Missing-Tag Detection and Energy–Time Tradeoff in Large-Scale RFID Systems With Unreliable Channels

Wen Luo, Shigang Chen, *Senior Member, IEEE*, Yan Qiao, and Tao Li

***Abstract*—Radio frequency identification (RFID) technologies are poised to revolutionize retail, warehouse, and supply chain management. One of their interesting applications is to automatically detect missing tags in a large storage space, which may have to be performed frequently to catch any missing event such as theft in time. Because RFID systems typically work under low-rate channels, past research has focused on reducing execution time of a detection protocol to prevent excessively long protocol execution from interfering normal inventory operations. However, when active tags are used for a large spatial coverage, energy efficiency becomes critical in prolonging the lifetime of these battery-powered tags. Furthermore, much of the existing literature assumes that the channel between a reader and tags is reliable, which is not always true in reality because of noise/interference in the environment. Given these concerns, this paper makes three contributions. First, we propose a novel protocol design that considers both energy efficiency and time efficiency. It achieves multifold reduction in both energy cost and execution time when compared to the best existing work. Second, we reveal a fundamental energy–time tradeoff in missing-tag detection, which can be flexibly controlled through a couple of system parameters in order to achieve desirable performance. Third, we extend our protocol design to consider channel error under two different models. We find that energy/time cost will be higher in unreliable channel conditions, but the energy–time tradeoff relation persists.**

***Index Terms*—Energy-efficient, missing tag detection, radio frequency identification (RFID), time-efficient.**

I. Introduction

RADIO frequency identification (RFID) technologies [1]–[12] are poised to revolutionize retail, warehouse, and supply chain management. One of the interesting applications is to detect missing items in a large storage. Consider a major warehouse that keeps thousands of apparel, shoes, pallets, cases, appliances, electronics, etc. How can one find out if anything is missing? We may have someone walk through the warehouse and count items. This is not only laborious, but also error-prone, considering that clothes may be stacked together, goods on racks may need a ladder to access, and they may be blocked behind columns. If we attach an RFID tag to each item,[1] the whole detection process can be automated with one or multiple RFID readers communicating with tags to find out whether any tags (and their associated objects) are absent.

There are two different missing-tag detection problems: *exact detection* and *probabilistic detection*. The objective of exact detection is to identify exactly *which tags are missing*. The objective of probabilistic detection is to detect *a missing-tag event* with a certain predefined probability. An exact detection protocol [13]–[15] gives much stronger results, but its overhead is far greater than a probabilistic detection protocol [5], [16]–[18]. Hence, they both have their values. In fact, they are complementary to each other and should be used together. For example, a probabilistic detection protocol may be scheduled to execute frequently, e.g., once every minute, in order to timely catch any loss event such as theft. Once it detects some tags are missing, it may invoke an exact detection protocol to pinpoint which tags are missing. If one execution of a probabilistic detection protocol detects a missing-tag event with 99% probability, five independent executions will detect the event with 99.99999999% probability. If that is not enough, we may schedule an exact detection protocol every five times the probabilistic detection protocol is executed.

This paper focuses on probabilistic detection. Because it is performed frequently, its performance becomes very important. Suppose a missing-tag detection protocol is scheduled to execute once every few minutes in a warehouse. If the execution time of the protocol is a minute, any normal operations that move goods out have a good chance to trigger a false alarm. To reduce the chance of interfering with normal operations, we want to make the protocol's execution time as small as possible. Another performance requirement is to minimize the protocol's energy cost. To cover a large area, battery-powered active tags are preferred. In order to prolong their lifetime, we need to make any periodically executed protocol as energy-efficient as possible, particularly if one is scheduled to execute once every few minutes for 24 h a day, day by day.

Despite its importance, the problem of probabilistic missing-tag detection is relatively new and underinvestigated. The basic detection method is introduced in the pioneer work [5]: An RFID reader monitors a time frame of f slots. Through a hash function, each tag pseudo-randomly selects a slot in the time frame to transmit. The reader can predict in which slot each known tag will transmit. It detects a missing-tag event if no tag transmits during a slot when there is supposed to be tag(s) transmitting. However, multiple tags may select the same slot to transmit. If a tag is missing, its slot may be

Manuscript received June 24, 2012; revised February 05, 2013 and May 29, 2013; accepted May 30, 2013; approved by IEEE/ACM Transactions on Networking Editor G. Bianchi. Date of publication July 22, 2013; date of current version August 14, 2014. A preliminary version of this paper appeared in the Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), Hilton Head Island, SC, USA, June 11–14, 2012.

W. Luo, S. Chen, and Y. Qiao are with the Department of Computer and Information Science and Engineering, University of Florida, Gainesville, FL 32611 USA (e-mail: wluo@cise.ufl.edu; sgchen@cise.ufl.edu; yqiao@cise.ufl.edu).

T. Li is with Google, Inc., Mountain View, CA 94041 USA (e-mail: lita@google.com).

Digital Object Identifier 10.1109/TNET.2013.2270444

[1]A tag may be attached in a way that ruins the product if it is detached inappropriately, such as releasing ink onto clothing.

kept busy by transmission from another tag. Consequently, the reader cannot guarantee the detection of a missing-tag event. The protocol in [5] only considers time efficiency, but not energy efficiency. A follow-up work [16] further improves the time efficiency. Firner *et al.* [17] design a simple communication protocol, Uni-HB, to detect missing items for fail-safe presence assurance systems and demonstrate it can lead to longer system lifetime and higher communication reliability than several popular protocols. The protocol, however, does not consider time efficiency and requires all tags to participate and transmit, which will be less efficient than a sampling-based protocol design that requires only a small fraction of the tags to participate. Similarly, the method in [18] also requires all tags to participate.

In addition, we observe that much of the existing literature assumes that the communication channel between an RFID reader and tags is reliable, which means that information transmitted is never corrupted. However, in reality, errors may occur due to low signal strength and noise interference in the operating environment. The occurrence of errors usually follows a certain distribution, which is characterized by an error model, describing the statistical properties of underlying error sequences.

In this paper, we make three contributions. First, we propose a new, more sophisticated protocol design for missing-tag detection. It takes both energy efficiency and time efficiency into consideration. By introducing multiple hash seeds, our new design provides multiple degrees of freedom for tags to choose in which slots they will transmit. This design drastically reduces the chance of collision, and consequently achieves multiple-fold reduction in both energy cost and execution time. In some cases, the reduction is more than an order of magnitude. Second, with the new design, we reveal a fundamental energy–time tradeoff in missing-tag detection. Our analysis shows that better energy efficiency can be achieved at the expense of longer execution time, and vice versa. The performance tradeoff can be easily controlled by a couple of system parameters. Through our analytical framework for energy–time tradeoff, we are able to compute the optimal parameter settings that achieve the smallest protocol execution time or the smallest energy cost. The framework also enables us to solve the energy-constrained least-time problem and the time-constrained least-energy problem in missing-tag detection. Third, we extend our protocol design to consider channel error under two different error models. Our protocol can be configured to work under these error conditions.

The rest of the paper is organized as follows. Section II gives the system model and problem definition, as well as the prior art. Section III proposes a new missing-tag detection protocol. Section IV investigates energy–time tradeoff in protocol configuration. Section V extends the protocol under two different error models. Section VI evaluates the protocol through simulations. Section VII draws the conclusion.

II. Preliminaries

A. System Model

There are three types of RFID tags. *Passive tags* are most widely deployed today. They are cheap, but do not have internal power sources. Passive tags rely on radio waves emitted from an RFID reader to power their circuit and transmit information back to the reader through backscattering. They have short operational ranges, typically a few meters in an indoor environment, which seriously limits their applicability. *Semi-passive tags* carry batteries to power their circuit, but still rely on backscattering to transmit information. *Active tags* use their own battery power to transmit, and consequently do not need any energy supply from the reader. Active tags operate at a much longer distance, making them particularly suitable for applications that cover a large area, where one or a few RFID readers are installed to access all tagged objects and perform management functions automatically. With richer on-board resources, active tags are likely to gain more popularity in the future, when their prices drop over time as manufacture technologies are improved and markets are expanded. They are particularly attractive for high-valued objects such as luxury bags, laptops, cell phones, TVs, etc., or when the tags are reused over and over again.

Communication between a reader and tags is time-slotted. The reader's signal synchronizes the clocks of tags. There are different types of time-slots [13], among which two types are of interest in this paper. The first type is called a *tag-ID slot*, whose length is denoted as T_{tag}, during which a reader is able to broadcast a tag ID. The second type is called a *short-response slot*, whose length is denoted as T_{short}, during which a tag is able to transmit one-bit information to the reader, for instance, announcing its presence.

In this paper, we consider active tags. Besides communicating with a reader, we assume the tags have the following capability: performing a hash function, carrying a small internal storage to keep a few parameters and a 96-bit seed-selection segment from the reader, being able to check the values in the segment, and having a clock that enables a tag to transmit at a specific slot of a time frame or wait up at a prescheduled time.

B. Missing-Tag Detection Problem

The problem is to design an efficient protocol for an RFID reader to detect whether some tags are missing, subject to a *detection requirement*: A single execution of the protocol should detect a missing-tag event with probability α if m or more tags are missing, where α and m are two system parameters. For example, consider a big shoe store that carries tens of thousands of shoes, and we may set the parameters to be $\alpha = 99\%$ and $m = 10$, so that one execution of the protocol will detect any event of missing 10 or more shoes with 99% probability. If we perform independent executions of the protocol periodically, the detection probability of any missing event will approach to 100%, no matter what the values of α and m are. Furthermore, as we have explained in the Introduction, a low-overhead probabilistic detection protocol may be used in conjunction with a high-overhead exact detection protocol (which is scheduled much less frequently) to catch any miss.

