RFID Counting over Time-Varying Channels

Ziling Zhou
Ampotech Pte. Ltd.
Republic of Singapore

Binbin Chen
Advanced Digital Sciences Center
Republic of Singapore

***Abstract*—For many applications that use RFID technology, it is important to count the number of RFID tags accurately. However, the wireless channel between the RFID tags and readers can introduce communication errors, and the error rate may vary significantly over time. No existing protocol can perform RFID counting robustly (i.e., maintaining the estimation quality) over time-varying channels. In this paper, we design *RRC*, a Robust RFID Counting protocol that offers provable guarantees on estimation quality over time-varying channels. Specifically, regardless of how the communication errors occur, the final output generated by RRC is always a standard (ϵ, δ) estimate of the correct count n. Furthermore, the expected amount of time needed by RRC is $O(Y + \frac{1}{\epsilon^2} + (\log\log n)^2)$ for a constant δ, where Y is the number of communication errors encountered by RRC. This makes the efficiency of RRC asymptotically near-optimal.**

I. Introduction

Radio-frequency Identification (RFID) technology involves the use of low-cost RFID tags to track physical objects. Through wireless communication with such tags, an RFID reader can gather information about the corresponding physical objects. RFID technology lies at the heart of many applications [1], from supply chain management, enhanced retailer operations, to environment monitoring. *RFID counting* is a basic functionality of RFID systems that counts (or estimates) the number of RFID tags in a certain region. It is directly used in many applications, such as product and people counting. It also serves as a building block to support other RFID functionalities, such as fast tag identification [8], [16] and popular categories identification [14]. Due to the fundamental role of RFID counting, researchers have developed many efficient RFID counting protocols over the past decade, including UPE [8], EZB [9], LOF [12], FNEB [5], PET [17], ART [13], ZOE [18], and SRC [19]. Here "efficiency" is defined in terms of the amount of time needed for RFID counting.

Since RFID counting is performed via radio communication between the tags and the reader, communication errors can happen, i.e., the information received by the reader can be different from that sent by the tags. Communication errors have been largely ignored by most of the existing RFID counting protocols (e.g., [8], [9], [12], [5], [17], [13], [19], [16]). Only a few recent ones [18], [15], [7] consider communication errors. For example, ZOE [18] estimates the communication error rate at the beginning of its execution, and then compensates the final result based on the estimated communication error rate. ZOE's approach assumes that the communication error rate is constant during its execution. The other protocols [15], [7] also make similar assumptions.

Unfortunately, wireless communication channels are notoriously unpredictable and are often plagued with bursty errors. Studies (e.g., [10], [2]) show that both movement (e.g., of surrounding objects) and wireless interference can cause sudden increase of communication error rate in RFID systems. As will be shown in Figure 1 in Section V, when the communication error rate varies over time, all existing protocols fail to achieve their claimed estimation quality. For example, when there are 2000 bursty communication errors, all existing protocols' actual estimation quality become 20 times worse than what they claim to provide. Such an observation is not surprising since these protocols either do not consider communication errors, or rely on the assumption that the communication error rate is not time-varying.

In this work, we design *RRC*, a *Robust RFID Counting Protocol* that offers provable estimation quality and near-optimal efficiency, despite of time-varying communication error rate. To the best of our knowledge, this is the first protocol with such a strong guarantee. Specifically, RRC has the following formal properties:

1) Robust against time-varying error rate. We do not make any assumption on how the communication errors occur over time. For example, there can be periods with bursty errors whose communication error rates are much higher than the average error rate during the execution of the RRC. In fact, RRC is even robust against a malicious attacker who can introduce a bounded number of errors by examining the RRC protocol before its execution and then strategically deciding which communication errors to be introduced at which time slots. Being able to achieve so serves as an ultimate attestation to the robustness of our protocol.

2) Asymptotically near-optimal efficiency. We prove that the expected amount of time needed by RRC is $O(Y + \frac{1}{\epsilon^2} + (\log\log n)^2)$. Here Y is the number of communication errors encountered by RRC during its execution, n is the correct count, and ϵ is the estimation quality (see below). A recent lower bound [19] has proved that RFID counting takes at least $\Omega(\frac{1}{\epsilon^2} + \log\log n)$ time for all possible protocols and when the communication is error-free. Under the typical setting that $\frac{1}{\epsilon^2} > (\log\log n)^2$ (for example, for $\epsilon = 0.05$, $\frac{1}{\epsilon^2} > (\log\log n)^2$ as long as $n < 10^{120}$), RRC takes $O(Y + \frac{1}{\epsilon^2})$ time and the communication-error-free lower bound is $\Omega(\frac{1}{\epsilon^2})$. This means that RRC is asymptotically optimal: Since the Y communication errors do not carry useful information and since even without error one needs $\Omega(\frac{1}{\epsilon^2})$ time, it is not possible to achieve $o(Y + \frac{1}{\epsilon^2})$ time while tolerating Y errors. We can also interpret this from another perspective: When Y is small, RRC has the same asymptotic efficiency as a protocol that cannot tolerate communication errors. As Y increases, RRC takes more time to count and the additional time scales linearly with Y. The key challenge for achieving so is that RRC does not know the value of Y in advance.

