An Efficient Tag Search Protocol in Large-Scale RFID Systems

Min Chen† Wen Luo† Zhen Mo† Shigang Chen† Yuguang Fang‡
†Department of Computer & Information Science & Engineering
‡ Department of Electrical & Computer Engineering
University of Florida, Gainesville, FL 32611, USA
Email:†{min, wluo, zmo, sgchen}@cise.ufl.edu ‡ fang@ece.ufl.edu

***Abstract*—Radio frequency identification (RFID) technology has many applications in inventory management, supply chain, product tracking, transportation and logistics. One research issue of practical importance is to search for a particular group of tags in a large-scale RFID system. Time efficiency is a core factor that must be taken into consideration when designing a tag search protocol to ensure scalability. In this paper, we design a new technique called *filtering vector*, which can significantly reduce transmission overhead during search process, thereby shortening search time. Based on this technique, we propose an iterative tag search protocol. In each round, we filter out some tags and eventually terminate the search process when the search result meets the accuracy requirement. The simulation results demonstrate that our protocol performs much better than the best existing ones.**

I. Introduction

Recent years have witnessed the rapid development of radio frequency identification (RFID) technology. It is becoming increasingly utilized in various applications, such as inventory management, supply chain, product tracking, transportation and logistics [1]–[5]. Generally speaking, a RFID system comprises three components: one or multiple RFID readers, a large set of RFID tags, and a backend server. Each tag has a unique ID to identify the object it is attached to. Equipped with an antenna, a tag is capable of transmitting and receiving radio signals, through which communications with the readers are achieved. Hence, the readers can collect the IDs and other useful information from tags located in their coverage areas, and then send the gathered data to the backend server for further process.

This paper focuses on the *tag search problem* in large RFID systems. We use an example to illustrate the problem. Suppose a manufacturer suspects that some of its products may be defective, but those products have already been distributed in different warehouses. The manufacturer knows the IDs of tags attached to those suspected products and wants to recall them for further inspection. Thus the manufacturer asks for tag search in each warehouse: Given a set of *wanted* tag IDs, the problem is to search in the coverage area of a reader and identify the tags that belong to the set. Note that there may exist other tags in the area that do not belong to the set. To meet the stringent delay requirements of real-world applications, time efficiency is a critical performance metric for the RFID tag search problem. In our example, it is highly desirable to make the search quick in a busy warehouse as lengthy searching process may interfere with other activities that move things in and out of the warehouse. The only prior work studying this problem is called CATS [6], which however does not work under some common conditions (e.g., if the size of the wanted set is much larger than the number of tags in the coverage area of the reader).

The main contribution of this paper is a fast tag search method based on a new technique called *filtering vectors*. A filtering vector is a compact one-dimension bit array constructed from tag IDs, which can be used not only for tag filtration, but also for parameter estimation. Using the filtering vectors, we design, analyze, and evaluate a novel iterative tag search protocol, which progressively improves the accuracy of search result and reduces the time for each iteration to a minimum by using the information learned from previous iterations. Given an accuracy requirement, the iterative protocol will terminate once the search result meets the accuracy requirement. We show that our protocol performs much better than the CATS protocol and other alternatives that we use for comparison. In particular, the new protocol is able to work efficiently under conditions when the CATS protocol no longer works.

The rest of this paper is organized as follows. Section II gives the system model and the problem statement. Section III briefly introduces the prior work. Section IV describes our new protocol in detail. Section V evaluates the performance of our protocol by simulations. Section VI presents some related RFID work. Section VII draws the conclusion.

II. System Model and Problem Statement

A. System Model

In our model, a RFID system consists of multiple readers, a large number of tags and a backend server. The tags may be battery-powered *active tags*, or *passive tags* that are powered by radio energy emitted from the reader. Each tag has a unique 96-bit ID according to the EPC global Class-1 Gen-2 standard [7]. A tag is able to communicate with the reader wirelessly and perform some computations such as hashing. The backend server is responsible for data storage and information processing. It is capable of carrying out high-performance computations. The reader and the backend server are connected via a high speed wired or wireless link. They can be regarded as an integrated unit, still called the reader for simplicity.

In practice, the tag-to-reader ($T \Rightarrow R$) transmission rate and the reader-to-tag ($R \Rightarrow T$) transmission rate may be different

and subject to the environment. For example, as specified in the EPC global Class-1 Gen-2 standard, the $T \Rightarrow R$ transmission rate is 40kbps $\sim$ 640 kbps in the FM0 encoding format or 5kbps $\sim$ 320kbps in the Miller modulated subcarrier encoding format, while the $R \Rightarrow T$ transmission rate is about 26.7kbps $\sim$ 128kbps [7]. However, to simplify our discussions, we assume the $T \Rightarrow R$ transmission rate and the $R \Rightarrow T$ transmission rate are the same, and it is straightforward to adapt our protocol for asymmetric transmission rates.

B. Time Slots

The RFID reader and the tags in its coverage area use a framed slotted MAC protocol to communicate. We assume that clocks of the reader and all tags in the RFID system are synchronized by the reader's signal. During each frame, the communication is initialized by the reader in a request-and-response mode, namely, the reader broadcasts a request with some parameters to the tags and then waits for the tags to reply in the subsequent time slots.

Consider an arbitrary time slot. We call it an *empty slot* if no tag replies in this slot, or a *busy slot* if one or more tags respond in this slot. Only one-bit information is needed for distinguishing an empty slot from a busy slot: '0' for an empty slot with an idle channel and '1' for a busy slot with a busy channel. We denote the length of a slot for one-bit information as t_s.

Some prior RFID work needs another type of slots, carrying 96-bit IDs, whose length is denoted as t_{id}.

C. Problem Statement

Suppose we are interested in a known set of tag IDs $X = \{x_1, x_2, x_3, \cdots\}$, each $x_i \in X$ is called a *wanted tag*. For example, the set may contain tag IDs on a certain type of products under recall by a manufacturer. Let $Y = \{y_1, y_2, y_3, \cdots\}$ be the set of tags within the coverage area of a RFID system (e.g., in a warehouse). Each x_i or y_i represents a tag ID. The *tag search problem* is to search for which wanted tags are present in the coverage area. Let W denote the subset of wanted tags that are present in the coverage area. Since every tag in W is a wanted tag, $W \subseteq X$. Since each tag in W is in the coverage area, $W \subseteq Y$. Therefore, $W = X \cap Y$. We define the *intersection ratio* of X and Y as

$$R_{INTS} = \frac{|W|}{\min\{|X|, |Y|\}}. \tag{1}$$

Exactly finding W can be expensive if X and Y are very large, and it is much more efficient to find W approximately, allowing small bounded error [6]. We take the approximate approach in this paper. Our solution performs iteratively. Each round rules out some tags in X when it becomes certain that they are not in the coverage area (i.e., Y), and it also rules out some tags in Y when it becomes certain that they are not wanted ones in X. These ruled-out tags are called *non-candidate tags*. Other tags that remain possible to be in both X and Y are called *candidate tags*. At the beginning, the search result is initialized to all wanted tags X. As our solution is iteratively executed, the search result shrinks towards W when more and more non-candidates are ruled out.

