Completely Pinpointing the Missing RFID Tags in A Time-efficient Way

Xiulong Liu, Keqiu Li*, Geyong Min, Yanming Shen, Alex X. Liu, Wenyu Qu

Abstract—Radio Frequency Identification (RFID) technology has been widely used in the inventory management in many scenarios, e.g., warehouses, retail stores, hospitals, etc. This paper investigates a challenging problem of complete identification of missing tags in large-scale RFID systems. Although this problem has attracted extensive attention from academy and industry, the existing work can hardly satisfy the stringent real-time requirements. In this paper, a Slot Filter-based Missing Tag Identification (SFMTI) protocol is proposed to reconcile some expected collision slots into singleton slots and filter out the expected empty slots as well as the unreconcilable collision slots, thereby achieving the better time-efficiency. We also present the theoretical analysis of the system parameters to minimize the execution time of the proposed SFMTI. We then propose a cost-effective method to extend SFMTI to the multi-reader scenarios. The extensive simulation results demonstrate that the proposed SFMTI protocol outperforms the most promising Iterative ID-free Protocol (IIP) by reducing nearly 45% of the required execution time, and is just within a factor of 1.18 from the lower bound of the minimum execution time.

Index Terms—RFID systems, Missing Tag, Complete Pinpointing, Time Efficiency, Optimization.

1 Introduction

Compared to the traditional barcode technology, the newly emerging Radio Frequency Identification (RFID) technology possesses many attractive advantages, e.g., (1) line of sight is not required for reading; (2) multiple items can be read with a single scan; (3) tags can be read from a relatively long distance. As a result, RFID systems are being increasingly used in various applications such as localization [1], [2], [3], supply chain management [4], [5], [6], warehouse management [7], [8], [9], etc. An RFID system usually consists of RFID readers and a large number of tags. RFID tags are labeled in designated objects where each tag has a small size of memory to store its unique ID and some other information (e.g., product price, expiry date, personal information, etc). There are two types of tags [10]: (1) passive tags that are powered up by harvesting the radio frequency energy from readers and have communication range often less than 20 feet; (2) active tags that have their own power sources and have relatively longer communication ranges. A reader has a dedicated power source with significant computing power. It transmits a query to a set of tags, and the tags respond over a shared wireless medium.

- *X. Liu, K. Li and Y. Shen are with the School of Computer Science and Technology, Dalian University of Technology, No 2, Linggong Road, Dalian 116023, China. E-mail: keqiu@dlut.edu.cn.*
- *G. Min is with the Department of Computing, University of Bradford, Bradford, BD7 1DP, United Kingdom. E-mail: G.Min@brad.ac.uk.*
- *Alex X. Liu is with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, U.S.A.. E-mail: alexliu@cse.msu.edu.*
- *W. Qu is with the School of Information and Technology, Dalian Maritime University, Dalian 116026, China. E-mail: eunice.qu@gmail.com.*

This study investigates the challenging problem of complete identification of missing tags in large-scale RFID systems. For example, imagine a large warehouse with tens of thousands of items (e.g., refrigerators, televisions, bicycles, etc.). One of the most fundamental tasks is to monitor whether some items are missing (due to management fault, theft, etc.). A traditional method is to manually check them one by one, which suffers two significant drawbacks. (1) *Poor accuracy*: some items may be blocked behind others. If a thief deliberately stolen some blocked items and replaced back the items that were in front to conceal this theft, the warehouse manager can hardly discover that these items are missing. (2) *Long checking interval*: obviously this manual checking process is seriously laborious and thus cannot be conducted frequently (the manager needs to have a rest after one checking). The interval between any two checking processes is usually long enough for the thief to escape. The thief has already fled, even if the manager could find this theft event. As an emerging technology, RFID could be used in the above monitoring application (if an RFID tag is missing, the corresponding bound item is also treated as missing) to tackle these drawbacks of the manual checking method. (1) RFID is a wireless communication technology, it does not require the line of sight. The presence of the tagged items can be accurately checked even if they are blocked behind other items. (2) This RFID-based monitoring processes can be performed frequently, and the the interval of any two consecutive checking processes can be as short as possible. As a result, the theft event can be discovered in time. Similar applications also exist in other scenarios, e.g., retail stores, hospitals, prisons, etc.

Although this problem has attracted extensive attention from academy and industry, the existing work still can hardly satisfy the stringent real-time requirements. To the best of our knowledge, the most promising missing tag identification is the Iterative ID-free Protocol (IIP) proposed in [8]. IIP is a variant of Framed Slotted Aloha protocol. All the tags use their IDs and a hash function to pseudo-randomly pick a slot in the time frame to respond for announcing its presence. Because the reader knows all the tag IDs as well as the used hash function, it is able to predict the empty slot that is expected to contain no tag response; the singleton slot that is expected to contain one and only one tag response; the collision slot that is expected to contain two or more tag responses. In fact, some of the tags may be missing, which makes it possible that the actual status of a slot different from its expected status. By comparing the observed slot statuses with the expected statuses, the reader is able to identify the missing tags. Specifically, if an expected singleton slot turns out to be an empty one, the reader asserts that the tag corresponding to this slot is missing. However, the expected empty slots and the expected collision slots are not used and wasted, which decreases the efficiency of IIP. Although the authors investigated a method [8] to turn some of the expected collision slots into the expected singleton slots in order to improve the efficiency of IIP, the effect of collision reconcilement is limited (only a small fraction of expected collision slots could be reconciled into the singleton ones). Moreover, the deficiency due to the expected empty slots are not noticed and still wasted directly. According to the theoretical analysis in [8], we find that the expected empty slots and expected collision slots still account for nearly 48%. That is, IIP is of low time-efficiency, and has a large room to be improved.

To approach a real-time missing tag identification, this paper proposes a Slot Filter-based Missing Tag Identification (SFMTI) protocol, which possesses two significant advantages over IIP. (1) a more effective collision reconcilement method is investigated, which can turn more collision slots into the expected singleton ones; (2) the expected empty slots and the unreconcilable collision slots are filtered out and not executed for time-efficiency. In the proposed SFMTI protocol, all the executed slots are expected to be singleton, and thus each of them is useful for the missing tag identification. The above two advantages precisely relieve the drawbacks of IIP. This paper theoretically analyze the parameter settings to minimize the execution time of the proposed SFMTI. The extensive simulation results demonstrate that this new SFMTI protocol outperforms the most promising IIP scheme by reducing 45% of the required execution time.

The major contributions of this paper are summarized as follows:

1) A new SFMTI protocol is proposed to efficiently identify the missing tags, by reconciling collision slots into the expected singleton slots; and filtering out the expected empty slots as well as the unreconcilable collision slots.
2) This paper theoretically analyzes the performance of the proposed protocol, and optimizes the parameter settings to achieve the best time-efficiency.
3) A cost-effective method is proposed to extend the SFMTI protocol to the multi-reader scenarios, where the readers could perform the missing tag identification in a parallel mode.
4) We conduct extensive simulation experiments to evaluate the performance of the proposed protocol. The simulation results match well with the analytical results, and demonstrate the proposed protocol performs much better in terms of execution time than the currently most promising protocol.

