Towards Adaptive Continuous Scanning in Large-Scale RFID Systems

Haoxiang Liu*, Wei Gong †, Xin Miao †, Kebin Liu † and Wenbo He ‡
*Department of Computer Science and Engineering, Hong Kong University of Science and Technology
†School of Software and TNLIST, Tsinghua University
‡School of Computer Science, McGill University
Email: {haoxiang, gongwei, miao, kebin}@greenorbs.com, wenbohe@cs.mcgill.ca

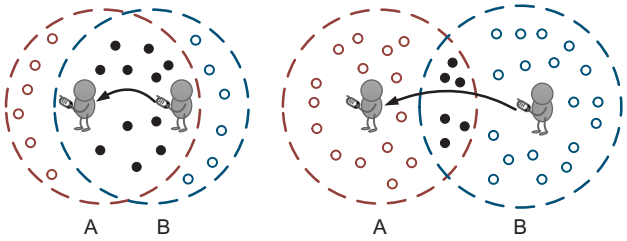
***Abstract*—Radio Frequency Identification (RFID) technology plays an important role in supply chain logistics and inventory control. In these applications, a series of scanning operations at different locations are often needed to cover the entire inventory (tags). In such *continuous scanning* scenario, adjacent scans inevitably read overlapping tags multiple times. Most existing methods suffer from low scanning efficiency when the overlap is small, since they do not distinguish the size of overlap which is an important factor of scanning performance. In this paper, we analytically unveil the fundamental relationship between the performance of continuous scanning and the size of overlap, deriving a critical threshold for the selection of scanning strategy. Further, we design an accurate estimator to approximate the overlap. Combining the estimate and a compact data structure, an adaptive scanning scheme is introduced to achieve low communication time. Through detailed analysis and extensive simulations, we demonstrate that the proposed scheme significantly outperforms previous approach in total scanning time.**

I. Introduction

Radio Frequency Identification (RFID) technology has been widely used in a number of applications, such as inventory control, supply chain logistics and object tracking [1]–[4]. Compared with traditional bar code systems, RFID has the following advantages. First, it extends the operation range from inches to meters. Second, RFID readers can communicate with tags in non-line-of-sight scenario, allowing tags to be embedded in objects.

A fundamental operation in RFID systems is tag scanning, which aims to collect tag IDs to identify the associated objects. For example, a clothes retailer stocks a large volume of clothes in a warehouse. Every piece of them is affixed with an RFID tag. In order to manage the inventory, it is necessary for the retailer to scan the tags to keep track of the clothes. Considering that the clothes (tag) volume may be as large as tens of thousands, and the scanning is frequently operated, a fast scanning method is desired.

In practice, the typical communication distance between an RFID reader and off-the-shelve passive tags is within 12 meters [5], [6]. A single scan clearly cannot cover the entire warehouse. Consequently, a series scanning operations are needed to identify all the tags, which is defined as *continuous scanning*. In continuous scanning, adjacent scans inevitably read overlapping tags multiple times, resulting in redundancy. Figure 1(a) illustrates an example. A is the tag set to be identified, and B is the tag set that is previously collected. Tags in the overlap of A and B are read twice.



(a) Large overlap (b) Small overlap

Fig. 1. The scanning performance depends on the portion of overlap. If it is large, select the unknown tags first. Otherwise, identify A directly.

Sheng et al. [7] investigated the continuous scanning problem and proposed an approach that avoids collecting tags in the overlapping region. Their idea is to take advantage of the tags previously gathered (i.e., B) to only collect the *unknown* tags (i.e., $A-B$). The approach starts with a tag selection phase that keeps the tags in $A-B$ active and meanwhile suppress tags in $A \bigcap B$. However, it suffers from low efficiency when the overlap is small (Figure 1(b)), since the overhead of selection phase overwhelms the benefits it brings. In this case, it is better to directly collect all the tags in A and remove the redundant IDs in the back-end server [1].

In this paper, we observe that the performance of continuous scanning heavily depends on the portion of overlap, and propose an Adaptive COntinuous Scanning scheme (ACOS). We systemically quantify the fundamental relationship between the scanning performance and the overlap, deriving a critical threshold for the selection of scanning strategy. In ACOS design, several challenges must be addressed. First, how to accurately and efficiently estimate the overlap? Second, how to design an efficient method to avoid reading redundancy ($A \bigcap B$) in case the overlap is large? To address the first challenge, we design a lightweight estimator to approximate the overlap based on the idea of MinHash [8]. The estimate is then compared with the threshold to select the scanning strategy. To address the second challenge, we construct a compact tag selector and broadcast it to the tags in A. Notified by the selector, tags in $A \bigcap B$ suppress communication with the reader and will not report their IDs afterwards.

978-1-4799-3360-0/14/$31.00 © 2014 IEEE

[1]We assume the server has powerful computing capability.

The major contributions of this work can be summarized as follows.

- We observe that the performance of continuous scanning is closely related to the portion of overlap. We analytically quantify the fundamental relationship between the scanning performance and the overlap, deriving a critical threshold for scanning strategy selection.
- We design an accurate estimator for the overlap based on MinHash. The scanning strategy is chosen by comparing the estimate with the threshold.
- We propose ACOS, an adaptive scanning scheme that incorporates the estimator and a compact tag selector. Detailed theoretical analysis and extensive simulations are conducted to verify the efficiency of our scheme.

The rest of this paper is organized as follows. In Section II, we review the related work. In Section III, we introduce the system model and formulate the problem. In Section IV, we design and analyze the adaptive continuous scanning scheme ACOS in detail. In Section V, we further discuss several design issues. We evaluate the performance of our design in VI, and finally conclude the paper in Section VII.

II. Related Work

Existing work on tag identification can be divided into two categories, namely single scanning and continuous scanning. In single scanning, the research focuses on designing anti-collision protocols. The protocols generally fall into two classes according to the collision arbitration method: ALOHA-based protocols [9], [10] and tree-based protocols [11], [12]. In ALOHA-based protocols, the reader broadcasts a frame size f to the tags indicating the number of slots in the frame. Each tag randomly selects a slot in the frame and responds its ID. If multiple tags respond in the same slot, their signals collide and none of them can be decoded by the reader. The collided tags are arranged to transmit in the next frame. Tree-based approaches resolve tag collision by splitting collided tags into subsets, and schedule the subsets to transmit separately. The splitting process terminates until the subset contains only one tag. The identification process can be organized as a binary tree.

