Efficient Protocols for Identifying the Missing Tags in a Large RFID System

Tao Li, Shigang Chen, *Senior Member, IEEE*, and Yibei Ling, *Senior Member, IEEE*

***Abstract*—Compared to the classical barcode system, radio frequency identification (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 full automation of many inventory management functions in industry. This paper studies the practically important problem of monitoring a large set of active RFID tags and identifying the missing ones—the objects that the missing tags are associated with are likely to be missing as well. 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 an order of magnitude when compared to existing protocols.**

***Index Terms*—Radio frequency identification (RFID) tags.**

I. Introduction

RADIO frequency identification (RFID) 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]–[4], which helps create a multibillion dollar market [5]. Compared to 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 an 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 merchandise (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. A 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 [6]–[8] and tree-based protocols [9]–[11]. Other work studies the *tag-estimation problem*, which is to use statistical methods to estimate the number of tags in a large system [12]–[14].

This paper studies the practically important *missing-tag problem*, which is to monitor a set of active 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 to the expected IDs that are stored in a database. However, collecting a large number of tag IDs by an 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 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 *et al.* [15], 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

Manuscript received October 12, 2010; revised March 07, 2011 and May 27, 2012; accepted January 02, 2013; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor M. Kodialam. Date of publication March 07, 2013; date of current version December 13, 2013. This work was supported in part by the US National Science Foundation under Grant CPS 0931969.

T. Li was with the Department of Computer and Information Science and Engineering, University of Florida, Gainesville, FL 32611 USA. He is now with Google, Inc., Mountain View, CA 94043 USA (e-mail: lita@google.com).

S. Chen is with the Department of Computer and Information Science and Engineering, University of Florida, Gainesville, FL 32611 USA (e-mail: sgchen@cise.ufl.edu).

Y. Ling is with the Applied Research Laboratories, Telcordia Technologies, Morristown, NJ 07960 USA (e-mail: lingy@research.telcordia.com).

Digital Object Identifier 10.1109/TNET.2013.2245510

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 *two-hash protocol* maps each tag to two time-slots in a frame by two hash functions and chooses the singleton slot, if there is any, to transmit. It also introduces a probabilistic approach to resolve the collision slots. The two-hash protocol does not require any tag ID to be transmitted (either by the tags or by the RFID reader).

When compared to the baseline protocol, our best protocol reduces the execution time by 85.7% if the parameters in the Philips I-Code system [16] are used. When compared to the tag-collection protocols that are adapted for the missing-tag problem, our best protocol reduces the execution time by an order of magnitude. 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 1.9 of the lower bound.

The rest of this paper is organized as follows. Section II defines the system model. Section III motivates our protocols in this study. Section IV describes our protocols for solving the missing-tag problem. Section V presents the simulation results. Section VI discusses the related work. Section VII draws the conclusion.

II. System Model

A. 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.

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.

B. 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 *nonempty slot*. If we only need to determine whether a slot is empty or nonempty, the tags can use 1-bit *short responses*—"0" (idle carrier) means empty, and "1" (busy carrier) means nonempty. If we need to determine whether a slot is empty/singleton/collision, the tags should use multibit *long responses*. For example, the Philips I-Code system [16] 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 multibits 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, $t_{\mathrm{s}} < t_{\mathrm{l}} < t_{\mathrm{tag}}$. 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 [16]. Using the parameters of the Philips I-Code system, it can be shown that $t_{\rm s} = 321\ \mu$s, $t_{\rm l} = 491\ \mu$s, and $t_{\rm tag} = 3927\ \mu$s (for a 96-bit tag ID) after the required waiting times (e.g., gap between transmissions) are included. Section V will discuss them in detail.

III. Motivation

A. Prior Art

Identifying the missing tags is an underinvestigated 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 [17] and EDFSA [18], the optimal system efficiency is 36.8%, i.e., at most 36.8% of the time-slots can be successfully used to collect tag IDs. Other slots either have collisions or are empty. It means that these protocols need at least $2.72N$ slots to identify N tags. After a tag transmits its ID, it must wait for the acknowledgment from the reader. Because the acknowledgment 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 $2.72N(t_{\rm tag} + t_{\rm s})$.

B. 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 (1 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 $Nt_{\rm s}$, which is the best that any protocol can achieve. It is unlikely that this lower bound is achievable because the reader has to transmit in order to coordinate the protocol execution.

C. Design Guidelines

To reduce the execution time $2.72N(t_{\rm tag} + t_{\rm s})$ toward the lower bound $Nt_{\rm s}$, 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 $t_{\rm tag}$. Clever protocol design may be able to replace an ID transmission with a short response, which takes much shorter time $t_{\rm s}$. Moreover, if the tags do not transmit their IDs, the acknowledgments from the RFID readers can also be removed. As we will demonstrate in Section IV, 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.

IV. Missing-Tag Detection Protocols

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

A. 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 $t_{\rm tag} + t_{\rm s}$, and the total execution time is $N(t_{\rm tag} + t_{\rm s})$. This is called the *baseline protocol*. Compared to the tag-collection protocols, it significantly reduces the execution time by eliminating the contention among the tags.

B. Two-Phase Protocol (TPP)

We propose a 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 $[0 \ldots f-1]$. 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 prestored 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. Two hundred 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 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 as follows:

$$T_1 = (t_{\text{tag}} + t_{\text{s}}) \times N_1 + f \times t_{\text{s}}$$

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

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

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

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

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

$$\frac{dE(T_1)}{df} = (t_{\text{tag}} + t_{\text{s}}) \times \frac{dE(N_1)}{df} + t_{\text{s}} = 0 \tag{3}$$

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

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

We find the optimal frame size f that minimizes $E(T_1)$ by solving (3) numerically. For example, when $N = 50\,000$ (imagine a large warehouse with 50 000 cell phones or a military base storing 50 000 guns and ammunition packages), Fig. 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 154 758, and the minimum execution time of TPP is 108.32 s.

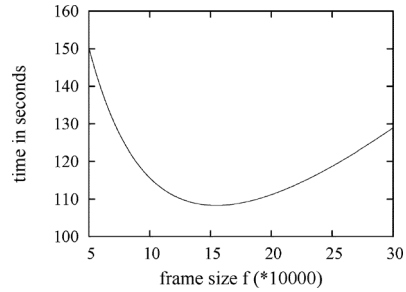


Fig. 1. Execution time of TPP with respect to the frame size f. ($*10\,000$) means that the numbers along the x-axis should be multiplied by 10 ,000.

