2015 44th International Conference on Parallel Processing

Towards Constant-Time Cardinality Estimation for Large-Scale RFID Systems

Binbin Li*†, Yuan He‡, Wenyuan Liu†§

*School of Economics and Management, YanShan University, China
†School of Information Science and Engineering, YanShan University, China
‡School of Software, TNLIST, Tsinghua University, China
§The Key Laboratory for Computer Virtual Technology and System Integration of HeBei Province, China
ysulbb@gmail.com, he@greenorbs.com, wyliu@ysu.edu.cn

***Abstract*—Cardinality estimation is the process to survey the quantity of tags in a RFID system. Generally, the cardinality is estimated by exchanging information between reader(s) and tags. To ensure the time efficiency and accuracy of estimation, numerous probability-based approaches have been proposed, most of which follow a similar way of minimizing the number of required time slots from tags to reader. The overall execution time of the estimator, however, is not necessarily minimized. The estimation accuracy of those approaches also largely depends on the repeated rounds, leading to a dilemma of choosing efficiency or accuracy. In this paper, we propose BFCE, a Bloom Filter based Cardinality Estimator, which only needs a constant number of time slots to meet desired estimation accuracy, regardless of the actual tag cardinality. The overall communication overhead is also significantly cut down, as the reader only needs to broadcast a constant number of messages for parameter setting. Results from extensive simulations under various tagIDs distributions shows that BFCE is accurate and highly efficient. In terms of the overall execution time, BFCE is 30 times faster than ZOE and 2 times faster than SRC in average, the two state-of-the-arts estimation approaches.**

***Keywords*—RFID; Cardinality Estimation; Bloom Filter;**

I. INTRODUCTION

Radio Frequency IDentification (RFID) systems have been becoming important platforms for a variety of applications, such as access control [1][2], object identification [3], inventory management [4], transportation and logistics [5], localization [6][7][8], and tracking [9][10][11][12]. Researches on RFID have received wide interests from both industrial and academical communities. Typically, a RFID system consists of a large volume of tags, one or multiple readers and a back-end server that stores the information of tags.

As mentioned in the literature, the problem of cardinality estimation is of primary importance in many RFID systems and applications. Because it is infeasible to get the exact count of all tags in a very short time, most existing works in this area follow the way of probabilistic estimation, such as PET [13], ZOE [14], SRC [15] and A^3 [16], etc.. Taking time efficiency as the first-place performance metrics, the state-of-the-art approaches can make the estimation in $\mathcal{O}(\log\log n + \frac{1}{\varepsilon^2})$ time slots, where n denotes the actual number of tags and ε denotes the confidence interval of estimation requirement.

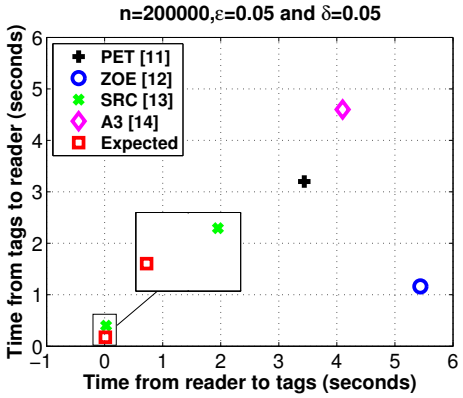

Fig. 1. The design space of RFID cardinality estimation.

An important fact, however, is often neglected. The number of time slots for estimation does not necessarily determine the total time of cardinality estimation. The temporal overhead of every communication between reader and tags, is usually a more impacting factor of the estimation efficiency. Taking ZOE [14] as an example, each frame in ZOE only contains one time slot, while ZOE totally requires at least $m = [\frac{c\sigma(x)_{max}}{e^{-\lambda}(1-e^{\varepsilon\lambda})}]^2$ slots to get the final estimation result, where (ε, δ) are the estimation accuracy requirement, c is a constant determined by the error probability of the accuracy requirement, $\sigma(x)_{max} = 0.5$ and λ is a parameter associated with the actual cardinality. Note that in ZOE, the reader needs to broadcast a 32-bits random seed in each slot, so that each tag can determine whether or not to participate in the current slot. The temporal overhead of communication from reader to tags (m×32), rather than the overhead of communication from tags to reader (m×1), is accounted as the major component in the overall execution time of ZOE. Other approaches like SRC[15] only require a limited number of estimation rounds, but their estimation accuracy is largely determined by the number of time slots. That leads to a dilemma of choose efficiency or accuracy, while in practice people often desire to ensure both these two performance metrics. Figure 1 presents the design space of RFID cardinality estimation, comparing our expected result with the existing works.

0190-3918/15 $31.00 © 2015 IEEE
DOI 10.1109/ICPP.2015.90

Based on the above discussion, one may immediately raise a crucial but open problem: For an arbitrary set of tags, is there any way to estimate its cardinality in constant time and simultaneously achieve a desired estimation accuracy? In this paper, we propose Bloom Filter based Cardinality Estimator (BFCE) to efficiently estimate the tag cardinality of large-scale RFID systems. BFCE consists of two phases, i.e. a rough estimation phase to get the lower bound of cardinality and an accurate estimation phase to get the exact result. In each phase, BFCE lets all the tags construct a w-bits Bloom Filter vector B in a distributed manner, using k independent hash functions and a persistence probability p. Intuitively, the number of 0s or 1s in B must have relationship with the tag cardinality. By modeling the relationship between the tag cardinality n and the ratio of 0s or 1s in B, BFCE can estimate the tag cardinality with very low communication overhead. Specifically, the first phase of BFCE uses a specific p to get a rough lower bound estimation of n (denoted by $\widehat{n_{low}}$), while in the second phase, BFCE employs the optimal p with $\widehat{n_{low}}$, so that the final estimation result can meet the accuracy requirement.

Our contributions can be summarized as follows.

- For the first time in the community, we propose a constant-time estimator BFCE for cardinality estimation in large-scale RFID systems. With constant number of $1024 + 8192$ bit-slots in just one round, BFCE can get the final estimation, while the temporal overhead is constant.
- Compared with the existing approaches, BFCE significantly reduces the overall temporal overhead between reader and tags, and requires no complicated computation or extra storage on RFID tags.
- We carry out extensive simulations to compare BFCE with the state-of-the-arts approaches under different settings and tagIDs distributions. The results demonstrate the advantages of BFCE in terms of time efficiency and estimation accuracy.

The rest of the paper is organized as follows. We discuss related works in Section II. In Section III, we present the system model and introduce basic concepts of tag cardinality estimation. In Section IV we elaborate on the design and analysis of BFCE. Section V presents extensive simulations to evaluate the performance of BFCE, and comparison results with recent related works. We conclude this work in Section VI.

