This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2019.2917444, IEEE Transactions on Mobile Computing

Implementation of Differential Tag Sampling for COTS RFID Systems

Xin Xie, Xiulong Liu, Xibin Zhao, Weilian Xue, Bin Xiao, *Senior Member, IEEE*, Heng Qi, *Member, IEEE*, Keqiu Li, *Senior Member, IEEE* and Jie Wu, *Fellow, IEEE*

Abstract—Tag inventory is one of the most fundamental tasks for RFID systems. However, the Framed Slotted Aloha (FSA) protocol specified in C1G2 standard is of low time-efficiency, because it needs to collect all tags in the system. To improve time-efficiency, research communities proposed a batch of sampling-based approaches, in which the reader only needs to collect a small set of sampled tags instead of all. Although time-efficiency has been improved, existing sampling-based approaches still have two common limitations. First, all tags in the system are assumed to have the same sampling probability. It is unfair that tags attached to differential items (*e.g.*, different values) have the same chance to be sampled and collected. Second, all existing sampling-based approaches stay in theory level and cannot be deployed on Commercial Off-The-Shelf (COTS) RFID devices, because the C1G2 standard does not support the sampling function at all. To deal with the above two limitations, this paper studies the new problem of *differential tag sampling*—letting each RFID tag be identified with a given sampling probability. In this paper, we use the COTS RFID devices including Impinj Speedway R420 reader and Monza 4QT tags to implement the Differential Tag Sampling (DTS) operation. Then, we apply probabilistic analytics on the collected tag data to address some practically important problems such as Multi-category Tag Cardinality Estimation (MTCE), and Value-based Missing Tag Detection (VMTD). Although the analytics results are not 100% accurate, the deviation in the results can be controlled below a small threshold and DTS can significantly improve the time-efficiency. DTS can be easily deployed on the COTS RFID systems, because it is totally compliant with the C1G2 standard. Extensive experiments demonstrate that DTS is able to let each tag take the given sampling probability to be sampled and identified. Moreover, the proposed DTS protocol can significantly reduce the execution time of MTCE and VMTD by nearly 70% than the FSA protocol.

Index Terms—RFID, Tag sampling, C1G2, Tag identification, Tag cardinality estimation

1 Introduction

Radio Frequency Identification (RFID) technique has reformed smart industry by acting as the eyes and ears of IoT. Millions of tags have been deployed in a wide spectrum of notable applications, *e.g.*, intelligent supply chain, unmanned supermarket, and airport [1]–[6]. For, example, in the area of intelligent supply chain, tags have been attached to various items including food [7], [8], clothes, cash and medicament [9]. RFID readers are able to automatically read the information embedded in these tags, and then forward the gathered information to server. Thus, these tagged items can be tracked in time by retailers and product suppliers. With real-time inventory data of the items, retailers can quickly adjust marketing strategy, and suppliers can efficiently arrange product planning [10].

- *X. Xie, H. Qi and K. Li are with the School of Computer Science and Technology, Dalian University of Technology, Dalian, Ganjinzi 116026, China. E-mail:{xiexin,likeqiu}@gmail.com, hengqi@dlut.edu.cn.*
- *X. Liu and B. Xiao are with the Department of Computing, the Hong Kong Polytechnic University, Hong Kong. E-mail: xiulongliudut@gmail.com, csbxiao@comp.polyu.edu.hk.*
- *X. Zhao is with the Key Laboratory for Information System Security, Ministry of Education (KLISS), Tsinghua National Laboratory for Information Science and Technology (TNList), School of Software, Tsinghua University, Beijing, Haidian 100084, China. E-mail: zxb@tsinghua.edu.cn.*
- *W. Xue is with the School of Management, Liaoning Normal University, Dalian, Shahekou 116026, China E-mail: xueweilian@lnnu.edu.cn.*
- *J. Wu is with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122. E-mail: jiewu@temple.edu.*

Corresponding authors: Xiulong Liu and Xibin Zhao

EPC Class-1 Generation-2 (C1G2) standard specifies that RFID devices use the Framed Slotted Aloha (FSA) protocol as the MAC layer communication mechanism. Specifically, the reader initializes a time frame that contains multiple time slots. Each tag in the reader's communication vicinity randomly selects a slot from the slotted time frame to reply its ID to the reader. Generally, there are three types of slots: *empty slot* in which no tag replies; *singleton slot* in which only one tag replies its ID; *collision slot* in which two or more tags simultaneously reply their IDs. The reader can only successfully identify tag IDs in singleton slots. The tags reply in singleton slots will keep silent; and the tags reply in collision slots will continue to participate in the next round of time frame. Such processes repeat until all tags are successfully identified. However, FSA suffers from low time-efficiency. It has been proved that the ratio of singleton slots in a time frame achieves is upper bound of 36.7% when the number of time slots is equal to the number of tags [11]. Since execution time of FSA is proportion to the number of tags, it cannot satisfy time-stringent application scenarios.

For time-efficiency, research communities proposed a batch of sampling-based approaches to address some practical problems in a *probabilistic* manner, *e.g.*, missing tag detection [12]–[15], and tag cardinality estimation [16]–[18]. In these approaches, the authors assumed that RFID tags are capable of computing a *sampling* function, and will participate in the process of detection or estimation with a given sampling probability. The reader only needs to collect data from a small set of sampled tags (instead of all tags), and perform probabilistic analytics on the collected

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2019.2917444, IEEE Transactions on Mobile Computing

data to discover the theft event or estimate the number of tags with a required confidence. Although these probabilistic approaches cannot obtain 100% accurate results, they significantly reduce the time cost than the FSA protocol. Moreover, the deviation in the results is usually limited within a predefined threshold.

However, the existing sampling-based approaches have two common major limitations. First, all tags in the system have the same sampling probability, which leads to unfairness among differential tags. For example, in practical RFID systems, tagged items are normally differential according to various metrics such as different item values, item categories, or expiry date. Generally, each tag should have a given sampling probability to be identified in the inventory process, and the sampling probability depends on what item this tag is attached to, *e.g.*, the tags attached on expensive items should be identified with large sampling probabilities, because they usually cause more serious economic loss if they are stolen. Second, all existing sampling-based approaches only stay in theory level. They are not able to be deployed on Commercial Off-The-Shelf (COTS) tags.

To overcome these two limitations, this paper studies the problem of *differential tag sampling*, which is a totally new problem and was never touched previously. Specifically, we study how to let each RFID tag be identified with a given sampling probability in the inventory process. This paper designs and implements Differential Tag Sampling (DTS) operation for C1G2-compliant RFID systems [19]. Our design strictly follows the specification in C1G2 standard [20], so it can work in any C1G2-compliant RFID systems. The proposed DTS protocol consists of two sub-operations: *assignment* and *sampling*. Specifically, the assignment operation is to assign a specific sampling probability into a target tag. It is implemented through writing a binary string into the tag's nonvolatile memory with C1G2-compatible commands. The proportion of '1's in the binary string should be equivalent to the given sampling probability. On the other hand, the sampling operation is to sample tags with the selective reading. The selective reading checks a random bit of the binary string and selects the tags whose corresponding bit equals to '1'. Thus, the tags can be sampled with the probability defined by the binary string, because the sampling probability is also equivalent to the proportion of '1's in the binary string stored in the tag.

The main contributions are summarized as follows.

- This paper takes the first step to study the new problem of differential tag inventory, and designs the Differential Tag Sampling (DTS) operation based on selective reading for C1G2-compliant RFID systems. We turn the unrealistic assumption in previous work, *i.e.*, RFID tags are able to perform sampling operation, into reality.
- We use Impinj R420 reader and Monza 4QT tags to implement a prototype to evaluate the performance and effectiveness of our DTS. With DTS, only a small set of tags, which we desire to collect with different probabilities, will be sampled to participate in the tag inventory process.
- Extensive experimental results reveal that DTS is able to let RFID tags take the given sampling probabilities to be identified in the inventory process. Moreover, benefiting from our DTS protocol, the representative RFID inventory protocols, *e.g.*, multi-category tag cardinality estimation and value-based missing tag detection, can reduce the execution time by nearly 70% the classical FSA protocol.

The remainder of paper is organized as follows. Section 3 presents the motivation and problem formulation of this paper. Section 4 presents the detailed design of DTS. Section 5 describes two practical applications of DTS. Section 6 gives a brief introduction to our prototype. Section 7 evaluates the performance of DTS and related applications through extensive experiments. We review the related work in Section 2 and conclude this paper in Section 8.

2 Related Work

In this section, we classify the related protocols into two types: tag identification protocols and tag grouping protocols, which will be discussed in detail below.

Identification Protocols: Existing tag identification protocols are based on either Frame Slotted Aloha (FSA) mechanism [21]–[23] or Binary Splitting (BS) mechanism [24]. In FSA-based protocols, a time frame is divided into multiple time slots. Each tag randomly selects a slot to send its data. A key problem in the FSA-based protocols is to optimize the frame size to maximize the frame utilization. Many protocols with various optimization strategies have been proposed. Q protocol that is adopted by the C1G2 standard [20] is the most popular one. On the other hand, in BS-based protocols *e.g.*, Tree Working (TW) [25], the reader first queries a prefix '0' and then listens to the tag responses. If more than one tags are sensed, the reader recursively split the tag population into two smaller groups by broadcasting a longer prefix. Until each group has one or none tag, the reader can successfully parse the tag response. However, this process causes too many tag collisions. To overcome this problem, the Smart Trend Traversal (STT) [26] protocol uses some ad-hoc heuristics to adjust the prefixes based on the historical tag responses. The Tree Hopping (TH) [24] protocol optimizes the number of queries based on solid theoretical analysis. Moreover, by combining the advantages of both Binary Splitting (BS) and Framed Slotted Aloha (FSA) approaches, BSTSA protocol [27] further improves the identification efficiency. BSTSA applies the BS protocol to split tags into different groups and identify each group of tags with FSA protocol. Since we can estimate the number of tags at the end of each group identification, the frame size can be optimized to improve the efficiency of FSA protocol. However, FSA has a limited throughput even with the optimized frame size. Buzz [28], a concurrent identification scheme, treats the corrupted information received in a collision slot as a sparse code across the bits transmitted by the tags, and decodes the transmitted data using the compressive sensing algorithm. The efficiency is improved because the tags can concurrently transmit their data in collisions slots. However, Buzz assumes the slot chosen by each tag is known by the reader, thus it can only be implemented using the programmable RFID devices (*e.g.*, WISP tags and USRP readers). For better deployability, our scheme uses C1G2-complaint FSA as the MAC layer protocol, because FSA is the anti-collision protocol supported by the EPC C1G2 standard.

