Complete and Fast Unknown Tag Identification in Large RFID Systems

Xuan Liu†, Shigeng Zhang‡, Kai Bu†, Bin Xiao†
The Hong Kong Polytechnic University†
Central South University‡

***Abstract*—The RFID technology greatly improves efficiency of many applications including inventory control, object tracking, and supply chain management. In such applications, it is common that new objects are added into the system or existing objects are misplaced in wrong regions. When this happens, fast and complete identification of such tags is very important. We name this problem *unknown tag identification*, as these tags appear to be unknown by the reader(s) currently covering them. In this paper, we propose a series of protocols to identify unknown tags *completely* and *fast*. In these protocols, we develop several novel techniques to efficiently resolve collisions caused by known tags when identifying unknown tags, which greatly improve the time efficiency. To our knowledge, this is the first work that completely identify all the unknown tags with deterministic approaches. Simulation results show the superior performance of the proposed protocols: Compared with a baseline method which collects IDs of all the tags in the system, our best protocol reduces the execution time by 63% in average and by 85% at most.**

***Index Terms*—RFID; complete unknown tag identification**

I. Introduction

The Radio Frequency IDentification (RFID) technology greatly improves efficiency of many applications such as warehouse management, object tracking, and inventory control. In these applications, active or passive RFID tags are attached to objects (e.g., persons and products). Every tag has an unique ID and stores some description data about the attached object. Many applications collect the static information of tags such as their IDs and total number periodically [1]–[9], and store them in a back end database for subsequent analysis or retrieval.

In many applications, dynamic information caused by the changes of tags in the RFID system is more important than static information. In practice, the changes in the set of tags may be caused by entering of new tags, misplacement of tags in wrong regions, or loss of tags. Consider a warehouse in which diverse products are frequently loaded and unloaded. New objects are added into the warehouse every day or even every hour. Furthermore, the stevedores may load products to wrong ships or trucks, which may cause high economic loss to retrieve them back. In these cases, we need to quickly identify the new (misplaced) tags to update the inventory (detect misplaced tags) timely. We name this problem *unknown tag identification*, as the tags to be identified are unknown to the reader(s) currently covering them.

Existing works on unknown tag identification cannot guarantee that all the unknown tags are collected, e.g., misplaced-tag pinpointing [10] and continuous scanning [11]. Misplaced-tag pinpointing focuses on finding misplaced tags that are moved to wrong regions and determining their approximate positions. In [10], [12], Bu et al. propose a reader vector based approach to pinpointing misplaced tags. However, their method needs category information of tags to function correctly, which is not provided by many RFID systems. The continuous scanning scheme [11] detects both missing tags and unknown tags. It uses a probabilistic method to detect unknown tags, and cannot guarantee that all the unknown tags are identified. It cannot be used in applications which strictly require identification of all the unknown tags, e.g., the medicine management in hospitals. This scheme also detects missing tags. Missing tag detection has been studied a lot and we do not cope with this problem in our paper.

There are already some works on missing tag detection and identification in a RFID system [13]–[15]. Tan et al. propose a probabilistic method to check whether there are tags missing from a RFID system, which can correctly detect the tag missing event with high probability when the number of missing tags exceeds a specified threshold. Noting that the reader knows IDs of all the tags in the system, Li et al. [13] propose a series of protocols that further determine which tags are missing. Zhang et al. [15] propose a fast missing tag detection protocol which exploits the parallel execution of multiple readers to achieve high time efficiency.

In this paper we focus on *complete unknown tag identification* which is very important to many practical applications but not thoroughly investigated. Applications that require complete unknown tag identification exist in schools, hospitals, shopping malls, or libraries. For example, in a large library, people sometimes put books onto wrong bookshelves, which makes it difficult for users to find these books. With unknown tag identification, the librarians could easily find all the misplaced books and put them back to the right places. Another typical scenario is to update the inventory in large warehouses or retailers. For example, in a large retailer like Warlmart, new products are added to the shelves periodically. Manually updating information of these products (e.g., which product is on which shelf) will be laborious and time consuming. With fast and complete unknown tag identification, we can do such inventory management work more efficiently. Furthermore, when being used to manage precious objects like jewelery, the RFID system definitely should provide the function of complete unknown tag identification: Even the failure in the identification of only one unknown tag will result in high

978-1-4673-2433-5/12/$31.00 ©2012 IEEE

economic loss.

An intuitive approach to complete unknown tag identification is to collect the IDs of all the tags in the system. However, this is far from efficient. ID collection is time consuming in large RFID systems, and it is a waste in time to recollect IDs of known tags. To avoid re-identification of known tags, we should prohibit the involvement of known tags when identifying unknown tags. However, the reader has no simple way to suppress transmissions of only known tags. In existing identification protocols, the reader can only deactivate one tag with one command: After receiving the tag's ID successfully, the reader sends it an *ACK* to deactivate it. This method is far from efficient. Thus the key issue needs to be solved is how to efficiently deactivate known tags without transmitting their IDs, especially when the number of known tags is large.

We believe this paper gives the first efficient solution to complete identification of unknown tags. First, we propose an efficient known tag deactivation protocol without transmitting tags' IDs. The reader recognizes known tags and unknown tags by comparing the expected replies from known tags with the actual observed replies. It then deactivates recognized known tags to prohibit their involvement in the identification of unknown tags. Second, we design two novel methods which greatly improve the deactivation efficiency by carefully arranging the interactions between the reader and tags. We develop a collision-empty slot paring technique, together with multiple hashing based slot reselection, to help tags select best slots to communicate with the reader. These techniques greatly improve the deactivation efficiency. We also design an zero-cost estimation algorithm to estimate the number of unknown tags and set optimal frame size in our protocols. The superior performance of our protocols is validated via simulations: Compared with a baseline method which collects IDs of all the tags in the system, our best protocol reduces the execution time by 63% in average and by 85% at most.

The rest of this paper is organized as follows. In Section II, we give the problem statement and system model and review related prior work. In Section III, we describe our basic unknown tag identification protocol (BUIP) and analyze its execution time. In Section IV, we give two extensions of BUIP which greatly improve time efficiency. We conduct extensive simulations to evaluate the performance of the proposed protocols in Section VI. At last, Section VII concludes the paper.