C. 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 nonempty, 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

$$T_2 = (t_{\text{tag}} + t_{\text{s}}) \times N_2 + f \times t_{\text{s}}$$

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 T_2

$$E(T_2) = (t_{\text{tag}} + t_{\text{s}}) \times E(N_2) + f \times t_{\text{s}}. \tag{5}$$

Following a similar process as we derive $E(N_1)$ in Section IV-B, we can derive $E(N_2)$ as a function of f

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

We make the following simplification:

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

and

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

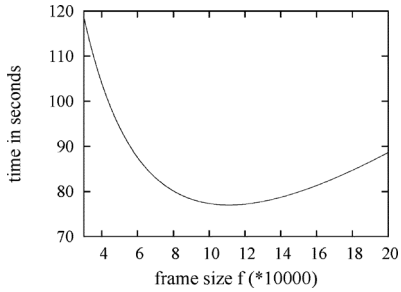


Fig. 2. Execution time of TPP/TR with respect of the frame size f.

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

$$E(N_2) = N\left(1-\left(1-\frac{1}{f}\right)^{N-1}\right) - f\left(1-\left(1-\frac{1}{f}\right)^{N}-\frac{N}{f}\left(1-\frac{1}{f}\right)^{N-1}\right) = N-f+f\left(1-\frac{1}{f}\right)^{N} \approx N-f+f\cdot\exp\left\{-\frac{N}{f}\right\}. \tag{9}$$

According to (5) and (9), $E(T_2)$ is a function of f. Fig. 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

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

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

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

Solving (10) numerically, we can find the optimal frame size f that minimizes $E(T_2)$. For example, the optimal frame size in Fig. 2 is 111 003.

D. 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 nonempty 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 as follows:

$$T_3 = (t_{\rm tag}+t_{\rm s})\times(N_3+M)+f\times t_{\rm l}$$

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

$$M \leq 2L \tag{12}$$

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. Moreover, Section V-C also shows that the second polling phase takes much smaller time than the first two phases and can be ignored.

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

$$T_3' = (t_{\rm tag}+t_{\rm s})\times N_3 + f\times t_{\rm l}$$
$$E(T_3') = (t_{\rm tag}+t_{\rm s})\times E(N_3) + f\times t_{\rm l}. \tag{13}$$

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

$$E(N_3) = f\sum_{i=3}^{N}(i-2)\binom{N}{i}\left(\frac{1}{f}\right)^{i}\left(1-\frac{1}{f}\right)^{N-i} = N-2f+2f\left(1-\frac{1}{f}\right)^{N}+N\left(1-\frac{1}{f}\right)^{N-1} \approx N-2f+2f\cdot\exp\left\{-\frac{N}{f}\right\}+N\cdot\exp\left\{-\frac{N-1}{f}\right\}. \tag{14}$$

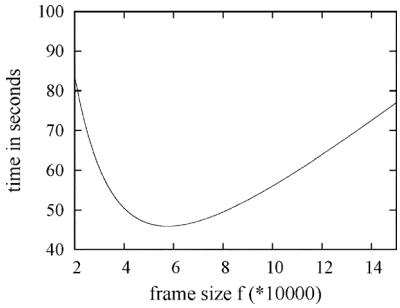


Fig. 3. Execution time of TPP/CSTR with respect of the frame size f.

According to (13) and (14), $E(T_3')$ is a function of f. Fig. 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

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

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

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

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 57 562.

E. Two-Hash Protocol (THP)

As is mentioned before, an ID transmission in the polling phase requires much longer time than a short response. Specifically, $t_{\text{tag}} = 12.2t_{\text{s}}$. In order to save time, we want to remove all ID transmissions. Our final protocol, THP, removes the polling phase all together. It iteratively performs the frame phase. Each frame verifies the presence of some tags. The protocol repeats the frame phase until the presence of all tags is successfully verified.

A time-slot can be used to verify the presence of a tag if the tag is the only one that is mapped to the slot by a hash function. The tag is expected to transmit in that slot; if the RFID reader finds that the slot is actually empty, then the tag must be missing. However, when we use a single hash function to map tags to slots, some slots may have multiple tags due to hash collision, and some others may have no tag. These slots are wasted. Our idea is to use more than one hash function to increase the number of singleton slots.

Consider an arbitrary frame in the execution of THP. Let R be the set of tags whose presence has not been verified in the previous frames. Let f be the number of time-slots in the current frame. The RFID reader maps the tags in R to the slots of the frame using two hash functions, H_1 and H_2, whose ranges are $[0, f)$. Each tag id is mapped to two slots at indices, $H_1(id, r)$ and $H_2(id, r)$, where r is a random number that is different for each frame. As long as one of the two slots is a singleton, the reader will be able to assign the tag to the slot for its response. THP uses two hash functions to increase the probability for a tag to be mapped to a singleton. It can be generalized by using more than two hash functions, which we do not explicitly discuss to avoid making the protocol too complex—an undesirable consequence for RFID tags.

The reader only assigns tags to singleton slots to which no other tags are assigned. There are two hash functions. The reader must determine which hash function is used in each time-slot. This information is recorded in a *preframe vector*, consisting of f hash indices, each of which corresponds to a time-slot in the current frame. A hash index is 2 bits long, and it indicates which function should be used for the corresponding time-slot. For example, if the first hash index in the vector is "00," it means that no hash function is used, and the first slot of the frame is not assigned to any tag. If the first hash index is "01," it means H_1 should be used when a tag transmits in the first time-slot. Similarly, a hash-index value of "10" means H_2 should be used. When a hash index is "11," it means that multiple tags are mapped to the corresponding time-slot, but the collision can be resolved, which we will discuss in more detail shortly.