Grouping Protocols: The problem of tag grouping is to classify tags into groups, and then operate group operations

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2019.2917444, IEEE Transactions on Mobile Computing

for saving time. The previous work tries to minimize the execution time to assign each tag a specific group ID. The straightforward solution is to write the group IDs to tags one by one using a simple polling approach [29]. However, polling operation incurs a long execution time. The ConCurrent Grouping (CCG) protocol [30] improves the time-efficiency by using a bit vector to tell tags when and where they will be assigned with correct group IDs. Compared with polling IDs, transmission of lightweight bit vector significantly reduces the communication overhead. However, it requires RFID tags to interpret a bit vector, which is not supported by the C1G2 standard.

EPC Filter-based Sampling: EPC is a 96-bit global unique ID assigned by the manufacturers, which acts as the barcode to store some business data related to the tagged item. Before the tag identification phase, a EPC filter can be added to allow the reader only to read the tags with the given pattern of EPC. This filter is generally used in the Query tree protocols to resolve tag collisions. In theory, the tag sampling operation can also be implemented using the EPC filter. The controller needs to determine the sampled tags off-line and find a proper pattern shared by all the EPCs of sampled tags. Compared with EPC filter-based tag sampling, our DTS scheme has two significant advantages. First, making use of USER memory, DTS provides flexibility to assign each tag a specific sampling probability. While it is hard to sample tags with EPC filter due to the difficulty in finding a proper EPC pattern that exactly matches with all the sampled tags. Second, our DTS has lower computational complexity. It only needs to pick a random number to match the sampled tags, which incurs $O(1)$ computational complexity. While the EPC filter approach takes $O(n * m)$ computational complexity to find the common pattern shared by all the sampled tags, where n denotes the number of tags in the systems and m denotes the length of an EPC ID.

3 Motivation and Problem Formulation

We first review the Frame Slotted Aloha (FSA) protocol used for RFID communication, and then explain why existing sampling-based protocols are attractive but not applicable for COTS RFID systems. Finally, we will formally give the problem formulation of this paper.

3.1 Motivation

3.1.1 Framed Slotted Aloha

An RFID system typically consists of readers and tags that can communicate with each other by radio waves. A reader needs to receive data from multiple tags, while the tags are unable to coordinate their radio transmission to avoid collisions. According to the C1G2 standard [20], an RFID reader should implement the Framed Slotted Aloha (FSA) protocol to resolve the tag collisions for multi-tag access. In FSA, the reader initiates a time frame that contains f slots. Each tag within the reader's communication range will randomly selects a time slot to respond to the reader with its tag ID. We can classify slots into three categories: the *empty slot*, where no tag reply; the *singleton slot*, where only one tag replies; the *collision slot*, where multiple tags reply. Multiple tags may choose the same slots, which leads to signal corruption, and thus the reader cannot receive any ID of the collided tags. Only the tags in singleton slots can be collected by the reader. However, it has been proved that the fraction of singleton slots in time frame is always below 36.7%. The low frame utilization results in a long inventory time when the number of tags is large. For example, it takes several minutes to collect data from thousands of tags, which is hard to meet the restrict operation time delay in the large-scale RFID systems.

3.1.2 Sampling-based RFID Protocols

Time-efficiency is a key metric for RFID applications. Both retailers and product suppliers prefer real-time and accurate inventory to adjust marketing strategies and arrange product replenishment. Unfortunately, achieving real-time inventory is an arduous job in large-scale systems because of the contradiction between low RFID communication rate and the large tag population. To keep cheap, RFID tags can only have weak hardware. Thus, it is not easy to fundamentally improve the communication rate. Therefore, the research communities turn to design a wide spectrum of sampling-based protocols [12], [31], [32], in which the reader only needs to collect data from a small set of sampled tags. Then, they apply statistical principle to fast derive some valuable information. For example, in an RFID warehouse that contains multiple categories of items, there are two typical applications that are badly in need of tag sampling, *i.e.*, multi-category tag estimation and value-based missing tag detection. First, multi-category tag estimation is to estimate the tag cardinality in each category, which helps the retailer to make the sales and market strategy. Second, value-based missing tag detection is to report the theft warning message to the retailer if the value of missing items in any category exceeds a pre-defined threshold. Although the above two protocols are practically important in RFID-enabled warehouse, none of them is implemented on COTS RFID systems due to the lack of tag sampling operation.

3.2 Problem Formulation

In this paper, we study the problem of Differential Tag Sampling (DTS) for C1G2-compliant RFID systems. We follow three goals when designing the DTS protocol: (i) DTS should permit users to conveniently set and revise the sampling probability of each tag. This flexible design allows DTS to fit a wide spectrum of sampling-based applications. (ii) DTS should be compliant with the C1G2 standard [20]. Therefore, DTS can only issue C1G2 commands. This requirement makes DTS be easily deployed on COST RFID devices without any need of hardware modifications. (iii) DTS should be time-efficient, and only incurs quite limited communication and computation complexity; otherwise, the time-efficiency improvement brought by collecting less tags may be overwhelmed by overhead.

4 Implementation of DTS

In this section, we will present the technique details of our DTS design. DTS applies the C1G2-compliant selective reading to implement a sampling functionality [20], thus can be directly deployed on COTS RFID readers and tags.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2019.2917444, IEEE Transactions on Mobile Computing

TABLE 1: Fields of *write* command

Fields	# of bits	description
MemBank	2	the target MemBank to write data
WorldPtr	8	address pointer
Data	16	data message
RN	16	used for memory access
CRC	16	used for error checking

TABLE 2: Fields of *select* command

Fields	# of bits	description
Target	3	modify **SL** flag or **inventoried** flag
Action	3	action defined in Table 6.30 of [20]
MemBank	2	the target MemBank to apply the Mask
Pointer	EBV	starting bit address of the Mask
Length	8	bit length of the Mask
Mask	Variable	a bit string starting at Pointer
Truncate	1	whether a tag reply shall be truncated
CRC	16	CRC-16 used for error check

4.1 Background of C1G2 Specification

First of all, we introduce three basic C1G2 functions [20] that are employed by us when designing DTS.

4.1.1 Tag Memory

C1G2 specifies a simple tag memory model in the section of logical interface (see pages 44-51 in [20]). Each tag contains four non-volatile memory banks: (1) **Reserved memory** is reserved for password associated with the tag, which is used for verifying sensitive commands like *kill* and *access*. (2) **EPC memory** stores the EPC ID, which is a 96-bit unique identifier and offers the similar functionality as the barcode printed on the item. It tells us the manufacture, category and serial number of the tag-attached item. (3) **TID memory** stores TID that defines the specification and supported functions of the tag. (4) **USER memory** provides storage for user-defined data. It is usually partitioned into one or more files. We can use C1G2 commands *write* or *read* to read or write one or several 16-bit words from or to these memory banks. In this paper, we only concern two memory banks: EPC memory identifies the tagged item, and helps us to determine the sampling probability based on the features of the tagged items; USER memory stores sampling probability of the tag, the reader simulates tag sampling by selecting tags with the specific content on their USER memory.

4.1.2 Write Command

C1G2 specifies the *write* command to store data on the tag memory. DTS applies it to store the sampling probability on tag's user memory bank. Each *write* command can write 16-bit data message on one of the memory banks. We may issue a group of *write* commands to store a long data message of more than 16 bits. Each *write* command comprises five fields as shown in Table 2. The fields related to this study are listed as follows.

• **MemBank** and **WordPtr**: These two fields are combined to define the target location to store the data message. MemBank specifies the target memory bank among Bank 00, 01, 10 and 11. WordPtr specifies the target address of the MemBank for storing the data message.

• **Data**: This field defines the data message that should be written on the tag.

4.1.3 Select Command

C1G2 specifies that *select* command to choose a certain group of tags for the upcoming tag inventory. In particular, each tag maintains a selected flag **SL** (see page 52 in [20])), and the *select* command will turn the SL flag of matched tags to **asserted** and unmatched tags to **de-asserted**. Only the asserted tags keep active and respond to the reader in the upcoming reading cycle. We can perform a group of *select* commands to obtain a logic combination of multiple subsets. Each *select* command comprises eight fields as shown in Table 2. The fields related to this study are listed as follows.

• **Target** and **Action**: These two fields are combined to define the "side effect" of the *select* command. In particular, it specifies how to modify the SL flag of the tag. In our application, we set Target to 100 and Action to 000, which is equivalent to turn the SL of the matched tags to asserted and unmatched tags to deasserted.

• **MemBank** and **Pointer**: These two fields are combined to define the starting bit address of the tag memory that is used to compare with the Mask filed embedded in *select* command. MemBank specifies the target memory bank. Pointer uses the EBV formatting (see page 115 in [20]) to specify the starting bit addressing of the corresponding memory bank.

• **Mask** and **Length**: These two fields are combined to define the mask, which is a binary string that a tag compares to a memory location, starting from Pointer and ending at Length bits later. The Length filed is 8-bit, so that the Mask should be a binary string varying from 1 to 255 bits.

4.1.4 Selective Reading

DTS simulates tag sampling with selective reading, which selects a subset of tags for participating the upcoming operations. Selective reading can be achieved by performing three basic operations: *select*, *inventory* and *read* in sequence. Followed by a group of *select* commands, the RFID reader issues the *inventory* command (see pages 76-80 in [20]) to resolve collisions between asserted tags. The reader initializes a time frame of f slots by issuing a *query* command. Each asserted tag replies in a randomly chosen slot. As we previously mentioned, only the tags in singleton slots can be successfully identified by the reader. In these slots, the reader can further issues *read* commands to collect data stored on tag memory (see pages 81-84 in [20]).

