pr((|\hat{n}-n| \le \epsilon n) \mid (|\tilde{n}-n| \le \frac{n}{2})) \times \Pr(|\tilde{n}-n| \le \frac{n}{2}) \\ =\ & (1-\frac{1}{9}) \times (1-0.1) = 0.8 \end{aligned}$$

Therefore, using $l = \frac{205}{(1-0.013^{\epsilon})^2}$ bins in the second phase of SRC$_M$ at each location is sufficient for achieving the required estimation quality, i.e., ensuring $\hat{n}$ is an $(\epsilon, 0.2)$ estimate of n. By Lemma 9, our chosen number of bins $l = \frac{205}{(1-0.013^{\epsilon})^2} = O(\frac{1}{\epsilon^2})$. By Lemma 24, the overhead of the second phase of SRC$_M$ at each location is $O(l) = O(\frac{1}{\epsilon^2})$. By Lemma 13, the overhead of the first phase of SRC$_M$ at the ith location is $O(\log\log n_i)$, where n_i is the number of tags in the ith set S_i. Therefore, for all the k locations together, the overhead of SRC$_M$ is $O(\sum_{i=1}^{k}(\frac{1}{\epsilon^2} + \log\log n_i))$. □

E Capability to Detect Collision

Our results can be generalized to a model where a reader can further distinguish two types of non-empty slot, i.e., *singleton slot* and *collision slot*. Exactly one tag transmits in a *singleton slot*, and at least two tags transmit in a *collision slot*.

We describe a prove sketch for our lower bound results in this generalized model, as summarized in Theorem E.

Theorem 25. *Even if the reader can detect collision, no single-set RFID counting protocol can output an* $(\epsilon, 0.2)$ *estimate with* $o(\frac{1}{\epsilon^2(\log\frac{1}{\epsilon})^2} + \log\log n)$ *overhead, for* $\epsilon \in [1/\sqrt{n}, 0.5]$.

Proof sketch. For the $o(\frac{1}{\epsilon^2(\log\frac{1}{\epsilon})^2})$ term, the proof is almost similar to the proof of Theorem 1, except that Alice and Bob need to interactively exchange $O(\log m)$ fingerprints instead of one fingerprint. Specifically, to simulate the ith slot of the single-set RFID counting protocol, Alice and Bob compute their first fingerprint in the same way as before. If their first fingerprints are identical, then they can stop by simulating an empty slot. Otherwise, Alice and Bob divide their own bit string at the middle into two substrings, and compute two fingerprints for the two substrings. If both fingerprints are different, it means there are at least two different bits between their strings, thus Alice and Bob will simulate a collision slot. Otherwise, they will further examine the substring with different fingerprint. (Note that it is impossible for the fingerprints of both substrings to be equal unless the fingerprints collide, the probability of which can be properly bounded.) If Alice and Bob's strings differ by only one single bit, recursively applying this process for $\log m$ times will reveal the position of this single bit. Then Alice and Bob can simulate a singleton slot. Using this construction and by similar argument as in the proof of Theorem 1 gives the $o(\frac{1}{\epsilon^2(\log\frac{1}{\epsilon})^2})$ term in the overhead.

For the $o(\log\log n)$ term in the overhead, the proof is similar to that of Theorem 2, and the only difference is that each slot can have 3 outcomes instead of 2.

Simply plugging Theorem 25 into our multiple-set lower bound proof for Theorem 4 (see Appendix A), one can derive a rather similar form of multiple-set lower bound for the generalized model.

F Additional Evaluation Results

F.1 Additional Results for Section 6

When analyzing the performance gain of ART and enhanced FNEB (see Section 6), we also evaluate the protocols under other settings of n. Specifically, Figure 9 and Figure 10 plot the time needed by EZB, ART, and revised ART when $n = 50,000$ and $n = 10,000$ respectively. Figure 11 and Figure 12 plot the time needed by EZB, enhanced FNEB (eFNEB for short), and the revised eFNEB for $n = 50,000$ and $n = 10,000$ respectively. As shown in the figures here (as well as Figure 2 and Figure 3 in Section 6), the relative performance of different protocols remain the same across different values of n.

F.2 Additional Results for Section 7.3

Since the overhead difference between our protocols and some existing protocols is partly due to the existence of per-trial overhead. To understand how significant this factor is, we have further compared the protocols when there is no per-trial overhead.