3) Guarantees on estimation quality. We prove that the final output $\hat{n}$ generated by RRC is always a standard (ϵ, δ) estimation of n, despite of the communication errors. Specifically, $\Pr[\hat{n} \in [(1-\epsilon)\times n, (1+\epsilon)\times n] \geq 1-\delta$. We only consider $\delta = \frac{1}{3}$ in the following, since to achieve a smaller δ, one can easily run RRC by $O(\log\frac{1}{\delta})$ times and takes the median value of the output results.

We achieve these strong guarantees in RRC by first designing a novel building block, called *Converging Retries (CR)*. As its name suggested, CR uses retries to deal with excessive communication errors, but it ensures that the overall false negative rate throughout the whole process does not accumulate in an unbounded manner. Next, we show that an existing Simple RFID counting (SRC) protocol [19] can be made robust, if we strategically replace some parts of that protocol with our new building block. Indeed, our RRC protocol is constructed by incorporating CR into SRC. This is quite interesting since while RRC offers substantially stronger guarantees than existing RFID counting protocols, it achieves so without introducing much complexity.

In the following, Section II formalizes the robust RFID counting problem. Section III presents *CR*, a novel building block for robust RFID counting. Section IV shows how we use CR to transform an existing RFID counting protocol into a robust one. Section V evaluates the designed protocol and we conclude in Section VI.

II. Problem Formulation and Related Work

RFID counting protocols. Consider an RFID reader and a set of RFID tags within its communication range. In an RFID counting protocol, the reader communicates with tags in synchronized time slots so as to count the number of tags (denoted by n). In each time slot, the reader sends $O(1)$ bits to tags, and a tag can either send a single bit of "1" to the reader or keep silent (i.e., send nothing). In the absence of communication errors, the reader receives nothing when all tags keep silent, and the reader receives "1" when at least one tag responds.

To avoid the high overhead of exact counting[1], prior RFID counting protocols are designed to provide an (ϵ, δ) approximate counting result $\hat{n}$ such that

$$Pr[\hat{n} \in [(1-\epsilon)\times n, (1+\epsilon)\times n]] \geq 1-\delta$$

The probability is taken over random coin flips done by the randomized protocol. Since each slot takes a constant amount of time, an RFID counting protocol's time overhead is asymptotically the same as the total number of slots it uses. Specifically, we say that the asymptotic overhead of an RFID counting protocol is $O(x)$, if for all inputs, it needs $O(x)$ slots on expectation (expectation is taken over the toss of a random coin) for achieving the accuracy target.

Time-varying channel and robust RFID Counting. When there are communication errors between the reader and the tags, the reader may receive a "1" when all tags are silent, or it may receive nothing when there is at least one tag sends "1". For a time-varying channel, the erroneous slots can distribute in an arbitrary manner over time. One can conceptualize a time-varying channel by an infinite long tape with three types of symbols: $\boxed{\mathbf{1}}$, $\boxed{\mathbf{0}}$, and $\boxed{}$. The RFID reader always receives "1" in a slot with $\boxed{\mathbf{1}}$ symbol, and always receives "nothing" in a slot with $\boxed{\mathbf{0}}$ symbol. The slot is error-free when the corresponding symbol is $\boxed{}$.

In this work, we aim to construct a robust RFID counting protocol that can continue to provide (ϵ, δ) estimation guarantee despite the *worst-case* time-varying

[1]The exact counting overhead is fundamentally high (see [19]).

channel. In other words, a robust RFID counting protocol should be able to continue to provide the guarantees on estimation quality, even if there is a malicious attacker who can examine the protocol before its execution and strategically decide which communication errors to be introduced at which time slots.

Related Work. While RFID counting protocols have been extensively studied in the literature, most efforts focus on improving the protocols' time efficiency and they do not consider communication errors. Some recent protocols, e.g. [18], [15], [7], consider communication errors, but they assume that the channel remains stable over time. Our evaluation in Section V shows that all existing protocols lose their estimation quality guarantees when the channel varies over time.

For wireless communication systems in general, there are many techniques (e.g., [11], [6], [3]) that deal with communication errors in time-varying channels. However, for RFID counting, the tags and the reader form a backscatter network, where the tags respond in a synchronized manner and their replies are "OR"-ed together to provide the input to the reader. Hence, existing techniques such as error correcting or error estimating coding schemes [3] are not directly applicable.

III. Converging Retries (CR)

To achieve robust RFID counting, we first design a building block called *Converging Retries (CR)*. A common structure in many RFID counting protocols is to have a sequence of *information slots*, where in each slot, each tag responds with some probability p. CR enables an RFID counting protocol to continue use a sequence of such information slots, despite of time-varying channels.

A. Overview and Intuition

Error-catching slots. A basic idea to achieve CR's functionality is to insert *error-catching slots* into the sequence of information slots. These slots have predetermined values (i.e., either 0 or 1). Specifically, the reader creates a 0 slot by asking all tags to remain silent, and it creates a 1 slot by sending some request that will trigger response. For a given error-catching slot, if the reader does not see the expected outcome, it knows that there must be a communication error.

CR uses the information slots and the error-catching slots in an interleaving manner, by performing a random permutation to determine the sequence of these slots. The random permutation can be done based on pseudo-random number generators, and hence the resulting positions of the error-catching slots are independent of the errors in the communication channel. In other words, one can imagine that the communication channel decides the positions of the errors first, and then CR flips coins to decide the positions of the error-catching slots. As a result, the error-catching slots will see their "fair share" of the errors.