Different from single scanning, the key challenge in continuous scanning is how to deal with overlapping tags. The most related work to ours is Collecting Unknown (CU) proposed by Sheng et al. [7]. CU employs an algorithm that simply avoids reading overlapping tags, which has two drawbacks. First, CU is an inefficient strategy when the portion of overlap is small. Second, even if the overlap is large, the tag selection approach in CU can be further improved. In this paper, we study the fundamental relationship between the scanning performance and the portion of overlap, and solve both deficiencies of CU. Our solution is based on ALOHA protocol, but orthogonal to all the anti-collision protocols. In addition, Xie et al. [13] conduct the first comprehensive experimental study on mobile reader scanning. They mostly focus on practical issues while our work is more generalized.

Some other topics also investigate tag set operations, such as missing tag detection [14] and tag searching [15], [16]. In missing tag detection, the objective is to quickly pinpoint missing tags in the whole tag set. The problem is essentially finding a subset of tags. Tag searching problem focuses on quickly searching a given set of tags in a batch of unknown ones. This problem can be formulated as finding the intersection of two sets. Both topics are different from continuous scanning problem, since continuous scanning aims to identify the *unknown* tags while the other two deal with *known* tags.

III. System Model and Problem Statement

A. System Model

An RFID continuous scanning system consists of three components: a reader (preferably a mobile reader), a number of tags and a back-end server. The back-end server stores the gathered tag IDs. The reader continuously changes its location to interrogate the tags and reports the IDs to the server via high speed networks. So the communication time between reader and server can be neglected, and we mainly focus on the wireless communication time between reader and tags.

According to the EPC Class-1 Gen-2 standard [17], RFID tags are required to implement a set of mandatory functions, such as responding ID, reading and writing memory. Besides, tags are capable of performing simple computations such as lightweight hash functions.

Our approach adopts the slotted ALOHA model as underlying MAC protocol. The time is divided into a sequence slots of equal size. At the beginning of each slot, the reader issues a request, and tags respond in the second half of the slot. A slot can fall in three categories according to the number of responding tags. If no tag responds, the slot is called an empty slot. If only one tag responds, it is denoted as singleton slot. The transmitted signal in a singleton slot can be successfully decoded by the reader. If multiple tags respond simultaneously, the reader will detect a collision, and thus the slot is called collision slot. Tags have two types of responds as specified by the reader's request: a 96-bit ID or a single bit. We formally denote the single-bit response as *short response* in this paper. The short response contains no information but to inform the reader a non-empty slot. The time of an ID slot and short response slot are denoted as t_{id} and t_s, respectively. Additionally, we denote t_b as the time for the reader to transmit 1 bit. A common relation is $t_b < t_s < t_{id}$.

B. Problem Statement

The objective of continuous scanning is to collect tag IDs in each reading location. As shown in Figure 1, we have two tag sets. A is the set of tags to be identified in current location. B contains the tags gathered in previous locations and stored in the server. The tags in $A-B$ are defined as unknown tags, and those in $A \bigcap B$ are defined as known tags. Therefore, the problem is to design an efficient method to collect the unknown tags with minimum communication time.

IV. ACOS Design and Analysis

In this section, we present the detailed design and analysis of ACOS. The sketch of presentation is as follows. In Section IV-A, we analytically unveil the relationship between the performance of continuous scanning and the portion of overlap. A critical threshold is derived to select the scanning strategy.

In Section IV-B, we present an efficient estimation algorithm for set overlap. In Section IV-C, we propose a compact tag selector in case the set overlap is large. Finally, we show the general design of ACOS in Section IV-D.

A. Quantifying Continuous Scanning

Generally, there are two strategies to identify the unknown tags in $A-B$ in continuous scanning. The first one is a one-phase approach called *Collect All*, or CA for short. Namely, the reader collects all the tags in A and uploads the IDs to the back-end server. The server is responsible for duplicate elimination, i.e., removing the redundant IDs at the back end. The second one is a two-phase approach called Select Unknown, or SU for short. It consists of a selection phase and a collection phase. In the selection phase, the reader suppresses the known tags from responding afterwards, while keeping the unknown tags active. In the collection phase, the reader only reads the active tags. These two strategies exhibit differently when the relationship between set A and B varies. To compare the performance of CA and SU, we separately measure the communication time between the reader and tags of each strategy and make a comparison. The details are as follows.

1) Communication Time of CA: We use the typical Framed Slotted ALOHA-based ID collection protocol (FSA) for CA. In FSA, the reader broadcasts a frame size f, and each tag randomly selects a slot in the frame to respond. Generally, let n be the number of tags to identify. The expected number of empty slots r_0 and singleton slots r_1 in the frame can be easily obtained.

$$r_0 = n(1-\frac{1}{f})^{n-1} \approx ne^{-\rho} \tag{1}$$

$$r_1 = f(1-\frac{1}{f})^{n} \approx fe^{-\rho} \tag{2}$$

where $\rho = n/f$. We define frame efficiency EF as the proportion of singleton slot time in the frame. Since singleton and collision slot time are approximately equal to t_{id} while empty slot time is relatively shorter, we have $EF = \frac{r_1}{r_0\beta+(f-r_0)}$, where constant $0<\beta<1$ is the ratio of empty and singleton (collision) slot time. EF should be maximized to achieve minimum communication time.

Now we calculate the optimal value of ρ that achieves maximum frame efficiency. Given r_0 and r_1, we have $EF = \frac{\rho}{e^{\rho}+\beta-1}$. Let the first order derivative of EF w.r.t. ρ be 0, we derive $e^{\rho}(1-\rho) = 1-\beta$. Therefore, the optimal value of ρ, denoted as $\hat{\rho}$, satisfies $e^{\hat{\rho}}(1-\hat{\rho}) = 1-\beta$.

FSA protocol needs several rounds of frames due to tag collision. Let n_i be the number of unidentified tags before the i^{th} round. Initially, $n_1 = n$. The number of tags identified (singleton slots) in the i^{th} round is $\Delta n = n_i e^{-\hat{\rho}}$ according to Equation 1. Then we have the following induction

$$n_{i+1} = n_i - \Delta n = (1-e^{-\hat{\rho}})n_i$$

Solving the induction, we have $n_i = n(1-e^{-\hat{\rho}})^{i-1}$. Since the process terminates when $n_i < 1$,The identification will run $r = ln(\frac{1}{n})/ln(1-e^{-\hat{\rho}})$ rounds. Finally, we sum over the frames in all rounds to calculate the total communication time. More precisely, the cost is $\sum_{i=1}^{r}(r_0^i\beta+(f_i-r_0^i))t_{id}$, where r_0^i is the expected number of empty slots in the i^{th} frame of size f_i. With a bit calculation, we have

$$\begin{aligned}\sum_{i=1}^{r}(r_0^i\beta+(f_i-r_0^i))t_{id} &= \sum_{i=1}^{r} n_i t_{id} \\ &\approx ne^{\hat{\rho}}t_{id}\end{aligned} \tag{3}$$

Therefore, the overall communication time to identify n tags is $ne^{\hat{\rho}}t_{id}$. With this general result, we establish Theorem 1.

