Computation–Communication Tradeoffs for Missing Multitagged Item Detection in RFID Networks

Hao Liu, Rongrong Zhang, Lin Chen, *Member, IEEE*, Jihong Yu, *Member, IEEE*, Jiangchuan Liu, *Fellow, IEEE*, Jianping An, *Member, IEEE*, and Qianbin Chen, *Senior Member, IEEE*

***Abstract*—Missing item event detection is one of the most important radio-frequency identification (RFID)-enabled functions. Yet it is largely unaddressed how to fast and reliably detect missing item event in multitagged RFID systems where multiple tags are tagged on one item. The canonical methods can only solve tag-level detection problem where each item is associated with one tag, and applying them to detect the missing multitagged items would falsely alarm and is time inefficient. To bridge the gap, this article formulates and analyzes the missing multitagged item detection problem. Our key idea is to search the proper seeds so that the reader only needs to probe a subset of the tags each being selected from different items instead of the entire tag set for the missing item detection. By employing the computation–communication tradeoffs, we design two protocols named M^{2}ID and M^{2}ID+ that classifies the tags before the segmentation compared to the former to improve time efficiency. With the derived optimum parameters, our protocols can achieve up to 4x performance gain in terms of time efficiency compared with the state-of-the-art solution.**

***Index Terms*—Computation–communication tradeoffs, missing item event detection, multitagged item, radio-frequency identification (RFID).**

I. Introduction

RADIO-FREQUENCY identification (RFID) is an enabling technology in a variety of Internet of Things (IoT) applications, ranging from inventory control [1], [2], supply chain management [3], [4], to object tracking [5], [6], and localization [7]. An RFID system typically consists of one or multiple *readers* and a massive number of *tags*. Readers, usually backed up with sufficient computation and storage capacity, interrogate the tags via wireless channel. On the other hand, tags characterized by unique IDs [8], [9], are able to capture energy from the RF signal of a nearby reader and backscatter their messages.

Item tracking and monitoring is a typical and important IoT application where RFID is deployed. It is reported that inventory shrinkage, shoplifting, internal theft, and human error, bring nearly 61.7 billion dollars in loss for U.S. retailers in 2020 [10] and almost 13.4 billion dollars for U.K. retailers annually [11]. Therefore, time-efficient and reliable missing item detection is of critical importance.

This article concentrates on a particular missing item detection scenario arising from multitagged RFID systems, where each item to be monitored is attached with multiple tags. Attaching multiple tags on an item has such advantages as enhancing security [12], [13], improving localization accuracy, and accurately monitoring an item's state [14]–[16]. However, this also brings new challenges for missing item detection, which are largely unexplored in the literature. Prior works [2], [17]–[25], cannot be applied in the missing multitagged item detection since here these tag-level algorithms would be time inefficient, repeating checking the present item. Specifically, the response of one tag attached to the item is enough to show the presence of the item. Therefore, the missing item detection problem in the multitagged RFID systems is more challenge and the systematic study is called for to bridge the gap between tag-level detection and item-level detection.

This article studies the missing multitagged item detection problem in RFID systems and devises time-efficient solutions. Motivated by the observation above, we query a subset of the tags instead of the entire tags in prior works. We develop two protocols following the guideline.

1) *Seed Searching:* Based on the characteristic of hash function, a proper seed, which makes one tag from each item map to a unique value, achieves extracting a subset of the tags in the system and assigning them to singleton slots in the response frame for the missing item detection.
2) *Reader-Tag Communication:* The reader broadcasts the seed chosen in the first step and the corresponding unique hashing values. The tags mapping to these values respond in sequence accordingly. The reader then detects missing items based on the tags' responses.

The first protocol named M^{2}ID provides a suit of the segmentation and the seed searching methods. M^{2}ID divides

Manuscript received January 4, 2021; revised March 27, 2021; accepted May 4, 2021. Date of publication May 11, 2021; date of current version January 7, 2022. This work was supported in part by the NSF of China under Grant 61901035 and Grant 61801064, and in part by the Beijing Institute of Technology Research Fund Program for Young Scholars and Young Elite Scientist Sponsorship Program by CAST and Chongqing Key Laboratory of Mobile Communications Technology. The work of Rongrong Zhang was supported by the Science and Technology Project of Beijing Municipal Education Commission under Grant KM202010028005. *(Corresponding author: Jihong Yu.)*

Hao Liu, Jihong Yu, and Jianping An are with the School of Information and Electronics, Beijing Institute of Technology, Beijing 100081, China (e-mail: jihong.yu@bit.edu.cn).

Rongrong Zhang is with the Information Engineering College, Capital Normal University, Beijing 100048, China (e-mail: zhangrr@cnu.edu.cn).

Lin Chen is with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510275, China (e-mail: chenlin69@mail.sysu.edu.cn).

Jiangchuan Liu is with the School of Computing Science, Simon Fraser University, Burnaby, BC V5A1S6, Canada (e-mail: jcliu@sfu.ca).

Qianbin Chen is with the School of Communications and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China (e-mail: chenqb@cqupt.edu.cn).

Digital Object Identifier 10.1109/JIOT.2021.3079269

all tags into multiple tag segments to reduce the computation time cost of the seed searching. It makes each segment contain the same number of the target tags with the unique hashing values fallen into a specific interval. This would reduce the communication time cost by avoiding broadcasting unique hashing values to the tags in sequence, and avoiding receiving time-inefficient response since these unique values implying the positions of the target tags' response slots are randomly distributed. On the top of M^2ID, we propose M^2ID+ to further improve the time efficiency. M^2ID+ works in a "sampling-classification-segmentation" pattern, which can reduce the reader-tag communication cost at the price of extra less computation cost and thus the overall time cost. The main contributions of this article are articulated as follows.

1) We formulate the largely unaddressed missing multitagged item detection problem in RFID systems and provide solutions from the perspective of the computation–communication tradeoffs.
2) We present two concrete protocols namely M^2ID and M^2ID+. M^2ID constructs a framework integrating the seed searching and the reader-tag communication, enabling the computation–communication tradeoffs. M^2ID+ exploiting the time difference between the computation and the communication reduces the communication cost at the expenses of affordable extra computational cost, improving the time-efficiency.
3) We optimize the protocol performance with the optimum parameters derived. The analytical results also reveal the relationship between the frame size and the probability of finding a proper seed, indicating the computation–communication tradeoffs.
4) We conduct extensive simulations to evaluate the performance of the proposed protocols. The results show that M^2ID and M^2ID+ achieve performance gain of 2x and 4x over the state-of-the-art one [26] in terms of time efficiency, respectively.

We would like to emphasize that we provide not only the efficient solutions to the missing multitagged item detection problem, but a new methodology embracing the computation–communication tradeoffs, which benefits to solve other protocol design problems in RFID systems.

II. Related Work

The works of missing item detection are separated into two categories: 1) probabilistic protocols [2], [17]–[22] and 2) deterministic protocols [23]–[25].

