Efficient Information Sampling in Multi-Category RFID Systems

Jia Liu, *Member, IEEE, ACM*, Shigang Chen, *Fellow, IEEE*, Qingjun Xiao, *Member, IEEE, ACM*, Min Chen, *Member, IEEE*, Bin Xiao, *Senior Member, IEEE*, and Lijun Chen

***Abstract*— In RFID-enabled applications, when a tag is put into use and associated with a specific object, the category-related information (e.g., the brands of clothes) about this object might be preloaded into the tag's memory for the purpose of live query. Since such information reflects category attributes, all tags in the same category carry identical category information. To collect this information, we do not need to repeatedly interrogate each tag; one tag's response in a category is sufficient. In this paper, we investigate the problem of category information collection in a multi-category RFID system, which is referred to as *information sampling*. We propose two time-efficiency protocols. The first is a two-phase sampling protocol (TPS) that works in the case of knowing tag IDs. By quickly zooming into a category and isolating a tag from this category, TPS is able to sample a category with small overhead. The second protocol, called back-and-forth sampling protocol (BFS), relaxes a key assumption in TPS and performs the sampling task efficiently without knowing any tag IDs or category IDs. By carrying out a step-forward frame and using the step-backward scheme, BFS is able to interrogate only 1.45 tags (close to the lower bound of one tag) on average for each category. We theoretically analyze the protocol performance of TPS and BFS and discuss the optimal parameter settings that minimize the overall execution time. Extensive simulations show that both the protocols outperform the benchmark, greatly improving the sampling performance.**

***Index Terms*— RFID, category information, sampling, polling, time efficiency.**

Manuscript received December 7, 2017; revised July 7, 2018; accepted November 14, 2018; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor M. Li. This work was supported in part by the National Natural Science Foundation of China under Grant 61702257 and Grant 61771236; in part by the Natural Science Foundation of Jiangsu Province under Grant BK20170648; in part by the Jiangsu Key R&D Plan (Industry Foresight and Common Key Technology) under Grant BE2017154; in part by the Project funded by China Postdoctoral Science Foundation; in part by the National Science Foundation under Grant CNS-1718708; in part by the Fundamental Research Funds for the Central Universities; and in part by the Collaborative Innovation Center of Novel Software Technology and Industrialization. *(Corresponding authors: Jia Liu, Lijun Chen.)*

J. Liu and L. Chen are with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China (e-mail: jialiu@nju.edu.cn; chenlj@nju.edu.cn).

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

Q. Xiao is with the School of Computer Science and Engineering, Southeast University, Nanjing 211189, China (e-mail: csqjxiao@seu.edu.cn).

M. Chen is with Google Inc., Mountain View, CA 94043 USA (e-mail: minchen@google.com).

B. Xiao is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong (e-mail: csbxiao@comp.polyu.edu.hk).

Digital Object Identifier 10.1109/TNET.2018.2883508

I. INTRODUCTION

RADIO frequency identification (RFID) is becoming ubiquitously available in a variety of applications, including library inventory [1]–[3], warehouse control [4]–[17], supply chain management [18]–[20], object tracking [21]–[30], etc. Among these applications, RFID tags are usually attached to objects that belong to different categories, e.g., subjects of books in a library, types of medicine in a pharmacy, or brands of clothes in a clothing outlet. When a tag is associated with a specific object, the category-related information about this object[1] can be preloaded into the tag's memory for the purpose of off-line query by RFID readers. Since this information reflects the category attributes, each tag in the same category carries identical category-level information.

To collect such category information in a multi-category RFID system, we do not need to repeatedly interrogate each tag. One tag's response in a category will suffice. For example, to know the manufacturer of Horizon Organic milk stocked in a warehouse, we just need to query one milk box instead of all of them as they are produced by the same manufacturer. In another example, consider a chilled food storage chamber, where each food is affixed with a sensor-augmented RFID tag (e.g., WISP [31]) equipped with a thermal sensor. The reader periodically samples temperature readings from tags to check whether any area goes beyond the normal temperature. Since tagged objects of the same category (i.e., tags with the same category IDs) are typically packed together or placed closely, the temperature reports from these nearby tags lead to high data redundancy. Hence, it is a waste to collect sensor information from all tags in this case.

In this paper, we study the problem of category information collection in a multi-category RFID system, which is referred to as *information sampling*. The existing data collection protocols [32]–[34] either collect all tags' information or take the entire tag set into account each time when isolating an interested tag from others, which is time-consuming. These solutions are not suitable for the problem of information sampling, which has two unique features: (i) Since tags in the same category carry identical category-related information, we do not need to query each individual tag; one tag's response from each category is sufficient to report the information.

[1] In the previous examples, the category-related information is the subject of a book, the type of a medicine, or the brand of clothes.

(ii) We do not care which tag in a category responds to the reader; anyone in the category can be a candidate for reporting.

By considering the above two features, we propose a two-phase sampling protocol (TPS) that works under the case of knowing tag IDs. In the first phase, the reader isolates one category from others, which helps us quickly zoom into a category from the entire tag set. In the second phase, the reader selects a single tag from the isolated category by using geometrically distributed tag indices. When efficiently done, these two phases make TPS far superior to the existing solutions. We then relax a key assumption with a new back-and-forth sampling protocol (BFS) that achieves the sampling task without any prior knowledge of tag IDs or category IDs. By carrying out a step-forward frame and using the step-backward scheme, BFS is able to interrogate only a few tags in each category and silence others to save communication overhead. With this protocol, only 1.45 tags (close to the lower bound of one tag) for each category are interrogated. We analyze the performance of TPS and BFS, and provide the optimal parameter settings that minimize the overall execution time. Extensive simulations demonstrate that both of the proposed protocols outperform the existing solutions, greatly improving the sampling efficiency.

The rest of the paper is organized as follows. Section II formulates the sampling problem. Section III proposes a two-phase sampling protocol. Section IV presents a back-and-forth sampling protocol. Section V evaluates the performance of the proposed protocols. Section VI discusses the related work. Finally, Section VII concludes this paper.

II. Problem Statement

A. System Model

We consider an RFID system that consists of a reader and a number of tags. Each tag has a unique ID that is used to identify the object the tag is attached to. The tag ID contains two components: *category ID* indicating which category the tag belongs to, and *member ID* identifying a specific member in this category. Tags with the same category ID share common information, which may be static category-related information (such as the brand of the tagged products) that is preloaded for live query by a reader after the tag is put into use, or dynamic information (such as sensor data) that is written to or measured by the tags. We refer to this information as *category information*. Besides, we assert that the collection of dynamic information is built on the assumption that tags from the same category are packed together or placed close, for example, in a warehouse. In this case, if we need a real-time check on their conditions such as temperature, reporting one from each tag becomes unnecessary because they share similar condition due to proximity. Instead, any one tag from a nearby tag set (the same category) is sufficient to return the information.

B. Problem Definition

Let $\mathcal{N}$ be the tag set in the RFID system, where $n = |\mathcal{N}|$. According to category IDs, $\mathcal{N}$ is partitioned into a family of disjoint sets $\mathcal{C} = \{C_1, C_2, ..., C_m\}$, such that $\bigcup_{i=1}^{m} C_i = \mathcal{N}$.

TABLE I
Key Notations

Symbols	Descriptions
$\mathcal{N}$	the tag set in the RFID system
n	the number of tags in $\mathcal{N}$
$\mathcal{C}$	the set of all categories (a partition of $\mathcal{N}$)
C_i	the ith category in $\mathcal{C}$
m	the number of categories in $\mathcal{C}$
$\lvert\cdot\rvert$	the cardinality of a set
f	the frame size in TPS
r	the random seed
$H(\cdot)$	a hash function shared by all tags
V	an ordering vector in the TPS protocol
K	the length of the index in the binary representation
Q	the slot index picked by the tag in BFS
L	the last busy slot in the step-forward phase of BFS
$\mathcal{R}(\cdot)$	the position of the right-most bit of '1' in binary representation of the input

For convenience, we use C_i to represent the category ID as well as the set of tags in this category. There are $m = |\mathcal{C}|$ categories in total; each tag in $\mathcal{N}$ belongs to one of them. The problem of *information sampling in a multi-category RFID system* is to select (or sample) one or a subset of tags from each category to report their data with the objective of minimizing the overall time for collecting category-level information from all categories. Since the category-level information carried by all tags in each category is identical, it is not necessary to ask all tags to report their data. Ideally, the reader should single out one tag from each category to report information. The selection process is however tricky in an RFID system, particularly when the reader does not even know which categories are currently in the system and which tags are in each category.

This paper considers two cases when designing its solutions for the category-level information sampling problem. The first case is that the tag set $\mathcal{N}$ is known a priori. Consider an RFID system deployed in a storage facility, where a reader is installed at the entrance (or exit) to keep track of the IDs of the tags (thus the associated objects) that are moved in and out. This setting allows the back-end server (to which the reader is connected) to maintain an updated list of all tag IDs in the storage. In the second case, we remove the above assumption and address the sampling problem without any pre-knowledge of tag IDs or category IDs. This is also common in practice. For example, consider an RFID system where products are packed (say, in boxes) when they are moved into the storage. The reader at the entrance may not be able to penetrate the whole packs and therefore may miss some tag IDs. Therefore, we may have some unknown tag IDs in the storage after the products are moved in and possibly unpacked and rearranged. The key notations are given below.

III. Two-Phase Sampling Protocol

In this section, we consider information sampling with a known set $\mathcal{N}$ of tags. We first explain two possible solutions to this problem and then detail our design of a two-phase sampling protocol.

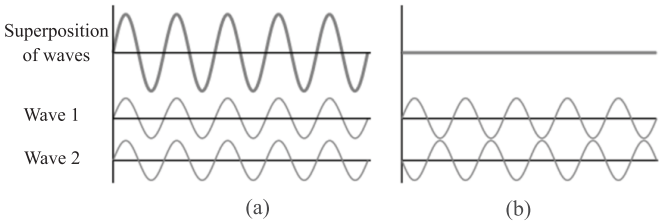


Fig. 1. An illustration of wave interference. (a) Two waves in phase. (b) Two waves out of phase.