II. Preliminaries

A. Problem Statement

We consider an RFID system consisting of one reader and two types of tags: known tags and unknown tags. The number of known tags and of unknown tags are n and m, respectively. We assume that all the IDs of known tags are stored in a database which can be accessed by the reader. This assumption is reasonable and has been adopted by many previous excellent works [11], [13], [16], [17]. Furthermore, we assume that all the known tags exist in the RFID system during the execution of our protocols. In case some tags are accidentally missing (e.g., stolen by theft), the system can run existing efficient missing tag identification protocols (e.g., [13]) to identify them quickly and remove their IDs from the database accordingly.

The problem we want to solve in this paper is *to fastly collect the IDs of all the m unknown tags*.

B. Modified Frame Slotted ALOHA Protocol

We use *frame slotted ALOHA* [5], [18] as the anti-collision protocol. In this protocol, the communications between the reader and tags are divided into several *frames* each consisting of some *slots*. At the beginning of every frame, the reader broadcasts two parameters $< f, r >$, where f indicates the number of slots in the coming frame and r is a random seed. After receiving the parameters, a tag calculates a value $sc = H(ID||r) \bmod f$ and selects the scth slot to transmit, where H is a hash function. When successfully receiving a tag's transmission, the reader replies an *ACK* to make the tag silent. Otherwise, it replies a *NACK* to keep replying tags active such that they can participate in the next frame.

We make two slight modifications to the standard frame slotted ALOHA protocol to improve time efficiency. First, our protocols use three types of ACK: ACK_s, ACK_n, and ACK_l. In our protocols every tag can be in one of three states: *active*, *silent*, or *labeled*. All tags are initially active. If a tag receives ACK_s in its transmission slot, it enters the silent state and would not response any query until the next execution of the protocol; we name this operation as "deactivating a tag". If a tag receives ACK_l, it enters the labeled state and responses only the ID-collection command issued by the reader; we call this operation "labeling a tag". If a tag receives ACK_n, it keeps active. We implement the three ACKs with the two existing ACKs in the standard frame slotted ALOHA protocol: ACK_n is represented by *NACK*, ACK_s is represented by adding a '0' bit at the end of *ACK*, and ACK_s is represented by adding a '1' bit at the end of *ACK*. That is, $ACK_s = ACK +' 0'$ and $ACK_l = ACK +' 1'$.

Second, we use the *indicator vector* technique to help tags select most suitable transmission slots, which can greatly reduce collisions. Indicator vector has already been adopted by many excellent previous works [11], [13], [16], [17] to improve efficiency. The core idea behind this technique is that, if a tag knows that there are other tags choosing the same slot as it does, it should select another slot to transmit or not transmit at all to avoid potential collision. To tell tags whether they should reselect replying slots, the reader broadcasts a f-bit indicator vector. Each bit in the vector corresponds to one slot in the frame. A tag choosing the ith slot checks the ith bit in the indicator vector to determine whether it should reply or not. The implementation details of indicator vector are elaborated in the next section.

C. Indicator Vector

Two key issues in the implementation of indicator vector are how to construct the indicator vector and how to send it to tags. For the first issue, the reader constructs the vector based on its knowledge of tags in the system. Recall that we assume that the reader knows the IDs of all the known tags. Given r and f, it can determine the slot that a known tag hashes to. If there are

more than two tags simultaneously selecting the ith slot, the reader assigns the ith bit in the vector to '1', otherwise assigns it to '0'. In our protocols, besides indicating collision slots, we also use indicator vector to disseminate some information for slot reselection. See Section III and Section IV for the details.

The second key issue is how to transmit the indicator vector to tags. To fit in the frame slotted ALOHA protocol, the vector is usually divided into segments of 96 bits long (the length of a tag's ID). With this method, a tag only needs to check the corresponding bit in a segment and does not need to receive and store the whole vector. For example, if a tag wants to check the value of the 100th bit, it can receive the second segment and check the fourth bit in it.

The indicator vector can be implemented on current RFID tags: The only requirement is that tags can perform some simple computational tasks such as counting and hashing, and current Class 1 Gen 2 tags already provide such abilities [2]. The broadcast of the indicator vector introduces some cost. However, many previous excellent works show that indicator vector can resolve collisions efficiently, and the performance gain by using it overwhelms the overhead it introduces.

D. Prior Work and Limitation

The most related work is *continuous scanning* [11] which detects both missing tags and unknown tags with probabilistic methods. This scheme aims at continuous and efficient management of tags in an RFID system. It uses two separate protocols to detect missing tags and identify unknown tags.

Continuous scanning uses a probabilistic method to identify unknown tags. It first predicts which slot should be empty according to the IDs of known tags in the system, assuming no unknown tag exists. In the predicted non-empty slots, the reader sends *ACK* to deactivate known tags. In the predicted empty slots, the reader sends *NACK* to keep unknown tags active for following identification. It then collects IDs of all the active unknown tags. Unavoidably, some unknown tags may be wrongly deactivated in the predicted non-empty slots and thus cannot be identified. The protocol runs multiple rounds to guarantee that a required fraction of unknown tags are identified with high probability. However, multiple executions significantly increase the execution time, meanwhile it still cannot ensure *complete* identification of all the unknown tags.

III. BUIP: The Basic Unknown tag Identification Protocol

The basic unknown tag identification protocol (BUIP) recognizes and deactivates known tags to prevent them from interfering the identification of unknown tags. Meanwhile, when the reader recognizes unknown tags, it labels them, which could further decrease the active tags in the system and consequently reduce more potential collisions in the following frames. The challenge is how to recognize known tags and unknown tags without ID transmissions. In this section, we first describe how to quickly deactivate known tags and label unknown tags in Section III-A, then detail BUIP in Section III-B, finally analyze its execution time in Section III-C.

During the tag separation process, a tag replies to the reader with a short response (10-bit long). This response is long enough for the reader to detect collisions. The transmission time of such a response, denoted by t_l, is far less than the transmission time of a tag ID (96-bit long), denoted by t_{ID}.

A. Known Tag Deactivation and Unknown Tag Labeling

Being aware of the indexes of slots that known tags will map to, the reader knows the *expected* status of every slot if there are no unknown tags: expected empty (no known tag replies), expected singleton (one known tag replies), or expected collision (more than one known tags reply simultaneously). Unknown tags also reply to the reader, which makes it possible that the actual status of a slot different from its expected status. The difference between the expected slot states and the observed ones can be used to recognize known tags and unknown tags.

