Probabilistic Missing-tag Detection and Energy-Time Tradeoff in Large-scale RFID Systems

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

ABSTRACT

RFID (radio frequency identification) technologies are poised to revolutionize retail, warehouse and supply chain management. One of their interesting applications is to automatically detect missing tags (and the associated objects) in a large storage space. In order to timely catch any missing event such as theft, the detection operation may have to be performed frequently. Because RFID systems typically work under low-rate channels, past research has focused on reducing execution time of a detection protocol, in order to prevent excessively-long protocol execution from interfering normal inventory operations. However, when active tags are used to provide a large spatial coverage, energy efficiency becomes critical in prolonging the lifetime of these battery-powered tags. Existing literature lacks thorough study on how to conserve energy in the process of missing-tag detection and how to jointly optimize energy efficiency and time efficiency. This paper makes two important contributions: First, we propose a novel protocol design that takes both energy efficiency and time efficiency into consideration. It achieves multi-fold reduction in both energy cost and execution time when comparing with the best existing work. In some cases, the reduction is more than an order of magnitude. Second, we reveal a fundamental energy-time tradeoff in missing-tag detection. Through our analytical framework, we are able to flexibly control the tradeoff through a couple of system parameters in order to achieve desirable performance.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless communication

General Terms

Algorithms, Performance

Keywords

RFID Systems, Energy-Time Tradeoff, Missing Tag Detection

1. INTRODUCTION

RFID (radio frequency identification) technologies [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11] are poised to revolutionize retail, warehouse and supply chain management. They are also having a profound impact on our daily lives, with important applications in automatic toll payment, access control to parking garages, object tracking, and theft prevention. Comparing with barcodes that have to be read from a very close range by a laser scanner, RFID tags have great advantages: they can be read wirelessly over a distance, and they are able to perform simple computations. Starting from August 1, 2010, Wal-Mart has begun to embed RFID tags in clothing [12].

An interesting application of RFID tags 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 to 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 a 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, 14, 15] gives much stronger results, but its overhead is far greater than a probabilistic detection protocol [6, 16, 17, 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 executions will detect the event with 99.99999999% probability. If that is not enough, we may schedule an exact detection protocol [13] every five times the probabilistic detection protocol is executed.[2]

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

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

[2]Another exact detection protocol [14] reports which tags are missing, but does not guarantee to report all missing ones. The protocol in [13] guarantees 100% reporting.

every few minutes in a warehouse. If the execution time of the protocol is a minute, any normal operations that move goods out are likely to trigger false alarm. To reduce the chance of interfering 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. 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 hours a day, day by day. To make the problem more challenging, energy efficiency and time efficiency are often conflicting objectives that cannot be optimized simultaneously.

Despite its importance, the problem of probabilistic missing-tag detection is relatively new and under-investigated. The basic detection method is introduced in the pioneer work [6]: A 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 kept busy by transmission from another tag. Consequently, the reader cannot guarantee the detection of a missing-tag event. The protocol in [6] only considers time efficiency, but not energy efficiency. As a follow-up work [16] points out, even for time efficiency, it is far from being optimal. 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 efficiency 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 this paper, we make two 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 which slots they will transmit in. 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.

The rest of the paper is organized as follows: Section 2 gives the system model and problem definition, as well as the prior art. Section 3 proposes a new missing-tag detection protocol. Section 4 investigates energy-time tradeoff in protocol configuration. Section 5 evaluates the protocol through simulations. Section 6 draws the conclusion.

2. PRELIMINARIES

2.1 System Model

There are three types of RFID tags [19]. *Passive tags* — the kind used by Wal-Mart — are most widely deployed today. They are cheap, but do not have internal power sources. Passive tags rely on radio waves emitted from a 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 manufactural 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. In this paper, we consider active tags.

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. Based on the parameters of Philips I-Code [20], T_{tag} is about 3927 μs, and T_{short} is about 321 μs. See Section 5 for details.

2.2 Problem

The problem is to design an efficient protocol for a 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, a big shoe store may carry tens of thousands of shoes. 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 the protocol is periodically executed, 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 consider two performance metrics: execution time of the protocol and energy cost to the tags. The former is measured as the time it takes the protocol to complete one execution. We cannot find a well-accepted energy model for RFID tags. However, in our design, each tag will wake up only if it will participate in a scheduled protocol execution, and the energy expenditure for participating tags is about the same; they receive the same amount of data, stay active for a similar amount of time, and for most of them, transmit the same amount of data. Therefore, we can use the number of tags that participate in each protocol execution as an approximate measurement for energy cost.

