Identifying the Missing Tags in a Large RFID System

Tao Li Shigang Chen
Computer & Information Science & Engineering
University of Florida, Gainesville, FL, USA
{tali, sgchen}@cise.ufl.edu

Yibei Ling
Applied Research Laboratories
Telcordia Technologies, NJ, USA
lingy@research.telcordia.com

ABSTRACT

Comparing with the classical barcode system, RFID extends the operational distance from inches to a number of feet (passive RFID tags) or even hundreds of feet (active RFID tags). Their wireless transmission, processing and storage capabilities enable them to support the full automation of many inventory management functions in the industry. This paper studies the practically important problem of monitoring a large set of RFID tags and identifying the missing ones — the objects that the missing tags are associated with are likely to be missing, too. This monitoring function may need to be executed frequently and therefore should be made efficient in terms of execution time, in order to avoid disruption of normal inventory operations. Based on probabilistic methods, we design a series of missing-tag identification protocols that employ novel techniques to reduce the execution time. Our best protocol reduces the time for detecting the missing tags by 88.9% or more, when comparing with existing protocols.

Categories and Subject Descriptors

C.2.1 [**Networks Architecture and Design**]: Wireless Communication

General Terms

Algorithms, Performance

Keywords

RFID, Missing-tag Detection and Identification

1. INTRODUCTION

RFID (radio-frequency identification) tags are becoming ubiquitously available in warehouse management, object tracking and inventory control. Researchers have been actively studying RFID systems as an emerging pervasive computing platform [1, 2], which helps create a multi-billion dollar market [3]. Comparing with the classical barcode system, RFID extends the operational distance from inches to a number of feet (passive RFID tags) or even hundreds of feet (active tags). The passive tags are most common today. However, the long operational range, together with their storage and processing capabilities, make the active tags ideal for automating inventory management and object tracking in a large area. For example, imagine a large Australian farm with tens of thousands of goats. Each night after the herd returns to the barn, the workers check whether some goats are missing (due to broken fence, predator attack, sickness or other reasons). Manual counting is laborious. Electronic counting as the goats rush through the gate is either slow (one goat at a time) or unreliable (when many goats simultaneously pass the wide gate). If each goat is attached with a tag, then a RFID reader will automatically find out (1) whether there is a missing goat and (2) if there is, which goat is missing.

Important applications also exist in other settings such as warehouses, hospitals, and prisons. In a large warehouse, the manager wants to know if any merchandize (such as apparel, shoes, pallets, cases, appliances, electronics, etc.) is missing due to theft, administrative error and vendor fraud. A fully automated counting procedure that can be frequently performed will be greatly helpful. Similar situation arises in a large hospital where tens of thousands of equipment and other objects need to be tracked.

Research in RFID technologies has made significant advance in recent years. Much prior work concentrates on the *tag-collection problem*, which is to collect the IDs of a large number of tags as quickly as possible. The main challenge is to resolve radio contention when the tags compete for the same low-bandwidth channel to report their IDs. The solutions fall in two broad categories: ALOHA-based protocols [4, 5, 6] and tree-based protocols [7, 8, 9]. Other work studies the *tag-estimation problem*, which is to use statistical methods to estimate the number of tags in a large system [10, 11, 12].

This paper studies the practically important *missing-tag problem*, which is to monitor a set of RFID tags and identify the missing ones. Few research papers have investigated this problem before. It may appear that, if we are able to collect the IDs of all tags (i.e., the tag-collection problem), then we will learn which tags are missing by comparing the collected IDs with the expected IDs that are stored in a database. However, collecting a large number of tag IDs by a RFID reader is a slow process. It is an inefficient overkill if we already have the IDs in the database. More efficient protocols can be designed without the expensive operation of reading

MobiHoc'10, September 20–24, 2010, Chicago, Illinois, USA.

these IDs again from the tags. Can we use the methods for estimating the number of tags to solve the missing-tag problem? After all, if some tags are missing, we are likely to estimate a smaller-than-usual number. However, the *estimated* number is not the *exact* number of tags currently in the system. If only one tag is missing, one can hardly make an assertion based on the estimated number due to statistical variance. Note that the estimated number can turn out to be greater than the actual number.

Most related is a recent paper by Tan, Sheng and Li [13], who designed novel protocols to *detect* the missing-tag *event* with probability α when the number of missing tags exceeds m, where α and m are two system parameters. However, the protocols cannot detect the missing-tag event with certainty (i.e., $\alpha = 100\%$) and more importantly, they cannot tell which tags are missing. In addition, when α is close to one and m is small (such as 1 or 2), the overhead and detection time will be both large.

We propose a series of efficient protocols that not only detect the missing-tag event with certainty but also tell exactly which tags (and the associated objects such as the goats in the Australian farm example) are missing. The most important performance criterion is to minimize the detection time. During the protocol execution, if normal operations — such as moving goods out of a warehouse — remove some tags from the system and the tag IDs in the database are not timely updated, a false alarm will be triggered. To alleviate such confusion to the warehouse management, we shall minimize the protocol execution time in order to reduce the chance for the false alarms to occur.

Our protocol design follows two general guidelines to achieve time efficiency: One is to reduce radio collision, such that the information reported from the tags is not wasted. The other is for the tags to report their presence by each transmitting a bit, instead of a whole tag ID. To realize them, we develop a number of interesting techniques that can progressively add on top of one another to improve the system performance. In order to quantify the effectiveness of each technique, we design a series of missing-tag detection protocols, each of which adds a new technique. More specifically, the *baseline protocol* eliminates the transmission contention among the tags and reduces the amount of information to be transmitted from the tags. The *two-phase protocol* significantly reduces the number of tag IDs that need to be transmitted during the detection process. A novel technique called *tag removal* is designed to further enhance the performance of the two-phase protocol. Our *three-phase protocol* with *collision-sensitive tag removal* utilizes collision slots to identify the tags that are present and the ones that are missing. Finally, the *iterative ID-free protocol* uses a probabilistic approach to resolve the collision slots and it does not require any tag ID to be transmitted (either by the tags or by the RFID reader).

When comparing with the baseline protocol, our best protocol reduces the execution time by 69% if the parameters in the Philips I-Code system [14] are used. When comparing with the tag-collection protocols that are adapted for the missing-tag problem, our best protocol reduces the execution time by 88.9% or more. We also establish a lower bound for the minimum time it takes to identify the missing tags. The execution time of our best protocol is within a factor 2.2 of the lower bound.