Probabilistic protocols detect a missing item event with a predefined probability. Tan *et al.* [17] initiated the study of probabilistic detection and propose a solution called trusted reader protocol (TRP). TRP detects missing item event by comparing the precomputed slots with those picked by the tags attached on items. If an expected singleton slot turns out to be empty, then the missing item event is detected. Luo *et al.* [18] and [19] employed multiple seeds to increase the probability of the singleton slot, which reduces the useless empty and collision slots and thus achieves better performance. RUN [20] and BMTD [21] address the influence of the unknown tags. Yu *et al.* [2] designed a suit of detection protocols for multicategories and multiregion RFID systems and study how to detect missing item by using COTS RFID devices.

Deterministic protocols, on the other hand, are able to exactly identify which items are absent. Li *et al.* develop a series of deterministic protocols in [23] to reduce the radio collision by reconciling collision slots and finally iron out a bit-level tag identification method by iteratively deactivating the tags of which the presence have been verified, hence affirming the presence of items. Subsequently, Zhang *et al.* proposed [24] to identify tag responses in all rounds and observe the change among the corresponding bits among all bitmaps to determine the present and the absent tags for identifying the presence of items. But how to configure the protocol parameters is not theoretically analyzed. More recently, Liu *et al.* [25] enhanced the work by reconciling both 2-collision and 3-collision slots and filtering the empty and unreconcilable collision slots to improve time efficiency.

With the presence of the multitagged items in RFID systems, the prior works show their weakness in terms of time efficiency. The key of addressing the multitagged missing item detection problem is to probe a subset of the tags for the detection, avoiding repeated checks of the present items. However, very limited works have studied the problem from this perspective. The most related work [26] utilizes a bloom filter to solve the tag identification problem for the multitagged RFID systems. Yet, the false positives of the bloom filter and the low ratio of the singleton slots (no more than 36.7%) make it time inefficient for the missing item detection.

III. System Model and Problem Formulation

In this section, we will introduce the system model used in our paper and formulate the problem of detecting the missing multitagged items in an RFID system.

A. System Model

We consider an RFID system consisting of one reader [1] and a large number of the tags, where each physical item is attached by multiple tags. The reader is connected with a backend server which has a powerful computational capability. For the purpose of simplification, we treat the reader and the server as an entity and just call it reader. Moreover, each tag has a unique ID and performs computations, such as hashing function. All tags' IDs in the system are recorded by the reader.

The communication between the reader and the tags follows the rule of "Listen-before-talk" [8], [28]. In the detection, for an example, the reader broadcasts commands and parameters including the frame size f and a seed s at first. Then, each tag uses its ID and the received seed s to generate one pseudo random value via hash function as $(H(\mathrm{ID}, s) \mod f)$, and executes the next step according to the received commands (i.e., compare, response or wait for next commands).

[1]For multiple readers, we can treat them as a single virtual reader as in [22] and [27]. Specifically, the backend server searches all proper seeds and corresponding hashing values and sends them to all readers such that readers broadcast these parameters. Consequently, the backend server can synchronize the readers and we can logically consider them as a whole.

The downlink (i.e., reader-to-tags) transmission is continuous. The uplink (i.e., tags-to-reader), on the other hand, contains a blank slot between any two tags' 1-bit responses [8]. For simplicity, we denote T_d and T_{tag} as the time duration of 1-bit broadcasting slot and 1-bit response slot, respectively. Consider an arbitrate response slot, it may experience three states. When no tags respond in this slot, it is an empty slot; when a single tag responds, it is a singleton slot; when multiple tags respond, it is a collision slot. The latter two states are also regarded as nonempty slots. Considering the unstable channel, there exists error transmissions. We assume that the downlink works in error-free channel since the reader supported by the external power source can increase the transmission power. On the contrast, the tag can not support much power to counter the interference. Therefore, we assume that the error occurs in the uplink and the manifestation of the error transmission is bit inversion. The "1" inverted to "0" brings the false positives and the "0" inverted to "1" induces the false negatives which will cause the practical damage.

B. Problem Formulation

In this article, we are interested in detecting the missing multitagged items in an RFID system where n tags monitor g items each attached by the multiple tags, i.e., $g < n$. Considering the instability of the uplink, we define m_a as the number of missing items and P_d as the probability that the reader successes in detecting the missing item event without the false alarm. We formulate the multitagged missing item detection problem as follows: The missing multitagged item detection problem is to devise an algorithm of minimum execution time to detect missing item event with $P_d \geq \alpha$ under the condition of $m_a \geq M_a$ in the unstable uplink. The α is the required correct detection probability among all detection, and the M_a is a predefined detection threshold meaning the tolerance to the minimum number of the missing items. Note that the problem is degraded to deterministically identify missing items when $\alpha = 1$ and $M_a = 1$. The proposed protocols in this article can also achieve deterministic missing item identification.

We would like to emphasize the key difference between the missing multitagged item detection problem and the prior missing single-tagged item detection problem: The successful response of one tag on a multitagged item indicates the presence of the item, it is thus feasible to query one tag for checking the state of an item. In contrast, if we use the prior algorithms for the missing single-tagged item detection problem, all tags on the item would respond to the interrogation, resulting in severe interference and thus considerably degrading time efficiency. For example, there are 10 000 items being monitored where each item is attached by three tags, the prior works have to probe 30 000 tags, which sharply increases the time cost. Instead, we only query 10 000 tags by picking one tag from each item.

C. Design Rational

The response of a tag means the presence of the item, the reader has no need to query the other tags on this item. Meanwhile, the absence of one tag indicates the potential missing item. Therefore, it is feasible to probe one of the tags on an item instead of all for the missing item event detection. If the probed tag is absent, we would further poll the left tags on the item, and a missing item would be found if all of them are absent. Considering the number of missing items is usually small, the idea above can improve time efficiency significantly.

In this article, we randomly select one tag from each item, referred to as a representative tag. These g tags constitute the representative tag set defined as $\mathcal{G}_{\mathcal{A}} = \{RT_1, RT_2, \ldots, RT_g\}$ where the RT_k is a tag on the item k for $1 \leq k \leq g$. The set of the remaining tags named pending tags is denoted by $\mathcal{G}_{\mathcal{B}}$. We are interested in interrogating the representative tags to detect potential missing item event. Unfortunately, the pending tags in $\mathcal{G}_{\mathcal{B}}$ would cause the sever interference to the representative tag detection. Hence, an efficient scheme should be able to eliminate this negative impact.

We distinguish the representative tags and the pending tags via their hashing values by selecting such a seed that there exists no common hash value mapped by the tags in $\mathcal{G}_{\mathcal{A}}$ and $\mathcal{G}_{\mathcal{B}}$. Furthermore, we prefer a proper seed that makes each tag in $\mathcal{G}_{\mathcal{A}}$ map to a unique hash value and respond in a singleton slot. Consequently, the reader only needs to broadcast the proper seed and corresponding unique hashing values. Each tag learns whether it should respond after comparing its hashing value with the received hashing value, and the response slot of the representative tag is determined by the hashing value. After receiving the representative tags' responses, the reader can find the potential missing items.

Yet, it is impractical to search a proper seed for all tags in a large-scale system for the extremely high computational complexity, it is necessary to design a strategy that can lessen the number of the tags simultaneously involving in the seed searching. On the other hand, these unique hashing values indicating the response slot positions of the representative tags are randomly distributed. Hence, we have to broadcast each of these hashing values and the response slots might involve empty slots, leading to the low efficiency. To address the obstacles, we induct these unique hashing values to a special range so that we only need to broadcast the boundary values of this range once and all response slots are singleton, which retrenches the reader-tags communications. This makes the communication cost reduced from broadcasting hashing values many times determining by the number of the representative tags to broadcasting only twice, meanwhile, from involving empty slots to reaching 100% utilization in response frame.