A. Possible Solutions

1) Basic Polling and ETOP: One solution is to randomly select m tags (denoted by M), one from each category, and have the reader poll these tags for their category-level information. The basic polling protocol transmits the tag IDs one after another while all tags listen for their IDs and a tag will transmit its information right after its ID is heard. It can be inefficient in a low-rate RFID channel to transmit a large number of IDs (96 bits each) [34]. A more sophisticated polling protocol is ETOP [34], in which the reader broadcasts a partitioned Bloom filter to solicit tag responses in a time efficient way. However, this approach will not work if the reader does not know the categories or their tags in the systems (the case we will consider in the next section). Even if the reader knows all tags, the design of ETOP requires the reader to transmit a segmented/partitioned Bloom filter to filter out tags that are not selected for reporting, which can be costly: When there are a large number of tags in the system, in order to filter out most non-selected tags, we need to drive down the filter's false positive ratio, which means larger filter size. Moreover, tags that cause false positives must be handled through a separate polling process. Our simulations show that a protocol that is specifically designed for the category-level information sampling problem will greatly outperform ETOP.

2) Wave Superpose: Because tags in the same category will report the same information, one idea is to let them transmit simultaneously, one category at a time, allowing the same signals from these tags to superpose. If the RF waves emitted from tags in the same category are superposed with positive interference, the RFID reader may be able to decode the category-level data correctly. Unfortunately, this idea does not work for two reasons. First, two wave superpose is likely to form a resultant wave of greater, lower, or the same amplitude, which cannot guarantee a positive interference all the time. Fig. 1(a) shows perfectly positive interference when the phase difference between two waves is an even multiple of π. However, perfectly negative interference will occur when the phase difference is an odd multiple of π, as shown in Fig. 1(b), making it impossible to decode data. In general, when the phases of numerous tags are not aligned well, the resulting superposed signals can take arbitrary form, making decoding unpredictable.

The other reason is clock difference. Due to clock rate differences, the tags' clocks will differ after some amount of time due to clock drift. This clock skew may result in the misalignment of data in bit level. To verify this conclusion, we conduct an experiment with wireless identification sensing platform (WISP) and universal software radio peripheral (USRP), where on-off keying modulation at the physical layer is adopted. Fig. 2 shows the signal level results from one set of experiments. The fourth plot is the mixed signals from three WISP tags that transmit the same 20-bit data '1001 0100 1010 1100 1100' to the reader at 40 kbps (which refers to the reader's clock). For comparison, we also show their individual signals in the first three plots. As we can see, it is hard to decode the mixed signals due to the clock drift caused by different tags. Taking the negative interference and clock difference into account, we stress that the scheme of wave superpose is not a good solution to the sampling problem.

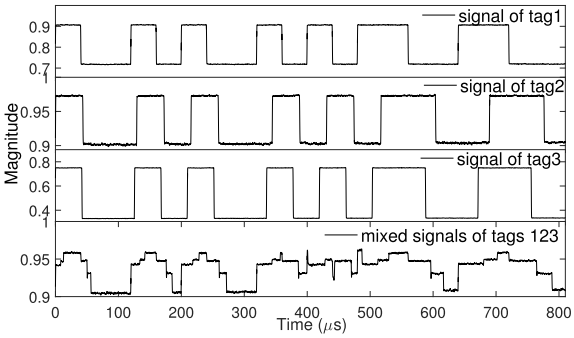


Fig. 2. Signal superpose of three WISP tags with clock difference.

In this section, we take a deeper look at the two features of the information sampling problem and propose a two-phase solution: 1) separating a category from others, and 2) singling out one tag from each category to report. The first step helps us quickly zoom into a single category from the entire tag set. The second step uses only a 4-bit index to select a single tag from each category to transmit, free of collision. In combination, these two steps allow our protocol to significantly outperform the sophisticated polling protocol ETOP.

B. Protocol Description

TPS is performed in a number of sampling rounds, each consisting of two phases: an *ordering phase* and a *polling phase*. The expected number of rounds is about 3.5, as we will analyze later. In each round, the reader collects information from a subset of categories, and tags in those categories are removed from the subsequent rounds. The protocol terminates after information from all categories is collected.

Our idea is for the reader to use a virtual time frame in each round to select a subset of categories for information reporting. The use of a virtual frame (which is not actually carried out in reader-tag communication) greatly reduces the overhead for the reader to inform the selected tags of when to transmit, and also inform other tags to stay silent. This is achieved in two phases: The order phase will silence all tags from the categories that are not selected in this round, while the polling phase will work with one selected category at a time and the reader transmits an index value to select a single tag from that category to report information.

More specifically, during the ordering phase, the reader plays out a frame of time slots *virtually* by randomly assigning the categories to the slots and identifying the slots that have

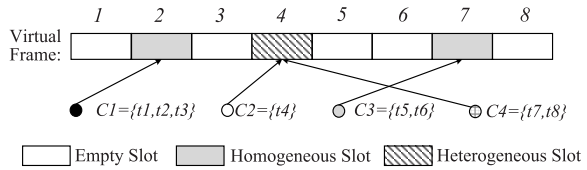


Fig. 3. Three kinds of slots in the virtual frame.

only one category assigned to — such a category is called a *homogeneous category*. For each homogeneous category, the reader assigns an index value from a geometric distribution to every tag in the category, and checks whether there exists a *singleton index* value that is assigned to one and only one tag; if so, this tag will be selected to transmit the category-level information, and the corresponding category will be active in the polling phase. At the end of the ordering phase, the reader will inform (by broadcasting an ordering vector to all tags) about which categories will be active in the polling phase and in which order they will report their information.

In the polling phase, only the slots in the frame with active categories will be actually played out. In each of these slots, only tags in the category assigned to the slot will participate. The reader starts a slot by transmitting the singleton index of the category that it finds in the ordering phase. This index selects one and only one tag from the category to transmit, free of collision. All other tags in the category will stay silent. The active categories in the current round, with their information being reported to the reader, will stay idle in all subsequent rounds. Below we describe the details of the protocol.

1) Ordering Phase: The reader chooses two parameters, the number f of time slots in the virtual frame and a random seed r. We will discuss how to choose the optimal value of f later. The reader then assigns every category cid that has not reported its information in the previous rounds to a slot through hashing $H(cid, r) \mod f$, where $H(\cdot)$ is a hash function shared by all tags.

Slots with a single assigned category are called *homogeneous slots*, slots chosen by multiple categories are called *heterogeneous slots*, and slots chosen by no category are called *empty slots*. Fig. 3 shows an example where there are three categories and tags in the same category will always hash to the same slot. The second slot and the seventh slot are homogeneous because each of them has a single category, C_1 to the second slot and C_3 to the seventh slot. In contrast, the fourth slot is heterogeneous because two categories, C_2 and C_4, are assigned to the slot. The remaining slots are empty. We are interested in the number of homogeneous slots, which is affected by the value of f and the number of categories, not by the number of tags since all tags of each category are assigned to the same slot.

The categories hashed to homogeneous slots are called *homogeneous categories* and their tags are called *homogeneous tags*. The reader knows all category IDs and tag IDs. It performs the hash to find all homogeneous slots and the corresponding categories. For each homogeneous category C_i, the reader further selects a single tag as follows: To each tag in C_i, it performs another hash to assign an index value, $\mathcal{R}(H(id, r) \mod 2^K)$, where id is the tag's ID, r is the hash seed introduced previously, K is the length of the value in number of bits, and $\mathcal{R}(\cdot)$ is a function that returns the index of the right-most bit of 1 in the binary representation of the input. For example, $\mathcal{R}(81_{10}) = \mathcal{R}(01010001_2) = 1$ and $\mathcal{R}(104_{10}) = \mathcal{R}(01101000_2) = 4$. Clearly, $H(id, r) \mod 2^K$ is a K-bit binary number, and the range of the index value is $[1, K]$. The index value follows a geometrical distribution: The probability for its value to be 1 is 50% because the chance for the rightmost bit in $H(id, r) \mod 2^K$ to be 1 is 50%. The probability for the index value to be 2 is 25% because the chance for the rightmost two bits in $H(id, r) \mod 2^K$ to be 10 is 25%. In general, the probability for the index value to be j is $\frac{1}{2^j}$. The number of tags having a certain index value decreases exponentially as the index value increases.

TABLE II
AN ILLUSTRATION OF RESOLVABLE CATEGORIES

	Tag	Hash Result				Index	Resolvable
C1	t1	0	0	1	0	2	√
	t2	0	1	0	1	1	
	t3	1	0	0	1	1	
C2	t4	0	1	0	1	1	√
C3	t5	1	0	1	0	2	×
	t6	0	1	1	0	2	
C4	t7	1	0	1	0	2	√
	t8	0	1	0	0	3	

We call an index value that is assigned to exactly one tag as a *singleton index*. Intuitively, the index value around $\log_2(|C_i|)$ has a high probability to be a singleton, where $|C_i|$ is the number of tags in C_i. After assigning indices to the tags in C_i, if the reader finds that there exists a singleton index, it will select the corresponding tag to report the category-level information — in this case, we call C_i a *resolvable category*, which will be active in the polling phase. In Table II, we illustrate how to determine which categories are resolvable, where K is set to 2. For C_1, the indices of tags t_1, t_2 and t_3 are 2, 1, 1, respectively. Hence, 2 is a singleton index and C_1 is resolvable. Similarly, C_2 and C_4 are also resolvable. In contrast, category C_3 is not resolvable and thus will not be active in the polling phase because its only two tags t_5 and t_6 have the same index 2.

A slot in the virtual frame is *useful* if and only if it is a homogeneous slot and the assigned category is resolvable; otherwise, it is called a *useless* slot. In above example, since C_1 is resolvable and homogeneous, the 2nd slot is useful. In contrast, the 7th slot is useless as C_3 is irresolvable.