Based on the errors observed by the error-catching slots, CR is able to estimate (with provable guarantees) the number of $\boxed{\mathbf{1}}$ and $\boxed{\mathbf{0}}$ errors in the sequence. This in turn allows CR to compensate for such communication errors occurred in the information slots. Note that this is only possible because of the random permutation. Specifically, with the random permutation, the information slots will also see their "fair share" of the errors, and hence we can use the estimation from the error-catching slots to compensate for those errors.

Retries and false negative. Intuitively, if we observe errors in a sufficiently large fraction of error-catching slots, a large fraction of the information slots will likely be erroneous as well. When this happens, the information slots may simply do not contain sufficient amount of useful information to generate a good estimate of the RFID count. In such a case, CR will retry with new slots. Conducting retries is necessary for any protocol that aims to tolerate communication errors: for example, if most information slots are erroneous, no protocol can provide a meaningful output without further retrying.

On the other hand, a key challenge is that such retries may result in uncontrolled growth of false negative rate: Whenever the protocol tries to judge whether the number of erroneous slots is excessive, there is a certain probability of having a false negative, i.e., the actual number of erroneous slots is excessive, but the protocol incorrectly believes that the number of erroneous slots is not excessive and then proceeds to produce an output. Such a false negative fundamentally comes from the randomization step (i.e., random permutation) in the protocol. As we explained earlier, the randomization step is a key step that we use to deal with time-varying distribution of communication errors.

If the protocol only needs to make the judgement once on whether it can output an estimate, it is easy to control the false negative rate. But with possible retries, the protocol may need to make the judgement for an unknown number of times, which can result in uncontrolled growth of the overall false negative rate.

For example, imagine that the number of erroneous slots is always above the threshold for proper estimation. Ideally, the protocol should never output in such a case, since it never collects enough information and whatever result it generates will be wrong. But if one designs the protocol naively, with each retry, the protocol has a small probability of having a false negative and generating an output. Eventually the protocol will always output, and the output will always be wrong!

Making the overall false negative rate converge. To overcome the challenge of uncontrolled overall false negative rate, CR uses the following simple yet effective technique. Whenever the protocol intends to make a judgement, it utilizes all information available, including the information in the current trial as well as in all previous trials. While it may be obvious that using more information helps, it is an interesting observation that by using all the information, the overall false negative rate becomes well-bounded.

More specifically, if using a trial by itself would result in a false negative rate of r, we observe (see proof later) that combing all the x trials the protocol sees so far would result in a false negative rate of r^x. Thus if the protocol does x trials and makes x judgements where each judgement is based on all trials so far, the overall false negative rate will be $r+r^2+r^3+\ldots+r^x < r/1-r$. Hence the overall false negative rate converges, despite that for each judgement the protocol introduces some extra false negative rate.

B. Detailed Design and CR's Formal Guarantees

R-Trial. Before presenting CR, we first describe a subroutine called R-Trial (Robust Trial) invoked by CR. An R-Trial consists of l slots, where:

- $\frac{4l}{5}$ slots are *information slots*. In each of them, each tag independently responds with probability p.
- $\frac{l}{5}$ slots are *error-catching slots*, which are used to estimate the number of $\boxed{\mathbf{1}}$ and $\boxed{\mathbf{0}}$ symbols on the channel tape. Specifically, in half ($\frac{l}{10}$) of them, the reader requests all tags to respond; hence, if $n > 0$ and the reader receives "nothing", it catches a $\boxed{\mathbf{0}}$ symbol on the channel tape. For the other half, the reader requests all tags to keep silent; hence, the reader catches a $\boxed{\mathbf{1}}$ symbol when it receives "1".

The two types of slots are randomly shuffled in an R-Trial. One can also choose other ratio between these two types of slots. Doing so will only affect the constant factors in our analysis, but not the asymptotic form of the protocol's overhead. Let the random variables b and w denote the number of $\boxed{\mathbf{1}}$ and $\boxed{\mathbf{0}}$ symbols caught by an R-Trial's error-catching slots. Let the random variable z denote the number of empty information slots. Since the information slots are uniformly randomly distributed in an R-Trial, we can use the fraction of slots that catch $\boxed{\mathbf{1}}$ symbols, i.e., $\frac{b}{l/10} = \frac{10b}{l}$ to estimate the fraction of $\boxed{\mathbf{1}}$ symbols in the whole R-Trial. Similarly, $\frac{10w}{l}$ is an estimate of the fraction of $\boxed{\mathbf{0}}$ symbols in the R-Trial and $\frac{l-10b-10w}{l}$ is an estimate of the fraction of $\boxed{}$ symbols.

We estimate n by examining the information slots with $\boxed{}$ symbols, the number of which can be estimated by

$$\frac{4l}{5} \times \frac{l-10b-10w}{l} = \frac{4l}{5} - 8b - 8w$$

We estimate the number of empty slots inside them by:

$$z - \frac{4l}{5} \times \frac{10w}{l} = z - 8w$$

This is because out of the z empty information slots the reader sees, on expectation $\frac{4l}{5} \times \frac{10w}{l}$ of them correspond to $\boxed{\mathbf{0}}$ symbols. Now look at the ratio of empty slots in the information slots with $\boxed{}$ symbols. Such a slot becomes empty iff all tags do not respond in it. Hence:

$$E[\frac{z-8w}{\frac{4l}{5} - 8b - 8w}] = (1-p)^n$$

With this, we can use

$$\ln(\frac{z-8w}{\frac{4l}{5} - 8b - 8w})/\ln(1-p)$$

as an estimate of n.