By monitoring the expected empty slots and the expected singleton slots, the reader recognizes the known tags and unknown tags. If only one tag transmits in an expected singleton slot, it must be a known tag. If some tags reply to the reader in an expected empty slot, they must be unknown tags. After recognizing known tags or unknown tags, the reader deactivates or labels them accordingly. In all other cases, although some tags may reply to the reader, the reader cannot distinguish whether they are known tags or unknown tags, thus they should keep active to participate in the next round.

Not all the known tags can be deactivated in a single frame. There are two possible cases in which known tags cannot be deactivated. First, some unknown tags may reply in expected singleton slots, which causes collision with known tags. In this case, the reader cannot rudely deactivate replying tags; otherwise some unknown tags are also deactivated without being identified. Second, tags replying in expected collision slots also cannot be rudely deactivated for the same reason.

Expected collision slots are not used in either known tag deactivation or unknown tag labeling, so we forbid tag transmissions in these slots. The reader broadcasts an indicator vector to achieve this goal. See Section III-B for details.

B. BUIP Detail

BUIP consists of two phases: known tag deactivation and unknown tag collection. In the first phase, the reader deactivates all the known tags and label all the unknown tags. In the second phase, the reader completely identifies unknown tags by collecting IDs of all the labeled tags. We mainly focus on the first phase.

The known tag deactivation phase consists of multiple rounds. In each round, the reader deactivates some known tags and labels some unknown tags until all the known tags are deactivated. At the beginning of each round, the reader broadcasts two parameters, f and r. Upon receiving f and r, a tag calculates its replying slot index as $sc = H(ID||r) \bmod f$. In order to prevent the reply from the tags that map to expected collision slots, the reader broadcasts an indicator vector $\mathbf{v}_c$ consisting of f bits. Each bit corresponds to one slot in the frame: '1' for expected collision slots, and '0' otherwise. A

tag that is hashed to the kth slot checks the kth bit in v_c. It transmits in the kth slot only when $\mathbf{v}_c[k] = 0$. Otherwise, it keeps silent and does not involve in the current round.

According to the replies in each slot, the reader sends different kinds of ACK. if the reader recognizes a known tag in an expected singleton slot, it sends an ACK_s to deactivate that tag. If the reader recognizes unknown tags in an expected empty slot, it send an ACK_l to label them. In all other cases, the reader sends an ACK_n. Tags receiving ACK_n keep active to participate in the next round.

The deactivation phase ends when all the known tags are deactivated. If the total number of deactivated known tags is less than the total number of known tags, the reader starts a new round. When all known tags are deactivated, the reader broadcasts an ACK_l to label all the remaining active unknown tags. Then the reader collects IDs of unknown tags with the DFSA [18] protocol.

C. Analysis of BUIP

We first analyze how to set optimal frame size in BUIP to achieve maximum deactivation efficiency, then calculate an upper bound on the execution time of BUIP. We also discuss some shortages of BUIP at the last of this section.

1) Optimal Frame Size: Assume that there are n_i active known tags and m_i active unknown tags at the beginning of the ith round. For any slot, the probability that one known tag can be deactivated in it equals the probability that exactly one known tag selects it and no unknown tag selects it, which is

$$p_s^i = \binom{n_i}{1} \frac{1}{f_i} (1 - \frac{1}{f_i})^{n_i + m_i - 1} \approx \frac{n_i}{f_i} e^{-\frac{n_i + m_i - 1}{f_i}}, \quad (1)$$

where f_i is the frame size. To maximize p_s^i , let

$$\frac{\partial p_s^i}{\partial f_i} = 0,$$

we can get $f_i = n_i + m_i - 1$. There are f_i slots, so the total number of deactivated tags is $f_i * p_s^i$. Thus the probability that a known tag is deactivated in the ith round is

$$p_d^i = \frac{f_i * p_s^i}{n_i} = e^{-\frac{n_i + m_i - 1}{f_i}}. \quad (2)$$

When $f_i = n_i + m_i - 1$, $p_d^i \approx e^{-1} = 0.368$. We can see that p_d^i is a constant value irrelevant to n_i and m_i when $f_i = n_i + m_i - 1$. For brevity in presentation, in the rest of this section we directly use p_d to denote this probability.

We need to know n_i and m_i to set optimal frame size in every round. Because the reader knows the initial number of known tags and can obtain the exact number of deactivated known tags in every round, we always know the exact value of n_i. We propose an zero-cost algorithm to estimate m_i in every round; the details are given in Section V.

2) Execution Time of BUIP: We first derive the expected number of rounds a known tag participates in before being deactivated. Assume that in all the rounds the frame size is optimally set. Then in every round, a known tag will be deactivated with a constant probability p_d. The probability that a known tag is deactivated *exactly* in the ith round is $p_d(1 - p_d)^{i-1}$. Thus the expected number of rounds a known tag participates in before it is deactivated is

$$E[i] = \sum_{i=1}^{\infty} i p_d (1 - p_d)^{i-1} = \frac{1}{p_d} \approx e. \quad (3)$$

This means that, in average, a known tag needs to participate in e rounds before being deactivated.

We then derive the total number of slots used in the known tag deactivation phase. Consider the ith round in which there are only n_i known tags but the frame size is $f_i = n_i + m_i - 1$. Let $\eta_i = f_i / n_i$. The following observation helps us derive an upper bound of η_i.

Observation 1: Let $\Delta n = n_i - n_{i+1}$ and $\Delta m = m_i - m_{i+1}$. Then the following inequality holds:

$$\frac{E[\Delta m]}{m_i} \geq \frac{E[\Delta n]}{n_i}. \quad (4)$$

Proof: The proof is omitted due to limit in space. ■

Observation 1 shows that, in average, m_i decreases faster than n_i. Because

$$\eta_i = \frac{f_i}{n_i} \approx (1 + \frac{m_i}{n_i}), \quad (5)$$

and m_i decreases (relatively) faster than n_i, we can conclude that $\eta_i \leq \eta_1 = 1 + m/n$ for all i.