How will a tag know whether it is selected to transmit in a useful slot? The reader broadcasts the parameters $\langle f, r, K \rangle$ to all tags so that each tag can compute $H(cid, r) \mod f$ for which slot its category is assigned to, and it can also compute $\mathcal{R}(H(id, r) \mod 2^K)$ for its assigned index. But it does not know whether this is a singleton index. The reader does, and it must inform tags of which slots are useful and which tags should transmit in these slots. The useless slots will be removed from execution. For this purpose, following $\langle f, r \rangle$, the reader broadcasts an f-bit *ordering vector* V.

Each bit in V corresponds to a slot in the virtual frame: '0' indicates useless and '1' indicates useful. In the example of Fig. 3 and Table II, the ordering vector V is '01000000'. If V is too long, the reader can split it into multiple smaller segments and transmit one after another, allowing each tag to keep only the segment containing the bit that corresponds to the slot it is assigned to.

2) Polling Phase: Consider an arbitrary tag in an arbitrary category C_i. The ordering vector V carries two pieces of information: (1) The tag can learn whether its category is active or not by examining the $(H(cid, r) \mod f)$th bit in V. If the bit is one, the tag will compute the assigned index $\mathcal{R}(H(id, r) \mod 2^K)$ and participate in the polling phase. (2) V also tells the order of a useful slot in the actual frame to be carried out. If a tag finds that there are i ones in V preceding its bit (which is also 1), the tag knows that it should participate in the $(i+1)$th slot.

In the polling phase, the reader plays out an actual frame comprised of only useful slots. Consider an arbitrary slot. Without loss of generality, suppose category C_i is assigned to this slot and all its tags participate in the slot by listening to the reader, which begins the slot by transmitting the K-bit singleton index that it identifies for this category previously in the order phase. All tags in category C_i will compare the received index with their assigned indice $\mathcal{R}(H(id, r) \mod 2^K)$, and only one tag will find a match. This tag transmits the category-level information to the reader in the same slot, while other tags in the category keep silent.

Note that a category assigned to a useful slot may have more than one singleton index. In this case, the reader can randomly choose one to transmit in the slot; any singleton index will select a unique tag to respond. Each slot collects information from one category. Tags in these categories will not participate further in the protocol execution. The categories that are assigned to useless slots will participate in the subsequent rounds until all categories's information is collected by the reader. The reader chooses a different random seed in each round. Hence, the categories that are not resolvable in one round will become resolvable in other rounds. The overall workflow of TPS is shown in Fig. 4.

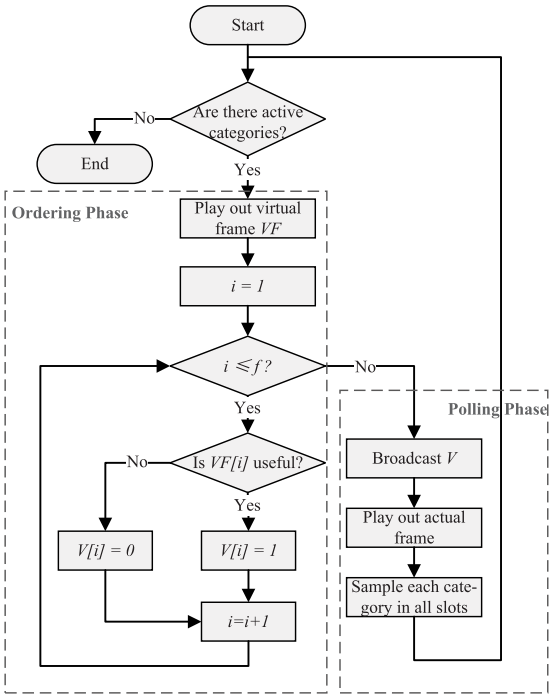


Fig. 4. The overall workflow of TPS.

C. Performance Analysis

We derive the expected execution time of the TPS protocol. Consider an arbitrary sampling round comprised of the ordering phase and the polling phase. The execution time t of this round is:

$$t = \frac{f}{96} \times t_{id} + f \times p \times (t_{poll} + t_{inf}), \tag{1}$$

where f is the length of the ordering vector V, p is the probability that a slot is useful, t_{poll} is the time for the reader to broadcast a $\log_2 K$-bit singleton index, and t_{inf} is the length of a time slot for a tag to transmit the required information. Note that, the control message transmission for launching each round is ignored here as this overhead covers only a couple of bits, which are negligible compared with the following frame transmission and index broadcasting by the reader. We define the *sampling efficiency*, denoted as λ, as the ratio of the number of sampled categories to the execution time of this round. Since a useful slot corresponds to a category to be sampled, the number of successfully sampled categories in this round is equal to that of useful slots, i.e., $f \times p$. We then get the sampling efficiency:

$$\lambda = \frac{f \times p}{t} = \frac{p}{\frac{t_{id}}{96} + p \times (t_{poll} + t_{inf})}. \tag{2}$$

Clearly, the bigger the value of λ is, the more categories will be sampled in each unit of execution time. We thus need to find the optimal p that maximizes λ. According to (2), λ monotonically increases as p increases; the objective is reduced to maximizing p. As aforementioned, a useful slot is both homogenous and resolvable, which are two independent events, we have:

$$p = p_h \times p_r, \tag{3}$$

where p_h is the probability that a slot is homogenous and p_r is the probability that the assigned category in this slot is resolvable. To maximize p, the goal further turns to maximizing both p_h and p_r, respectively. Consider the probability p_h. Since category ID is taken as the input for hash, the tags belonging to the same category must reside in the same slot. Hence, we can treat one category as 'one tag' and the homogenous slots are actually those slots picked by exactly 'one tag'. We have:

$$p_h = \binom{m'}{1} (\frac{1}{f})(1 - \frac{1}{f})^{m'-1} \approx \frac{m'}{f} \times e^{-\frac{m'-1}{f}}, \tag{4}$$

where e is the natural constant and m' is the number of unsampled categories before this round. Letting $\frac{d\lambda(f)}{df} = 0$, we derive the maximal p_h:

$$p_h^* = e^{-1} \quad \text{when} \quad f = m'. \tag{5}$$

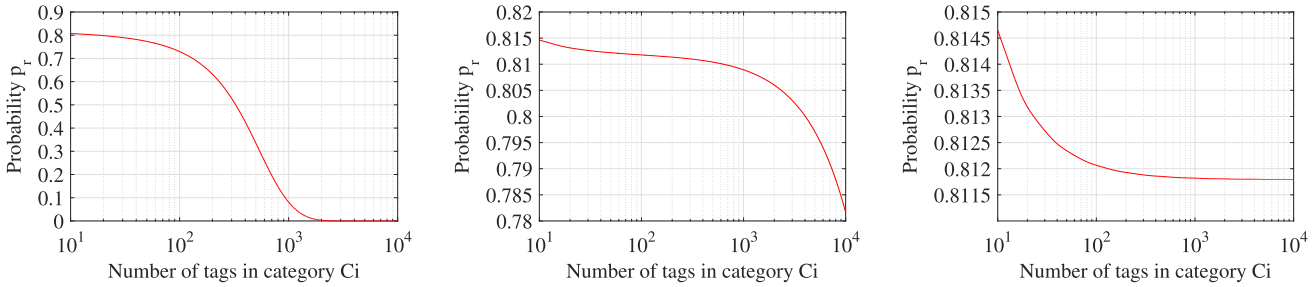


Fig. 5. Relationship between the probability p_r of a resolvable slot and the number of tags in a category C_i. (a) $K = 8$. (b) $K = 16$. (c) $K = 32$.

For the probability p_r, let us consider an arbitrary category C_i in a useful slot. After the $\mathcal{R}(\cdot)$ operation, each tag in C_i picks the index j with the probability of $\frac{1}{2^j}$. Let q_j be the probability that the index j is a non-singleton index (j is picked by none or multiple tags). We have:

$$q_j = 1 - \binom{|C_i|}{1} \times \frac{1}{2^j} \times (1 - \frac{1}{2^j})^{|C_i|-1}, \tag{6}$$

where $|C_i|$ is the cardinality of C_i, i.e., the number of tags in C_i. Let q_j^* be the probability that the maximal singleton index is j. We have:

$$q_j^* = \begin{cases} 1 - q_K, & if\ j = K \\ q_{j+1,K} - q_{j,K}, & if\ j \leq K, \end{cases} \tag{7}$$

where K is the last index and $q_{j,K}$ is the probability that all indices from j to K are non-singleton.