II. RELATED WORK

As the number of tags may be up to hundreds of thousands and there are always collisions at the reader side, it is infeasible to identify all the tags one by one for the purpose of cardinality estimation. A series of probabilistic approaches have been proposed to achieve the approximate tag cardinality efficiently.

M. Kodialam *et al.* propose the cardinality estimation scheme UPE in [17], which needs to distinguish the slots to empty, single or collision slots, and utilizes the number of empty or collision slots in the frame to get the estimation. In [18], M. Kodialam *et al.* propose another enhanced Framed-Slotted-Aloha based estimator EZB, which takes the average number of zeros in the frame as clues for estimation. In [19], C. Qian *et al.* propose LOF, which employs the geometric distribution hash functions to itemize all tags in order, so as to make estimation quickly. H. Han *et al.* propose another tag estimation scheme called First Non Empty Based estimator (FNEB) [20], which is based on the size of the first run of 0s in the frame. With the goal of minimizing power consumption of active tags, T. Li *et al.* [21] propose an estimation scheme called Maximum Likelihood Estimator (MLE) for active tags. In [22], V. Shah-Mansouri *et al.* propose a multi-reader tag estimation scheme, but it is based on an unrealistic assumption that any tag covered by multiple readers only replies to one among them. Shahzad *et al.* [23] propose Average Run based Tag estimation (ART), which uses the average run size of 1s to estimate the tag cardinality.

On the basis of a Probabilistic Estimating Tree (PET), Zheng *et al.* [13] further improve the estimation efficiency to $O(\log \log n)$ time slots. In [14], Zheng *et al.* propose another efficient estimate scheme Zero-One-Estimator (ZOE), which only needs $O(\log \log n)$ time slots. B. Chen *et al.* [15] establish strong lower bounds for both the single-set and multiple-set problems. They also design new protocols SRC that is more time-efficient than existing schemes. In [16], W. Gong *et al.* propose a new mechanism, Arbitrarily Accurate Approximation (A^3), to reliably estimate the number of tags with arbitrary accuracy requirement.

Limitations of the existing approaches mainly lie in the following aspects. First and foremost, it does not guarantee to minimize the overall time for estimation, when one simply takes the number of required slots as indicator of time efficiency. Second, some existing works require prior knowledge of the rough magnitude of cardinality, so that they can reasonably tune the parameter settings for accurate estimation. Last but not least, the accuracy of existing estimators largely depends on the number of repeated rounds, and can not get the accurate estimation within a controllable length of time. The above problems either hurt the efficiency and accuracy of cardinality estimation, or limit the usability of those schemes in practical scenarios. Such facts motivate our work in this paper.

III. PRELIMINARIES

A. System Model

Consider a large-scale RFID systems which consists of a large volume of tags, one or multiple readers and a back-end server. And we only consider the cases of large-scale tags, e.g. there are more than 1000 tags in the RFID systems, as it is easy and fast to get the exact number of tags by using traditional identification protocols when the cardinality is small. Each tag is assigned an unique identification (tagID), with capacity of simple computations and communications with reader through RF signal. We assume that all the tags are within the communication range of readers, and all the readers are connected to the back-end server via Ethernet. The back-end server can coordinate and synchronize all the readers, so if multiple readers are deployed, these readers can be logically considered as one reader [14].

The communication model between the reader and tags is *Reader-Talks-First* and time-slotted, which follows the EPC-global C1G2 standard [24]. To enhance the efficiency of RFID systems, a number of parallel protocols[25][26] have been

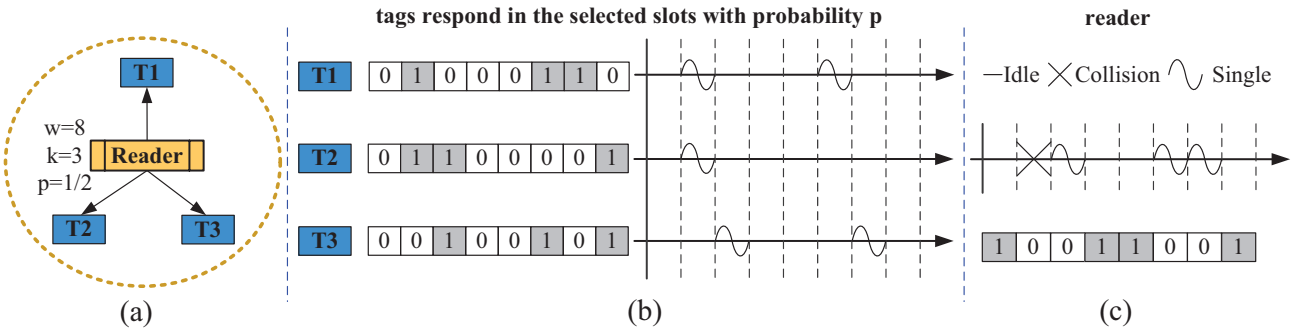

Fig. 2. A simple example of BFCE. (a) The reader first broadcasts w, k, p and random seeds R. (b) Each tag responds in the k selected bit-slots with probability p. (c) The reader senses the physical channel in all the w bit-slots, so as to get a vector B and does the estimation with B.

proposed in recent years. In those parallel protocols, tags are allowed to transmit short information (such as 1bit) in the same slot, the reader only needs to sense the physical channel and distinguish the slots to busy or idle. If there is a busy channel, the reader gets one bit '1'. Otherwise, it gets one bit '0'. For presentation clarity, we call such a slot as *bit-slot*.

We adopt the *bit-slot* mode in BFCE. As described in Figure 2, the reader initializes the communication by sending out a request message (e.g., estimate), together with a series of parameters, such as the length of Bloom Filter w, the number of hash functions k, k random seeds R and a persistence probability p. Once receiving the estimation request, each tag uses k independent hash functions to randomly pick k bit-slots, and responds with probability p in each selected bit-slot. The reader then only needs to sense the physical channel and does the estimation with B, which represents the status of all the w bit-slots. The clocks of tags are synchronized by the reader's signal. The communication channel is assumed to be perfect (without channel error) in our work.

B. Problem Description

We formally describe the tag cardinality estimation problem as follows: Given an accuracy requirement of (ε,δ)-approximation, we expect an estimation result $\hat{n}$, which satisfies $Pr\{|\hat{n} - n| \leq \varepsilon n\} \geq 1 - \delta$. For example, if there are actually 500000 tags in the whole system, with (ε = 5%, δ = 5%) approximation, we expect to get an estimation result within the interval [475000,525000] with a probability of 95% or above.

For tag cardinality estimation, there are several essential principles. First, the estimation accuracy must be guaranteed. Second, since the communication range of a reader can be up to 30 feet. The number of tags in the range may easily exceed tens of thousands. Hence the estimation scheme should be time-efficient and scalable as much as possible. Moreover, the temporal overhead between reader and tags, which is often ignored in prior literature, should be minimized.

IV. Bloom Filter based Cardinality Estimation

In this section, we introduce the detail of BFCE. Table I summarizes the symbols used across this paper.