Next, we explain how the reader assigns tags to slots and how to construct the preframe vector. Initially, all hash indices in the vector are set to "00," and an auxiliary set S is empty. The reader first maps tags in R to slots of the current frame by using the first hash function H_1. It identifies which tags are mapped to singleton slots and assigns these tags to their slots. The reader inserts these tags in S. It then sets the hash indices of those singleton slots as "01," meaning that only one tag is mapped to each of those slots via H_1, and the tag should transmit in the slot to confirm its presence. After that, the reader maps the remaining tags in $R - S$ to slots by using the second hash function H_2. It identifies which tags are mapped to singleton slots whose hash indices remain "00." If a slot's hash index is "00," we know that the slot is not assigned to any tag using H_1. Now, if only one tag is mapped to that slot by H_2, then the reader should assign the tag to the slot, insert the tag in S and set the corresponding hash index to be "10." Finally, we consider the case where multiple tags are mapped by H_2 to a slot whose hash index is "00." To resolve the collision, the reader applies a third hash function H_3 on the involving tags. The output of H_3 is either "0" or "1." If the condition $H_3(id, r) = 1$ is true for one and only one tag, the reader sets the hash index of the slot to "11," indicating that a tag should transmit in this slot only if it is mapped to this slot by H_2 and satisfies $H_3(id, r) = 1$. On the other hand, if $H_3(id, r) = 1$ is true for zero, two, or more tags, the reader keeps the hash index of the slot as "00," meaning that the slot is not assigned to any tag. The pseudocode is given in Algorithm 1.

Algorithm 1: $Compute\_Preframe(R, f)$

1. Generate a preframe vector F of size f. Initially each hash index in F is set to "00."
2. Create an empty auxiliary set S and pick a random number r.
3. For each tag $id \in R$, compute the hash index $H_1(id, r)$.
4. Identify which tags are mapped to singleton slots. Insert these tags in S and set the hash indices of those slots to "01."

TABLE I
TRANSMISSION DECISION TABLE

	$H_1(id,r)$	$H_2(id,r)$
00	No	No
01	Yes	No
10	No	Yes
11	No	Yes $H_3(id,r)=1$

5. For each tag $id \in R - S$, compute the hash index $H_2(id, r)$.
6. Identify which tags are mapped to singleton slots whose hash indices are "00." Set the hash indices of those slots to "10."
7. When multiple tags are mapped by H_2 to a slot whose hash index is "00," apply H_3 on these tags. If $H_3(id, r) = 1$ is true for one and only one tag, set the hash index of the slot to "11."

To begin the current frame, the reader broadcasts a request $\langle r, f \rangle$, together with the preframe vector, consisting of the hash indices of the slots in the frame. If the size of a preframe 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 t_{tag}. Note that a tag does not have to receive the whole vector. It knows it is mapped to the $H_1(id, r)$th and $H_2(id, r)$th slots, and thus it also knows which vector segments it should receive to find the corresponding hash-index information.

After a tag receives the request $\langle r, f \rangle$, it knows that it is mapped to the $H_1(id, r)$th slot by the first hash function, where id is the tag's ID. The tag checks the corresponding hash index in the preframe vector. If the hash index is "01," the tag knows that it must be the only tag that is mapped to the slot by H_1. It will transmit a short response in this slot and remain silent in other slots. On the other hand, if the hash index of the slot is not "01," the tag resorts to H_2. It checks whether the $H_2(id, r)$th hash index in the preframe vector is "10." If it is, the tag will transmit in the corresponding slot. However, if the hash index is "11" instead, the tag will compute $H_3(id, r)$ to see if the value is "1." If it is "1," the tag will still transmit in this slot. If it is not "1," the tag will stay silent in the current frame. All tags that do not transmit in the current frame will participate in the next frame. Table I summarizes the transmission decision made by a tag. The first column is the hash index value. The second and third columns show whether the tag will transmit in a slot, given a certain hash index value. If a tag transmits in the $H_1(id, r)$th slot, it will not transmit in the $H_2(id, r)$th slot. The pseudocode is given in Algorithm 2.

Algorithm 2: $THP(R, r, f, F)$

1. The reader broadcasts a request $\langle r, f \rangle$ and F.
2. Each tag $id \in R$ does the following:
3. **if** the hash index for the $H_1(id, r)$th slot is "01" **then**
4. it transmits a short response in the $H_1(id, r)$th slot and remains silent in other slots;
5. **else if** the hash index of the $H_2(id, r)$th slot is "10" **then**
6. it transmits a short response in the $H_2(id, r)$th slot and remains silent in other slots;
7. **else if** the hash index of the $H_2(id, r)$th slot is "11" and $H_3(id, r) = 1$ **then**
8. it transmits a short response in the $H_2(id, r)$th slot and remains silent in other slots.

After the frame is completed, the reader updates $R := R - S$ and performs the next frame if R is not empty. According to the design of THP, the reader always assigns no more than one tag to each slot. Hence, all frames are collision-free.

Next, we explain how to determine an appropriate size for each frame. The execution time for a frame of size f is

$$T_4 = \left\lceil \frac{2f}{96} \right\rceil \times t_{\text{tag}} + f \times t_{\text{s}}.$$

Let N_4 be the number of tags whose presence has not been verified before the current frame, and X_i be a random variable for the hash index of the ith slot in the frame. Since each of the N_4 tags will be randomly mapped to a slot by the first hash function, we have

$$\begin{aligned} \text{Prob}\{X_i = \text{“01”}\} &= \binom{N_4}{1} \left(\frac{1}{f}\right) \left(1 - \frac{1}{f}\right)^{N_4 - 1} \\ &= \frac{N_4}{f} \left(1 - \frac{1}{f}\right)^{N_4 - 1}. \end{aligned} \tag{17}$$

Therefore, the expected number of "01"s in the preframe vector is

$$N_4 \left(1 - \frac{1}{f}\right)^{N_4 - 1} \tag{18}$$

which is also the expected number of tags whose presence can be verified based on the first hash function. Let N_5 be the expected number of remaining unverified tags after the first hash function is applied. We have

$$N_5 = N_4 - N_4 \left(1 - \frac{1}{f}\right)^{N_4 - 1}. \tag{19}$$

Let Y_i be the random variable for the number of tags that are mapped to the ith slot of the frame by the second hash function. Since each of the N_5 tags will randomly map to a slot, we have

$$\text{Prob}\{Y_i = k\} = \binom{N_5}{k} \left(\frac{1}{f}\right)^k \left(1 - \frac{1}{f}\right)^{N_5 - k}. \tag{20}$$

According to our protocol, if the hash index of a slot is set to "01" (i.e., the slot has been assigned to a tag using the first hash function), then the slot cannot be assigned to another tag by the second hash function. Thus, we have

$$\begin{aligned} &\text{Prob}\{X_i = \text{“10”}\} \\ &\quad = \text{Prob}\{X_i \neq \text{“01”}\} \times \text{Prob}\{Y_i = 1\} \\ &\quad = \left(1 - \frac{N_4}{f} \left(1 - \frac{1}{f}\right)^{N_4 - 1}\right) \cdot \frac{N_5}{f} \left(1 - \frac{1}{f}\right)^{N_5 - 1}. \end{aligned} \tag{21}$$