4.2 Detailed Design of DTS

We design DTS to select each tag with a user-defined sampling probability. The proposed DTS is totally built upon the selective reading, and does not require custom functions such as hash functions or comprehension of binary string received from the reader. DTS mainly consists of two sub-operations: *assignment* and *sampling*. We use assignment operation to write a binary string on each tag for recording the sampling probability. We use *sampling* operation to select tags based on the binary string stored on tag memory. Our scheme is able to ensure that the tags will be selected with expected probabilities. In the following, we will present the two operations in detail.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2019.2917444, IEEE Transactions on Mobile Computing

TABLE 3: *select* command in assignment operation

Fields	# of bits	message	description
Target	3	100	**SL** flag
Action	3	000	**asserted** if mask matched
MemBank	2	01	EPC memory
Pointer	8	00100000	address 0x20
Length	8	01100000	96-bit
Mask	96	ID	choose the target tag
Truncate	1	0	disable truncate
CRC	16	CRC-16	used for error checking

TABLE 4: *write* command in assignment operation

Fields	# of bits	message	description
MemBank	2	11	USER memory
WorldPtr	8	00000000	address pointer
Data	16	binary string	responding probability
RN	16	handle	used for memory access
CRC	16	CRC-16	used for error checking

4.2.1 Assignment Operation

The *assignment* operation is to assign a specific sampling probability to a tag. For each tag, the reader needs to write a binary string on it. This binary string needs to reflect the sampling probability, and ensures that the tags can be sampled with the expected probability in the following sampling operation. By default, DTS uses the fraction of '1' bit in the binary to represent the sampling probability. For example, if the number of '1' bit in a 10-bit binary string is 5, the corresponding sampling probability is $1/2$. Although some advanced scheme like IEEE 754 can represent the floating number in a space efficient way, it is hard to use selective reading to resolve the probability in IEEE 754 formatting. To transform the sampling probability p to a k-bit binary string, we first compute the number of '1' bit (denoted as $n1$) in the string by $n1 = \text{round}(p \times k)$. Then, we set the leading $n1$ bits in the binary string to '1'. Finally, we randomly shuffle the binary string to obtain a new binary string that contains $n1$ '1'. Due to the shuffle operation, two tags of the same sampling probability can be assigned two different random string, which makes sure that they can be sampled at different inventory rounds. For instance, when $k = 10$, two tags of probability $p = 0.3$ can be assigned two different binary strings *1000001001* and *1100000010*. Hence, tag1 is sampled if the selected bit is 1,7 or 10; otherwise, tag2 is sampled if the selected bit is 1,2 or 9.

To write each binary string to the target tag, the reader needs to issue three types of C1G2 commands: *select*, *query* and *write* in sequence.

select command is used to select the tag with the target ID. The detailed fields of *select* command are shown in Table 3, which has a total length of 117 bits. Specifically, the MemBank filed is set to EPC memory (01) and the Pointer filed is set to oX20, which is combined to present the address of EPC ID. The Mask is set to the EPC ID of the target tag, thus only one target tag is asserted to reply to the reader.

query command starts a time frame to identify the asserted tag. Since there is one tag at most, the Q field is set to 0000, meaning a time frame that contains only 1 slot. In this slot, the reader communicates with the tag following the steps detailed in *Appendix E* in [20], and obtains a *handle* for accessing tag memory.

TABLE 5: *select* command in sampling operation

Fields	# of bits	message	description
Target	3	100	**SL** flag
Action	3	000	**asserted** if mask matched
MemBank	2	11	USER memory
Pointer	8	variable	starting address
Length	8	00000001	1-bit
Mask	1	1	1-bit '1' mask
Truncate	1	0	disable truncate
CRC	16	CRC-16	used for error checking

write command writes the binary string to the USER memory of the target tag once the reader obtains the handle. The *write* command comprises five fields as shown in Table 4. Since the binary string should be written to the USER memory, the MemBank filed to 11. We assume the front address of USER memory is left blank for storing the sampling probability, so that the WorldPtr Points is set to $0\text{x}00$, which points to the first 16-bit word of USER memory. Each *write* command can write 16-bit word on tag at most. If the binary string to be written is longer than 16 bits, we have to issue multiple *write* commands to store it to the tag. Each of them have the same fields except WorldPtr, which should be moved forward and is set to $0\text{x}01, 0\text{x}02, \cdots$ in the following *write* commands.

This assignment operation is only performed on the tags, which are newly moved into the system or the ones whose sampling probabilities need to be changed. Hence, we do not need to frequently perform this operation.

4.2.2 Sampling Operation

The *sampling* operation is to sample tags with the preassigned sampling probability. The basic idea is to issue a *select* command to sample tags by matching the binary string stored in the tag memory. The issued *select* command checks a random bit of the binary array. If the bit specified by the *select* command equals to '1', the tag is sampled and will respond to the reader in the upcoming session; otherwise, the tag will keep silent until it is sampled by the future *select* command.

The fields of the *select* command are shown in Table 5. It has three different fields compared with the commands issued in the assignment operation. First, the MemBank field is point to the USER memory (11), where it stores the probability binary string. Second, the Pointer filed is a random variable generated by the controller, which simulates the random sampling process. Suppose the length of the binary string is k bits, the random variable should be an integer uniform distributed between 0 to $k-1$. Third, the Mask field used for comparing with the binary string is one bit '1'. Thus, the Length filed is also set to $0\text{x}01$. The total length of this *select* command is only 42 bits, which is much smaller than the previous one.

4.3 Implementation Detail

In what follows, we will discuss the details when implementing the DTS protocol, including *Pointer Filed*, *Array Length*, and *Probability Array*. We also will analyze the DTS performance under different implementations, which provides a guidance on the selection of algorithms and parameters.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2019.2917444, IEEE Transactions on Mobile Computing

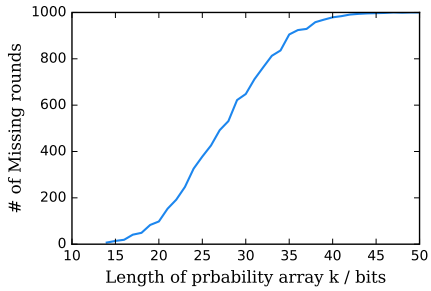


Fig. 1: The number of rounds which contain missing bit *vs.* varying k

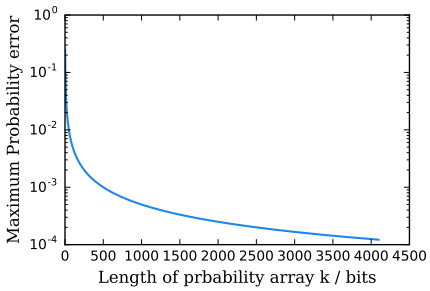


Fig. 2: The maximum error of sampling probability *vs.* varying k

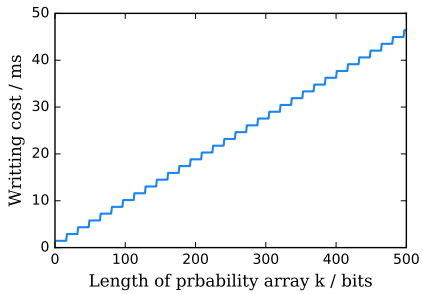


Fig. 3: The time cost of writing the probability array on tag *vs.* varying k

4.3.1 Pointer Field: Random vs. Shuffle

During each sampling operation, the controller needs to generate a random integer as the Pointer field of the *select* command. A straightforward solution is to apply uniform random integer generator to obtain an integer every time before issuing the *select* command. This simple approach ensures that each bit can be chosen with the same probability $1/k$ but has a major deficiency that some bits may not be sampled even after many rounds of sampling operations. For a certain bit in the bit array, the probability that it is missed after i rounds can be represented as:

$$p_m = \left(\frac{k-1}{k}\right)^i \tag{1}$$

To clearly present the *miss-sampling* probability, we simulate the random sampling with varying values of k. For each k, we first conduct 100 times of sampling operations, and then check if any missing bits exist. We independently repeat each simulation for 1000 times and present the results in Fig. 1. We observe that miss-sampling is a common phenomenon even when k is relatively small. For example, we may miss some bits of the 20-bit array with a probability of 10% even after 100 sampling operations are done. The tags with a small sampling probability may have only one '1' bit in its probability array, if that bit happens to be the missing bit, this tag cannot be sampled after a very long time. We call this as the *tag starving issue*.

To address the tag starving issue, we propose an effective shuffle scheme. Specifically, we first generate an integer list ranging from 0 to $k-1$. Then, we shuffle the list to get a list with integers in a random order. Next, we loop through the integers in the shuffled list and apply them as the Pointer field of the *select* command. Once the cursor reaches the end of the list, we shuffle the list again and move the cursor return to the beginning of the list. With the shuffle scheme, miss-sampling probability after i rounds execution becomes:

$$p'_m = \left(\frac{k-i}{k}\right) \tag{2}$$

Thus, each bit can be sampled at least once every k times of sampling operations. However, it should be noted here, the maximum sampling interval in the shuffle scheme is $2k-2$ instead of k. The worst case happens when an integer at the head of the list is shuffled to the tail of the next list. Therefore, when $k <= 51$, all the bits are sampled at least once after 100 rounds of sampling operations, which significantly outperforms the straightforward random sampling.

4.3.2 Array Length: Accuracy vs. Space Cost

The length of bit array trades off between sampling accuracy and memory space occupation. Recall that $n1 = \mathrm{round}(p \times k)$, the gap between the actual sampling probability $p' = \mathrm{round}(p \times k)/k$ and desired sampling probability can be up to $\frac{1}{2k}$. The sampling accuracy is *inversely* related to k. Although a large k improves the accuracy of sampling probability, it also consumes much more time for writing the sampling probability to a tag since each *write* command can write 16-bit data at most. Thus, we have to choose a proper k according to the specific RFID applications.