TABLE I.
SYMBOLS USED IN THE PAPER

Symbols	Descriptions
n	Actual number of tags
$\hat{n}$	Final estimated number of tags
$\widehat{n_r}$	A rough estimation of n
$\widehat{n_{low}}$	A rough lower bound of n
ε	Confidence interval
δ	Error probability
$H(\cdot)$	Uniform hash functions
R	Random seeds set and $\|R\| = k$
B	Bloom Filter vector at reader side
w	Length of vector B
k	Number of hash functions
p	Persistence probability
p_s	Specific persistence probability
p_o	Optimal persistence probability
X	A bernoulli random variable
X_i	The observation of ith slot in B
$\overline{p}$	The ratio of 1s in B
c	A constant coefficient used for rough estimation
d	A constant determined by δ

A. Overview

To get an accurate estimation of tag cardinality in the reader's communication range, the reader first tries to construct a bitmap which can well reflect the actual cardinality, and then does the estimation with this bitmap. Specifically, the reader first constructs a w-bit Bloom Filter vector B. All the bits in B are initialized to be 0s. Then the reader sends out the estimation command, together with several parameters, such as the length of bloom filter vector w, k random seeds R ($k = |R|$), and a persistence probability p. The reader then waits for the response from tags in the following w bit-slots.

Once receiving the estimation command, each tag randomly selects k different bit-slots with k independent hash functions, whose value ranges in [1,w] and follows an uniform distribution. Then the tag transmits a short signal (e.g. 1 bit) with a probability p in each selected bit-slot.

For the arbitrary ith slot, the reader only needs to sense the physical channel. If the channel is idle, it means there is no tag participating in this bit-slot. Correspondingly $B(i)$ is set to 1. And if the channel is busy, which indicates that at least one tag transmits in this bit-slot, so $B(i)$ is set to 0. After w bit-slots, the reader can get a w-bits vector B, which is filled with 0s and 1s.

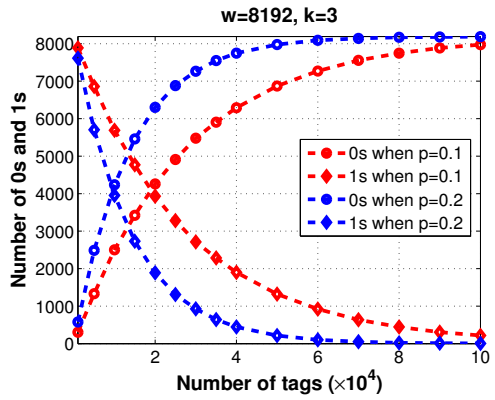

Fig. 3. The feasibility of $BFCE$.

Intuitively, the number of 0s or 1s in B is associated with all the parameters as aforementioned. Without distinguishing the difference of hash functions and the random seeds, the number of 0s or 1s is only associated with the number of tags n, the length of bloom filter vector w, the number of hash functions k, and the response probability p. Figure 3 shows the interrelation between n and the numbers of 0s and 1s in B, when we fix w=8192, k=3 and set p=0.1, p=0.2 respectively. From the figure, we can see that there is linear relationship between the number of tags and the number of 0s or 1s in the vector B. Intuitively it is feasible to accurately estimate the number of tags $\hat{n}$ according to the w-bits bitmap B, k and p. The concrete design will be introduced in the rest of this section.

B. Generic Algorithm

Assuming that all hash functions follow uniform distribution, the probability of the arbitrary ith bit in B being 0 or 1 can be calculated using Theorem 1.

Theorem 1: Let n be the actual tag cardinality, w be the length of Bloom Filter vector, k be the number of hash functions and p be the persistence probability (i.e., the probability for a tag participates in each selected bit-slot), then,

$$Pr\{B(i) = 1 | i \in [1, w]\} = e^{-\lambda}, \quad (1)$$

$$Pr\{B(i) = 0 | i \in [1, w]\} = 1 - e^{-\lambda}, \quad (2)$$

where $\lambda = \frac{kpn}{w}$.

Proof: Because all hash functions follow uniform distribution in the range $[1, w]$, the probability that a hash function of arbitrary tag being the ith bit-slot in B is

$$Pr\{H(\cdot) = i | i \in [1, w]\} = \frac{1}{w}.$$

With the persistence probability p, the probability for the tag responds in the ith bit-slot is $\frac{p}{w}$, and the probability for the tag doesn't respond in ith slot is $1 - \frac{p}{w}$.

In B, the probability of the arbitrary bit i (where $i \in [1, w]$) being 1 is

$$Pr\{B(i) = 1\} = (1 - \frac{p}{w})^{kn},$$

which means all the kn hash functions of n tags have not selected the ith bit-slot.

Using the approximation of

$$\lim_{x \to \infty} (1 - \frac{1}{x})^x = e^{-1},$$

the above equation can be simplified as

$$Pr\{B(i) = 1\} = (1 - \frac{p}{w})^{kn} \approx e^{-\frac{kpn}{w}} = e^{-\lambda},$$

where $\lambda = \frac{kpn}{w}$.

Correspondingly, the probability of $B(i)$ being 0, which means more than one tag respond in the ith bit-slot, can be calculated by

$$Pr\{B(i) = 0\} = 1 - Pr\{B(i) = 1\} \approx 1 - e^{-\lambda}.$$

■

We define a random variable X which takes value 1 with probability $Pr\{B(i) = 1\} \approx e^{-\lambda}$ and takes value 0 with probability $Pr\{B(i) = 0\} \approx 1 - e^{-\lambda}$. Then we have

$$Pr\{X = 1\} = e^{-\lambda}, Pr\{X = 0\} = 1 - e^{-\lambda}.$$

It is not hard to get that the random variable X follows the Bernoulli distribution. Therefore, the expectation and the standard deviation of X are as follows:

$$E(X) = e^{-\lambda}, \sigma(X) = \sqrt{Var(X)} = \sqrt{e^{-\lambda}(1 - e^{-\lambda})}.$$

Theorem 2: Let $\overline{\rho} = \frac{1}{w}\sum_1^w X(i)$ be the average of w independent observations, where $X(i)$ denotes the ith observation of random variable X. Then the tag cardinality can be calculated by

$$\hat{n} = -\frac{w \ln \overline{\rho}}{kp}. \quad (3)$$

Proof: Assuming that all the trials of X_i $(1 \le i \le w)$ are independent, we have $E(\overline{\rho}) = E(X)$ and $\sigma(\overline{\rho}) = \frac{\sigma(X)}{\sqrt{w}}$.