2. SYSTEM MODEL

2.1 Problem and Assumption

Consider a large RFID system of N tags. Each tag carries a unique ID and has the capability of performing certain computations as well as communicating with the RFID reader wirelessly. The problem is to design efficient protocols for the reader to exchange necessary information with the tags in order to identify the missing ones.

The RFID system may use battery-powered active (or semi-passive) tags that have long transmission ranges, or use passive tags that are powered by radio waves transmitted by the reader. In order to support advanced management functions that cover a large area, when passive tags are used, we expect that a reader array is installed to extend the coverage. When there are multiple synchronized readers, we logically treat them as one.

We assume that the RFID reader has access to a database that stores the IDs of all tags. This assumption is necessary for any missing-tag detection protocol. If we do not have the IDs of the tags, even after the reader collects the IDs directly from the tags, we still do not know if any one is missing, let alone the ones that are missing, because the missing tags do not send over their information.

This assumption can be easily satisfied if the tag IDs are read into a database when new objects are moved into the system and they are removed from the database when the objects are moved out — this is what a typical inventory management procedure will do. Even if such information is lost due to a database failure, we can recover the information by executing a tag-collection protocol to read the IDs from the tags. In this case, we will not detect the tags that have already been lost because we have no way to know their existence in the first place. However, now that we have the IDs of the remaining tags, those tags that are missing after this point of time will be detected, not through the expensive tag-collection protocol but through more efficient protocols to be proposed shortly.

2.2 Time Slots

Communication between the reader and the tags is time-slotted. The reader's signal will synchronize the clocks of the tags. In our protocols, the communication is driven by the reader in a request-and-response pattern, in which the reader issues a request in a time slot and then zero, one or more tags respond in the subsequent time slot(s). If no tag responds in a slot, the slot is said to be *empty*. If one and only one tag responds, it is called a *singleton slot*. If more than one tag responds, it is a *collision slot*. More specifically, if k tags respond where $k \geq 2$, it is referred to as a *k-collision slot*.

A singleton or collision slot is also called a *non-empty slot*. If we only need to determine whether a slot is empty or non-empty, the tags can use one-bit *short responses* — '0' (idle carrier) means empty and '1' (busy carrier) means non-empty. If we need to determine whether a slot is empty/ singleton/collision, the tags should use multi-bit *long responses*. For example, the Philips I-Code system [14] requires 10 bits to distinguish a singleton slot from a collision slot.

Another way of classifying the time slots is based on their lengths: *tag slots*, *long-response slots* and *short-response slots*. The length of a tag slot is denoted as t_{tag}, which allows the transmission of a tag ID, 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_l, which allows the transmission of a long response carrying multi-bits information. The length of a short-response slot is denoted as t_s, which allows the transmission of a *short response* carrying only one bit information. Clearly, $t_s < t_l < t_{tag}$. To design a time-efficient protocol, we prefer the use of short-response slots over long-response slots or the use of long-response slots over tag slots.

In the numerical examples and the simulation settings of this paper, we determine the values of t_s, t_l, and t_{tag} based on the specification of the Philips I-Code system [14]. Using the parameters of the Philips I-Code system, it can be shown that $t_s = 0.4ms$, $t_l = 0.8ms$, and $t_{tag} = 2.4ms$ (for a 96-bit tag ID) after the required waiting times (e.g., gap between transmissions) are included.

3. MOTIVATION

3.1 Prior Art

Identifying the missing tags is an under-investigated problem that has practical importance. As we have discussed in the introduction, only the existing tag-collection protocols can be adapted to solve this problem. Although they are not specifically designed for the purpose of identifying the missing tags, we use them as a performance benchmark to demonstrate how much a specially designed protocol can do better. In a tag-collection protocol, due to signal collision, each tag may have to transmit its ID several times before the RFID reader correctly receives the ID. For example, in ALOHA-based protocols such as DFSA [15] and EDFSA [16], each tag transmits its ID for 2.72 times on average, which is the theoretically optimal value. After a tag transmits its ID, it must wait for the acknowledgement from the reader. Because the acknowledgement is a binary response ('1' for correct receipt and '0' otherwise), it can be completed in a short-response slot. Therefore, the expected protocol execution time is $2.72N(t_{tag} + t_s)$.

3.2 Lower Bound on Minimum Execution Time

We give a lower bound on the minimum execution time that any protocol can possibly achieve. Each tag has to transmit at least a short response (one bit) to announce its presence in order to avoid being classified as a missing tag by the RFID reader. Even if the reader does not transmit anything, the time it takes the tags to transmit their short responses is Nt_s, which is the best that any protocol can achieve. It is unlikely that this lower is achievable because the reader has to transmit in order to coordinate the protocol execution.

3.3 Design Guidelines

To reduce the execution time $2.72N(t_{tag}+t_s)$ towards the lower bound Nt_s, our protocol design follows two general guidelines. First, we should reduce radio collision, such that each tag transmits once instead of multiple times. By doing so, we can remove the constant factor 2.72 from the time complexity. Second, we should avoid transmitting the ID tags, each of which takes t_{tag}. Clever protocol design may be able to replace an ID transmission with a short response, which takes much shorter time t_s. Moreover, if the tags do not transmit their IDs, the acknowledgements from the RFID readers can also be removed. As we will demonstrate in the next section, there are various ways to partially realize the above goals. They build on top of one another to push the performance increasingly closer to the lower bound.

4. MISSING-TAG DETECTION PROTOCOLS

In this section, we propose five new protocols for detecting the missing tags in a large RFID system.

4.1 Baseline Protocol

We observe that, since the RFID reader has access to the database of tag IDs, it does not have to read such information directly from the tags. Instead, it can broadcast these IDs one after another. After it transmits an ID, it waits for a short response from the tag that carries the ID. If it receives the response, the tag must be in the system; otherwise, the tag is missing. The verification of each tag's existence takes $t_{tag} + t_s$, and the total execution time is $N(t_{tag} + t_s)$. This is called the *baseline protocol*. Comparing with the tag-collection protocols, it significantly reduces the execution time by eliminating the contention among the tags.