Moreover, motivated by the difference between the computation time of the reader and the transmission rate among the reader-tag communications (i.e., the clock period of the CPU in the reader is about 0.3 ns and a time slot in the communication is over 10 μs), we could tradeoff the computation cost and the communication cost to further minimize the overall execution time. Following the design rational above, we construct M^{2}ID and the improved M^{2}ID+.

IV. M^{2}ID: Missing Multitagged Item Detection Protocol

A. Motivation

The seed and a tag's ID determine the hashing value of the tag in RFID systems, so we can find out a proper seed that makes each of g representative tags have a unique hashing value. The reader then broadcasts the seed and the corresponding hashing values informing the tags whether/when they should respond. We denote the jth representative tag's hashing value under the proper seed s and the frame size f by $h_j = (H(ID_j, s) \mod f) + 1$, meaning its response slot is also h_j. To avoid sequentially broadcasting these unique hashing values, we make them all designated in a range of $[1, g] \subseteq [1, f]$ that we only need to broadcast the parameter g once at the beginning of the protocol. This makes the communication cost reduced from broadcasting every unique hashing values to broadcasting only one value.

We take an example to illustrate this. As shown in Fig. 1, there exists two items and each is attached with three tags (i.e., $g = 2$ for two representative tags). We pick one tag from each item as a representative tag (i.e., tag 1,4). Originally, we select a proper seed s' so that all representative tags' hashing values are unique and their corresponding hashing values represent the slot position in the response frame. For example, the hashing value of the tag 1 is 2 and it should respond in the second slot in the response frame, the tag 4's hashing value is 5 and it should respond in the fifth slot in the response frame [see Fig. 1(a)]. Hence, we have to broadcast these hashing values (i.e., 2, 5) to tags and the response frame involves empty slots, leading to low efficiency. In our design, as shown in Fig. 1(b), our required seed s makes all unique hashing values fall in the range of [1, 2], while the pending tags' hashing values are out of this range. Finally, tag 1, 4 will respond in the response slots 1, 2, respectively. Thus, we only need to broadcast hashing value 2 and the utilization of response frame reaches 100%, which retrenches communication time cost.

Finding such a seed is effective for the missing item detection, but the time cost would soar if we search for all tags in the system. Specifically, the probability of seeking out the proper seed can be expressed as follows:

$$p_{ft} = \frac{g!}{g^g}\left(\frac{g}{f}\right)^g\left(1 - \frac{g}{f}\right)^{n-g}. \tag{1}$$

It can be proven that p_{ft} is maximum when $f = n$ (see Section IV-D). We next show that the time cost is unaffordable even for a small-scale system. For example, when $n = 100$ and $g = 50$, we have $p_{ft} = 2.701 \times 10^{-51}$. That is to say, we need $N_s \times T_{\text{cpu}} \approx 3.702 \times 10^{38}$s $\approx 1.174 \times 10^{31}$years on average with 1000-GHz CPU to find the seed for 100 tags. Therefore, a scheme of lower computation complexity is called for to achieve the time-efficient detection.

In this article, we follow the principle of "Divide and Conquer." Take the system of $n = 100$ and $g = 50$ as an example again. We first divide them into ten segments and each segment consists of $n_i = 10$ tags including $g_i = 5$ representative tags. Recall (1), it only takes 5.333 μs to finding the proper seed in a segment when the reader works on a 5-GHz CPU. And the total searching time for all segments is 53 μs, which is significantly less than the unsegmented one above. Aggressively, if we divide all tags into 50 segments where $n_i = 2$, $g_i = 1$ on average, the total searching time decreases to 40 ns. Therefore, the segmentation is an effective method to decrease the time cost in the seed searching.

B. Segmentation

Segmentation can reduce the computational complexity, but directly segmenting the tags randomly is undesirable. Recall that the tags conduct hashing operations with a seed and the frame size, and are segmented by their hashing values and a given segment size. The random segmentation would unbalance the number of the tags falling into the segments, degrading the performance gain of the segmentation. On the other hand, we have to spend extra time on telling tags the range of unique hashing values in each segment due to the different number of the representative tags in each segment.

In this article, we propose a segmentation method approximately uniformly segmenting the set of the tags. By the uniform segmentation, each segment contains the identical number of the representative tags and the similar number of the pending tags. Hence, the hashing value range of each segment might be different. More specifically, this method operates as follows. First, the reader records all tags' IDs, we thus simulate at the reader that all tags calculate their hashing values under a seed s_{sg} and frame size f_{sg}. The setting of frame size f_{sg} will be discussed in Section IV-D.

Second, the reader determines the hashing value range of each segment guaranteeing the identical number of the representative tags in each segment. Each segment's hashing value range can be expressed via boundary values. The lower bound and upper bound values can be set as follows. We denote g_i as the number of the representative tags in the ith segment for $i = 1, 2, \ldots$ For the first segment, the reader sets lower bound as $b_{t1} = 1$ and then seeks out the largest value of upper bound b_{e1}. The required b_{e1} should satisfy $g_1 = \sum_{k=b_{t1}}^{b_{e1}} \Phi(k) \le g_d$, where g_1 is the number of the representative tags whose hashing values are in the range of $[b_{t1}, b_{e1}]$, $\Phi(k)$ is the number of the representative tags whose hashing values equal to k, and g_d is the required number of the representative tags in each segment. Once b_{e1} set, the boundary values of the first segment are set as $[b_{t1}, b_{e1}]$. Thus, the tags should be in the first segment when their hashing values are into the range of $[b_{t1}, b_{e1}]$. For the second segment, we set the lower bound as $b_{t2} = b_{e1} + 1$ and find the largest value of b_{e2} as the first segment does. After repeating above processes, we eventually set each segment's boundary values so that each segment contains no more than g_d representative tags.

C. Protocol Description

Our proposed multitagged missing item detection protocol M^{2}ID can be described as follows.

We start at the view of the reader.

1) In the segmentation, the reader picks up an arbitrary seed s_{sg} from the seed pool and encodes each tag via its hashing value which is in a range of $[1, f_{sg}]$, meanwhile,

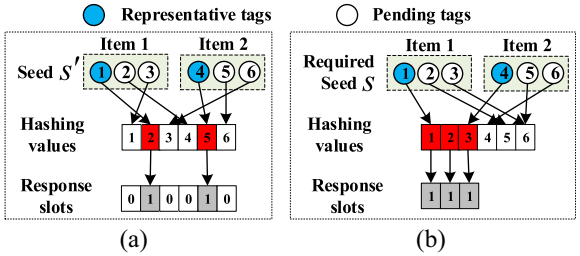

Fig. 1. Example of required seed s when $n = 6$, $g = 2$, $f = 6$.

confirms the boundary values of each segment. Then, the reader compares each tag's hashing value with the segment boundary to decide each tag's affiliation.