If $k(>1)$ tags are mapped to a slot by H_2, the collision slot can be turned into a singleton by H_3 with probability

$$\binom{k}{1}\left(\frac{1}{2}\right)^1\left(1-\frac{1}{2}\right)^{k-1}=k\left(\frac{1}{2}\right)^k. \tag{22}$$

Therefore, the probability for X_i to be "11" is

$$\begin{aligned}&\text{Prob}\{X_i=\text{“11”}\}\\&=\text{Prob}\{X_i\neq\text{“01”}\}\times\sum_{k=2}^{N_5}\text{Prob}\{Y_i=k\}\cdot k\left(\frac{1}{2}\right)^k\\&=\left(1-\frac{N_4}{f}\left(1-\frac{1}{f}\right)^{N_4-1}\right)\\&\quad\cdot\frac{N_5}{2f}\left(\left(1-\frac{1}{2f}\right)^{N_5-1}-\left(1-\frac{1}{f}\right)^{N_5-1}\right).\end{aligned} \tag{23}$$

For simplicity, we introduce a *load factor* $\rho=\frac{N_4}{f}$, which is the ratio between the number of tags whose presence needs to be verified and the number of slots in the current frame. From (19), we have

$$\begin{aligned}\frac{N_5}{f}&=\rho-\rho\cdot\left(1-\frac{1}{f}\right)^{N_4-1}\\&\approx\rho-\rho e^{-\rho}.\end{aligned}$$

A slot is a singleton (i.e., it is assigned to one and only one tag) if its corresponding hash index in the preframe vector is not "00." Based on (17), (21), and (23), the probability for a slot to become a singleton is

$$\begin{aligned}&\text{Prob}\{X_i=\text{“01”}\}+\text{Prob}\{X_i=\text{“10”}\}+\text{Prob}\{X_i=\text{“11”}\}\\&=\frac{N_4}{f}\left(1-\frac{1}{f}\right)^{N_4-1}+\left(1-\frac{N_4}{f}\left(1-\frac{1}{f}\right)^{N_4-1}\right)\\&\quad\cdot\frac{N_5}{f}\left(1-\frac{1}{f}\right)^{N_5-1}+\left(1-\frac{N_4}{f}\left(1-\frac{1}{f}\right)^{N_4-1}\right)\\&\quad\cdot\frac{N_5}{2f}\left(\left(1-\frac{1}{2f}\right)^{N_5-1}-\left(1-\frac{1}{f}\right)^{N_5-1}\right)\\&=\frac{N_4}{f}\left(1-\frac{1}{f}\right)^{N_4-1}+\left(1-\frac{N_4}{f}\left(1-\frac{1}{f}\right)^{N_4-1}\right)\\&\quad\cdot\frac{N_5}{2f}\left(\left(1-\frac{1}{2f}\right)^{N_5-1}+\left(1-\frac{1}{f}\right)^{N_5-1}\right)\\&\approx\rho e^{-\rho}+\frac{\rho(1-e^{-\rho})(1-\rho e^{-\rho})}{2}\cdot\left(e^{-\frac{\rho(1-e^{-\rho})}{2}}+e^{-\rho(1-e^{-\rho})}\right)\\&=\rho e^{-\rho}+\frac{\delta(1-\rho e^{-\rho})}{2}\cdot\left(e^{-\frac{\delta}{2}}+e^{-\delta}\right)\end{aligned} \tag{24}$$

where $\delta=\rho(1-e^{-\rho})$.

There are f slots in the current 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 current frame. We have

$$N'\approx f\left(\rho e^{-\rho}+\frac{\delta(1-\rho e^{-\rho})}{2}\cdot\left(e^{-\frac{\delta}{2}}+e^{-\delta}\right)\right). \tag{25}$$

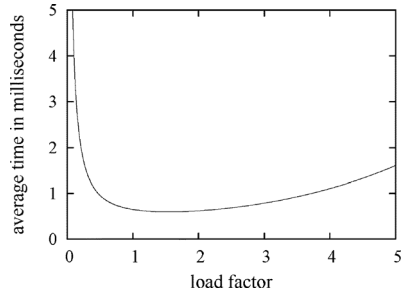


Fig. 4. Average time for verifying the presence of one tag with respect to the load factor ρ.

The average time for verifying the presence of one of those tags is

$$\begin{aligned}\frac{T_4}{N'}&\approx\frac{2\left\lceil\frac{f}{96}\right\rceil\times t_{\text{tag}}+f\times t_{\text{s}}}{f\left(\rho e^{-\rho}+\frac{\delta(1-\rho e^{-\rho})}{2}\cdot\left(e^{-\frac{\delta}{2}}+e^{-\delta}\right)\right)}\\&\approx\frac{403}{\rho e^{-\rho}+\frac{\delta(1-\rho e^{-\rho})}{2}\cdot\left(e^{-\frac{\delta}{2}}+e^{-\delta}\right)}\end{aligned} \tag{26}$$

where $t_s=321\ \mu$s and $t_{\text{tag}}=3927\ \mu$s (based on the parameters in [16]).[1] The average time spent per tag, $\frac{T_4}{N'}$, is a function of ρ. Fig. 4 shows the value of $\frac{T_4}{N'}$ with respect to ρ.

We can minimize $\frac{T_4}{N'}$ by maximizing the denominator of (26), i.e., $\rho e^{-\rho}+\frac{\rho\delta(1-\rho e^{-\rho})}{2}\cdot(e^{-\frac{\rho\delta}{2}}+e^{-\rho\delta})$. Using a numerical method such as bisection search, we obtain the optimal value $\rho=1.540$, such that the average time for verifying the presence of one tag is minimized to 599 μs. In order to achieve a load factor of 1.540, the frame size should be set to $f=N_4/1.540$.

It is important to see that $\frac{T_4}{N'}$ only depends on ρ but not N_4 (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 $\rho=1.540$ for all frames, the average time for verifying the presence of a tag becomes a constant 599 μs across all frames during the execution of THP. In this case, the execution time of the protocol is 599 $\mu\text{s}\times N$. Recall that a lower bound for the minimum execution time of any missing-tag detection protocol is $t_{\text{s}}N=321\ \mu\text{s}\times N$. Hence, THP is within a factor of 1.9 from the optimal.