Let W^* be the final search result. We have the following two requirements:

1) All wanted tags in the coverage area must be detect, namely, $W \subseteq W^*$.
2) A *false positive* occurs when a tag in $X - W$ is included in W^*, i.e., a tag not in the coverage area is kept in the search result by the reader. The *false positive ratio* is the probability for any tag in $X - W$ to be in W^* after the execution of a search protocol. We want to bound the false positive ratio by a pre-specified system requirement P_{REQ}, whose value is set by the user. In other words, we expect
$$\frac{|W^* - W|}{|X - W|} \leq P_{REQ}. \tag{2}$$

III. Background

We discuss some prior work that can be applied to the tag search problem.

A. Tag Identification

Plenty of RFID research concentrates on designing efficient tag identification protocols that collect the IDs of tags in a RFID systems. These protocols collect all tag IDs in Y and thus can be used to solve the tag search problem simply by computing the intersection $X \cap Y$ once Y is known. Each 96-bit ID transmission from a tag to the reader takes a time slot of t_{id}, which is much longer than the one-bit slot t_s. Due to collision, the lower bound for ALOHA-based identification protocols such as DFSA [8] and EDFSA [9] to collect all tag IDs is $e \times |Y|$ time slots, where e is the natural constant. Hence, when a tag identification protocol is used, the search time is at least

$$T_{identify} = e \times |Y| \times t_{id}. \tag{3}$$

B. Baseline Protocol

A *baseline protocol* for the tag search problem is given in [6], which is much faster than the tag identification protocols when $|X| \ll |Y|$. Instead of collecting all IDs in Y, the reader broadcasts the IDs in X one by one. Each tag checks whether the received ID is identical to its own ID. If so, the tag transmits a one-bit short response to notify the reader about its presence; otherwise, the tag keeps silent. Hence, the search time of baseline protocol is

$$T_{baseline} = |X| \times (t_{id} + t_s). \tag{4}$$

The baseline protocol improves time efficiency due to the following reasons:

1) It avoids collecting all IDs of a large tag set Y when $|Y| \gg |X|$.
2) It eliminates collision incurred in the tag identification protocols.

However, the baseline protocol also has serious limitations. It does not work well when $|X| \gg |Y|$. The energy consumption of tags (particularly when active tags are used) is significant because tags in Y have to continuously listen to the channel and receive a large number of IDs until its own ID is received.

C. CATS Protocol

To further reduce the search time, Zheng et al. propose a two-phase protocol named *Compact Approximator based Tag Searching protocol* (CATS) [6], which is the most efficient solution for the tag search problem to date.

The main idea of the CATS protocol is to encode tag IDs into an L_1-bit Bloom filter and then transmit the Bloom filter instead of the IDs themselves. In its first phase, the reader encodes all IDs of wanted tags in X into a Bloom filter, and then broadcasts this filter together with some parameters to tags in the coverage area. Having received this Bloom filter, each tag tests whether it belongs to the set X. If the answer is negative, the tag is a non-candidate and will keep silent for the remaining time. After the filtration of phase one, the number of candidate tags in Y is reduced. During the second phase, the remaining candidate tags in Y report their presence in a second L_2-bit Bloom filter constructed from a frame of time slots t_s. Each candidate tag transmits in k slots that it is mapped to. Listening to channel, the reader builds the Bloom filter based on the status of the time slots: '0' for an idle slot where no tag transmits, and '1' for a busy slot where at least one tag transmits. Using this Bloom filter, the reader conducts filtration for the IDs in X to see which of them belong to Y, and the result is regarded as $X \cap Y$.

With a pre-specified false positive ratio requirement P_{REQ}, the CATS protocol uses the following optimal settings for L_1 and L_2:

$$L_1 = |X| \log_\phi \left(-\frac{\alpha |X|}{\beta |Y| \ln P_{REQ}} \right), \tag{5}$$

$$L_2 = \frac{|X|}{\ln \phi} \left(\ln P_{REQ} - \frac{\alpha}{\beta} \right), \tag{6}$$

where ϕ is a constant which equals 0.6185, α and β are constants pertaining to $R \Rightarrow T$ transmission rate and $T \Rightarrow R$ transmission rate respectively. Because $\phi < 1$, it is required that $|X| < -\frac{\beta}{\alpha}|Y| \ln P_{REQ}$; otherwise, L_1 will become negative. When $\alpha = \beta$, i.e. the $R \Rightarrow T$ transmission rate and the $T \Rightarrow R$ transmission rate are identical, the total search time of the CATS protocol is:

$$\begin{aligned} T_{CATS} &= (L_1 + L_2) \times t_s \\ &= |X| \left(\log_\phi \left(\frac{-|X|}{|Y| \ln P_{REQ}} \right) + \frac{\ln P_{REQ} - 1}{\ln \phi} \right) \times t_s. \end{aligned} \tag{7}$$

IV. A Fast Tag search Protocol Based on Filtering Vectors

In this section, we propose an *Iterative Tag Search Protocol* (ITSP) to solve the tag search problem in large-scale RFID systems.

A. Motivation

Although the CATS protocol takes a significant step forward in solving the tag search problem, it still has several important drawbacks. First, when optimizing the Bloom filter sizes L_1 and L_2, CATS approximates $|X \cap Y|$ simply as $|X|$. This rough approximation may cause considerable overhead when $|X \cap Y|$ deviates significantly from $|X|$.

Second, it assumes that $|X| < |Y|$ in its design. In reality, the number of wanted tags may be far greater than the number in the coverage area of a RFID system. For example, there may be a huge number $|X|$ of tagged products that are under recall, but as the products are distributed to many warehouses, the number $|Y|$ of tags in a particular warehouse may be much smaller than $|X|$.

Third, the performance of CATS is sensitive to the false positive ratio requirement P_{REQ}. The performance deteriorates when the value of P_{REQ} is very small. While the simulations in [6] set P_{REQ} = 5%, its value may have to be much smaller in some practical cases. For example, suppose $|X| = 100,000$, and $|W| = 1,000$. If we set $P_{REQ} = 5\%$, the number of wanted tags that are *falsely claimed* to be in Y by CATS will be up to $|X - W| \times P_{REQ} = 4,995$, far more than the 1,000 wanted tags that are actually in Y.

We will show that an iterative way of implementing Bloom filters is much more efficient than the classical way that the CATS protocol adopts.