Theorem 1: The communication time of CA is $|A|e^{\hat{\rho}}t_{id}$.

Proof: The result directly follows Equation 3. ■

2) Communication Time of SU: The communication time of SU is the sum of selection cost and collection cost. The selection time essentially depends on the specific selection method. Despite this, we can temporarily use a general form $\lambda|B|\,(\lambda \geq 1)$ to describe the cost, since the reader has to transmit at least 1 bit to preclude each tag in B from A, i.e., select tags in $A-B$. The parameter λ is method specific. We will give the λ for our selection approach later in Section IV-C. The time of collection phase can be easily derived from Equation 3.

Theorem 2: The communication time of SU is $\lambda|B|+|A-B|e^{\hat{\rho}}t_{id}$.

Proof: The proof is simple and omitted. ■

3) Performance comparison: Based on Theorem 1 and Theorem 2, we have the following conclusion. If $|A|e^{\hat{\rho}}t_{id} \geq \lambda|B|+|A-B|e^{\hat{\rho}}t_{id}$, i.e., $\frac{|A\bigcap B|}{|B|} \geq \lambda e^{-\hat{\rho}}/t_{id}$, we choose SU as the scanning strategy. Otherwise, we choose CA. The term $C(A,B) = \frac{|A\bigcap B|}{|B|}$, which depicts the portion of overlap, is called the containment of B in A. Denote T as the optimal scanning time. The relationship between T and $C(A,B)$ can be quantified in Equation 4

$$T = \begin{cases} \lambda|B|+|A-B|e^{\hat{\rho}}t_{id}, & C(A,B) \geq \lambda e^{-\hat{\rho}}/t_{id} \\ |A|e^{\hat{\rho}}t_{id}, & C(A,B) < \lambda e^{-\hat{\rho}}/t_{id} \end{cases} \tag{4}$$

where the critical threshold used for strategy selection is $\lambda e^{-\hat{\rho}}/t_{id}$.

B. Set Containment Estimation

In the last section, we analytically quantify the relationship between scanning performance and $C(A,B)$. The key point, therefore, is to estimate $C(A,B)$ to optimize the performance. By inclusion-exclusion principle, we have

$$C(A,B) = \frac{|A|}{|B|} + 1 - \frac{|A\bigcup B|}{|A\bigcap B|}C(A,B) \tag{5}$$

We introduce $J(A,B) = \frac{|A\bigcap B|}{|A\bigcup B|}$, which is called the *Jaccard similarity* of A and B [18]. So from Equation 5 we get

$$C(A,B) = \frac{J(A,B)}{1+J(A,B)}(1+\frac{|A|}{|B|}) \tag{6}$$

Since the server knows B as a priori, there are two unknowns to estimate $C(A,B)$, namely $|A|$ and $J(A,B)$. To obtain $|A|$, various tag cardinality estimation schemes can be leveraged.

For example, Qian et al. proposed a fast estimation scheme in [19], which achieves fairly accurate result with small overhead. Hence, we focus on estimating $J(A,B)$. Note that the estimation algorithm should be lightweight, otherwise the overhead it incurs might outweigh the benefits. It is clearly infeasible to enumerate the IDs in A and B to exactly figure out $J(A,B)$ due to the high communication cost of collecting A. We instead design lightweight *probabilistic* algorithms based on MinHash [8]. We first propose a preliminary estimator for $J(A,B)$ using multiple hash functions. Then, we present an enhanced estimator using only one hash function while achieves higher efficiency.

1) Multiple Hash-based Similarity Estimator: We design the Multiple Hash-based similarity Estimator (MHE) $\hat{J}_m(A,B)$. The basic idea of MHE is to let each tag in A and B generate a hash value, and the minimum hash values of two sets are compared to derive $\hat{J}_m(A,B)$. Let h be a hash function that uniformly maps the tag IDs in A and B to distinct values in $\{0,1,2,...,D-1\}$. We define $min(h(A))$, $min(h(B))$ as the smallest values in set $h(A)$ and $h(B)$, respectively. If $min(h(A)) = min(h(B))$, it implies that this minimum hash value is generated by the same tag ID in A and B, which obviously lies in $A \bigcap B$. Since each tag ID has the same probability to be mapped to the the smallest hash value, we obtain $Pr(min(h(A)) = min(h(B))) = J(A,B)$. Note that hash collisions should be avoided to guarantee the distinctness of hash values among A and B. According to the arguments in the birthday problem [20], $D = O(|A \bigcup B|^2)$ if each tag ID is mapped to $\{0,1,2,...,D-1\}$ uniformly at random. In the rest of this paper, we assume all hash values are distinct, i.e., no hash collisions exist.

Denote $X \in \{0,1\}$ as the random variable that is 1 if $min(h(A)) = min(h(B))$ and 0 otherwise. Then X is an unbiased estimator for $J(A,B)$, despite that the estimation variance is too high to be useful. To reduce the variance of the estimator, we use k independent hash functions and repeat the procedure k times. Let the k observations be $\{X_1, X_2, ..., X_k\}$. $J(A,B)$ can be estimated using the mean $\bar{X}$ as follows.

$$\hat{J}_m(A,B) = \bar{X} = \frac{1}{k}\sum_{i=1}^{k} X_i$$

$\hat{J}_m(A,B)$ is also an unbiased estimator since $E[\hat{J}_m(A,B)] = E[\bar{X}] = J(A,B)$. Observe that the estimation variance decreases as we increase k. We show how many hash functions k would be sufficient to reach desired accuracy requirement in the following Lemma 1 and Lemma 2. We adopt the typical (ε,δ)-approximation to describe the accuracy of an estimator, where ε and δ are called absolute error bound and failure probability, respectively.

Lemma 1: To meet the accuracy requirement that $Pr(|\hat{J}_m(A,B) - J(A,B)| < \varepsilon) > 1-\delta$, $k \geq \frac{2}{\varepsilon^2} ln\frac{2}{\delta}$.

Proof: This is a typical Chernoff bound [20] for 0-1 random variable X_i. The detailed proof is omitted. ■

Lemma 1 gives a bound w.r.t. absolute error ε. The bound w.r.t. relative error is shown in Lemma 2.