We assume that the RFID reader has access to a database that stores the IDs of all tags. This assumption is necessary [5]. Without any prior knowledge of a tag's existence, how can we know that it is missing? The 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 an ID-collection protocol [19]–[23] that reads the IDs

TABLE I
NOTATIONS

n	number of RFID tags in the system
m	number of missing tags
α	probability of detecting the missing-tag event
f	number of slots in a time frame
r	random number
T_{tag}	time for transmitting a tag ID
T_{short}	time for transmitting one-bit information
k	number of hash seeds
P_{msmd}	probability for MSMD to detect a missing-tag event
P_{emd}	probability for EMD to detect a missing-tag event
p	sampling probability in MSMD or EMD
$f^*(p)$	frame size that minimizes the execution time under a given sampling probability p
f_{opt}	optimal frame size that achieves the smallest execution time, $f_{opt} = \min_p\{f^*(p)\}$
p_t	sampling probability under which f_{opt} is achieved, i.e., $f_{opt} = f^*(p_t)$
p_{opt}	optimal sampling probability that minimizes the energy cost

from the tags. In this case, we will not detect missing-tag events that have already happened. However, once we have the IDs of the remaining tags, we can detect the missing-tag events after this point of time.

Notations (most of which are introduced later) are summarized in Table I for quick reference.

C. Performance Metrics

We consider two performance metrics, *execution time* of the protocol and *energy cost* to the tags. First, RFID systems use low-rate communication channels. Low rates, coupled with a large number of tags, often take RFID protocols long time to finish their operations. Hence, in order to apply such protocols in a busy warehouse environment, it is desirable to adopt novel designs to reduce execution time as much as possible.

Second, active tags carry limited battery power. Replacing tags is a tedious, manual operation. One way of saving energy is to minimize the number of tags that are needed to participate in each protocol execution. When a tag participates in a protocol execution, it has to power its circuit during the execution, receive request information from the reader, and transmit back. When a tag does not participate, it goes into the sleep mode and incurs insignificant energy expenditure.

The energy cost to the RFID reader is less of a concern because the reader's battery can be easily replaced or it may be powered by an external source.

D. Clock Synchronization

For any missing-tag detection protocol that is scheduled to execute at fixed time intervals, there is a need to synchronize the clocks of the tags so that they can wake up at the right moments. To achieve very low energy consumption during sleep, these active RFID tags may use low-power RC oscillators as clock sources, which however have relatively large drift. The drift will become significant if the clock is left unsynchronized for an extended period of time. Following the argument in the Introduction, we expect the missing-tag detection protocols to run frequently. One solution to deal with the drift problem is to calibrate all tags' clocks at the beginning of each scheduled protocol execution in order to keep them synchronized. The tags that are not supposed to participate in a round of execution will go back to sleep after clock synchronization. It will add a fixed amount of energy expenditure to all missing-tag detection protocols that require the reader to pull information from tags at fixed time intervals. Because synchronization overhead is common to all such protocols, we will not include it in performance comparison. Also note that this overhead is relatively small when compared to the energy cost needed to power a tag for receiving, transmitting, and computing in the entire duration of a protocol execution. Another solution is only letting the participating tags wake up, but they need to wake up a little earlier than scheduled to compensate for the clock drift, such that they can receive the requests from the reader. This approach also incurs additional energy overhead because tags have to be powered a little longer for receiving.

The energy overhead for clock synchronization does not exist for a push-based missing-item detection protocol such as Uni-HB [17], where every sensor (attached to an item) transmits its ID and a sequence number to a base station in each epoch. In order to lower the collision probability, Uni-HB spreads sensor transmissions in each epoch, which means the protocol execution continues over each epoch since the IDs may be sent by the sensors at any time. In the Introduction, however, we have argued for short protocol execution time to avoid interference from busy warehouse operations that move items in and out. Furthermore, Uni-HB requires all tags to participate by sending their IDs to the base station in each epoch, whereas we prefer an approach that involves only a fraction of tags to save energy. For these reasons, Uni-HB is not suitable for meeting the requirements in this paper.

E. Prior Work

We first describe the Trusted Reader Protocol (TRP) by Tan *et al.* [5]. Given a time frame of f slots, the RFID reader maps each tag to a slot in the frame by hashing its ID and a random number r. After the reader maps all tags to the slots, it classifies slots into three categories. A slot is said to be *empty* if no tag is mapped to the slot. It is called a *singleton slot* if exactly one tag is mapped to the slot. It is a *collision slot* if more than one tag is mapped to the slot. Because the reader knows the IDs of all tags, it knows which tags are mapped to which slots. It knows exactly which slots are empty, which are singletons, and which are collision slots.

To initiate the execution of the protocol, an RFID reader broadcasts a detection request, asking the tags to respond in a time frame of f slots. The detection request has two parameters, the frame size f and the random number r. After receiving the request, each tag maps itself to a slot in the frame through the same hash function. It then transmits during that slot.

Listening to the channel, the reader records the state of each slot, which is either *busy* when one or more tags transmit or *idle* when no tag transmits. This is binary information where each slot carries either "1" or "0." When a tag transmits, it does not have to send any particular information. It only needs to make the channel busy. When no tag is missing, the reader expects all singleton and collision slots are busy. However, if the reader finds an expected busy slot to be actually idle, it knows that the tag(s) that is mapped to this slot must be missing.

TRP is designed to minimize execution time by using the smallest frame size that ensures a detection probability α if m or more tags are missing. Certainly, if fewer tags are missing, the detection probability will be lower. A follow-up work [14] essentially executes TRP iteratively to identify which tags are missing.

A serious limitation of TRP is that it only considers time efficiency. It is not energy-efficient because all tags must be active and transmit during the time frame. Firner *et al.* [17] consider energy cost, but their protocol requires all tags to participate and transmit, which will be less efficient than a sampling-based solution where only a small fraction of tags participate.

The efficient missing-tag detection protocol (EMD) [16] is similar to TRP except that each tag is sampled with a probability p for participation in each protocol execution. Only a sampled tag will select a slot to transmit. Simulations show that EMD performs better than TRP. However, the paper does not give a way to determine the optimal sampling probability.

In this paper, we show TRP and EMD are special cases of a much broader protocol design space. Not only are there protocol configurations that perform much better than TRP and EMD in terms of both time and energy efficiencies, but we also reveal a fundamental energy–time tradeoff in this design space, which allows us to adapt protocol performance to suit various needs in practical systems.

III. Multiple-Seed Missing-Tag Detection Protocol (MSMD)

In this section, we begin our protocol design by assuming a reliable channel. We will then expand the new protocol to work under different error models in the next section.

A. Motivation

Both TRP [5] and EMD [16] map tags to time-slots using a hash function. We derive the probability θ that an arbitrary slot t will become a singleton, which happens when only one tag is sampled and mapped to slot t while all other tags are either not sampled or mapped to other slots. The probability for any given tag to be sampled and mapped to t is $\frac{p}{f}$, where f is the number of slots and p is the sampling probability, which is 100% for TRP. The probability for all other tags to be either not sampled or not mapped to slot t is $(1-\frac{p}{f})^{n-1}$, where n is the number of tags. Hence, we have

$$\theta = n\frac{p}{f}\left(1-\frac{p}{f}\right)^{n-1} \approx \frac{np}{f}e^{-\frac{np}{f}} \leq \frac{1}{e} \approx 36.8\%$$

where $\frac{np}{f}e^{-\frac{np}{f}}$ reaches its maximum value when $np = f$. This upper bound for θ is true for both TRP and EMD.

Singletons are important in missing-tag detection. If a missing tag is sampled and mapped to a singleton slot, since no other tag is mapped the same slot, this expected singleton slot will turn out to be idle, which is observed by the reader, resulting in missing-tag detection.

The problem is that the majority of all slots, 63.2% or more of them, are either empty slots or collision slots. They are mostly wasted. Obviously, empty slots do not contribute anything in missing-tag detection. If a collision slot only has missing tags, detection will be successfully made because the reader will find this expected busy slot to be actually idle. However, when the number of missing tags is small when compared to the total number of tags, the chance for a collision slot to have only missing tags is also small.

Naturally, we want a protocol design that ensures a large value of θ, much larger than 36.8%, because more singleton slots increase detection power. However, the value of θ in TRP is in fact much smaller than 36.8% because TRP minimizes its execution time by using as few time-slots as possible, which results in a large percentage of collision slots. The detection probability of TRP is about $1-(1-\theta)^m$ because each of the m missing tags has a probability of θ to map to a singleton slot and thus be detected.[2] As an example, if the requirement is to detect a missing-tag event with 99% probability when 100 tags are missing, TRP will reduce its frame size to such a level that $\theta = 4.5\%$, just enough to ensure 99% detection probability.

This leaves a great room for improvement. We show that a new protocol design, different from that of TRP and EMD, can reduce the frame size to a level that is much smaller than they can do, yet keep θ at a value much greater than 36.8%. Our design, called *Multiple-Seed Missing-tag Detection protocol* (MSMD), turns most empty/collision slots into singletons. There is a compound effect of such a new design when it is coupled with sampling: Suppose $\alpha = 99\%$ and $m = 100$, same as in the previous paragraph. Under sampling, the detection probability is $1-(1-p\theta)^m$ because each of the m missing tags has a probability of $p\theta$ to be sampled and mapped to a singleton slot. If our protocol design can improve θ to 90%, we will be able to set $p = 5\%$. With such a sampling probability, we achieve much better energy efficiency because only 5% of all tags participate in each protocol execution. We also achieve far better time efficiency because, with much fewer tags transmitting, the chance of collision is reduced and a fewer number of time-slots is needed to ensure a certain level of singletons.