2) In the seed searching, the reader searches a proper seed under the corresponding optimum frame size to separate the representative tags and the pending tags in each segment. The position of a tag's response is determined by the serial number of the segment and its hashing value. In the ith segment, the representative tags' hashing values are unique and in a range of $[1, g_d]$ with the proper seed s_i and the optimum frame size f_i. In case that we can not always seek out the proper seed for all representative tags mapping to $[1, g_d]$ since the limited scale of the seed pool, we prefer the suboptimum seed that makes the representative tags map to $[1, g_d]$ as many as possible.
3) In the reader-tag communications, the reader first broadcasts parameters including the seed s_{sg} and the frame size f_{sg} for the segmentation. Then, it broadcasts the boundary values $\{b_{ti}, b_{ei}\}$ of the ith segment and its corresponding seed s_i and the frame size f_i. After finishing broadcasting, the reader interrogates the tags and listens to the tags' responses for detecting the missing item event.

At the tags end, each tag receives parameters from the reader and first calculates its hashing value for the segmentation with the seed s_{sg} and the frame size f_{sg}. Then, it compares its hashing values with the boundary values of segments. If its hashing value falls into $\{b_{ti}, b_{ei}\}$, it should be in the ith segment. After that, it uses the received s_i and f_i to do another hashing. If the value falls into $[1, g_d]$, the tag will regard itself as a representative tag and then respond in the corresponding position of the response frame when interrogated. Otherwise, the tag will keep silent and wait for the reader's new command. Consequently, only the representative tags respond and all the response slots should be singleton.

Since the response slots are orchestrated to be mapped by one representative tag, the reader knowing all representative tags' mapping positions can detect the missing item event if there exists at least one empty slot. After finding any empty slot, the reader will poll the other tags on the item attached by this missing representative tag to affirm the item state.

By conducting the operations above across all segments, we can identify all representative tags. One of the challenges in M^{2}ID is how to tune parameters for the minimum execution time. We will address this in the following.

D. Parameter Optimization

In this part, our optimization is described based on the situation that all proper seeds are sought out in the seed pool. Our goal is to configure the frame size f_{sg} for the segmentation, the frame size f_i for the seed searching in the ith segment, and the required number of the representative tags g_d in each segment.

1) At the beginning, we first discuss the setting of the frame size f_{sg} in the segmentation. Based on the description in Section IV-B, the number of the representative tags in each segment is no more than g_d. Thus, the positions of the representative tags' responses are decided as follows:

$$\text{RePos}_j = g_d(i-1) + y_j \tag{2}$$

where the y_j is the hashing value of the jth representative tag in the ith segment. In a segment, g_i representative tags' hashing values are unique and in the range of $[1, g_d]$. Sometimes, g_i may be smaller than g_d because of the coincident hashing values of the multiple representative tags. If we make $g_i = g_d$ in each segment, we would achieve 100% utilization of response frame without empty slots. For example, each segment has two representative tags and we consider the ith segment. The proper seed in this segment makes the representative tags' hashing values be $y_1 = 1$ and $y_2 = 2$. Thus, the tags will respond in the $(2(i-1)+1)$th slot and the $(2(i-1)+2)$th slot, respectively. The key of achieving $g_i = g_d$ is that each representative tag maps to a unique value without considering the pending tags' hashing values. The probability that each representative tag maps to a unique hashing value can be written as

$$p_g = \frac{\prod_{j=0}^{g-1}(f_{sg}-j)}{f_{sg}^{g}} \geq 66.7\%. \tag{3}$$

The expected round of the seed searching defined as N_g is $1/p_g$. Here, $p_g \geq 66.7\%$ means that an arbitrary seed makes the representative tags map to the unique values, i.e., $\mathbb{E}[N_g] = 1/p_g \leq 1.499 \approx 1$. The value of f_{sg} can thus be derived while the number of the representative tags in each segment is g_d.

2) We next discuss the optimum frame size f_i in the seed searching. The probability of seeking out a proper seed for the ith segment can be expressed as follows:

$$p_{s_i} = g_d!\left(\frac{1}{f_i}\right)^{g_d}\left(1-\frac{g_d}{f_i}\right)^{n_i-g_d} \tag{4}$$

where n_i is the number of the tags in the ith segment and f_i is the frame size for calculating hashing value. We then should derive f_i to maximize p_{s_i}.

Theorem 1: Given g_d and n_i in the ith segment, the optimum size of f_i maximizing p_{s_i} should satisfy $f_i = n_i$.

Proof: The partial differential function of p_{s_i} by f_i can be expressed as

$$\frac{\partial p_{s_i}}{\partial f_i} = \frac{g_d g_d!}{f_i^{g_d+1}}\left(1-\frac{g_d}{f_i}\right)^{n_i-g_d-1}\left(\frac{n_i}{f_i}-1\right). \tag{5}$$

When $(\partial p_{s_i}/\partial f_i) = 0$, we have two points, such as $f_{i_1} = g_d$ and $f_{i_2} = n_i$, and $g_d \leq f_i$. According to the requirement of the proper seed, we set $f_i = n_i$. As shown in (5), it is positive when $f_i < n_i$ and negative when $f_i > n_i$. Hence, we can get the maximum p_{s_i} with $f_i = n_i$. ■

Theorem 1 indicates the optimum frame size f_i is determined by the number of the tags n_i in this segment.

3) Now, we discuss the selection of g_d. The expected execution time of the segmentation, defined as T_{sg}, consists of the time spent on the seed searching and broadcasting the parameters including the seed value, the frame size for the segmentation and each segment's boundary values. The cost of broadcasting g_d can be negligible compared with the cost of other parameters broadcasting. Thus, the cost of the segmentation and broadcasting each boundary values for all segments can be expressed as

$$T_{sg} = T_{\text{searching}} + T_{\text{each}} = nN_gN_cT_{\text{cpu}} + \big(L(f_{sg}) + L(s_{sg})\big)T_d + \sum_{i=1}^{r}(L(b_{ti}) + L(b_{ei}))T_d \tag{6}$$

where $N_g = 1$ holds following (3). We use $L(\cdot)$ to stand for $\log_2(\max\{\cdot\})$ and the number of the segments r is equal to g/g_d. The operator of $L(\cdot)$ shows the length of data expressed by the binary sequence. As the boundary values of each segment should be less than f_{sg}, the length of binary sequence expressing boundary values is twice of $L(f_{sg})$. Therefore, (6) can be rewritten as

$$T_{sg} = nN_cT_{\text{cpu}} + \left(L(s_{sg}) + \left(\frac{2g}{g_d} + 1\right)L(f_{sg})\right)T_d. \tag{7}$$

We can observe that the execution time of the segmentation is decided by g_d when f_{sg} is derived by (3).

Then, we will discuss the execution time after each tag recognizes its belonged segment. As described in Section IV-C, the expected total execution time of the seed searching, broadcasting the corresponding optimum frame sizes and the seed values can be written as

$$T_s = \sum_{i=1}^{r} N_{s_i}n_iN_cT_{\text{cpu}} + (L(f_i) + L(s_i))T_d \tag{8}$$

where the N_{s_i} is the round of the seed searching and we have $N_{s_i} = 1/p_{s_i}$. As stated in Theorem 1, we prefer p_{ms} to represent the maximum of p_{s_i} with the respect to f_i if we make $f_i = n_i$

$$p_{ms}(g_d, n_i) = \frac{g_d!}{g_d^{g_d}}\left(\frac{g_d}{n_i}\right)^{g_d}\left(1 - \frac{g_d}{n_i}\right)^{n_i - g_d}. \tag{9}$$

Hence, the minimum expected round for searching the proper seed is $1/p_{ms}(g_d, n_i)$.