The rest of this paper is organized as follows. Section 2 surveys the related work. Section 3 presents the system model and problem description. The detailed SFMTI protocol is presented in Section 4. Section 5 evaluates the performance of the SFMTI protocol. Finally, Section 6 concludes this paper.

2 Related Work

In the infancy stage of the RFID technology, the *tag collection* problem attracted extensive attention, which is to collect the IDs from a large number of tags as quickly as possible. The solutions to *tag collection* problem are generally classified into two categories: Aloha-based protocols [11], [12], [13] and Tree-based protocols [14], [15], [16].

The former works as follows. The reader first tells the frame size f and a random number R to the tags in its vicinity. Each tag then uses the received parameters f, R and its ID to select a slot in the frame by calculating a hash function $h(ID, R) \mod f$ whose result is in $[0, f-1]$ following a uniform distribution. Then each tag responds its ID in the selected slot. In any slot, if one and only one tag responds, the reader ia able to successfully get the ID information of that tag. This type of slot is referred to as a singleton slot. An RFID tag that is successfully collected in a singleton slot will keep silent for the rest of the collection process. If multiple tags simultaneously transmit their IDs in a common collision slot, the responses are garbled due to collision and thus retransmission is required. The collection process does not terminate until all the tags are collected.

On the other hand, a Tree-based protocol [14], [15], [16], [17] organizes all IDs in a binary tree where the height of this tree is equal to the length of a tag ID. Each left branch of the tree is marked by '0' and each right branch by '1'. A reader first queries '0' and all

the tags whose IDs start with '0' respond. If result of the query is a successful read (i.e., exactly one tag responds) or an empty read (i.e., no tag responds), the reader queries '1' and all the tags whose IDs start with '1' respond. If the result of the query is a collision, the reader generates two new query strings by appending a '0' and a '1' to the previous query string and queries the tags with these new query strings. All the tags whose IDs start with the new query string respond. This process continues until all the tags have been identified.

In recent years, RFID technology is widely used in many monitoring applications, where the missing tag problem is yet under-investigated by the research community. The missing tag problem can be generally classified into two categories: (1) the missing tag detection focuses on detecting whether any RFID tags are missing instead of exactly identifying which ones are missing. (2) the missing tag identification concentrates on identifying the exact missing tags.

Obviously, the solutions to *tag collection* problem can solve the *missing tag* problem by collecting all the tag IDs and then comparing the collected IDs with the ID information stored in the database. However, these methods are seriously time-consuming because of recollecting a large number of reductant IDs.

In order to efficiently address the problem of missing tag detection, Tan *et al.* proposed the Trust Reader Protocol (TRP) to detect the missing-tag event with a predefined probability α when the number of the missing tags exceeds m (a tolerance threshold) [18]. To improve the time-efficiency and energy-efficiency of TRP, Luo *et al.* introduced the sampling idea, and thus proposed the Efficient Missing-tag Detection (EMD) protocol, where they used the detection result on the sampled tags to probabilistically reflect the whole intactness of RFID systems [19]. Based on their prior work, Luo *et al.* proposed a multi-hash approach to further improve the performance of the missing tag detection protocol in [20]. The above schemes can only detect the missing tag event but cannot exactly find out which tags are missing and thus fails to provide the details of the missing tags.

The problem of missing tag is also of great practical importance, and many efforts have been made to address this problem. Unfortunately, the existing solutions can hardly satisfy the stringent real-time requirements. The most promising missing tag identification protocol is the Iterative ID-free Protocol (IIP) proposed in [8]. Its basic principle has been described in Section I, and its deficiency is also analyzed. In [21], Zhang *et al.* investigated the problem of missing tag identification in the multi-reader scenarios, where all the readers perform *synchronized and parallel* scans. The authors claimed that their best protocol (i.e., Protocol 3) reduces the time for identifying all the missing tags by up to 75% in comparison to IIP. In fact, the superiority of their protocol over IIP benefits from the cooperation of the readers. In the single reader scenarios (or in the scenarios where the number of readers is small), IIP still runs faster than the Protocol 3 in [21].

3 System Model and Problem Description

3.1 System Model

We consider a large RFID system with a single reader and N tags where all the tags are within the interrogating range of this reader. Please note that, for the purpose of clarity, we first present the SFMTI protocol in the case of a single reader and then extend the use of this protocol in large-scale RFID systems with multiple readers. The tag set is denoted as S_{all}, i.e., $S_{all} = \{t_1, t_2, \ldots, t_i, \ldots, t_N\}$. Each tag, say t_i, has a unique ID_i and is equipped with the same uniform Hash generator $H(\cdot)$. The reader has access to a database that stores the IDs of all tags. The reader communicates with the central computer through a high-speed network link.

3.2 Communication Overview

The reader continuously sends synchronization signals to create a slotted time frame. The interactive communications are in the Reader Talks First (RTF) mode [22], i.e., the reader queries the tags first, and a tag picks a slot to respond according to the reader's commands. Li *et al.* classified the slots into three categories: *tag slots*, *long-response slots* and *short-response slots*, based on their length [8]. The length of a tag slot is denoted as t_{tag}, which allows the transmission of a tag ID (96 bits), either from the reader to the tags or from a tag to the reader. The length of a long-response slot is denoted as t_{long}, which can afford transmitting a long response carrying 10 bits information. The length of a short-response slot is denoted as t_{short}, which allows the transmission of a short-response carrying only one bit information. Based on the specification of the Philips I-Code system [23], the length t_{tag} of *tag slot* is set to 2.4 *ms* (including the wait time between any two consecutive transmissions) for transmission of a tag ID (96 bits) from a tag to a reader or vice versa; the length t_{long} and t_{short} are set to 0.8 *ms* and 0.4 *ms*, respectively [8].

3.3 Problem Statement

Some tags may be missing due to theft, management fault, etc. We do not know which tags are missing or even the number of the missing ones. The problem addressed by this paper is to quickly identify all the missing tags. Obviously, the execution time is the most important performance metric for a missing tag identification protocol. The used notations are summarized in Table 1.

TABLE 1
Notations used in this paper.

Symbols	Descriptions
N	The number of tags in the system
S_{all}	The set of all tags in the system
S_{miss}	The set of the missing tags
ID_i	The ID of Tag t_i
N^*	The number of the tags participating in the current round
f	The length of the filter vector
ρ	A variable given by N^*/f
R	The random number that is fresh in each round
$H(\cdot)$	The Hash generator with a uniform random distribution
e	The natural constant which is approximately equal to 2.71828
C	The expected execution round count

4 The Proposed Slot Filter-based Missing Tag Identification Protocol

This section will first analyze the deficiency of two typical solutions for identifying the missing tags, which inspires the motivation and ideas of the proposed SFMTI protocol. We will then present the SFMTI protocol in detail and investigate how to optimize its parameter settings. After that we propose a cost-effective method to extend the proposed SFMTI protocol to the multi-reader scenarios.