B. Basic Idea

We have known that under a random mapping from tags to slots, an arbitrary slot only has a probability of up to 36.8% to be a singleton. Now, if we separately apply two independent random mappings from tags to slots, a slot will have a probability of up to $1-(1-36.8\%)^2 \approx 60.1\%$ to be a singleton in one of the two mappings. If we separately apply k independent mappings from tags to slots, it has a probability of $1-(1-36.8\%)^k$ to be a singleton in one of the k mappings. The value of $1-(1-36.8\%)^k$ quickly approaches to 100% as we increase k.

It is easy to generate multiple mappings. In the detection request, the RFID reader can broadcast k seeds, $s_1, s_2, \ldots, s_k$, to tags. Each seed s_i corresponds to a different mapping, where a tag is mapped to a slot indexed by $H(id, s_i)$, which is a hash function such as [13] that takes an ID and a seed to produce an output (belonging to a required range through modulo operation).

A slot may be a singleton under one mapping, but a collision slot under other mappings. Different slots may be singletons under different mappings. To maximize the number of singletons, the reader—with the knowledge of all tag IDs and all seeds—selects a mapping (i.e., a seed) for each slot, such that the slot can be a singleton. The reader also makes sure that each

[2]To quickly get to the point without dealing with too much detail, we ignore the small contribution of collision slots in detection.

tag is assigned to a singleton only once. From each slot's point of view, a *specific* seed is used to map tags to it. From the whole system's point of view, multiple seeds are used to map different tags to different slots.

In our protocol, the reader determines system parameters, including the sampling probability p and the frame size f. After selecting k random seeds, the reader chooses a seed for each slot and constructs a *seed-selection vector* V (or selection vector for short), which contains f *selectors*, one for each slot in the time frame. Each selector z has a range of $[0, k]$. If $z > 0$, it means that the zth seed, i.e., s_z, should be used for its corresponding slot. If $z = 0$, it means that the slot is not a singleton under any seed. Finally, the reader broadcasts the selection vector to the tags. Based on the selectors, each tag determines which slot it should use to respond.

We will address the problems of how to choose the optimal system parameters, p and f, and how the number k of seeds will affect the protocol performance in Section IV. Before we describe the operations of the protocol, we introduce the concept of segmentation. In our design, the above idea is actually applied segment by segment.

C. Segmentation

The seed-selection vector has f selectors, each of which are $\lceil \log_2(k+1) \rceil$ bits long. f may be too large for the whole vector to fit in a single slot. For example, if $k = 7$, each selector is 3 bits long. If we use the same slot T_{tag} for carrying a 96-bit ID to carry the selection vector, it can only accommodate 32 selectors. When f is more than that, we have to divide the selection vector into 96-bit segments, so that they can fit in T_{tag} slots. Each segment contains $l = \frac{96}{\lceil \log_2(k+1) \rceil}$ selectors. The total number of seed-selection segments are $\frac{f}{l}$, and the jth segment is denoted as V_j.

Since we divide the selection vector into segments, we also divide the time frame into subframes, each containing l slots accordingly. The jth time subframe is denoted as F_j. This allows our protocol to deal with one subframe at a time.

D. Protocol Overview

Our protocol consists of two phases. Phase One performs tag assignment, where the reader identifies the set of sampled tags and assigns the sample tags to the subframes uniformly at random. The subset of sampled tags that are assigned to the jth subframe is denoted as N_j. For each subframe F_j, the reader selects a seed for each of its slots, constructs the seed-select segment V_j, and maps the tags in N_j to slots in F_j using the selected seeds.

Phase Two performs missing-tag detection. The reader broadcasts the seed-selection segments one after another, each in a slot of T_{tag}. Each seed-selection segment is followed by a time subframe of l slots, each of which is T_{short} long. The tags in N_j will respond in these slots. Each tag only needs to be active during its subframe, which conserves energy. The exchange between the reader and tags in Phase Two is illustrated in Fig. 1.

E. Phase One: Tag Assignment

Phase One consists of three steps, which are explained below. An illustrative example can be found in Fig. 2.

1) Determining Sampled Tags: The reader starts Phase One by uniquely identifying the set of participating tags through sampling. To implement the sampling probability p, the reader broadcasts an integer $x = \lceil pX \rceil$ and a prime number q, where X is a large, preconfigured constant (e.g., 2^{16}). During the ith round of protocol execution, a tag is sampled if and only if the hash result $H(id, qi)$, which is a pseudo-random number in the range of $[0, X)$, is smaller than x, where id is the tag's ID.

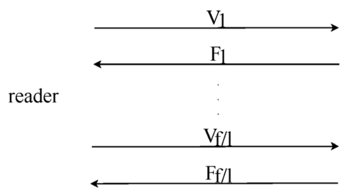


Fig. 1. In Phase Two, the reader broadcasts the seed-selection segments, V_1 through $V_{f/l}$, one at a time. Each segment V_i is immediately followed by a subframe F_i of l slots, during which the tags transmit.

After receiving x and q, each tag can predict in which rounds of protocol execution it will participate. Since the protocol is scheduled to execute periodically with predefined intervals, each tag knows when it should wake to participate. The reader, with the knowledge of all tag information, can predict which tags are sampled for each protocol execution.

2) Assigning Sampled Tags to Subframes: When assigning sampled tags to time subframes, the reader selects an additional random seed s, which is different from $s_1, \ldots, s_k$. For each sampled tag, the reader produces a hash output $H(id, s)$ and assigns the tag to the $H(id, s)$th subframe, where id is the tag's ID and the range of $H(id, s)$ is $[0, \frac{f}{l})$. Note that each tag will know to which subframe it is assigned, after it receives s in the detection request broadcast by the reader at the beginning of Phase Two.

3) Determining Seed-Selection Segments: Each seed-selection segment is determined independently. All selectors in V_j are initialized to zeros as shown in Fig. 2(b). The reader begins by using the first seed s_1 to map tags in N_j to slots in F_j, as shown in Fig. 2(c). For each tag in N_j, the reader produces a hash output $H(id, s_1)$ and maps the tag to the $H(id, s_1)$th slot in F_j, where id is the tag's ID and the range of $H(id, s_1)$ is $[0, l)$. After mapping, the reader finds singleton slots. Each singleton has one and only one tag mapped to it—as an example, the first and third slots in Fig. 2(c). We assign the tag to the slot so that it will transmit in the slot, free of collision, during Phase Two. The reader sets the corresponding selector in V_j to be 1, meaning that the first seed s_1 should be used for this slot. The slot is now called a *used* slot, and the sole tag mapped to it will be called an *assigned* tag.

The reader repeats the above process with other seeds, one at a time, for the remaining mappings. For each mapping, we only consider the slots whose selectors have not been determined yet and only consider the tags that have not been assigned to any slots yet, as shown by Fig. 2(d). In other words, the used slots and the assigned tags will not be considered. For a singleton slot that is found using seed s_i, the corresponding selector in V_j will be set to be i.

After all k mappings, if the value of a selector in V_j remains zero, it means that the corresponding slot in F_j is not a singleton under any seed. As a final attempt to utilize these unused slots, if there exist unassigned tags in N_j, the reader randomly assigns the unassigned tags to unused slots. More specifically, it chooses an additional random seed s' and produces a hash

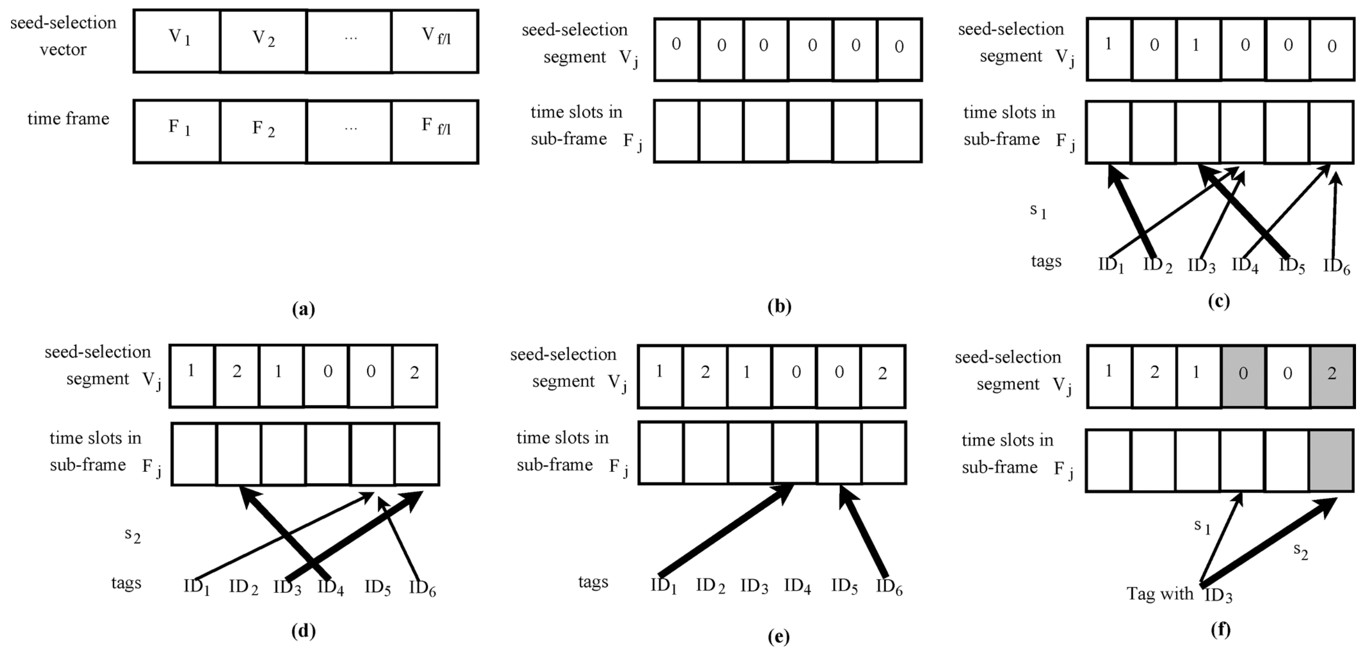

seed-selection vector
time frame
V_1 V_2 ... $V_{f/l}$
F_1 F_2 ... $F_{f/l}$
(a)
seed-selection segment V_j
time slots in sub-frame F_j
0 0 0 0 0 0
(b)
1 0 1 0 0 0
s_1
tags ID_1 ID_2 ID_3 ID_4 ID_5 ID_6
(c)
1 2 1 0 0 2
s_2
(d)
(e)
Tag with ID_3
(f)