The above two performance metrics are important due to the following reasons: First, RFID systems use low-rate communication channels. In the Philips I-Code system, the rate from a reader to a tag is about 27Kbps and the rate from a tag to a reader is about 53Kbps. Low rates, coupled with a large number of tags, often cause long execution times for RFID protocols. To apply such protocols in a busy warehouse environment, it is desirable to reduce protocol execution time as much as possible. Second, active tags carry limited battery power. Replacing tags or their batteries is a tedious, manual operation. One way to save energy is to minimize the number of tags that are needed to participate in each protocol execution. When a tag participates in 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 readerŠs battery can be easily replaced or it may be powered by an external source.

We assume that the RFID reader has access to a database that stores the IDs of all tags. This assumption is necessary [6]. Without any prior knowledge of a tag's existence, how can we know that it is missing?

The above 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 [21, 22, 23, 24] that reads the IDs from the tags. In this case, we will not detect missing-tag events that have already happened. However, now that we have the IDs of the remaining tags, we can detect the missing-tag events after this point of time.

2.3 Prior work

We first describe the Trusted Reader Protocol (TRP) by Tan, Sheng and Li [6]. To initiate the execution of the protocol, a 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 a random number r. Each tag maps itself to a slot in the frame by hashing its ID and r. It then transmits during that slot.

A slot is said to be *empty* if no tag responds (transmits) in the slot. It is called a *singleton slot* if exactly one tag responds. It is a *collision slot* if more than one tag responds. A singleton or collision slot is also called a *busy slot*.

The reader records which slots are busy and which are empty. 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. Because the reader knows the IDs of all tags, it knows which tags are mapped to which slots. More specifically, it knows which slots are expected to be busy. If an expected busy slot turns out to be empty, 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. B. 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 also we reveal a fundamental energy-time tradeoff in this design space, which allows us to adapt protocol performance to suit various needs in practical systems.

3. MULTI-SEED MISSING-TAG DETECTION PROTOCOL (MSMD)

3.1 Motivation

Both TRP [6] 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

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

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. 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 empty, 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 empty. However, when the number of missing tags is small when comparing with 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 fewer time slots as possible, which results in a large percentage of collision slots. More specifically, 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.[3] Now, 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 idea is that if we can find a way to turn most empty/collision slots into singletons,

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

we shall be able to improve time/energy efficiencies significantly. 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 $1 - (1 - p \times \theta)^m$ because each of the m missing tags has a probability of $p \times \theta$ 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 are needed to ensure a certain level of singletons.

Before we present our design called *Multiple-Seed Missing-tag Detection protocol* (MSMD), we first discuss how to implement a hash function efficiently for RFID tags below.

3.2 Hash Function

There exist many efficient hash functions in the literature. In order to keep the tag circuit simple, we build our hash function on top of the simple scheme in [13] using a ring of pre-stored random bits: Before a tag is deployed, an offline random number generator uses the ID of a tag as seed to produce a string of pseudo-random bits, which are stored in the tag. The bits form a logical ring. After deployment, the tag generates a hash value $H(id, s)$ by returning a certain number of bits after the sth bit in the ring, where id is the tag ID and s is a given *hash seed* that can alter the hash output. This hash output is predictable by a RFID reader that knows the tag ID and the seed s.

More sophisticated hash implementations can be designed based on a ring of pseudo-random bits. For example, we may interpret s as a concatenation of a flag x and two random numbers, r_1 and r_2. To produce a hash output, we go clockwise along the ring if $x = 0$ or counterclockwise if $x = 1$. We then output the r_1th bit on the ring, and then output one more bit after every r_2 bits on the ring. If the hash output is required to be in a range $[0, y)$, we first take a sufficient number of hash bits as described above and then perform modulo y.

This hash function is easy to implement in hardware and thus suitable for tags. But it can only produce a limited number of different hash values, depending on the size of the ring. It is not suitable for a protocol whose operations require each tag to produce a large number of different hash values, but it works well for a protocol that only requires each tag to produce a few independent hash values.