4.1 Motivation

4.1.1 An Ideal Lower Bound

A widely accepted lower bound on the minimum execution time for the problem of missing tag identification was given in [8], which is redescribed as follows. All N tags relay 1-bit short-responses one after another to declare their presence. If the reader does not receive a response as expected, the corresponding tag must be missing. Clearly, one scanning is adequate to verify the presence of all N tags. If the transmission of control information is not considered, the execution time is $N \times t_{short}$, where t_{short} is the time for transmitting a 1-bit response. Although this lower bound is unlikely to be achieved because the reader has to transmit control information to coordinate the protocol execution, it offers a guidance and also a target for any missing tag identification protocols to approach.

4.1.2 A Straightforward Polling Protocol

The most straightforward way for identifying the missing RFID tags is a polling method [8]. As the reader gets full knowledge of all the tag IDs stored in a database, it could request the IDs one by one. A tag responds a short-response if it finds that the current request contains its ID information. Clearly, if the reader receives a response as expected, the requested tag must be present; otherwise, this tag is missing. The execution time of this polling method is $N \times (t_{tag} + t_{short})$, where t_{tag} is the time for transmitting a tag ID. This polling method is far from the the lower bound, and the key reason is that transmission of IDs is extremely time-consuming.

4.1.3 The Most Promising Protocol

The most promising missing tag identification protocol is the Iterative ID-free Protocol (IIP) [8] which is a variant of the classical Framed Slotted Aloha protocol. IIP avoids the transmissions of IDs, which uses the expected singleton slots to verify the presence of the corresponding tags. Specially, if an expected singleton slot turns out to be empty, the corresponding tag must be missing.

However, according to the theoretical analysis in [8], we find that the expected empty slots and the expected collision slots in IIP account for nearly 48%, which degrades its efficiency. Clearly, IIP still has a large room to be improved. Comparing IIP with the ideal lower bound, the gap between them is that IIP contains a large proportion of expected empty slots and collision slots. Hence, the key to approaching the ideal lower bound is how to reduce (or eliminate) the expected empty slots and collision slots, which is a challenging issue.

4.2 Protocol Design

The proposed Slot Filter-based Missing Tag Identification (SFMTI) protocol possesses two main innovation points: (1) reconciling some expected collision slots into the expected singleton slots; (2) avoiding the execution of the expected empty slots and the *un*reconcilable collision slots, which precisely relieves the drawbacks of IIP. The proposed SFMTI includes multiple rounds, each of them consists of three stages: (1) *Collision Slot Reconciling* stage, in which the reader reconciles the expected collision slots and turns some of them into the expected singleton ones; (2) *Slot Filtering* stage, in which the reader creates a filter vector to filter out the expected empty slots and the unreconcilable collision slots. (3) *Presence Verifying* stage, in which the tags that pick the expected singleton slots respond shot-responses to announce their presence. In a round, the presence of some tags are able to be verified. SFMTI repeats for multiple rounds until all the tags are verified. In what follows, we present this protocol in detail.

4.2.1 Collision Slot Reconciling Stage

As aforementioned, the reader gets full knowledge of the tag IDs. Then, it uses a uniform hashing function $H(\cdot)$ and a random number R_1 to map all the IDs to a filter vector, which contains f elements, each element corresponds to a slot. For example, a tag t_i is mapped to the s_i^{th} element of the filter, where $s_i = H(ID_i, R_1) \mod f$. As a result, there are generally three types of slots: the expected empty slot, to which no tag is mapped; the expected singleton slot, to which one and only one tag is mapped; the expected collision slot, to

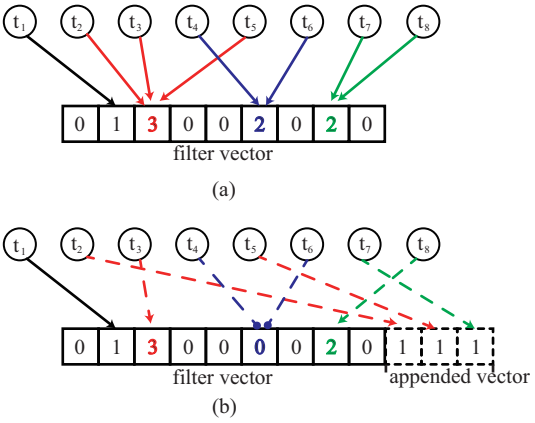

Fig. 1. An example of reconcilement of k-collision slots ($k = 2$ or 3). (a) Mapping all the tags to a filter vector with f elements; (b) Relocating the tags confined in the reconcilable collision slots.

which two or more tags are mapped. To mark these 3 types of slots, the length ℓ of each element in the filter is at least 2 bits. For the clarity of description, we assume $\ell = 2$, i.e., each element in the filter vector has 2 bits. Then the elements corresponding to the expected empty slots are set to '0s'; the elements corresponding to the expected singleton slots are set to '1s'. Let $k-$collision slot denote the slot to which k tags are mapped at the same time. The elements corresponding to the 2-collision slots and 3-collision slots are set to '2s', and '3s', respectively. Intuitively, the larger the k is, the harder the collision reconcilement is. In this paper, we do not try to reconcile the $k-$collision slots ($k \geq 4$), and the elements corresponding to these collision slots are directly set to '0s'. The above process is exemplified in Fig. 1 (a).

The reader generates another random number R_2 to reconcile the 2-collision slots and 3-collision slots. *What is the collision reconcilement?* For a certain k-collision slot ($k = 2$ or 3), the reader uses the new random number R_2 to calculate $H(ID, R_2) \mod k$ for each tag confined in this slot, the hashing result is within $[0, k-1]$. If each of the k tags is hashed to a unique number $b \in [0, k-1]$, we say this k-collision slot is successfully reconciled; otherwise, this slot is unreconcilable. This collision reconciling process is exemplified in Fig. 1 (b). For example, tags t_2, t_3 and t_5 are mapped to a 3-collision slot in Fig. 1 (a). By performing $H(ID, R_2) \mod 3$, t_2 is hashed to 1; t_3 is hashed to 0; t_5 is hashed to 2; they are hashed to different $b \in [0, 2]$—this 3-collision slot is successfully reconciled. Another example: tags t_4 and t_6 confined in a 2-collision slot are hashed to the same number $b \in [0, 1]$ by performing $H(ID, R_2) \mod 2$—this 2-collision slot is unreconcilable. The elements corresponding to the *un*reconcilable collision slots are re-set to '0s'.