Converging Retries (CR). Algorithm 1 provides the pseudo code of CR. CR invokes R-Trial as a subroutine to estimate n. For that, it first needs to determine l, the length of R-Trial according to the estimation quality requirement of ϵ. In general, there are two approaches to determine l [19]. The first approach is to use a closed-form solution that is derived from mathematical analysis. The second approach is to construct a lookup table. Since the results from the mathematical approximation is not tight, we use the lookup table approach in RRC to determine l. To construct this lookup table, we run R-Trials extensively under varying error settings and a wide range of n value to find the value of l needed to achieve a "safe" result. By Lemma 3 (see later), given ϵ and a constant bound for false negative rate, it is always possible to use such an approach to find an $l = O(\frac{1}{\epsilon^2})$ that satisfies our condition.

Algorithm 1 **CR (Converging Retries)**

```
    // invoked with $p$, $\epsilon$, and a global variable $K$
1:  determine $l$ according to $\epsilon$;
2:  invoke R-Trial($p$, $l$) for $K$ times;
3:  while true do
4:      examine all slots this CR incurred so far;
5:      if error fraction in error-catching slots $> \frac{1}{8}$ then
6:          invoke one more R-Trial($p$, $l$);
7:      else
8:          return an estimate of $n$ based on all slots;
9:      end if
10: end while
11: update global variable $K \leftarrow K + 1$;
```

As shown in Algorithm 1, if CR detects a significant fraction of slots containing communication errors (line 5 - 6), it automatically retries. CR terminates when the channel becomes good enough for it to estimate n (line 8). Importantly, CR ensures that the overall false negative rate is bounded, regardless of the number of retries it conducts throughout this process. As explained earlier, CR achieves so by using all slots it sees so far (line 4).

Furthermore, our design associates CR with a global variable K. The use of K allows CR itself to be called multiple (even infinite) times while ensuring that despite of all these invocations, the overall false negative rate for all of them together is bounded. This is achieved since each call of CR increases K by 1 (line 11). This translates to a multiplicative reduction of false negative rate for the next CR invocation, making the overall false negative rates of all CR invocations bounded.

To formally describe CR's guarantees, we first characterize what kind of estimate is considered "safe", i.e., the returned estimate won't cause false negative.

Definition 1. *Given ϵ and p, let x be an estimate of n. x is "safe" to use iff:*

- $p \in [\frac{0.515}{n}, \frac{1.04}{n}] \Longrightarrow x \in [(1-\epsilon)n, (1+\epsilon)n]$;
- $p < \frac{0.515}{n} \Longrightarrow (1-p)^x > 0.55$;
- $p > \frac{1.04}{n} \Longrightarrow (1-p)^x < 0.45$.

Intuitively, when the tag participating probability p falls inside a proper range of $[\frac{0.515}{n}, \frac{1.04}{n}]$, the returned estimate x should provide the expected estimation quality. This is characterized by the statement 1 in the definition. However, CR may be invoked with bad value of p, for example, when p is too small, too few tags respond, and the fraction of empty slots $(1-p)^n$ will be too big, affecting the estimation quality of x. However, the returned estimate x will not cause false negative, if the fraction of empty slots calculated based on x is also big enough to distinguish this case (as characterized by the statement 2). Hence, the value of x itself can safely advise that x should not be used as an estimate.

Theorem 2. *Given $\epsilon < 0.25$ and K, CR returns an estimate x such that $Pr[x \text{ is safe}] \geq 1 - \frac{9}{8} \times (\frac{1}{9})^K$ and it incurs $O(y + \frac{K}{\epsilon^2})$ overhead, where y is the number of erroneous slots encountered by CR during its execution.*

To prove Theorem 2, we first introduce a Lemma that characterizes the property of a single R-Trial.

Lemma 3. *Consider an R-Trial and a target false negative rate of r. Let x be the estimate of n based on its execution, f be the error fraction in all of its slots, and f' be the error fraction in its error-catching slots. An R-Trial can ensure:*

- $f < \frac{1}{9} \Longrightarrow Pr[x \text{ is safe}] \geq 1 - r$;
- $f > \frac{1}{7} \Longrightarrow Pr[f' > \frac{1}{8}] \geq 1 - r$.
- $f \in [\frac{1}{9}, \frac{1}{7}] \Longrightarrow Pr[x \text{ is safe } \vee f' > \frac{1}{8}] \geq 1 - r$;

while incurring $O(\frac{1}{\epsilon^2}\log(\frac{1}{r}))$ overhead.

Intuitively, this Lemma shows that when the actual error fraction f is low, with high probability the returned estimate from an R-Trial is safe to use, hence the RFID counting process can make progress. When the actual error fraction is high, no estimate should be returned and this will be indicated by the increased error fraction f' in the error-catching slots with high probability. In the third case when f falls inside the two thresholds, the probability that an unsafe x is returned and yet the f' does not indicate that risk is bounded by r. The Lemma also shows that an R-Trial incurs an $O(\frac{1}{\epsilon^2}\log(\frac{1}{r}))$ of overhead in order to ensure so.

Proof Sketch of Theorem 2. By Lemma 3, one can find an $l = O(\frac{1}{\epsilon^2})$ such that for any $i = 1, 2, \dots$, an R-Trial of length $i \times l$ satisfies the three statements in Lemma 3 for $r_i = (\frac{1}{9})^i$. CR sets l to that value.