According to (6) and (7), we have the probability $p_r[K, |C_i|]$ that the assigned category C_i in this slot is resolvable:

$$\begin{aligned} p_r[K, |C_i|] &= \sum_{j=1}^{K} q_j^* \\ &= \sum_{j=1}^{K-1} (q_{j+1,K} - q_{j,K}) + (1 - q_K) \\ &= 1 - q_{1,K} \\ &\approx 1 - \prod_{j=1}^{K} (1 - \frac{|C_i|}{2^j} \times e^{-\frac{|C_i|}{2^j}}). \end{aligned} \tag{8}$$

Fig. 5 plots the relationship between p_r and the number $|C_i|$ of tags in C_i. When $K = 8$, the probability p_r decreases as $|C_i|$ increases. For example, $p_r \approx 0.8$ when $|C_i| = 20$, whereas $p_r \approx 0$ when $|C_i| = 10,000$. That is because, with the increase of $|C_i|$, the case of $K = 8$ hardly provides tags with sufficient indices to pick, leading to most collisions and lowering the probability p_r of a singleton index. In contrast, when $K = 16$, p_r sees only a slight decrease from 0.82 to 0.78 when $|C_i|$ varies from 10 to 10,000. The case of $K = 32$ almost remains stable at 0.81 regardless of $|C_i|$. The main reason is that the values of K in these two cases are big enough such that few tags in C_i can reach up to the last indices. We have the maximal probability p^* of a useful slot:

$$p^* = p_h^* \times p_r[K, |C_i|] = e^{-1} \times (1 - \prod_{j=1}^{K} (1 - \frac{|C_i|}{2^j} \times e^{-\frac{|C_i|}{2^j}})). \tag{9}$$

TABLE III
OPTIMAL K

$\mathbf{K}$	4	8	16	32
$\log_2 \mathbf{K}$	2	3	4	5
$\|\mathbf{C_i}\|$	[1,13]	[14,206]	[207,52892]	>52892

When this happens, the category C_i will be sampled in a useful slot and keep silent in the following rounds. In the ordering phase, the average number of bits for generating a useful slot is $\frac{1}{p^*}$. In the polling phase, the reader broadcasts $\log_2 K$-bit singleton index to interrogate a tag. Adding the both overhead, we get the total overhead $O(C_i)$ for singling out a tag in C_i:

$$O(C_i) = \frac{1}{p^*} + \log_2 K. \tag{10}$$

The value of K plays an important role in the overall communication overhead. Table III depicts the setting of the optimal K under different $|C_i|$. As we can see, $K = 16$ covers the interval [207, 52892], which can meet most of applications in practice. In this case, the polling vector is only $\log_2 K = 4$ bits long. Assume $|C_i| = 10,000$, we have the maximum of $O(C_i)$ by letting $p_r[K, |C_i|] = 0.78$, which is equal to $\frac{e}{0.78} + 4 \approx 7.5$ bits. Compared with transmitting 96-bit tag ID in basic polling, it is a great performance boost.

Since an arbitrary category C_i is sampled with the probability of p^* in a sampling round, the expected number of sampling rounds that C_i participates is $\frac{1}{p^*}$. According to (9), when $K = 16$ and $|C_i| = 10,000$, we have $p^* \approx 0.29$. The expected number of sampling rounds is 3.5.

IV. Back-and-Forth Sampling Protocol

In this section, we consider information sampling without any prior knowledge of tag IDs or category IDs.

A. Protocol Description

In TPS, a virtual frame is used by the reader to select a tag from each category for information reporting, which avoids redundant data transmission. However, if the reader does not know the IDs of tags in the system, how will it achieve high sampling efficiency when collecting category-level information? A naive solution is to use one of the existing identification protocols, e.g., Frame Slotted ALOHA (FSA) [35], which is designed to collect the IDs of tags and resolve the collision as tags transmit their IDs. When any tag

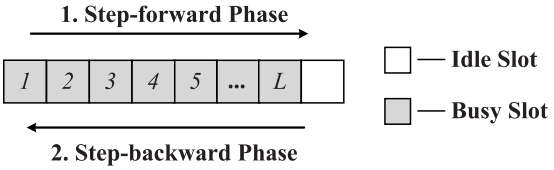


Fig. 6. Two phases of the BFS protocol.

sends its ID to the reader, it will piggybacks the category information. This approach causes high data redundancy as the same category information will be sent repetitively by all tags in the same category. To address this issue, we propose a back-and-forth sampling protocol (BFS) that selects a few tags from each category (without any pre-knowledge of their IDs) to report category-level information, while silencing other tags in the category. This is achieved in two phases: a *step-forward phase* for tag selection and a *step-backward phase* for information collection.

In the step-forward phase, the reader carries out a slotted frame and assigns a *slot index from a geometric distribution* to every tag, such that the number of tags assigned to each subsequent slot will decrease exponentially. The reader finds out the last busy slot, which has the highest index among the slots with at least one tag assigned. Then, it moves to the step-backward phase, where the busy slots are replayed in the reverse order for information collection. In each slot, the reader collects information from the tags assigned to that slot, including each tag's category ID and its category-level information. At the end of the slot, the reader will silence all other tags in the categories whose information has just been collected, informing them not to participate in the next slots. Our numerical results show that only a few tags will participate in each slot and with the optimal parameter settings, only about 1.45 tags from each category are sampled, very close to the lower bound of one tag. Below we describe the details.

1) Step-Forward Phase: The reader initiates a slotted frame by broadcasting a request with a random seed r. Upon receiving this request and r, each tag computes the slot index that it is assigned to as $\mathcal{R}(H(id, r))$, where id is the tag's ID, $H(\cdot)$ is a hash function, and $\mathcal{R}(\cdot)$ is a function that returns the index of the right-most bit of 1 in the binary representation of the input. After the request, the reader plays out the frame slot by slot, in ascending order of slot indices. Consider an arbitrary tag and suppose it is assigned to the jth slot. It will transmit a short response to the reader in each of the first jth slots, i.e., the assigned slot and all preceding slots. A slot is either *busy* when one or multiple tags transmit or *idle* when no tag transmits. This design helps our protocol determine when to stop in the step-forward phase. The protocol stops when it observes an idle slot. In this case, all subsequent slots must be idle (thus no need to continue). The reason is that, had there be a tag transmitting in a subsequent slot (say the jth slot), the tag would have transmitted in the current slot (thanks to the first-j-slots design). The slot right before the idle slot is the last busy slot, whose index is denoted as L.

Consider an arbitrary category C_i, $1 \leq i \leq m$. Its tags are assigned to the slots with an exponentially decreasing distribution. While all the $|C_i|$ tags will transmit in the first slot,

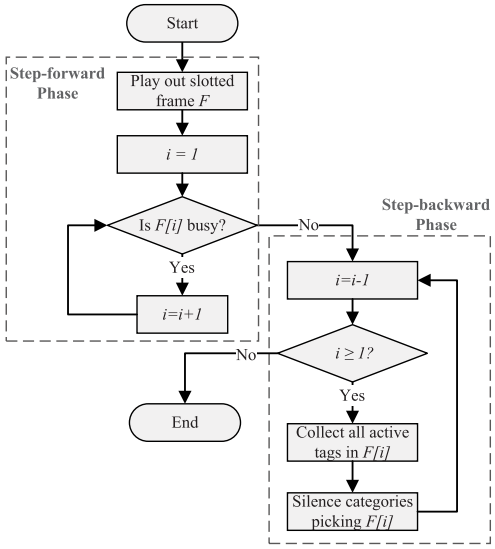


Fig. 7. The overall workflow of BFS.

the expected number of tags transmitting in the jth slot is $\frac{|C_i|}{2^{j-1}}$, due to the geometric distribution of $\mathcal{R}(\cdot)$. Consider the last slot that any tag in C_i is assigned to. It is likely that this slot contains only one or a small number of tags from C_i. We will collect the information of category C_i in this slot, while silencing all other tags from C_i assigned to other slots. That is the basic design of the step-backward phase.

2) Step-Backward Phase: This phase replays the busy slots from the previous phase in the reverse order, from the last busy slot of index L to the first slot, for a different purpose — collecting category-level information from tags assigned to the slots. Consider the jth slot, $1 \leq j \leq L$. The reader collects information from the tags assigned to this slot by using one of the tag identification protocols, e.g., the widely used C1G2 protocol [35]. The reader begins the slot by transmitting the slot index, which informs tags of this assigned index to participate by executing a tag identification protocol. Since only category information is needed, the tag will transmit its category ID together with the category information in the protocol; when the reader acknowledges the successful receipt of a tag's information, it will include the category ID in the ack, which will be overheard by all other tags — those with the same category ID will be silenced. The silenced tags will not participate in the future slots. With this design of replaying the slots in reverse (with fewest assigned tags first) and aggressively silencing tags, we ensure that only a small number of tags will be active in each category for information collection. Specifically, the information of each category is collected at the highest-indexed slot to which any tag from that category is assigned. The overall workflow of BFS is shown in Fig. 7.

We illustrate BFS with a scale-down RFID system. As shown in Fig. 8, there are three categories $C_1 = \{t_1, t_2, t_3\}$, $C_2 = \{t_4, t_5, t_6\}$, and $C_3 = \{t_7, t_8\}$. In the step-forward phase, each tag individually picks a slot j and replies to the reader in the first j slots. For example, the tag t_2 will respond in the first three slots. According to Fig. 8(a), the forth slot is the last busy slot. After that, the reader moves

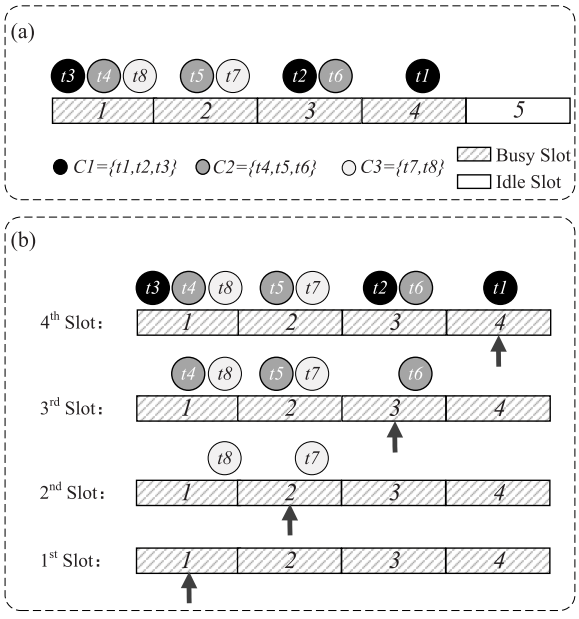


Fig. 8. An illustration of the BFS protocol. (a) Step-forward phase. (b) Step-backward phase.

to the step-backward phase and plays out the frame reversely. As shown in Fig. 8(b), the reader checks the 4th slot first. The tag t_1 picking this slot replies to the reader and all tags (t_{1-3}) belonging to C_1 keep silent after receiving the category ID of C_1. The reader then moves back to the 3rd slot and the tag t_6 is interrogated. The tags t_{4-6} from C_2 are silenced after that. Similarly, in the 2nd slot, C_3 is sampled; no tag responses in the 1st slot. By the back-and-forth operation, all category information is successfully collected and only one tag of each category is interrogated, saving communication overhead.