B. Iterative Implementation of Bloom Filter

A Bloom filter is a compact data structure that encodes the membership for a set of items. To represent a set $S = \{e_1, e_2, \cdots, e_m\}$ using a Bloom filter, we need a bit array of length l in which all bits are initialized to zeros. To encode each element $e \in S$, we use k hash functions, h_1, h_2, $\cdots$, h_k, to map the element randomly to k bits in the bit array, and set those bits as ones. For membership lookup of an element b, we again map the element to k bits in the array and see if all of them are ones. If so, we claim that b belongs to S; otherwise, it must be true that $b \notin S$. A Bloom filter may cause false positives: a non-member element is falsely claimed as a member in S. The probability for a false positive to occur in a membership lookup is given as follows [10]:

$$P_B = \left(1 - \left(1 - \frac{1}{l} \right)^{km} \right)^k \approx \left(1 - e^{-km/l} \right)^k. \tag{8}$$

When $k = \ln 2 \times \frac{l}{m}$, P_B is minimized to $\left(\frac{1}{2}\right)^k = \left(\frac{1}{2}\right)^{\ln 2 \frac{l}{m}}$. In order to achieve a target value of P_B, the minimum size of the filter is $-\frac{\ln P_B}{(\ln 2)^2} m$. CATS sends one Bloom filter from the reader to tags and another Bloom filter from tags back to the reader. Suppose we want to have $P_B = 0.001$ for the first Bloom filter that encodes X. Since $m = |X|$, the minimum size of the filter becomes $14.4 \times |X|$ bits, where each bit is implemented by a time slot t_s. Similarly, the size of the second filter is also related to its target false-positive probability. Below we provide motivation for the idea of filtering vectors that can significantly reduce the filter size.

A Bloom filter can be implemented in a segmented way. We can divide the bit array into k equal segments, and the i^{th} hash function will map each element to a random bit in the i^{th} segment, for $i \in [1..k]$. We name each segment as a *filtering vector*. The number of bits in a segment is l/k. The following formula gives the false-positive probability of a single filtering vector, i.e., the probability for a non-member to be hashed to

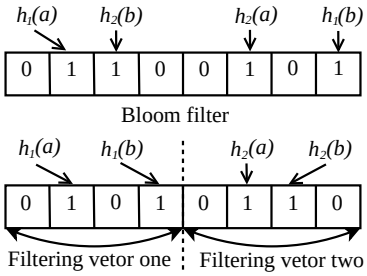

Fig. 1. Bloom filter and filtering vectors

a '1' bit in the vector:

$$P_{FV} = 1 - \left(1 - \frac{1}{l/k}\right)^m \approx 1 - e^{-km/l}. \quad (9)$$

Since there are k independent segments, the overall false-positive probability of the Bloom filter is

$$P'_B = (P_{FV})^k \approx \left(1 - e^{-km/l}\right)^k, \quad (10)$$

which is approximately the same as the result in (8). This means the two ways of implementing the Bloom filter have similar effect. The value P'_B is also minimized when $k = \ln 2 \times \frac{l}{m}$, and thus the optimal size of each filtering vector is

$$\frac{l}{k} = \frac{m}{\ln 2}, \quad (11)$$

which results in

$$P_{FV} \approx \frac{1}{2}. \quad (12)$$

Hence, each filtering vector can filter out half of non-members.

Fig. 1 illustrates the concept of filtering vectors. Suppose we have two elements a and b, two hash function h_1 and h_2, and an 8-bit bit array. First, suppose $h_1(a)$ mod 8 = 1, $h_1(b)$ mod 8 = 7, $h_2(a)$ mod 8 = 5, $h_2(b)$ mod 8 = 2, and we construct a Bloom filter for a and b in the upper half of the figure. Next, we divide the bit array into two 4-bit filtering vectors (note 4-bit is not the optimal size of a filtering vector in this case, we just use it for illustration), and apply h_1 on the first segment and h_2 on the second segment. Since $h_1(a)$ mod 4 = 1, $h_1(b)$ mod 4 = 3, $h_2(a)$ mod 4 = 1, $h_2(b)$ mod 4 = 2, we build the two filtering vectors in the lower half of the figure.

In this work, we use filtering vectors in a novel iterative way: The Bloom filters between the reader and tags are exchanged in rounds; only one filtering vector is exchanged in each round, and the size of filtering vector is continuously reduced in subsequent rounds, such that the overall size of the whole Bloom filter is much reduced. Below we use a simplified example to illustrate the idea: Suppose there is no wanted tag in the coverage area of a RFID reader, namely, $X \cap Y = \emptyset$. In round one, we firstly encode X in a filtering vector of size $|X|/\ln 2$ through a hash function h_1, and broadcast the vector to filter tags in Y. Using the same hash function, each candidate tag in Y knows which bit in the vector it is mapped to, and it only needs to check the value of that bit. If the bit is zero, the tag becomes a non-candidate and will not participate in the execution further. The filtering vector reduces the number of candidate tags in Y to about $|Y| \times P_{FV} = |Y|/2$. Then a filtering vector of size $|Y|/(2\ln 2)$ is sent from the remaining candidate tags in Y back to the reader in order to filter X. After filtering, the number of candidate tags in X is reduced to about $|X| \times P_{FV} = |X|/2$. Only the candidate tags in X need to be encoded in the next filtering vector, using a different hash function. Hence, in the second round, the size of the filtering vector from the reader to tags is reduced by half to $|X|/(2\ln 2)$, and similarly the size of the filtering vector from tags to the reader is also reduced by half to $|Y|/(4\ln 2)$. Repeating the same process, it is easy to see that, in the i_{th} round, the size of the filtering vector from the reader to tags is $|X|/(2^{i-1}\ln 2)$, and the size of the filtering vector from tags to the reader is $|Y|/(2^i \ln 2)$. After n rounds, the total size of all filtering vectors from the reader to tags is

$$\frac{1}{\ln 2}\sum_{i=1}^{n}\frac{|X|}{2^{i-1}} < \frac{2|X|}{\ln 2}, \quad (13)$$

which compares favorably to the traditional approach of sending $14.4 \times |X|$ bits of Bloom filter in one shot in our earlier example. Similarly, the total size of all filtering vectors from tags to the reader is

$$\frac{1}{\ln 2}\sum_{i=1}^{n}\frac{|Y|}{2^{i}} < \frac{|Y|}{\ln 2}, \quad (14)$$

and $P_{FP} = (P_{FV})^n \approx \left(\frac{1}{2}\right)^n$. We can make P_{FP} as small as we like by increasing n, while the total transmission overhead never exceeds $\frac{1}{\ln 2}(2|X| + |Y|)$ bits. The strength of filtering vectors in bidirectional filtration lies in their ability to reduce the candidate sets during each round, thereby diminishing the sizes of filtering vectors in subsequent rounds.