Since CR examines all slots as a single R-Trial (line 4), the first trial it examines has a length of $K \times l$. By CR's logic (line 5), it is easy to see that CR returns an unsafe result x in this loop only if the x value estimated from that R-Trial is unsafe and at the same time $f' \leq \frac{1}{8}$. Hence, by Lemma 3, regardless of the channel condition (as reflected by f), if CR returns an x here, $Pr[x \text{ is safe}] \geq 1 - r_K > 1 - \frac{9}{8} \times (\frac{1}{9})^K$. If CR does not return in the first loop, it examines an R-Trial with length of $(K+1) \times l$ in the second loop. Again, by Lemma 3, if CR returns an x at this loop, the false negative rate for this trial is bounded by $r_{K+1} = (\frac{1}{9})^{K+1}$. Taking a union bound of both cases (i.e., returning at first loop and returning at the second loop), we still have: $Pr[x \text{ is safe}] \geq 1 - r_K - r_{K+1} > 1 - \frac{9}{8} \times (\frac{1}{9})^K$. Following the same argument, regardless of how many loops CR incurs before it returns an x, the overall false negative rate is bounded by $\sum_{i=K}^{\infty} r_i = \frac{9}{8} \times (\frac{1}{9})^K$, hence $Pr[x \text{ is safe}] \geq 1 - \frac{9}{8} \times (\frac{1}{9})^K$.

We now show that CR's overhead is $O(y + \frac{K}{\epsilon^2})$. With a global variable K, CR incurs $K \times l$ slots even when there is no communication error. Since $l = O(\frac{1}{\epsilon^2})$ (see above), this translates to an $O(\frac{K}{\epsilon^2})$ overhead. Suppose CR stops on the ith loop, where $i > 1$. the condition at line 5 of Algorithm 1 must hold for the $i-1$th iteration. Hence, $\frac{y}{(i-1) \times \frac{l}{5}} > \frac{1}{8}$. Hence, the total number of slots CR uses, i.e., $i \times l = O(y)$. Combining these two results, the overhead of CR is $O(y + \frac{K}{\epsilon^2})$. $\square$

IV. Robust RFID Counting (RRC) Protocol

A key observation of our work is that one can simply incorporate the building block CR into an existing non-robust RFID protocol and transform the latter into a robust protocol. It is quite interesting since our transformation shows that one can make an RFID counting protocol robust while retaining all the good properties the non-robust protocol has — including strong guarantees on estimation quality, high efficiency, and design simplicity.

A. SRC: a Simple Non-robust Protocol

We first briefly describe SRC, a simple non-robust RFID counting protocol from [19][2]. SRC consists of two phases. Its first phase generates a rough estimate, based on which its second phase outputs the final estimate.

To bound the overhead of its first phase at $O(\log\log n)$, SRC calls a *revised PET protocol*. The original PET protocol [17] generates an estimate of n by conducting a sequence of independent trials. In one trial, each tag randomly chooses a positive integer u with probability of $(\frac{1}{2})^u$. Given an upper bound x on n, PET conducts a binary search over $[1, \log x]$ to find the largest value of u that has been chosen by at least one tag. Intuitively, with more tags, that value increases monotonically on expectation. Hence, one can base on that value to estimate n. If the upper bound x is within a constant polynomial of n, PET's binary search incurs $O(\log\log x) = O(\log\log n)$ overhead in each trial. Since SRC only needs to get a rough estimate in its first phase, it uses a constant number of PET trials. To release the assumption that a good upper bound x is given in advance, [19] uses some extra "bound-searching" slots before PET. Specifically, in the ith bound-searching slot, tags with $u \geq 2^{i-1}$ respond. The bound-searching stops on the first slot that is empty. Denote that slot's index by m. The original PET's binary search will be invoked over $[1, 2^{m-1}]$. It is easy to show that on expectation $m = O(\log\log n)$. Hence, the revised PET still incurs an overhead of $O(\log\log n)$.

After that, the second phase of SRC conducts a single Balls-into-Bins trial consisting of $O(\frac{1}{\epsilon^2})$ slots, where each tag participates by independently choosing a slot in the trial and responding with a probability p. Here, the rough estimate $\tilde{n}$ from the first phase is used to decide the value of p so that the expected number of responding tags is in the same order as the length of the trial. Hence, the outcome of the trial will show a healthy mixture of empty and occupied slots. The second phase of SRC then uses the fraction of empty slot in the trial to calculate the final estimate of n, which provides the (ϵ, δ) guarantee on the estimation quality.

B. Transforming SRC to RRC using CR

SRC uses multiple randomized trials in order to obtain its final results, and its estimation quality guarantee is based on the assumption that all of these trials are executed without experiencing any communication error. If communication errors affect the outcomes of some trials, those trials lose their original stochastic properties, which in turn leads to the ultimate loss of the end-to-end guarantee of the SRC protocol.

Having CR as a convenient building block, we can replace SRC's non-robust trials with CRs to construct a Robust RFID Counting (RRC) protocol. Algorithm 2 and Algorithm 3 together provide the pseudo code for our RRC protocol. If we ignore the difference in their building blocks, RRC follows similar design of SRC: It consists of two phases. The first phase is to find $\tilde{n}$, a rough $(\frac{1}{3}, \frac{9}{64})$ estimate of n. If $\tilde{n} \in [\frac{2n}{3}, \frac{4n}{3}]$, the second phase can return an $(\epsilon, \frac{3}{16})$ estimate $\hat{n}$. Since $\frac{9}{64} + \frac{3}{16} < \frac{1}{3}$, by applying union bound over the two phases, we can see that $\hat{n}$ is an $(\epsilon, \frac{1}{3})$ estimate of n.