The maximum k is limited by the size of User memory bank. For example, a Monza 4QT tag [33], a COTS RFID tag used in this paper, has 512 bytes User memory. Theoretically, the binary string used for representing sampling probability can be up to 4096 bits. It supports a quite high accurate sampling probability with a offset no more than 1.22×10^{-4}, which is accurate enough for most sampling-based applications. To balance between the space and accuracy, we limit the maximum k to 500, which only takes at most $1/8$ of the total USER memory of Monza 4QT tag, meanwhile providing highly accurate sampling probability with an offset of just 10^{-3}. We set this limit due to two major reasons: First, it is not wise to cost most of the tag memory for a single operation. Second, increasing k fails to bring significant accuracy improvement when k is large enough. As shown in Fig. 2. When $k = 10$, the increase of k will significantly reduce the offset in storing sampling probability. However, when k is relatively large (*e.g.*, 1500), further increasing k achieves little improvement on the sampling accuracy.

Since each *write* command can only write 16-bit word on memory, we may need to cut a long *probability array* into multiple pieces and use a series of *write* commands to write these pieces into the tag. The time cost of assignment operation is in proportion to the array length k. Suppose the RFID transmission rate between reader and tags are 40 kb/s, it takes 25 us to transmit a bit. Since the length of a *write* command is 58 bits, the time cost for issuing such a command is 1.45 ms. The total time cost is a step function as shown in Fig. 3. To make full use of the *write* command, we prefer to set $k = 16n$, where n is a positive integer. In conclusion, the length k of *bit array* should be a multiple of 16 and smaller than 500 at the same time.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2019.2917444, IEEE Transactions on Mobile Computing

4.3.3 Probability Array: Space vs. Randomness

In this section, we present another probability array, which can represent highly precise sampling probability with fewer bits. In the origin DTS implementation, we use k bits array to represent the sampling probability. All bits in the probability array have the same weight, each of which represents a probability unit of $1/k$. The sampling probability within the range $[w/k - 1/2k, w/k + 1/2k)$ is rounded to w/k, resulting a maximum $1/2k$ gap between the actual and expected sampling probability. It takes too much bits to represent the sampling probability in high precision as shown in Fig. 2.

To reduce the array size for highly precise probability, we can assign each bit with a distinct weight following the negative exponential distribution. The first bit has the largest weight, which represents a probability unit of $1/2$. The weight of next bit is half of that of the previous bit. Therefore, the weight of the i^{th} bit in the probability array is $1/2^i$. The only exception is the last bit of the array, whose weight equals to the previous bit. This is to ensure that the sum of all the bits' weights equals to 1. We can combine these bits to obtain any sampling probability that is a multiple of $1/2^{k-1}$. Let p_m denote the maximum probability error that we can tolerate, the required array size could be calculated as follows:

$$k = \left\lceil \log_2 \left(\frac{1}{p_m} \right) \right\rceil \tag{3}$$

Compared with the original method, the new method needs a probability array of $\lceil \frac{1}{2p_m} \rceil$ bits, meaning that it significantly reduces the array size from linear to logarithmic. To be compliant with new array design, the pointer generation algorithm used in sampling operation should also follow the exponential distribution. Specifically, the index $i(0 \leq i \leq k-2)$ can be chosen with a probability of $1/2^{i+1}$.

5 Practical DTS Applications

In this section, we will present two typical RFID applications, and explain how to use our DTS to facilitate them.

5.1 Multi-category Tag Cardinality Estimation

The problem of Multi-category Tag Cardinality Estimation (MTCE) is to quickly obtain the approximate number of tags in each category, without collecting all tags. MTCE is practically important, especially in a large-scale RFID-enabled warehouse with thousands of categories. In what follows, we formally give the definition of the MTCE problem.

Problem 1. *Given a set of RFID tags with categories $C_1, C_2, \cdots, C_\lambda$, whose tag cardinalities are unknown and denoted as $n_1, n_2, \cdots, n_\lambda$, a tolerance of $\beta \in (0,1)$, and a required confidence level $\alpha \in (0,1)$, we desire to estimate the number of tags in each category such that $Pr(|\hat{n}_i - n_i| \leq \beta n_i) = \alpha$, where $\hat{n}_i$ is the estimation result of n_i and $i \in [1, \lambda]$.*

A straightforward method is to collect the IDs of all tags. After that, we can easily classify tags into different categories, and thus naturally know the exact number of tags in each category. However, this solution is too time-consuming, especially in large-scale RFID systems. To this end, we apply our DTS to propose a time-efficient probabilistic estimator to address the problem of MTCE.

Probabilistic Estimator: We use DTS to assign each tag a distinct sampling probability based on the category it belongs to, where p_i is used to denote the sampling probability corresponding to category C_i. After running the FSA protocol in C1G2 standard to collect the sampled tags' IDs, we could obtain the number of sampled tags that belong to category C_i, which is denoted as X_i. Obviously, we have $X_i \in [0, n_i]$, and X_i follows a standard Binomial distribution with the parameters n_i and p_i. The probability that X_i tags among n_i in category C_i are sampled can be calculated as follows.

$$Pr(X_i) = \binom{n_i}{X_i} \times p_i^{X_i} \times (1 - p_i)^{n_i - X_i} \tag{4}$$

The expectation of X_i is $\mu = n_i p_i$, and the variance of X_i is $\delta^2 = n_i p_i (1 - p_i)$. Based on the expectation, we can calculate the estimation of n_i by following equation.

$$\hat{n}_i = X_i / p_i \tag{5}$$

We have the confidence interval of the estimator as follow:

$$Pr\left(\hat{n}_i - \frac{z_\alpha \delta}{p_i} \leq n_i \leq \hat{n}_i + \frac{z_\alpha \delta}{p_i} \right) = \alpha, \tag{6}$$

Where z_α is a percentile of α, which can be looked up in the binary distribution table. For example, $\alpha = 95\%$, z_α will be 1.96. To meet the tolerance requirement, the standard deviation should satisfy the following constraint:

$$\delta \leq \frac{\beta n_i p_i}{z_\alpha} \tag{7}$$

By equating the constraint, we can derive the sampling probability to ensure the estimation accuracy within the given tolerance level β with a confidence level α:

$$p_i = \frac{Z_\alpha^2}{\beta^2 n_i + Z_\alpha^2} \tag{8}$$

According to the above equation, we find that p_i depends on the number n_i of tags in category C_i. A large category should be assigned with a small sampling probability for reducing the time cost of collecting sample tags, and vice versa. We can use our DTS to easily achieve this. However, in traditional sampling methods, all tags have the same sampling probability regardless of large-size categories or small-size categories. Thus, to ensure the estimation accuracy for small-size category, they need to set a large sampling probability, which, however, is not time-efficient for large-size category. Clearly, assigning different sampling probabilities to differential tags is also a significant advantage of our DTS than the previous sampling methods. In what follows we will explain how to set the sampling probability p_i without knowing n_i. In practice, inventory process should be frequently performed for timely monitoring the stock. At the very beginning, we can set a value p_i according to the experience on the tag cardinality (*e.g.*, the number of items in a category should be with a range). In the following rounds of inventory, we can use the estimation results $\hat{n}_i$ obtained from previous round of estimation to determine p_i, due to the fact that tag cardinality between two adjacent rounds of inventory does not change too much.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2019.2917444, IEEE Transactions on Mobile Computing

5.2 Value-based Missing Tag Detection

The problem of Value-based Missing Tag Detection (VMTD) is to assign the channel resource to tags according to their values instead of equal assignment. Specifically, the tags attached to expensive items should obtain more channel resource than the tags attached to cheap items. Thus, the expensive ones could have a larger chance to be collected for timely theft reporting. Such operation is practically important for RFID system because we normally need to pay more attention to the expensive items, which may result high economic loss if some of them are stolen. The problem of TVS is formally defined as follow:

Problem 2. *Given a set of RFID tags with categories $C_1, C_2, \cdots, C_\lambda$, each tagged item in category C_i has the value of v_i, the threshold of missing tags' value $\mathcal{T}$, the confidence level α, we desire to quickly report the missing tag event if the total value of missing items in any category exceeds $\mathcal{T}$ with the confidence α.*

We assume that the EPC IDs and categories of all tags in the system are known in advance. This assumption is reasonable and necessary, because it is impossible for us to detect missing tag event if we do not know what tags should be present in the system. The basic idea is to compare the actually collected sampled tags with the expected ones to detect missing tags. In the inventory process, we are able to know what tags will be sampled because the server has the sampling probability array corresponding to each tag. We need to assign each tag category a suitable sampling probability, which ensures that we can detect at least one missing tag when the value of missing items in the category exceeds $\mathcal{T}$. The tolerance number of missing tags in category C_i is $m_i = \lceil \frac{\mathcal{T}}{v_i} \rceil$. Assume exactly m_i tags in category C_i are absent from the system. Let X denote the number of detected missing tags. The probability that we can detect the missing tag event, *i.e.*, $X \geq 1$, can be calculated as follows.

$$Pr(X \geq 1) = 1 - (1 - p_i)^{m_i} \tag{9}$$

To ensure the required detection accuracy, we have to meet the following constraint:

$$(1 - p_i)^{m_i} \leq \alpha \tag{10}$$

By solving the above equation, we can obtain the required minimum sampling probability:

$$p_i = 1 - \sqrt[m_i]{\alpha} \tag{11}$$

According to the above equation, we conclude that tags in a low-value category should be sampled with a small probability, because we can tolerate that a relatively large number of missing items in it. We can simply apply our DTS to assign each tag a sampling probability based on Eq. (11), which provides a time-efficient and accurate solution to the problem of VMTD for multi-category RFID systems.