According to $thelawoflargenumbers$, when w is large enough we have

$$\overline{\rho} = E(\overline{\rho}) = E(X) = e^{-\lambda}.$$

So we can estimate λ as follows,

$$\hat{\lambda} = -\ln \overline{\rho}.$$

The observation of $\overline{\rho}$ can be used to estimate the tag cardinality as follows,

$$\hat{n} = -\frac{w \ln \overline{\rho}}{kp}.$$

■

From Equation 3, we can get that w, k, p and $\overline{\rho}$ all influence the estimation accuracy of $\hat{n}$. Particularly, the estimator will not work when $\overline{\rho}$ =0 or $\overline{\rho}$=1, which means that all bits in vector B are identical (0s or 1s). They are the two exceptions we should to avoid.

The parameter k in Equation 3 whose value denotes the number of hash functions is introduced to cope with the various

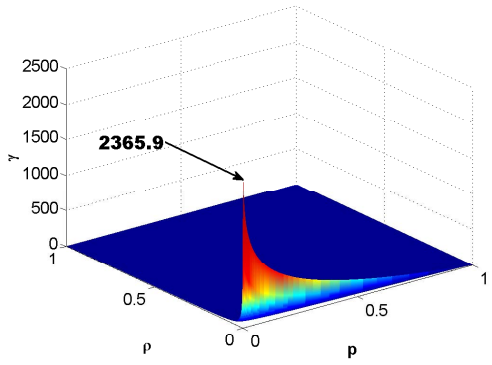

Fig. 4. The variation of $\gamma = -\frac{\ln \overline{\rho}}{3p}$, when both persistence probability p and the ration of 1s ρ are varied in the range (0,1).

distribution of tagIDs. It can't be too small. A small k will lead to a great variance of $\hat{n}$ because of the pseudo-random of hash functions. On the other hand, it is also time-consuming for tags to get k random numbers when k is large. The communication overhead between reader and tags will also be increased, as the reader needs to broadcast more random seeds to all the tags. Taking all these factors into consideration, we empirically set k=3 in BFCE for a reasonable tradeoff between overhead and accuracy.

When it comes to w, similar situations occur. A large w will cause both the exception of all 1s in B and high temporal overhead for BFCE, while a small w will also cause another exception of all 0s in B. Besides, we should also take the scalability of $BFCE$ into account when determining the value of w. We define $\gamma = -\frac{\ln \overline{\rho}}{kp}$, where k is set to 3. As both p and $\overline{\rho}$ vary in the range (0,1), we can get the variation of γ with different p and $\overline{\rho}$, as depicted in Figure 4. Then we can find 0.000326$\leq$ γ $\leq$2365.9. According to Equation 3, we get 0.000326$\times w$ $\leq$ $\hat{n}$ $\leq$2365.9$\times w$. That is to say, the value of w actually bounds the scalability of the estimator. In our work, to achieve a constant-time estimator, we set w=8192, which is scalable enough for most RFID systems. Under this setting, the maximum cardinality that the estimator can estimate exceeds 19 millions, which is sufficient for almost all kinds of application scenarios.

The reasons why w is set at 8192 is two fold. First, the cardinality of tags in a practical scenario is not infinitely large. Setting w to an appropriate value enables one to simultaneously obtain good scalability of the estimator and sufficient capacity to accommodate all the tags in a reader's communication range. Second, the overhead of implementing hash functions should also be considered. We may control this overhead by adopting a reasonable length of the Bloom filter, namely w.

Algorithm 1 regulates the behavior of the RFID reader. Depending on which phase the estimator is in, the reader either gets a specific persistence probability for rough estimation of the lower bound of cardinality, or calculates the approximate optimal persistence probability p for accurate estimation (line 4). The reader initiates the estimation process by sending out w, R and p (line 5). After that, the reader senses the channel

Algorithm 1 BFCE algorithm for reader

1: $B = 0$ //8192bits
2: $w \leftarrow$8192
3: $k = 3$
4: Get p for rough or accurate estimation
5: Initiate the estimation, broadcast w, k, R and p
6: **for** $i \leftarrow 1$ to w **do**
7: **if** there is no response in the slot **then**
8: $B(i) \leftarrow 1$
9: **else**
10: $B(i) \leftarrow 0$
11: **end if**
12: **end for**
13: $\overline{\rho} \leftarrow \frac{1}{w}\sum_1^w B(i)$
14: **return** $\hat{n} \leftarrow -\frac{w \ln \overline{\rho}}{kp}$

Algorithm 2 BFCE algorithm for RFID tag

1: Receive w, k, R, p
2: $S \leftarrow H_j(tagID, R_j, w)$, $j = 1 \cdots k$
3: **for** $i \leftarrow 1$ to w **do**
4: $S \leftarrow S - 1$
5: **if** any s in S equals 0 **then**
6: Respond with a probability of p immediately
7: **else**
8: Keep silent
10: **end if**
11: **end for**

and records the status into the vector B (line 6-12). The ratio of 1s $\overline{\rho}$ in B is calculated after all the w bit-slots end (line 13). Finally, the estimation of tag cardinality is calculated using Equation 3 (line 14).

Each tag performs simple tasks as regulated in Algorithm 2. In each estimation phase, when receiving an estimation command, the tag computes the selected S bit-slots with k different hash functions (line 2). If any s in S equals 0, the tag sends a response with a probability of p. Otherwise, it keeps silent (line 3-11).

C. Rough Lower Bound Estimation Phase

Before performing the final (ε,δ) estimation, we first try to get a rough lower bound of tag cardinality (denoted by $\widehat{n_{low}}$). Nevertheless, since we don't have any prior knowledge about the tag cardinality n, we turn to get a rough estimation (denoted by $\widehat{n_r}$) of n firstly. According to Equation 3, as long as the ratio $\overline{\rho} \neq 0$ and $\overline{\rho} \neq 1$, we can get an estimation of n. We set a specific persistence probability p_s=$\frac{2^3}{2^{10}}$, and observe the received Xs in the coming 32 bit-slots. If all the 32 slots are idle slots, which means there are no response in all slots, we adjust the response probability p_s to $p_s + \frac{2}{2^{10}}$. On the contrary, if all the 32 bit-slots are busy slots, which indicates the probability p_s is too large for the current cardinality, we reduce it to $p_s - \frac{1}{2^{10}}$. This procedure is immediately terminated once both idle and busy slots appear in the 32 bit-slots.

Through several tests, we can get a valid persistence probability p_s quickly. With this p_s, BFCE starts a new round to get $\widehat{n_r}$ according to Algorithm 1. As we only expect a rough lower bound of n, rather than the actual n, we can

terminate the estimation at any time (e.g. after 1024 bit-slots). The feasibility of using only 1024 trials of Xs to get the rough estimation is that we assume all the hash functions follow uniform distribution. So the $E(\overline{\rho})$ of 1024 trials theoretically equals to the $E(\overline{\rho})$ of 8192 trials. That is to say, the $\overline{\rho}$ of 1024 bit-slots could approximately represents the $\overline{\rho}$ of 8192 slots. However, there may be difference between $\widehat{n_r}$ and n. So we take $c * \widehat{n_r}$ as the rough lower bound, namely

$$\widehat{n_{low}} = c * \widehat{n_r}$$

where the value of c ranges in $[0.1 \cdots 0.9]$, and we also empirically set $c = 0.5$ in BFCE which can guarantee $\widehat{n_{low}} \leq n$ hold in most cases as validated in Section V.