C. Generalized Approach

Unlike the CATS protocol, our iterative approach breaks the bidirectional filtration in tag search process into multiple rounds. Before the i^{th} round, the set of candidate tags in X is denoted as X_i ($\subseteq X$), which is also called the search result after the $(i-1)^{th}$ round. The final search result is the set of remaining candidate tags in X after all rounds are completed. Before the i^{th} round, the set of candidate tags in Y is denoted as Y_i ($\subseteq Y$). Initially, $X_1 = X$ and $Y_1 = Y$. We define $U_i = X_i - W$ and $V_i = Y_i - W$. Because W is always a subset of both U_i and V_i, we have

$$\begin{aligned} |U_i| &= |X_i| - |W| \\ |V_i| &= |Y_i| - |W|. \end{aligned} \quad (15)$$

Instead of exchanging a single filtering vector at a time, we generalize our iterative approach by allowing multiple filtering vectors to be sent consecutively. Each round consists of two phases. In phase one of the i^{th} round, the RFID reader broadcasts a number m_i of filtering vectors, which shrink the set of remaining candidate tags in Y from Y_i to Y_{i+1}. In phase two of the i^{th} round, one filtering vector is sent back to the reader in the following distributed way: The candidate tags in Y_{i+1} randomly select slots in a time frame to transmit one-bit responses to the reader. By listening to the states of the slots, the reader reconstructs the filtering vector from the candidate tags in Y_{i+1}. The received filtering vector shrinks the set of

remaining candidates on the reader's side from X_i to X_{i+1}, setting the stage for the next round. This process continues until the false positive ratio meets the requirement of P_{REQ}.

The values of m_i will be determined in the next subsection. If $m_i > 0$, multiple filtering vectors will be sent consecutively from the reader to tags in one round. If $m_i = 0$, no filtering vector is sent from the reader in this round. When this happens, it essentially allows multiple filtering vectors to be sent consecutively from tags to the reader (across multiple rounds).

D. Values of m_i

Let K be the total number of rounds. After all K rounds, we use X_{K+1} as our search result. There are in total K filtering vectors sent from tags to the reader. We know from subsection IV-B that each filtering vector can filter out half of non-members (in our case, tags in $X - W$). To meet the false positive ratio requirement P_{REQ}, the following constraint should hold

$$(P_{FV})^K = \left(\frac{1}{2}\right)^K \leq P_{REQ}. \tag{16}$$

Hence, the value of K must be at least $-\frac{\ln P_{REQ}}{\ln 2}$.

Next, we discuss how to set the values of m_i, $1 \leq i \leq K$, in order to minimize the execution time of each round. We use $FV(\cdot)$ to denote the filtering vector of a set. In phase one of the i^{th} round, the reader builds m_i filtering vectors, denoted as $FV_{i1}(X_i)$, $FV_{i2}(X_i)$, $\cdots$, $FV_{im_i}(X_i)$, which are consecutively broadcasted to the tags. From (11), we know the size of each filtering vector is $|X_i|/\ln 2$. After filtering based on these vectors, the number of remaining candidate tags in Y_{i+1} is

$$\begin{aligned} |Y_{i+1}| &= |V_i| \times (P_{FV})^{m_i} + |W| \\ &\approx |V_i| \times (1/2)^{m_i} + |W| \\ &= |V_i|/2^{m_i} + |W|. \end{aligned} \tag{17}$$

In phase two of the i^{th} round, the tags in Y_{i+1} use a time frame of $\frac{1}{\ln 2} \times |Y_{i+1}|$ slots to report their presence. After receiving the responses, the reader builds a filtering vector, denoted as $FV_i(Y_{i+1})$. After the filtration with $FV_i(Y_{i+1})$, the size of the search result X_{i+1} is

$$\begin{aligned} |X_{i+1}| &= |U_i| \times P_{FV} + |W| \\ &\approx |U_i|/2 + |W| \\ &= (|X_i| + |W|)/2. \end{aligned} \tag{18}$$

We denote the transmission time of the i^{th} round by $f(m_i)$, which can be expressed as:

$$\begin{aligned} f(m_i) &= \frac{1}{\ln 2} \times m_i \times |X_i| \times t_s + \frac{1}{\ln 2} \times |Y_{i+1}| \times t_s \\ &= \frac{t_s}{\ln 2} \left(m_i |X_i| + (|V_i|/2^{m_i} + |W|)\right). \end{aligned} \tag{19}$$

To find the value of m_i that minimizes $f(m_i)$, we take the first order derivative and set the right side to zero.

$$\frac{df(m_i)}{dm_i} = \frac{t_s}{\ln 2} \left(|X_i| - \ln 2 |V_i|/2^{m_i}\right) = 0 \tag{20}$$

Hence, the value of $f(m_i)$ is minimized when

$$m_i = \frac{\ln(\ln 2 |V_i|/|X_i|)}{\ln 2}. \tag{21}$$

Because m_i cannot be a negative number, we reset $m_i = 0$ if $\frac{\ln(\ln 2 |V_i|/|X_i|)}{\ln 2} < 0$. Furthermore, m_i must be an integer. If $\frac{\ln(\ln 2 |V_i|/|X_i|)}{\ln 2}$ is not an integer, we round m_i either to the ceiling or to the floor, depending on which one results in a smaller value of $f(m_i)$.

For now, we assume that we know $|W|$ and $|Y|$ in our computation of m_i. Later we will show how to estimate these values on the fly in execution of each round of our protocol. Initially, $|X_1|$ $(= |X|)$ is known. $|V_1|$ can be calculated from (15). Hence, the value of m_1 can be directly computed from (21). After that, we can estimate $|Y_2|$, $|X_2|$, and $|V_2|$ based on (17), (18), and (15), respectively. From $|X_2|$ and $|V_2|$, we can calculate the value m_2. Following the same procedure, we can iteratively compute all values of m_i for $1 \leq i \leq K$.

We find it often happens that the m_i sequence has several consecutive zeros at the end, that is, $\exists p < K$, $m_i = 0$ for $i \in [p, K]$. In this case, we may be able to further optimize the value of m_p with a slight adjustment. We first explain the reason for $m_p = 0$: It costs some time for the reader to broadcast a filtering vector in phase one of the p^{th} round. It is true that this filtering vector can reduce set Y_p, thereby reducing the frame size of phase two of the p^{th} round. However, if the time cost of sending the filtering vector cannot be compensated by the time reduction of phase two, it will be better off to remove this filtering vector by setting $m_p = 0$. (This situation typically happens near the end of the m_i sequence because the number of unwanted tags in the remaining candidate set is already very small.) But if all values of m_i in the subsequent rounds (after m_p) are zeros, increasing m_p to a non-zero value m'_p may help reduce the transmission time of phase two of the subsequent rounds, and the total time reduction may compensate the time cost of sending those m'_p filtering vectors, or even reduce the overall transmission time.