Same as SRC, RRC's first phase starts with a "bound-searching" stage (line 3 - 6 in Algorithm 2), until the identified upper bound x happens to be a good rough estimate of n (line 7 - 8) or $x > n$ (line 9). In the latter case, a binary search like PET (Algorithm 3) is

[2]The protocol is called SRC_S in [19].

Algorithm 2 RRC (Robust RFID Counting) Protocol

```
    // First phase:
1:  $K \leftarrow 1$; $i \leftarrow 1$;
2:  while value of $\tilde{n}$ unassigned do
3:      $p \leftarrow (\frac{2}{3})^i$;
4:      $x \leftarrow$ CR($p$, $\frac{1}{3}$);
5:      if $(1-p)^x < 0.45$ then
6:          $i \leftarrow i \times 2$;
7:      else if $(1-p)^x \in [0.45, 0.55]$ then
8:          $\tilde{n} \leftarrow x$;
9:      else
10:         $\tilde{n} \leftarrow$ Binary-Search($\frac{i}{2}, i$);
11:     end if
12: end while
    // Second phase:
13: reset $K \leftarrow 1$;
14: $\hat{n} \leftarrow$ CR($0.69/\tilde{n}$, $\epsilon$);
15: return $\hat{n}$;
```

Algorithm 3 Binary-Search(low, $high$)

```
1:  while true do
2:      $m \leftarrow \frac{low+high}{2}$;
3:      $x \leftarrow$ CR($(\frac{2}{3})^m$, $\frac{1}{3}$);
4:      if $(1-(\frac{2}{3})^m)^x \in [0.45, 0.55] \vee low \geq high$ then
5:          return $x$;
6:      else if $(1-(\frac{2}{3})^m)^x < 0.45$ then
7:          $low \leftarrow m+1$;
8:      else
9:          $high \leftarrow m-1$;
10:     end if
11: end while
```

conducted to find the rough estimate (line 10). Note that to replace all the trials invoked in the first phase, CR needs to be invoked multiple times (both during the bound-searching and the binary-search stage). Since the value of K is initialized in the beginning of the first phase and it is not reset until the second phase, the converging property of CR (as formalized in Theorem 2) nicely guarantees that all the first phase CR invocations together incur a bounded probability of having a false negative (i.e., returning an "unsafe" result).

After the rough estimate $\tilde{n}$ is found, the second phase of RRC simply replaces SRC's Balls-into-Bins trial with a CR. We reset the value of K here since RRC reserves separate false negative budget for the second phase. For the communication-error-free case, RRC further improves its efficiency using an early termination technique. Specifically, RRC outputs $\hat{n}$ immediately after it examines l' information slots, if its error-catching slots catch no error at that moment. Here l' is the number of information slots needed to provide an $(\epsilon, \frac{3}{16})$ estimate as in the second phase of the original SRC protocol.

C. RRC's Formal Guarantees

Theorem 4 formalizes the RRC protocol's guarantee.

Theorem 4. *For $\epsilon < 0.25$, RRC outputs an $(\epsilon, \frac{1}{3})$ estimate of n and incurs $O(Y + \frac{1}{\epsilon^2} + (\log\log n)^2)$ overhead, where Y is the number of erroneous slots experienced by RRC during its execution.*

Proof Sketch of Theorem 4. We first prove RRC's estimation quality guarantee under the condition that all results from the CRs are safe. Since $\frac{0.515}{n} < \frac{2}{3} \times \frac{1.04}{n}$, there must exist some i^* such that $(\frac{2}{3})^{i^*} \in [\frac{0.515}{n}, \frac{1.04}{n}]$. Given that all CRs invoked by RRC return safe results, based on Definition 1 and RRC's pseudo code, the first phase of RRC will assign the value of $\tilde{n}$ using the x returned by an CR invoked with $p = (\frac{2}{3})^{i^*}$. By Definition 1 and Theorem 2, $\tilde{n} \in [\frac{2n}{3}, \frac{4n}{3}]$. Hence the $p = \frac{0.69}{\tilde{n}}$ value used in the second phase (line 14 in Algorithm 2) satisfies $p \in [\frac{0.515}{n}, \frac{1.04}{n}]$. By statement 1 in Definition 1, the final output $\hat{n} \in [(1-\epsilon)n, (1+\epsilon)n]$.

Now we analyze the overall false negative rate that any of the CRs may return an unsafe result. For the first phase, applying Theorem 2 and union bounds, the overall false negative rate is bounded by $\sum_{i=1}^{\infty} \frac{9}{8} \times (\frac{1}{9})^i \leq \frac{9}{8} \times \frac{\frac{1}{9}}{1-\frac{1}{9}} = \frac{9}{64}$. For the second phase, the false negative rate is bounded by $\frac{9}{8} \times \frac{1}{9} = \frac{1}{8}$. With early termination, false negative rate of the second phase is still bounded by $\frac{3}{16}$. Since $\frac{9}{64} + \frac{3}{16} < \frac{1}{3}$, by union bound, the overall false negative rate across both phases is bounded by $\frac{1}{3}$. Hence, $\hat{n} \in [(1-\epsilon)n, (1+\epsilon)n]$ holds with probability greater than $1 - \frac{1}{3}$.