B. Performance Analysis

The execution time of BFS consists of two parts: the step-forward phase and the step-backward phase. In the step-forward phase, the reader needs to play out L busy slots and one idle slot; the corresponding communication overhead is $(L+1)t_s$, where t_s is the communication delay that a tag gives a one-bit short response to the reader. In the step-backward phase, consider an arbitrary category C_i, $1 \le i \le m$. It will be sampled in the slot when it appears at the first time. Let v_i be the number of tags belonging to C_i in this slot. To isolate these v_i tags from others, the ALOHA-based approach in the C1G2 protocol [35] needs to carry out e sub-frames including $e \times v_i$ slots in total [36], where e is the natural constant. Amongst these slots, there are v_i singleton slots (exactly one tag picks) and about v_i empty slots (no tag picks); the left are collision slots (more than one tag pick). The reader in the first singleton will collect the category ID together with the category information of C_i from the tag, and later silence C_i by broadcasting its category ID. The communication overhead is $t_{cid} + t_{inf} + t^r_{cid}$, where t_{cid} and t_{inf} are the delay for the tag to transmit the category ID and category-level information respectively, t^r_{cid} is the delay for the reader to broadcast a category ID. The other $v_i - 1$ singletons will become empty slots and there are $2v_i - 1$ empty slots with the length of t_s now. The left $(e-2)v_i$ would-be collision slots are likely to become singleton or empty after the reader silences some tags. For simplicity, we still treat them as collisions and get a upper bound of the execution time:

$$t = (L+1)t_s + \sum_{i=1}^{m} \big((e-2)v_i t_c + (2v_i - 1)t_s + \mu\big) + eLt_f, \tag{11}$$

where $\mu = t_{cid} + t_{inf} + t^r_{cid}$, t_c is the delay of a collision slot, t_f is the time interval between neighbor sub-frames in ALOHA-based protocols [8]. Since e sub-frames are required for sampling in each slot, the total overhead of inter-frame is $e \times L \times t_f$. In (11), all terms are constants except for L and v_i. We derive their expected values with the two lemmas below.

Lemma 1: The expect value $E(L)$ of L is:

$$E(L) = \sum_{j=1}^{\infty} j \times \big((1 - \frac{1}{2^j})^n - (1 - \frac{1}{2^{j-1}})^n\big),$$

where n is the number of tags in $\mathcal{N}$.

Proof: Consider a tag that picks the slot Q, $Q \ge 1$. The probability that Q is the jth slot is:

$$p(Q = j) = \frac{1}{2^j}. \tag{12}$$

According to (12), the probability that the tag chooses one of the first j slots is:

$$p(Q \le j) = \sum_{k=1}^{j} \frac{1}{2^k} = 1 - \frac{1}{2^j}. \tag{13}$$

Given n tags in the tag set $\mathcal{N}$, we have the probability that all of them pick the first j slots:

$$p^n(Q \le j) = (1 - \frac{1}{2^j})^n. \tag{14}$$

According to (14), we have the probability that the biggest slot index L picked by the tag set $\mathcal{N}$ is equal to j:

$$\begin{aligned} p(L = j) &= p^n(Q \le j) - p^n(Q \le j-1) \\ &= (1 - \frac{1}{2^j})^n - (1 - \frac{1}{2^{j-1}})^n. \end{aligned} \tag{15}$$

Hence, the expected value $E(L)$ of L is:

$$\begin{aligned} E(L) &= \sum_{j=1}^{\infty} j \times p(L = j) \\ &= \sum_{j=1}^{\infty} j \times \big((1 - \frac{1}{2^j})^n - (1 - \frac{1}{2^{j-1}})^n\big). \end{aligned} \tag{16}$$

■

Lemma 2: Consider an arbitrary category C_i, $1 \le i \le m$. The expected value $E(v_i)$ of v_i is:

$$E(v_i) = |C_i| \sum_{j=1}^{\infty} (\frac{1}{2^j})(1 - \frac{1}{2^j})^{|C_i|-1}, \tag{17}$$

where $|C_i|$ is cardinality of C_i, i.e., the number of tags in C_i.

Proof: Let $\mathbb{E}_{k,j}$ represent the event that k tags in C_i pick the jth slot and j is the last busy slot for C_i. Hence, we have the probability of $\mathbb{E}_{k,j}$:

$$p(\mathbb{E}_{k,j}) = \binom{|C_i|}{k} (\frac{1}{2^j})^k (1 - \frac{1}{2^{j-1}})^{|C_i|-k}, \tag{18}$$

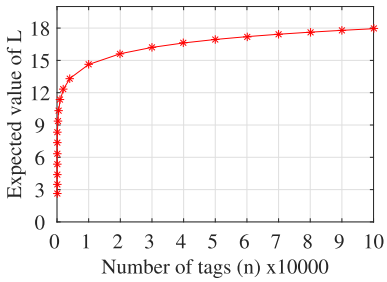


Fig. 9. The expected value of L with respect to the number n of tags.

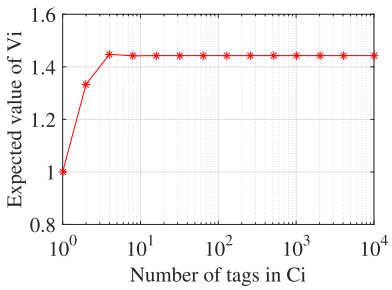


Fig. 10. The expected value of v_i with respect to the number of tags in the category C_i.

where the term $(\frac{1}{2^j})^k$ denotes the probability that k tags pick the jth slot, and the term $(1-\frac{1}{2^{j-1}})^{|C_i|-k}$ indicates that the left $(|C_i|-k)$ tags choose the slots preceding the jth slot. According to (18), we have the expected value of v_i:

$$\begin{aligned}
E(v_i) &= \sum_{j=1}^{\infty}\sum_{k=1}^{|C_i|} k \times p(\mathbb{E}_{k,j}) \\
&= \sum_{j=1}^{\infty}\sum_{k=1}^{|C_i|} k\binom{|C_i|}{k}(\frac{1}{2^j})^k(1-\frac{1}{2^{j-1}})^{|C_i|-k} \\
&= \sum_{j=1}^{\infty}\sum_{k=0}^{|C_i|} k\binom{|C_i|}{k}(\frac{1}{2^j})^k(1-\frac{1}{2^{j-1}})^{|C_i|-k} \\
&= \sum_{j=1}^{\infty}|C_i|\frac{1}{2^j}\sum_{k=0}^{|C_i|-1}\binom{|C_i|-1}{k}(\frac{1}{2^j})^k(1-\frac{1}{2^{j-1}})^{|C_i|-1-k} \\
&= |C_i|\sum_{j=1}^{\infty}(\frac{1}{2^j})(1-\frac{1}{2^j})^{|C_i|-1}.
\end{aligned}$$

■

Substituting $E(L)$ and $E(v_i)$ for L and v_i in (11) respectively, we can derive the overall execution time t of BFS. According to Lemma 1, we show the expected value of L with respect to the number n of tags in Fig. 9. In this figure, the number n of tags ranges from 1 to 100,000. As we can see, the expected value of L experiences a logarithmic growth over n. For example, L is equal to 17.9 when the number of tags is 100,000. That means the reader under this case needs to carry out only $(L+1) \approx 19$ slots in the step-forward phase for making tags in each category be distributed unevenly. According to Lemma 2, Fig. 10 shows the expected value of v_i with respect to the number $|C_i|$ of tags in the category C_i. The results show that no more than 1.45 tags for each category are sampled by the reader, which is very close to the lower bound of one tag.

C. Multiple Readers

So far, we have discussed the sampling problem in a single reader case. In some real scenarios, multiple readers can be deployed to manage a large number of tags. Our protocols can be easily generalized to the multi-reader case when the collision-free transmission schedule (e.g., [37]) among the readers is established. More specifically, for TPS, we assume that each reader has the knowledge of the subset of tags under its coverage. With this information, each reader executes TPS as-is when it is the reader¡¯s turn to run information sampling according to the scheduling algorithm. Note that, the reader in this case just needs to hold the information about the subset of tags in its field-of-view; the global tag information is unnecessary. For BFS, since it is able to do sampling without any assumption about the tag information, no any modifications to BFS are required for running the information sampling. Therefore, we assert that, our protocols can be easily generalized to the multi-reader RFID system.

V. Evaluation

A. Simulation Setting

Our simulation settings follow the specification of the C1G2 standard [35]. Any two consecutive communications, from the reader to tags or vice versa, are separated by a time interval. For one, after the reader transmits a command, all tags have to wait the transmit-to-receive turn-around time T_1 before replying to the reader. For another, upon receiving the reply from tags, the reader needs to wait the receive-to-transmit turn-around time T_2 before talking to tags. According to the specification, T_1 is $\max(RTcal, 20T_{pri})$ and T_2 ranges from $3T_{pri}$ to $20T_{pri}$, where $RTcal$ is the reader-to-tag calibration symbol that equals the length of the data-0 symbol plus the length of the data-1 symbol, and T_{pri} is the backscatter-link pulse-repetition interval. In our simulation, we set $T_1 = T_2 = T = 200\ \mu s$ that complies with the C1G2 standard.

Depending on the physical implementation and the real environment, the transmission rates between the reader and tags are not necessarily symmetric. The tag-to-reader transmission rate varies with the data coding: 40 kbps to 640 kbps for FM0 and 5 kbps to 320 kbps for Miller-modulated subcarrier. We get the intersection set 40 kbps to 320 kbps and adopt the lower bound 40 kbps as the data rate. In other words, it takes 25 μs to transmit one bit by the tag. The data rate from the reader to tags ranges from 26.7 kbps to 128 kbps. Similarly, we set the data rate to the lower bound 26.7 kbps, which takes 37.45 μs to transmit one bit by the reader. Besides, the length of the category ID is set to 32 bits throughout the simulations. Note that other parameter settings may change the absolute metric, but the simulation conclusions can be drawn in a similar way.