Consider the transmission time of these $(K - p + 1)$ rounds as a whole, denoted by $G(m'_p, p)$. It is easy to know

$$G(m'_p, p) = \left(\frac{m'_p}{\ln 2}|X_p| + \frac{K - p + 1}{\ln 2}\left(\frac{|V_p|}{2^{m'_p}} + |W|\right)\right) t_s. \tag{22}$$

To minimize $G(m'_p, p)$, we have

$$m'_p = \begin{cases} 0 & \text{if } \gamma < 0 \\ \gamma & \text{if } \gamma \geq 0 \end{cases} \tag{23}$$

where $\gamma = \frac{\ln(\ln 2 (K-p+1)|V_p|/|X_p|)}{\ln 2}$. As a result, m_p is updated to m'_p, while other m_i, where $i \neq p$, remain unchanged.

Here, we give an example to illustrate how to calculate the values of m_i. Suppose $|X| = 5,000$, $|Y| = 50,000$, $|W| = 500$, and $P_{REQ} = 0.001$, so $K = \lceil \frac{-\ln 0.001}{\ln 2} \rceil = 10$. Using (21), we can calculate the values from m_1 to m_{10}. The result is listed in Table I. There is a sequence of zeros from m_7 to m_{10}. Thus, we can make an improvement using (23), and the optimized result is shown in Table II.

m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}
3	1	0	1	0	1	0	0	0	0

TABLE I
THE INITIAL VALUES OF m_i.

m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}
3	1	0	1	0	1	2	0	0	0

TABLE II
THE OPTIMIZED VALUES OF m_i.

E. Iterative Tag Search Protocol

Having calculated the values of m_i, we can present our iterative tag search protocol (ITSP) based on the generalized approach in Section IV-C. The protocol consists of K iterative rounds. Each round consists of two phases. Consider the i^{th} round, where $1 \leq i \leq K$.

1) Phase one: The RFID reader constructs m_i filtering vectors for X_i using m_i hash functions. According to (11), we set the size L_{X_i} of each filtering vector as

$$L_{X_i} = \frac{1}{\ln 2} \times |X_i| = \frac{1}{\ln 2} \left(|U_i| + |W| \right). \tag{24}$$

The RFID reader then broadcasts those filtering vectors one by one. Once receiving a filtering vector, each tag in Y_i maps its ID to a bit in the filtering vector using the same hash function that the reader uses to construct the filter. The tag checks whether this bit is '1'. If so, it remains a candidate tag; otherwise, it is excluded as a non-candidate tag and drops out of the search process immediately. The set of remaining candidate tags is Y_{i+1}.

From (12), we know that the false positive probability after using m_i filtering vectors is $(P_{FV})^{m_i} \approx (1/2)^{m_i}$. Therefore, $|Y_{i+1}| = |V_i| \times (P_{FV})^{m_i} + |W| \approx |V_i|/2^{m_i} + |W|$.

2) Phase two: The reader broadcasts the frame size $L_{Y_{i+1}}$ of phase two to the tags, where

$$L_{Y_{i+1}} = \frac{1}{\ln 2} \times |Y_{i+1}| = \frac{1}{\ln 2} \left(|V_i|/2^{m_i} + |W| \right). \tag{25}$$

After receiving $L_{Y_{i+1}}$, each tag in Y_{i+1} randomly maps its ID to a slot in the time frame using a hash function and transmits a one-bit short response to the reader in that slot. Based on the observed state (busy or empty) of the slots in the time frame, the reader builds a filtering vector, which is used to filter non-candidates from X_i. The number of tags in the search result X_{i+1} of this round is about $|X_{i+1}| = |U_i| \times P_{FV} + |W| \approx |U_i|/2 + |W|$.

The overall transmission time of all K rounds in the ITSP is

$$T_{ITSP} = \sum_{i=i}^{K} (m_i \times L_{X_i} + L_{Y_{i+1}}) \times t_s. \tag{26}$$

F. Cardinality Estimation

Recall from Section IV-D that we must know the values of $|X_i|$, $|W|$ and $|V_i|$ to determine m_i and $L_{Y_{i+1}}$. For the purpose of accuracy, we may estimate $|X_i|$, $|W|$ and $|V_i|$ in every round, and then recalculate subsequent m_i sequence and $L_{Y_{i+1}}$. It is trivial to find the value of $|X_i|$ by counting the number of tags in the search result of the $(i-1)^{th}$ round. Meanwhile, we know $|V_i| = |Y_i| - |W|$. Therefore, we only need to estimate $|W|$ and $|Y_i|$.

Besides serving as a filter, a filtering vector can also be used for cardinality estimation, a feature that is not exploited in [6]. Since no filtering vector is available at the very beginning, the first round of the the ITSP should be treated separately: We may use the efficient cardinality estimation protocol ART proposed in [11] to estimate $|Y|$ (i.e., $|Y_1|$) if its value is not known at first. As for $|W|$, it is initially assumed to be $min\{|X|, |Y|\}$.

Next, we can take advantage of the filtering vector built in phase two of the $(i-1)^{th}$ $(i \geq 2)$ round to estimate $|W|$ and $|Y_i|$ without any extra transmission expenditure. The estimation process is as follows: First, counting the actual number of '1' bits in the filtering vector, denoted as N_1^*, we know the real false positive ratio, denoted by P_{i-1}^*, using this filtering vector is

$$P_{i-1}^* = N_1^* / L_{Y_i}. \tag{27}$$

Meanwhile, we can record the number of tags in the search results before and after the $(i-1)^{th}$ round, i.e., $|X_{i-1}|$ and $|X_i|$, respectively. We have $|X_{i-1}| = |U_{i-1}| + |W|$, $|X_i| = |U_i| + |W|$, and $|U_i| \approx |U_{i-1}| \times P_{i-1}^*$. Therefore,

$$|W| \approx \frac{|X_i| - |X_{i-1}| \times P_{i-1}^*}{1 - P_{i-1}^*}. \tag{28}$$

Second, using the same filtering vector, we can estimate the value of $|Y_i|$ as well. After mapping all tags in Y_i to the filtering vector, the probability that a certain bit in the vector remains '0' (i.e., no tag maps its ID to this bit) is

$$P_0 = \left(1 - \frac{1}{L_{Y_i}}\right)^{|Y_i|} \approx e^{-\frac{|Y_i|}{L_{Y_i}}}. \tag{29}$$

Let N_0 be the number of '0' bits in the filtering vector. Each slot of frame L_{Y_i} independently has probability P_0 of being '0'. So $N_0 \sim B(L_{Y_i}, P_0)$, and

$$E(N_0) = L_{Y_i} \times P_0 \approx L_{Y_i} \times e^{-\frac{|Y_i|}{L_{Y_i}}}. \tag{30}$$

Also, we can count the filtering vector to obtain the actual number of '0' bits, denoted as N_0^*. When L_{Y_i} is large, we have $N_0^* \approx E(N_0)$, namely, $N_0^* \approx L_{Y_i} \times e^{-\frac{|Y_i|}{L_{Y_i}}}$, so

$$|Y_i| \approx -L_{Y_i} \ln \frac{N_0^*}{L_{Y_i}}. \tag{31}$$

G. Additional Filtering Vectors

Estimation may have error. Using the values of m_i and L_{Y_i} computed from estimated $|W|$ and $|Y_i|$, a direct consequence is that the actual false positive ratio, denoted as P_T, can be greater than the requirement P_{REQ}. Fortunately, from (27), the reader is able to compute the actual false positive ratio P_i^*, $1 \leq i \leq k$, of each filtering vector received in phase two of the ITSP. Thus, we must have

$$P_T = \prod_{1}^{K} P_i^*. \tag{32}$$

If $P_T > P_{REQ}$, our protocol will automatically add additional filtering vectors to further filter X_{K+1} until $P_T \leq P_{REQ}$.