With η_i we now derive an upper bound on the total number of slots used in the known tag deactivation phase. We can treat η_i as the average cost of *trying to deactivate a known tag* in the ith round: If a known tag is successfully deactivated, it consumes η_i slots; otherwise, it wastes η_i slots. Assume that a known tag is deactivated exactly in the ith round. The total number of slots that this tag consumes is

$$ns_k = \sum_{i=1}^{k} \eta_i \leq \sum_{i=1}^{k} \eta_1 = \sum_{i=1}^{k} (1 + \frac{m}{n}) = k(1 + \frac{m}{n}).$$

Then the expected number of slots needed to deactivate a known tag is

$$\begin{aligned} E[ns] &= \sum_{k=1}^{\infty} ns_k p_d (1 - p_d)^{k-1} \\ &\leq (1 + \frac{m}{n}) \sum_{k=1}^{\infty} k p_d (1 - p_d)^{k-1} \approx (1 + \frac{m}{n}) e. \end{aligned} \quad (6)$$

The total number of slots needed to deactivate all the n known tags is bounded by $n * E[ns] = (n + m)e$.

We then have the following result:

Theorem 1: The execution time of BUIP in the known tag deactivation phase is bounded by

$$T_{BUIP} = \lceil \frac{(n + m)e}{96} \rceil * t_{ID} + (n + m) * e * t_l, \quad (7)$$

where t_{ID} is the time needed to transmit a tag ID, and t_l is the time needed to transmit a 10-bit short response.

Proof: The execution time of BUIP in the known tag deactivation phase consists of two parts. The first part is the time used to broadcast $\mathbf{v}_c$[1]. As we have analyzed, the total

[1]The time needed to broadcast f and r can be ignored because they are very short.

number of slots used is at most $(n+m)e$. For every slot, there is one bit in $\mathbf{v}_c$ corresponding to it. Thus the total number of bits in $\mathbf{v}_c$ in all the rounds is also at most $(n+m)e$. We assume that the reader divides $\mathbf{v}_c$ into segments of 96 bits long when transmitting it, and transmitting a segment uses t_{ID} time. Thus the total time needed to transmit $\mathbf{v}_c$ in all the round is at most $\lceil \frac{(n+m)e}{96} \rceil * t_{ID}$.

The second part is the time consumed in collecting tags' replies. There are at most $(n+m)*e$ slots, and each slot incurs at most t_l time. Thus the total time of the second part is at most $(n+m)*e*t_l$.

Combining the two parts completes this proof. ■

The deactivation time of known tags can be further reduced if we take use of expected collision slots. In BUIP we prohibit the transmission from tags that originally choose expected collision slots. If these tags reselect other slots, some expected collision slots may become singleton. Following this guideline, we improve time efficiency of BUIP by making tags that originally choose expected collision slots reselect their replying slots.

IV. Enhancements of BUIP

In this section, we propose two extensions of BUIP: BUIP-CE and BUIP-CF.

A. BUIP-CE: BUIP with Collision-Empty Slot Pairing

1) Collision-empty Slot Pairing: We develop a technique called *slot paring* to provide a second chance for tags that originally choose an expected collision slot to reselect their replying slot, which can generate more expected singleton slots and consequently improve deactivation probability. In BUIP with collision-empty slot pairing (BUIP-CE), an expected collision slot is paired with an expected empty slot. A tag that originally chooses this collision slot randomly selects either of the two pairing slots as its final replying slot. Assume that there are k known tags originally choosing the expected collision slot. After reselection, if $k-1$ out of them choose the same slot, then the paring slot becomes an expected singleton slot. What is better, if $k=2$ (which is the most possible case for an expected collision slot because $f_i \geq n_i$), we may get two expected singleton slots.

The challenge is how to let a tag know the index of its pairing slot in the frame. Theoretically, there are always more expected empty slots ($\approx e^{-n_i/f_i}$) than expected collision slots ($\approx 1-\frac{n_i}{f_i}e^{-n_i/f_i}-e^{-n_i/f_i}$). We pair the jth expected collision slot with the jth expected empty slot. A tag that originally chooses the jth expected collision slot may re-map to the jth expect empty slot in the second chance. How to determine the value of j for a tag and how to determine the index of the jth expected empty slot in the frame is detailed below.

2) BUIP-CE Detail: BUIP-CE also consists of two phases: the known tag deactivation phase and the unknown tag collection phase. The second phase is the same as in BUIP, so we only describe the first phase here.

In every round of BUIP-CE, after sending f, r and $\mathbf{v}_c$, the reader broadcasts another seed r' and another indicator vector $\mathbf{v}_e$ to help colliding tags determine their pairing slots. $\mathbf{v}_e$ consists of f bits and indicates the expected empty slots in the frame: If the kth slot is expected to be empty, $\mathbf{v}_e[k]=1$; otherwise, $\mathbf{v}_e[k]=0$.

Upon receiving f and r, a tags first calculates its replying slot index. It then checks the expected status of its chosen slot with $\mathbf{v}_c$, which is the same as in BUIP. If the tag finds that it selects an expected collision slot, it determines its pairing slot with $\mathbf{v}_c$ and $\mathbf{v}_e$, and reselects replying slot with r'.

A tag determines its pairing slot in two steps. First, it determines the collision index of its slot. The collision index of a slot is its rank when considering only expected collision slots. For example, assume that $\mathbf{v}_c$ ="011010...", then the collision index of the fifth slot is 3, meaning that the fifth slot is the third expected collision slot. The tag counts the number of "1" before its slot in $\mathbf{v}_c$ to determine the collision index of its slot. Second, the tag finds the index of its paring slot in the frame. To do this, it finds the index of the jth expected empty slot by searching the $(j+1)$th "1" in $\mathbf{v}_e$. Denote the index of the $(j+1)$th "1" in $\mathbf{v}_e$ as p_{j+1}. The tag uses slot p_{j+1} as its pairing slot.

After choosing its pairing slot, a tag randomly choose either slot as its final replying slot. The reselection result is controlled by r'. The tag calculates $H(ID||r') \bmod 2$. It replies in its original slot if the value equals 0, and replies in the pairing slot otherwise. With r' and known tags' IDs, the reader exactly knows the reselected result of every known tag.

3) BUIP-CE Analysis: We now analyze the deactivation probability of known tags in BUIP-CE. Assume that there are n_i known tags and m_i unknown tags in the ith round. A known tag can be deactivated in two cases: 1) It originally chooses an expected singleton slot and no unknown tag chooses this slot, or 2) it originally chooses an expected collision slot but picks an expected singleton slot after reselection. Denote by p_0, p_1, and p_c the probability that a slot is expected empty, expected singleton, and expected collision before reselection, respectively. Then

$$p_0 = e^{-n_i/f}, p_1 = \frac{n_i}{f}e^{-n_i/f}, p_c = 1-p_0-p_1, \tag{8}$$

where $f = m_i + n_i - 1$.