Lemma 2: To meet the requirement that $Pr(|\hat{J}_m(A,B) - J(A,B)| < \varepsilon J(A,B)) > 1-\delta$, $k \geq \frac{2+\varepsilon}{\varepsilon^2 J} ln\frac{2}{\delta}$.

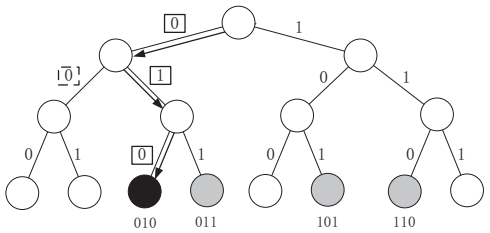


Fig. 2. The minimum hash value search process. The black and gray nodes are existing tags, in which the black node represents the tag with the minimum hash value. Solid boxes denote queries that have non-empty reply, and dotted boxes denote queries with empty reply.

Proof: By Chernoff Bound, we have $Pr(|\hat{J}_m(A,B) - J(A,B)| \geq \varepsilon J(A,B)) \leq 2exp(-\frac{\varepsilon^2}{2+\varepsilon} \cdot J(A,B)k)$. It follows that $Pr(|\hat{J}_m(A,B) - J(A,B)| \leq \varepsilon J(A,B)) > 1 - 2exp(-\frac{\varepsilon^2}{2+\varepsilon} \cdot J(A,B)k)$. To achieve (ε,δ)-accuracy, we get $2exp(-\frac{\varepsilon^2}{2+\varepsilon} \cdot J(A,B)k) > \delta$. Solving this inequality, we have $k \geq \frac{2+\varepsilon}{\varepsilon^2 J} ln\frac{2}{\delta}$. ■

Now the critical issue is to search the minimum hash value from tags in A, i.e., $min(h(A))$, since the server can easily calculate $min(h(B))$ given B as priori. We design a algorithm that exploits binary search. In specific, the reader keeps broadcasting a one-bit query that is used to sequentially match each bit in the hash values, until the minimum hash value is found. Figure 2 illustrates an example of the search algorithm. In the first slot, the reader broadcast a '0' to check if any hash value starts with prefix '0'. Two tags with hash values 010, 011 issue a short response to claim their existence. The reader detects a non-empty slot and proceeds to the second bit. The reader again queries with '0' in the second slot, but detects an empty slot as neither 010 or 011 has 0 in their second bit. So it queries with '1' afterwards. The process is repeated until the reader finds '010' as the minimum hash value. Algorithm 1 and 2 show the pseudo code of the the search algorithm for reader and tags. Note that the each should have a bit string in their memory to store the queries from the reader so far to make the algorithm work. It follows from the algorithm that for each '0' in the minimum hash value, the communication cost between the reader and tags is one slot, and for each '1', the cost is 2 slots. Therefore, the total number of slots of this algorithm is $m_0(x)+2m_1(x)$, where m_0 and m_1 are the number of '0' and '1' in the minimum hash value $x = min(h(A))$, respectively. Moreover, we have

$$\begin{aligned} m_0(x)+2m_1(x) &< 2(m_0(x)+m_1(x)) \\ &= 2logD \end{aligned}$$

Considering that the algorithm should run $k = O(\frac{1}{\varepsilon^2} ln\frac{1}{\delta})$ rounds (for absolute error in Lemma 1), the cost of MHE should be $O(\frac{logD}{\varepsilon^2} ln\frac{1}{\delta})$ slots.

2) Single Hash-based Similarity Estimator: The multiple hash-based estimator requires many rounds of estimation. Therefore, It will be computationally expensive for tags to calculate many hash functions. Can we reduce the computational cost of tags while maintaining the communication cost? In fact, we can just use *one* hash function to achieve

Algorithm 1: The minimum hash value search algorithm for reader.

```
min ← "";
for i ← 1 to logD do
    Broadcast '0';
    Wait for response;
    if response is empty then
        Broadcast '1';
        Append '1' to min;
    else
        Append '0' to min;
    end
end
return min;
```

Algorithm 2: The minimum hash value search algorithm for tags.

```
prefix ← "";
i ← 0;
while True do
    query ← wait for query;
    if query ≠ 1 then
        Append query to prefix;
    else
        Replace the last bit of prefix to 1;
    end
    if h starts with prefix then
        Short response;
    else
        Keep silent;
    end
    i = i + 1;
    if i = logD then
        i = 0;
        if prefix = h then
            Keep silent until activated;
        end
        Clear prefix;
    end
end
```

even faster estimation. Specifically, we let the tags in A and B compute one hash function and select the k smallest hash values from $h(A)$ and $h(B)$. Define $h_k(A)$ as the k smallest hash values of A. $h_k(B)$ and $h_k(A \bigcup B)$ are similarly defined for B and $A \bigcup B$. Due to the property of uniform hashing, $h_k(A \bigcup B)$ can be considered as a simple random sample[2] $\mathcal{S}$ drawn from $A \bigcup B$, but with the tag IDs mapped to hash values. Then, $|h_k(A \bigcup B) \bigcap h_k(A) \bigcap h_k(B)|$ is the subset of samples in $\mathcal{S}$ that are hashed from $|A \bigcap B|$. We therefore can use $\frac{|h_k(A \bigcup B) \bigcap h_k(A) \bigcap h_k(B)|}{k}$ to approximate $J(A,B) = \frac{|A \bigcap B|}{|A \bigcup B|}$,

[2]A simple random sample is a subset of individuals (a sample) chosen from a larger set (a population).

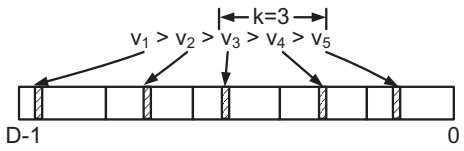


Fig. 3. The ideal distribution of 5 hash values in $[0, D-1]$. We evenly partition the interval to 5 subintervals. $k = 3$ smallest hash values are located in the 3 rightmost subintervals.

given all hash values are distinct. Thus the Single Hash-based Estimator (SHE) is $\hat{J}_s(A,B) = \frac{|h_k(A \bigcup B) \bigcap h_k(A) \bigcap h_k(B)|}{k}$. We show that $\hat{J}_s(A,B)$ is unbiased.

Lemma 3: $\hat{J}_s(A,B)$ is an unbiased estimator for $J(A,B)$.