D. Final Accurate Estimation Phase

Different from previous literature, which require numerous rounds to get an approximate $\overline{\rho}$, we tune the value of p to get an accurate estimation of $\overline{\rho}$ in just one round, and then get the cardinality estimation result that meets the accuracy requirement $Pr\{|\hat{n} - n| \leq \varepsilon n\} \geq 1 - \delta$. Next, We will show how to get an approximate optimal p with the rough lower bound estimation $\widehat{n_{low}}$.

Theorem 3: Given the accuracy requirement of (ε,δ), $\hat{n}$ is an (ε,δ) estimation of n if

$$f_1 \leq -d \ and \ f_2 \geq d, \tag{5}$$

where $f_1 = \frac{e^{-\lambda(1+\varepsilon)} - e^{-\lambda}}{\frac{\sigma(X)}{\sqrt{w}}}$, $f_2 = \frac{e^{-\lambda(1-\varepsilon)} - e^{-\lambda}}{\frac{\sigma(X)}{\sqrt{w}}}$, and $d = \sqrt{2} erfinv(1-\delta)$.

Proof: Because $\lambda = \frac{kpn}{w}$, according to Equation 3, the estimation accuracy requirement can be represented by

$$Pr\{e^{-\lambda(1+\varepsilon)} \leq \overline{\rho} \leq e^{-\lambda(1-\varepsilon)}\} \geq 1 - \delta. \tag{6}$$

Based on the fact that the variance of $\overline{\rho}$ is reduced if the experiment is repeated for many times (e.g. w=8,192 times), we define a random variable $Y = \frac{\overline{\rho} - \mu}{\sigma}$, where $\mu = E(\overline{\rho}) = e^{-\lambda}$, and $\sigma = \sigma(\overline{\rho}) = \frac{\sigma(X)}{\sqrt{w}}$. Thus, Equation 6 becomes

$$Pr\{f_1 \leq Y \leq f_2\} \geq 1 - \delta, \tag{7}$$

By the$centrallimittheorem$, we know Y is approximately a standard normal random variable. Given a particular error probability δ, we can find a constant d that satisfies

$$Pr\{-d \leq Y \leq d\} = 1 - \delta. \tag{8}$$

Combining Equations (7) and (8), one can guarantee the accuracy requirement $Pr\{|\hat{n} - n| \leq \varepsilon n\} \geq 1 - \delta$ if the following conditions are satisfied:

$$f_1 \leq -d \ and \ f_2 \geq d.$$

■

BFCE takes the minimal p that satisfies Equations 5 as the optimal p_o, so we can guarantee that $\hat{n}$ is a (ε,δ) estimation of n. Therefore, the optimal p_o is usually small (e.g. $p = \frac{3}{2^{10}}$), especially when n is large. However, it is impossible to get the optimal p_o by solving Equation 5, as the actual value of n is unknown. To get the optimal value for p_o, we first

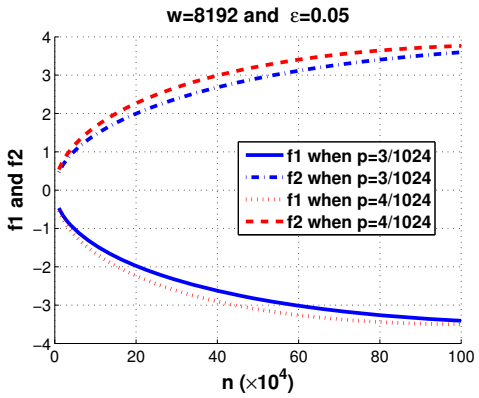

Fig. 5. The monotonicity of f_1 and f_2 when the persistence probability p is small.

take both f_1 and f_2 as functions of n. As shown in Figure 5, when p_o is small, given $w = 8192$, $k = 3$ and the confidence interval $\varepsilon = 0.05$, f_1 and f_2 are monotonically decreasing and increasing functions of n, respectively. So we have the following Theorem.

Theorem 4: Let $\widehat{n_{low}}$ be a rough lower bound estimation of n, which has been obtained in the previous estimation phase, i.e., $\widehat{n_{low}} \leq n$. let p_o be the minimal probability that satisfies

$$f_1(\widehat{n_{low}}) \leq -d \ and \ f_2(\widehat{n_{low}}) \geq d. \tag{9}$$

Equation 5 holds when using this p_o.

Proof: Because f_1, f_2 are monotonically decreasing and increasing functions of n, respectively, we have

$$f_1(n) \leq f_1(\widehat{n_{low}}) \ and \ f_2(n) \geq f_2(\widehat{n_{low}}). \tag{10}$$

Combining Equations 9 and 10, Equation 5 holds. ■

Based on the above analysis, we get the approximate optimal p_o via brute-force calculation (from $\frac{1}{2^{10}}$ to $\frac{2^{10}-1}{2^{10}}$) with priori knowledge of $\widehat{n_{low}}$ (which has been obtained in the first rough estimation phase). We take the the minimal p_o that satisfies Equation 9 as the approximate optimal persistence probability, since p_o is usually small. With this p_o, we can guarantee the result calculated with Equation 3 is an (ε,δ) estimation.

E. Analysis and Discussion

1) Overhead Analysis: As we mention in the previous sections, BFCE finishes estimation in just one round, which consists of two phases, namely a rough estimation phase and an accurate estimation phase. In each phase, the reader only needs to transmit constant number of parameters, and then senses the physical channel to get a bloom vector B within 8192 bit-slots. In the rough estimation phase, as we only expect a rough lower bound of the tag cardinality, we may terminate the phase in just 1024 bit-slots. Hence the temporal overhead of this phase, denoted by t_1, is calculated by

$$t_1 = (l_w + l_k + k * l_R + l_p) * t_{r \to t} + t_{int} + 1024 * t_{t \to r},$$

where l_w, l_k, l_R, and l_p are the length of w, k, random seeds, and p, respectively $t_{r \to t}$ is the time for the reader to