H. False positive ratio Requirement

Users can set their false positive requirements P_{REQ} arbitrarily. We observe that it may be desirable to set the value of P_{REQ} relative to $|W|$. For example, consider a RFID system with $|X| = 20,000$. If $|W| = 10,000$, $P_{REQ} = 0.01$ may be good enough because the number of false positives is about $(|X| - |W|) \times P_{REQ} = 100$, which is much fewer than $|W|$. However, if $|W| = 10$, $P_{REQ} = 0.01$ may become unacceptable since $(|X| - |W|) \times P_{REQ} \approx 200 \gg |W|$. It is desirable to set the value of P_{REQ} such that the number of false positives in the search result is much smaller than $|W|$, namely, $(|X| - |W|) \times P_{REQ} \leq \frac{1}{\lambda}|W|$, where λ is an integer constant. Thus, we have

$$P_{REQ} \leq \frac{|W|}{\lambda\,(|X| - |W|)}. \tag{33}$$

Our protocol is able to set the value of P_{REQ} using the estimated value of $|W|$ after the first round.

I. Hash Functions

To keep the complexity of a tag's circuit low, we only implement one uniform hash function $h(\cdot)$, and use it to simulate multiple independent hash functions: In phase one of the i^{th} ITSP round, we use $h(\cdot)$ and m_i unique hash seeds $\{s_1, s_2, \cdots, s_{m_i}\}$ to achieve m_i independent hash outputs. Thus, a tag id is mapped to position $(h(id \oplus s_1) \bmod L_{X_i})$, $(h(id \oplus s_2) \bmod L_{X_i})$, $\cdots$, $(h(id \oplus s_{m_i}) \bmod L_{X_i})$ in each filtering vector, respectively. Each hash seed, together with its corresponding filtering vector, will be broadcasted to the tags. In phase two of the i^{th} round, the reader generates a new hash seed r and sends it to the remaining candidate tags. Each candidate tag in Y_{i+1} maps its id to the slot of index $\big(h(id \oplus r) \bmod L_{Y_{i+1}}\big)$, and waits to transmit a one-bit short response to the reader in that slot.

V. Performance Evaluation

A. Simulation Setting and Performance Metrics

The simulation setting is based on the EPC global Class-1 Gen-2 standard [7]. In the ITSP, filtering vectors constitute most of the overall transmission overhead, while other transmission cost, such as estimation of $|Y|$ and transmission of hash seeds, is comparatively negligible [6]. Consequently, the key metric concerning the efficiency of the ITSP is the total size of filtering vectors, and (26) can be used for calculating the search time required by the ITSP.

After the search process is completed, we will calculate the false positive ratio P_{FP} using $P_{FP} = \frac{|W^* - W|}{|X - W|}$, where W^* is the set of tags in the search result. P_{FP} will be compared with P_{REQ} to see whether the search result meets the false positive ratio requirement.

In our simulation, we set both the $R \Rightarrow T$ transmission rate and the $T \Rightarrow R$ transmission to be 100kbps. Accordingly, $t_s = \frac{1bit}{100kbps} = 10^{-5}$sec.

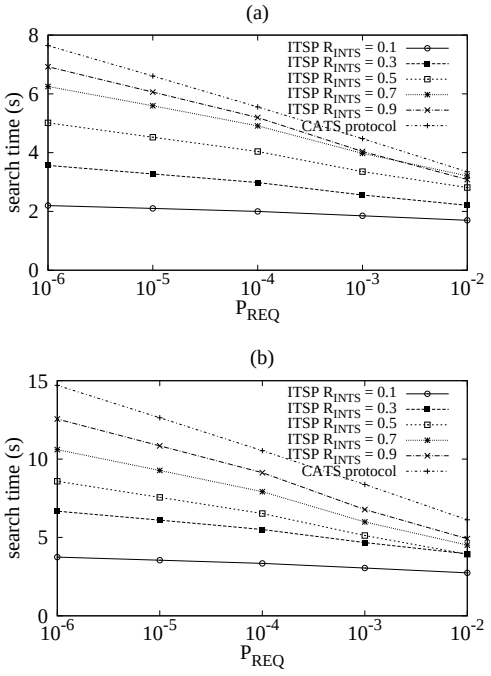

Fig. 2. Relationship between search time and P_{REQ}. Parameter setting: $|Y| = 50,000$; (a) $|X| = 20,000$, (b) $|X| = 40,000$.

B. Simulation Results

1) Performance comparison: In this subsection, we will evaluate the performance of our protocol, using the protocols mentioned in Section III: the CATS protocol, the baseline protocol, and the tag identification protocols, as benchmarks for comparison.

In the simulation, we assume X and Y are already known. We set $P_{REQ} = 0.001$, $|Y| = 50,000$, vary $|X|$ from 5,000 to 640,000, and let R_{INTS} = 0.1, 0.3, 0.5, 0.7, 0.9. For simplicity, we assume $t_{id} = 96 \times t_s$, during which a 96-bit ID is transmitted. Table III shows the number of t_s slots needed by the protocols under different parameter settings. As the CATS protocol is designed for applications where $|X| < |Y|$, it may not always work when $|X| \geq |Y|$ (N.A. in the table).

From Table III, we observe that when R_{INTS} is small, the ITSP performs much better than the CATS protocol, the baseline protocol, and the tag identification protocol. For example, when $R_{INTS} = 0.1$, the ITSP reduces the search time of the CATS protocol, the baseline protocol and any tag identification protocol, by as much as 90.0%, 98.8%, and 99.5%, respectively. As we increase R_{INTS}, the gap between the performance of the ITSP and the performance of the CATS gradually shrinks. The CATS protocol performs poorly when $|X| \geq |Y|$, and more seriously, it does not work when $|X| \gg |Y|$ due to the failure of (5). In contrast, the ITSP can work efficiently in all cases. In practice, the wanted tags may be spatially distributed in many different RFID systems (e.g., warehouses in the example we use in the introduction), and thus R_{INTS} can be small. As a result, the ITSP is a far better protocol for solving the tag search problem in such practical scenarios.