Proof: The sampling probability is $k/|A \bigcup B|$. Let random variable X_i be 1 if element i is sampled in S and else 0. Then we have $\hat{J}_s(A,B) = \frac{1}{k} \sum_{i \in A \bigcap B} X_i$. The expectation of $\hat{J}_s(A,B)$ is

$$\begin{aligned} E[\hat{J}_s(A,B)] &= \frac{|A \bigcap B|}{k} E[X_i] \\ &= \frac{|A \bigcap B|}{k} \frac{k}{|A \bigcup B|} = J(A,B) \end{aligned}$$

Hence $\hat{J}_s(A,B)$ is an unbiased estimator. ■

The value of k to achieve (ε, δ)-approximation is the same as that in Lemma 1 and Lemma 2.

In SHE, we have to collect the k smallest hash values from A. To achieve this, we design an efficient k minimum hash value search algorithm. The intuition of our algorithm is as follows. The hash values in $h(A)$ can be considered as evenly distributed in space $[0, D-1]$. Hence, if we divide $[0, D-1]$ into subintervals of equal size $\frac{D}{|h(A)|} \approx \frac{D}{|A|}$, then *ideally* each subinterval contains one hash value, as shown in Figure 3. By interval partition, we expectedly reduce the search space for one hash value, from the entire interval D to $\frac{D}{|A|}$. To achieve the partition, the reader finds the smallest integer r that satisfies $2^r \geq \frac{D}{|A|}$, i.e., $r = \lceil log_2 \frac{D}{|A|} \rceil$. The first $logD - r$ bits in the hash values are used to distinguish the subintervals, which we call subinterval prefix. When searching the hash values in a specific subinterval, the reader simply broadcasts the corresponding subinterval prefix to select the corresponding tags. For example, suppose $|A| = 1{,}000$, then $r = 9$ and the subinterval prefixes are: $\{0\}^9, \{0\}^8 1, ..., \{1\}^9$. Note that $\frac{D}{|A|}$ is not necessarily integral power of 2. So prefix matching might divide $[0, D-1]$ into *approximately* even subintervals.

The search process starts from subinterval $[0, 2^r]$. In each subinterval, we conduct binary search to find the contained hash values as described in Algorithm 1 and 2, until k smallest hash values are gathered. If a subinterval contains multiple hash values, we should repeatedly search the minimum hash value among unknown ones the until the subinterval is clear. When a hash value is found, we let the corresponding tag keep silence to prevent it from interrupting the following search (they will be activated when the search process terminates). In Figure 4 we use a binary tree to illustrate how the algorithm works. Initially, we search in the subinterval with prefix 00, which contains hash value 001. Then in the second subinterval,

Algorithm 3: The k-minimum hash value search algorithm for reader.

```
minArray ← []; r ← ⌈log2 D/|A|⌉;
for i ← 0 to 2^(logD−r) − 1 do
    Broadcast the binary string of i;
    Wait for response;
    while response is non-empty do
        min ← binarySearch();
        Add min to minArray;
        Broadcast the binary string of i;
        Wait for response;
    end
end
return minArray;
```

no hash value with prefix 01 exists. So we turn to the third subinterval and find 100, 101. So far, we have gathered three hash values and the search terminates. Algorithm 3 shows the pseudo code of the k minimum hash value search algorithm for the reader. The algorithm for tags is the same as Algorithm 2.

The communication cost of SHE depends on the distribution of k minimum hash values. Consider the situation where each subinterval contains one hash value, the communication cost is $O(\frac{1}{\varepsilon^2} log \frac{D}{|A|} ln \frac{1}{\delta})$ slots, which is better than MHE. Even in the worst case, the cost of SHE is $O(\frac{logD}{\varepsilon^2} ln \frac{1}{\delta})$ slots, which is the same as MHE. Note that both MHE and SHE are sublinear algorithms w.r.t. $|A \bigcup B|$. Hence, the communication cost of estimation is much more efficient than that of identification, which is linear to the number of tags. This ensures the lightweight property of our estimator.

C. Compact Tag Selector

In case Select Unknown (SU) is the better strategy, we need an efficient unknown tag selection method. We show that the previous selection method in [7] is inefficient. A new compact tag selector is introduced and incorporated in ACOS.

1) Previous Method: The existing method proposed in [7] exploits the transition of slot state to select the unknown tags. Specifically, the reader sends a frame f and a random number r. Each tag in A replies a short response in slot $h(ID,r) \in \{0,1,...f-1\}$, where h is a hash function. The reader locally generates a virtual frame f' and maps the tags in B to f' with the same h and r. If an empty slot in f' turns out to be a non-empty slot in f, the tags in A that map to this slot must be unknown. Multiple frames are used until all unknown tags are selected with high probability (with false positive). The communication cost of this method is proportional to $|A \bigcup B|$ according to the analysis in [7].

2) The Compact Tag Selector: We design a compact selector to efficiently select the unknown tags by utilizing Bloom filter [21]. Bloom filter is a bit vector that compactly represents a set. To build the tag selector, the reader constructs a vector of m bits and initialize every bit to 0. It chooses k independent hash functions $h_1, h_2, ..., h_k$, and each has an output uniformly distributed in $\{1,2,...,m\}$. For each tag ID $b \in B$, k positions $h_1(b), h_2(b), ..., h_k(b)$ in the bit vector are set to 1. Consequently, the reader succinctly encodes set B into a compact structure.

In the selection phase, the selector is broadcast to A. When receiving the selector, every tag with ID $a \in A$ checks k positions $h_1(a), h_2(a), ..., h_k(a)$ in the selector. If any of the k positions is 0, then $a \notin B$. So a is an unknown tag and keeps active. If all the k positions in the vector are 1, a classifies itself into $A \bigcap B$, and keeps silence in the collection phase. Since Bloom filter yields false positive, $|A-B| \times P_{FP}$ tags in $A-B$ may be incorrectly suppressed. It is worth noting that both existing selection method and the Bloom filter allow some false positives.