How to relocate the tags in the reconcilable k-collision slot? Obviously, the reader is able to know which k-collision slots ($k = 2$ or 3) are reconcilable. For an arbitrary reconcilable k-collision slot, let λ_2 denote the number of reconcilable 2-collision slots preceding it; similarly, let λ_3 denote the number of reconcilable 3-collision slots preceding it. Clearly, the k tags confined in this reconcilable k-collision slot are hashed to different $b \in [0, k-1]$. The tag that is hashed to 0 still *stays* in this slot; and the tag that is hashed to $b \in [1, k-1]$ is *relocated* to the $[(\lambda_2 \times 1 + \lambda_3 \times 2) + b]^{th}$ slot in the *appended vector*—the appended vector is a vector with dynamic length appended to the tail of the filter vector. As exemplified in Fig. 1 (b), for the first reconcilable 3-collision slot, t_3 is hashed to 0 when reconciling, so it still stays in this slot; t_2 is hashed to 1, then it is relocated to the 1^{st} slot in the appended vector; t_5 is hashed to 2, then it is relocated to the 2^{nd} slot in the appended vector. Another example is given in the following. For the second reconcilable collision slot (a 2-collision slot), t_8 is hashed to 0 when reconciling, then it still stays in the original slot; t_7 is hashed to 1, and because there is one reconcilable 3-collision slot preceding this slot, then it is relocated to the $(1 \times 2 + 1)^{th}$ slot (i.e., the 3^{rd} slot) in the appended vector.

4.2.2 Slot Filtering Stage

Recall that the elements in the filter vector corresponding to the *un*reconcilable collision slots are re-set to '0s'. As a result, each of the non-zero elements in the filter vector represents either an expected singleton slot or a reconcilable k-collision slot ($k = 2$ or 3). Let $\overline{\chi}$ denote the number of all non-zero elements in the filter vector. In the next stage, *only* the slots corresponding to non-zero elements in the filter vector and in the appended vector are executed. The expected empty slots, the k-collision slots ($k \geq 4$) and the unreconcilable k-collision slots ($k = 2$ or 3) that correspond to '0s' in the filter vector are directly skipped, namely, *filtered out*.

4.2.3 Presence Verifying Stage

The reader broadcasts the parameters R_1, R_2, f, $\overline{\chi}$, and the filter vector to the tags. Each tag, say t_i, first uses the received parameters and its ID to calculate $s_i^{'th} = H(ID_i, R_1) \mod f$. When receiving the filter vector broadcasted by the reader, tag t_i records the number of '1s', '2s', and '3s' preceding the $s_i^{'th}$ element in the filter vector, denoted as χ_{i1}, χ_{i2} and χ_{i3}, respectively. After receiving the $s_i^{'th}$ element, it checks the $s_i^{'th}$ element. Then, it picks a slot to respond based on the following rules:

1) If the $s_i^{'th}$ element is '1', which means that t_i selects an expected singleton slot, it will respond in the $(\chi_{i1} + \chi_{i2} + \chi_{i3} + 1)^{th}$ slot.
2) If the $s_i^{'th}$ element is '2', which means that t_i selects a reconcilable 2-collision slot, then it

calculates $H(ID_i, R_2) \mod 2$. (i) If the hashing result of $H(ID_i, R_2) \mod 2$ is 0, it will respond in the $(\chi_{i1} + \chi_{i2} + \chi_{i3} + 1)^{th}$; (ii) if the hashing result of $H(ID_i, R_2) \mod 2$ is 1, it will respond in the $(\overline{\chi} + \chi_{i2} + \chi_{i3} \times 2 + 1)^{th}$ slot.

3) If the $s_i^{'th}$ element is '3', which means that t_i selects a reconcilable 3-collision slot, then it calculates $H(ID_i, R_2) \mod 3$. (i) If the hashing result of $H(ID_i, R_2) \mod 3$ is 0, it will respond in the $(\chi_{i1}+\chi_{i2}+\chi_{i3}+1)^{th}$ slot; (ii) if the hashing result of $H(ID_i, R_2) \mod 3$ is 1, it will respond in the $(\overline{\chi} + \chi_{i2} + \chi_{i3} \times 2 + 1)^{th}$ slot; (iii) if the hashing result of $H(ID_i, R_2) \mod 3$ is 2, it will respond in the $(\overline{\chi} + \chi_{i2} + \chi_{i3} \times 2 + 2)^{th}$ slot.
4) If the $s_i^{'th}$ element is '0', which means that (i) t_i selects an *un*reconcilable k-collision slot ($k = 2$ or 3); or (ii) t_i selects a k-collision slot ($k \geq 4$), which is hard to reconcile and is directly abandoned in this paper. As a result, tag t_i will not respond in this round.

Because all the used parameters and the hashing function $H(\cdot)$ are shared by the reader and all the tags, the reader is able to predict all the decisions of tags. Clearly, all the executed slots are expected to be the singleton slots. If the reader does not receive a response in a slot as expected, the tag corresponding to this slot must be missing.

In this round, the presence of the tags that pick the expected singleton slots or reconcilable collision slots can be verified, and they will not participate the following rounds. As illustrated in Fig. 1 (b), all tags excluding t_4 and t_6 can be verified. However, tags t_4 and t_6 can not be verified because they pick an *un*reconcilable collision slots. As aforementioned, SFMTI repeats for multiple rounds until the presence of all tags are verified.

Note that the filter vector broadcasted in the third stage may be too long to be transmitted in one tag slot (i.e., t_{tag}). To address this problem, the long filter vector is divided into multiple segments of 96-bits to be sequentially transmitted in multiple tag slots [8].

4.3 Determining the Optimal Filter Vector Length

In what follows, we present how to choose the optimal filter vector length f in order to achieve the best time-efficiency. In an arbitrary round of executing the SFMTI protocol, let N^* denote the number of tags that are not verified and will participate in this round.

For a certain element in the filter vector, the corresponding slot can be used only if (1) it is an expected singleton slot; or (2) it is a reconcilable 2-collision slot; or (3) it is a reconcilable 3-collision slot. Let P_1, P_2, P_3 denote the probability that this slot is an expected singleton slot, is a reconcilable 2-collision slot, is a reconcilable 3-collision slot, respectively. They are given as follows:

$$\begin{aligned} P_1 &= \binom{N^*}{1} \times \frac{1}{f} \times (1 - \frac{1}{f})^{N^*-1} \\ &\approx \frac{N^*}{f} \times e^{-\frac{N^*-1}{f}} \\ &\approx \rho e^{-\rho}, \end{aligned} \tag{1}$$

where $\frac{N^*}{f}$ is denoted as ρ for the clarity. Note that, because the length f of the filter vector is usually very large, $(1 - \frac{1}{f})^{N^*-1}$ is approximated to $e^{-\frac{N^*-1}{f}}$ in Eq. (1).