Denote by x_s the average number of known tags deactivated in an original expected singleton slot, we have

$$x_s = (1-\frac{1}{f})^{m_i} = e^{-m_i/f}. \tag{9}$$

Denote by x_c the average number of known tags deactivated in an original expected collision slot and its pairing slot. Then the expected number of deactivated known tags in a slot is

$$E[s] = p_1 * x_s + p_c * x_c. \tag{10}$$

To calculate $E[s]$, we need to know x_c. For an original expected collision slot, the probability that exactly k_1 known tags and k_2 unknown tags choose this slot is ($k_1 \geq 2$, $k_2 \geq 0$)

$$p_{nc}(k_1,k_2) = \frac{B(k_1,n_i,1/f)}{\sum_{j=2}^{\infty}B(j,n_i,1/f)}\frac{B(k_2,m_i,1/f)}{\sum_{j=0}^{\infty}B(j,m_i,1/f)}, \tag{11}$$

TABLE I
THE VALUE OF p_d^i WITH DIFFERENT m_i ($n_i = 10,000, f = n_i + m_i$)

m_i	250	500	1000	2000	3000	4000	5000
p_d^i	.602	.598	.589	.575	.562	.550	.540

TABLE II
THE VALUE OF p_d^i WITH DIFFERENT k
($n_i = 10,000, m_i = 2,000, f = n_i + m_i$).

k	1	2	3	4	5	6	7
p_d^i	.575	.671	.717	.740	.752	.758	.761

where $B(z, n, 1/f)$ is the probability that a random variable following a binomial distribution $B(n, 1/f)$ takes value of z.

Denote by $es(k_1, k_2)$ the expected number of known tags deactivated in an expected collision slot and its pairing slot when exactly k_1 known tags and k_2 unknown tags originally choose this slot. Then we have

$$x_c = \sum_{k_1=2}^{\infty} \sum_{k_2=0}^{\infty} es(k_1, k_2) * p_{nc}(k_1, k_2). \tag{12}$$

For an expected collision slot, assume that there are exactly k_1 known tags and k_2 unknown tags originally choose it. After reselection, this slot and its pairing slot may become singleton and known tags can be deactivated. The probability that at least one slot becomes singleton is

$$p_s(k_1, k_2) = \begin{cases} \frac{1}{2} & k_1 = 2, k_2 = 0 \\ k_1(\frac{1}{2})^{k_1+k_2-1} & k_1 \geq 3, \text{or } k_1 = 2, k_2 \geq 1. \end{cases}$$

If $k_1 = 2$ and $k_2 = 0$, two known tags may be deactivated. Thus

$$es(2, 0) = p_s(2, 0) * 2 + (1 - p_s(2, 0)) * 0 = 1.$$

Otherwise, only one known tag may be deactivated in the generated singleton slot. Thus

$$es(k_1, k_2) = p_s(k_1, k_2) * 1 + (1 - p_s(k_1, k_2)) * 0 = p_s(k_1, k_2).$$

The total number of deactivated known tags is $n_s = f * E[s]$. So the probability that a known tag is deactivated is

$$p_d^i = n_s / n_i. \tag{13}$$

Different from in BUIP, the deactivation probabilities in different rounds of BUIP-CE are different. Thus it is difficult to theoretically analyze its execution time. We evaluate the execution time of BUIP-CE by simulation in Section VI. Some values of p_d^i for different m_i when $n_i = 10,000$ are listed in Table I. We can see that the deactivation probability of BUIP-CE is much higher than that of BUIP.

B. BUIP-CF: BUIP with Collision-Fresh Slot Paring

BUIP-CE still has two problems. First, an expected empty slot may not be actually empty when some unknown tags hash to this slot originally. These unknown tags would collide with the reselecting known tags, which prevents the reader from deactivating them. What's worse, the reader cannot label these unknown tags either. If the pairing slots are guaranteed to be empty, the reader would deactivate more known tags.

Second, with one reselection seed r', BUIP-CE resolves only a part of expected collision slots. Obviously, a collision slot that cannot be resolved with r' may be resolved with some other seed r''. If the reader sends multiple seeds and tells tags to use the most suitable one, more collisions could be resolved.

1) BUIP-CF Detail: Instead of pairing the jth ($j \geq 1$) expected collision slot with the jth expected empty slot, BUIP-CF pairs it with the $(f + j - 1)$th slot in the frame. See Fig. 1 for the illustration. No unknown tag selects the $(f + j - 1)$th slot originally, which increases the probability that this slot turn to be singleton after slot reselection. The reader collects responses from all the $f + n_c$ slots, where n_c is the number of original expected collision slots.

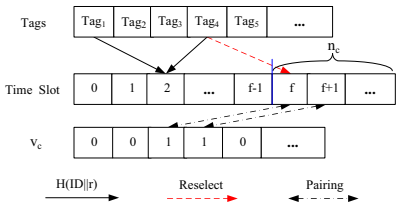


Fig. 1. Pairing slots in BUIP-CF.

To tell tags which seed to use for slot reselection, after sending f, r and $\mathbf{v}_c$, the reader sends k seeds $\{r'_1, r'_2, ..., r'_k\}$ and a second indicator vector $\mathbf{v}_s$. (Note here BUIP-CF does not use $\mathbf{v}_e$.) $\mathbf{v}_s$ consists of n_c elements, one for each expected collision slot. An element has a form of "$\underbrace{0 \ldots 0}_{l-1}1$", which means that tags choosing this collision slot should use the lth seed for slot reselection. Upon receiving these parameters, a tag first selects its replying slot and checks the expected status of its chosen slot with $\mathbf{v}_c$, the same as in BUIP and BUIP-CE. If the tag finds that its chosen slot is expected to be colliding, it checks corresponding element in $\mathbf{v}_s$ to determine which seed to use for slot reselection.

A tag originally chooses the jth expected collision slot uses the jth element in $\mathbf{v}_s$ to determine which seed to use. Upon receiving $\mathbf{v}_s$, the tag determines the jth element by searching a substring of $\mathbf{v}_s$ that starts from the $(j-1)$th '1' (exclusive) and ends with the jth '1' (inclusive). For example, assume that $\mathbf{v}_s$ ="010011...". The second element is "001", thus tags that originally choose the second expected collision should use r'_3 for slot reselection. Similarly, the first element is "01" and thus tags that originally choose the first expected collision slot will use r'_2 to reselect replying slots.