During the searching process, we initialize the seed value to "1" and increase by "1" in the next round if we do not find out the proper one. Therefore, the value of s_i is equal to $1/p_{ms}(g_d, n_i)$, and T_{ss} can be rewritten as

$$T_s = \sum_{i=1}^{r}\frac{n_iN_cT_{\text{cpu}}}{p_{ms}(g_d, n_i)} + \left(L(n_i) + L\left(\frac{1}{p_{ms}(g_d, n_i)}\right)\right)T_d. \tag{10}$$

In addiction, the execution time of the response frame can be expressed as

$$T_r = \sum_{i=1}^{r} g_iT_{\text{tag}} = rg_dT_{\text{tag}} = gT_{\text{tag}}. \tag{11}$$

Recall (6), (10), and (11), the expected execution time of the M^{2}ID, defined as the T_{whole}, is

$$T_{\text{whole}} = T_{sg} + T_s + T_r = nN_cT_{\text{cpu}} + gT_{\text{tag}} + \left(L(s_{sg}) + \left(\frac{2g}{g_d} + 1\right)L(f_{sg})\right)T_d + \sum_{i=1}^{\frac{g}{g_d}}\frac{n_iN_cT_{\text{cpu}}}{p_{ms}(g_d, n_i)} + \left(L(n_i) + L\left(\frac{1}{p_{ms}(g_d, n_i)}\right)\right)T_d. \tag{12}$$

We observe that T_{whole} is determined by the latter part of T_{sg} (i.e., $2g/g_dL(f_{sg})T_d$) and the whole part of T_s. Considering f_i and f_{sg} are fixed so that T_d and T_s are determined by g_d, we thus only optimize these two part with g_d where we approximate n_i with the expected number of the tags $\mathbb{E}[n_i] = ng_d/g$. However, using the expected value of n_i may be inaccurate for $L(n_i)$ that depends on the maximum n_i. To solve this problem, we set an upper bound for the maximum n_i. Based on extensive numerical analysis, we fix $\max\{n_i\} = 3n/g$. Intuitively, we extract the part related with g_d from (12), the expression is written as follows:

$$T_{\text{sim}} = \frac{g}{g_d}\left(\frac{ng_d}{g}\frac{N_cT_{\text{cpu}}}{p_{ms}(g_d, \mathbb{E}[n_i])}\right) + \frac{2g}{g_d}\log_2(f_{sg})T_d + \frac{g}{g_d}\left(\log_2\left(\frac{3ng_d}{g}\right) + \log_2\left(\frac{1}{p_{ms}(g_d, \max\{n_i\})}\right)\right)T_d. \tag{13}$$

Now, the problem is converted to find out the proper value of g_d to minimize T_{sim}. Generally, we are going to directly conduct differential function of T_{sim} to calculate the extreme point where the proper g_d can be found. Yet, the differential function is too complex to derive the closed form of g_d. A feasible way is to find such an upper bound that the values of g_d over this bound would make T_{sim} increasing.

Conducting algebraic operations, we can observe that the first part of T_{sim} is of the order of the magnitude $\Theta((ne/g)^{g_d})$ while the sum of the other parts is in $\Theta(1/g_d)$. Consequently, we can find the upper bound for g_d due to the fact that T_d is significantly larger than T_{cpu}. Considering the exponential increase and the reciprocal decrease of T_{sim} with g_d, the upper bound is usually not large. Once finding it, we would search an optimum g_d from 1 to the upper bound.

We conduct the numerical experiment to understand this with varying n and g where the period of CPU is 0.27 ns and doing hash function needs 344 clock cycles. As shown in Fig. 2, T_{sim} increases extremely when g_d is greater than 4, and the minimum of T_{sim} can be reached by $g_d \leq 4$.

After the three steps above, we can obtain f_{sg}, f_i, and g_d that would minimize the overall execution time of M^{2}ID. Let us take an example to interpret M^{2}ID. As shown in Fig. 3, there exists $g = 3$ items each being attached by 3 tags. Using the parameter configuration method presented in this section, we have $f_{sg} = 9$ and $g_d = 1$. The tags are divided into three

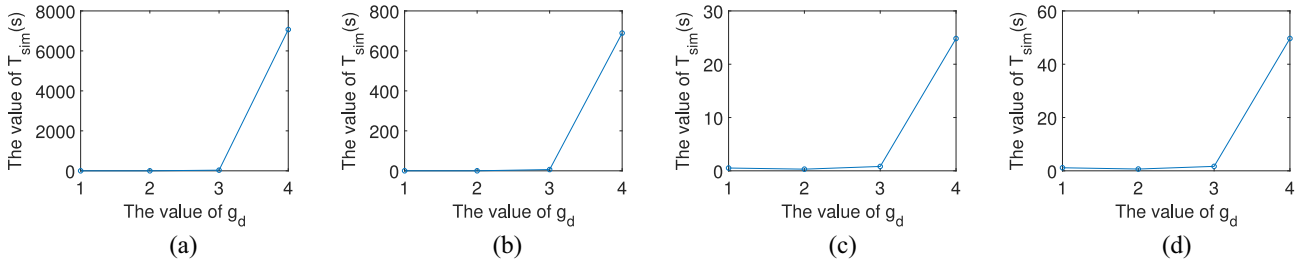

Fig. 2. Impact of g_d on T. (a) Value of T_{sim} when $n = 1000$ and $g = 100$. (b) Value of T_{sim} when $n = 1000$ and $g = 200$. (c) Value of T_{sim} when $n = 1000$ and $g = 500$. (d) Value of T_{sim} when $n = 2000$ and $g = 1000$.

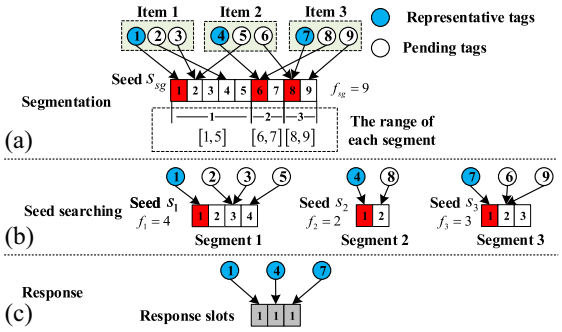

Fig. 3. Illustration of M^{2}ID when $n = 9$ and $g = 3$.

segments with s_{sg} [see Fig. 3(a)] and then the reader finds the proper seed s_i of the ith segment [see Fig. 3(b)]. Finally, each tag calculates its position with the received parameters and decides whether/when to respond following (2). If recognizing itself as a representative tag, the tag replies in the calculated position in the response frame. Consequently, only the representative tags respond in sequence [see Fig. 3(c)].