According to the above parameter settings, the duration t_s of the 1-bit short response from tags is equal to $25+T = 225\ \mu s$; the duration t_{cid} of transmitting a category ID by a tag is $25 \times 12 + T = 500\ \mu s$; the duration t^r_{cid} of broadcasting a

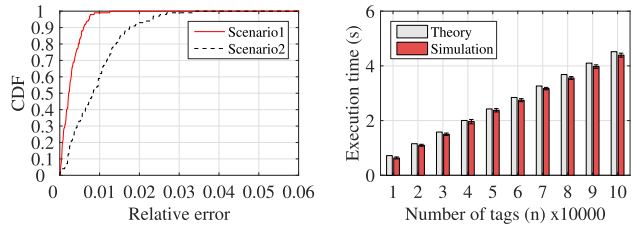


Fig. 11. Confirmation of theoretical results through simulations for (a) the execution-time formulas of TPS, (b) the execution-time formulas of BFS.

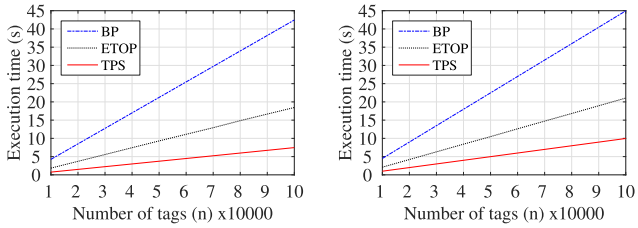


Fig. 12. Execution time with respect to the number n of tags. (a) $w = 10$. (b) $w = 20$.

category ID by the reader is $37.45 \times 32 + T = 1398.4$ μs; the duration t_{id}^r of broadcasting a tag ID by the reader is $37.45 \times 96 + T = 3795.2$ μs; For transmitting w-bit category information by a tag, the duration t_{inf} is equal to $25w + T$ μs. All results are the average outcome of 100 simulation runs using MATLAB.

B. Verification of Execution Time

In Fig. 11(a) and Fig. 11(b), we conduct simulations to verify the correctness of the derived execution-time formulas for TPS and BFS respectively. For TPS, we compare the theoretical derivations with the simulation results under two scenarios. In the first scenario, the value of K in the polling phase is set to 16, the number of categories is 100 and each category consists of 100 tags. In the second scenario, we keep the same value of K, but change the number of categories to 1000 and the number of tags in each category to 50. We perform 100 independent simulation runs in each scenario and plot the CDF of relative error. The relative error is computed as $\frac{|t_1 - t_2|}{t_1}$, where t_1 is the simulation time and t_2 is the theoretical time. In 11(a), we observe that the relative error of TPS is less than 0.03. Its 90 percentile is about 0.02. With the increase of the number of categories (scenario2), the relative error further decreases, which is no more than 0.01. The tightness between the simulation value and the theoretical one demonstrates that the derived execution time can well depict the real situation.

For BFS, since the formulas derive a upper bound of the execution time, we verify the gap between the upper bound and the real case. In this simulation, we set the number of tag in each category to 100 and vary the total number of tags from 10,000 to 100,000. As shown in 11(b), the theoretical value is slightly larger than the simulation one. That is because, some collision slots are likely to become singleton or empty after the reader silences some tags and we still treat them as collisions, which increases the communication overhead. Even so, we assert that the given upper bound is close to the real execution time.

C. Evaluation With Tag IDs

In this subsection, we evaluate the execution time of our sampling protocol TPS in the case of knowing tag IDs. As aforementioned in Section III, Basic Polling (BP) and ETOP can be modified for the purpose of information sampling. To achieve this goal, the reader first randomly picks a tag from each category and these tags constitute a tag set S. After that, for BP, the reader in turn broadcasts each tag's ID in S and then waits for their replies after each broadcasting. All tags keep listening and only the exactly matched tag transmits the required category information to the reader. The sampling process terminates until all tags in S are interrogated. For another, ETOP is specifically designed for collecting tag information from a tag subset in an efficient way. In our problem, we can treat S as the wanted subset of $\mathcal{N}$ and execute ETOP as-is to collect the category information as required. In the simulations, we keep the same parameter settings of ETOP as that in [34], i.e., the frame size is $24 \times |S|$, the segment is 80 bits long, and each segment consists of 4 partitions.

Fig. 12 compares the execution time of BP, ETOP, and TPS under different numbers of tags. In the simulations, we fix the number of tags in each category at 10 and vary the number of tags from 10,000 to 100,000 (the number of categories ranges from 1000 to 10,000). Two kinds of category information with different lengths $(w = 10, 20)$ are sampled under varied parameter settings, as shown in Fig. 12(a) and Fig. 12(b). Among these figures, we observe that TPS outperforms the other two protocols as it isolates a tag of each category from others by broadcasting only about 7.5-bit polling vector on average. For example, to sample 10-bit category information from 50,000 tags (shown in Fig. 12(a)), BP takes the longest time 21.2s as it needs to transmit tedious tag IDs. ETOP reduces the execution time by 56.1% to 9.3s since it uses segmented Bloom filters to isolate and order tags belonging to S, avoiding most ID transmissions. TPS performs the best and consumes only 3.7s, producing $5.7\times$ and $2.5\times$ performance gains, compared with BP and ETOP respectively. The similar conclusion can also be drawn on other parameter settings in the other figure: TPS is the best, ETOP follows, and BP performs the worst.

In Fig. 13, we compare the execution time of BP, ETOP, and TPS under different numbers of tags in each category. In the simulation, we fix the number of tags at 100,000 and vary the number of tags in each category from 5 to 50. Once given the length w of the category information, we observe that the execution time of these three protocols decreases as the number of tags in each category increases. That is because the number of categories decreases with the increase of the number of tags in each category, thereby reducing the number of samples and saving the communication overhead. Similar to Fig. 12, the same conclusion can also be drawn here: TPS is the most time-efficient, ETOP is worse, and BP is the most time-consuming. For instance, when the length of category

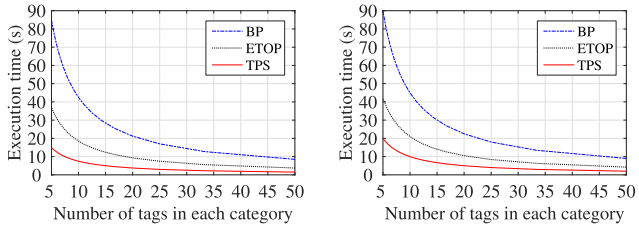


Fig. 13. Execution time with respect to the category size. (a) $w = 10$. (b) $w = 20$.

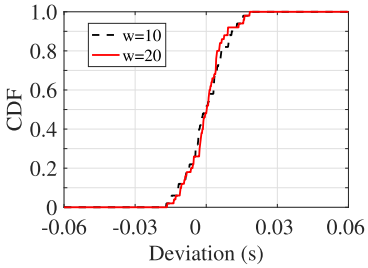


Fig. 14. Variances of the execution time of TPS.

information is 20 bits long and each category has 25 tags (as shown in Fig. 13(b)), the execution time of BP is 17.0s, which is the longest amongst the three protocols. By contrast, TPS spends the minimal execution time. It takes less than 3.0s to achieve the same sampling task, producing an about $6\times$ performance gain. Although ETOP is far superior to BP, it takes longer time than TPS, i.e., 7.4s. Note that, the execution time of the three protocols increases as w increases. This is intuitive as the tag needs to transmit more category-related data when w is bigger. Based on above simulation results, we conclude that, by transmitting a few bits to pick a tag in each category, TPS outperforms BP and ETOP when tag IDs are known a priori, greatly improving sampling efficiency.

In Fig. 14, we study the variances of the execution time of TPS. The number of tags and the category size are set to 50,000 and 10, respectively. Two kinds of category information are collected, with different lengths $(w = 10, w = 20)$. Each CDF curve in the figure is the results of 100 simulation runs for a fixed length of information. When $w = 10$, the mean of the execution time is 3.6s, and each deviation between the result of a run and the mean value is computed. As we can see, the 95th percentile of the execution-time deviations is bounded within 0.03s, which is less than 1% of the mean execution time by TPS. Similarly, when $w = 20$, the mean execution time is 4.9s and the 95th percentile of the deviations is also less than 0.03s, which is about 0.6% of the mean.

D. Evaluation Without Tag IDs or Category IDs

In the following simulations, we relax the assumption in TPS and evaluate the sampling performance of BFS without any knowledge of tag IDs or category IDs. Before the evaluation, we first give two modified solutions to the sampling problem that serve as the baseline protocols for comparison. The first solution is built on the widely used ID collection protocol: Framed Slotted ALOHA (FSA) [36]. Unlike basic FSA, this FSA-based sampling (FSAS) solution just collects each tag's category ID instead of the long tag ID to save communication overhead. More specifically, FSAS consists of multiple execution rounds. In each round, the reader carries out a slotted frame and each tag in $\mathcal{N}$ randomly picks one slot in the frame. Once a slot is picked by only one tag, the tag reports its category ID together with the category information to reader. In this way, the reader achieves the sampling task of this category and then broadcasts the category ID to silence left tags belonging to the category. The protocol terminates until all categories are sampled.

Although we make some efforts in FSAS to avoid the waste of querying multiple tags in a category, the number of slots played by the reader is proportioned to the number n of tags, which is time-consuming when $n \gg m$, e.g., $m = 0.01n$, where m is the number of categories. The main reason is that, when a category is sampled in the previous slot, the following slots picked by only the tags (which are silent) belonging to this category become empty (no tag replies) and the reader still needs to spend time to check each of them. To avoid this waste, we come up with the second solution called enhanced FSAS (EFSAS). EFSAS generally consists of about m execution rounds, where m is the number of categories. Unlike FSAS, in each round, the reader in EFSAS samples only one category using FSAS. Specifically, after collecting the category information from a tag and silencing others in the category, the reader terminates the current frame, instead of advancing to the next slot in the frame. Another category will be sampled in the next frame in the same way. By this means the reader does not need to take extra time to check empty slots picked by silenced tags, greatly reducing the delay.