If this slot is a k-collision slot ($k = 2$ or 3), the probability that it can be successfully reconciled is $\frac{k!}{k^k}$. Then, we have:

$$\begin{aligned} P_2 &= [\binom{N^*}{2} \times (\frac{1}{f})^2 \times (1 - \frac{1}{f})^{N^*-2}] \times \frac{2!}{2^2} \\ &\approx \frac{N^*(N^*-1)}{4f^2} \times e^{-\frac{N^*-2}{f}} \\ &\approx \frac{1}{4}\rho^2 e^{-\rho}, \end{aligned} \tag{2}$$

$$\begin{aligned} P_3 &= [\binom{N^*}{3} \times (\frac{1}{f})^3 \times (1 - \frac{1}{f})^{N^*-3}] \times \frac{3!}{3^3} \\ &\approx \frac{N^*(N^*-1)(N^*-2)}{27f^3} \times e^{-\frac{N^*-3}{f}} \\ &\approx \frac{1}{27}\rho^3 e^{-\rho}, \end{aligned} \tag{3}$$

There are f elements (i.e., slots) in the filter vector, each having the probability P_1 to be an expected singleton slot; having the probability P_2 to be a reconcilable 2-collision slot; having the probability P_3 to be a reconcilable 3-collision slot. Let $\aleph_1$, $\aleph_2$, $\aleph_3$ denote the expected number of singleton slots, reconcilable 2-collision slot, reconcilable 3-collision slot, respectively. We have:

$$\begin{aligned} \aleph_1 &= f \times P_1 \approx f \times \rho e^{-\rho} \\ \aleph_2 &= f \times P_2 \approx f \times \frac{1}{4}\rho^2 e^{-\rho} \\ \aleph_3 &= f \times P_3 \approx f \times \frac{1}{27}\rho^3 e^{-\rho} \end{aligned} \tag{4}$$

Note that, each *reconcilable* k-collision slot ($k = 2$ or 3) is reconciled into k singleton slots finally. As a result, $\aleph_1 + 2\aleph_2 + 3\aleph_3$ expected singleton slots are achieved. We use T to denote the execution time of this round, which includes the time for transmitting the filter vector ($2f$ bits) and the $\aleph_1 + 2\aleph_2 + 3\aleph_3$ short-response slots. Hence, T is given as follows:

$$T = \lceil \frac{2f}{96} \rceil \times t_{tag} + (\aleph_1 + 2\aleph_2 + 3\aleph_3) \times t_{short} \tag{5}$$

Since each of the $\aleph_1 + 2\aleph_2 + 3\aleph_3$ singleton slots can be used to verify the presence of a tag, the number of tags that can be verified in this round, denoted as $\Re$, can be given as follows:

$$\Re = \aleph_1 + 2\aleph_2 + 3\aleph_3 \tag{6}$$

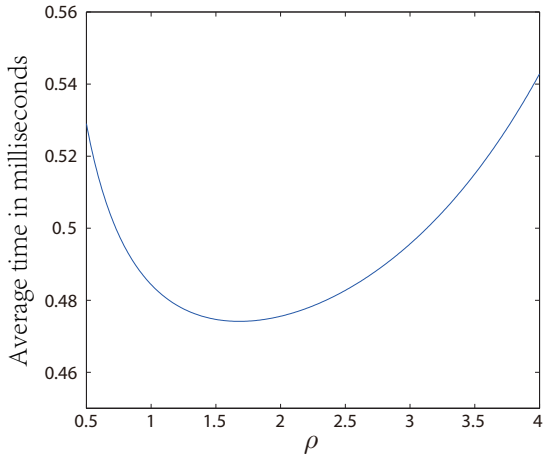

Fig. 2. The average execution time for verifying a tag with respect to ρ.

According to Eqs. (5) and (6), the average time for verifying the presence of one tag is given as follows:

$$\begin{aligned}\frac{T}{\Re} &= \frac{\lceil \frac{2f}{96} \rceil \times t_{tag} + (\aleph_1 + 2\aleph_2 + 3\aleph_3) \times t_{short}}{\aleph_1 + 2\aleph_2 + 3\aleph_3} \\ &= \frac{\lceil \frac{2f}{96} \rceil \times t_{tag} + (f\rho e^{-\rho} + \frac{1}{2} f\rho^2 e^{-\rho} + \frac{1}{9} f\rho^3 e^{-\rho}) \times t_{short}}{f\rho e^{-\rho} + \frac{1}{2} f\rho^2 e^{-\rho} + \frac{1}{9} f\rho^3 e^{-\rho}} \\ &\approx \frac{\frac{t_{tag}}{48} + (\rho e^{-\rho} + \frac{1}{2}\rho^2 e^{-\rho} + \frac{1}{9}\rho^3 e^{-\rho}) \times t_{short}}{\rho e^{-\rho} + \frac{1}{2}\rho^2 e^{-\rho} + \frac{1}{9}\rho^3 e^{-\rho}},\end{aligned} \tag{7}$$

where $t_{tag} = 2.4ms$ and $t_{short} = 0.4ms$. As illustrated in Fig. 2, the average time spent per tag, $\frac{T}{\Re}$, is a function of ρ. To achieve the best time-efficiency, we need to minimize $\frac{T}{\Re}$ in Eq. (7). Hence, we get its derivative as follows:

$$(\frac{T}{\Re})' = \frac{-\frac{t_{tag}}{48} \times e^{-\rho} \times (1 - \frac{1}{6}\rho^2 - \frac{1}{9}\rho^3)}{(\rho e^{-\rho} + \frac{1}{2}\rho^2 e^{-\rho} + \frac{1}{9}\rho^3 e^{-\rho})^2} \tag{8}$$

It is easy to get a ρ_0 to satisfy the $(\frac{T}{\Re})' = 0$ in Eq. (8). Clearly, $(\frac{T}{\Re})'$ is larger than 0 when $\rho > \rho_0$; and $(\frac{T}{\Re})'$ is smaller than 0 when $\rho < \rho_0$. That is, $\frac{T}{\Re}$ is minimized when $\rho = \rho_0$, where $\rho_0 \approx 1.68$ and the average execution time per tag, $\frac{T}{\Re}$, is approximately equal to $0.474ms$. Clearly, $\frac{T}{\Re}$ only depends on ρ (i.e., the ratio $\frac{N^*}{f}$) instead of N^*. Hence, if we set $\rho = 1.68$ (i.e., $f = \frac{N^*}{1.68}$) for all rounds, the average execution time spent to verify a tag becomes a constant $0.474ms$ across all the rounds of SFMTI. As a result, the execution time of the proposed SFMTI is $0.474Nms$, where N is the number of all tags that need to be monitored.

4.4 Estimating The Expected Execution Round Count C

In the previous subsection, we have investigated the optimal length f of the filter vector in each round. The filter vector length f is optimized to $\frac{N^*}{1.68}$ (i.e., $\rho = 1.68$) in an arbitrary round, where N^* is the number of tags that participate in this round. Under this condition, we estimate how many rounds the SFMTI protocol needs to execute to verify the presence of all RFID tags.