3.3 Basic Idea

We know in Section 3.1 that random mapping from tags to slots generates a limited number of singleton slots. An arbitrary slot only has a probability of up to 36.8% to be a singleton. Now, if we 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 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, ..., s_k, to tags. Each seed s_i corresponds to a different mapping where a tag is mapped to a slot indexed by $H(id, s_i)$.

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. 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 4. 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.

3.4 Segmentation

The seed-selection vector has f selectors, each of which is $\lceil \log(k+1) \rceil$ 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 $\frac{96}{\lceil \log(k+1) \rceil}$ selectors. Let $l = \frac{96}{\lceil \log(k+1) \rceil}$. 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 sub-frames, each containing l slots accordingly. The jth time sub-frame is denoted as F_j. This allows our protocol to deal with one sub-frame at a time.

Our protocol consists of two phases. In Phase one, the reader identifies the set of sampled tags, and randomly assigns the sample tags to the sub-frames. The subset of sampled tags that are assigned to the jth sub-frame is denoted as N_j. For each sub-frame 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.

In Phase two, 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 sub-frame 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 sub-frame, which conserves energy.

Below we give details of the protocol design.

3.5 Phase One: Finding Sampled Tags

To implement a sampling probability p, the reader will broadcast an integer $x = \lceil p \times X \rceil$ and a prime number q, where X is a large, pre-configured constant (e.g., 2^{16}). During the ith round of protocol execution, a tag is sampled if and only if the hash result $H(id, q \times i)$, which is a pseudo-random number in the range of $[0, X)$, is smaller than x, where id is the tag's ID.

After receiving x and q, each tag can predict which rounds of protocol execution it will participate. Since the protocol is scheduled to execute periodically with pre-defined 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.

3.6 Phase One: Assigning Sampled Tags to Subframes

When assigning sampled tags to time sub-frames, the reader selects an additional random seed s, which is different from $s_1, ..., s_k$. For each sampled tag, the reader produces a hash output $H(id, s)$ and assigns the tag to the $H(id, s)$th sub-frame, 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 which sub-frame it is assigned to, after it receives s in the detection request broadcast by the reader at the beginning of Phase two.

3.7 Phase One: Determining Seed-selection Segments

Each seed-selection segment is determined independently. Consider the jth segment V_j, the jth time sub-frame F_j, and the set N_j of sampled tags that are assigned to F_j. All selectors in V_j are initialized to zeros. The reader begins by using the first seed s_1 to map tags in N_j to slots in F_j. 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. 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. 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 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. 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,..., s_k, each tag will know whether it is assigned to a slot. If not, from the received s', it will know which unused slot it is assigned to.

3.8 Phase Two

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, ..., s_k, and s'. The time frame is divided into sub-frames. Before each sub-frame 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 sub-frame, 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 sub-frame has 32 slots.

Consider an arbitrary tag t. It wakes up to participate in a scheduled protocol execution that it is sampled for. After t receives the detection request from the reader, it uses $H(id, s)$ to determine which sub-frame it is assigned to. Without loss of generality, let the sub-frame be F_j. The tag sets a 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 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 3.7, 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) which sub-frame each sampled tag is assigned to, (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 empty, 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.

4. ENERGY-TIME TRADEOFF IN PROTOCOL CONFIGURATION

We formally derive the detection probability and the energy-time tradeoff curve. We show how to compute system parameters and how to solve the constrained least-time (or least-energy) problem.

4.1 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 sub-frame as t does. n' follows a binomial distribution, $Bino(n, p\frac{l}{f})$, i.e.,

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

$P_0 = 0$. 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 below.

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

$$p' = (1 - (1 - P_{i-1})\frac{1}{l})^{n'-1} \cdot (1 - \frac{n'P_{i-1}}{l}). \qquad (2)$$

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

$$P_i = P_{i-1} + (1 - P_{i-1}) \cdot \sum_{j=0}^{n} Prob\{n' = j\} \cdot p'$$
$$= P_{i-1} + (1 - P_{i-1}) \cdot \sum_{j=0}^{n} \binom{n}{j} (p\frac{l}{f})^j (1 - p\frac{l}{f})^{n-j} \cdot \quad (3)$$
$$(1 - (1 - P_{i-1})\frac{1}{l})^{j-1} \cdot (1 - \frac{jP_{i-1}}{l}),$$

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 empty slot instead, resulting in the detection. Therefore, the detection probability of MSMD, denoted as $P_{msmd}(p, f)$, is

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

The value of $P_{msmd}(p, f)$ 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 pre-known, 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.