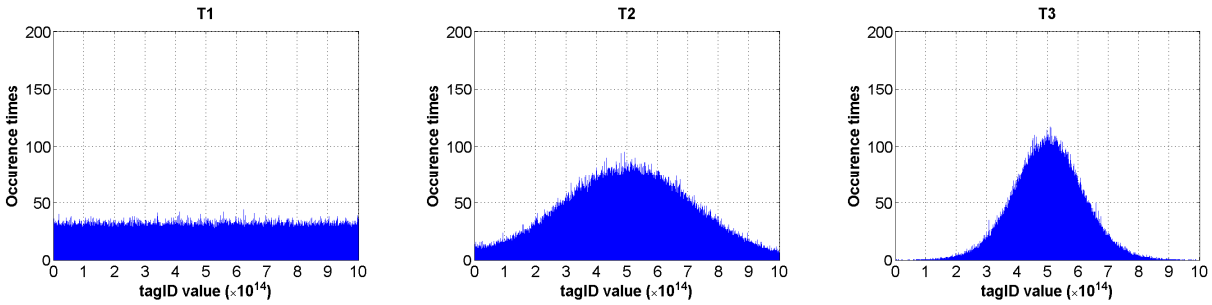

Fig. 6. Three tagIDs sets used in the simulation under different distribution.

transmit 1-bit information, t_{int} is the time interval between two consecutive transmissions from the reader to tags or vice versa, and $t_{t\to r}$ is the time for the tags to transmit 1-bit information. Since both w=8192 and k=3 are constant, we can preload them to tags and need not transmit them at runtime. Therefore, t_1 can be simplified as

$$t_1 = (3 * l_R + l_p) * t_{r\to t} + t_{int} + 1024 * t_{t\to r}.$$

Similarly, the temporal overhead of the accurate estimation phase, denoted by t_2, is calculated by

$$t_2 = t_{int} + (3 * l_R + l_p) * t_{r\to t} + t_{int} + 8192 * t_{t\to r}.$$

Based on the above analysis, the overall temporal overhead of BFCE (denoted by t) is the summation of t_1 and t_2, which is

$$\begin{aligned} t &= t_1 + t_2 \\ &= (6 * l_R + 2 * l_p) * t_{r\to t} + 3 * t_{int} + 9216 * t_{t\to r}. \end{aligned}$$

According to the EPCglobal C1G2 standard[24], the time for a reader to transmit one-bit information is 37.76μs. The time interval is 302μs. The time for tags to transmit one-bit information is 18.88μs, namely $t_{r\to t}$=37.76μs, t_{int}=302μs and $t_{t\to r}$=18.88μs. If we restrict the lengths of both the random seeds l_R and the persistence probability l_p to be 32 bits, the overall temporal overhead of BFCE is less than 0.19s. It means that BFCE can rapidly get the final accurate estimation within constant-time, regardless the actual tag cardinality and the estimation accuracy requirement.

2) Implementation of the hash functions: In BFCE, all the tags are required to select $k = 3$ bit-slots using hash functions and respond in the selected slots with a probability of p. Instead of storing many hash functions on resource-constrained tags, a 32-bits random number (denoted by RN in the binary form) is prestored on each tag, prior to the RFID system deployment. To implement the hash functions, the reader generates three uniformly distributed random seeds (denoted by $RS[i]$ in the binary form, where $i \in 1, 2, 3$) at the very start of each phase and broadcasts them to all the tags. When receiving the random seeds, each tag computes the three hash values by

$$H(id) = bitget(RN \oplus RS(i), 13 : 1),$$

where $\oplus$ denotes the bitwise XOR operation and $bitget$ is a function to get the lowest 13 bits of the XOR results. Such a simple method only requires the tags to perform lightweight bitwise XOR computation and $bitget$ operations to get the hash values.

3) Setting the persistence probability: Then, most existing works implement the persistence probability p by virtually extending frame size for $\frac{1}{p}$ times, i.e., the reader announces a frame size of w/p and terminates the frame after the first w slots. This scheme seems not usable in BFCE, because the value of p is usually small. The size of virtual vector after being extended will be large and slows down the hash function related computations. Instead, we let the reader broadcast the numerator of p (denoted by p_n) rather than the actual p. On receiving p_n, each tag randomly selects 10 bits from the pre-stored random number. If the selected value (in the decimal form) turns out to be smaller than p_n-1, the tag will respond in current bit-slot. Otherwise, the tag will keep silent. In this way, we also get a lightweight p-persistence. All the conclusions and theorems proved before still hold under this setting.

V. Performance Evaluation

We conduct extensive simulations under various tagIDs distributions to evaluate the performance of BFCE. First, we assess the estimation accuracy of BFCE with varied cardinalities of tags under different settings. We then compare BFCE with two typical state-of-the-arts approaches, ZOE[14] and SRC[15], in terms of estimate accuracy and time efficiency.

A. Setup and Metrics

We first generate three tagID sets following different distribution as the input data for our simulations. As shown in Figure 6, the first set (denoted by T1) follows uniform distribution between 1 and 10^{15}. The second tag sets(denoted by T2) follows an approximate normal distribution. And the third tag sets (denoted by T3) follows normal distribution.

Instead of repeating hundreds rounds of estimation and taking the average as the final outputs in previous approaches, we just take the result of one round estimation as the final result. We adopt a relative metric to evaluate the accuracy, namely

$$Accuracy = \frac{|\hat{n} - n|}{n},$$

where $\hat{n}$ denotes the estimation result and n refers to the actual number of tags. A good estimator is expected to return an estimation result close to the actual value. The closer it is to 0, the higher the estimation accuracy is.

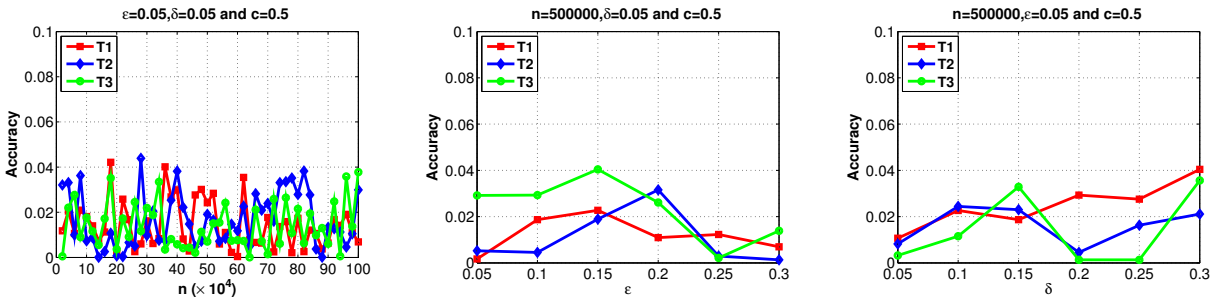

Fig. 7. Estimation accuracy with different n, ε and δ under different tagIDs distribution.

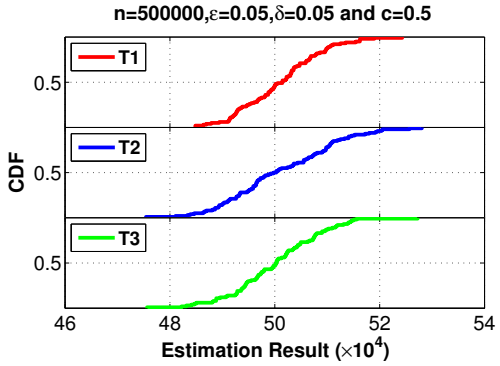

Fig. 8. Cumulative distribution of BFCE under different tagIDs distribution.