As mentioned in Section III-A, we assume the downlink works in the error-free channel and the uplink works in the unstable channel. Hence, we assume the probability of each received bit from the tags occurring bit inversion is p_e. The probability of a missing representative tag which is undetected in M^{2}ID resulted from the bit inversion is written as

$$P_{\text{unde}} = 1 - P_m(1 - p_e) = p_e \tag{14}$$

where P_m is the probability that a representative maps to a singleton slot and the value is 1. Therefore, the probability of detecting the real missing representative tag is expressed as

$$P_{cd} = 1 - P_{\text{unde}}^{M_a} = 1 - p_e^{M_a} \tag{15}$$

where the M_a is the detection threshold. Therefore, the reliability of M^{2}ID working in the unstable channel is estimated.

V. M^{2}ID+: The Improvement of M^{2}ID

In this section, we introduce M^{2}ID+ to more actively tradeoff the computation time and the communication time to further improve the time efficiency.

A. Motivation

M^{2}ID can effectively complete the missing multitagged item event detection, its performance, however, is hindered by the time cost of the reader-tag communication, as described below. First, probing all representative tags is time-consuming because the detection time cost is proportionate to their size while it is adequate for the probabilistic detection to interrogate part of the representative tags with the given reliability requirement. Second, (3) indicates that the frame size in the segmentation f_{sg} increases as the system scales up, the time spent on broadcasting boundary values of each segment will thus soar following (7), degrading the time efficiency. Moreover, we have to broadcast the different frame sizes used for the seed searching of each segment in M^{2}ID, which introduces extra time cost.

To tackle the drawbacks of M^{2}ID, we introduce the following three approaches to further embrace the computation–communication tradeoffs, improving the time efficiency.

1) We select the partial tags via sampling instead of the entire tag set in M^{2}ID, which reduces the number of the representative tags participating in the detection. Note that if both the sampling ratio and the required reliability are 1, M^{2}ID+ can achieve the deterministic detection.
2) We avoid spending too much time on broadcasting the boundary values of each segment and improve the time efficiency via two steps.
   a) We divide the sampled tags into categories and each is further segmented into multiple segments with the smaller frame size for the subsequent segmentation.
   b) We introduce the seed searching into the segmentation instead of selecting an arbitrary seed in M^{2}ID, which would further reduce the frame size used in the segmentation.
3) During the seed searching of each segment, we set the identical frame size across all segments of a category so that the readers broadcasts its value once instead of multiple times in M^{2}ID.

We would like to explain that M^{2}ID+ brings extra computation time cost compared to M^{2}ID, but this reduces more communication time cost and thus the overall time cost. Let us take an example to make the explanation. Consider a system input of $g_d = 1, g = 50, n = 500$, M^{2}ID would use 1212-bit binary sequence to broadcast the boundary values of all segments. In contrast, this cost reduces to 411 in M^{2}ID+ where the tags are classified into ten categories before the segmentation, including 300 bits representing the boundary values, 100 bits expressing the ten 10-bit seed sequences used in the subsequent segmentation for all categories, and about 11-bit cost equivalently to the computation time of 279 μs.

B. Protocol Description

In M^{2}ID+, the reader works in two modes, i.e., offline or online. In the offline mode, the reader will conduct the parameters configuration, the operations of the sampling and the classification according to the user's requirements. They are done in the reader once the system is confirmed before the detection. On other side, online mode will search the seed and communicate with the tags for each executed detection. Therefore, the time cost for the detection discussed in this article refers to the cost of online mode.

1) *Sampling Classification:* In the offline mode before the detection, the reader first picks up two arbitrary seeds and two corresponding thresholds to conduct the sampling and the classification. For example, an arbitrary tag's hashing value is $h_{s,j} = (H(ID_j, s_{\text{sample}}) \bmod f_{\text{sample}}) + 1$ with the sampling seed s_{sample} and the frame size f_{sample}. The threshold Th is $\lceil p_{\text{sample}} f_{\text{sample}} \rceil$. If $h_{s,j} \leq Th$, this tag is sampled. Consequently, np_{sample} of the tags are sampled to participate in the sequent operations. The reader then makes these sampled tags do another hashing operation with the second seed s_{class} and its corresponding classification size f_{class}, i.e., $h_{c,j} = (H(ID_j, s_{\text{class}}) \bmod f_{\text{class}}) + 1$. The value of $h_{c,j}$ indicates the category the tag belongs to. For example, If $h_{c,j} = u$, it would be classified into the category u for $u = 1, 2, \ldots, f_{\text{class}}$.
2) *Segmentation of a Category:* During the online mode for the detection, consider an arbitrary category u, the reader first divides it into multiple segments. Specifically, the reader searches for such a seed s_{seg_u} that all representative tags of this category map to unique hashing values with the frame size f_{seg_u}. Furthermore, we only record the length of each segment (i.e., $d_{q_u} = b_{eq_u} - b_{tq_u} + 1$, where q_u represents the qth segment in the category u) instead of the boundary values in M^{2}ID. The reader then finds another proper seeds for the representative tags in each segment ensuring that their hashing values fall into $[1, g_d]$. The process of the seed searching in each segment is similar with M^{2}ID. The difference here lies in that we use an identical frame size f_{s_u} across all segments instead of the different frame sizes in M^{2}ID.
3) *Parameters Broadcasting:* The reader first broadcasts the seeds, the frame sizes and the thresholds used in sampling and classification. For the category u, the reader broadcasts the parameters used in the segmentation, namely the frame size f_{seg_u} and the seed s_{seg_u}. The reader then sends the identical frame size f_{s_u} used in the seed searching for each segment and each segment's length d_{q_u} and the found seeds. Sequentially, the reader interrogates the sampled representative tags in the category u and waits for their responses. Repeating these operations for all categories, the reader checks the observed response slots of the representative tags. It can detect missing items if the predicated busy slots turn to be empty.

At the tag side, each tag does hash function to check whether it is sampled according to the received parameters. Only a sampled tag determines which the category it belongs to, while the unsampled tags will keep silent. After knowing its category, the tag computes its segment and checks whether it is a representative tag. Each representative tag will then respond at a corresponding slot as M^{2}ID does.

M^{2}ID+ makes the sampled representative tags map to singleton slots and then identified when all proper seeds are sought out. The key left is to configure the parameters used in the sampling, the classification and the segmentation to minimize the overall execution time.

C. Parameter Setting

We here introduce how to set the parameters used in M^{2}ID+ so that the detection reliability can be satisfied while the overall execution time can be minimized. Consider we have U categories (i.e., $U = f_{\text{class}}$) and the expected number of the tags and the representative tags in each category is n_s/U and g_s/U, respectively, under the sampling ratio p_{sample}, where $n_s = np_{\text{sample}}$, $g_s = gp_{\text{sample}}$. In an arbitrary category u, the reader divides the tags into several segments with the frame size f_{seg_u} and each segment contains g_d representative tags. Recall Section V-B, the expected overall time cost of M^{2}ID+ is

$$T_{\text{total}} = T_{r\_\text{whole}} + T_{\text{res}} = UT_u + g_s T_{\text{tag}} \tag{16}$$

where T_{r_whole} is the expected time cost used by the reader to complete the computation and the broadcasting for U categories. T_u is the expected time cost of the category u, and T_{res} is the time duration of the response frame determined by the number of the sampled representative tags g_s.

For the uth category, the expected time cost can be divided into the following three parts:

$$T_u = T_{\text{seg}_u} + T_{\text{iden}_u} + T_{ssb_u} \tag{17}$$

where T_{seg_u} is the expected time cost used for the segmentation and its parameters transmission, T_{ssb_u} is the expected time cost of the seed searching and the seed transmission for all segments, and T_{iden_u} is the cost of broadcasting the identical frame size f_{s_u} for all segments expressed as $T_{\text{iden}_u} = L(f_{s_u})T_d$.