Now, we show how the reader determines the optimal parameters m and k, given a desired false positive rate P_{FP}. Let $BF(B)$ represent the Bloom filter for set B. The false positive rate can be easily obtained as

$$\begin{aligned} p_{FP} &= Pr[x \in BF(B) \bigcap x = 1]^k \\ &= (1-(1-\frac{1}{|BF(B)|})^{k|B|})^k \\ &\approx (1-e^{-k|BF(B)|/|B|})^k \end{aligned} \tag{7}$$

From Equation 7, we can calculate the optimal number of hash functions that achieve minimum false positive rate

$$k = ln2 \frac{|BF(B)|}{|B|} \tag{8}$$

Hence, given the optimal k and p_{FP}, the Bloom filter size becomes

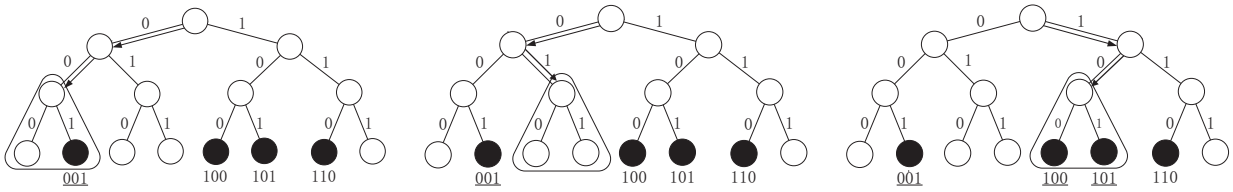
$$|BF(B)| = \frac{-ln\, p_{FP}}{(ln2)^2} |B| \tag{9}$$

With false positive rate P_{FP}, the optimal number of hash functions and Bloom filter size can be derived from Equation 8 and 9. Here we can determine the value of λ in Section IV-A, which is $\lambda = \frac{-ln\, p_{FP}}{(ln2)^2} t_b$ from Equation 9. Therefore, the relationship between total scanning time and $C(A,B)$ in Equation 4 can be renewed in Equation 10, where T_{ACOS}, T_{est} are the scanning time of ACOS and estimation time, respectively. The communication cost of our compact tag selector is proportional to $|B| < |A \bigcup B|$, given the same false positive rate. Hence, our approach is more efficient, especially when $|A \bigcup B|$ is much larger than $|B|$.

$$T_{ACOS} = \begin{cases} \frac{-ln\, p_{FP}|B|}{(ln2)^2} t_b + |A-B| e^{\hat{\rho}} t_{id} + T_{est},\ C(A,B) \geq \frac{-ln\, p_{FP} e^{-\hat{\rho}} t_b}{(ln2)^2 t_{id}} \\ |A| e^{\hat{\rho}} t_{id} + T_{est},\ C(A,B) < \frac{-ln\, p_{FP} e^{-\hat{\rho}} t_b}{(ln2)^2 t_{id}}. \end{cases} \tag{10}$$

D. Adaptive Continuous Scanning Scheme

Now we wrap up and show the general design of ACOS. ACOS basically comprises of two phases, namely estimation phase and execution phase. In estimation phase, we first quickly estimate $|A|$. Next, we adopt the estimator in Section IV-B to estimate $J(A,B)$, and thus $\hat{C}(A,B)$ can be computed.



(a) Subinterval 1, prefix 00 (b) Subinterval 2, prefix 01 (c) Subinterval 3, prefix 10

Fig. 4. An example of $k = 3$ minimum hash value search. The black nodes represent existing hash values. A hash value is underlined if it is found. The highlighted subtrees are subintervals. The search terminates in the 3^{rd} subinterval from the left.

Algorithm 4: The adaptive continuous scanning scheme ACOS.

Input: B, P_{FP}, $\hat{\rho}$.
Estimates $|A|$;
Estimates $J(A,B)$;
Calculate $\hat{C}(A,B) = \frac{\hat{J}(A,B)}{1+\hat{J}(A,B)}(1+\frac{|A|}{|B|})$;
if $\hat{C}(A,B) < \frac{-lnP_{FP}e^{-\hat{\rho}}t_b}{(ln2)^2 t_{id}}$ **then**
  Directly collect the IDs in A using ALOHA-based protocol;
else
  Generate a Bloom filter $BF(B)$ and select tags in $A-B$;
  Collect the IDs in $A-B$ using ALOHA-based protocol;
end

In the execution phase, we can adaptively decide the better scanning strategy (CA or SU) according to Equation 10. If SU is adopted, we use compact selector to select the unknown tags. Algorithm 4 presents the adaptive continuous scanning scheme in detail.

V. Discussion

A. Hash Functions

The set similarity estimation is based on MinHash. In theory, perfect MinHash requires that all elements in the set have equal probability to be the minimum element of image under the hash function. This requirement, however, is unrealistic in practical use. Several work addresses this problem by proposing approximate MinHash families [22]. In Section VI, we show that simple hash functions with pseudorandom output can achieve ideal results, which are favorably lightweight and friendly for tags to use.

B. Bloom Filter Segmentation

As shown before, the length of Bloom filter is proportional to the size of set it encodes. For example, if the set cardinality is $5,000$, and the desired false positive rate is 10^{-3}, the optimal Bloom filter length is given by $5,000 \times \frac{ln(10^3)}{(ln(2))^2} = 71,888$. Therefore, it would be impossible to fit the Bloom filter into a single ID slot. A reasonable solution is to partition the Bloom filter into 96-bit segments. Thus each segment can be enclosed in a single slot and transmitted. When receiving a segment, each tag can only check the corresponding bits it maps to in this segment, and then discard it.

C. The Biased Estimation

We have proven that the two estimators for set similarity, namely $\hat{J}_m(A,B)$ and $\hat{J}_s(A,B)$, are unbiased. However, rather than estimated straightforwardly, the set containment is derived from $\hat{J}(A,B)$. Referring to Equation 6, we have $\hat{C}(A,B) \sim \frac{\hat{J}(A,B)}{1+\hat{J}(A,B)}$. By Jensen's inequality, we have

$$E[\frac{\hat{J}(A,B)}{1+\hat{J}(A,B)}] \leq \frac{E[\hat{J}(A,B)]}{1+E[\hat{J}(A,B)]} = \frac{J(A,B)}{1+J(A,B)}$$

Therefore, the estimate of $C(A,B)$ is biased. We may choose to add a bias correction to acquire more accurate estimate. According to our evaluation, however, this biased estimator already achieves satisfactory result.

VI. Performance Evaluation

In this section, we evaluate the performance of ACOS under extensive simulations. First, we assess the accuracy and cost of set containment estimators. Then we compare ACOS with the recent method CU [7] in terms of total scanning time, under various settings.

A. Simulation Settings

Timings: We follow the EPC Class-1 Generation-2 standard [17] and Phillips I-code specification [23]. The following timing relation is kept in the simulation

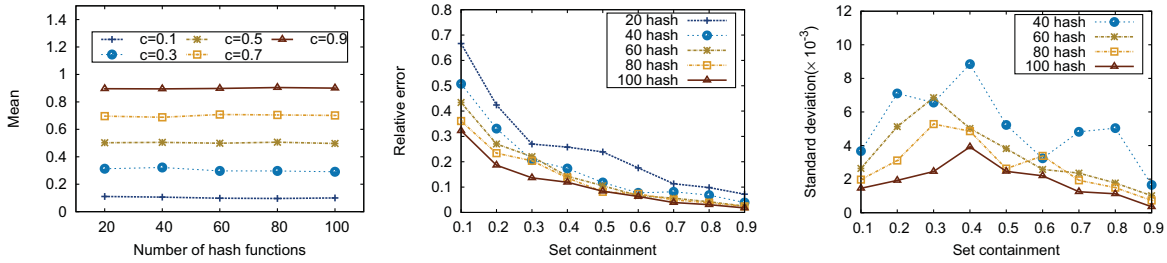
$$t_b \ : \ t_s \ : \ t_{id} \ = \ 1 \ : \ 5 \ : \ 50 \qquad (11)$$

where the unit time is about 0.042ms. Due to the diversity of manufacturers, the timings might be slightly different, but generally have the same scale.