Another performance issue we want to investigate is the

$\|X\|$	ITSP R_{INTS}=0.1	ITSP R_{INTS}=0.3	ITSP R_{INTS}=0.5	ITSP R_{INTS}=0.7	ITSP R_{INTS}=0.9	CATS	Baseline	Tag identification
5,000	61,463	96,989	105,828	108,346	124,553	126,370	485,000	13,047,752
10,000	108,017	145,553	206,709	199,586	231,236	238,313	970,000	13,047,752
20,000	185,204	255,898	335,426	397,462	403,954	447,772	1,940,000	13,047,752
40,000	304,767	467,433	512,156	598,718	678,066	837,837	3,880,000	13,047,752
80,000	414,686	590,150	656,426	721,347	721,347	1,560,259	7,760,000	13,047,752
160,000	472,677	630,669	721,347	721,347	721,347	2,889,689	15,520,000	13,047,752
320,000	529,835	668,794	721,347	721,347	721,347	5,317,715	31,040,000	13,047,752
640,000	573,270	696,015	721,347	721,347	721,347	N.A.	62,080,000	13,047,752

TABLE III
PERFORMANCE COMPARISON OF THE ITSP, THE CATS PROTOCOL, THE BASELINE AND TAG IDENTIFICATION PROTOCOLS.

relationship between the search time and P_{REQ}. We set $|X| = 20,000$ or $40,000$, $|Y| = 50,000$, vary R_{INTS} from 0.1 to 0.9, and vary P_{REQ} from 10^{-6} to 10^{-2}. Fig. 2 compares the search times required by the CATS and the ITSP under different false positive ratio requirements. Generally speaking, the gap between the search time required by the ITSP and the search time by the CATS keeps getting larger with the decrease of P_{REQ}, particularly when $RINTS$ is small. For example, in Fig. 2 (b), when $P_{REQ} = 10^{-2}$ and $R_{INTS} = 0.1$, the search time by the ITSP is about one half of the time by the CATS; when we reduce P_{REQ} to 10^{-6}, the time by the ITSP becomes about one fourth of the time by the CATS. The reason is as follows: When R_{INTS} is small, $|W|$ is small and most tags in X and Y are non-candidates. After several ITSP rounds, as many non-candidates are filtered out iteratively, the size of filtering vectors decreases exponentially and therefore subsequent ITSP rounds do not cause much extra time cost. This merit makes the ITSP particularly applicable in cases where the false positive ratio requirement is very strict, requiring many ITSP rounds. On the contrary, the CATS protocol does not have this capability of exploiting low R_{INTS} values.

2) False positive ratio: Next, we examine whether the search results after execution of the ITSP will meet the requirement of P_{REQ}. Recall that the false positive ratio is defined as $P_{FP} = \frac{|W^* - W|}{|X - W|}$. We use (33) with $\lambda = 10$ to set the value of P_{REQ}. The ITSP is tested under three different parameter settings:

(a) $|X| = 5,000$, $|Y| = 50,000$, and R_{INTS} varies from 0.1 to 0.9 ($|W|$ varies from 500 to 4,500). According to (33), $P_{REQ} \leq \frac{500}{10\times(5,000-500)} \approx 0.01111$. We set $P_{REQ} = 10^{-2}$.

(b) $|X| = 20,000$, $|Y| = 50,000$, and R_{INTS} varies from 0.01 to 0.9 ($|W|$ varies from 200 to 18,000). According to (33), $P_{REQ} \leq \frac{200}{10\times(20,000-200)} \approx 0.00101$. We set $P_{REQ} = 10^{-3}$.

(c) $|X| = 80,000$, $|Y| = 50,000$, and R_{INTS} varies from 0.01 to 0.9 ($|W|$ varies from 500 to 45,000). According to (33), $P_{REQ} \leq \frac{500}{10\times(80,000-500)} \approx 0.00063$. We set $P_{REQ} = 10^{-4}$.

For each parameter setting, the simulation repeats 500 times to obtain the average false positive ratio.

Fig. 3 shows the simulation results. In (a), (b), and (c), we can see that the average P_{FP} is always smaller than the corresponding P_{REQ}. Hence, the search results using the ITSP meet the prescribed requirement of false positive ratio in the average sense.

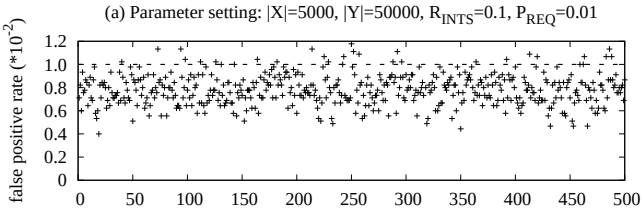

Fig. 4. False positive ratio in the search result after running the ITSP. The bold horizontal line in each figure stands for P_{REQ}. X-coordinate marks the ordinal of simulations.

If we look into the details of individual simulations, we find that a small fraction of simulation runs have P_{FP} beyond P_{REQ}. For example, Fig. (4) depicts the results of 500 runs with $|X| = 5,000$, $|Y| = 50,000$, $|W| = 500$ and $P_{REQ} = 10^{-2}$. There are about 5% runs having $P_{FP} > P_{REQ}$, but that does not come as a surprise because the false positive ratio in the context of filtering vectors (ITSP) or Bloom filters (CATS) is defined in a probability way: The probability for each tag in $X - W$ to be misclassified as one in W is no greater than P_{REQ}. This probabilistic definition enforces a requirement P_{REQ} in an average sense, but not for each individual run.

VI. RELATED WORK

The basic technologies for RFID have been around for a long time. In the past, much research concentrated on two fronts: (1) physical-layer technologies for transmitting IDs from tags to an RFID reader more reliably, over a longer distance, and using less energy; (2) MAC-layer technologies for improving the rate at which a reader can collect IDs from tags. Tag identification protocols, which read IDs from all tags in a RFID system, mainly fall into two categories. One is *ALOHA-based* [8], [9], [12]–[15], and the other is *tree-cased* [16]–[19]. The *ALOHA-based protocols* work as follows: The reader broadcasts a query request. With a certain probability, each tag chooses a time slot in the current frame to transmit its ID. If there is a collision and the reader does not acknowledge positively, the tag will continue participating in the next frame. This process repeats until all tag IDs are read successfully. The *tree-based protocols* organize all IDs in a tree of ID prefixes [16]–[19]. Each in-tree prefix has two child nodes that have one additional bit, '0' or

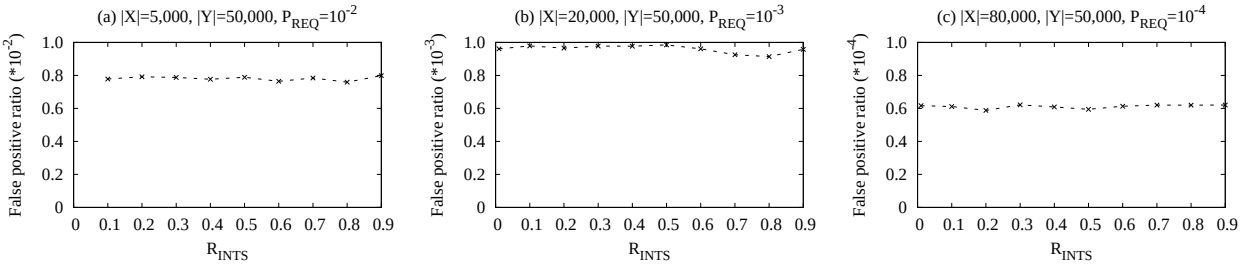

Fig. 3. False positive ratio after running the ITSP.