4.2 Two-Phase Protocol (TPP)

We propose a two-phase protocol (TPP) to reduce the number of tag IDs that the RFID reader has to transmit. The protocol consists of two phases: a *frame phase* and a *polling phase*. The frame phase verifies the presence for a majority of the tags without any ID transmission. At the beginning of this phase, the RFID reader transmits a request $\langle r, f\rangle$, where r is a random number and f is the frame size. The frame consists of f short-response time slots right after the request. Each tag is pseudo-randomly mapped to a slot at index $H(id, r)$, where id is the tag's ID and H is a hash function whose range is $[0..f-1]$. The tag transmits a short response at that slot. Because the reader knows the IDs of all tags, it knows which slot each tag is supposed to respond. Hence, it knows the locations of the empty, singleton and collision slots. If a slot is supposed to be singleton but the reader finds it to be empty, then the tag that is mapped to the slot must be missing. The frame phase can verify the existence of all tags that are mapped to the singleton slots. However, it cannot verify the existence of the tags that are mapped to the collision slots.

There are many efficient hash functions in the literature. In order to keep the tag's circuit simple, its hash value may be derived from a pool of pre-stored random bits: We use an offline random number generator with the ID of a tag as seed to generate a string of 200 random bits, which are then stored in the tag. (Note that the random number generator is not executed by the tag.) The bits form a logical ring. $H(id, r)$ returns a certain number of bits after the rth bit in the ring. 200 random bits provide 200 different hash values, which are sufficient for our purpose considering that the next three protocols require each tag to perform only one hash, and our final protocol requires each tag to perform several hashes on average. The hash value is no more than 17 bits when the system has 50,000 tags. Even though hashing based on 200 random bits works well in our simulations, the above hash design does not place any restriction on the number of random bits, and a number larger than 200 can be chosen when necessary.

The polling phase performs the baseline protocol on the tags that are mapped to the collision slots in the frame

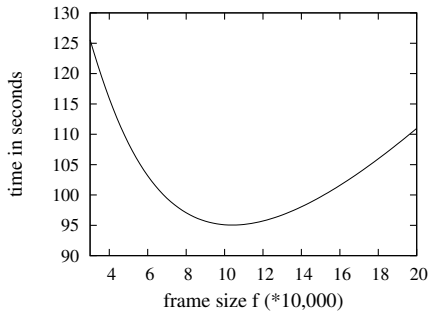

Figure 1: Execution time of TPP with respect to the frame size f. $(*10,000)$ means that the numbers along the x axis should be multiplied by 10,000.

phase. The reader broadcasts their IDs one after another. Upon receiving an ID, the tag that carries the same ID transmits a short response, allowing the reader to learn its presence.

Next, we show how to set the value of the protocol parameter f. Our goal is to find the optimal value of f that minimizes the expected protocol execution time. The execution time of TPP, denoted as T_1, is given below:

$$T_1 = (t_{tag} + t_s) \times N_1 + f \times t_s,$$

where N_1 is the number of tags mapped to the collision slots. N_1 is a random variable whose distribution is dependent on the value of f. So is T_1.

$$E(T_1) = (t_{tag} + t_s) \times E(N_1) + f \times t_s \quad (1)$$

Consider an arbitrary slot in the frame. The probability for exactly i tags to be mapped to the slot is $p_i = \binom{N}{i}(\frac{1}{f})^i(1-\frac{1}{f})^{N-i}$. Under the condition that this is a collision slot (i.e., $i \geq 2$), the *expected number* of tags that are mapped to this slot is $\sum_{i=2}^{N} ip_i$. There are f slots in the frame. Hence, the expected number of tags mapped to all collision slots, $E(N_1)$, is $f\sum_{i=2}^{N} ip_i$. Therefore, we have

$$\begin{aligned} E(N_1) &= f\sum_{i=2}^{N} i\binom{N}{i}(\frac{1}{f})^i(1-\frac{1}{f})^{N-i} \\ &= N - N(1-\frac{1}{f})^{N-1} \\ &\approx N - N\cdot exp\{-\frac{N-1}{f}\} \quad as\ N, f \to \infty. \end{aligned} \quad (2)$$

From (1) and (2), we know that $E(T_1)$ is a function of f. To compute the minimum value of $E(T_1)$, we let the first derivative of $E(T_1)$ be zero.

$$\frac{dE(T_1)}{df} = (t_{tag} + t_s) \times \frac{dE(N_1)}{df} + t_s = 0, \quad (3)$$

where $\frac{dE(N_1)}{df}$ can be derived from (2) as follows:

$$\frac{dE(N_1)}{df} \approx \frac{-N(N-1)}{f^2} \cdot exp\{-\frac{N-1}{f}\}. \quad (4)$$

We find the optimal frame size f that minimizes $E(T_1)$ by solving (3) numerically. For example, when $N = 50,000$ (imagining a large warehouse with 50,000 cell phones or a military base storing 50,000 guns and ammunition packages), Figure 1 shows the value of $E(T_1)$ with respect to f. The curve is calculated based on (1) and (2). The optimal frame size computed from (3) is 104,028, and the minimum execution time of TPP is 95.04 seconds.

4.3 Two-Phase Protocol with Tag Removal (TPP/TR)

TPP can be further improved. Suppose two tags, x and y, are mapped to a collision slot in the frame phase. When the reader detects the slot is non-empty, it cannot determine whether both tags are present or only one of them is present. Hence, it has to broadcast both IDs in the polling phase. This approach is inefficient because the information carried in the collision slot is totally unused. To make the collision slot useful, we shall turn it into a singleton slot by removing one of the two tags from the frame phase. If we remove x from the frame phase (so that it does not transmit any short response), y has a singleton slot and thus its presence can be verified. In the polling phase, we only need to broadcast the ID of x (instead of the IDs of both x and y).

Our third protocol, TPP/TP, also has two phases, but the polling phase goes before the frame phase. In the polling phase, a tag removal procedure is invoked to determine the set S of tags that will not participate in the frame phase. In this procedure, the reader first maps the tags to the slots as what TPP does. For each k-collision slot, it randomly removes $k-1$ tags to turn the slot into a singleton. The removed tags are inserted in S. After all collision slots are turned into singletons, the reader broadcasts the IDs of the tags in S one after another to verify their presence. When a tag receives its ID, it will transmit a short response and keep silent in the frame phase. The frame phase is the same as in TPP except that the tags in S do not participate.