F. 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 timeframe 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

[1]The rate from a tag to the reader is 53 kb/s, while the rate from the reader to tags is 26.5 kb/s; it takes 37.76 μs for the reader to transmit 1 bit. However, each tag only needs to transmit a bit to announce its presence by making the channel busy. In order to transmit this bit, it needs a time-slot, and each time-slot requires 302 μs idle time to separate consecutive slots. Therefore, due to the *information* nature in missing-tag detection, the *effective* rate from tags to the reader is low. On the contrary, the reader transmits 96 bits in each slot of t_{tag} (which also contains 302 μs idle time). The higher *effective* rate from the reader helps THP reduce execution time by reducing the overall number of t_{s} slots that tags need.

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 to avoid excessive verbosity.

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 misdetection in [15] 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-resistant multibit 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 nonempty, 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. The length of a long-response slot is about 1.5 times the length of a short-response slot. If we replace all short-response slots with long-response slots in a protocol, the execution time will be increased by about 50% due to this replacement. Certainly, we may use checksums of more than 10 bits (such as 16 bits or more) to improve the resilience against channel error.

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 III-A). Suppose two tags collide in their ID transmissions. The negative acknowledgment, 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 Section V will demonstrate through simulations.

G. Limited Frame Size

The maximum frame size in a practical RFID system is likely to be limited due to clock synchronization, power restoration, or other reasons. When we describe the protocols, frames are treated as *logical ones* whose sizes are not limited. In practice, if the size s of a (logical) frame is larger than the maximum size m, the logical frame has to be implemented as $\frac{s}{m}$ consecutive physical frames, during which tags transmit. Knowing the index of its time-slot in the logical frame, a tag can easily determine in which physical frame and which slot of the frame it should transmit.

H. Pushing Complexity to Reader

Significant asymmetry exists between readers and tags: Unlike tags, the cost for a reader is not a serious concern because it is not needed in large quantities; sometimes, only one is needed. Therefore, unlike tags, the reader is not limited in storage space, computation power, or energy supply. If necessary, it can be connected to a powerful server for resources. With a high-quality antenna, a reader is able to receive weak signals from tags. With low-quality antennas, although tags can receive strong signals from the reader, they may not receive each other's weak signals. They may not even sense whether the channel is busy or idle, i.e., whether another tag is transmitting. Nor can they sense if collision has occurred when two tags transmit simultaneously. However, the reader can detect whether the channel is idle or whether collision occurs. Such asymmetry points out a design principle that we should follow: pushing the complexity to the reader while leaving the tags simple. In our protocol design, a tag only needs to perform very simple operations: receiving a few of parameters from the reader, implementing a simple hash function based on a prestored bit ring, and transmitting in a certain slot. All the intelligent work such as tag removal or preframe vector computation is performed by the reader.

V. Simulation Results

In this section, we evaluate the efficiency of the baseline protocol, TPP, TPP/TR, TPP/CSTR, and THP by simulations. We compare our protocols to the state-of-the-art protocols in the related work. They are the Trusted Reader Protocol (TRP) [15], the Enhanced Dynamic Framed Slotted ALOHA (EDFSA) [18], and the Binary Tree Protocol (BTP) [19].

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. The maximum standard deviation in our simulations is less than 5% of the average.

The simulation setting is based on the Philips I-Code specification [16]. Any two consecutive transmissions (from the reader to tags, or vice versa) are separated by a waiting time of 302 μs. According to the specification, the transmission rate from a tag to the reader is different than the transmission rate from the reader to a tag. The rate from a tag to the reader is 53 kb/s; it takes 18.88 μs for a tag to transmit 1 bit. The value of the information transmission is calculated as the sum of a waiting time and the time for transmitting the information, which is 18.88 μs

TABLE II
EXECUTION TIME WITH RESPECT TO N

N	Execution time in seconds							
	Baseline	TPP	TPP/TR	TPP/CSTR	THP	TRP	EDFSA	BTP
5,000	21.2	10.8	7.7	4.7	3.0	31.3	34.6	35.9
10,000	42.5	21.7	15.4	9.4	6.0	62.6	66.1	70.8
20,000	85.0	43.3	30.8	18.8	12.0	125.2	135.7	140.5
30,000	127.4	65.0	46.2	28.1	18.0	187.7	205.7	212.3
40,000	169.9	86.7	61.6	37.5	24.0	250.3	269.7	283.8
50,000	212.4	108.3	77.0	46.9	30.0	312.8	342.8	357.2
75,000	318.6	162.5	115.5	70.4	44.9	408.5	510.1	533.2
100,000	424.8	216.6	154.1	93.8	59.9	543.4	683.3	709.3

multiplied by the length of the information. In this study, both short response (1 bit) and long response (10 bits) are transmitted from the tags to the reader. Therefore, t_{s} is 321 μs, and t_{l} is 491 μs.

The transmission rate from the reader to tags is 26.5 kb/s; it takes 37.76 μs for the reader to transmit one bit. Each tag ID contains 96 bits, which includes a 16-bit CRC code according to the Gen2 standard. Recall that all the ID-transmissions are from the reader to the tags in this study, which is 96 bits. Therefore, t_{tag} is 3927 μs (including a waiting time before the transmission). The time for the reader to transmit a segment of the preframe vector (hash indices for THP) is the same. However, if a tag transmits a 96-bit ID to the reader, it only takes 2114 μs due to a different transmission rate.

Our protocols except for TPP/CSTR need only to identify empty and nonempty slots. TPP/CSTR has to identify empty, singleton, and collision slots. Therefore, the tags in the baseline protocol, TPP, TPP/TR, THP, and TRP transmit short responses, each taking $t_{\mathrm{s}} = 321$ μs, and the tags in TPP/CSTR transmit long responses, each taking $t_l = 491$ μs. EDFSA and BTP require the tags to transmit their IDs, each taking 2114 μs. Unless specified otherwise, the default number of missing tags in the simulations is 1.

It is worth to mention that while we use large frame sizes in the proposed protocols, clock synchronization happens at the beginning of each protocol execution. RFID systems operate in low-rate wireless channels. The data rate is about 53 kb/s for the I-Code system, which is a thousand times slower than 802.11g. That means each bit takes a thousandfold more time to transmit, and therefore the requirement for clock synchronization is less stringent, particularly when considering that the protocols only take a few minutes to complete (Table II). Clock drift should not be a major issue in a low-rate channel within such a short period of time. When we have to set an upper bound on frame size (say, 3000 slots, which translates into less than 1 s), we can implement a larger frame size by using multiple frames, each having 3000 or less slots. In this case, the reader transmits a signal at the beginning of each frame to reset the clock. This overhead, when amortized over the large number of slots in a frame, will be negligible.