To evaluate the time efficiency of different estimators, we take the overall execution time of estimators as the second metric, the execution time of BFCE is the total time paid for the communication between reader and tags. According to EPCglobal C1G2 standard[24], any two consecutive transmission from the reader to tags or vice versa are separated by a waiting time of 302μs. The transmission rate from the reader to tags is 26.5Kb/s. It takes 37.76μs to transmit 1 bit. Assuming that the length of a random seed is 32bits, it totally takes 1,510μs for the reader to broadcast a 32-bits random seed. The rate from a tag to the reader is 53Kb/s, it takes 18.88μs for a tag to transmit 1 bit. So the time for tags to transmit l bits signal is approximately 18.88$\times l$+302μs.

B. Performance under Different Settings

We first examine the accuracy of BFCE with different parameters settings under all the three tagIDs distributions. Figure 7(a) presents the different estimation accuracy to get an (0.05,0.05) estimation under different actual cardinality n . The results under all the three distributions are shown together. Recall that $c = 0.5$ is the constant coefficient used in the rough lower bound estimation phase. From this figure, we can see that the accuracies are very close to 0 regardless of the actual tag cardinality, and always can meet the desired accuracy requirement in all cases. This group of experiments reveal that different tagIDs distributions have little impact on the estimation accuracy.

Then we fix the actual tag cardinality n=500000, and evaluate the estimation accuracy with different ε and δ. Figure 7(b) plots the accuracy when ε is varied from 0.05 to 0.3 and other parameters are fixed. Whatever ε is, BFCE always achieves estimation accuracy below 0.04, which is far better than the required ε. We see similar results when δ is varied from 0.05 to 0.3 under all the T1, T2 and T3 distributions as shown in Figure 7(c).

To further validate the stability of BFCE, we run the BFCE for 100 rounds when n=500000, ε=0.05 and δ=0.05. Figure 8 presents the cumulative distribution of the estimation results in T1, T2 and T3 respectively. According to the simulation results, we find that the estimation results of BFCE are tightly concentrated around the actual cardinality under all the three tagIDs distributions. It means that BFCE offers more accurate estimation after multiple runs. Compared with previous approaches which need to be executed hundreds of repeated rounds, we can achieve an extremely accurate estimation in no more than 100 rounds.

C. Comparison

We compare the performance of BFCE with two typical state-of-the-art schemes, ZOE[14] and SRC[15]. Note that ZOE requires a rough estimation of n as input to get the final accurate estimation, we slightly modify ZOE, and add a rough estimation phase to ZOE. For simplicity, we invoke LOF [19] and run it for 10 rounds. We then use LOFs output as the rough estimation input of ZOE. To achieve an (ε, δ) estimation with SRC where δ is smaller than 0.2, we repeat the second phase of SRC for m rounds, where m is the smallest integer that satisfies $\sum_{i=(m+1)/2}^{m}\binom{m}{i} \times 0.8^i \times 0.2^{m-i} \geq 1-\delta$.

we conduct performance comparison with all the three tagIDs distributions. Due to the page limit, Figure 9 and Figure 10 present the accuracy and execution time in only one distribution (T2). As shown in Figure 9, both ZOE and SRC can achieve the desired estimation in almost all the case except several exceptions. Specifically, when n=50000, the accuracy requirement is set to ε=0.05 and δ=0.05, SRC gets a final estimation 53430, and the accuracy is about 0.68. Given n=500000, ε=0.05 and δ=0.3, ZOE outputs an estimation result 537656, which also exceeds the desired confidence interval. The reason for the exceptions of ZOE and SRC is as follows. The estimation results of ZOE and SRC largely depends on the accuracy of rough estimation, namely the output results of the first estimation phase in ZOE and SRC. In contrast, BFCE

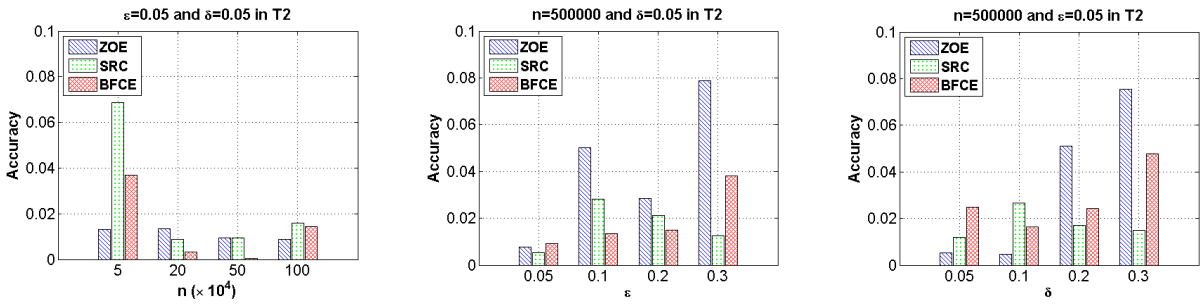

Fig. 9. The comparison of accuracy with different n, ε and δ in one of tagIDs set T2.

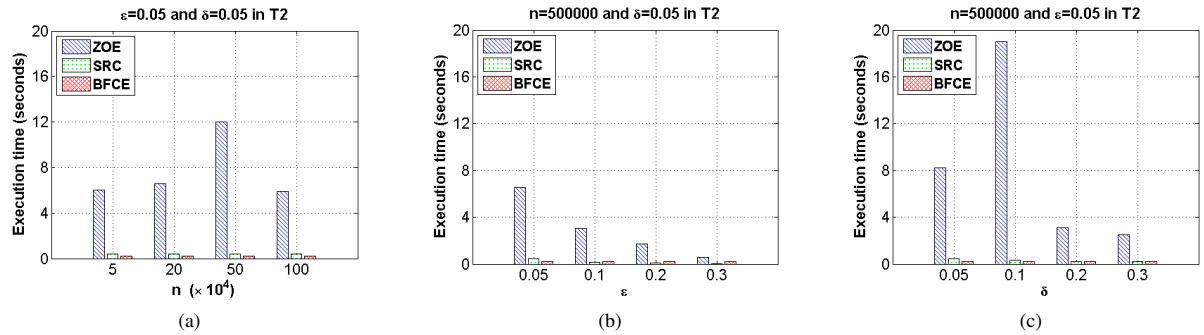

Fig. 10. The comparison of overall execution time with different n, ε and δ in one of tagIDs set T2.

always can achieve the desired accuracy in all the cases in only one round, because BFCE's final estimation is only concerned with the rough lower bound of cardinality, rather than an exact value of roughly estimated cardinality.