Let θ denote the ratio of the tags that can be verified in an arbitrary round. According to Eqs. (4) and (6), θ is given as follows:

$$\begin{aligned}\theta &= \frac{\Re}{N^*} = \frac{\aleph_1 + 2\aleph_2 + 3\aleph_3}{N^*} \\ &= e^{-\rho}(1 + \frac{1}{2}\rho + \frac{1}{9}\rho^2),\end{aligned} \tag{9}$$

where $\rho = 1.68$. That is, the ratio θ of the tags that can be verified in this round is a constant. The other $1-\theta$ of the N^* tags cannot be verified and will participate in the next round. It is easy to know that the number of tags that participate in the C^{th} round is $N \times (1-\theta)^{C-1}$ (expectation value). Using the knowledge that this value is a positive integer, we get the following inequality:

$$N \times (1-\theta)^{C-1} \geq 1 \tag{10}$$

By solving Eq. (10), we get $C \leq log_{(1-\theta)}(\frac{1}{N}) + 1$, where θ is a constant given in Eq. (9). That is, the execution round count of SFMTI is bounded by $log_{(1-\theta)}(\frac{1}{N}) + 1$.

4.5 Extension: Considering Multiple Readers

In this subsection, we propose an effective method to extend SFMTI for the use in the multi-reader scenarios. Considering η readers are deployed in a large-scale area, and they are denoted as $r_i | i \in [1, \eta]$. We assume all the tags are uniformly placed and covered by at least one reader, the tag set covered by reader r_i is denoted as γ_i. obviously $\gamma_1 \bigcup \gamma_2 \bigcup, \cdots, \bigcup \gamma_\eta = S_{all} - S_{miss}$; and $S_{miss} = \emptyset$ when no tag is missing. Note that, if the adjacent readers simultaneously interrogate the overlapped tags, reader-collision will occur. In [24], Yang *et al.* investigated a protocol stack named Season to solve this reader-collision problem. Briefly, an identification procedure is splitted into two phases. In Phase-I, the system identifies all non-contentious tags. This phase is called Shelving Interference. In Phase-II, neighboring readers jointly and collaboratively identify contentious tags. This phase is called Joint Identification. Due to the limitation of space, we do not discuss this issue in detail.

In the multi-reader scenarios, how to let a reader know which tags are confined in its interrogating range is a challenging issue. Fortunately, this can be achieved using the well-known Bloom filter [25], [26] technique. A Bloom Filter is a data structure that probabilistically represents a set of n elements $Y = \{y_1, y_2, \cdots, y_n\}$, which can be used to test set membership. Specially, each of the n elements in this set is compressed into a Bloom filter vector with w bits using k hashing functions $h_1, h_2, \cdots, h_k$. A bit in the vector is set to '1' if at least one element is hashed to that index in the vector. If we want to check whether a given element y belongs to the set Y, we compute

$h_1(y), h_2(y), \cdots, h_k(y)$ and assert $y \in S$ if and only if all these k bits are '1s' in the vector; otherwise, $y \notin Y$. Yue *et al.* leveraged the synchronized physical layer transmissions to distributively construct the desired Bloom filter, which represents the tag set in the vicinity of a reader [27]. Using the constructed Bloom filter to test the membership of all the IDs in the database, an arbitrary reader, say r_i, is expected to learn that $|\gamma_i| + p \times (N - |\gamma_i|)$ tags are within interrogating range, where p is the false positive probability. We denote this tag set as $\hat{\gamma}_i$, and it is available to the reader r_i. It then executes the SFMTI protocol using $\hat{\gamma}_i$ as the input, and returns the missing tag set S_i. In other words, $\hat{\gamma}_i - S_i$ is the actual tag set within the range of reader r_i. It is easy to known that $\bigcup_{i=1}^{\eta}(\hat{\gamma}_i - S_i)$ is the set of all present tags in the system. Thus, $S_{all} - \bigcup_{i=1}^{\eta}(\hat{\gamma}_i - S_i)$ is the final set of missing tags.

The execution time of the this parallelizing method consists of two parts: (1) the time for constructing the Bloom filter distributively; (2) the time for executing the SFMTI protocol in a parallel mode. For an arbitrary reader r_i, its execution time for constructing the Bloom filter is $-\frac{|\gamma_i| \times lnp}{(ln2)^2} \times t_{bit}$ ms, where t_{bit} is the time (in milliseconds) for transmitting a bit, according to [27]. And the execution time of SFMTI is about $0.474 \times (|\gamma_i| + p \times (N - |\gamma_i|))$ ms. The total execution time for r_i is $T_{r_i} = -\frac{|\gamma_i| \times lnp}{(ln2)^2} \times t_{bit} + 0.474(|\gamma_i| + p \times (N - |\gamma_i|))$ ms. We need to optimize the false positive p to minimize this execution time. By setting its derivative $(T_{r_i})' = 0$, we get $p = \frac{|\gamma_i| \times t_{bit}}{0.474 \times (ln2)^2 (N - |\gamma_i|)}$. Moreover, when $p > \frac{|\gamma_i| \times t_{bit}}{0.474 \times (ln2)^2 (N - |\gamma_i|)}$, $(T_{r_i})' > 0$; when $p < \frac{|\gamma_i| \times t_{bit}}{0.474 \times (ln2)^2 (N - |\gamma_i|)}$, $(T_{r_i})' < 0$. Hence, the execution time T_{r_i} is minimized when p is configured to $\frac{|\gamma_i| \times t_{bit}}{0.474 \times (ln2)^2 (N - |\gamma_i|)}$. And the global execution time is the largest execution time of all readers. Note that, there is a problem that the actual set γ_i of tags that are covered by reader r_i is not available in prior. Fortunately, we do not need to know exactly which tags are within r_i; we only need its number $|\gamma_i|$, which could be roughly approximated by $Coverage\ Acreage\ of\ r_i \times Tag\ Density$.

5 Performance Evaluation

This section will evaluate the performance of the proposed SFMTI protocol. For fair comparison with the most compromising protocol, we adopt the same settings of simulation parameters as IIP [8]. We consider a *error-free* communication channel and simulate a single reader in the experiments unless otherwise specified. In the *Presence Verifying Stage* stage, transmission of each segment (96 bits) of the filter vector takes a tag slot (i.e., $t_{tag} = 2.4\ ms$). And it takes each tag a short-response slot (i.e., $t_{short} = 0.4\ ms$) to relay a 1-bit response to the reader for declaring itself. We run each simulation 1000 times and gather the average experimental results.

5.1 Validate the Optimal Length f of the Filter Vector

In this set of simulations, we validate the optimal length f of the filter vector, which is the most important parameter setting in the proposed SFMTI protocol. In Section 4.3, we have proven that the filter length f should be set to $\frac{N^*}{1.68}$ in each round (i.e., $\rho = \frac{N^*}{f} = 1.68$). In our simulations, we vary the parameter ρ from 0.5 to 4, and record the corresponding average execution time per tag. Fig. 3 shows that the simulation results well match the analytical results.