The execution time of TPP/TP, denoted as T_2, is

$$T_2 = (t_{tag} + t_s) \times N_2 + f \times t_s,$$

where N_2 is the number of tags in S. We want to find the optimal value of f that minimizes the expected value of E_2.

$$E(T_2) = (t_{tag} + t_s) \times E(N_2) + f \times t_s \quad (5)$$

Following a similar process as we derive $E(N_1)$ in the previous subsection, we can derive $E(N_2)$ as a function of f.

$$E(N_2) = f\sum_{i=2}^{N}(i-1)\binom{N}{i}(\frac{1}{f})^i(1-\frac{1}{f})^{N-i} \quad (6)$$

We make the following simplification:

$$f\sum_{i=2}^{N} i\binom{N}{i}(\frac{1}{f})^i(1-\frac{1}{f})^{N-i} = N - N(1-\frac{1}{f})^{N-1} \quad (7)$$

and

$$\sum_{i=2}^{N}\binom{N}{i}(\frac{1}{f})^i(1-\frac{1}{f})^{N-i} = 1-(1-\frac{1}{f})^N - \frac{N}{f}(1-\frac{1}{f})^{N-1}. \quad (8)$$

Applying (7) and (8) to (6), we have

$$\begin{aligned} E(N_2) &= N\left[1-(1-\frac{1}{f})^{N-1}\right] - f\left[1-(1-\frac{1}{f})^N - \frac{N}{f}(1-\frac{1}{f})^{N-1}\right] \\ &= N - f + f(1-\frac{1}{f})^N \approx N - f + f\cdot exp\{-\frac{N}{f}\}. \end{aligned} \quad (9)$$

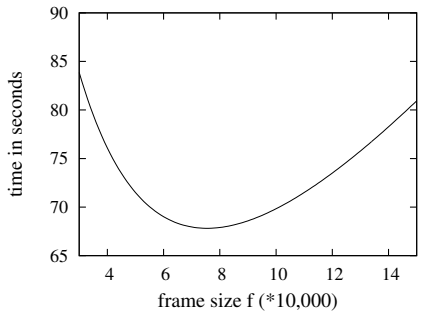

Figure 2: Execution time of TPP/TR with respect of the frame size f.

According to (5) and (9), $E(T_2)$ is a function of f. Figure 2 shows the value of $E(T_2)$ with respect to f when $N = 50,000$. To compute the minimum value of $E(T_2)$, we let the first derivative of $E(T_2)$ be zero.

$$\frac{dE(T_2)}{df} = (t_{tag} + t_s) \times \frac{dE(N_2)}{df} + t_s = 0, \qquad (10)$$

where $\frac{dE(N_2)}{df}$ can be derived from (9) as follows:

$$\frac{dE(N_2)}{df} \approx -1 + exp\{-\frac{N}{f}\} + \frac{N}{f} \cdot exp\{-\frac{N}{f}\}. \qquad (11)$$

Solving (10) numerically, we can find the optimal frame size f that minimizes $E(T_2)$. For example, the optimal frame size in Figure 2 is 75,479.

4.4 Three-Phase Protocol with Collision Sensitive Tag Removal (TPP/CSTR)

To make further improvement on TPP/TR, we observe that when f is reasonably large, most collision slots are 2-collision slots. Consider an arbitrary 2-collision slot to which two tags are mapped. If the tags transmit short responses, the reader cannot distinguish the following two cases: (1) both tags are present and (2) only one tag is present. That is because in either case the reader detects the same non-empty slot. However, if the tags transmit long responses, the reader will observe a collision slot if both tags are present, and it will observe a singleton slot if only one tag is present. Hence, observing an expected collision slot confirms that both tags are not missing, whereas observing an unexpected singleton slot means one of the tags is missing (but we do not know which one is missing). If an expected collision slot turns out to be empty, then both tags are missing.

The above idea leads to our fourth protocol, TPP/CSTR, which has three phases: a polling phase, a frame phase, and then another polling phase. At the beginning of the first polling phase, TPP/CSTR executes a different tag removal procedure: The reader maps the tags to the slots in the same way as TPP does. For each k-collision slot with $k \geq 3$, it randomly removes $k-2$ tags to turn the slot into a 2-collision slot. The removed tags are inserted in S. After all collision slots are turned into 2-collision slots, the reader broadcasts the IDs of the tags in S one after another to verify their presence. When a tag receives its ID, it will transmit a short response and keep silent in the frame phase.

In the frame phase, the tags that are not in S transmit long responses. The reader records the slots that are expected to be 2-collision slots but turn out to be singletons. Only the tags that are mapped to these slots cannot be verified. Hence, in the second polling phase that follows the frame phase, the reader broadcasts the IDs of these tags to verify their presence.

The execution time of TPP/CSTR is given below.

$$T_3 = (t_{tag} + t_s) \times (N_3 + M) + f \times t_l,$$

where N_3 is the number of tags whose IDs are broadcast in the first polling phase and M is the number of tags whose IDs are broadcast in the second polling phase. Let L be the number of missing tags. It is easy to see that

$$M \leq 2L, \qquad (12)$$

because each missing tag can produce at most one case in which an expected 2-collision slot becomes a singleton. In such a case, the IDs of the two tags mapped to the slot will be broadcast in the second polling phase. Clearly, when no tag is missing (i.e., $L = 0$), the second polling phase does not exist because $M = 0$.

Since M is unknown, the reader cannot determine the optimal value of f that minimizes $E(T_3)$. Instead, it determines the optimal value of f that minimizes the execution time of the first polling phase and the frame phase. This is reasonable because we expect the missing-tag events are relatively rare. If the protocol is executed once every hour in a warehouse and theft happens once in a week, then $M = 0$ for 167 out 168 executions. For the one execution when $M \neq 0$, spending more time is well justified to identify the lost object.

Let T_3' be the combined execution time of the first polling phase and the frame phase.

$$T_3' = (t_{tag} + t_s) \times N_3 + f \times t_l$$

$$E(T_3') = (t_{tag} + t_s) \times E(N_3) + f \times t_l \qquad (13)$$

Following the procedure of deriving $E(N_1)$ in Section 4.2, we can derive $E(N_3)$ as a function of f.

$$\begin{aligned} E(N_3) &= f \sum_{i=3}^{N} (i-2) \binom{N}{i} (\frac{1}{f})^i (1-\frac{1}{f})^{N-i} \\ &= N - 2f + 2f(1-\frac{1}{f})^N + N(1-\frac{1}{f})^{N-1} \\ &\approx N - 2f + 2f \cdot exp\{-\frac{N}{f}\} + N \cdot exp\{-\frac{N-1}{f}\}. \end{aligned} \qquad (14)$$

According to (13) and (14), $E(T_3')$ is a function of f. Figure 3 shows the value of $E(T_3')$ with respect to f when $N = 50,000$. To compute the minimum value of $E(T_3')$, we let the first derivative of $E(T_3')$ be zero.