6 Prototype Implementation

In this section, we discuss how to implement a prototype to validate the correctness and feasibility of the proposed DTS scheme. Our prototype consists of a laptop PC, a reader and several tags, which are connected follow the sketches shown in Fig. 4. The reader is connected to the PC controller

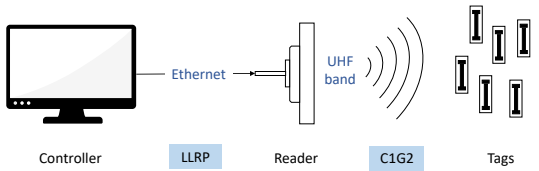


Fig. 4: Connection of main hardware components

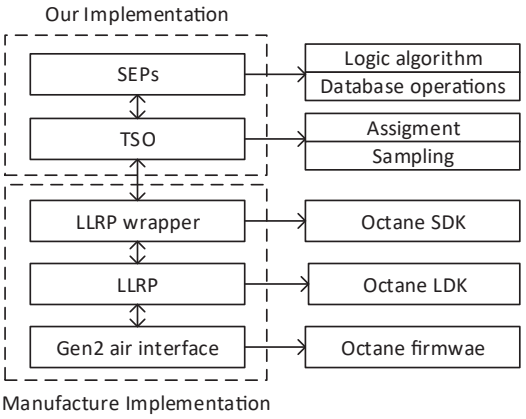


Fig. 5: The architecture of our prototype

through a wired Ethernet cable. The PC leverages LLRP protocol to manipulate a reader to broadcast the desired C1G2 commands. All the LLRP messages are represented in XML format and transmitted to the reader through TCP/IP. The reader communicates with the tags through an Ultra High Frequency wireless channel (900 MHz) following the specification of C1G2 protocol [20].

The architecture of our prototype is shown in Fig 5. The first layer is C1G2 protocol that defines the logical and physical information interactions between reader and tags. This protocol has been implemented on the reader by RFID manufacturer. We use the Impinj Speedway R420 reader [19] with the Octane firmware 5.8. The second layer is the Low-level Reader Protocol (LLRP) which is the communication protocol between software and a reader. The controller applies LLRP to define a sequence of C1G2 operations that need to be executed by the reader. LLRP is also implemented by the RFID manufacturer, thus we can simply import it (*i.e.*, Impinj Octane LTK) when developing our software. Upon LLRP level, some manufacturers also provide high level sdk for easily developing software with high-level languages such as Java and C#. It acts as a wrapper for extracting, modifying the applications of a Reader's Low-Level Reader Protocol (LLRP) settings. With SDK, we can execute C1G2 commands by calling API and avoids the tedious work for defining LLRP messages in XML format. The DTS implementation is a software built upon Octane 2.02 SDK (C#) [34], which provides both assignment and sampling operations implemented by invoking corresponding API. The other applications software can apply the DTS protocol to manipulate tags in the system. Note that, DTS is only responsible for implementing the tag sampling with the given sampling probability. The logic algorithm and database operations should be implemented by the sampling-based applications themselves.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2019.2917444, IEEE Transactions on Mobile Computing

7 Performance Evaluation

In this section, we first conduct real experiments using COTS Impinj RFID devices to validate the effectiveness and feasibility of our DTS. Then, we conduct extensive simulations to evaluate the performance of our DTS when being applied to address the practically important problems of Multi-category Tag Cardinality Estimation (MTCE) and Value-based Missing Tag Detection (VMTD).

7.1 Validate the Effectiveness of DTS

7.1.1 Experiment Settings

We evaluate the proposed DTS by using COTS Impinj RFID reader and passive tags. The reader model is Speedway R420 [19] and the tag model is Impinj Monza 4QT [33]. The reader is connected to a 900 MHz 8 db gain right-hand circularly polarized antenna manufactured by Laird. Besides, the reader is also connected to the controller, a ThinkPad desktop with Intel i5 CPU (2.2 GHz) and 8 GB RAM. The Moza 4QT tags have 512 bytes user memory, which is large enough to store the probability array when implementing DTS.

7.1.2 Assignment Operation

In this set of experiments, we will evaluate the assignment operation in DTS, which is implemented with three main SDK classes, including *TargetTag*, *TagWriteOp* and *TagWriteOpResult*. Among them, *TargetTag* and *TagWriteOp* are the wrapper class of C1G2 *select* and *write*, respectively. To assign sampling probability, our program applies *TargetTag* and *TagWriteOp* to write a probability array to each target tag. *TagWriteOpResult* class is used to catch the writing results on each tag, which is used to count the successful rate of assignment operations.

First of all, we evaluate the successful rate of assignment operation. In our experiments, we use 6 passive tags with distinct IDs, each of them is assigned with a distinct probability. The tags are placed at different distances from the RFID antenna. 2 tags at surface of the antenna, 2 tags at 2 meters distance, and 2 tags at 4 meters from the antenna. We conduct 300 rounds of experiments. During each experiment, we sequentially execute assignment operation to write data to each tag and check whether the operation is successful by catching the results filed of the *TagWriteOpResult* class. We observe from the results in Table 6 that, when the target tags are placed near the antenna, the assignment operation has an extremely high successful rate. However, the rate declines when the distance between tag and antenna becomes larger. The rate gets even worse when the tag is placed on the side direction with a large angle to the front direction. For example, the operation on Tag 6 never succeed. This observation is reasonable because the antenna used in our experiments has different gain on distinct directions [35]. The tag at the side direction fails to capture sufficient energy from the antenna, thus cannot be identified and operated by the reader. In summary, when applying assignment operation to write data on tags that are newly moved into the system, we prefer to place the tags right in the front of the antenna, which is reasonable for new item registration. However, when we want to execute the assignment operation to modify the sampling probabilities on old tags, which have been deployed in the system already, we have to face the potential assignment failure caused by shelf block, out of range and shadowing effect. When a assignment operation fails, we may retry it several times and report a temporary tag/item lost waring to the controller if all the assignment operation are failed.

TABLE 6: Identification rate vs varying distance

Tag	Distance	Direction	Rate
Tag 1	surface	front	100%
Tag 2	surface	side ($\pi/2$)	100%
Tag 3	2m	front	100%
Tag 4	2m	side ($\pi/2$)	92.3%
Tag 5	4m	front	91.7 %
Tag 6	4m	side ($\pi/2$)	0 %

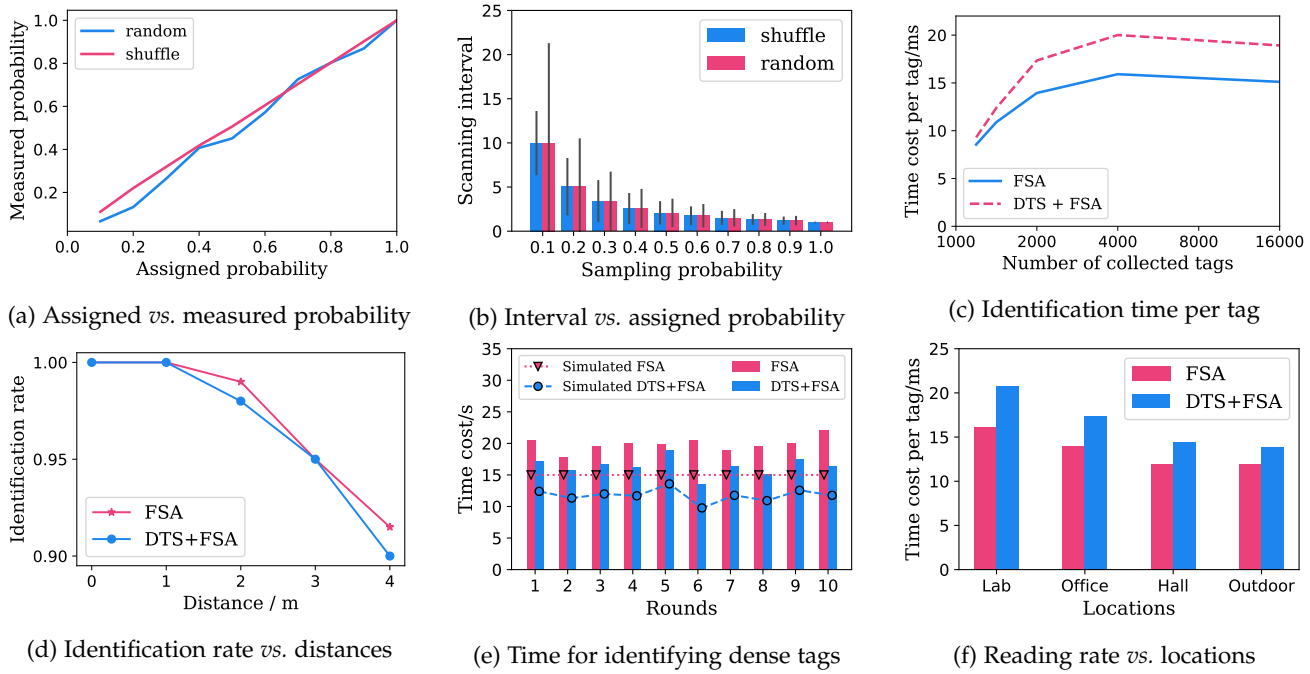
7.1.3 Sampling Operation

In this set of experiments, we will evaluate the sampling operation in DTS. Our program implements the shuffle algorithm detailed in Section 4.2.2 and uses the returned pointer value to sample tags with *TargetTag* class. Besides, we also implement the sampling method using random pointer for comparing with our method based on shuffle.

Sampling accuracy is the major metric of sampling operation. To verify whether the tags are sampled with the desired probabilities, we place 10 tags in the front direction of the antenna, the tag-antenna distance ranging from 0 to 4 meters. Each tag is assigned a distinct sampling probability ranges from 0.1 to 1.0. Specifically, the report mode of the reader is set to *individual*, hence, once a tag is identified, its ID will be immediately reported to the controller though TCP/IP. Then, we capture the IDs of the collected tags and store them in a file. We terminate the experiments after receiving 1000 tag IDs. We compare the assigned sampling probability and the actual sampling probability in Fig. 6a. We can find that both random-based and shuffle-based solution work as expected. The assigned and actual sampling probabilities of shuffle-based sampling method fit a little bit better compared with the random-based sampling method. This is reasonable because the pointers chosen by the random-based sampling are not completely uniform randomness. When designing the shuffle-based sampling operation, we have to sacrifice some randomness to improve the sampling accuracy. Fig. 6b shows the scanning intervals of shuffle sampling and random sampling with different sampling probabilities, respectively. Obviously, shuffle-based sampling has a smaller interval variance, which ensures that the tags can be sampled with a stable probability. This is important for a missing tag detection application, which normally desires to present from tag-starving issue, *i.e.*, a tag cannot be sampled for a long time.