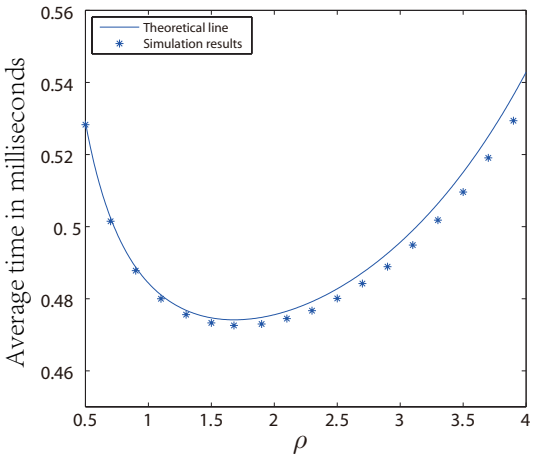

Fig. 3. Fix the number N of all the RFID tags to 10,000, and vary the parameter $\rho = \frac{N^*}{f}$ from 0.5 to 4, where N^* indicates the number of tags that participate in an arbitrary round, and f indicates the length of filter vector in the corresponding round.

5.2 Validate the Execution Round Count C

In Section 4.4, we have theoretically analyzed the expected execution round count C SFMTI needs to execute so as to verify all RFID tags. Theoretically, the execution round count C is bounded by $log_{(1-\theta)}(\frac{1}{N}) + 1$, where N is the number of tags and θ can be got according to Eq. (9). In this set of simulations, we validate the expected round count of SFMTI, where we fix the parameter ρ to 1.68 (i.e., the optimal value) and vary the number N of tags from $10,000$ to $50,000$. The simulation results in Table 2 demonstrate that the execution round count (average) of the proposed SFMTI protocol is well bounded by $log_{(1-\theta)}(\frac{1}{N}) + 1$.

TABLE 2
Validate the execution round count C, with the number N of tags varying from 10,000 to 50,000.

N	10,000	20,000	30,000	40,000	50,000
Theoretical Round Count	18.9	20.3	21.1	21.7	22.1
Simulation Round Count	16.5	17.9	18.6	19.2	19.7

TABLE 3
The execution time with respect to the number N of tags.

N	Execution time (s)				
	BTP	EDFSA	IIP	SFMTI	Lower Bound
5000	40.02	38.68	4.37	2.36	2.00
10000	80.87	77.15	8.68	4.73	4.00
20000	162.04	157.23	17.29	9.47	8.00
30000	242.93	231.94	25.97	14.21	12.00
40000	324.21	311.41	34.58	18.95	16.00
50000	404.46	387.80	43.21	23.69	20.00

5.3 Execution Time

The most important performance criterion for the problem of missing tag identification is the required execution time. We mainly compare the proposed SFMTI protocol with the currently most promising IIP protocol [8]. As the solutions to *tag-collection* problem can also be used to identify the missing tags, we compare the SFMTI with the well-known tag-collection protocols, including the Enhanced Dynamic Framed Slotted ALOHA (EDFSA) [13] and the Binary Tree Protocol (BTP) [14]. The number of tags, N, increases from $5,000$ to $50,000$. Table 3 lists the execution time required by these protocols, respectively. The simulation results reveal that the proposed SFMTI protocol performs much better than BTP and EDFSA. For example, when $N = 50,000$, the execution time of the BTP and EDFSA is 404.46 and 387.80 seconds, respectively. However, the execution time of the SFMTI protocol is just 23.69 seconds, representing 94.1% and 93.9% reduction when compared with the BTP and EDFSA. In comparison with IIP, the SFMTI protocol reduces the execution time by about 45%. For example, When N is $50,000$, the execution time of the SFMTI protocol is 45.2% less than the time required by the IIP protocol. Moreover, the execution time of the proposed SFMTI protocol is very close to the lower bound of the minimum execution time. For example, when $N = 50,000$, the execution time of the SFMTI protocol is 23.69 seconds which is just 1.18 times the lower bound.

5.4 The Impact of Missing Tag Number

In this set of simulations, we investigate the impact of the number of missing tags. Besides the *total execution time* for identifying all the missing tags, the time required for identifying the first missing tag is also an important performance criterion. For example, a thief steals multiple items from a warehouse. Clearly, the faster the protocol identifies the first missing tags, the sooner the alarm is given. We set $N = 10,000$ and vary the number of missing tags, M, from 10 to 100. The proposed SFMTI protocol has to verify the presence of all RFID tags so as to completely identify all the missing tags. As illustrated in Fig. 4, with M (the number of missing tags) increasing, the total execution time for identifying all the missing tags keeps unchanged. Whereas, the time for detecting the first missing RFID tag decreases with M increasing, because of an intuitive reason—the more RFID tags are missing, the more easier to detect one of them.

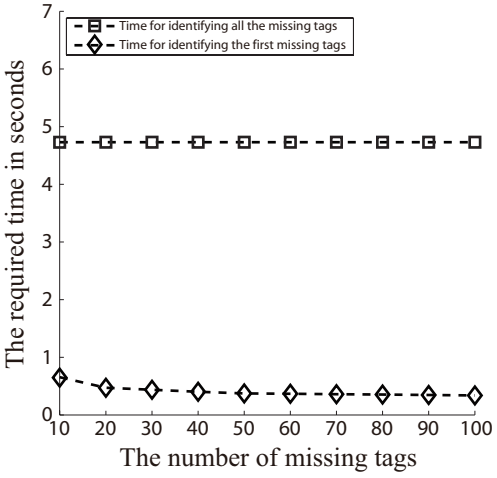

Fig. 4. Fix the number N of the total tags to 10,000, and vary the number M of the missing tags from 10 to 100.

5.5 Evaluate the Performance of SFMTI in the Multi-reader Scenarios

To fully evaluate the performance of the proposed SFMTI protocol, we conducted simulation experiments contain multiple readers in this subsection. As aforementioned, Protocol 3 in [21] performs better than IIP in the multi-reader scenarios. Hence, in this set of simulations, we mainly compare the proposed SFMTI protocol with the Protocol 3 in [21]. Following the simulation settings in [21], we simulated a region consisting of square zones, where we use L to denote the number of the readers. The tags are uniformly distributed and D is used to denote the tag density in each zone, for example, $D = 1000\ tags/zone$ means that there are 1000 tags in each zone on average. The simulation results in Fig. 5 demonstrate that the proposed SFMTI protocol still performs much better than both IIP and Protocol 3, where the number L of the readers is fixed to 50 and the tag density D

varies from 400 tags/zone to 2000 tags/zone. For example, when $D = 2000\ tags/zone$, $L = 50$ (meaning 2000×50=100,000 tags), the execution time of IIP and Protocol 3 is $86.07s$ and $18.69s$, respectively. And the execution time of SFMTI is just $6.71s$, representing 92.2% and 64.1% reduction compared with IIP [8] and Protocol 3 [21].