As mentioned above, the time cost of the segmentation contains the cost of the seed searching and the parameters broadcasting. Thus, T_{seg_u} is written as

$$T_{\text{seg}_u} = \frac{C_u n_s N_c}{U} T_{\text{cpu}} + \left(\frac{g_s}{g_d U} L(D) + L\big(f_{\text{seg}_u}\big) + L(C_u) \right) T_d \tag{18}$$

where D is the expected length of each segment in all categories, and C_u is the expected round needed to find a seed for the segmentation. In addition, the value of the seed equals to C_u. Specifically, the expected length of each segment is

$$D = \mathbb{E}\big[d_{q_u}\big] = \frac{f_{\text{seg}_u} g_d U}{g_s}. \tag{19}$$

And the expected number of the rounds is

$$C_u = \frac{1}{\frac{\prod_{q=0}^{\frac{g_s}{U}-1} f_{\text{seg}_u} - q}{f_{\text{seg}_u}^{\frac{g_s}{U}}}} = \frac{f_{\text{seg}_u}^{\frac{g_s}{U}}}{\prod_{q=0}^{\frac{g_s}{U}-1} f_{\text{seg}_u} - q} \tag{20}$$

where f_{seg_u} should meet $f_{\text{seg}_u} \geq \frac{g_s}{U}$ in order to make all representative tags map to the unique hashing values.

Moreover, T_{ssb_u} is the sum of the cost of the seed searching for all segments T_{ss_u} and the cost of broadcasting the proper seeds T_{ssb_u}. Specifically, T_{ss_u} can be written as

$$T_{ss_u} = \sum_{q=1}^{r_u} \frac{n_{q_u} N_c T_{\text{cpu}}}{p_{sms}(g_d, n_{q_u}, f_{s_u})} = \frac{n_s N_c T_{\text{cpu}}}{U p_{sms}\left(g_d, \frac{g_d n}{g}, \frac{g_d n}{g}\right)} \tag{21}$$

where r_u is the number of the segment in the category u and its expected value is $[g_s/(g_d U)]$. n_{q_u} is the number of the tags in the qth segment of the category u and its expected value is $[(g_d n)/g]$. The probability $p_{sms}(g_d, [(g_d n)/g], [(g_d n)/g])$ of finding a proper seed with the identical frame size $[(g_d n)/g]$ can be written as

$$p_{sms}\left(g_d, \frac{g_d n}{g}, \frac{g_d n}{g}\right) = \frac{g_d!}{g_d^{g_d}} \left(\frac{g}{g_d n}\right)^{g_d} \left(1 - \frac{g}{g_d n}\right)^{\frac{g_d n}{g} - g_d}. \tag{22}$$

Correspondingly, the expected time cost T_{ssb_u} of broadcasting the proper seeds of all segments in the category u is

$$T_{ssb_u} = \frac{g_s}{g_d U} L\left(\frac{1}{p_{sms}\left(g_d, \frac{g_d n}{g}, \frac{g_d n}{g}\right)}\right) T_d. \tag{23}$$

Therefore, the (17) can be rewritten by substituting with (18), (21), and (23)

$$\begin{aligned} T_u = {} & \frac{n_s N_c}{U} \left(C_u + \frac{1}{p_{sms}\left(g_d, \frac{g_d n}{g}, \frac{g_d n}{g}\right)}\right) T_{\text{cpu}} \\ & + \left(\frac{g_s}{g_d U} L(D) + L(f_{\text{seg}_u}) + L(C_u) + L\left(\frac{g_d n}{g}\right)\right) T_d \\ & + \frac{g_s}{g_d U} L\left(\frac{1}{p_{sms}\left(g_d, \frac{g_d n}{g}, \frac{g_d n}{g}\right)}\right) T_d. \end{aligned} \tag{24}$$

Recall the operation of $L(\cdot)$, we make the following settings for the analysis feasibility: $\max\{D\} = 3 f_{\text{seg}_u} g_d U / g_s$, $\max\{n_{q_u}\} = 3 g_d n / g$, $\max\{f_{\text{seg}_u}\} = f_{\text{seg}_u}$, $\max\{C_u\} = C_u$. Thus, the expression (24) can be expanded as

$$\begin{aligned} T_u^* = {} & \frac{n_s N_c}{U} \left(C_u + \frac{1}{p_{sms}\left(g_d, \frac{g_d n}{g}, \frac{g_d n}{g}\right)}\right) T_{\text{cpu}} \\ & + \left(\log_2(f_{\text{seg}_u}) + \log_2(C_u) + \log_2\left(\frac{g_d n}{g}\right)\right) T_d \\ & + \frac{g_s}{g_d U} \log_2\left(\frac{3 f_{\text{seg}_u} g_d U}{g_s p_{sms}\left(g_d, \frac{3 g_d n}{g}, \frac{g_d n}{g}\right)}\right) T_d. \end{aligned} \tag{25}$$

And we thus have the expected overall execution time of all categories at the side of the reader as follows:

$$\begin{aligned} T_{r\_\text{whole}} = {} & \sum_{u=1}^{U} T_u^* = U T_u^* \\ = {} & \left(C_u + \frac{1}{p_{sms}\left(g_d, \frac{g_d n}{g}, \frac{g_d n}{g}\right)}\right) n_s N_c T_{\text{cpu}} \\ & + \left(\log_2(f_{\text{seg}_u}) + \log_2(C_u) + \log_2\left(\frac{g_d n}{g}\right)\right) U T_d \\ & + \frac{g_s}{g_d} \log_2\left(\frac{3 f_{\text{seg}_u} g_d U}{g_s p_{sms}\left(g_d, \frac{3 g_d n}{g}, \frac{g_d n}{g}\right)}\right) T_d. \end{aligned} \tag{26}$$

Therefore, the expected overall execution time of our proposed M^{2}ID+ is expressed as

$$\begin{aligned} T_{\text{total}} = {} & \left(C_u + \frac{1}{p_{sms}\left(g_d, \frac{g_d n}{g}, \frac{g_d n}{g}\right)}\right) n_s N_c T_{\text{cpu}} \\ & + \left(\log_2(f_{\text{seg}_u}) + \log_2(C_u) + \log_2\left(\frac{g_d n}{g}\right)\right) U T_d \\ & + \frac{g_s}{g_d} \log_2\left(\frac{3 f_{\text{seg}_u} g_d U}{g_s p_{sms}\left(g_d, \frac{3 g_d n}{g}, \frac{g_d n}{g}\right)}\right) T_d + g_s T_{\text{tag}}. \end{aligned} \tag{27}$$

Obviously, T_{total} is determined by p_{sample}, U, f_{seg_u}, and g_d. The key of achieving the best time-efficiency is to minimize T_{total} with the optimum values of these four parameters.

To this end, we first show the configuration of the sample ratio p_{sample} under the requirement of the detection. Second, we discuss the optimum number of the categories U, the optimum frame size for the segmentation f_{seg_u} in the category u, and the optimum number of the representative tags in each segment g_d. Third, we derive the identical frame size for the specific situation after the classification and the segmentation.