$$\frac{dE(T_3')}{df} = (t_{tag} + t_l) \times \frac{dE(N_3)}{df} + t_l = 0 \qquad (15)$$

where $\frac{dE(N_3)}{df}$ can be derived from (14) as follows:

$$\begin{aligned} \frac{dE(N_3)}{df} \approx -2 + 2 \cdot exp\{-\frac{N}{f}\} + \frac{2N}{f} \cdot exp\{-\frac{N}{f}\} \\ + \frac{N(N-1)}{f^2} \cdot exp\{-\frac{N-1}{f}\}. \end{aligned} \qquad (16)$$

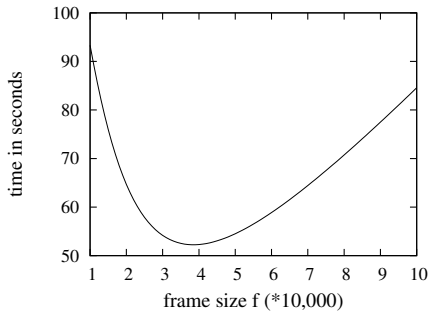

Figure 3: Execution time of TPP/CSTR with respect of the frame size f

Solving (15) numerically, we can find the optimal frame size f that minimizes $E(T_3')$. For example, when $N = 50,000$, the optimal frame size is 38,466.

4.5 Iterative ID-free Protocol (IIP)

Transmitting the tag IDs in the polling phase is an expensive operation. In our final protocol (IIP), we remove the polling phase all together. IIP iteratively performs the frame phase. Each frame verifies the presence of a portion of the tags. It repeats until short responses are received from all tags that are present. The tags whose responses are not received must be missing.

Let S be the set of tags whose presence has been verified in the previous frames. Before a frame begins, the reader maps the tags not in S to the slots of the frame in the same way as TPP does. When the reader sends the request $\langle r, f\rangle$ to the tags, it also transmits a *pre-frame vector*, which consists of f bits, each indicating the expected state of one slot, '0' for empty or singleton and '1' for collision. Recall that a tag is mapped to the slot of index $H(id, r)$ in the frame. Since the reader knows all tags, it has the full knowledge of which are the collision slots. If a tag learns that it is mapped to a collision slot (i.e., the bit at index $H(id, r)$ in the pre-frame vector is '1'), it will decide with 50% probability to not participate in the current frame. More specifically, the tag performs another hashing $H'(id, r)$ whose result is either '0' or '1'. Only when the hashing result is '1', it will participate in the current frame by transmitting a short response at slot $H(id, r)$. Since half of the tags mapped to collision slots will not participate, it helps resolve some collision slots and turn them into singletons. Knowing all the IDs, the reader can also determine which tags will not participate, which collision slots will be turned into singletons, and which other tags will respond in those singletons and thus be verified.

After the tags respond in the slots of the frame, the reader measures the state of the slots and constructs a post-frame vector, consisting of f bits, each indicating the actual state of one slot, '0' for empty or collision and '1' for singleton. The presence of the tags that respond in the singleton slots is successfully verified, and the reader inserts them into S. It then transmits the post-frame vector. If a tag sees that its slot is a singleton (i.e., the bit at index $H(id, r)$ in the post-frame vector is '1'), it will not participate further in the protocol execution. After transmitting the post-frame vector, the reader starts the next frame with a reduced size because there are fewer tags left to respond.

When no tag responds in a frame, the reader will repeat the same frame with a pre-frame vector of all zeros, which essentially requires all remaining tags, if there are any, to respond. If still no tag responds, the protocol terminates.

If the size of a pre-frame or post-frame vector is too long, the reader divides it into segments of 96 bits (equivalent to the length of the tag IDs) and transmits each segment in a time slot of size t_{tag}. Knowing the index $H(id, r)$, each tag knows from which segment it should look for the information needed.

In the following, we determine an appropriate size for each frame. The execution time for a frame of size f is

$$T_4 = 2\lceil \frac{f}{96} \rceil \times t_{tag} + f \times t_s.$$

Let N^* be the number of tags whose presence has not been verified before the current frame, and X_i be the random variable for the number of tags that respond in the ith slot of the frame. Since each of the N^* tags will randomly choose a slot to respond, we have

$$Prob\{X_i = k\} = \binom{N^*}{k} (\frac{1}{f})^k (1 - \frac{1}{f})^{N^*-k}. \quad (17)$$

If a slot is a singleton, the corresponding tag can be verified. If a slot is a k-collision one ($k > 1$), according to our protocol, it will be turned into a singleton with probability

$$\binom{k}{1} (\frac{1}{2})^1 (1 - \frac{1}{2})^{k-1} = k(\frac{1}{2})^k.$$

Hence, under our protocol, the probability for a slot to become a singleton is

$$\begin{aligned}
&Prob\{X_i = 1\} + \sum_{k=2}^{N^*} Prob\{X_i = k\} \times k(\frac{1}{2})^k \\
&= \tbinom{N^*}{1} (\frac{1}{f})^1 (1 - \frac{1}{f})^{N^*-1} + \sum_{k=2}^{N^*} \tbinom{N^*}{k} (\frac{1}{f})^k (1 - \frac{1}{f})^{N^*-k} \cdot k(\frac{1}{2})^k \\
&= \frac{N^*}{2f} \left[(1 - \frac{1}{2f})^{N^*-1} + (1 - \frac{1}{f})^{N^*-1} \right] \\
&\approx \frac{N^*}{2f} \left[exp\{-\frac{N^*}{2f}\} + exp\{-\frac{N^*}{f}\} \right]. \quad (18)
\end{aligned}$$

There are f slots in the frame, each having the above probability to be a singleton. Let N' be the expected number of tags whose presence will be verified by the frame. We must have

$$\begin{aligned}
N' &\approx f \cdot \frac{N^*}{2f} \left[exp\{-\frac{N^*}{2f}\} + exp\{-\frac{N^*}{f}\} \right] \\
&= \frac{N^*}{2} \left[exp\{-\frac{N^*}{2f}\} + exp\{-\frac{N^*}{f}\} \right]. \quad (19)
\end{aligned}$$

The average time for verifying the presence of one tag is

$$\begin{aligned}
\frac{T_4}{N'} &\approx \frac{2\lceil \frac{f}{96} \rceil \times t_{tag} + f \times t_s}{\frac{N^*}{2} \left[exp\{-\frac{N^*}{2f}\} + exp\{-\frac{N^*}{f}\} \right]} \\
&\approx \frac{0.9}{\rho \cdot [exp\{-\rho/2\} + exp\{-\rho\}]}, \quad (20)
\end{aligned}$$

where $\rho = N^*/f$ is called the *load factor*, $t_s = 0.4ms$ and $t_{tag} = 2.4ms$ (based on the parameters in [14]). The average time spent per tag, $\frac{T_4}{N'}$, is a function of ρ. Figure 4 shows the value of $\frac{T_4}{N'}$ with respect to ρ.