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

$$P_{emd}(p, f) = 1 - (1 - p \times P_1)^m. \quad (5)$$

TRP [6] is a special case of EDM with $p = 1$. Namely, sampling is turned off.

4.2 Energy-time Tradeoff Curve

The protocol's energy cost can be characterized by the sampling probability p because the expected number of tags that participate in each protocol execution is $p \times n$. A smaller value of p means a smaller number of participating tags, which in turn means a smaller energy cost. The protocol's execution time can be characterized by the frame size f. The total number of slots for tags to respond is f, and the total size of the seed-selection vector is also f. A smaller value of f generally means a shorter protocol execution time. The time for the reader to broadcast the detection request is a constant, which is negligible when comparing with the time frame and the seed-selection vector if f is large. The actual protocol execution time, measured in seconds, will be studied in the next section based on the Philips I-code specification.

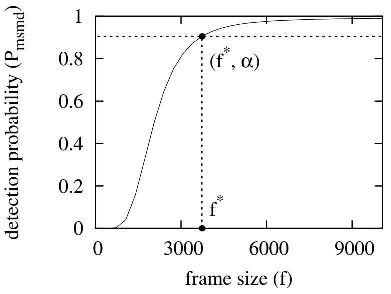

Figure 1: Detection probability $P_{msmd}(p, f)$ with respect to the frame size f when $n = 50,000$, $m = 100$, $k = 3$, and $p = 5\%$.

We cannot arbitrarily pick small values for p and f. They must satisfy the requirement $P_{msmd}(p, f) \geq \alpha$. 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.

If we fix the value of p, $P_{msmd}(p, f)$ becomes a function of f. The solid line in Fig. 1 shows an example of the curve $P_{msmd}(p, f)$ with respect to f when $n = 50,000$, $m = 100$, $k = 3$, and $p = 5\%$. Because $P_{msmd}(p, f)$ is an increasing function, the minimum value of f that satisfies the requirement $P_{msmd}(p, f) \geq \alpha$ can be found by solving the following equation,

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

The solution is denoted as f^*. See Fig. 1 for illustration.

For each different sampling probability p, we can compute the smallest usable frame size f^* that satisfies $P_{msmd}(p, f) \geq \alpha$. 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^* below.

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

The algorithm that computes $f^*(p)$ based on bi-section search is given in Algorithm 1.

Algorithm 1 Search for $f^*(p)$

```
INPUT: $n, m, \alpha, k, p$
OUTPUT: frame size that minimizes execution time under sampling probability $p$

if $P_{msmd}(p, U) < \alpha$ then exit; \\requirement cannot be met
$f_0 = 1, f_1 = U$;
while $f_1 - f_0 > 1$ do
    $f_2 = \lceil \frac{f_0+f_1}{2} \rceil$;
    if $P_{msmd}(p, f_2) < \alpha$ then $f_0 = f_2$ else $f_1 = f_2$;
end while
return $f_1$;
```

The left plot in Fig. 2 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 $n \times p$ 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.

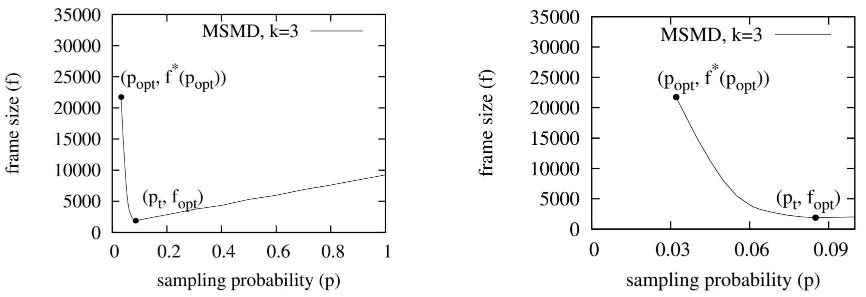

Figure 2: • *Left plot:* Energy-time tradeoff curve, i.e., frame size $f^*(p)$ with respect to sampling probability p, when $n = 50,000$, $m = 75$, $k = 3$, and $\alpha = 95\%$. • *Right plot:* Energy-time tradeoff curve in the range $p \in [p_{opt}, p_t]$.