Time efficiency is another important metric of the sampling operation. Although the sampling operation can benefit a wide range of applications, it will become useless if sampling operation takes too much time. To measure the time-efficiency of sampling operation, we deploy 10 tags in our lab, each of them is placed within 3 meters from the antenna. Each tag is assigned with a distinct sampling probability ranging from 0.1 to 1.0. We set the reader to *DualTarget* mode, which enables the reader to repetitively identify these 10 tags to obtain a large amount of observations as shown in Fig. 6c. Since the time cost of sampling

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2019.2917444, IEEE Transactions on Mobile Computing



(a) Assigned *vs.* measured probability

(b) Interval *vs.* assigned probability

(c) Identification time per tag

(d) Identification rate *vs.* distances

(e) Time for identifying dense tags

(f) Reading rate *vs.* locations

Fig. 6: Experiment results based on the small-scale prototype

operation is hard to measure, we use sampling-based identification as an indirect measurement of the cost of sampling operation. Specifically, we first conduct experiments to evaluate the time cost for collecting a certain number of tags with traditional FSA protocol, and then compare it with the time cost for collecting the same number of tags with DTS+FSA. Since the only difference between two protocols are the additional sampling operations, the gap between their execution time can be approximated as the time cost of sampling operations. Fig. 6c shows that the sampling operation incurs non-negligible overhead, which increases the total overhead of tag collection by about 30%. Besides, we can find that the average per tag identification time first increases and then keeps stable when the number of collected tag exceeds 2000. The IO speed of reading/writing file, and the connection speed may together contribute to this curve. In the experiments, we find that a large size output file lowers the total execution time of protocols due to the larger IO cost. The cost of FSA is 15 ms and that of DTS+FSA is about 20 ms. Thus, the sampling operation takes about 5 ms cost on each tag.

Then, we evaluate the protocol performance under different scenario. To evaluate the side impacts of the coupling effects, we place 4 pairs of stacked tags with varying distances from the reader antenna. The experimental results are shown in Fig. 6d, from which we can find that both FSA and DTS+FSA have similar curves. The rates of identifying tags decrease with the increase of distance from the reader antenna. This is because coupled nearby tags may absorb too much energy, which results in identification failure, especially when a tag is far from the antenna. To evaluate the effect of multi-path fading, we measure the reading speed of the proposed DTS scheme in different scenarios, *e.g.*, lab, empty office, department hall and outdoor. The results in Fig. 6f show that, the identification speed is higher in the environment with less obstacles, *e.g.*, the identification

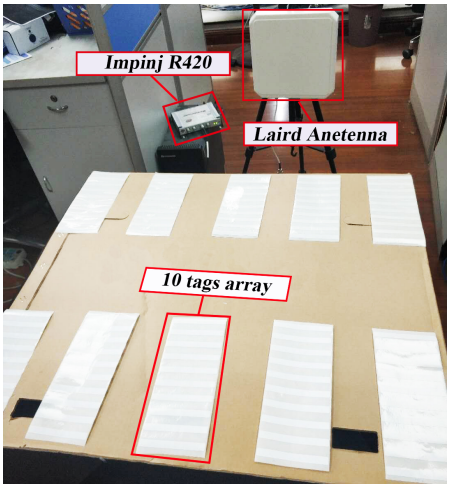
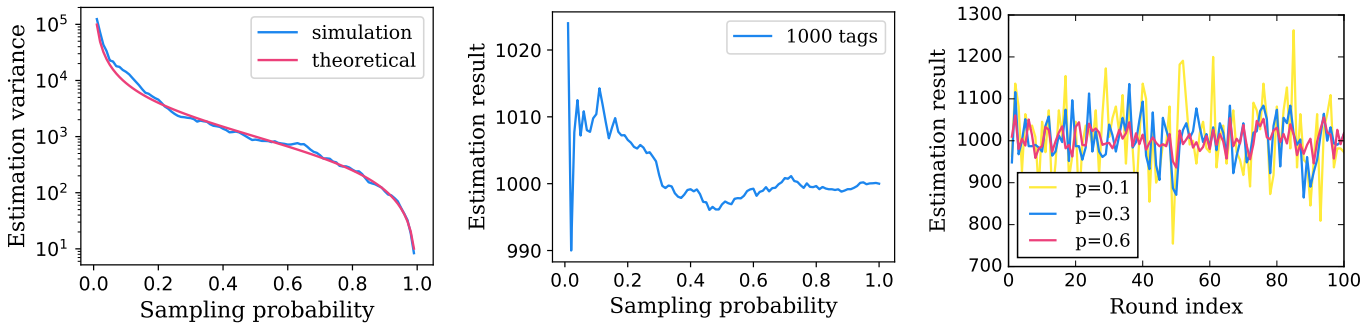


Fig. 7: Deployments of 100 tags

time of DTS in the lab scenario is 1.35x of that in the outdoor scenario because more obstacles in lab cause more serious multi-path affects. Multi-path fading makes some tags cannot harvest enough energy to be powered up, thus failing to be identified by the reader in a frame. Therefore, it will take more time frames to read such tags, resulting in lower time-efficiency.

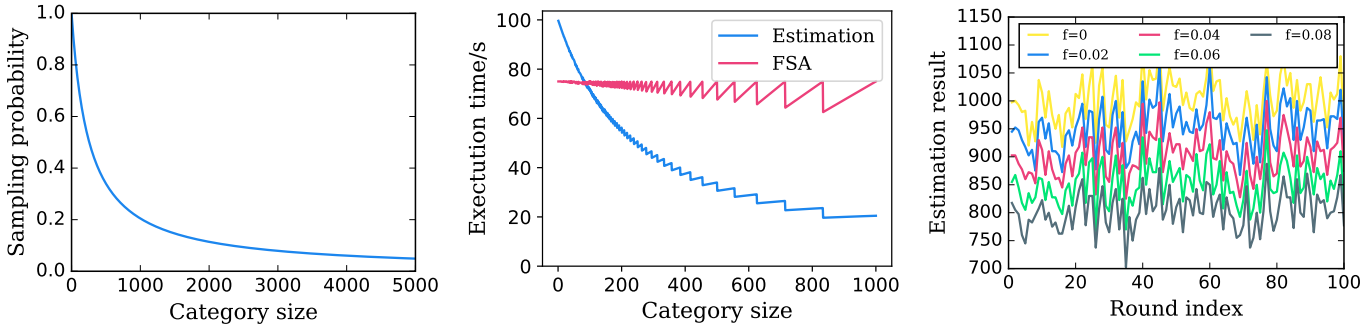
Finally, we evaluate the proposed scheme with 100 tags to verify its performance in the tag-dense scenario with the serious tag coupling effect. As shown in Fig. 7, 100 tags are deployed on a 1 m $\times$ 1 m table and are randomly assigned with sampling probabilities ranging from 0.1 to 1.0. We conduct 10 rounds evaluations, and run both FSA and DTS+FSA 10 times to get the total execution time during each evaluation round. We also use the per tag identification cost measured in the last experiment to compute the simulated execution time of FSA and DTS+FSA. The evaluation results in Fig. 6e shows that for both FSA and DTS+FSA, the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2019.2917444, IEEE Transactions on Mobile Computing



(a) Estimation variance (b) Mean estimation results (c) Single round estimation results

Fig. 8: Estimation accuracy with varying sampling probability



(a) Sampling probability *vs.* category size (b) Execution time *vs.* category size (c) Estimation results *vs.* failure rate

Fig. 9: Estimation performance with varying factors

actual execution time is always higher than the simulated execution time. This is because the coupling effect between tags reduces the rate of identifying tags. Thus, the reader has to take more time to identify a tag, which increases the total execution time.

In summary, both the assignment and sampling operations in DTS work as what we expect. DTS can significantly reduce the number of collected tags and arbitrarily allocate the wireless channel resource to tags in the system.

7.2 Evaluate DTS+MTCE and DTS+VMTD

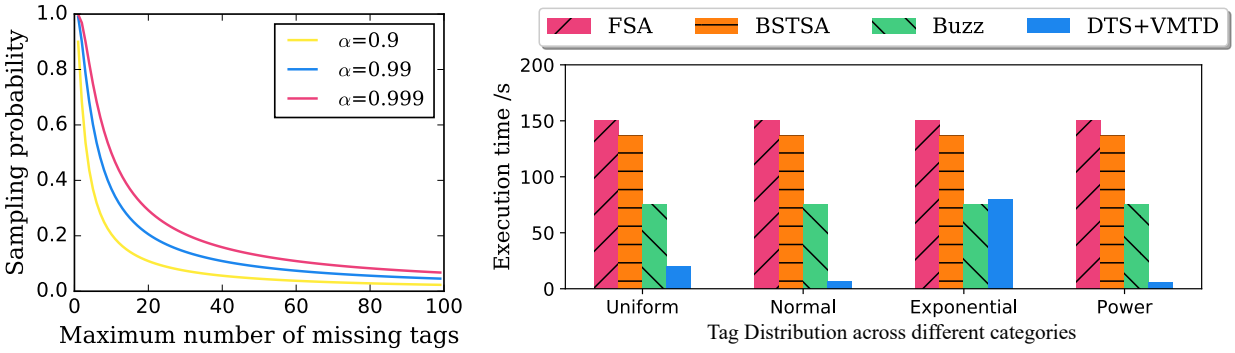
Besides implementing a small-scale prototype, we also simulate a large-scale warehouse scenario to evaluate the performance of our scheme. Following many related papers [36]–[38] in top journals or conferences, we assume there are thousands of tags within the interrogation range of a reader. This assumption is reasonable because we can significantly extend the reading range of a reader using the following methods. First, we can set the transmitting power of the reader to its maximum value to increase the reading distance. Second, we know that some kinds of COTS reader antennas can cover hundreds of square meters, *e.g.*, a single Impinj LHCP Far Field Antenna can cover a large area of 139 square meters [39]. Third, we can connect multiple antennas to the reader to significantly extend the reading range of the reader. For instance, each Impinj R420 Reader can be connected to 32 antennas at most [19]. By jointly using above three methods, the monitoring area of a reader can be extended to 4,448 square meters, where it is possible to place thousands of tags. We investigate the benefit of DTS in two typical applications: MTCE and VMTD. We use the system and communication parameters obtained from small-scale experiments (*e.g.*, time cost for collecting each tag) as the simulation settings.