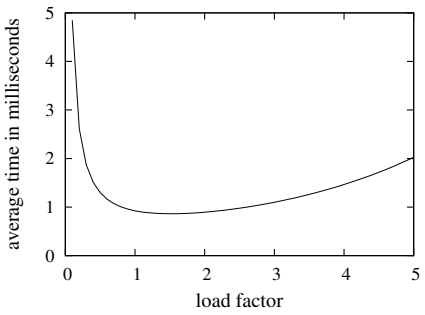

Figure 4: Average time for verifying the presence of one tag with respect to the load factor ρ

We can minimize $\frac{T_4}{N'}$ by maximizing the denominator of (20), i.e., $\rho \cdot [exp\{-\rho/2\} + exp\{-\rho\}]$. Using a numerical method such as bisection search, we obtain the optimal value $\rho = 1.516$, such that the average time for verifying the presence of one tag is minimized to $0.86ms$. In order to achieve a load factor of 1.516, the frame size must be set to $f = N^*/1.516$.

It is important to see that $\frac{T_4}{N'}$ only depends on ρ but not N^* (which is the number of tags whose presence has not been verified at the beginning of a frame). Hence, if we choose the same load factor $\rho = 1.516$ for all frames, the average time for verifying the presence of a tag becomes a constant $0.86ms$ across all frames during the execution of IIP. In this case, the execution time of the protocol is $0.86ms \times N$. Recall that a lower bound for the minimum execution time of any missing-tag detection protocol is $t_s N = 0.4ms \times N$. Hence, IIP is within a factor of 2.2 from the optimal.

4.6 Correctness and Impact of Imperfect Channel

If the wireless channel is error-free, all our protocols are able to identify any missing tag. The reason is that the presence of a tag can be unambiguously verified either by the RFID reader explicitly transmitting the tag ID and polling for the tag's response, or by implicitly mapping the tag to a singleton slot in a time frame where the tag can announce its existence without collision. Our protocols except for TPP/CSTR do just that. They either assign each tag a singleton slot, or else explicitly poll for the tag's response. The only exceptional case is in TPP/CSTR: For two tags that are mapped to an expected 2-collision slot, their presence is verified if the slot turns out to be indeed a collision slot. This is obviously true because no other tag is mapped to the slot and if one of the two tags does not respond, there will be no collision. We omit the detailed correctness proof for the five protocols due to space limitation.

However, if the channel is not error-free, it can cause false positives and false negatives. This is not only true for our protocols but also true for others. For example, suppose a missing tag is mapped to a singleton slot. The slot is supposed to be empty. However, a false positive may occur if the RFID reader senses a busy channel due to high noise. Channel errors can also cause mis-detection in [13] even though it is not designed to identify each individual missing tag. Occasional false positives do not impose a serious problem because a missing-tag detection protocol is executed periodically and a missing tag that is undetected due to a false positive in one execution round will be detected in a later round. If the channel error is significant, we can replace all short-response slots in our protocols with long-response slots that carry noise-resistent multi-bit checksums of the tags. Consider a missing tag that is mapped to a singleton slot. Suppose the tag sends a 10-bit checksum of its ID (which is 96 bits for the GEN2 standard). Even when the slot is non-empty, if the reader does not receive the correct checksum, it will not confirm the existence of the tag. The tag must be queried again by the polling phase or, in the case of IIP, by the subsequent frames.

A false negative occurs when the RFID reader transmits the ID of a tag to poll for its response, but the transmission is corrupted by channel error and consequently the tag does not respond. In this case, the reader believes that the tag, which is not missing, does not exist. False negatives can also happen in the prior tag-collection protocols (Section 3.1). Suppose two tags collide in their ID transmissions. The negative acknowledgement, a flag of '0', from the reader may be changed to a positive one, a flag of '1', due to channel error. As the tags stop transmitting their IDs, the reader will treat them as missing ones. For all protocols, the false negatives can be easily handled in the same way: The reader performs an extra verification step that polls each "missing" tag to see if it responds.

False positives and false negatives may also happen if tags are moved in or out of the system during protocol execution. They are handled by the approaches described above. However, in order to reduce the false alarms caused by normal inventory operations, we should minimize the execution time. This is where our protocols enjoy significant advantages, as the next section will demonstrate through simulations.

5. SIMULATION RESULTS

In this section, we evaluate the efficiency of the baseline protocol, TPP, TPP/TR, TPP/CSTR and IIP by simulations. We compare our protocols with the state-of-the-art protocols in the related work. They are the Trusted Reader Protocol (TRP) [13], the Enhanced Dynamic Framed Slotted ALOHA (EDFSA) [16] and the Binary Tree Protocol (BTP) [17].

Two performance metrics are used: (1) the execution time of a protocol before it identifies all missing tags, and (2) the execution time of a protocol before it detects the first missing tag. The first performance metric tells us how long it takes a protocol to identify exactly which tags are missing. The second metric tells us how long it takes a protocol to identify the missing-tag event. We run each simulation 100 times with different random seeds and average the results to produce a data point.

Based on the specification of the Philips I-Code system [14], after the required waiting times (e.g., gap between transmissions) are included, a reader needs $0.4ms$ to detect an empty slot, $0.8ms$ to detect a collision or a singleton slot, and $2.4ms$ to transmit a 96-bit ID. Our protocols except for TPP/CSTR need only to identify empty and non-empty slots. TPP/CSTR has to identify empty, singleton and collision slots. Therefore, the tags in the baseline protocol, TPP, TPP/TR, IIP and TRP transmit short responses, each taking $t_s = 0.4ms$ and the tags in TPP/CSTR transmit long responses, each taking $t_l = 0.8ms$. EDFSA and BTP require the tags to transmit their IDs, each taking $t_{tag} = 2.4ms$.

Unless specified otherwise, the default number of missing tags in the simulations is 1.