Now we analyze the overhead. For the first phase, following the same proof as SRC [19], on expectation, there will be $m = O(\log\log n)$ CR invocations. Note that we are pursuing a rough estimate with $\epsilon = \frac{1}{3}$ here. Hence, by Theorem 2, the ith invocation of CR incurs an overhead of $O(y_i + i)$. All CRs invoked in the first phase hence incur an expected overhead of $\sum_{i=1}^{m} O(y_i + i) = O(Y_{\text{phase 1}} + \sum_{i=1}^{m} i)$. Since $m = O(\log\log n)$, this translates to an overhead of $O(Y_{\text{phase 1}} + (\log\log n)^2)$. For the second phase, since it only invokes CR once with $K = 1$, by a direct application of Theorem 2, the second phase overhead is $O(Y_{\text{phase 2}} + \frac{1}{\epsilon^2})$. Hence, the overall overhead of RRC is $O(Y + \frac{1}{\epsilon^2} + (\log\log n)^2)$, where $Y = Y_{\text{phase 1}} + Y_{\text{phase 2}}$ is the total number of communication errors encountered by RRC. □

V. Evaluation

We conduct simulations to study the robustness, estimation quality guarantees, and efficiency of RRC protocol under different communication error settings. Table I lists some settings that we evaluate. Based on EPCglobal C1G2 standard [4], each RFID counting slot is 0.4ms long. For experiments presented in this Section, there are $n = 50000$ tags and we consider an (ϵ, δ) estimation quality requirement with a constant $\delta = \frac{1}{3}$. We also conduct experiments with other values of n and δ, and observe consistent results under all of these settings.

Robustness and estimation quality guarantee. Our formal analysis proves the robustness and estimation quality guarantee of RRC, which we demonstrate here by examining the estimation quality provided by RRC over various types of error settings. We have tested RRC's robustness in more than a dozen of different error settings, where we vary the types of errors — $\boxed{\mathbf{1}}$ errors, $\boxed{\mathbf{0}}$ errors, or a mixture of them; the way they distribute over time — uniform, bursty, and settings with an increasing fraction of errors happening in the first phase. We also try different combinations of them.

Under all of these settings, RRC always provides the promised $(\epsilon, \frac{1}{3})$ error estimation quality. Figure 1 illustrates RRC's robustness by presenting its behaviour under the settings 1-3 in Table I. Here the target estimation quality is $\epsilon = 3\%$. As shown in the Figure, RRC's actual estimation error never exceeds that threshold in all settings, regardless of the number of errors. RRC achieves this strong guarantee, thanks to the convergence of the overall false negative rate as provided by the CR building block. Though not shown in the Figure, we also evaluate the settings 5-8 in Table I where an increasing number of errors are diverted to the first phase of RRC. Since the first phase needs to invoke CR multiple times, the false negative in any invocation can make all the rest invocations meaningless. Even in these settings, RRC delivers its guaranteed estimation quality.

As a comparison, we also evaluate the actual estimation quality provided by other existing RFID protocols under the same settings. Only a few recent protocols [18], [15], [7] explicitly consider the impact of communication errors. We evaluate ZOE [18] in our experiment, since ZOE's model conforms with the standard RFID counting model (see Section II)[3]. ZOE explicitly deals with communication errors, but it makes an implicit assumption that the error rate remains stable over its execution. To avoid any unfair comparison with RRC, we prepend ZOE with 100% extra free slots to estimate the channel error rate before it gets started. In addition to ZOE, we also compare with the SRC protocol [19]. As described in Section IV-A, SRC incurs the lowest overhead among existing RFID protocols, but it is designed without considering communication error.

[3][7] assumes that the protocol can use additional physical layer information, and [15] considers tag-specific channel condition.

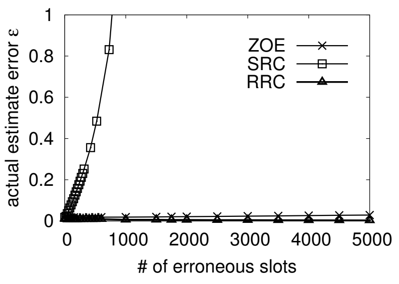
	Channel error type	Distribution pattern
Setting 1	100% $\boxed{\mathbf{1}}$	Uniformly distributed
Setting 2	100% $\boxed{\mathbf{1}}$	Bursty
Setting 3	100% $\boxed{\mathbf{0}}$	Bursty
Setting 4	50% $\boxed{\mathbf{1}}$, 50% $\boxed{\mathbf{0}}$	Bursty
Setting 5	100% $\boxed{\mathbf{1}}$	25% phase 1
Setting 6	100% $\boxed{\mathbf{1}}$	50% phase 1
Setting 7	100% $\boxed{\mathbf{1}}$	75% phase 1
Setting 8	100% $\boxed{\mathbf{1}}$	100% phase 1

TABLE I: Selected communication error settings

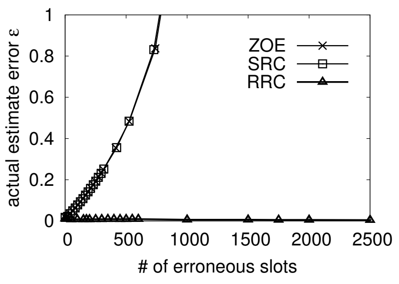
As shown in Figure 1, for all the three settings presented, SRC loses its estimation quality guarantee rapidly when the number of communication errors increases. This is not surprising since SRC is not designed with robustness in mind. Also as expected, ZOE works reasonably well in setting 1, where the channel errors distribute uniformly over time. However, it performs as bad as the non-robust SRC when the channel error rate varies over time (setting 2 and 3). In particular, both protocols lose their guarantees on estimation quality when operating over time-varying channels. For example, with 2000 bursty $\boxed{\mathbf{0}}$ slots (setting 3), the results from both of them have more than 60% of relative estimation error, which is 20 times worse than what they claim to provide (i.e., the target error of $\epsilon = 3\%$).