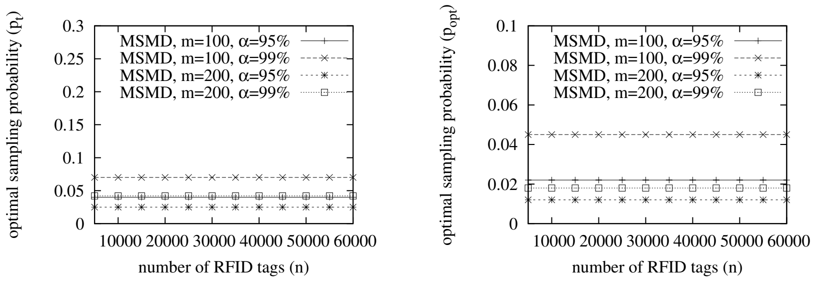

Figure 3: • *Left Plot*: the value of p_t with respect to n. • *Right Plot*: the value of p_{opt} with respect to n.

Algorithm 2 Search for p_{opt}

```
INPUT: $n, m, \alpha, k$
OUTPUT: optimal sampling probability that minimizes energy cost

$p_0 = 0, p_1 = 1, \delta = 0.01, U$ = 1,000,000; \\400 seconds
while $p_1 - p_0 > \delta$ do
    $p_2 = \lceil \frac{p_0+p_1}{2} \rceil$;
    if $P_{msmd}(p_2, U) < \alpha$ then $p_0 = p_2$ else $p_1 = p_2$;
end while
return $p_1$;
```

Algorithm 3 Search for f_{opt}

```
INPUT: $n, m, \alpha$
OUTPUT: optimal frame size and sampling probability that minimize execution time

$p_0 = p_{opt}, p_1 = 1, \delta = 0.01$;
while $p_1 - p_0 > \delta$ do
    $p_2 = \lceil \frac{p_0+p_1}{2} \rceil$;
    if $f^*(p_2) > f^*(p_2 + \frac{\delta}{2})$ then $p_0 = p_2$ else $p_1 = p_2$;
end while
return $f^*(p_1)$ and $p_1$;
```

4.3 Minimum Energy Cost

When the sampling probability p is too small, the detection probability $P_{msmd}(p, f)$ will be smaller than α for any value of f. Such a small sampling probability cannot be used. We design a bi-section search method in Algorithm 2 to find the smallest value of p, denoted as p_{opt}, which can satisfy $P_{msmd}(p, f) \geq \alpha$ with a frame size no greater than the upper bound U. The sampling probability returned by the algorithm is within an error of δ from the true optimal value p_{opt}, where δ is a parameter that can be set arbitrarily small. When p_{opt} and $f^*(p_{opt})$ are used, the energy cost is minimized.

4.4 Minimum Execution Time

From the energy-time tradeoff curve (the left plot in Fig. 2), we can find the smallest value of $f^*(p)$, denoted as f_{opt}, that minimizes the execution time.

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

Let p_t be the corresponding sampling probability. Namely, $P_{msmd}(p_t, f_{opt}) = \alpha$. The values of f_{opt} and p_t are determined through bi-section search in Algorithm 3, where the **if** statement uses the local gradient to guide the search direction towards the minimum. When p_t and f_{opt} are used, the protocol execution time is minimized.

We amplify the segment of the energy-time tradeoff curve between point (p_{opt}, $f^*(p_{opt})$) and point (p_t, f_{opt}) in the right plot of Fig. 2. When we increase the value of p from p_{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.

4.5 Offline Computation

Because the computation of p_{opt}, $f^*(p_{opt})$, p_t, and f_{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 pre-configured as part of the system requirement. The value of k is determined before tag deployment. Hence, we can pre-compute p_{opt}, $f^*(p_{opt})$, p_t, and f_{opt} in a table format with respect to different values of n (for instance, from 100 to 100,000 with an increment step of 100), 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_{opt} remain largely constants, as shown in Fig. 3. 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_{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_{opt} or a value between them. It then downloads p to all tags and synchronizes their clocks. For the rest of the hour, the reader does not have to transmit the sampling probability again.