7.2.1 *Multi-category Tag Cardinality Estimation*

Estimation Accuracy: Estimation accuracy is the key performance metric for the problem of Multi-category Tag Cardinality Estimation. Our scheme uses the number of collected tags in each category and the sampling probability to estimate the corresponding tag population. We use 1000 tags to evaluate the performance of the MTCE, Fig. 8 shows the estimation results with varying sampling probability. We independently repeat the estimation process for 100 rounds and present the averaged result in Fig. 8a. With the increase of sampling probability, the estimation error significantly decreases at first (*e.g.*, when $p < 0.1$) and then keeps relatively stable; whereas, the estimation variance keeps becoming small when the sampling probability increases. These results show that our sampling-based estimation scheme can provide highly accurate estimation results with a large enough sampling probability. Fig. 8a shows the observed estimation variance meets our theoretical analysis. Fig. 8c shows a large sampling probability leads to a better estimation accuracy. In fact, when $p = 1$, the estimation protocol goes back to the exact tag identification protocol with an estimation error of 0.

Time Efficiency: In a multi-category RFID system, the number of tags in each category may be different. To ensure the estimation accuracy, we need to apply DTS to assign the tags in a category with a certain sampling probability, which is determined by the number of tags in the category

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2019.2917444, IEEE Transactions on Mobile Computing



(a) Probability *vs.* missing threshold

(b) Execution time *vs.* category distributions

Fig. 10: Simulation results of value-based missing tag detection protocol

(see Eq. 8 for details). Fig. 9a presents the required sampling probability for a category with varying number of tags in this category when $\beta = 0.1$ and $\alpha = 0.1$. We can find that the required sampling probability becomes small with the increase of category size. Therefore, the time-efficiency of the Multi-category Tag Cardinality Estimation protocol is affected by the category size. Fig. 9b presents the time cost of estimating about 5000 tags in multiple categories. When a category size is smaller than 100, the estimation protocol even takes more time than collecting all tags via the FSA protocol. When conducting Multi-category Tag Cardinality Estimation protocol, we need to ensure that the weighted size of all the categories should be large than 100 at least; otherwise the estimation protocol will not bring any improvement in terms of time-efficiency than FSA.

Effect of Identification Failure: Identification failure is a common issue in real-world RFID systems, which means that a tag is present in the system but fails to be identified by the reader due to electromagnetic interference or absorption of RF energy. This issue may affect the estimation accuracy of the proposed sampling-based estimation protocol. Let p_f denote the identification failure rate. Fig. 9c shows the estimation results of 1000 tags with a sampling probability of 0.4. We can find that identification failure significantly reduces the estimation accuracy because it results in an underestimation of tag cardinality. For example, when the failure rate equals 0.8, the number of estimated tag approaches 800, which is only 80% of the actual value. We can deploy multiple antennas in the system to alleviate this adverse effect, because sampled tags that cannot be collected by an antenna may be collected by another antenna.

7.2.2 Value-based Missing Tag Detection

Sampling Probability: In the Value-based Missing Tag Detection (VMTD), each category of tags should be assigned with a certain sampling probability based on the item value in this category. To meet the required detection accuracy, Fig. 10a presents the required sampling probability with varying tolerable number of missing tags. The required sampling probability decreases significantly with the increase of the maximum tolerable missing tag number. For example, when the confidence level α equals to 0.999 and the tolerable missing tag number equals to 10, the sampling probability should be no less than 0.6. When the maximum number of missing tags that we can tolerate increases to 40, the sampling probability reduces to 0.2. Therefore, the low-value tag can be sampled with a much smaller probability for reducing the number of tags to be identified and accelerating the detection process.

Time Efficiency: Time-efficiency is the key metric of VMTD because a long execution time may disturb the execution of other RFID protocols, *e.g.*, tag localization and tag estimation. In this set of simulations, we simulate 10000 tags with different category distributions and run each protocol 300 times to report the averaged results. We assume there are 10 categories with distinct sampling probabilities ranging from $1/2$ to $1/1024$. We use four different numpy distribution models to generate the category distributions, including *uniform(1,5,10000)*, *norm(6,1.5,10000)*, *exponential(1.0, 10000)* and *power(5, 10000)*. As shown in Fig. 10b, the FSA, BSTSA and Buzz protocols always need to read a fixed number of 10000 tags under any distributions, thus always finishing the missing tag identification process with a fixed amount of time. Among them, Buzz has the best performance and is $2\times$ faster than FSA and BSTSA. This result is understandable because Buzz speeds up the tag identification by decoding corrupted information in collisions slots. However, implementation of Buzz requires the customized functionalities of mapping and decoding on programmable RFID devices, therefore, Buzz cannot be deployed to the commercial RFID systems. In contrast, the performance of VMTD changes a lot under different tag distributions. VMTD takes a longest time of 72.7 s under the exponential distribution, while it takes a shortest time of 6.12 s under the power distribution. This is because the number of sampled tags in VMTD is significantly affected by the tag distribution. When the tag population follows the power distribution across categories, most of the tags have small sampling probabilities. VMTD only needs to identify hundreds of tags, and thus being $10\times$ faster than the state-of-the-art Buzz protocol. On the other hand, when the tag population follows the exponential distribution across categories, most of tags need to be sampled in VMTD, thus its execution time approaches that of Buzz. Our VMTD uses the FSA scheme mainly because it is complaint to the current EPC C1G2 standard. Since DTS is a high-level solution that can work together with any available low-level MAC layer protocols, the performance of VMTD can be further improved if the Commercial Off-The-Shelf (COTS) RFID devices can support more efficient identification schemes in

MAC layer, *e.g.*, BTSTA and Buzz.

In summary, Value-based Missing Tag Detection (VMTD) can well address the missing tag detection problem. More channel resources are allocated to the tags that are attached to high-value items, by using DTS to assign large sampling probabilities to these tags. However, if most items in the system are of high values, we prefer to use FSA to collect all tags instead of invoking VMTD. Fortunately, pareto principle tells us that most items in a warehouse are of low values. In such an RFID system, the value-based missing tag detection protocol is quite time-efficient, because only a small number of tags in most categories need to be sampled and collected.

8 Conclusion

This paper studies the problem of Differential Tag Sampling (DTS). The core contribution of this paper is in using only commands available in C1G2 standard to implement the DTS protocol. The major advantages of DTS over previous work is two-fold: (1) DTS is able to assign each tag with a distinct sampling probability, however, previous work can only assign the same sampling probability to all tags in the system; (2) DTS is the first one that can be directly deployed on COTS RFID systems, however, previous work only stays in the theory level because they require unrealistic functionalities unavailable in C1G2 standard. To validate the effectiveness of our DTS, we apply DTS to address two practically important problems: Multi-category Tag Cardinality Estimation (MTCE), and Value-based Missing Tag Detection (VMTD). Extensive experimental results demonstrate that DTS is able to let each tag take the given sampling probability to be identified in the inventory process. Benefiting from our DTS protocol, the proposed MTCE and VMTD protocols can significantly reduce the execution time by nearly 70% than the FSA protocol.

Acknowledgment

This work was supported in part by the National Key Research and Development Program of China No. 2016YFB1000205, in part by the State Key Program of National Natural Science of China under Grants 61432002 and 61832013, in part by the NSFC under Grants 61772251, 61772112, 61672379, U1701263 and 61702365 and in part by the Dalian High-level Talent Innovation Program under Grant 2015R049.

References

[1] L. Yang, Q. Lin, X. Li, T. Liu, and Y. Liu, "See through walls with cots rfid system!" in *Proc. of ACM MOBICOM*, 2015.

[2] L. Yang, Y. Chen, X.-Y. Li, C. Xiao, M. Li, and Y. Liu, "Tagoram: Real-time tracking of mobile rfid tags to high precision using cots devices," in *Proc. of ACM MOBICOM*, 2014.

[3] L. Shangguan and K. Jamieson, "The design and implementation of a mobile rfid tag sorting robot," in *Proceedings of ACM MOBISYS*.

[4] J. Liu, B. Xiao, K. Bu, and L. Chen, "Efficient distributed query processing in large rfid-enabled supply chains," in *Proc. of IEEE INFOCOM*, 2014.

[5] L. Yang, Y. Li, Q. Lin, X.-Y. Li, and Y. Liu, "Making sense of mechanical vibration period with sub-millisecond accuracy using backscatter signals," in *Proc. of ACM MOBICOM*, 2016.

[6] Y. Ma, N. Selby, M. Singh, and F. Adib, "Fine-grained rfid localization via ultra-wideband emulation," in *Proc. of ACM MOBICOM*.

[7] E. Abad, F. Palacio, M. Nuin, A. G. De Zarate, A. Juarros, J. Gómez, and S. Marco, "Rfid smart tag for traceability and cold chain monitoring of foods: Demonstration in an intercontinental fresh fish logistic chain," *Elsevier Journal of food engineering*, vol. 93, no. 4, pp. 394–399, 2009.

[8] P. S. Taylor, I. Cullimore, and D. Wrench, "System and method of rfid data tracking," May 9 2006, uS Patent 7,040,532.

[9] E. W. Schuster, S. J. Allen, and D. L. Brock, *Global RFID: the value of the EPCglobal network for supply chain management*. Springer Science & Business Media, 2007.

[10] R. Weinstein, "Rfid: a technical overview and its application to the enterprise," *IT professional*, vol. 7, no. 3, pp. 27–33, 2005.