A. Time Efficiency

The first set of simulations evaluates the time efficiency of the protocols. We compare our five protocols to EDFSA and BTP for the time it takes each of them to identify all missing tags. TRP [15] 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 [18], 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 5000 to 100 000. Table II presents the execution times of the protocols. Our baseline protocol performs better than TRP, EDFSA, and BTP, cutting the execution time by about half when compared to the best result of these existing protocols. For example, when $N = 50\,000$, the time of the baseline protocol is 67.7% of the time taken by TRP, 61.3% of the time by EDFSA, and 59.1% of the time by BTP. Our other protocols, TPP, TPP/TP, TPP/CSTR, and THP, perform increasingly better. The execution time of THP is around 14.1% 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 THP is around 10.3 (11.2, 11.7). When $N = 50\,000$, TRP (EDFSA, BTP) requires 312.8 (342.8, 357.2) s while THP requires only 30.0 s, representing 90.4% (91.3%, 91.6%) reduction in the execution time.

B. 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, $\alpha = 95\%$ 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 III 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 III, 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 312.8 (86.1, 85.3) s, while our best protocol THP requires only 8.0 s, representing 97.4% (90.7%, 90.6%) reduction in the execution time. When $L = 50$, TRP (EDFSA, BTP) requires 312.8 (6.3, 6.7) s, while our best protocol THP requires only 0.5 s.

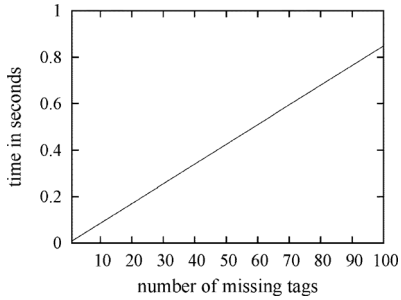


Fig. 5. Execution time of the second polling phase in TPP/CSTR with respect to the number of missing tags when $N = 50\,000$.

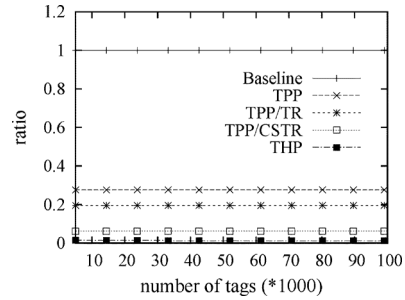


Fig. 6. Ratio between the number of ID transmissions and the number of tags N.

TABLE III
TIME TO DETECT THE FIRST MISSING TAG WITH $N = 50\,000$

L	Execution time in seconds							
	Baseline	TPP	TPP/TR	TPP/CSTR	THP	TRP	EDFSA	BTP
1	107.7	41.7	55.1	30.1	15.5	312.8	156.0	191.9
2	72.6	22.3	39.5	23.7	10.0	312.8	114.5	102.9
3	50.0	15.1	36.8	19.9	8.0	312.8	86.1	85.3
4	40.9	11.4	32.3	18.8	5.9	312.8	69.9	68.4
5	32.8	10.2	27.8	17.9	4.8	312.8	60.7	59.6
10	18.3	6.2	16.5	13.1	2.6	312.8	34.8	39.1
25	8.5	2.4	7.6	7.6	1.0	312.8	12.6	11.6
50	4.1	1.5	3.0	3.9	0.5	312.8	6.3	6.7

C. TPP/CSTR and Number of Missing Tags

Except for TPP/CSTR, the execution times of all other protocols are 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 section, we evaluate the impact of L on the execution time of TPP/CSTR. Fig. 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 only about 0.8 s, which is insignificant when compared to the 46.9 s for the other two phases of the protocol.

D. Radio Collision and Number of ID Transmissions

Section III-C proposes two design guidelines in this study. The first one is to reduce radio collision since the collision incurs retransmission in prior art. In our five protocols, only TPP and TPP/CSTR have collision slots. Moreover, the collision slots in TPP/CSTR are all 2-collision ones and can be used to verify the presence of the collided tags.

The second one is to avoid ID transmissions since it takes much longer time to transmit an ID than a short (long) response. We vary the number of tags N from 5000 to 100 000. Fig. 6 presents the ratio between the number of ID transmissions and N in the system. Recall that THP does not require any ID transmission, but it needs to transmit a long preframe vector. The reader first divides the vector into a number of segments, each having 96 bits long except for the last one, and then transmits the segments one by one. Since the segment has same number of bits as an ID. We treat the transmission of each segment in the vector as one ID transmission. The baseline protocol requires one ID transmission for each tag. Therefore, its ratio is always 1. Other four protocols—TPP, TPP/TR, TPP/CSTR, and THP—have decreasing values of ratio. When N is 50 000, the ratio of the four protocols are 27.6%, 19.5%, 6.2%, and 1.3%. The results match our expectation for efficient protocol design.

E. Impact of Channel Error

So far, we have demonstrated that the proposed protocols work fine when channel error is negligible. When channel error is not negligible, we can enhance robustness of the protocols using mechanisms described in Section IV-F. In this section, we use our best protocol, THP, to evaluate the effectiveness of such mechanisms.

In THP, there are two types of slots: 1) response slots ($t_{\rm s}$), each of which carries 1-bit information for a tag to announce its presence, and 2) hash-index slots, each of which is 96 bits long ($t_{\rm tag}$), carrying a segment of hash indices in the preframe vector. Any slot may be corrupted due to channel error. It may cause false positives[2] as we have explained in Section IV-F. We mainly use checksums to deal with this problem and use periodic protocol execution as a supplement (Section IV-F). More specifically, we modify THP as follows: 1) All response slots are replaced with long slots, each carrying a 16-bit CRC checksum that is computed based on the ID of the transmitting tag. If the reader expects a tag to respond in a slot but receives an incorrect checksum, the tag treats it as channel error and records the tag as a "missing" tag, which may however be a case of false negative.[3] 2) A 16-bit CRC checksum is included in each 96-bit slot that carries a segment of the preframe vector. If a tag finds that the segment carrying the hash index it needs is corrupted, it will not transmit in any subsequent frames (because it does not know which slot it is supposed to transmit due to corrupted hash index). Eventually, the reader will classify this tag as a "missing" one because the tag does not transmit at its assigned slot. This is again false negative. To handle false negative cases, the reader polls the detected missing tags after all frames of THP are completed. If a "missing" tag responds after being polled, the reader knows that it is a false negative case and reclassifies the tag as not missing.