4.6 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 4.2: Starting from (4), if we fix $f = u'$, $P_{msmd}(p, f)$ becomes a function of p. We can use bi-section search to find p that meets $P_{msmd}(p, f) = \alpha \wedge p \leq p_t$.

4.7 Impact of k

We study how the number k of hash seeds will affect the protocol's performance. Figure 4 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.[4]

In Figure 4, 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.

[4]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.

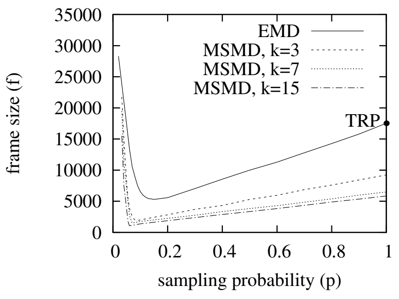

Figure 4: Energy-time tradeoff curves of EMD and MSMD under different k values, when $n = 50,000$, $m = 100$, and $p = 5\%$.

5. NUMERICAL RESULTS

We have performed extensive simulations to study the performance of the proposed MSMD, and compare it with EMD [16] and TRP [6]. The simulation setting is based on the Philips I-Code specification [20]. Any two consecutive transmissions (from the reader to tags or vice versa) are separated by a waiting time of 302 μs. The transmission rate from the reader to tags is 26.5 Kb/sec; it takes 37.76 μs for the reader to transmit one bit. A 96-bit slot that carriers a seed-selection segment is 3927 μs long, which includes a waiting time before the transmission. The transmission rate from a tag to the reader is 53Kb/sec; it takes 18.88 μs for a tag to transmit one bit. A single-bit slot T_{short} for a tag to respond (i.e., make the channel busy) is 321 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 $f \times T_{short}$ plus the time for broadcasting a detection request. The energy cost of TRP is measured as n participating tags. MSMD 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. The energy cost of these two protocols is measured as $n \times p$ participating tags. For EMD, its execution time is $f^* \times T_{short}$, plus the time for a request. For MSMD, we need to add the time slots for the selection vector.

The design of all three protocols ensures that the detection requirement specified by m and α is always met. This is indeed what we observe in our simulations. Our comparison below is made in terms of energy efficiency and time efficiency, given a certain detection requirement.

5.1 Energy-time Tradeoff

Let $n = 50,000$, $\alpha = 95\%$, and $m = 50$. Fig. 5 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

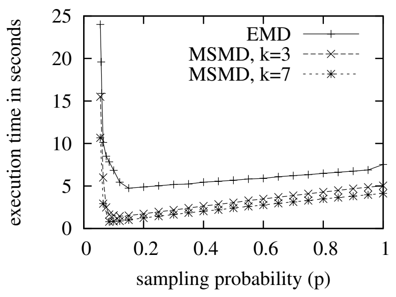

Figure 5: Protocol execution time with respect to sampling probability, when $\alpha = 95\%$, $m = 50$, and $n = 50,000$.

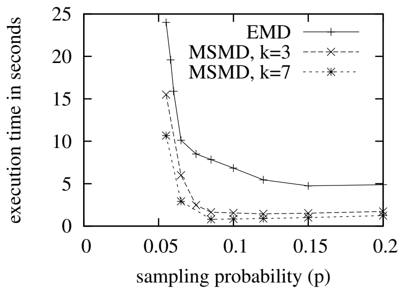

Figure 6: Zoom-in view of energy-time tradeoff in Figure 5 in the sampling probability range of [0, 0.2].

due to overly large overhead for the selection vector. In Fig. 6, 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 seconds, the number of participating tags in MSMD with $k = 7$ is 46.7% of the number in EMD. We vary the values of n, α and m. Similar conclusions can be drawn from the simulation results, which are omitted due to space limitation.

The tradeoff curves in Fig. 6 agree with our analytical results in Fig. 4 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.

In Table 1, 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 $p_{opt} = 0.055$. 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.

5.2 Performance Comparison

Next, we compare the performance of MSMD ($k = 7$), EMD, and TRP under different values of m, α and n. MSMD and EMD are operated with their optimal sampling probabilities p_t. In Fig. 7-9, we keep $m = 50$ and vary the value of α. In Fig. 7, 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 three 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. In the meanwhile, MSMD also has a much smaller execution time than EMD and TRP. Similar results can be drawn from Fig. 8 where $\alpha = 99\%$ and Fig. 9 where $\alpha = 90\%$. In the latter case, the execution time of MSMD is less than a second.