Fig. 2. Arrows represent the mapping from tags to slots based on hash functions. Among them, thick arrows represent the assignment of tags to slots. In this example, $k = 2$. (a) Segmentation of time frame and seed-selection vector. We consider the tags that are assigned to a subframe F_j. (b) Phase One: initial status of seed-selection vector segment. (c) Phase One: Tags are mapped to slots using the first hash seed s_1. Two tags, ID_2 and ID_5, are assigned to the first slot and the third slot in F_j, respectively. The first and third selectors in the seed-selection segment are thus set to 1. (d) Phase One: Remaining tags are mapped to slots using the second seed s_2. Two more tags, ID_3 and ID_4, are assigned. The corresponding selectors in the seed-selection segment are set to "2." (e) Phase One: In the final attempt, the reader randomly assigns the remaining unassigned tags to the unused slots. (f) Phase 2: After receiving the seed-selection segment, the tag with ID_3 determines to which slot it is assigned. As it is mapped to the sixth slot by the second hash seed and the corresponding selector is also 2, the tag knows that it must be assigned to the sixth slot.

output $H(id, s')$ to assign each tag that is not assigned yet to the $H(id, s')$th unused slot, where id is the tag's ID. In case that only one tag is assigned to an unused slot, we will have an extra singleton, as shown in Fig. 2(e). Since the whole tag-to-slot assignment is pseudo-random, the reader knows which unused slots will become singletons. As we will see later in Phase Two, after receiving $s_1, \ldots, s_k$, each tag will know whether it is assigned to a slot. If not, from the received s', it will know to which unused slot it is assigned.

F. Phase Two: Missing-Tag Detection

At the beginning of this phase, the reader broadcasts a detection request, which is followed by a time frame for sampled tags to respond. The detection request consists of a frame size f and a sequence of seeds, $s, s_1, \ldots, s_k$, and s'. The time frame is divided into subframes. Before each subframe F_j, the reader broadcasts the corresponding seed-selection segment V_j in a single tag-ID slot T_{tag}. It is followed by l short slots (T_{short}) of the subframe, during which the tags in N_j can respond. Recall that each selection segment is 96 bits long. If $k = 7$, a segment has $l = \frac{96}{\log_2(7+1)} = 32$ selectors, and thus each time subframe has 32 slots.

Consider an arbitrary tag t. It wakes up to participate in a scheduled protocol execution for which it is sampled. After t receives the detection request from the reader, it uses $H(id, s)$ to determine to which subframe it is assigned. Without loss of generality, let the subframe be F_j. The tag sets the timer to wake up before F_j begins. After receiving the seed-selection segment V_j, tag t uses $H(id, s_1)$ to find out to which time-slot it is mapped by seed s_1. It then checks whether the corresponding selector in V_j is 1. If the selector is 1, according to the construction of V_j in Section III-E.3, t must be the sole tag that is mapped (and assigned) to this slot under s_1. If the selector is not 1, it means that s_1 should not be used to map any tag to this slot. In the latter case, t will move on to other seeds and repeat the same process to determine if it is assigned to a slot. If so, it will transmit during that slot. Otherwise, if t is not assigned to a slot after all k seeds, it will make a final attempt by finding out all unused slots (whose corresponding selectors in V_j are zeros) and using $H(id, s')$ as index to identify an unused slot to transmit.

In summary, after Phase One, the reader knows: 1) to which subframe each sampled tag is assigned; 2) which slot each sampled tag is expected to transmit; 3) which slots are expected to be singletons; and 4) which slots are expected to be collision slots (due to the final attempt using s'). After Phase Two, if an expected singleton/collision slot turns out to be idle, the reader detects a missing-tag event. Because multiple mappings reduce the number of empty/collision slots, both energy efficiency and time efficiency are greatly improved, as we will demonstrate analytically and by simulations in the following sections.

IV. Energy–Time Tradeoff in Protocol Configuration

We study the energy–time tradeoff of our protocol and show how to compute the system parameters.

A. Execution Time and Energy Cost

The protocol execution time includes the time for the reader to transmit a detection request, the time for the reader to transmit the seed-selection vector of f selectors, and the time frame of f slots for tags to transmit, where the seed-selection vector is divided into segments and the time frame is divided into subframes. The request only carries a few parameters. Its time is negligible when compared to the time frame and the seed-selection vector if f is large. Hence, the protocol execution time is roughly proportional to f. To investigate the energy–time tradeoff in relative terms, we characterize the protocol execution time by using the frame size f. A smaller value of f means a shorter protocol execution time. The actual execution time, measured in seconds, will be studied through simulations in Section VI.

The computation at each tag is mostly hashing. The hash function can be made very simple, such as [13] where hash output is produced by selecting a certain number of bits from a prestored bit ring. Moreover, once the tags receive the hash seeds from the reader's detection request, they can produce the needed hash values ahead of time, and all tags do so in parallel, while the reader is sending its seed-selection vector.

Our protocol design pushes most of its complexity to the reader. The tags' operation is simple: A tag wakes up to participate in a scheduled protocol execution for which it is sampled. It receives the detection request, determines to which subframe it is assigned, wakes up again before the subframe starts, receives the 96-bit seed-selection segment, determines to which slot it is assigned, and transmits a signal in that slot. Because each participating tag performs a similar operation, the energy cost to each participating tag is also similar. The total energy cost among all tags for each protocol execution is proportional to the number of participating tags; note that different tags will be sampled to participate in different protocol executions uniformly at random. Hence, we may characterize the energy cost of a protocol execution by using the expected number of participating tags, pn, which is in turn proportional to the sampling probability p.

B. Detection Probability

To find the detection probability after one protocol execution, we need to first derive the probability for an arbitrary sampled tag t to be assigned to a singleton slot during Phase One. There are k mappings. Let P_i be the probability that tag t is assigned to a singleton slot after the first i mappings. Let n be the total number of tags and n' be the number of sampled tags that are mapped to the same subframe as t does. Assume the hash function assigns sampled tags to subframes uniformly at random. n' follows a binomial distribution, $\mathrm{Bino}(n, p\frac{l}{f})$, i.e.,

$$\mathrm{Prob}\{n' = j\} = \binom{n}{j}\left(p\frac{l}{f}\right)^{j}\left(1 - p\frac{l}{f}\right)^{n-j}. \quad (1)$$

$P_0 = 0$. We derive a recursive formula for $P_i, 1 \le i \le k$. After the first $i-1$ mappings, there are two cases.

Case 1) Tag t has been assigned to a slot; the probability for this to happen is P_{i-1}.

Case 2) Tag t has not been assigned to a slot; the probability for this case is $1 - P_{i-1}$.

We focus on the second case.

In the ith mapping, the slot that tag t is mapped to has a probability of $(1 - \frac{n'P_{i-1}}{l})$ to be unused. Each of the other $n'-1$ tags has a probability $(1 - P_{i-1})$ to be unassigned. If it is unassigned, the tag has a probability of $\frac{1}{l}$ to be mapped to the same slot as t does. Hence, the probability p' for tag t to be the only one that is mapped to an unused slot is

$$p' = \left(1 - (1 - P_{i-1})\frac{1}{l}\right)^{n'-1}\left(1 - \frac{n'P_{i-1}}{l}\right). \quad (2)$$

Recall that we are considering Case 2 here. Combining both cases, we have

$$\begin{aligned} P_i &= P_{i-1} + (1 - P_{i-1})\sum_{j=1}^{n} Prob\{n' = j\}p' \\ &= P_{i-1} + (1 - P_{i-1})\sum_{j=1}^{n}\binom{n}{j}\left(p\frac{l}{f}\right)^{j}\left(1 - p\frac{l}{f}\right)^{n-j} \\ &\quad \times \left(1 - (1 - P_{i-1})\frac{1}{l}\right)^{j-1}\left(1 - \frac{jP_{i-1}}{l}\right) \end{aligned} \quad (3)$$

where the first item on the right side is the probability for a tag to be assigned to a slot by the first $i-1$ mappings, and the second item is the probability for the tag to be assigned to a slot by the ith mapping. The probability for tag t to be assigned to a slot after all k mappings is P_k.

After the k mapping, we have a final attempt, in which an unassigned tag may be mapped to a singleton slot or a collision slot. If the tag is mapped to a collision slot, it is highly unlikely that all tags in that slot will be missing because the parameter m is typically set far smaller than n. Hence, the contribution of collision slots to missing-tag detection can be ignored. When the tag is mapped to a singleton slot, detection will be made if the tag is missing. Therefore, the final mapping has no difference from the previous mappings. The probability for tag t to transmit in a singleton slot is P_{k+1}, which can be computed recursively from (3).

Each of the m missing tags has a probability p to be sampled. When the tag is sampled, it has a probability of P_{k+1} to be assigned a singleton slot. When that happens, since a missing tag cannot transmit, the reader will observe an idle slot instead, resulting in the detection. Therefore, the detection probability of MSMD, denoted as $P_{\mathrm{msmd}}(p, f)$, is

$$P_{\mathrm{msmd}}(p, f) = 1 - (1 - pP_{k+1})^{m}. \quad (4)$$

The value of $P_{\mathrm{msmd}}(p, f)$ not only depends on the choice of p and f, but also depends on n, m, and k, which are not included in the notation for simplicity. The values of p and f are determined by the reader and broadcast to tags. They control the energy–time tradeoff as we will reveal shortly. The values of n, m, and k are preknown, where n is known because it is simply the number of tags that the reader expects to be in the system, m is known as a given parameter in the detection requirement, and k is determined before the tags are deployed.