Efficiency of RRC. In all the settings we evaluated, the overhead of RRC increases linearly with the number of communication errors it encounters. For example, RRC's overhead increases by around 10 seconds if it encounters 2500 communication errors and increases by around 20 seconds if it encounters 5000 errors. This confirms our analysis that asymptotically RRC provides near-optimal overhead of $O(Y + \frac{1}{\epsilon^2} + (\log\log n)^2)$, where Y is the number of communication errors encountered by RRC. Note that, if the fraction of error slots always remains above the level RRC can tolerate, RRC may not terminate. In this case, both Y and the RRC's overhead become unbounded.

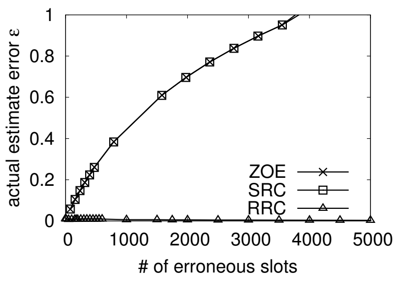
As another indicator of RRC's efficiency, we also look at its overhead when the channel is error-free. In this



(a) Setting 1.



(b) Setting 2.



(c) Setting 3.

Fig. 1: Actual estimation error ϵ obtained by different protocols under different communication error settings.

setting, RRC uses less than 2 seconds to provide an estimate with $\epsilon = 3\%$ estimation quality. In comparison, even the most efficient non-robust protocol, i.e., the SRC protocol, requires around 1.5 second to achieve the same estimation quality. RRC only incurs small increase of overhead when the channel is error-free. Asymptotically, while RRC's first phase incurs $O((\log\log n)^2)$ overhead, which is greater than the $O(\log\log n)$ overhead at SRC's first phase, the overhead of its second phase remains at $O(\frac{1}{\epsilon^2})$ as SRC. Since $O(\frac{1}{\epsilon^2})$ dominates $O((\log\log n)^2)$ (unless n approaches the astronomically large number of 10^{120}), SRC and RRC's overall overheads are in the same order of $O(\frac{1}{\epsilon^2})$ in a communication-error-free setting.

VI. Conclusion

In this paper, we study the RFID counting problem over time-varying channels. We propose a novel building block called Converging Retries (CR), and use CR to design a Robust RFID Counting (RRC) protocol, which provides strong formal properties, including robustness against time-varying error rates, asymptotically near-optimal efficiency, and guarantees on estimation quality.

Acknowledgment

The work and presentation are partly supported by the research grant for the Human-Centered Cyber-physical Systems Programme at the Advanced Digital Sciences Center from Singapore's Agency for Science, Technology and Research (A*STAR), and partly supported by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme. This work was partly done when Ziling Zhou was a research intern at Advanced Digital Sciences Center. We thank Prof. Haifeng Yu from National University of Singapore for his insightful inputs to the paper.

References

[1] http://www.rfidjournal.com/internet-of-things.

[2] A. Bekkali, S. Zou, A. Kadri, and R. Penty. Performance analysis of passive uhf rfid systems under cascaded fading channels and interference effects. *IEEE Transactions on Wireless Communications*, 14(3), March 2015.

[3] B. Chen, Z. Zhou, Y. Zhao, and H. Yu. Efficient Error Estimating Coding: Feasibility and Applications. *IEEE/ACM Transactions on Networking (ToN)*, 20(1):29–44, February 2012.

[4] 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.

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

[6] F. Hlawatsch and G. Matz. *Wireless Communications Over Rapidly Time-Varying Channels*. Academic Press, 2011.

[7] Y. Hou, J. Ou, Y. Zheng, and M. Li. PLACE: Physical layer cardinality estimation for large-scale rfid systems. In *INFOCOM*, 2015.

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

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

[10] J. Mitsugi. UHF Band RFID Readability and Fading Measurements in Practical Propagation Environment, Auto-ID Labs White Paper, 2005.

[11] E. Newcombe and S. Pasupathy. Error rate monitoring for digital communications. *Proceedings of the IEEE*, 70(8):805–828, 1982.

[12] 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.

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

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

[15] W. Sze, Y. Deng, W. Lau, M. Kodialam, T. Nandagopal, and O. Yue. Channel-oblivious counting algorithms for large-scale rfid systems. *IEEE Transactions on Parallel and Distributed Systems*, 26(12):3303 – 3316, December 2015.

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

[17] 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.

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

[19] Z. Zhou, B. Chen, and H. Yu. Understanding RFID Counting Protocols. *IEEE/ACM Transactions on Networking (ToN)*, 24(1):312–327, February 2016.