The reader calculates $\mathbf{v}_s$ based on the known tags. For an expected collision slot, if it can be resolved with r'_l but cannot be resolved with any of $r'_1, \ldots, r'_{l-1}$, the element for this slot is set to "$\underbrace{0 \ldots 0}_{l-1}1$". If none of the k seeds can be used to resolve this collision slot, the element for this slot is simply set to "1". It is obviously that a larger k leads to better performance but also increases the average length of elements. However, as we will see in Section VI-A, both the performance gain and the increase in average element length become insignificant after k is larger than a threshold.

2) BUIP-CF Analysis: We now analyze the deactivation probability in BUIP-CF. Notations here have the same meaning as in previous analysis. Assume that k seeds are used and they are independent with each other. For a collision slot that is chosen by k_1 known tags and k_2 unknown tags, the probability that it is resolved exactly with r'_j is

$$p_s(k_1, k_2, j) = (1 - p_s(k_1, k_2))^{j-1} p_s(k_1, k_2).$$

The probability that it is resolved with any of the k seeds is

$$p_s^k(k_1, k_2) = \sum_{j=1}^{k} p_s(k_1, k_2, j) = 1 - (1 - p_s(k_1, k_2))^k.$$

Accordingly, the value of $es(k_1, k_2)$ is

$$es(2, 0) = p_s^k(2, 0) * 2 + (1 - p_s^k(2, 0)) * 0 = 2 - (\frac{1}{2})^{k-1} \quad (14)$$

when $k_1 = 2$ and $k_2 = 0$, and in other cases it is

$$es(k_1, k_2) = p_s^k(k_1, k_2) * 1 + (1 - p_s^k(k_1, k_2)) * 0 = p_s^k(k_1, k_2). \quad (15)$$

Combining Eqns. 10, 11, 12, 14, and 15, we can compute p_d^i for BUIP-CF when k is given. Table II lists the value of p_d^i when k increases from one to seven when $n_i = 10,000, m_i = 2,000$. We can see that when k increases, the deactivation probability is greatly improved. More analysis on the impact of k on p_d^i is given in Section VI-A.

V. Unknown Tag Estimation

In our protocols the reader needs to estimate m_i in every round to set the optimal frame size. Some cardinality estimation algorithms [19]–[23] are proposed in recent years, but they are not applicable to our protocols. First, they estimate the total number of tags in the system, while our protocols only need to estimate m_i (we know the exact value of n_i). Second, we need to add a separate estimation phase in every round to use these approaches, which inevitably introduces additional overhead. We wish a method to estimate m_i with high accuracy and as few overhead as possible.

We use the information collected in the $(i-1)$th round to estimate m_i. Recall that we label unknown tags in expected empty slots. Assume that x unknown tags are labeled in y slots in the $(i-1)$th round. Tags select replying slots uniformly and randomly, thus we can estimate the number of active unknown tags in this round as $f_{i-1} * x/y$, where f_{i-1} is the frame size in the $(i-1)$th round.

The value of x and y can be obtained by tracing the labeling of unknown tags in the $(i-1)$th round. Denote by ne_1 the number of expected empty slots which is actual singleton, and denote by ne_2 the number of expected empty slots which is actual colliding. Then we have $y = ne_1 + ne_2$. Because m_{i-1} is usually far less than f_{i-1}, most collisions in expected empty slots are caused by two unknown tags. Thus we can estimate $x = ne_1 + 2 * ne_2$. The total number of unknown tags active in the current round can be estimated as

$$m'_{i-1} = \frac{f_{i-1} \times x}{y} = \frac{f_{i-1}}{ne_1 + ne_2}(ne_1 + 2 * ne_2). \quad (16)$$

Then the number of unknown tags that are still active at the beginning the ith round is estimated as $m_i = m'_{i-1} - x$. At the beginning of the first round, we simply set $f_1 = n$ because we have no knowledge of m. After the first round, we can also get an estimation of m and use the estimation as the initial frame size in DFSA [18] when collecting IDs of unknown tags.

Simulation results show that this algorithm performs well. Fig.2 plots the normalized estimation error of m averaged over all rounds in BUIP, BUIP-CE, and BUIP-CF when $n = 10,000$ and m changes from 250 to 2,000. The normalized estimation errors of BUIP and BUIP-CF are almost always smaller than 0.05. In BUIP-CE many expected empty slots are used as pairing slots thus only a small fraction of them are used in the estimation, which results slightly higher estimation error of BUIP-CE. However, in most cases the normalized estimator error of BUIP-CE is smaller than 0.1, which is satisfactory to our protocols.

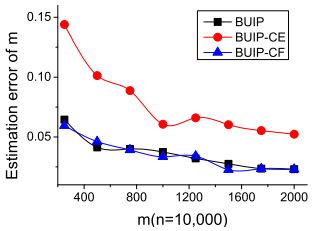


Fig. 2. Estimation accuracy of m_i.

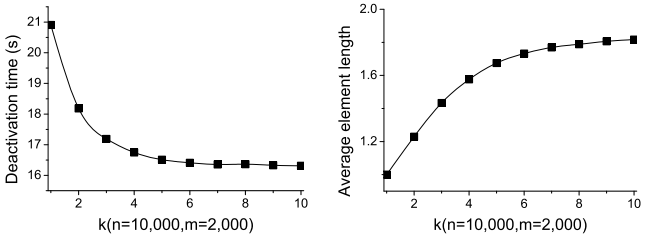
VI. Performance Evaluation

We evaluate the performance of the proposed protocols via extensive simulations. We develop a simulator in JAVA to conduct the simulations. In the simulation, we first investigate the effect of the number of reselection seeds (k) on the performance of BUIP-CF, then validate our theoretical analysis on the deactivation probability of BUIP-CE and BUIP-CF, finally evaluate the execution time of the proposed protocols. We compare the execution time of our protocols with two methods: A *Baseline* protocol which collects IDs of all the tags in the system, and an *Ideal* protocol which directly collects IDs of only unknown tags. The former represents an upper bound on the time needed to identify all the unknown tags while the latter represents a lower bound.

When computing the execution time of different protocols, we adopt the same timing scheme as in [11]: Transmitting a short response takes a time of $t_l = 0.75ms$, and transmitting a tag ID takes a time of $t_{ID} = 7.5ms$. We consider two parameters that may impact the performance of considered protocols: the number of known tags n, and the number of unknown tags m. All the data reported here are averaged over 100 independent runs.