EMD [16] is a special case of MSMD with $k = 1$ and without the final attempt. Hence, the detection probability of EMD, denoted as $P_{\mathrm{emd}}(p, f)$, is

$$P_{\mathrm{emd}}(p, f) = 1 - (1 - pP_1)^{m}. \quad (5)$$

TRP [5] is a special case of EMD with $p = 1$. Namely, sampling is turned off.

C. Energy–Time Tradeoff Curve

We cannot arbitrarily pick small values for p and f. They must satisfy the requirement $P_{\mathrm{msmd}}(p, f) \ge \alpha$. Subject to this constraint, we show that the values of p and f cannot be minimized simultaneously. The choice of p and f represents an energy–time tradeoff.

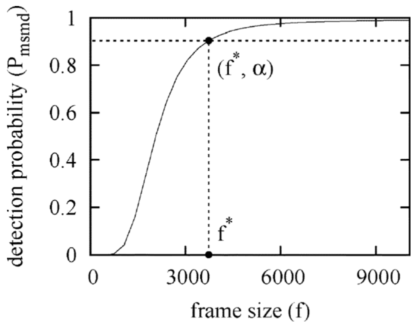


Fig. 3. Detection probability $P_{\rm msmd}(p,f)$ with respect to the frame size f when $n = 50\,000, m = 100, k = 3$, and $p = 5\%$.

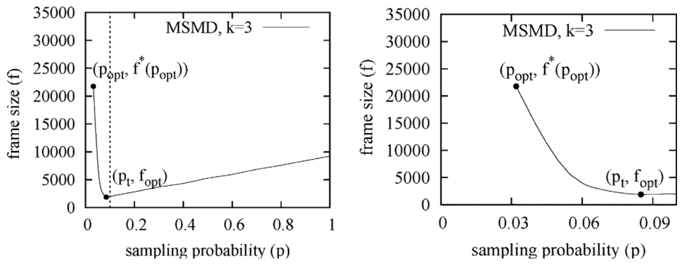


Fig. 4. *(left)* Energy–time tradeoff curve, i.e., frame size $f^*(p)$ with respect to sampling probability p, when $n = 50\,000, m = 75, k = 3$, and $\alpha = 95\%$. *(right)* Energy–time tradeoff curve in the range $p \in [p_{\rm opt}, p_t]$, which corresponds to the curve segment to the left of the dashed line in the left plot.

If we fix the value of p, $P_{\rm msmd}(p,f)$ becomes a function of f. The solid line in Fig. 3 shows an example of the curve $P_{\rm msmd}(p,f)$ with respect to f when $n = 50\,000, m = 100, k = 3$, and $p = 5\%$. Because $P_{\rm msmd}(p,f)$ is an increasing function, the minimum value of f that satisfies the requirement $P_{\rm msmd}(p,f) \geq \alpha$ can be found by solving the following equation:

$$P_{\rm msmd}(p,f) = \alpha.$$

The solution is denoted as f^*. See Fig. 3 for illustration.

For each different sampling probability p, we can compute the smallest usable frame size f^* that satisfies $P_{\rm msmd}(p,f) \geq \alpha$. Hence, f^* can be considered as a function of p, denoted as $f^*(p)$. A practical RFID system may consider a frame size beyond a certain upper bound U to be unacceptable due to excessively long execution time. In addition, f^* must be an integer. Considering these factors, we give a more accurate definition of f^* as follows:

$$f^*(p) = \min\{f | P_{\rm msmd}(p,f) \geq \alpha \wedge f \leq U, f \in I^+\}. \quad (6)$$

We can find the value of $f^*(p)$ through bisection search.

The left plot in Fig. 4 shows the curve of $f^*(p)$ when $n = 50\,000, m = 75, k = 3$, and $\alpha = 95\%$. We call it the *energy–time tradeoff curve*. Each point, $(p, f^*(p))$, represents an operating point whose energy cost is measured as np participating tags and whose time frame consists of $f^*(p)$ slots. The symbols in the plot will be explained later. The energy–time tradeoff is controlled by the sampling probability p. If we decrease the value of p, we decrease the energy cost, but at the mean time the value of $f^*(p)$ may have to increase, which increases the execution time.

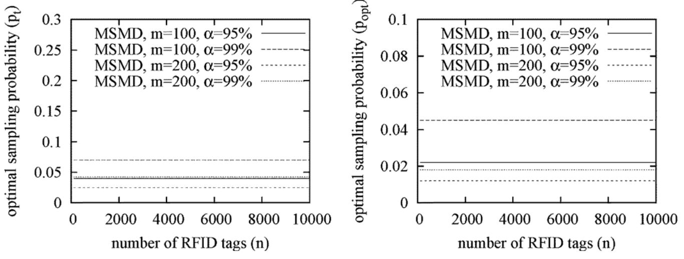


Fig. 5. *(left)* Value of p_t with respect to n. *(right)* Value of $p_{\rm opt}$ with respect to n.

D. Minimum Energy Cost

When the sampling probability p is too small, the detection probability $P_{\rm msmd}(p,f)$ will be smaller than α for any value of f. Such a small sampling probability cannot be used. We can use bisection search to find the smallest value of p, denoted as $p_{\rm opt}$, which can satisfy $P_{\rm msmd}(p,f) \geq \alpha$ with a frame size no greater than the upper bound U. When $p_{\rm opt}$ and $f^*(p_{\rm opt})$ are used, the energy cost is minimized.

E. Minimum Execution Time

From the energy–time tradeoff curve (the left plot in Fig. 4), we can find the smallest value of $f^*(p)$, denoted as $f_{\rm opt}$, that minimizes the execution time

$$f_{\rm opt} = \min\{f^*(p) \mid p_{\rm opt} \leq p \leq 1\}. \quad (7)$$

Let p_t be the corresponding sampling probability. Namely, $P_{\rm msmd}(p_t, f_{\rm opt}) = \alpha$. The values of $f_{\rm opt}$ and p_t can be determined through bisection search. When p_t and $f_{\rm opt}$ are used, the protocol execution time is minimized.

We amplify the segment of the energy–time tradeoff curve between point $(p_{\rm opt}, f^*(p_{\rm opt}))$ and point $(p_t, f_{\rm opt})$ in the right plot of Fig. 4. When we increase the value of p from $p_{\rm opt}$ to p_t, the energy cost of the protocol is linearly increased, while the execution time of the protocol is decreased. We should not choose $p > p_t$ because both energy cost and execution time will increase when the sampling probability is greater than p_t.

F. Offline Computation

Because the computation of $p_{\rm opt}, f^*(p_{\rm opt}), p_t$, and $f_{\rm opt}$ relies only on the values of n, m, α, and k, we can calculate them offline in advance. The values of m and α are preconfigured as part of the system requirement. The value of k is determined before tag deployment. Hence, we can precompute $p_{\rm opt}, f^*(p_{\rm opt}), p_t$, and $f_{\rm opt}$ in a table format with respect to different values of n, so that these values can be looked up during online operations.

When performing such computation, we observe that when we change n, the values of p_t and $p_{\rm opt}$ remain largely constants, as shown in Fig. 5. Hence, their values are actually determined by m, α, and k. It means that as long as the detection requirement specified by m and α does not change, p_t and $p_{\rm opt}$ can be approximately viewed as constants even though the number of tags in the system changes.

Suppose the values of m and α may be changed only at the beginning of each hour. The reader picks a sampling probability p, which is $p_t, p_{\rm opt}$, or a value between them. It then downloads p to all tags and synchronizes their clocks. For the rest of the

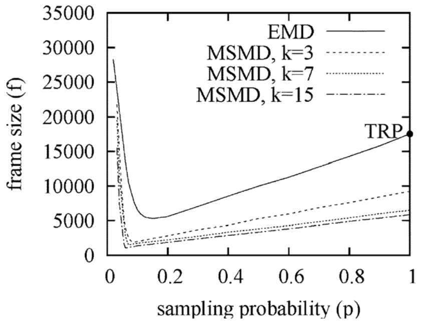


Fig. 6. Energy–time tradeoff curves of EMD and MSMD under different k values, when $n = 50\,000$, $m = 100$, and $p = 5\%$.

hour, the reader does not have to transmit the sampling probability again.

G. Constrained Least-Time (or Least-Energy) Problem

The *energy-constrained least-time problem* is to minimize the protocol's execution time, subject to a detection requirement specified by m and α and an energy constraint specified by an upper bound u on the expected number of tags that participate in each protocol execution. To minimize execution time, we need to reduce the frame size as much as possible. Our previous analysis has already given the solution to this problem, which is simply $f^*(\frac{u}{n})$, where $\frac{u}{n}$ is the maximum sampling probability that we can use under the energy constraint.

The *time-constrained least-energy problem* is to minimize the number of tags that participate in protocol execution, subject to a detection requirement specified by m and α and an execution time constraint specified by an upper bound u' on the frame size. A solution can be designed by following a similar process as we derive $f^*(p)$ in Section IV-C: Starting from (4), if we fix $f = u'$, $P_{\text{msmd}}(p, f)$ becomes a function of p. We can use bisection search to find p that meets $P_{\text{msmd}}(p, f) = \alpha \wedge p \leq p_t$.