In Fig. 10-11, we keep $\alpha = 99\%$ and vary the value of m. In Fig. 10, $m = 25$. In Fig. 11, $m = 100$. The performance of MSMD remains the best among all three.

Table 1: 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%

6. 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 degree of freedom for tags to select time slots when they announce their presence to the RFID reader in the process of missing-tag detection. The result is a multi-fold cut in both energy cost and execution time. Such performance improvement is critical for a protocol that needs to be executed frequently. 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.

7. REFERENCES

[1] T. Kriplean, E. Welbourne, N. Khoussainova, V. Rastogi, M. Balazinska, G. Borriello, T. Kohno, and D. Suciu, “Physical Access Control for Captured RFID Data,” *IEEE Pervasive Computing*, 2007.

[2] Y. Liu, L. Chen, J. Pei, Q. Chen, and Y. Zhao, “Mining Frequent Trajectory Patterns for Activity Monitoring Using Radio Frequency Tag Arrays,” *Proc. of IEEE PerCom*, 2007.

[3] L. M. Ni, Y. Liu, Y. C. Lau, and A. Patil, “LANDMARC: Indoor Location Sensing Using Active RFID,” *ACM Wireless Networks (WINET)*, vol. 10, no. 6, November 2004.

[4] Y. Li and X. Ding, “Protecting RFID Communications in Supply Chains,” *Proc. of ASIACCS*, 2007.

[5] B. Sheng, C. Tan, Q. Li, and W. Mao, “Finding Popular Categories for RFID Tags,” *Proc. of ACM Mobihoc*, 2008.

[6] C. Tan, B. Sheng, and Q. Li, “How to Monitor for Missing RFID Tags,” *Proc. of IEEE ICDCS*, 2008.

[7] A. Nemmaluri, M. Corner, and P. Shenoy, “Sherlock: Automatically Locating Objects for Humans,” *Proc. of ACM MobiSys*, 2008.

[8] L. Ravindranath, V. Padmanabhan, and P. Agrawal, “Sixthsense: RFID-based Enterprise Intelligence,” *Proc. of ACM MobiSys*, 2008.

[9] T. Li, S. Wu, S. Chen, and M. Yang, “Energy Efficient Algorithms for the RFID Estimation Problem,” *Proc. of IEEE INFOCOM*, March 2010.

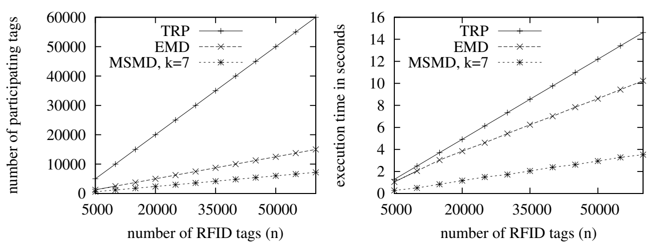

Figure 7: • *Left plot:* The number of participating tags with respect to the number of tags, when $m = 50$ and $\alpha = 99.9\%$. • *Right plot:* The protocol execution time with respect to the number of tags, when $m = 50$ and $\alpha = 99.9\%$.

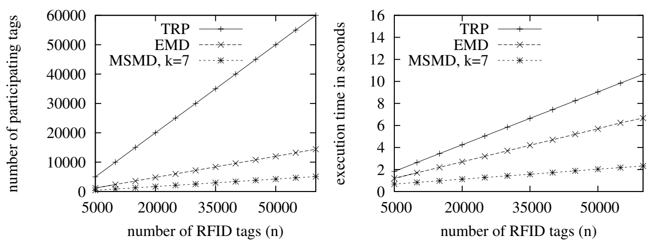

Figure 8: Same as the caption of Fig. 7 except for $\alpha = 99\%$.

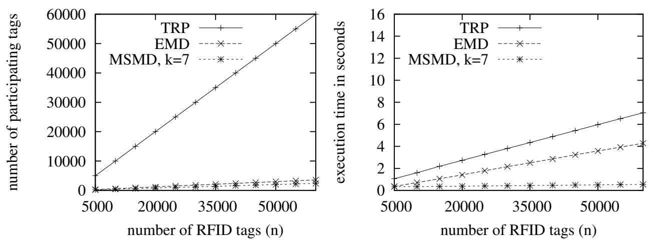

Figure 9: Same as the caption of Fig. 7 except for $\alpha = 90\%$.