Figure 13 and Figure 14 plot the overhead of single-set RFID counting protocols for $n = 10,000$ and $n = 100,000$ respectively. In both settings, SRC_s continues to have the lowest overhead among all protocols. For example, when $\epsilon = 0.01$, SRCS is 20% to 100% faster than the most efficient existing protocol, i.e., ZOE.

Figure 15 plots the overhead of multiple-set RFID counting protocols. We find that SRCM continues to be 300% faster than the most efficient existing protocol.

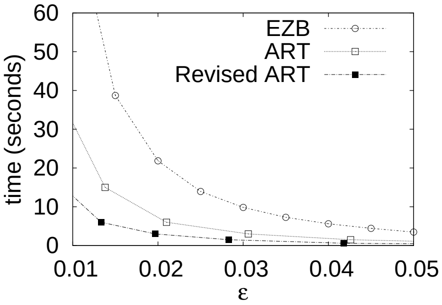

Figure 9: Time needed to achieve relative error ϵ under $\delta = 0.2$ ($n = 50,000$).

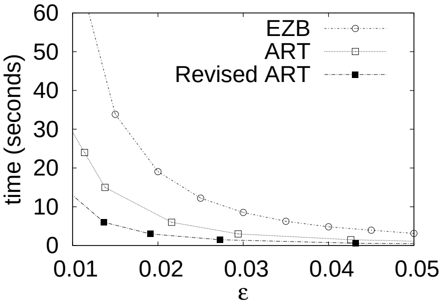

Figure 10: Time needed to achieve relative error ϵ under $\delta = 0.2$ ($n = 10,000$).

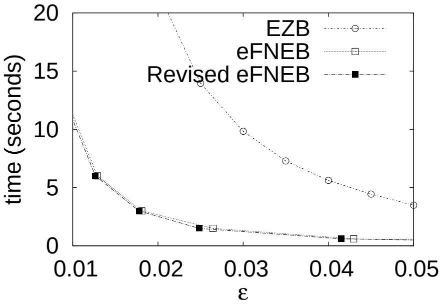

Figure 11: Time needed to achieve relative error ϵ under $\delta = 0.2$ ($n = 50,000$).

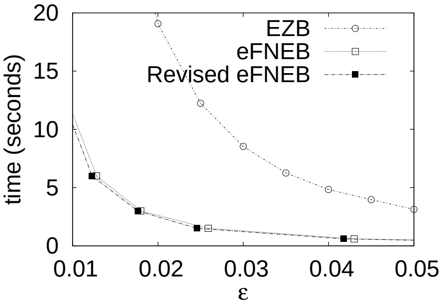

Figure 12: Time needed to achieve relative error ϵ under $\delta = 0.2$ ($n = 10,000$).

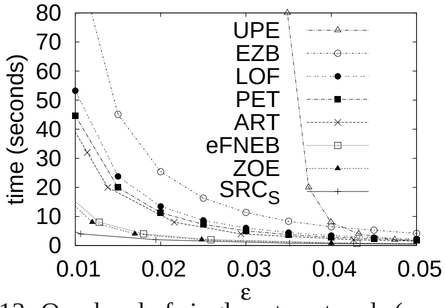

Figure 13: Overhead of single-set protocols ($n = 10,000$, without per-trial overhead).

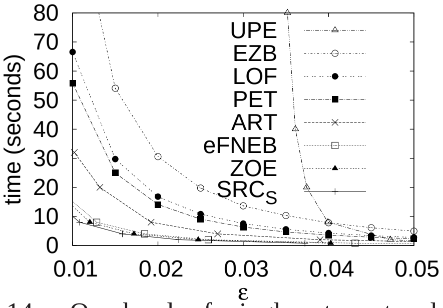

Figure 14: Overhead of single-set protocols ($n = 100,000$, without per-trial overhead).

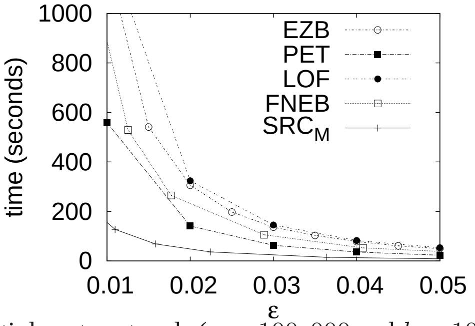

Figure 15: Overhead of multiple-set protocols ($n = 100,000$ and $k = 10$, without per-trial overhead).