Notice that, in the simulation of EFSAS, we are supposed to consider the inter-frame duration as EFSAS needs to execute many sampling rounds when m is large. The inter-frame duration is the time interval between two neighbor frames, which includes the duration of powering down the reader and the duration of transmitting carrier waves to power tags before communication [8]. We refer to the inter-frame duration as t_f, which is usually larger than t_{id}, as shown in the experimental results in [8]. Let $t_f = \lambda t_{id}$. In the simulation, we set $\lambda = 2$. Besides, since the reader detects a collision via RN16 in the C1G2 standard, we get the duration t_c of collision slots (picked by more than one tag): $25 \times 16 + T = 600\ \mu s$. For fairness, we also count the inter-frame duration when executing the protocols FSAS and BFS.

In Fig. 15 and Fig. 16, we study the execution time of FSAS, EFSAS, and BFS when sampling two different lengths of category information, i.e., $w = 10$, and $w = 20$. Fig. 15 compares the execution time of the three protocols under different numbers of tags. In each simulation, we fix the number of tags in each category at 100 and vary the number n of tags from 10,000 to 100,000 (the number of categories ranges from 100 to 1000, which differs from the setting of TPS). Among the two subfigures, we observe that BFS is faster than the other two protocols. That is because, by carrying out the back-and-forth frame, BFS is able to query only 1.45 tags on average for each category, close to the lower bound. FSAS performs the worst as it has to check a great number of slots

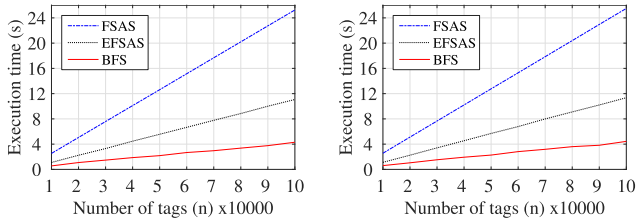


Fig. 15. Execution time with respect to the number n of tags. (a) $w = 10$. (b) $w = 20$.

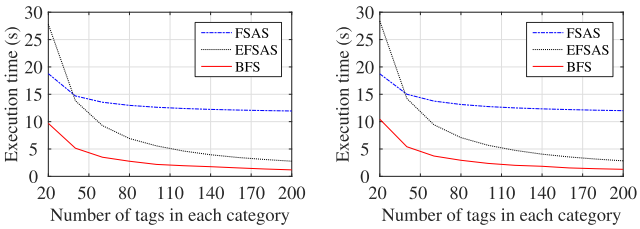


Fig. 16. Execution time with respect to the category size. (a) $w = 10$. (b) $w = 20$.

which is proportional to n instead of m. EFSAS is better than FSAS since it avoids the waste of checking most empty slots. However, it is still worse than BFS. We examine the performance gap under an arbitrary parameter setting, such as $w = 10$ and $n = 10^5$. As shown in Fig. 15(a), FSAS spends the longest sampling time 25.3s. EFSAS reduces the execution time to 11.2s, 44.3% of FSAS. TPS further decreases the execution time to 4.1s, just 16.3% of FSAS.

In Fig. 16, we also study the relationship between the execution time of the three protocols and the number of tags in each category. In the simulations, we fix the number of tags at 100,000 and vary the number of tags in each category from 20 to 200. Clearly, under various settings, TPS always performs the most efficiently. The performance comparison of FSAS and EFSAS depends on the number of tags in each category. If this number is smaller than 40 in our simulations, FSAS is better than EFSAS. Otherwise, EFSAS is better. The main reason is that, although EFSAS is able to avoid the waste of empty slots and make the sampling duration proportional to the number m of category (rather than n in FSAS), EFSAS needs to issue about m frames and the inter-frame overhead t_f between neighbor frames is much bigger than duration t_s of empty slots ($t_f \approx 38\ t_s$). Hence, when each category has a small number of tags, it is not worthwhile frequently issuing new frames for sampling each category.

In Fig. 17, we study the variances of the execution time of BFS. The number of categories and the category size are set to 1000 and 100, respectively. Two kinds of category information are collected, with different lengths ($w = 10, w = 20$). Each CDF curve in the figure is the results of 100 simulation runs for a fixed length of information. When $w = 10$, the mean of the execution time is 4.1s, and each deviation between the result of a run and the mean value is computed. As we can see, the 90th percentile of the execution-time deviations is bounded within 0.2s, which is less than 5% of the mean. Similarly, when $w = 20$, the mean execution time is 4.4s and the 95th

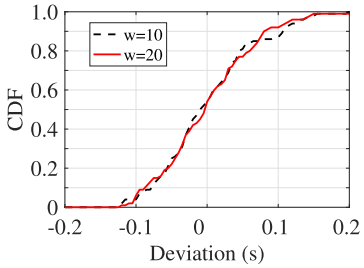


Fig. 17. Variances of the execution time of BFS.

percentile of the deviations is also less than 0.2s, which well indicates the good stability of our protocol.

According to above simulation results, we say that, by carrying out the back-and-forth frame, the proposed protocol BFS is superior to the baseline protocols FSAS and EFSAS, greatly improving the sampling efficiency under the case of unknowing tag IDs or category IDs.

VI. Related Work

A great number of research has been conducted on various issues in RFID systems. Much prior work focuses on the fundamental ID-collection problems. The key ideas are to avoid tag-to-tag collisions in the open wireless channel, which generally fall into two categories: the ALOHA-based [35], [36], [38] and the tree-based [11], [14], [39]. The former collects a tag's ID by carrying out a slotted frame and isolating the tag in a singleton slot (picked by exactly one tag) in the frame. The tree-based solutions iteratively split a tag set into smaller ones by dynamically adjusting and broadcasting an ID prefix. This process repeats until only one tag is left and queried by the reader. In recent years, the research interests in RFID systems have been shifted to some application-oriented functions. For example, cardinality estimation [6], [13], [40] is to count the number of tags; missing tag identification [9], [10] is to identify whether and which tags are absent; searching problems [7], [41] try to find a group of interested tags from the existing tag set.

Information collection [32]–[34], [42], as another branch of these new functional research, has attracted wide attentions due to its practical importance. Chen *et al.* [32] first formulate this problem and propose a time-efficient multihash information collection protocol (MIC) to collect sensor information from all tags. By using multiple hash functions, MIC is able to resolve most hash collisions in a slotted frame, greatly improving the protocol performance. In the follow-up work, Yue *et al.* [33] propose a Bloom filter based Information Collection protocol (BIC) that is tailored to the information collection under the case of multiple RFID readers. By distributively constructing and transmitting a Bloom filter, each reader can quickly identify which tags are under its coverage, speeding up the overall information collection. Liu *et al.* [42] propose a tree-based polling protocol (TPP) that improves the time efficiency of information collection by shortening the length of the polling vector. Qiao *et al.* [34] design an efficient polling-based protocol that collects tag information from only a wanted tag subset. Although these work achieve high performance, they need to collect all tags' information or take

the entire tag set into account each time, which is time-consuming for the task of category information collection in multi-category RFID systems.

VII. Conclusion

In this paper, we study the problem of category information collection in a multi-category RFID system. We propose two time-efficient sampling protocols, two-phase sampling (TPS) and back-and-forth sampling (BFS), that solve the sampling problem under two different cases. In the first case of knowing tag IDs, TPS shortens the length of the polling vector to only 7.5 bits for each category with two-phase hash technology. In the second case of unknowing any tag information, by carrying out the step-forward frame and using the step-backward scheme, BFS just needs to query about 1.45 tags for sampling each category. Extensive results show that our protocols TPS and BFS outperform the baseline protocols, greatly improving the sampling efficiency.

References

[1] L. Shangguan and K. Jamieson, "The design and implementation of a mobile RFID tag sorting robot," in *Proc. ACM MobiSys*, 2016, pp. 31–42.

[2] L. Shangguan, Z. Yang, A. X. Liu, Z. Zhou, and Y. Liu, "Relative localization of RFID tags using spatial-temporal phase profiling," in *Proc. USENIX NSDI*, 2015, pp. 251–263.

[3] J. Liu *et al.*, "RF-scanner: Shelf scanning with robot-assisted RFID systems," in *Proc. IEEE INFOCOM*, May 2017, pp. 1–9.

[4] X. Liu *et al.*, "Top-k queries for categorized RFID systems," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 2587–2600, Oct. 2017.

[5] X. Liu *et al.*, "RFID estimation with blocker tags," *IEEE/ACM Trans. Netw.*, vol. 25, no. 1, pp. 224–237, Feb. 2017.

[6] Y. Zheng and M. Li, "Towards more efficient cardinality estimation for large-scale RFID systems," *IEEE/ACM Trans. Netw.*, vol. 22, no. 6, pp. 1886–1896, Dec. 2014.

[7] Y. Zheng and M. Li, "Fast tag searching protocol for large-scale RFID systems," *IEEE/ACM Trans. Netw.*, vol. 21, no. 3, pp. 924–934, Jun. 2013.

[8] L. Xie, H. Han, Q. Li, J. Wu, and S. Lu, "Efficient protocols for collecting histograms in large-scale RFID systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 9, pp. 2421–2433, Sep. 2015.

[9] M. Shahzad and A. X. Liu, "Fast and reliable detection and identification of missing RFID tags in the wild," *IEEE/ACM Trans. Netw.*, vol. 24, no. 6, pp. 3770–3784, Dec. 2016.

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

[11] M. Shahzad and A. X. Liu, "Probabilistic optimal tree hopping for rfid identification," *IEEE/ACM Trans. Netw.*, vol. 23, no. 3, pp. 796–809, Jun. 2015.

[12] X. Liu, S. Zhang, B. Xiao, and K. Bu, "Flexible and time-efficient tag scanning with handheld readers," *IEEE Trans. Mobile Comput.*, vol. 15, no. 4, pp. 840–852, Apr. 2016.