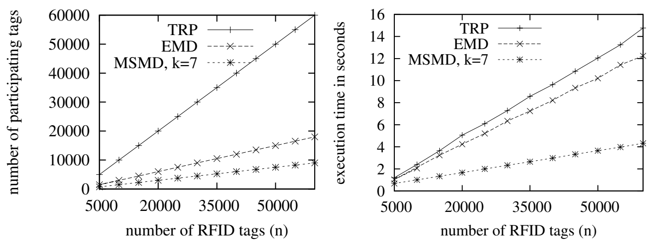

Figure 10: • *Left plot:* The number of participating tags with respect to the number of tags, when $m = 25$ and $\alpha = 99\%$. • *Right plot:* The protocol execution time with respect to the number of tags, when $m = 25$ and $\alpha = 99\%$.

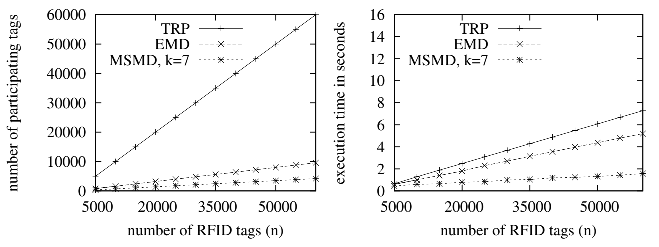

Figure 11: Same as the caption of Fig. 10 except for $m = 100$.

[10] S. Chen, M. Zhang, and B. Xiao, "Efficient Information Collection Protocols for Sensor-augmented RFID Networks," *Proc. of IEEE INFOCOM*, April 2011.

[11] Yan Qiao, Shigang Chen, Tao Li, and Shiping Chen, "Energy-efficient Polling Protocols in RFID Systems," *Proc. of ACM MobiHoc*, May 2011.

[12] J. Wolverton, "Wal-Mart to Embed RFID Tags in Clothing Beginning August 1," *http://www.thenewamerican.com/index.php/tech-mainmenu-30/computers/4157-wal-mart-to-embed-rfid-tags-in-clothing-beginning-august-1*, July 2010.

[13] T. Li, S. Chen, and Y. Ling, "Identifying the Missing Tags in a Large RFID System," *Proc. of ACM Mobihoc*, 2010.

[14] B. Sheng, Q. Li, and W. Mao, "Efficient Continuous Scanning in RFID Systems," *Proc. of IEEE INFOCOM*, 2010.

[15] R. Zhang, Y. Liu, Y. Zhang, and J. Sun, "Fast Identification of the Missing Tags in a Large RFID System," *Proc. of the 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2011.

[16] W. Luo, S. Chen, T. Li, and S. Chen, "Efficient Missing Tag Detection in RFID Systems," *Proc. of IEEE INFOCOM, mini-conference*, 2011.

[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," *Proc. of the 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2009.

[18] P. Popovski, K. F. Nielsen, R. M. Jacobsen, and T. Larsen, "Robust Statistical Methods for Detection of Missing RFID Tags," *IEEE Communications Magazine, Wireless Communications Series*, 2010.

[19] A. Juels, "RFID Security and Privacy: A Research Survey," *IEEE Journal on Selected Areas in Communications (JASC)*, vol. 24, no. 2, pp. 381–394, February 2006.

[20] Philips Semiconductors, "Your Supplier Guide to ICODE Smart Label Solutions," *http://www.nxp.com/acrobat_download2/other/icode_supplier_list_2008_10.pdf*, October 2008.

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

[22] N. Bhandari, A. Sahoo, and S. Iyer, "Intelligent Query Tree (IQT) Protocol to Improve RFID Tag Read Efficiency," *Proc. of IEEE International Conference on Information Technology (ICIT)*, December 2006.

[23] V. Sarangan, M. R. Devarapalli, and S. Radhakrishnan, "A Framework for Fast RFID Tag Reading in Static and Mobile Environments," *International Journal of Computer and Telecommunications Networking*, vol. 52, no. 5, pp. 1058–1073, 2008.

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