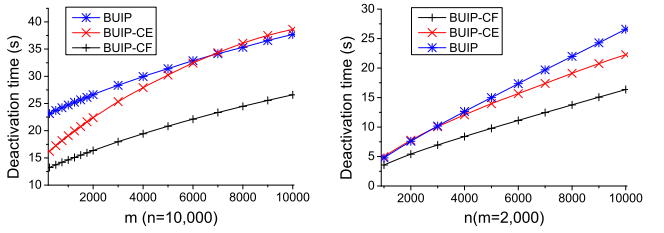
A. Effect of Reselection Times in BUIP-CF

We first investigate how the number of reselection times (k) impacts the performance of BUIP-CF. Theoretically, a larger k should lead to better performance of BUIP-CF. Fig.3(a) illustrates the time used in the deactivation phase in BUIP-CF when k varies. The deactivation time of BUIP-CF decreases along with the increase of k, which coincides well with



(a) (b)

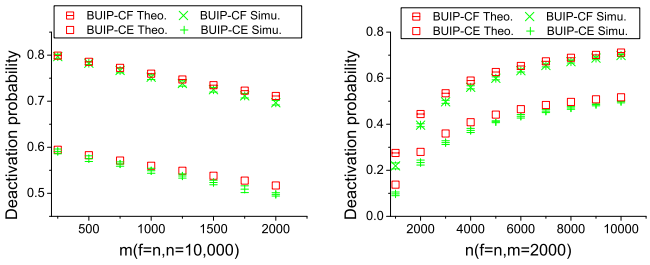
Fig. 3. Effect of reselection times on deactivation time of BUIP-CF (a), and average element length in BUIP-CF (b).



(a) (b)

Fig. 5. Deactivation time of different protocols: (a) when m changes; and (b) when n changes.

theoretical results. The reason is that the probability that a known tag is deactivated becomes higher when k increases, as shown in Table II. The average element length also increases when k increases, as Fig.3(b) shows. However, when k is larger than 7, the decrease in deactivation time and the increase in average element length both become insignificant.



(a) m (b) n

Fig. 4. Actual deactivation ratio vs. theoretical values.

B. Validation of Deactivation Probability Analysis

We then validate our theoretical analysis on the deactivation probability of BUIP-CE (Section IV-A3) and BUIP-CF (Section IV-B2). Fig.4 plots the average deactivation probability of known tags in the first round of BUIP-CE and BUIP-CF in the simulation with different combinations of n and m, together with their corresponding theoretical values. We can see that in all the cases, the actual values of deactivation probability in the simulation match the theoretical values well.

In Fig.4(a) we observe that the deactivation probability of BUIP-CE and BUIP-CF both decrease when m increases. When m increases, the chance that a slot is simultaneously chosen by both known tags and unknown tags increases, thus the probability that a known tag is deactivated in an expected empty slot decreases. Compared with BUIP-CE, the deactivation probability in BUIP-CF is 12.2% higher in average. In Fig.4(b) we observe that the deactivation probability of the two protocols both increase when n increases. When n is large, more expected singleton slots are generated, thus more known tags may be deactivated. Compared with BUIP-CE, the deactivation probability of BUIP-CF is 18.4% higher in average.

C. Deactivation Time

In this section we compare the execution time in the known tag deactivation phase in different protocols. The deactivation time of BUIP, BUIP-CE, and BUIP-CF is plotted in Fig.5. In Fig.5(a), we observe that the deactivation time of all the three protocols increase when m increases. Thanks to its high deactivation probability, BUIP-CF performs best among the three protocols. Compared with BUIP and BUIP-CE, it uses 24% and 39% less time to deactivate all the known tags, respectively. In Fig.5(b) we observe that the deactivation time of the tree protocols also increases when n increases. BUIP-CF still performs best: Compared with BUIP and BUIP-CE, it uses 29% and 34% less time to deactivate known tags, respectively.

When m/n is small (≤ 0.5), BUIP-CE outperforms BUIP. But when m/n is large (≥ 0.5), BUIP-CE performs nearly the same as BUIP, sometimes even slightly worse. The reason is that, in BUIP-CE, the slots used to pair collision slots are only "expected empty". When m is large, most of these slots are actually non-empty, thus cannot be used to deactivate known tags effectively. What's worse, slot reselection reduces the chance of unknown tags to be labeled. Thus when m/n is large, BUIP-CE cannot effectively improve over BUIP. However, BUIP-CF always outperforms BUIP because it uses fresh slots as pairing slots.

D. Total Execution Time

We compare the total execution time of our protocols with the Baseline protocol and the Ideal protocol. The results are shown in Fig.6.

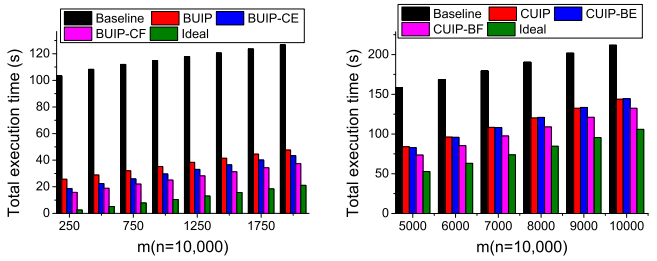
Fig.6(a) shows the execution time of considered protocols when $n = 10,000$ and m increases from 250 to 2,000. In this case, the execution time is dominated by the time used to deactivate known tags. Compared with the Baseline protocol, BUIP, BUIP-CE, and BUIP-CF reduce the execution time by 69%, 73%, and 77%, respectively. Compared with the Ideal protocol, our protocols still introduces some overhead in the deactivation of known tags. In average, the extra time used in BUIP, BUIP-CE, and BUIP-CF are 24.9s, 19.4s, and 14.8s, respectively. Considering that there are 10,000 known tags to deactivate and collecting their IDs needs more than 100s, our protocols are very efficient.

Fig.6(b) shows the execution time of considered protocols when $n = 10,000$ and m increases from 5,000 to 10,000. In this case, the execution time is dominated by the time used to collect IDs of the unknown tags. Compared with the Baseline protocol, BUIP, BUIP-CE, and BUIP-CF reduce the execution by 43%, 43%, and 49%, respectively. We can also observe that in this case the performance gap between our protocols and the Ideal protocol is rather small. In average, the extra time

TABLE III
TOTAL EXECUTION TIME OF BUIP-CF AND CU WHEN m CHANGES($n = 10000$).