The above enhancement comes with a price: The protocol execution time will be longer. We perform simulations to evaluate

[2]We refer to the confirmation of a tag's presence as positive. False positive means wrongly confirming the presence of a missing tag.

[3]We use false negative for wrongly claiming that an existing tag is missing.

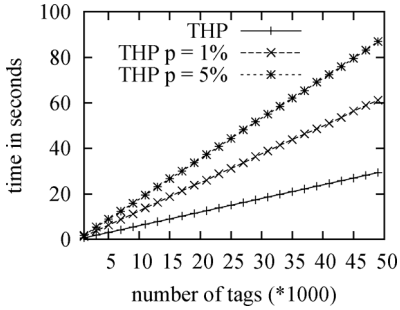


Fig. 7. Execution time of enhanced THP protocol that handles channel error, where p is the probability for any slot to be corrupted.

the significance of its impact. The results are shown in Fig. 7. If 1% of all slots are corrupted, the execution time of the enhanced THP is about twice the time of the original THP; if 5% of all slots are corrupted, the execution time of the enhanced THP increases is less than tripling the time of the original THP. Such an increase does not change the basic conclusions of this paper because, even after doubling or tripling, our execution time is still far smaller than the times of other protocols, which are mentioned in Section V, before they take channel error into account.

VI. Related Work

The tag-collection protocols mainly fall into two categories. One is *tree-based* [2], [19]–[23], and the other is *ALOHA-based* [24], [18], [17], [8], [25], [26]. The *tree-based protocols* organize all IDs in a tree of ID prefixes [19]–[21]. 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 protocol tries to balance the tree by letting the tags randomly pick to which branches they belong [19], [22], [23]. Namboodiri and Gao [27] propose a binary search tree scheme and design three anti-collision protocols to converse energy of mobile readers. Pan and Wu propose a Smart Trend-Traversal protocol [2] for RFID tag arbitration. It reduces energy cost in the arbitration process through a Query Tree-based scheme.

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 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) [18] increases the identification probability by adjusting the frame size and restricting the number of responding tags in the frame.

The tag estimation [12], [13], [6], [8], [14] is another important problem in RFID system. Kodialam and Nandagopal [12] 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 [12] 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 [13] in order to address the constraint mentioned above. Qian *et al.* [14] 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 *et al.* [15] 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.

Sheng *et al.* [28] propose a slotted ALOHA-based protocol to probabilistically detect the missing event. The basic intuition is as follows. At the beginning of a detection, the reader broadcasts a frame size and a random seed. Each tag randomly picks a slot in the frame to respond based on the random seed and its ID. Since the reader has the ID information of all tags and the seed, it can compute which slots are empty and which are not. If a slot should be occupied by a tag id_1, but it turns out to be empty, the reader knows that id_1 is missing. However, if that slot is also occupied by another tag id_2, which is not missing, the slot will still be nonempty. In this case, the missing tag id_1 cannot be detected any more. In order to solve this problem and achieve better detection accuracy, the reader repeats the detection protocol multiple rounds with different random seeds. If a tag is reported to be missing in at least one round, it is considered to be missing. This protocol has similar weaknesses as TRP. First, it cannot guarantee 100% detection accuracy. Second, the execution time will be extremely large as the detection accuracy approaches 100%.

The protocols in [15] and [28] are very simple and impose minimum hardware requirements, which is advantageous as far as RFID tags are considered. The protocols in this paper are relatively more complex. For example, in TPP/TRP, a tag needs

to be "silenced" after being polled, such that it does not transmit in the frame phase. This requires additional complexity in tag design. Nevertheless, these protocols are designed for different purposes. The protocols in [15] and [28] detect missing-tag events, whereas the ones in this paper identify the missing tags.

VII. Conclusion

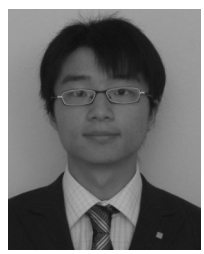
In this paper, we study the problem of monitoring the set of tags in a large RFID system and identifying the missing ones. The solution to this problem has important inventory management applications in large livestock farms, warehouses, and hospitals. To avoid interfering with other normal operations, 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.

Acknowledgment

The authors would like to thank the anonymous reviewers for their constructive comments.