H. Impact of k

We study how the number k of hash seeds will affect the protocol's performance. Fig. 6 compares the energy–time tradeoff curves of EMD and MSMD with $k = 3, 7, 15$, respectively. Recall that EMD is a special case of MSMD with one hash seed, and TRP is a special case of EMD with $p = 1$, represented by a point on the curve of EMD as shown in the figure. For MSMD, when $k = 3$, each seed selector needs 2 bits; recall that the value zero is reserved for non-singleton slots. When $k = 7$, each selector needs 3 bits. When $k = 15$, each selector needs 4 bits.[3]

In Fig. 6, a lower curve indicates better performance because, for any sampling probability, its frame size is smaller, i.e., its execution time is smaller. Alternatively, it can be interpreted as, for any frame size, its sampling probability is smaller, i.e., it needs fewer tags to participate in each protocol execution. Clearly, MSMD significantly outperforms EMD and TRP. As k increases, the performance of MSMD improves. However, the amount of improvement shrinks rapidly, demonstrated by the small gap between $k = 7$ and $k = 15$. When we further increase k to 31 using 5-bit selectors, the improvement becomes negligible. Increasing the value of k does not come for free; larger selectors mean more overhead. For one, it takes more time for the reader to broadcast the seed-selection vector. Therefore, we believe $k = 7$ is a good choice in practice because the performance gain beyond that is very limited.

[3]One may ask why we do not use $k = 8$ or other values. The reason is that each selector needs 4 bits even when $k = 8$. In that case, we should certainly choose $k = 15$ for better performance.

TABLE II
IMPACT OF CORRUPTED SLOTS

Original slot	After corruption	Tag status	Reader thinks
idle	busy	missing	not missing
busy	busy	not missing	not missing

V. MSMD Over Unreliable Channels

We now consider unreliable channels. The intuition behind EMD and MSMD is that a slot will be found idle if the tags transmitting in the slot are all missing. It is true if no error occurs in this slot. In reality though, the communication between a reader and tags is, to varying degrees, subject to noise/interference in the environment, which may corrupt slots, for example, turning a would-be empty slot to a busy slot. Table II gives different possibilities of corrupted slots and their consequences. In the first row of this table, we assume that a tag t is missing. Then, the slot that t is mapped to should become idle during the protocol execution. However, this slot may be corrupted and turn out to be a busy slot. In this case, even if a tag that is supposed to transmit in a slot is missing, the reader can still sense a response induced by channel error, resulting in undetection of a missing tag. If all slots assigned to missing tags happen to be corrupted, the reader will fail to detect the missing-tag event. The second row illustrates the impact of channel error when t is present in the system. Recall that we only distinguish two states for a slot: idle (no signal detected in the channel) or busy (signal detected). If t is not missing, the original slot should be busy. Since noise or interference in the environment may change the signal but is unlikely to cancel the signal out altogether, the slot should remain a busy slot. In this case, noise/interfere does not cause harm.

We evaluate the impact of channel error under two different models: random error model and burst error model. The former can be characterized by a parameter err, which denotes the probability for each slot to incur error. However, it may happen that channel introduces error in bursts for consecutive slots rather than at random. For example, communications between a reader and tags may be interfered by electromagnetic emission from nearby devices that share the same frequency band. When those devices are transmitting (e.g., sending a packet), their signals keep the channel busy for a small period of time until they stop. As the interfering transmissions are turned on and off, it causes bursts of error to the RFID system, which is characterized by the burst error model.

We stress that the error models are applied only to the tag-to-reader link in order to simplify the analysis. Error on the reader-to-tag link is addressed separately as follows: The transmission from the reader carries a CRC checksum. For example, the 96-bit seed-selection segment may contain a 16-bit CRC checksum and use the remaining 80 bits to encode seed selectors. When a tag receives the transmission from the reader, it computes a CRC based on the received information and then compares the result to the received CRC. If they are the same, the tag performs the operation according to the protocol. Otherwise, the tag will not participate further in the protocol execution, and instead it will transmit its ID to the reader to announce its presence after the protocol execution. This adds additional execution time and energy cost. However, if the error ratio is small, the additional overhead will be small.

A. MSMD Under Random Error Model

In the random error model, the impact of channel error is characterized by a parameter err, which is the probability for a slot to incur error. For example, if $err = 5\%$, a would-be idle slot has a chance of 5% to become busy.

MSMD under this model is called MSMD-re. From Section IV-B, we know that each of the m missing tags has a probability of pP_{k+1} to be detected in MSMD. Then, the probability for a missing tag to be detected under the random error model is $pP_{k+1}(1-err)$. Therefore, the detection probability of MSMD-re, denoted as $P_{\text{msmd-re}}(p, f, err)$, is

$$P_{\text{msmd-re}}(p, f, err) = 1 - (1 - pP_{k+1}(1-err))^m. \tag{8}$$

Clearly, MSMD is a special case of MSMD-re with $err = 0$. For MSMD-re, the computation of $p_{\text{opt}}, f^*(p_{\text{opt}}), p_t$, and f_{opt} is similar to that of MSMD except that (5) is replaced with (8). MSMD-re uses the same offline computation process as described in Section IV-F and follows the same protocol, only with modified parameters that consider the impact of channel error.

B. MSMD Under Burst Error Model

We now consider the burst error model. According to [24], the number of bursts can be approximated as Poisson distribution. We give a brief description here, and readers are referred to [24] for details. The probability density function for the number of bursts is given by

$$h(x) = \sum_{i=0}^{\infty} \frac{\eta^i}{i!} e^{-\eta} \delta(x - i) \tag{9}$$

where η is the average number of bursts, and $\delta(\cdot)$ is the Dirac Delta Function [25].

According to convolutional codes and trellis code modulations, the probability density function for the number of errors in a burst can be derived by Erlang density of second order. The Erlang distribution of second order is

$$g_c(z) = (2\mu)^2 z e^{-2\mu z} \tag{10}$$

where μ is the rate parameter. Because the random variable for the number of errors in a burst assumes only discrete values [24], the probability density function can be obtained by discretizing the Erlang distribution $g_c(z)$

$$g(y) = \sum_{w=1}^{\infty} P_E(w) \delta(y - w) \tag{11}$$

with

$$\begin{aligned} P_E(w) &= P(w-1 < z_c \le w) = \int_{w-1}^{w} \sum g_c(z) dz \\ &= e^{-2\mu w}[(e^{2\mu} - 1)(1 + 2\mu w) - 2\mu e^{2\mu}]. \end{aligned} \tag{12}$$

Here, z_c is the continuous Erlang distributed random variable and $P_E(w)$ represents the probability of having w errors in a burst. Then, the mean value of the distribution given in (11) is

$$E\{g(y)\} = \frac{e^{2\mu}(e^{2\mu} + 2\mu - 1)}{(e^{2\mu} - 1)^2}. \tag{13}$$

The probability of having w errors in N bits, $P_N(w)$, depends on the number of bursts in the interval, $P_B(j)$, and on the number of errors in bursts, $P_E(w)$. Therefore, we have

$$P_N(w) = \begin{cases} P_B(0), & \text{for } w = 0 \\ \sum_{j=1}^{\infty} P_e^{(j)}(w) P_B(j), & \text{for } w > 0. \end{cases} \tag{14}$$

Here, $P_e^j(w)$ is the probability to have w errors in j bursts in the interval of N bits

$$P_e^{(j)}(w) = \begin{cases} \sum_{n=1}^{w} P_e^{(j-1)}(w - n) P_E(n), & \text{for } j > 1 \\ P_E(w), & \text{for } j = 1 \end{cases} \tag{15}$$

and $P_B(j)$ represents the probability of having j bursts in an interval

$$P_B(j) = \frac{\eta^j}{j!} e^{-\eta}. \tag{16}$$

From (12) and (14)–(16), we know that the computation of $P_N(w)$ relies on the values of μ and η. Now we should find a way to obtain the values of these two parameters.

We denote the mean value of errors in N bits (i.e., the value of $E\{g(y)\}$ when there are N bits) as N_e. If we know N_e, we can compute the value of μ from (13). According to [24], the value of η depends on the probability that a burst occurs and causes errors in the interval of N bits. It is given by

$$\eta = \frac{Nr}{N_e} \tag{17}$$

where r is a parameter called Bit Error Rate. Finally, the value of N_e is evaluated as follows:

$$N_e = \frac{N N_m}{p_0(N + L_m - 1)} \tag{18}$$

where p_0 is the probability of having at least one error in the considered bits when a burst occurs, and p_0 is given by

$$p_0 = 1 - \left(1 - \frac{N_m}{N + L_m - 1}\right)^N. \tag{19}$$

Here, N_m is the mean value for number of errors per burst and L_m is the mean value of burst error length.

From (13), (17), and (18), we can see that the computation of μ and η depends on the values of N_m, L_m, r, and N, which are input system parameters. After computing the values of μ and η, we can obtain the value of $P_N(w)$. For example, if $N_m = 9.5, L_m = 33.5, r = 10^{-3}$, and $N = 10$, the values of μ and η are respectively 0.52 and 4.110^{-3}. Then, $P_N(w)$ can be calculated by (14).

From Section III, we know that each slot only carries binary information of either "1" or "0." The information in the time frame therefore represents a bit array of length f. As the time frame is divided into subframes, each containing l slots, we also divide the bit array into segments of l bits, which allows our protocol to deal with one segment at a time. Consider the jth subframe F_j. Recall from (14) that the probability of having $w(0 \leq w \leq l)$ errors in l bits is $P_l(w)$. Then, the probability for a missing tag, which is assigned to F_j, to be detected is $pP_{k+1}P_l(w)(1-\frac{w}{l})$. The detection probability of MSMD under the burst error model can be computed by summing over all possible values of w.

Hence, the detection probability of MSMD under the burst error model, denoted as $P_{\text{msmd-be}}$, is

$$P_{\text{msmd-be}}(p, f, N_m, L_m, r) = 1 - \left(1 - \sum_{w=0}^{l} pP_{k+1}P_l(w)\left(1-\frac{w}{l}\right)\right)^m. \quad (20)$$

Under this model, we can obtain the values of $p_{\text{opt}}, f^*(p_{\text{opt}}), p_t$, and f_{opt} by following the same procedure as described in Section IV-F, except that (5) is replaced with (20).