Alg. Name	Total Execution Time (s)									
	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
CU(99%)	49.63	62.92	80.94	98.80	117.3	135.6	153.5	171.8	180.6	198.0
CU(95%)	36.60	47.07	62.34	77.44	93.30	109.1	124.7	140.1	156.7	172.2
CU(90%)	27.38	40.89	54.93	68.93	83.10	97.97	112.6	126.9	141.4	138.7
BUIP-CF	25.08	37.38	49.53	61.74	73.62	85.40	97.68	109.0	121.0	132.5

used to deactivate known tags in BUIP-CF (BUIP-CE, BUIP) only accounts for 26% (34%, 34%) of its total execution time.



(a) (b)

Fig. 6. Total execution time of different protocols: (a) small m; and (b) large m. BUIP-CF uses 63% (85%) less time than Baseline in average (at most).

E. Comparison with Continuous Scanning

The CU protocol proposed in [11] can collect a required fraction of all the unknown tags with probability higher than a specified value. We compare the time needed for CU to collect 90%, 95% and 99% unknown tags with the execution time of BUIP-CF. Here the parameters in CU are determined by using the methods given in [11]. Table III lists the results (CU(90%) means the time for CU to collect at least 90% of all the unknown tags, similar for CU(95%) and CU(99%)). We can see that BUIP-CF uses less time than CU to collect all the unknown tags. CU runs multiple rounds to guarantee that the required fraction of unknown tags are collected. In every round of CU, all the known tags participate in the protocol. In contrast, in BUIP-CF the number of known tags decreases after every round, thus its execution time is less.

VII. CONCLUSION

In this paper, we propose a series of protocols to perform fast and complete unknown tag identification in a large RFID systems. Simulation results show the superior performance of the proposed protocols. While in an ideal unknown identification protocol the execution time should only depends on the unknown tags, in the proposed protocols the execution time is still impacted by known tags to some extent. In the future, we will investigate how to further reduce the impact of known tags on collecting unknown tags.

VIII. ACKNOWLEDGMENT

This work is partially supported by HK RGC PolyU 5299/11E, National Natural Science Foundation of China under Grant No. 61103203.

REFERENCES

[1] Vinod Namboodiri and Lixin Gao. Energy-aware tag anti-collision protocols for RFID systems. *IEEE PerCom*, pages 23–36, 2007.
[2] EPCglobal. Epc radio-frequency identity protocols class-1 generation-2 uhf RFID protocol for communications at 860 mhz-960 mhz version 1.2.0. *Specification for RFID Air Interface*, 2008.
[3] V. Namboodiri and L. Gao. Energy-aware tag anticollision protocols for RFID systems. *IEEE Trans. on Mobi. Comp.*, pages 44–59, 2009.
[4] D.H. Shih, P.L. Sun, and D.C. Yen et al. Taxonomy and survey of RFID anti-collision protocols. *Eslevier Journal of Computer communications*, 29(11):2150–2166, 2006.
[5] Z. Bin, M. Kobayashi, and M. Shimizu. Framed aloha for multiple RFID objects identification. *IEICE Trans. on Commun*, 88(3):991–999, 2005.
[6] Z. Zhou, H. Gupta, and S.R. Das et al. Slotted scheduled tag access in multi-reader RFID systems. *ACM ICNP*, pages 61–70, 2007.
[7] Thomas F. La Porta, Gaia Maselli, and Chiara Petrioli. Anti-collision protocols for single-reader RFID systems: Temporal analysis and optimization. *IEEE Trans. on Mobi. Comp.*, 10(2):267–279, 2011.
[8] Gaia Maselli, Chiara Petrioli, and Claudio Vicari. Dynamic tag estimation for optimizing tree slotted aloha in RFID networks. *IEEE MSWiM*, pages 315–322, 2008.
[9] Maurizio A. Bonuccelli, Francesca Lonetti, and Francesca Martelli. Dynamic tag estimation for optimizing tree slotted aloha in RFID networks. *IEEE WOWMOM*, pages 603–608, 2006.
[10] K. Bu, B. Xiao, and Q. Xiao et al. Efficient Pinpointing of Misplaced Tags in Large RFID Systems. *IEEE SECON*, pages 260–268, 2011.
[11] B. Sheng, Q. Li, and W. Mao. Efficient continuous scanning in RFID systems. *IEEE Infocom*, pages 1–9, 2009.
[12] K. Bu, B. Xiao, Q. Xiao, and S. Chen. Efficient misplaced-tag pinpointing in large RFID systems. *IEEE Transactions on Parallel and Distributed Systems*, 2012.
[13] T. Li, S. Chen, and Y. Ling. Identifying the missing tags in a large RFID system. *ACM Mobihoc*, pages 1–10, 2010.
[14] C.C. Tan, B. Sheng, and Q. Li. How to monitor for missing RFID tags. *IEEE ICDCS*, pages 295–302, 2008.
[15] R. Zhang, Y. Liu, and Y. Zhang et al. Fast Identification of the Missing Tags in a Large RFID System. *IEEE SECON*, pages 1–9, 2011.
[16] S. Chen, M. Zhang, and B. Xiao. Efficient information collection protocols for sensor-augmented RFID networks. *IEEE Infocom*, pages 3101–3109, 2011.
[17] T. Li, S. Wu, and S. Chen et al. Energy efficient algorithms for the RFID estimation problem. *IEEE Infocom*, pages 1–9, 2010.
[18] S.R. Lee, S.D. Joo, and C.W. Lee. An enhanced dynamic framed slotted aloha algorithm for RFID tag identification. *ACM MobiQuitous*, pages 166–172, 2005.
[19] M. Kodialam and T. Nandagopal. Fast and reliable estimation schemes in RFID systems. *ACM Mobicom*, pages 322–333, 2006.
[20] C. Qian, H. Ngan, and Y. Liu. Cardinality estimation for large-scale RFID systems. *IEEE Percom*, pages 30–39, 2008.
[21] M. Kodialam, T. Nandagopal, and W.C. Lau. Anonymous tracking using RFID tags. *IEEE Infocom*, pages 1217–1225, 2007.
[22] H. Han, B. Sheng, and C.C. Tan et al. Counting RFID tags efficiently and anonymously. *IEEE Infocom*, pages 1–9, 2010.
[23] Y. Zheng, M. Li, and C. Qian. Pet: Probabilistic estimating tree for large-scale RFID estimation. *IEEE ICDCS*, pages 37–46, 2011.