We do not simulate channel errors. As we have explained in Section 4.6, if necessary, the false positives caused by channel errors can be handled by replacing short-response slots with long-response slots, which may double the execution time of our protocols. That however does not change the basic conclusions of this paper because, even after doubling, our execution times are still far smaller than the times of other protocols before they take channel errors into account.

5.1 Time Efficiency

The first set of simulations evaluates the time efficiency of the protocols. We compare our five protocols with EDFSA and BTP for the time it takes each of them to identify all missing tags. TRP [13] is not designed to identify individual missing tags. Instead, it detects the event that at least one tag is missing with a certain probability α when the number of missing tags exceeds m. For this set of simulations, $m = 0$, and we let $\alpha = 95\%$. Even although TRP does not achieve what the other protocols do, we include the results of TRP because it is the only existing work that explicitly deals with a variant of the missing-tag problem. For EDFSA [16], we remove its component for estimating the number N of tags because in our model the reader knows the information of the tags.

We vary N from 5,000 to 100,000. Table 1 presents the execution times of the protocols. Our baseline protocol performs much better than TRP, EDFSA and BTP, cutting the execution time by more than half when comparing with the best result of these existing protocols. For example, when $N = 50,000$, the time of the baseline protocol is 35.9% of the time taken by TRP, 36.1% of the time by EDFSA, and 34.7% of the time by BTP. Our other protocols, TPP, TPP/TP, TPP/CSTR and IIP, perform increasingly better. The execution time of IIP is around 30.7% of the time taken by the baseline protocol.

The ratio between the execution time required by TRP (EDFSA, BTP) and that by our best protocol IIP is around 9.1 (9.0, 9.4). When $N = 50,000$, TRP (EDFSA, BTP) requires 390.0 (387.8, 404.4) seconds while IIP requires only 43.0 seconds, representing 89.0% (88.9%, 89.4%) reduction in the execution time.

5.2 Time to Detect the First Missing Tag

The second set of simulations studies the relation between the number of missing tags and the time to detect the first missing tag. It takes less time to find out whether some tag(s) is missing than to actually identify them. This is also an important function for RFID systems that require a quick answer on whether the set of tags is intact. For TRP, $\alpha = 100\%$ in this set of simulations.

We set $N = 50,000$ and vary the number L of missing tags from 1 to 50. Note that $L = 0$ means the set of tags is intact. Table 2 shows that the detection time of all protocols except for TRP decreases as L increases. That is because more missing tags make it easier to detect one of them. The detection time of TRP is a constant since it requires the reader to collect the responses from all tags before the detection decision is made. From Table 2, our protocols require far less time to detect the first missing tag than TRP, EDFSA and BTP, especially when L is small. For example, when $L = 3$, TRP (EDFSA, BTP) requires 390.0 (99.8, 113.9) seconds while our best protocol IIP requires only 12.1 seconds, representing 96.9% (87.9%, 89.5%) reduction in the execution time. When $L = 50$, TRP (EDFSA, BTP) requires 390.0 (7.8, 7.2) seconds while our best protocol IIP requires only 0.8 seconds.

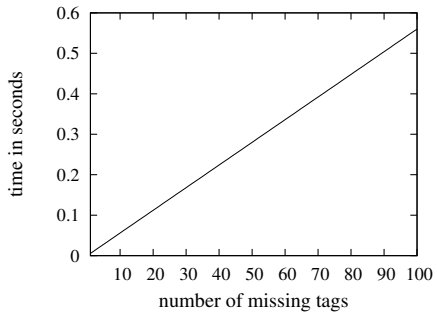

Figure 5: Execution time of the second polling phase in TPP/CSTR with respect to the number of missing tags when $N = 50,000$.

5.3 TPP/CSTR and Number of Missing Tags

Except for TPP/CSTR, the execution times of all other protocols are largely independent of the number L of missing tags when they are used to identify the missing tags. TPP/CSTR consists of three phases: the first polling phase, the frame phase and the second polling phase. The execution time of the first polling phase and the frame phase is also independent of L. However, the execution time of the second polling phase is dependent on L. In this subsection, we evaluate the impact of L on the execution time of TPP/CSTR. Figure 5 shows the execution time of the second polling phase in TPP/CSTR with respect to L. The time increases linearly in L and stays small when L is small. For example, when N is $50,000$ and L is 100, the execution time of the second polling phase is just 0.56 seconds, which is insignificant when comparing with the 52.24 seconds for the other two phases of the protocol.

6. RELATED WORK

The tag-collection protocols mainly fall into two categories. One is *tree-based* [17, 18, 19, 20, 21] and the other is *ALOHA-based* [22, 16, 15, 6, 23, 24]. The *tree-based protocols* organize all IDs in a tree of ID prefixes [17, 18, 19]. Each in-tree prefix has two child nodes that have one additional bit, '0' or '1'. The tag IDs are leaves of the tree. The RFID reader walks through the tree. As it reaches an in-tree node, it queries for tags with the prefix represented by the node. When multiple tags match the prefix, they will all respond and cause collision. Then the reader moves to a child node by extending the prefix with one more bit. If zero or one tag responds (in the one-tag case, the reader receives an ID), it moves up in the tree and follows the next branch. Another type of tree-based protocols tries to balance the tree by letting the tags randomly pick which branches they belong to [17, 20, 21].

The *ALOHA-based protocols* work as follows. The reader first broadcasts the query request. Each tag chooses a time slot to transmit its ID. If a tag selects a slot that none of

Table 1: The execution time with respect to N

N	Execution time in seconds							
	Baseline	TPP	TPP/TR	TPP/CSTR	IIP	TRP	EDFSA	BTP
5000	14.0	9.5	6.8	5.2	4.3	39.0	38.6	40.0
10000	28.0	19.0	13.6	10.4	8.6	78.0	77.1	80.8
20000	56.0	38.0	27.1	20.9	17.2	156.0	157.2	162.0
30000	84.0	57.0	40.7	31.3	25.8	234.0	231.9	242.9
40000	112.0	76.0	54.3	41.8	34.4	312.0	311.4	324.2
50000	140.0	95.0	67.8	52.2	43.0	390.0	387.8	404.4
75000	210.0	142.6	101.7	78.35	64.5	585.0	580.7	607.9
100000	280.0	190.1	135.6	104.5	86.0	780.0	776.7	807.8