[11] C. Qian, Y. Liu, H. Ngan, and L. M. Ni, "ASAP: Scalable Identification and Counting for Contactless RFID Systems," in *Proc. of IEEE ICDCS*, 2010.

[12] C. C. Tan, B. Sheng, and Q. Li, "How to Monitor for Missing RFID tags," in *Proc. of IEEE ICDCS*, 2008.

[13] W. Luo, S. Chen, T. Li, and S. Chen, "Efficient missing tag detection in RFID systems," in *Proc. of IEEE INFOCOM*, 2011.

[14] C. C. Tan, B. Sheng, and Q. Li, "Efficient Techniques for Monitoring Missing RFID Tags," *IEEE Transactions on Wireless Communications*, vol. 9, no. 6, pp. 1882–1889, 2010.

[15] W. Luo, S. Chen, T. Li, and Y. Qiao, "Probabilistic missing-tag detection and energy-time tradeoff in large-scale RFID systems," in *Proc. of ACM MOBIHOC*, 2012.

[16] X. Liu, B. Xiao, K. Li, A. X. Liu, J. Wu, X. Xie, and H. Qi, "RFID Estimation with Blocker Tags," *IEEE/ACM Transactions on Networking*, vol. 25, no. 1, pp. 224–237, 2017.

[17] X. Liu, X. Xie, K. Li, B. Xiao, J. Wu, H. Qi, and D. Lu, "Fast Tracking the Population of Key Tags in Large-scale Anonymous RFID Systems," *IEEE/ACM Transactions on Networking*, vol. 25, no. 1, pp. 278–291, 2017.

[18] X. Liu, K. Li, A. X. Liu, S. Guo, A. L. Wang, X. Xie, and J. Wu, "Multi-category RFID Estimation," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 264–277, 2017.

[19] *R420 RFID reader*. [Online]. Available: https://support.impinj.com/

[20] *EPC Radio-Frequency Identity Protocols Class-1 Gen-2 UHF RFID Protocol for Communications at 860MHz-960MHz, EPCglobal*, Apr 2011. [Online]. Available: http://www.epcglobalinc.org/standards/uhfc1g2/

[21] B. Sheng, Q. Li, and W. Mao, "Efficient continuous scanning in RFID systems," in *Proc. of IEEE INFOCOM*, 2010.

[22] L. Xie, B. Sheng, C. C. Tan, H. Han, Q. Li, and D. Chen, "Efficient tag identification in mobile RFID systems," in *Proc. of IEEE INFOCOM*, 2010.

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

[24] M. Shahzad and A. X. Liu, "Probabilistic optimal tree hopping for rfid identification," *ACM SIGMETRICS Performance Evaluation Review*, vol. 41, no. 1, pp. 293–304, 2013.

[25] C. Law, K. Lee, and K.-Y. Siu, "Efficient memoryless protocol for tag identification," in *Proc. of ACM DIAL-M*, 2000.

[26] L. Pan and H. Wu, "Smart trend-traversal: a low delay and energy tag arbitration protocol for large RFID systems," in *Proc. of IEEE INFOCOM*, 2009.

[27] T. F. La Porta, G. Maselli, and C. Petrioli, "Anticollision protocols for single-reader rfid systems: Temporal analysis and optimization," *IEEE Transactions on Mobile Computing*, vol. 10, no. 2, pp. 267–279, 2011.

[28] J. Wang, H. Hassanieh, D. Katabi, and P. Indyk, "Efficient and reliable low-power backscatter networks," in *Proc. of ACM SIGCOMM*. ACM, 2012, pp. 61–72.

[29] D.-H. Shih, P.-L. Sun, D. C. Yen, and S.-M. Huang, "Taxonomy and survey of rfid anti-collision protocols," *Computer communications*, vol. 29, no. 11, 2006.

[30] J. Liu, B. Xiao, S. Chen, F. Zhu, and L. Chen, "Fast rfid grouping protocols," in *Proc. of IEEE INFOCOM*, 2015.

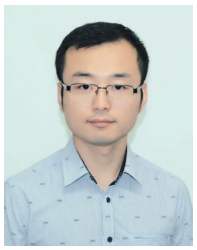
[31] W. Luo, S. Chen, T. Li, and S. Chen, "Efficient missing tag detection in rfid systems," in *Proc. of IEEE INFOCOM*, 2011.

[32] X. Liu, H. Qi, K. Li, I. Stojmenovic, A. X. Liu, Y. Shen, W. Qu, and W. Xue, "Sampling bloom filter-based detection of unknown rfid tags," *IEEE Transactions on Communications*, vol. 63, no. 4, pp. 1432–1442, 2015.

[33] *Impinj Monza 4 RFID Tag Chips*. [Online]. Available: https://support.impinj.com/hc/en-us/articles/

[34] *Octane SDK .NET*. [Online]. Available: https://support.impinj.com/hc/en-us/
[35] *S8658WPL 865-960 MHZ RFID PANEL ANTENNA*, Oct 2015. [Online]. Available: https://assets.lairdtech.com/home/brandworld/files/ANT-DS-S8658WPR%20S8658WPL-0915.pdf
[36] W. Gong, K. Liu, X. Miao, Q. Ma, Z. Yang, and Y. Liu, "Informative counting: fine-grained batch authentication for large-scale rfid systems," in *Proc. of ACM MOBIHOC*, 2013.
[37] H. Yue, C. Zhang, M. Pan, Y. Fang, and S. Chen, "Unknown-target information collection in sensor-enabled rfid systems," *IEEE/ACM Transactions on Networking*, vol. 22, no. 4, pp. 1164–1175, 2014.
[38] Y. Zheng and M. Li, "Zoe: Fast cardinality estimation for large-scale rfid systems," in *Proc. of IEEE INFOCOM*, 2013.
[39] *Impinj Xarray Solution*. [Online]. Available: https://support.impinj.com/hc/article_attachments/115001459570/Impinj_xPortalProductBrief_7.23.17_FINAL.pdf

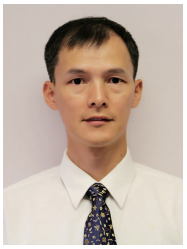
Xin Xie received the B.E. degree in computer science from Dalian University of Technology, Dalian, China, in 2013. He is currently pursuing the Ph.D. degree in Computer Science at Dalian University of Technology. He was a visiting scholar with the Department of Computer Sciences, Purdue University, USA, in 2018; His research interests include RFID and mobile sensing technologies.



Xiulong Liu received the B.E. degree from the School of Software Technology, Dalian University of Technology, China, in 2010; and the Ph.D. degree from the School of Computer Science and Technology, Dalian University of Technology, China, in 2016. He was a visiting scholar with the Department of Computer and Information Sciences, Temple University, USA, in 2015; and a Postdoctoral Fellow with the School of Computer Science and Engineering, the University of Aizu, Japan, 2016. Currently, he is a Postdoctoral Fellow with the Department of Computing, The Hong Kong Polytechnic University. His research interests include wireless sensing, ubiquitous computing, internet of things, etc.

Xibin Zhao received the BS, ME, and PhD degrees from the School of Computer Science and Telecommunication Engineering from Jiangsu University, in 1994, 2000, and 2004, respectively. He is now an associate professor in the School of Software, Tsinghua University. His research interests include reliability analysis of hybrid network systems and information system security.

Weilian Xue received the bachelor's and master's degrees both from Liaoning Normal University, China in 1989 and 2003, and the doctoral degree from Dalian University of Technology, China in 2011. She is a Professor at the School of Management, Liaoning Normal University. Her research interests are in wireless networks, cloud computing, data mining, and information systems.



Bin Xiao received the B.Sc. and M.Sc. degrees in electronics engineering from Fudan University, China, in 1997 and 2000, respectively,and the Ph.D. degree in computer science from the University of Texas at Dallas in 2003. After his Ph.D. graduation, he joined the Hong Kong Polytechnic University as an assistant professor. Currently, he is an associate professor in the Department of Computing at The Hong Kong Polytechnic University, Hong Kong. His research interests include mobile cloud computing, data management, network security, wireless sensor networks, and RFID systems. He is an associate editor for the International Journal of Parallel, Emergent and Distributed Systems.

Heng Qi is an associate professor at the School of Computer Science and Technology, Dalian University of Technology, China. He received bachelor's degree from Hunan University in 2004 and master's degree from Dalian University of Technology in 2006. Then he received his Ph.D. degree from Dalian University of Technology in 2012. His research interests include computer network, wireless network and multimedia computing.

Keqiu Li received the bachelor's and master's degrees from the Department of Applied Mathematics at the Dalian University of Technology in 1994 and 1997, respectively. He received the Ph.D. degree from the Graduate School of Information Science, Japan Advanced Institute of Science and Technology in 2005. He also has two-year postdoctoral experience in the University of Tokyo, Japan. He is currently a professor in the School of Computer Science and Technology, Dalian University of Technology, China. He has published more than 100 technical papers, such as IEEE TPDS, ACM TOIT, and ACM TOMCCAP. He is an Associate Editor of IEEE TPDS and IEEE TC. He is a senior member of IEEE. His research interests include internet technology, data center networks, cloud computing and wireless networks.

Jie Wu is the associate vice provost for international affairs with Temple University. He also serves as the chair and Laura H. Carnell professor in the Department of Computer and Information Sciences. Prior to joining Tempe University, he was a program director at the US National Science Foundation and was a distinguished professor with Florida Atlantic University. His current research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. He regularly publishes in scholarly journals, conference proceedings, and books. He serves on several editorial boards, including the IEEE Transactions on Service Computing and the Journal of Parallel and Distributed Computing. He was general co-chair/chair of the IEEE Mobile Adhoc and Sensor Systems 2006, the IEEE International Parallel & Distributed Processing Symposium 2008, IEEE ICDCS 2013, and ACM MobiHoc 2014, as well as program co-chair for IEEE INFOCOM 2011 and CCF CNCC 2013. He was an IEEE Computer Society Distinguished Visitor, ACM Distinguished Speaker, and chair of the IEEE Technical Committee on Distributed Processing (TCDP).