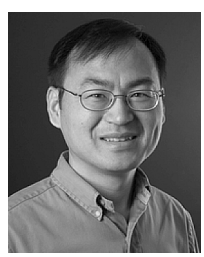
References

[1] X. Xu, L. Gu, J. Wang, and G. Xing, "Negotiate power and performance in the reality of RFID systems," in *Proc. IEEE PerCom*, 2010, pp. 88–97.
[2] L. Pan and H. Wu, "Smart trend-traversal: A low delay and energy tag arbitration protocol for large RFID systems," in *Proc. IEEE INFOCOM*, 2009, pp. 2571–2575.
[3] C. Wang, H. Wu, and N. Tzeng, "RFID-based 3-D positioning schemes," in *Proc. IEEE INFOCOM*, 2007, pp. 1235–1243.
[4] Z. Yang and H. Wu, "Featherlight information network with delay-endurable RFID support (FINDERS)," in *Proc. IEEE SECON*, 2009, pp. 1–9.
[5] N. Bombourg, "Global RFID market forecast to 2014," *PR Newswire* May 2012 [Online]. Available: http://www.prnewswire.com/news-releases/global-rfid-market-forecast-to-2014-151513495.html
[6] J. Zhai and G. N. Wang, "An anti-collision algorithm using two-functioned estimation for RFID tags," in *Proc. ICCSA*, 2005, pp. 702–711.
[7] J. Cha and J. Kim, "Novel anti-collision algorithms for fast object identification in RFID system," in *Proc. IEEE ICPADS*, 2005, pp. 63–67.
[8] H. Vogt, "Efficient object identification with passive RFID tags," in *Proc. IEEE PerCom*, 2002, pp. 98–111.
[9] D. Hush and C. Wood, "Analysis of tree algorithm for RFID arbitration," in *Proc. IEEE ISIT*, 1998, p. 107.
[10] J. Myung and W. Lee, "An adaptive memoryless tag anti-collision protocol for RFID networks," in *Proc. IEEE ICC*, 2005, pp. 10–12.
[11] H. Choi, J. Cha, and J. Kim, "Fast wireless anti-collision algorithm in ubiquitous ID system," in *Proc. IEEE VTC*, 2004, vol. 6, pp. 4589–4592.
[12] M. Kodialam and T. Nandagopal, "Fast and reliable estimation schemes in RFID systems," in *Proc. ACM MobiCom*, 2006, pp. 322–333.
[13] M. Kodialam, T. Nandagopal, and W. Lau, "Anonymous tracking using RFID tags," in *Proc. IEEE INFOCOM*, 2007, pp. 1217–1225.
[14] C. Qian, H. Ngan, and Y. Liu, "Cardinality estimation for large-scale RFID systems," in *Proc. IEEE PerCom*, 2008, pp. 30–39.
[15] C. C. Tan, B. Sheng, and Q. Li, "How to monitor for missing RFID tags," in *Proc. IEEE ICDCS*, 2008, pp. 295–302.
[16] Philips Semiconductors, Eindhoven, The Netherlands, "I-CODE smart label RFID tags," Jan. 2004 [Online]. Available: http://www.nxp.com/acrobat_download/other/identification/SL092030.pdf
[17] J. R. Cha and J. H. Kim, "Dynamic framed slotted ALOHA algorithms using fast tag estimation method for RFID systems," in *Proc. IEEE CCNC*, 2006, vol. 2, pp. 768–772.
[18] S. Lee, S. Joo, and C. Lee, "An enhanced dynamic framed slotted ALOHA algorithm for RFID tag identification," in *Proc. IEEE MOBIQUITOUS*, 2005, pp. 166–172.
[19] J. Myung and W. Lee, "Adaptive splitting protocols for RFID tag collision arbitration," in *Proc. ACM MobiHoc*, 2006, pp. 202–213.
[20] N. Bhandari, A. Sahoo, and S. Iyer, "Intelligent query tree (IQT) protocol to improve RFID tag read efficiency," in *Proc. IEEE ICIT*, 2006, pp. 46–51.
[21] 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," in *Proc. ISLPED*, 2004, pp. 357–362.
[22] *Information Technology Automatic Identification and Data Capture Techniques C Radio Frequency Identification for Item Management Air Interface—Part 6: Parameters for Air Interface Communications at 860–960 MHz*, Final Draft International Standard ISO 18000-6, Nov. 2003.
[23] J. I. Capetenakis, "Tree algorithms for packet broadcast channels," *IEEE Trans. Inf. Theory*, vol. IT-25, no. 5, pp. 505–515, Sep. 1979.
[24] I. E. Teleta and R. G. Gallager, "Combining queuing theory with information theory for multiaccess," *IEEE J. Sel. Areas Commun.*, vol. 13, no. 6, pp. 963–969, Aug. 1995.
[25] V. Sarangan, M. R. Devarapalli, and S. Radhakrishnan, "A framework for fast RFID tag reading in static and mobile environments," *Int. J. Comput. Telecommun. Netw.*, vol. 52, no. 5, pp. 1058–1073, 2008.
[26] B. Zhen, M. Kobayashi, and M. Shimizu, "Framed ALOHA for multiple RFID objects identification," *IEICE Trans. Commun.*, vol. E88-B, no. 3, pp. 991–999, 2005.
[27] V. Namboodiri and L. Gao, "Energy-aware tag anti-collision protocols for RFID systems," in *Proc. IEEE PerCom*, 2007, pp. 23–36.
[28] B. Sheng, Q. Li, and W. Mao, "Efficient continuous scanning in RFID systems," in *Proc. IEEE INFOCOM*, 2010, pp. 1–9.



Tao Li received the B.S. degree from the University of Science and Technology of China, Hefei, China, in 2007, and the Ph.D. degree from the University of Florida, Gainesville, FL, USA, in 2012, both in computer science.

He is working with Google, Inc., Mountain View, CA, USA, as an Engineer. His research interests include network security and sensor networks.



Shigang Chen (M'02–SM'12) received the B.S. degree from the University of Science and Technology of China, Hefei, China, in 1993, and the M.S. and Ph.D. degrees from the University of Illinois at Urbana–Champaign, Urbana, IL, USA, in 1996 and 1999, respectively, all in computer science.

He is an Associate Professor with the Department of Computer and Information Science and Engineering, University of Florida, Gainesville, FL, USA. After graduation, he had worked with Cisco Systems, San Jose, CA, for three years before joining the University of Florida in 2002. He served on the Technical Advisory Board for Protego Networks from 2002 to 2003. He published more than 100 peer-reviewed journal/conference papers. He holds 11 US patents. His research interests include computer networks, Internet security, wireless communications, and distributed computing.

Dr. Chen is an Associate Editor for the IEEE/ACM Transactions on Networking, *Computer Networks*, and the IEEE Transactions on Vehicular Technology. He has been serving on the Steering Committee of IEEE IWQoS since 2010. He received the IEEE Communications Society Best Tutorial Paper Award in 1999 and the NSF CAREER Award in 2007.

Yibei Ling (M'00–SM'06) received the B.S. degree in electrical engineering from Zhejiang University, Hangzhou, China, in 1982, the M.S. degree in statistics from Shanghai Medical University (currently Fuda University), Shanghai, China, in 1988, and the Ph.D. degree in computer science from Florida State University at Miami, Miami, FL, USA, in 1995.

He is a Senior Research Scientist with the Applied Research Laboratories, Telcordia Technologies (formerly Bellcore), Morristown, NJ, USA. His research interests include distributed computing, query optimization in database management system, scheduling, biological moduling, checkpointing, system performance/evaluation, fault localization/self-healing in mobile ad hoc networks, and power-aware routing in mobile ad hoc networks. He is the Architect, as well as the Developer, of the voice subsystem of Telcordia Notification System. He is a key team member of the Telcordia-led DARPA Adaptive Cognitive-Enhanced Radio Team Project, in which he is responsible for designing/implementing the distributed positioning subsystem using GPS and ultrawideband technologies. He is the Architect, as well as the Developer, of the Telcordia flagship IP Assure System. He is an Invited Reviewer for *Mathematical Reviews*. He has published several papers in the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, the IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING, and the *Transactions on Sensor Networks*. He also has published several papers in prestigious conferences such as the Special Interest Group on Management of Data, the International Conference on Data Engineering, the International Conference on Distributed Computing Systems, the International Conference on Principles of Distributed Computing, and IEEE INFOCOM.

Dr. Ling. served as a TPC member for QShine in 2005–2007 and for the Computer and Network Security Symposium of the IEEE International Wireless Communication and Mobile Computing Conference in 2006.