False Positive Rate: The tags to be scanned might be thousands. Hence, we set the false positive rate P_{FP} to be 10^{-4} to keep almost all tags identified. Although P_{FP} impacts the threshold in Equation 10, our scheme is adaptive thus does not rely on a specific P_{FP}.

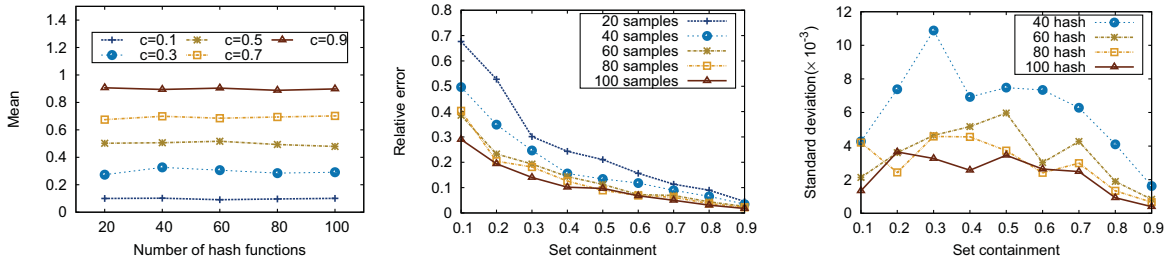
Optimal Identification Time: According to our analysis in Section IV-A, the optimal identification time for n tags is $ne^{\hat{\rho}}$. The parameter $\hat{\rho}$ is 0.39 by solving $e^{\hat{\rho}}(1-\hat{\rho}) = 1-\frac{t_s}{t_{id}}$, where singleton (collision) slot time is set to t_{id} and empty slot time is set to t_s.

Number of tags in A: In the simulation, we assume $|A|$ is known as a prior. One can employ the approach in [19] to quickly estimate $|A|$.



(a) Mean (b) Relative error (c) Standard deviation

Fig. 5. Estimate c using MHE for $J(A,B)$. Then std when number of hash functions is 20 is not shown, since it is much larger than others.



(a) Mean (b) Relative error (c) Standard deviation

Fig. 6. Estimate c using SHE for $J(A,B)$. Then std when number of samples is 20 is not shown, since it is much larger than others.

B. Set Containment Estimation

We use three metrics to evaluate the accuracy of set containment estimation. For simplicity, we use $\hat{c}$ to represent $\hat{C}(A,B)$. The first metric is the mean $E[\hat{c}]$. The second metric is relative error, defined as $\frac{|c-\hat{c}|}{c}$, where c is the actual set containment. The third metric is standard deviation (std) $\sigma = \sqrt{E[(c-\hat{c})^2]}$.

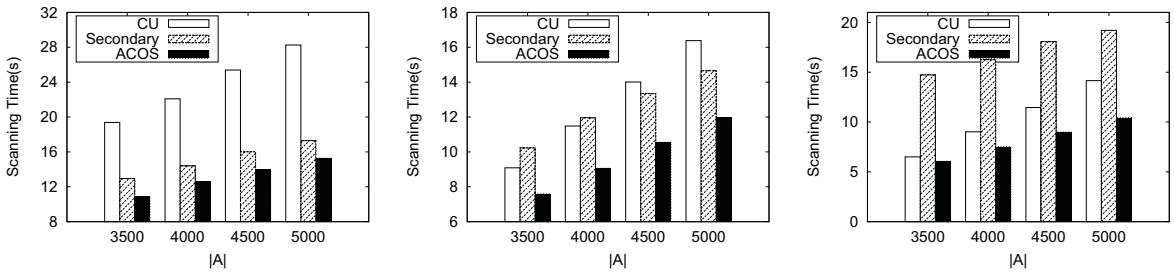
The results in Figure 5 and 6 show the accuracy of $\hat{c}$, when estimator MHE and SHE are adopted for $J(A,B)$. We make the following observation from these figures. First, $\hat{c}$ is approximately unbiased, since the $E[\hat{c}]$ is very close to c when c ranges from 0.1 to 0.9. Second, the relative error of estimation drops when we increase the number of hash functions (in MHE) or the number of samples (in SHE). As illustrated in Figure 5(b) and 6(b), if we choose 100 hash functions/samples, the relative error decreases to approximately 0.1 when $c = 0.5$. Therefore, we claim that 100 hash functions/samples are fairly enough to acquire an estimation that helps us select correct scanning strategy. In addition, the relative error diminishes when c increases. This is easy to understand by Lemma 2. Third, the standard deviation gets smaller when more hash functions/samples are used, as shown in Figure 5(c) and 6(c). For example, the std drops from 5.2×10^{-3} with 40 hash functions to 2.5×10^{-3} with 100 hash functions, when $c = 0.5$.

Moreover, we examine the communication cost of estimation. Note that the communication cost of estimating c is essentially the communication cost of estimating $J(A,B)$. Therefore, we just evaluate the cost of MHE and SHE separately. Two parameters $|A|$ and k may impact the cost, where k is the number of hash functions in MHE or number of samples in SHE. The evaluation is conducted from two perspectives. First, we fix $|A|$ to $5,000$, and vary k from 40 to 100. The results are presented in Figure 8. The communication cost of MHE and SHE increase with k. Particularly, SHE has slower growth trend since the cost of SHE is approximately $O(log\frac{D}{|A|}k)$, while that of MHE is $O(logDk)$. Second, we vary $|A|$ and keep $k = 100$. In Figure 9, we observe that SHE is more efficient than MHE. Additionally, the cost of MHE is independent of $|A|$, whereas the cost of SHE slightly decreases as the $|A|$ grows. This observation further verifies our cost analysis. Due to its better efficiency and comparable accuracy, we use SHE in our following simulations.