Table 2: The time to detect the first missing tag with $N = 50,000$

L	Execution time in seconds							
	Baseline	TPP	TPP/TR	TPP/CSTR	IIP	TRP	EDFSA	BTP
1	70.4	36.2	43.8	33.4	23.2	390.0	194.8	229.3
2	44.6	23.8	31.0	27.0	15.4	390.0	134.9	131.1
3	39.6	18.8	28.5	22.4	12.1	390.0	99.8	113.9
4	32.2	14.6	25.8	18.8	9.8	390.0	78.2	83.5
5	21.7	9.7	21.8	16.3	7.6	390.0	60.6	56.5
10	13.1	5.6	12.4	11.5	4.0	390.0	34.5	36.4
25	5.6	2.4	5.8	5.4	1.7	390.0	13.3	12.9
50	2.7	1.3	3.0	2.6	0.8	390.0	7.8	7.2

other tags select, it can be successfully identified and will keep silent for the rest of the process. If multiple tags transmit simultaneously, the responses are garbled due to collision and retransmissions are required. The process terminates when all the tags are identified successfully. The enhanced dynamic framed slotted ALOHA (EDFSA) [16] increases the identification probability by adjusting the frame size and restricting the number of responding tags in the frame.

The tag estimation [10, 11, 4, 6, 12] is another important problem in RFID system. Kodialam and Nandagopal [10] propose a probabilistic model to estimate the number of tags. The reader uses slotted ALOHA-protocol and counts the number of empty and collision slots. Based on the obtained information, the reader generates the estimation. The process repeats until the specified accuracy is achieved. The drawback of the estimators in [10] is the reader should know approximately the magnitude of the number of tags to be estimated. The authors design an Enhanced Zero-Based (EZB) estimator in [11] in order to address the constraint mentioned above. Qian et. al [12] present the Lottery-Frame scheme (LoF) for the multiple-reader scenario. By employing the hash functions with geometric distribution, the replicate-insensitive estimation protocol achieves high accuracy with low overhead.

Tan, Sheng and Li [13] design the Trust Reader Protocol (TRP) to detect the missing-tag event with probability α when the number of missing tags exceeds m, where α and m are system parameters. In TRP, the reader broadcasts a random number r and a frame size f. Based on the received random number and its ID, each tag pseudo-randomly chooses a slot in the frame to reply. A slot is denoted as '0' if no tag replies in the slot. Otherwise it is denoted as '1'. In this way, the reader can generate a bitstring of '0's and '1's from the reply. Since the reader knows all the IDs as well as the parameters $\langle r, f \rangle$, it is able to determine the resulting bitstring for an intact set. The reader compares the bitstring generated from the reply and the bitstring generated from the records, and will report that the set of tags is not intact if a mismatch is found. TRP uses probabilistic method to choose the frame size, which is the smallest value that satisfies the system accuracy requirement. However, TRP cannot detect the missing-tag event with certainty (i.e., $\alpha = 100\%$) and more importantly, it cannot tell which tags are missing. Moreover, when α is close to one and m is small (such as 1 or 2), the detection time will be extremely large.

7. CONCLUSION

In this paper, we study the problem of monitoring the set of tags in a large RFID system and identifying the missing ones. The solution to this problem has important inventory management applications in large livestock farms, warehouses, and hospitals. To avoid interfering with other normal operations, we should minimize the execution time of the protocol for identifying the missing tags. We propose five missing-tag detection protocols with increasingly better time efficiencies. A number of novel techniques are introduced in the protocols, including hybrid of frame and polling phases, tag removal, collision-sensitive tag removal, and probabilistic iterative frame phases. These new techniques achieve far smaller missing-tag detection times than the existing protocols.

8. ACKNOWLEDGMENTS

This work was supported in part by the US National Science Foundation under grant CPS-0931969. We would also like to thank our shepherd, Dr. Prabal K. Dutta, and the anonymous reviewers for their constructive comments.

9. REFERENCES

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

[2] G. Zecca, P. Couderc, M. Banatre, and R. Beraldi, "Swarm Robot Synchronization Using RFID Tags," *Proc. of IEEE PERCOM*, 2009.

[3] R. Das, "Global RFID Market Tops $5.5 Billion," *http://www.convertingmagazine.com/article/CA6653688.html*, April 2009.

[4] J. Zhai and G. N. Wang, "An Anti-Collision Algorithm Using Two-functioned Estimation for RFID Tags," *Proc. of ICCSA*, 2005.

[5] J. Cha and J. Kim, "Novel Anti-collision Algorithms for Fast Object Identification in RFID System," *Proc. of IEEE ICPADS*, 2005.

[6] H. Vogt, "Efficient Object Identification with Passive RFID Tags," *Proc. of IEEE PERCOM*, 2002.

[7] D. Hush and C. Wood, "Analysis of Tree Algorithm for RFID Arbitration," *Proc. of IEEE ISIT*, 1998.

[8] J. Myung and W. Lee, "An Adaptive Memoryless Tag Anti-collision Protocol for RFID Networks," *Proc. of IEEE ICC*, 2005.

[9] H. Choi, J. Cha, and J. Kim, "Fast Wireless Anti-collision Algorithm in Ubiquitous ID System," *Proc. of IEEE VTC*, 2004.

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

[11] M. Kodialam, T. Nandagopal, and W. Lau, "Anonymous Tracking Using RFID Tags," *Proc. of IEEE INFOCOM*, 2007.

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

[13] Chiu C. Tan, Bo Sheng, and Qun Li, "How to monitor for missing RFID tags," *Proc. of IEEE ICDCS*, 2008.

[14] Philips Semiconductors, "I-CODE Smart Label RFID Tags," *http://www.nxp.com/acrobat_download/other/identification/SL092030.pdf*, January 2004.

[15] J. R. Cha and J. H. Kim, "Dynamic Framed Slotted ALOHA Algorithms Using Fast Tag Estimation Method for RFID Systems," *Proc. of IEEE CCNC*, 2006.

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

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

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

[19] F. Zhou, C. Chen, D. Jin, C. Huang, and H. Min, "Evaluating and Optimizing Power Consumption of Anti-collision Protocols for Applications in RFID Systems," *Proc. of ISLPED*, 2004.

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

[21] J. I. Capetenakis, "Tree Algorithms for Packet Broadcast Channels," *IEEE Transactions on Information Theory*, vol. 25, no. 5, 1979.

[22] I. E. Teleta and R. G. Gallager, "Combining Queuing Theory and Information Theory for Multiaccess," *IEEE Journal on Selected Areas Communication*, vol. 13, no. 6, 1995.

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

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