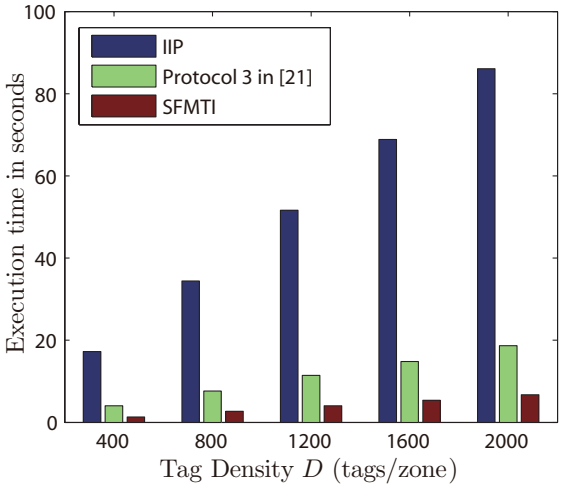

Fig. 5. The number L of the readers is fixed to 50 and the tag density D varies from 400 tags/zone to 2000 tags/zone.

6 Conclusion

This study has addressed the important problem of complete identification of missing tags in large-scale RFID systems. The solutions to this problem are desirable in many valuable monitoring applications in warehouses, hospitals, prisons, etc. This paper has proposed a Slot Filter-based Missing Tag Identification (SFMTI) protocol to identify the missing tags in a time-efficient way. We have also presented how to choose the optimal parameters of the SFMTI protocol. To accommodate multiple readers deployed in some large area scenarios, we have proposed a cost-effective extension method, in which all the readers work in a parallel mode. Furthermore, extensive simulation experiments have been conducted to evaluate the performance of the SFMTI protocol. The simulation results show that the proposed SFMTI protocol outperforms the currently most promising protocol by reducing 45% of the required execution time. Moreover, the execution time of the SFMTI protocol is just within a factor of 1.18 from the lower bound of the minimum execution time.

Acknowledgments

This work was supported by NSFC (grant nos 61173162, 61173160, 61103234, 61272417,61173161,and 61173165), and the National Science Foundation for Distinguished Young Scholars of China (grant no 61225010).

References

[1] C. Wang, H. Wu, and N.-F. Tzeng, "RFID-Based 3-D Positioning Schemes," *Proc. of IEEE INFOCOM*, 2007.
[2] L. M. Ni, Y. Liu, Y. C. Lau, and A. P. Patil, "Landmarc: Indoor Location Sensing Using Active RFID," *Wireless Networks*, vol. 10, no. 6, pp. 701–710, 2004.
[3] A. Nemmaluri, M. D. Corner, and P. Shenoy, "Sherlock: Automatically Locating Objects for Humans," *Proc. of ACM MobiSys*, 2008.
[4] B. Sheng, C. C. Tan, Q. Li, and W. Mao, "Finding Popular Categories for RFID Tags," *Proc. of ACM MobiHoc*, 2008.
[5] C. Qian, H. Ngan, Y. Liu, and L. M. Ni, "Cardinality Estimation for Large-Scale RFID Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 9, pp. 1441–1454, 2011.
[6] C. Lee and C. Chung, "Efficient Storage Scheme and Query Processing for Supply Chain Management using RFID," *Proc. of ACM SIGMOD*, 2008.
[7] M. Kodialam, T. Nandagopal, and W. C. Lau, "Anonymous Tracking using RFID tags," *Proc. of IEEE INFOCOM*, 2007.
[8] T. Li, S. Chen, and Y. Ling, "Identifying the Missing Tags in a Large RFID System," *Proc. of ACM MobiHoc*, 2010.
[9] Y. Zheng and M. Li, "Fast Tag Searching Protocol for Large-Scale RFID Systems," *Proc. of IEEE ICNP*, 2011.
[10] M. Shahzad and A. X. Liu, "Every Bit Counts-Fast and Scalable RFID Estimation," *Proc. of ACM MobiCom*, 2012.
[11] F. C. Schoute, "Dynamic Frame Length ALOHA," *IEEE Transactions on Communications*, vol. 31, no. 4, pp. 565 – 568, 1983.
[12] L. G. Roberts, "Aloha Packet System with and without Slots and capture," *ACM SIGCOMM Computer Communication Review*, vol. 5, no. 2, pp. 28–42, 1975.
[13] S. Lee, S. Joo, and C. Lee, "An Enhanced Dynamic Framed Slotted ALOHA Algorithm for RFID Tag Identification," *Proc. of IEEE MobiQuitous*, 2005.
[14] J. Myung and W. Lee, "Adaptive Splitting Protocols for RFID Tag Collision Arbitration," *Proc. of ACM MobiHoc*, 2006.
[15] N. Bhandari, A. Sahoo, and S. Iyer, "Intelligent Query Tree (IQT) Protocol to Improve RFID Tag Read Efficiency," *Proc. of IEEE ICIT*, 2006.
[16] V. Namboodiri and L. Gao, "Energy-Aware Tag Anticollision Protocols for RFID Systems," *IEEE Transactions on Mobile Computing*, vol. 9, no. 1, pp. 44–59, 2010.
[17] M. Shahzad and A. X. Liu, "Probabilistic Optimal Tree Hopping for RFID Identification," *Proc. of ACM SIGMETRICS*, 2013.
[18] C. C. Tan, B. Sheng, and Q. Li, "Efficient Techniques for Monitoring Missing RFID Tags," *IEEE Transactions on Wireless Communications*, vol. 9, no. 6, pp. 1882–1889, 2010.
[19] W. Luo, S. Chen, T. Li, and S. Chen, "Efficient Missing Tag Detection in RFID Systems," *Proc. of IEEE INFOCOM*, 2011.
[20] W. Luo, S. Chen, T. Li, and Y. Qian, "Probabilistic Missing-tag Detection and Energy-Time Tradeoff in Large-scale RFID systems," *Proc. of ACM MobiHoc*, 2012.
[21] R. Zhang, Y. Liu, Y. Zhang, and J. Sun, "Fast Identification of the Missing Tags in a Large RFID System," *Proc. of IEEE SECON*, 2011.
[22] H. Han, B. Sheng, C. C. Tan, Q. Li, W. Mao, and S. Lu, "Counting RFID Tags Efficiently and Anonymously," *Proc. of IEEE INFOCOM*, 2010.
[23] P. Semiconductors, "I-CODE Smart Label RFID Tags," *http://www.nxp.com/acrobat_download/other/identification/SL092030.pdf*, Jan 2004.
[24] L. Yang, J. Han, C. Wang, T. Gu, and Y. Liu, "Season: Shelving Interference and Joint Identification in Large-scale RFID Systems," *Proc. of IEEE INFOCOM*, 2011.
[25] B. Bloom, "Space/time tradeoffs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426.
[26] H. Fang, K. Murali, and L. TV, "Building high accuracy bloom filters using partitioned hashing," vol. 35, no. 1, pp. 277–288, 2007.
[27] H. Yue, C. Zhang, M. Pan, Y. Fang, and S. Chen, "A Time-efficient Information Collection Protocol for Large-scale RFID Systems," *Proc. of IEEE INFOCOM*, 2012.