C. Performance Comparison

We compare the performance of ACOS with the recent continuous scanning approach CU. ACOS selects the better scanning strategy from Collect All (CA) and Select Unknown (SU), based on the estimation $\hat{c}$. We call the other strategy that is not eventually selected by ACOS as *secondary* strategy. Thus, we compare three methods, namely ACOS, the secondary strategy and CU. The estimation time is included in ACOS but not in the other two methods.

To allow different continuous scanning scenarios, we vary the set containment c. Namely c is set to 0.1, 0,5 and 0.9. For each c, we set $|B| = 5,000$ and change $|A|$ from 3,500 to 5,000. Figure 7 illustrates the comparison in terms of scanning time among the three methods. In all scenarios, ACOS significantly outperforms CU and secondary. For example, when $c = 0.1$ and $|A| = 3,500$, ACOS is 16% faster than secondary and 44% faster than CU. The reason why ACOS always outperforms lies in two aspects. First, when c is small (Figure 7(a)), ACOS chooses CA, which is apparently the better strategy. In contrast, CU selects tags in $A - B$ first, resulting in large overhead. Second, when c is large (Figure 7(b) and 7(c)), both



(a) $c = 0.1$ (b) $c = 0.5$ (c) $c = 0.9$

Fig. 7. The scanning time with different parameters. $|B| = 5{,}000$. $c \in \{0.1, 0.5, 0.9\}$. Hence, $|A \bigcap B| \in \{500, 2{,}500, 4{,}500\}$. For each c, $|A|$ is varied from 3,500 to 5,000.

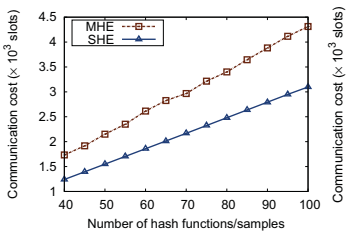


Fig. 8. Vary k, $|A|$=5,000.

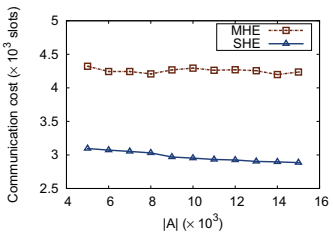


Fig. 9. Vary $|A|$, k=100.

TABLE I
PERCENTAGE OF ESTIMATION COST IN ACOS.

c	Number of tags			
	3,500	4,000	4,500	5,000
0.1	5.88%	5.05%	4.48%	4.05%
0.5	8.42%	7.02%	5.93%	5.17%
0.9	10.2%	8.21%	6.83%	5.87%

ACOS and CU adopt a tag selection phase. ACOS is still better than CU due to the efficiency of compact tag selector. In Table I, we further present the percentage of estimation cost in ACOS under various settings. It is evident that our estimation scheme is lightweight, constituting no more than 10.2% of total scanning cost. Therefore, even if estimation cost is counted, ACOS still achieves much performance gain compered with CU.

VII. CONCLUSION

In this paper, we propose an Adaptive COntinuous Scanning scheme ACOS. In ACOS, we analytically unveil the fundamental relationship between the performance of continuous scanning and the tag set containment. In addition, lightweight estimation algorithms for tag set containment are designed to choose the scanning strategy. ACOS significantly outperforms the existing continuous scanning method in scanning time.

VIII. ACKNOWLEDGEMENT

This work is supported in part by NSFC Young Scholar Grant No. 61103187 and No. 61303196, National High-Tech R&D Program of China (863) under grant No. 2011M500019 and China Postdoctoral Science Foundation No. 2013M540950.

REFERENCES

[1] Y. Liu, Y. Zhao, L. Chen, J. Pei, and J. Han, "Mining frequent trajectory patterns for activity monitoring using radio frequency tag arrays," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 23, no. 11, pp. 2138–2149, 2012.

[2] L. Shangguan, Z. Li, Z. Yang, M. Li, and Y. Liu, "Otrack: Order tracking for luggage in mobile rfid systems," in *Proceedings of Infocom*, 2013.

[3] L. Yang, Y. Qi, J. Fang, X. Ding, T. Liu, and M. Li, "Frogeye: Perception of the slightest tag motion," in *Proceedings of Infocom*, 2014.

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

[5] D. Sen, P. Sen, and A. M. Das, *RFID for Energy and Utility Industries*. Pennwell Books, 2009.

[6] L. Yang, J. Han, Y. Qi, and Y. Liu, "Identification-free batch authentication for rfid tags," in *ICNP*, 2010.

[7] B. Sheng, Q. Li, and W. Mao, "Efficient continuous scanning in rfid systems," in *Proceedings of IEEE Infocom*, 2010.

[8] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher, "Min-wise independent permutations," in *Proceedings of ACM STOC*, 1998.

[9] S. Lee, S. Joo, and C. Lee, "An enhanced dynamic framed slotted aloha algorithm for rfid tag identification," in *Proceedings of Mobiquitous*, 2005.

[10] J. R. Cha and J. H. Kim, "Dynamic framed slotted aloha algorithms using fast tag estimation method for rfid system," in *Proceedings of IEEE CCNC*, 2006.

[11] J. Myung and W. Lee, "Adaptive splitting protocols for rfid tag collision arbitration," in *Proceedings of ACM MobiHoc*, 2006.

[12] M. Shahzad and A. X. Liu, "Probabilistic optimal tree hopping for rfid identification," in *Proceedings of ACM Sigmetrics*, 2013.

[13] L. Xie, Q. Li, X. Chen, S. Lu, and D. Chen, "Continuous scanning with mobile reader in rfid systems: an experimental study," in *Proceedings of ACM Mobihoc*, 2013.

[14] T. Li, S. Chen, and Y. Ling, "Identifying the missing tags in a large rfid system," in *Proceedings of ACM Mobihoc*, 2010.

[15] Y. Zheng and M. Li, "Fast tag searching protocol for large-scale rfid systems," in *Proceedings of IEEE ICNP*, 2011.

[16] M. Chen, W. Luo, Z. Mo, S. Chen, and Y. Fang, "An efficient tag search protocol in large-scale rfid systems," in *Proceedings of IEEE Infocom*, 2013.

[17] "Epc radio-frequency identity protocols class-1 generation-2 uhf rfid protocol for communications at 860mhz-960mhz," 2008.

[18] "Jaccard index," http://en.wikipedia.org/wiki/Jaccard_index.

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

[20] M. Mitzenmacher and E. Upfal, *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

[21] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[22] P. Indyk, "A small approximately min-wise independent family of hash functions," in *Proceedings of SODA*, 1999.

[23] "Philips i-code uid smart label ic functional specification," http://www.nxp.com/documents/data_sheet/SL092030.pdf, 2004.