[13] C. Qian, H. Ngan, Y. Liu, and L. M. Ni, "Cardinality estimation for large-scale RFID systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 9, pp. 1441–1454, Sep. 2011.

[14] C. Qian, Y. Liu, R. H. Ngan, and L. Ni, "ASAP: Scalable collision arbitration for large RFID systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 7, pp. 1277–1288, Jul. 2013.

[15] Q. Xiao, S. Chen, and M. Chen, "Joint property estimation for multiple RFID tag sets using snapshots of variable lengths," in *Proc. ACM MobiHoc*, 2016, pp. 151–160.

[16] Q. Xiao, M. Chen, S. Chen, and Y. Zhou, "Temporally or spatially dispersed joint RFID estimation using snapshots of variable lengths," in *Proc. ACM MobiHoc*, 2015, pp. 247–256.

[17] M. Chen, J. Liu, S. Chen, Y. Qiao, and Y. Zheng, "DBF: A general framework for anomaly detection in RFID systems," in *Proc. IEEE INFOCOM*, May 2017, pp. 1–9.

[18] S. Qi, Y. Zheng, M. Li, Y. Liu, and J. Qiu, "Scalable industry data access control in RFID-enabled supply chain," *IEEE/ACM Trans. Netw.*, vol. 24, no. 6, pp. 3551–3564, Dec. 2016.

[19] C.-H. Lee and C.-W. Chung, "RFID data processing in supply chain management using a path encoding scheme," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 5, pp. 742–758, May 2011.

[20] A. Sarac, N. Absi, and S. Dauzère-Peres, "A literature review on the impact of RFID technologies on supply chain management," *Int. J. Prod. Econ.*, vol. 128, no. 1, pp. 77–95, 2010.

[21] G. Wang *et al.*, "HMRL: Relative localization of RFID tags with static devices," in *Proc. IEEE SECON*, Jun. 2017, pp. 1–9.

[22] J. Han *et al.*, "Twins: Device-free object tracking using passive tags," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1605–1617, Jun. 2016.

[23] J. Han *et al.*, "Cbid: A customer behavior identification system using passive tags," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2885–2898, Oct. 2016.

[24] L. Yang *et al.*, "Tagoram: Real-time tracking of mobile RFID tags to high precision using COTS devices," in *Proc. ACM MobiCom*, 2014, pp. 237–248.

[25] L. Yang, Q. Lin, X. Li, T. Liu, and Y. Liu, "See through walls with COTS RFID system!" in *Proc. ACM MobiCom*, 2015, pp. 487–499.

[26] J. Wang, J. Xiong, H. Jiang, X. Chen, and D. Fang, "D-Watch: Embracing 'Bad' multipaths for device-free localization with COTS RFID devices," *IEEE/ACM Trans. Netw.*, vol. 25, no. 6, pp. 3559–3572, Dec. 2017.

[27] L. Xie, C. Wang, A. X. Liu, J. Sun, and S. Lu, "Multi-touch in the air: Concurrent micromovement recognition using RF signals," *IEEE/ACM Trans. Netw.*, vol. 26, no. 1, pp. 231–244, Feb. 2018.

[28] C. Wang *et al.*, "RF-kinect: A wearable RFID-based approach towards 3D body movement tracking," in *Proc. ACM UbiComp*, vol. 2, no. 1, 2018, pp. 41:1–41:28.

[29] X. Liu *et al.*, "Range-based localization for sparse 3D sensor networks," *IEEE Internet Things J.*, to be published, doi: 10.1109/JIOT.2018.2856267.

[30] J. Liu, M. Chen, S. Chen, Q. Pan, and L. Chen, "Tag-compass: Determining the spatial direction of an object with small dimensions," in *Proc. IEEE INFOCOM*, May 2017, pp. 1–9.

[31] J. R. Smith, *Wirelessly Powered Sensor Networks and Computational RFID*. New York, NY, USA: Springer-Verlag, 2013.

[32] 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.

[33] H. Yue, C. Zhang, M. Pan, Y. Fang, and S. Chen, "A time-efficient information collection protocol for large-scale RFID systems," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 2158–2166.

[34] Y. Qiao, S. Chen, T. Li, and S. Chen, "Energy-efficient polling protocols in RFID systems," in *Proc. ACM MobiHoc*, 2011, p. 25:1–25:9.

[35] *EPC Radio-Frequency Identity Protocols Generation-2 UHF RFID Standard*, Standard ISO/IEC 18000-63, GS1, Jul. 2018. [Online]. Available: https://www.gs1.org/standards/epc-rfid/uhf-air-interface-protocol

[36] S.-R. Lee, S.-D. Joo, and C.-W. Lee, "An enhanced dynamic framed slotted ALOHA algorithm for RFID tag identification," in *Proc. MobiQuitous*, Jul. 2005, pp. 166–172.

[37] J. Waldrop, D. W. Engels, and S. E. Sarma, "Colorwave: A MAC for RFID reader networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, vol. 3. Mar. 2003, pp. 1701–1704.

[38] H. Vogt, "Efficient object identification with passive RFID tags," in *Proc. IEEE PerCom*, Aug. 2002, pp. 98–113.

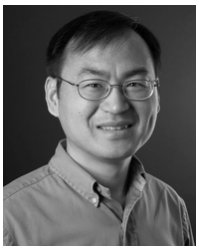
[39] L. Pan and H. Wu, "Smart trend-traversal: A low delay and energy tag arbitration protocol for large RFID systems," in *Proc. IEEE INFOCOM*, Apr. 2009, pp. 2571–2575.

[40] Q. Xiao, B. Xiao, and S. Chen, "Differential estimation in dynamic RFID systems," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 295–299.

[41] M. Chen, W. Luo, Z. Mo, S. Chen, and Y. Fang, "An efficient tag search protocol in large-scale RFID systems with noisy channel," *IEEE/ACM Trans. Netw.*, vol. 24, no. 2, pp. 703–716, Apr. 2016.

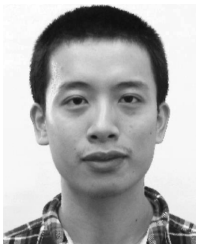
[42] J. Liu, B. Xiao, X. Liu, and L. Chen, "Fast RFID polling protocols," in *Proc. ICPP*, Aug. 2016, pp. 304–313.

Jia Liu (M') received the B.E. degree in software engineering from Xidian University, Xi'an, China, in 2010, and the Ph.D. degree in computer science and technology from Nanjing University, Nanjing, China, in 2016. He is currently a Research Assistant Professor with the Department of Computer Science and Technology, Nanjing University, Nanjing, China. His research mainly focuses on RFID systems. He is a member of the IEEE and ACM.



Shigang Chen (F') received the B.S. degree from the University of Science and Technology of China in 1993 and the M.S. and Ph.D. degrees from the University of Illinois at Urbana–Champaign in 1996 and 1999, respectively, all in computer science. He was with Cisco Systems for three years before joining the University of Florida in 2002. He served on the Technical Advisory Board for Protego Networks Inc. from 2002 to 2003 and as the CTO for Chance Media Inc. from 2012 to 2014. He is currently a Professor with the Department of Computer and Information Science and Engineering, University of Florida. He has published more than 140 peer-reviewed journal/conference papers. He holds 12 U.S. patents. His research interests include computer networks, Internet security, wireless communications, and distributed computing. He is a fellow of the IEEE. He received the IEEE Communications Society Best Tutorial Paper Award and the NSF CAREER Award. He served in various chair positions or as committee members for numerous conferences. He is an Associate Editor for the IEEE/ACM TRANSACTIONS ON NETWORKING. He served as an Editor for a number of other journals.

Qingjun Xiao (M') received the B.Sc. degree from the Computer Science Department, Nanjing University of Posts and Telecommunications, China, in 2003, the M.Sc. degree from the Computer Science Department, Shanghai Jiao Tong University, China, in 2007, and the Ph.D. degree from the Computer Science Department, The Hong Kong Polytechnic University, in 2011. He held a post-doctoral position with the Computer Science Department, Georgia State University, and the Computer Science Department, University of Florida, for three years combined. He is currently an Assistant Professor with Southeast University, China. He has published 16 papers as the first author on high-quality conferences or journals. His research interests include protocol and algorithm design in wireless sensor networks, RFID systems, or for network traffic measurement. He is a member of the IEEE, ACM, and China Computer Federation.



Min Chen received the B.E. degree in information security from the University of Science and Technology of China in 2011 and the M.S. and Ph.D. degrees in computer science from the University of Florida in 2015 and 2016, respectively. He is currently a Software Engineer with Google Inc. His advisor is Dr. S. Chen. His research interests include Internet of Things, big network data, next-generation RFID systems, and network security.

Bin Xiao (SM') received the B.Sc. and M.Sc. degrees in electronics engineering from Fudan University, China, and the Ph.D. degree in computer science from The University of Texas at Dallas, USA. He joined the Department of Computing, The Hong Kong Polytechnic University, as an Assistant Professor, where he is currently an Associate Professor and the Director of the Mobile Cloud Computing Laboratory. He is the Editor of three books and has published more than 100 technical papers in international journals and conferences. His research is mainly on mobile cloud computing, smart phone technology, network security, and wireless networks. He is the IEEE Senior member. He is currently the Associate Editor of the *International Journal of Parallel, Emergent and Distributed Systems*.

Lijun Chen received the B.S. degree in electrical engineering from the Xi'an University of Science and Technology, China, in 1982, and the M.S. and Ph.D. degrees in automatic control from the China University of Mining and Technology, China, in 1993 and 1998, respectively. He was a Post-Doctoral Fellow at Nanjing University, China, from 1998 to 2000, and Michigan State University, USA, from 2001 to 2002, and a Visiting Scholar at The Hong Kong Polytechnic University in 2007. His current research interests include distributed computing and ubiquitous network.