'1'. The tag IDs are leaves of the tree. The RFID reader walks through the tree, and requires tags with matching prefixes to transmit their IDs.

Another related research topic is cardinality estimation in an RFID system. Kodialam and Nandagopal [20] estimate the number of tags based on the probabilistic counting methods [21]. The same authors propose a non-biased follow-up work in [22]. Han et al. [23] improve the performance of [20]. Qian et al. [24] present the Lottery-Frame scheme (LoF) for estimating the number of tags in a multiple-reader scenario. The work in [25] uses the maximum likelihood method. Sheng et al. design two probabilistic algorithms to identify large tag groups [4].

VII. Conclusions

This paper studies the tag search problem in large-scale RFID systems. To improve time efficiency and eliminate limitation of the prior tag search protocol (CATS), we propose an iterative tag search protocol (ITSP) based on a new technique called filtering vectors. The main contributions of our work are summarized as follows: (1) The iterative method of ITSP based on filtering vectors is very effective in reducing the amount of information to be exchanged between tags and the reader, and consequently saves time in the searching process; (2) the ITSP performs much better than the existing solutions; (3) the ITSP works well under all system conditions, particularly under conditions of $|X| \gg |Y|$ when CATS no longer works well or even fails.

VIII. Acknowledgments

This work was supported in part by the National Science Foundation under grants CPS-0931969 and CNS-1147813.

References

[1] L. Ni, Y. Liu, and Y. C. Lau, "Landmarc: Indoor Location Sensing using Active RFID," *Proc. of IEEE PerCom*, 2003.

[2] Y. Li and X. Ding, "Proecting RFID Communications in Supply Chains," *Proc. of IEEE ASIACCS*, 2007.

[3] C. H. Lee and C. W. Chung, "Efficient Storage Scheme and Query Processing for Supply Chain management Using RFID," *Proc. ACM SIGMOD*, 2008.

[4] B. Sheng, C. Tan, Q. Li, and W. Mao, "Finding Popular Categories for RFID Tags," *Proc. of ACM Mobihoc*, 2008.

[5] "AEI Technology," *Softrail. http://www.aeitag.com/aeirfidtec.html*, October 2008.

[6] Y. Zheng and M. Li, "Fast Tag Searching Protocol for Large-Scale RFID System," *Proc. IEEE ICNP*, October 2011.

[7] "EPC Radio-Frequency Identity Protocols Class-1 Gen-2 UHF RFID Protocol for Communications at 860MHz-960MHz, EPCglobal," *http://www.epcglobalinc.org/uhfclg2*, April 2011.

[8] J. R. Cha and J. H. Kim, "Dynamic Framed Slotted ALOHA Algorithms Using Fast Tag Estimation Method for RFID Systems," *Proc. of IEEE Consumer Communications and Networking Conference(CCNC)*, January 2006.

[9] S. Lee, S. Joo, and C. Lee, "An Enhanced Dynamic Framed Slotted ALOHA Algorithm for RFID Tag Identification," *Proc. of IEEE MobiQuitous*, 2005.

[10] A. Broder and M. Mitzenmacher, "Network Applications of Bloom Filters: A Survey," *Internet Math*, vol. 1, no. 4, pp. 485–509, 2003.

[11] M. Shahzad and A. Liu, "Every Bit Counts - Fast and Scalable RFID Estimation," *Proc. of ACM MOBICOM*, 2012.

[12] H. Vogt, "Efficient Object Identification with Passive RFID Tags," *Proc. of IEEE PerCom*, April 2002.

[13] B. Sheng, Q. Li, and W. Mao, "Efficient Continuous Scanning in RFID Systems," *Proc. of IEEE INFOCOM*, 2010.

[14] V. Sarangan, M. R. Devarapalli, and S. Radhakrishnan, "A Framework for Fast RFID Tag Reading in Static and Mobile Environments," *The International Journal of Computer and Telecommunications Networking*, vol. 52, no. 5, pp. 1058–1073, 2008.

[15] B. Zhen, M. Kobayashi, and M. Shimizu, "Framed ALOHA for Multiple RFID Objects identification," *IEICE Transactions on Communications*, Mar 2005.

[16] "Information technology automatic identification and data capture techniques C radio frequency identification for item management air interface - part 6: parameters for air interface communications at 860-960 MHz," Final Draft International Standard ISO 18000-6, November 2003.

[17] J. Myung and W. Lee, "Adaptive Splitting Protocols for RFID Tag Collision Arbitration," *Proc. of ACM MOBIHOC*, May 2006.

[18] N. Bhandari, A. Sahoo, and S. Iyer, "Intelligent Query Tree (IQT) Protocol to Improve RFID Tag Read Efficiency," *Proc. of IEEE International Conference on Information Technology (ICIT)*, December 2006.

[19] F. Zhou, C. Chen, D. Jin, C. Huang, and H. Min, "Evaluating and Optimizing Power Consumption of Anti-collision Protocols for Applications in RFID Systems," *Proc. of ACM International Symposium on Low Power Electronics and Design (ISLPED)*, August 2004.

[20] M. Kodialam and T. Nandagopal, "Fast and Reliable Estimation Schemes in RFID Systems," *Proc. of ACM MobiCom*, September 2006.

[21] K. Huang, B. Vander-Zanden, and H. Taylor, "A Linear-time Probabilistic Counting Algorithm for Database Application," *ACM Transactions on Database Systems*, vol. 15, no. 2, June 1990.

[22] M. Kodialam, T. Nandagopal, and W. Lau, "Anonymous Tracking using RFID tags," *Proc. of IEEE INFOCOM*, 2007.

[23] H. Han, B. Sheng, C. Tan, Q. Li, W. Mao, and S. Lu, "Counting RFID Tags Efficiently and Anonymously," *Proc. of IEEE INFOCOM*, 2010.

[24] C. Qian, H. Ngan, and Y. Liu, "Cardinality Estimation for Large-scale RFID Systems," *Proc. of IEEE PerCom*, 2008.

[25] T. Li, S. Wu, S. Chen, and M. Yang, "Energy Efficient Algorithms for the RFID Estimation Problem," *Proc. of IEEE INFOCOM*, March 2010.