In Figure 10, we examine the overall execution time of BFCE, compared with that of ZOE and SRC with different parameters settings in the distribution T2. We can see from the figures that the execution time of ZOE is usually large, about several seconds in all the cases, and even goes up to 18s in the worst case. There are two reasons for the poor performance of ZOE. First, ZOE needs to continually broadcast 32-bits random seeds for each slot, so the communication time from the reader to tags accounts for the major portion of execution time. Second, the number of required slots of ZOE has great relationship with the output of the rough estimation phase. An estimation that fairly deviates from the actual cardinality will lead to a sharp growth of the required time slots. Although the overall execution time of SRC is much shorter than ZOE's, there are still apparent variance because the execution time of SRC also has relationship with the accuracy of rough estimation. In comparison, BFCE always gets the desired estimation in a constant time, within just 0.19s, which is 30 times faster than ZOE, and 2 times faster than SRC in average.

VI. Conclusion

In this paper, we propose a Bloom Filter based Cardinality Estimation (BFCE) scheme for tag cardinality estimation in RFID systems. BFCE achieves guaranteed estimation accuracy in constant time. Moreover, implementing BFCE only requires slight updates to the EPCglobal C1G2 standard and fits a wide variety of application purposes. We conduct extensive simulations to evaluate the performance of BFCE under different settings. The experiment results demonstrate that BFCE outperforms state-of-the-arts schemes in terms of time efficiency and estimation accuracy.

Acknowledgment

This work is supported in part by National Natural Science Foundation of China (NSFC) (No.61272466, No.61303233, No. 61170213 and No. 61373181) and National Science Fund for Excellent Young Scientist (No. 61422207).

References

[1] W. Gong, K. Liu, X. Miao, Q. Ma, Z. Yang, and Y. Liu, "Informative counting: fine-grained batch authentication for large-scale rfid systems," in *Proceedings of ACM Mobihoc*, 2013.

[2] W. Gong, Y. Liu, A. Nayak, and C. Wang, "Wise counting: fast and efficient batch authentication for large-scale rfid systems," in *Proceedings of ACM Mobihoc*, 2014.

[3] H. Vogt, "Efficient object identification with passive rfid tags," in *Pervasive Computing*, pp. 98–113, Springer, 2002.

[4] Y. Zheng and M. Li, "Fast tag searching protocol for large-scale rfid systems," *IEEE/ACM Transactions on Networking*, vol. 21, no. 3, pp. 924–934, 2013.

[5] S. Qi, Y. Zheng, M. Li, L. Lu, and Y. Liu, "Collector: A secure rfid-enabled batch recall protocol," in *Proceedings of IEEE INFOCOM*, 2014.

[6] L. M. Ni, Y. Liu, Y. C. Lau, and A. P. Patil, "Landmarc: indoor location sensing using active rfid," *Wireless networks*, vol. 10, no. 6, pp. 701–710, 2004.

[7] J. Wang and D. Katabi, "Dude, where's my card?: Rfid positioning that works with multipath and non-line of sight," in *Proceedings of the ACM SIGCOMM*, 2013.

[8] T. Liu, L. Yang, Q. Lin, Y. Guo, and Y. Liu, "Anchor-free backscatter positioning for rfid tags with high accuracy," in *Proceedings of IEEE INFOCOM*, 2014.

[9] L. Yang, J. Cao, W. Zhu, and S. Tang, "A hybrid method for achieving high accuracy and efficiency in object tracking using passive rfid," in *Proceedings of IEEE PerCom*, 2012.

[10] J. Han, H. Ding, C. Qian, D. Ma, W. Xi, Z. Wang, and L. Jiang, Zhiping amd Shangguan, "Cbid: A customer behavior identification system using passive tags," in *Proceedings of IEEE ICNP*, 2014.

[11] L. Yang, Y. Chen, X.-Y. Li, C. Xiao, M. Li, and Y. Liu, "Tagoram: Real-time tracking of mobile rfid tags to high precision using cots devices," in *Proceedings of ACM MobiCom*, 2014.

[12] J. Han, C. Qian, X. Wang, D. Ma, J. Zhao, W. Xi, Z. Jiang, and Z. Wang, "Twins: Device-free object tracking using passive tags," *Networking, IEEE/ACM Transactions on*, 2015.

[13] Y. Zheng and M. Li, "Pet: Probabilistic estimating tree for large-scale rfid estimation," *Mobile Computing, IEEE Transactions on*, vol. 11, no. 11, pp. 1763–1774, 2012.

[14] Y. Zheng and M. Li, "Zoe: Fast cardinality estimation for large-scale rfid systems," in *Proceedings of IEEE INFOCOM*, 2013.

[15] B. Chen, Z. Zhou, and H. Yu, "Understanding rfid counting protocols," in *Proceedings of ACM MobiCom*, 2013.

[16] W. Gong, K. Liu, X. Miao, and H. Liu, "Arbitrarily accurate approximation scheme for large-scale rfid cardinality estimation," in *Proceedings of IEEE INFOCOM*, 2014.

[17] M. Kodialam and T. Nandagopal, "Fast and reliable estimation schemes in rfid systems," in *Proceedings of ACM MobiCom*, 2006.

[18] M. Kodialam, T. Nandagopal, and W. C. Lau, "Anonymous tracking using rfid tags," in *Proceedings of IEEE INFOCOM*, 2007.

[19] C. Qian, H. Ngan, Y. Liu, and L. M. Ni, "Cardinality estimation for large-scale rfid systems," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 22, no. 9, pp. 1441–1454, 2011.

[20] H. Han, B. Sheng, C. Tan, Q. Li, W. Mao, and S. Lu, "Counting rfid tags efficiently and anonymously," in *Proceedings of IEEE INFOCOM*, 2010.

[21] T. Li, S. Wu, S. Chen, and M. Yang, "Energy efficient algorithms for the rfid estimation problem," in *Proceedings of IEEE INFOCOM*, 2010.

[22] V. Shah-Mansouri and V. W. Wong, "Cardinality estimation in rfid systems with multiple readers," *Wireless Communications, IEEE Transactions on*, vol. 10, no. 5, pp. 1458–1469, 2011.

[23] M. Shahzad and A. X. Liu, "Every bit counts: fast and scalable rfid estimation," in *Proceedings of ACM MobiCom*, 2012.

[24] EPCglobal, "Epc radio-frequency identity protocols class-1 generation-2 uhf rfid protocol for communications at 860 mhz - 960 mhz version 1.2.0," tech. rep., 2008.

[25] J. Wang, H. Hassanieh, D. Katabi, and P. Indyk, "Efficient and reliable low-power backscatter networks," in *Proceedings of the ACM SIGCOMM*, 2012.

[26] L. Kong, L. He, Y. Gu, M.-Y. Wu, and T. He, "A parallel identification protocol for rfid systems," in *Proceedings of IEEE INFOCOM*, 2014.