VI. Numerical Results

We have performed extensive simulations to study the performance of the proposed MSMD, MSMD-re, and MSMD-be, and compared it to EMD [16] and TRP [5]. The design of all fives protocols ensures that the detection requirement specified by m and α is always met. This is indeed what we observe in our simulations.

The performance comparison is made in terms of energy efficiency and time efficiency, given a certain detection requirement. To measure the protocol execution time, we set the transmission parameters based on the typical setting of the EPC-global Gen-2 standard [26]. Any two consecutive transmissions (from the reader to tags or vice versa) are separated by a waiting time of 266.4 μs. The transmission rate from the reader to tags is 40.97 kb/s; it takes 24.41 μs for the reader to transmit one bit. A 96-bit slot that carries a seed-selection segment is 2609.76 μs long, which includes a waiting time before the transmission. The transmission rate from a tag to the reader is also 40.97 kb/s, so that a single-bit slot T_{short} for a tag to respond (i.e., make the channel busy) is 290.81 μs, also including a waiting time.

For each set of system parameters, including m, α, and n, TRP will compute its optimal frame size. Once the frame size f is determined, the execution time is known, which is fT_{short} plus the time for broadcasting a detection request. MSMD, MSMD-re, MSMD-be and EMD will choose a sampling probability p, and compute the optimal frame size f^* under that sampling probability. EMD does not give a way to compute its optimal frame size. However, since EMD is a special case of MSMD, we use our analytical framework in the previous section to compute it. For EMD, its execution time is f^*T_{short}, plus the time for a request. For MSMD, MSMD-re, and MSMD-be, we need to add the time for transmitting the selection vector.

We cannot find a well-accepted energy model for RFID tags or detailed parameters of energy expenditure for an RFID standard. However, as we have explained in Section IV-A, the energy cost can be indirectly measured by the number of participating tags because the former is proportional to the latter. We use this measurement to study the energy–time tradeoff in relative terms and make performance comparison. As part of our future work, we will investigate the exact energy cost (in mJ) of tags when an appropriate energy model for RFID tags becomes available. Although the exact energy consumption of a tag can be simulated at this time (which depends on physical-layer implementation), we point out that each participating tag only needs to receive a small amount of data from the reader and transmits one-bit information.

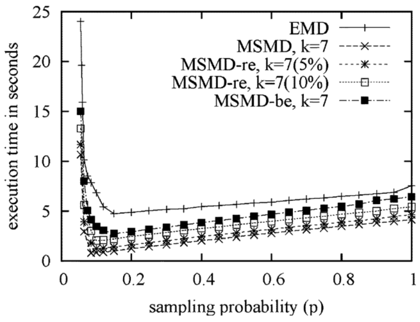


Fig. 7. Protocol execution time with respect to sampling probability, when $\alpha = 95\%$, $m = 50$, and $n = 50\,000$.

A. Energy–Time Tradeoff

Let $n = 50\,000, \alpha = 95\%$, and $m = 50$. For MSMD-be, the required input parameter setting is similar to that in [25]: $N_m = 9.5, L_m = 33.5$, and $r = 10^{-3}$. Fig. 7 shows the energy–time tradeoff curves produced by simulations. Recall that the energy cost of MSMD or EMD is proportional to p. The point at $p = 1$ on the EMD curve represents TPR. Clearly, MSMD significantly outperforms EMD. MSMD with $k = 7$ uses three-bit elements in the selection vector, while MSMD with $k = 3$ uses two-bit elements. Even though it incurs more overhead in the selection vector, MSMD with $k = 7$ slightly outperforms MSMD with $k = 3$. Further increasing k cannot bring performance gain due to overly large overhead for the selection vector. Furthermore, it is also shown in Fig. 7 that when we take the impact of channel error into consideration, the performance of MSMD with $k = 7$ degrades slightly, which can be illustrated by the curves of MSMD-re and MSMD-be. For MSMD-re, increasing err will cause more execution time and energy cost because of large probability for a slot to incur error. Even though channel errors cause more overhead in missing-tag detection (which should be an expected consequence), a key finding is that the general energy–time tradeoff relation stays the same. In Fig. 8, we zoom in for a detailed look at the curve segment in the sampling probability range of [0, 0.2]. When $p = 0.08$, the execution time of MSMD with $k = 7$ is 11.2% of the time taken by EMD. When we fix the execution time at 5 s, the number of participating tags in MSMD with $k = 7$ is 46.7% of the number in EMD. We do not directly compare MSMD-re and MSMD-de to EMD because the latter does not consider channel error. We vary the values of n, α and m. Similar conclusions can be drawn from the simulation results.

The tradeoff curves in Fig. 8 agree with our analytical results in Fig. 6 in principle. We want to stress that our simulations do not simply reproduce the analytical results. Simulations consider system details by using a real RFID specification. Such details are not captured by analysis. In addition, simulations consider the exact impact of selection vector on execution time (measured in seconds), instead of characterizing time in an indirect way using the frame size.

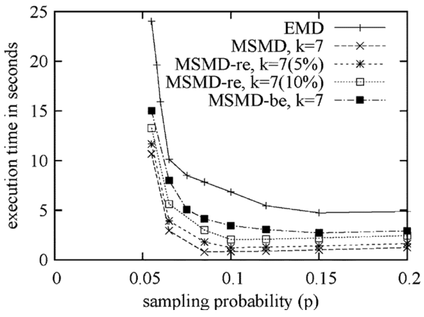


Fig. 8. Zoom-in view of energy–time tradeoff in Fig. 7 in the sampling probability range of [0, 0.2].

TABLE III
RELATIVE ENERGY COST AND EXECUTION TIME OF MSMD ($k = 7$) UNDER p_{opt} AND p_t, WHEN $\alpha = 95\%$ AND $n = 50\,000$

	p_t		p_{opt}	
	energy	time	energy	time
$m = 200$	2.1%	4.09%	1.4%	269.1%
$m = 100$	3.6%	5.61%	2.5%	226.2%
$m = 50$	8.1%	10.84%	5.5%	82.3%

In Table III, we show the relative performance of MSMD ($k = 7$) with respect to TRP, where $n = 50\,000, \alpha = 95\%$, and $m = 50, 100$, or 200. MSMD is operated under sampling probability p_t and p_{opt}. For example, when $m = 50, p_t = 0.085$ and $p_{\text{opt}} = 0.055$. The numbers in the table are ratios of MSMD's energy cost (or execution time) to TRP's energy cost (or execution time). For example, when $m = 200$, the energy cost of MSMD with p_t is 2.1% of what TRP consumes, and its execution time is 4.09% of the time TRP takes. Again, we do not directly compare MSMD-re and MSMD-de with TRP because the latter does not consider channel error.

B. Performance Comparison

Next, we compare the performance of MSMD ($k = 7$), MSMD-re ($err = 5\%$ and $k = 7$), MSMD-be ($k = 7$), EMD, and TRP under different values of m, α, and n. MSMD, MSMD-re, MSMD-be, and EMD are operated with their optimal sampling probabilities p_t. In Figs. 9–11, we keep $m = 50$ and vary the value of α. In Fig. 9, we let $\alpha = 99.9\%$, meaning that each protocol execution should detect any missing-tag event with probability 99.9%. The left plot compares the energy cost of five protocols with respect to n, and the right plot compares their execution times. MSMD has a smaller energy cost than EMD, which in turn has a much smaller energy cost than TRP. Taking the impact of channel error into consideration, the energy costs by MSMD-be and MSMD-re increase modestly over MSMD. In the meanwhile, MSMD also has a much smaller execution time than EMD and TRP. Taking channel error into consideration, the execution times of MSMD-be and MSMD-re increase modestly over MSMD. Similar results can be drawn from Fig. 10, where $\alpha = 99\%$, and Fig. 11, where $\alpha = 90\%$. In the latter case, the execution time of MSMD is less than a second.

In Figs. 12 and 13, we keep $\alpha = 99\%$ and vary the value of m. In Fig. 12, $m = 25$. In Fig. 13, $m = 100$. The performance of MSMD remains the best among all five.

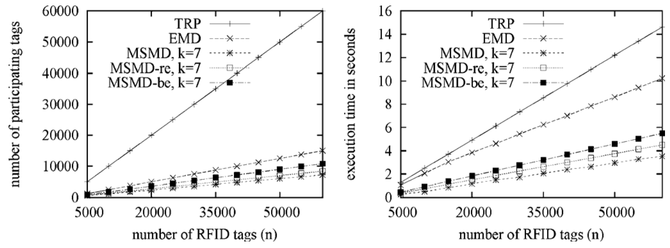


Fig. 9. *(left)* Number of participating tags with respect to the number of tags, when $m = 50$ and $\alpha = 99.9\%$. *(right)* Protocol execution time with respect to the number of tags, when $m = 50$ and $\alpha = 99.9\%$.

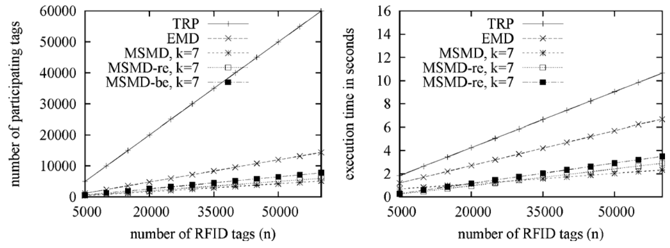


Fig. 10. Same as the caption of Fig. 9, except for $\alpha = 99\%$.

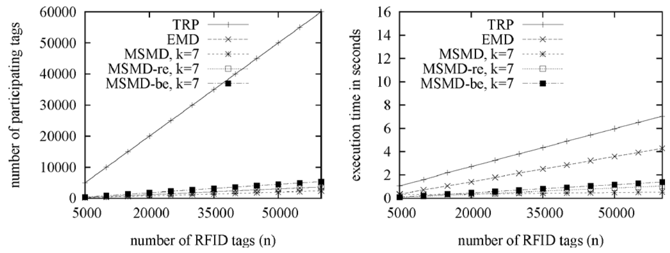


Fig. 11. Same as the caption of Fig. 9, except for $\alpha = 90\%$.

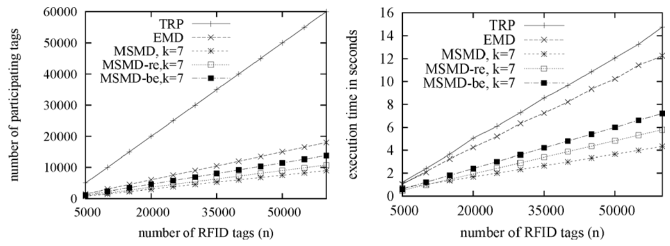


Fig. 12. *(left)* Number of participating tags with respect to the number of tags, when $m = 25$ and $\alpha = 99\%$. *(right)* Protocol execution time with respect to the number of tags, when $m = 25$ and $\alpha = 99\%$.

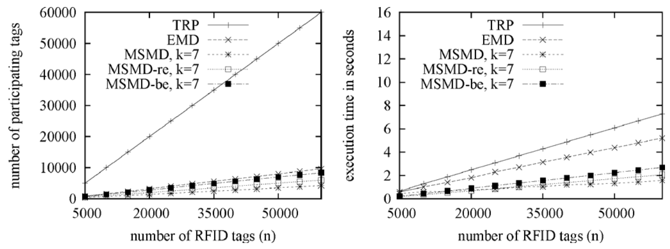


Fig. 13. Same as the caption of Fig. 12, except for $m = 100$.

VII. CONCLUSION

This paper proposes a new protocol design that integrates energy efficiency and time efficiency for missing-tag detection. It uses multiple hash seeds to provide multiple degrees of freedom for the RFID reader to assign tags to singleton slots, during which the tags announce their presence in the process of missing-tag detection. We first present this new protocol with reliable channels. The result is a multifold cut in both energy cost and execution time. Such performance improvement is critical for a protocol that needs to be executed frequently. Then, we extend the protocol to consider two categories of channel errors induced by noise/interference in the environment. The involving of channel errors will make the energy/time gains slightly reduced, but remain significant compared to EMD and TRP. We also reveal a fundamental energy–time tradeoff in the protocol design. This tradeoff gives flexibility in performance tuning when the protocol is applied in practical environment.

REFERENCES

[1] Y. Liu, L. Chen, J. Pei, Q. Chen, and Y. Zhao, "Mining frequent trajectory patterns for activity monitoring using radio frequency tag arrays," in *Proc. IEEE PerCom*, 2007, pp. 37–46.

[2] L. M. Ni, Y. Liu, Y. C. Lau, and A. Patil, "LANDMARC: Indoor location sensing using active RFID," *Wireless Netw.*, vol. 10, no. 6, pp. 701–710, Nov. 2004.

[3] Y. Li and X. Ding, "Protecting RFID communications in supply chains," in *Proc. ASIACCS*, 2007, pp. 234–241.

[4] B. Sheng, C. Tan, Q. Li, and W. Mao, "Finding popular categories for RFID tags," in *Proc. ACM MobiHoc*, 2008, pp. 159–168.

[5] C. Tan, B. Sheng, and Q. Li, "How to monitor for missing RFID tags," in *Proc. IEEE ICDCS*, 2008, pp. 295–302.

[6] A. Nemmaluri, M. Corner, and P. Shenoy, "Sherlock: Automatically locating objects for humans," in *Proc. ACM MobiSys*, 2008, pp. 187–198.

[7] Y. Zheng and M. Li, "Fast tag searching protocol for large-scale RFID systems," in *Proc. IEEE ICNP*, Oct. 2011, pp. 363–372.

[8] Y. Zheng, M. Li, and C. Qian, "PET: Probabilistic estimating tree for large-scale RFID estimation," in *Proc. IEEE ICDCS*, Jun. 2011, pp. 37–46.

[9] L. Ravindranath, V. Padmanabhan, and P. Agrawal, "SixthSense: RFID-based enterprise intelligence," in *Proc. ACM MobiSys*, 2008, pp. 253–266.

[10] T. Li, S. Wu, S. Chen, and M. Yang, "Energy efficient algorithms for the RFID estimation problem," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.

[11] S. Chen, M. Zhang, and B. Xiao, "Efficient information collection protocols for sensor-augmented RFID networks," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 3101–3109.

[12] Y. Qiao, S. Chen, T. Li, and S. Chen, "Energy-efficient polling protocols in RFID systems," in *Proc. ACM MobiHoc*, May 2011, Art. no. 25.

[13] T. Li, S. Chen, and Y. Ling, "Identifying the missing tags in a large RFID system," in *Proc. ACM MobiHoc*, 2010, pp. 1–10.

[14] B. Sheng, Q. Li, and W. Mao, "Efficient continuous scanning in RFID systems," in *Proc. IEEE INFOCOM*, 2010, pp. 1–9.

[15] R. Zhang, Y. Liu, Y. Zhang, and J. Sun, "Fast identification of the missing tags in a large RFID system," in *Proc. 8th Annu. IEEE Commun. Soc. Conf. Sensor, Mesh Ad Hoc Commun. Netw.*, 2011, pp. 278–286.

[16] W. Luo, S. Chen, T. Li, and S. Chen, "Efficient missing tag detection in RFID systems," in *Proc. IEEE INFOCOM, Mini-Conf.*, 2011, pp. 356–360.

[17] B. Firner, P. Jadhav, Y. Zhang, R. Howard, W. Trappe, and E. Fenson, "Towards continuous asset tracking: Low-power communication and fail-safe presence assurance," in *Proc. 6th Annu. IEEE Commun. Soc. Conf. Sensor, Mesh Ad Hoc Commun. Netw.*, 2009, pp. 1–9.

[18] P. Popovski, K. F. Nielsen, R. M. Jacobsen, and T. Larsen, "Robust statistical methods for detection of missing RFID tags," *IEEE Wireless Commun.*, vol. 18, no. 4, pp. 74–80, Aug. 2011.

[19] R. M. Jacobsen, K. F. Nielsen, P. Popovski, and T. Larsen, "Reliable identification of RFID tags using multiple independent reader sessions," in *Proc. IEEE RFID*, 2009, pp. 64–71.

[20] J. Myung and W. Lee, "Adaptive splitting protocols for RFID tag collision arbitration," in *Proc. ACM MobiHoc*, May 2006, pp. 202–213.

[21] N. Bhandari, A. Sahoo, and S. Iyer, "Intelligent query tree (IQT) protocol to improve RFID tag read efficiency," in *Proc. IEEE Int. Conf. Inf. Technol.*, Dec. 2006, pp. 46–51.

[22] 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.

[23] B. Zhen, M. Kobayashi, and M. Shimizu, "Framed ALOHA for multiple RFID object identification," *IEICE Trans. Commun.*, vol. 88-B, no. 3, pp. 991–999, Mar. 2005.

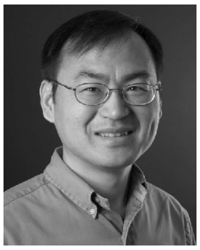
[24] B. Cornaglia and M. Spini, "Letter: New statistical model for burst error distribution," *Eur. Trans. Telecommun.*, vol. 7, no. 3, pp. 267–272, May 1996.

[25] "Dirac delta function," 2013 [Online]. Available: http://en.wikipedia.org/wiki/Dirac_delta_function

[26] EPCglobal, "EPC radio-frequency identity protocols Class-1 Generation-2 UHF RFID protocol for communications at 860 MHz–960 MHz," ver. 1.0.9, 2005 [Online]. Available: http://www.epcglobalinc.org/standards/uhfc1g2/uhfc1g2_1_0_9-standard-20050126.pdf

Wen Luo received the B.S. degree in computer science and technology from the University of Science and Technology of China, Hefei, China, in 2008, and is currently pursuing the Ph.D. degree in computer and information science and engineering at the University of Florida, Gainesville, FL. USA.

His advisor is Dr. Shigang Chen, and his research interests include RFID technologies and Internet traffic measurement.



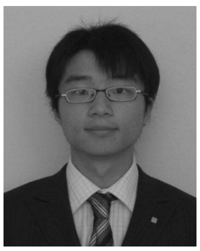
Shigang Chen (M'04–SM'12) received the B.S. degree from the University of Science and Technology of China, Hefei, China, in 1993, and the M.S. and Ph.D. degrees from the University of Illinois at Urbana–Champaign, Urbana, IL, USA, in 1996 and 1999, respectively, all in computer science.

After graduation, he worked with Cisco Systems, San Jose, CA, USA, for three years before joining the University of Florida, Gainesville, FL, USA, in 2002, where he is currently an Associate Professor with the Department of Computer and Information Science and Engineering. He served on the technical advisory board for Protego Networks from 2002 to 2003. He holds 11 US patents. He published more than 100 peer-reviewed journal/conference papers. 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 served in the steering committee of IEEE IWQoS from 2010 to 2013. He received the IEEE Communications Society Best Tutorial Paper Award in 1999 and the NSF CAREER Award in 2007.

Yan Qiao received the B.S. degree in computer science and technology from Shanghai Jiao Tong University, Shanghai, China, in 2009, and is currently pursuing the Ph.D. degree in computer and information science and engineering at the University of Florida, Gainesville, FL, USA.

Her advisor is Dr. Shigang Cheng. Her research interests include network measurement, algorithms, and RFID protocols.



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

He has since worked with Google, Mountain View, CA, USA. His research interests include network traffic measurement and RFID technologies.