1) *Optimum Sampling Ratio:* A smaller sampling ratio usually yields the less time cost, but the unbounded decreasing would make the detection unreliable. Consequently, we must set an appropriate sampling ratio. The correct detection probability in the unstable channel is expressed as

$$P_d = 1 - \left(1 - p_{\text{sample}} P_m (1 - p_e)\right)^{M_a} \tag{28}$$

where P_m is the probability of an arbitrary representative tag mapping to a singleton slot in the response frame, p_e is the probability of the bit inversion induced by the unstable channel, and M_a is the threshold. Recall Section V-B, we ensure the probability of an arbitrary representative tag mapping to a singleton slot in the response frame is 1. Consequently, given the reliability requirement α and establishing $P_d \geq \alpha$, we have

$$p_{\text{sample}} \geq \frac{1}{1 - p_e}\left(1 - (1 - \alpha)^{\frac{1}{M_a}}\right) \tag{29}$$

which is the sampling ratio we need.

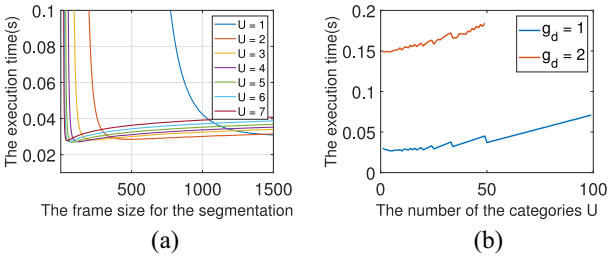

Fig. 4. Parameters configuration when $n = 1000$ and $g = 100$. (a) Search the optimum f_{seg_u}. (b) Search the optimum U.

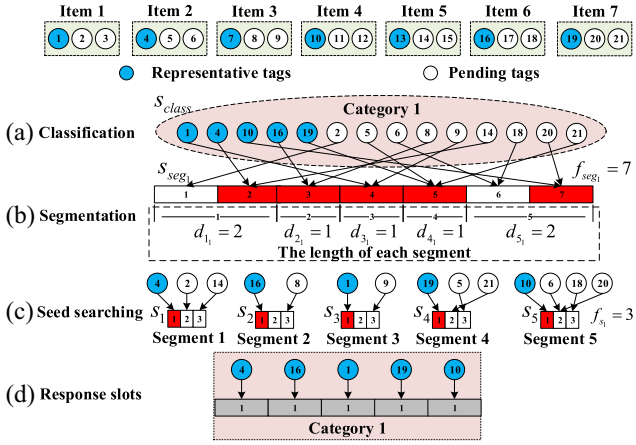

Fig. 5. Illustrating M^{2}ID+ with seven items each attached by three tags.

2) *Configuring the Number of the Categories U, the Frame Size Used for the Segmentation f_{seg_u}, and the Number of the Representative Tags in Each Segment g_d:* Recall (27), the object moves to configure the number of the categories U, the number of the representative tags g_d in each segment, and the frame size of the segmentation f_{seg_u} in each category to minimize the execution time T_{total}. Because it is difficult to directly derive them, we have to search the optimum value of these parameters. Let us start with the g_d. We have shown in Section IV-D that g_d would be small with the high probability because of the exponentially increasing rate. We set the range of g_d as $1 \leq g_d \leq 4$. And U should be set to ensure that each category contains the representative tag(s), we thus have $1 \leq U \leq (g_s/g_d)$. The lower bound of f_{seg_u} is set as (g_s/U) according to the (20). The upper bound f_{seg_u} can be set if f_{seg_u} makes $C_u(f_{\text{seg}_u}, U) < 2$, which means that the expected round of searching a proper seed for the segmentation is less 2. Now, the range of these three parameters have been determined, we start to seek out the proper values. We will take an example to explain the searching process with a system consisting of 100 items and ten tags on each item (i.e., total 1000 tags in the system), as shown in Fig. 4. The period of the CPU is 0.27 ns with the clock round of 344 doing hash function and the downlink transmission rate is 40.97 kb/s. The curves in Fig. 4(a) shows the time cost with the different number of categories and the different frame size for the segmentation. Note that it is adequate to show $1 \leq U \leq 7$ because the execution time soars for $U > 6$. More specifically, the curves at the different g_d in Fig. 4(b) shows the time cost with the different number of categories under the corresponding optimum f_{seg_u}. Hence, we can obtain the optimum parameters as follows: the number of the representative tags in each segment $g_d^* = 1$, the frame size $f_{\text{seg}_u}^* = 35$ used for the segmentation of a category u, and the number of the categories $U^* = 5$. As mentioned in Section V-B, the optimum parameter configuration for the sample, the classification and the segmentation can be calculated offline and can be recorded in the reader's storage. The reader will select the proper configuration from the storage once the system input is fixed.

3) *Identical Frame Size of the Seed Searching for Each Segment:* Before the classification and the segmentation, the expected number of the tags segmented into the qth segment of the category u is $\mathbb{E}[n_{q_u}] = ng_d/g$ under the fixed g_d. After that, the true number of the tags n_{q_u} might be different, which results in directly using the $n_{q_u} = g_d n/g$ is inefficient.

For the category u after the classification and the segmentation, the number of segments is r_u under the fixed g_d. The expected cost of the seed searching for all segments is

$$T_{ss_u}^* = \sum_{q=1}^{r_u} \frac{n_{q_u} N_c T_{\text{cpu}}}{p_{sms}\left(g_d, n_{q_u}, f_{s_u}\right)}. \tag{30}$$

The difference f_{s_u} between (17) and (30) is that the former is an expected value used to set U, g_d and f_{seg_u} while the latter optimum based on the true value of the n_{q_u}. Now, our objective is to minimize the (30) with a proper $f_{s_u}^*$. We have

$$\frac{\partial T_{ss_u}^*}{\partial f_{s_u}} = \frac{N_c T_{\text{cpu}} g_d^{g_d}}{g_d!} \sum_{q_u=1}^{r_u} \frac{g_d^2 \left(1 - \frac{g_d}{f_{s_u}}\right)^{g_d - n_{q_u}}}{f_{s_u}^2 \left(\frac{g_d}{f_{s_u}}\right)^{g_d+1}} + \frac{g_d\left(g_d - n_{q_u}\right)\left(1 - \frac{g_d}{f_{s_u}}\right)^{g_d - n_{q_u} - 1}}{f_{s_u}^2 \left(\frac{g_d}{f_{s_u}}\right)^{g_d}}. \tag{31}$$

The result is o when $f_{s_u}^* = (1/r_u) \sum_{q_u=1}^{r_u} n_{q_u}$. Thus, the proper frame size $f_{s_u}^*$ of the category u is fixed.

We next take an example to explain M^{2}ID+. Our system is shown in Fig. 5. The configuration is $g_d = 1$, $U = 2$ and all tags are sampled. Then, the tags in the category 1 are divided into five segments with the $f_{\text{seg}_1} = 7$ and the reader searches the proper seed for each segment with the identical frame size $f_{s_1} = 3$. Finally, only the representative tags respond for the missing item detection, and all the slots in the response frame are singleton, that said the utilization ratio of the slots to identify the representative tags reaches 100%. The tags in the category 2 would repeat the above process.

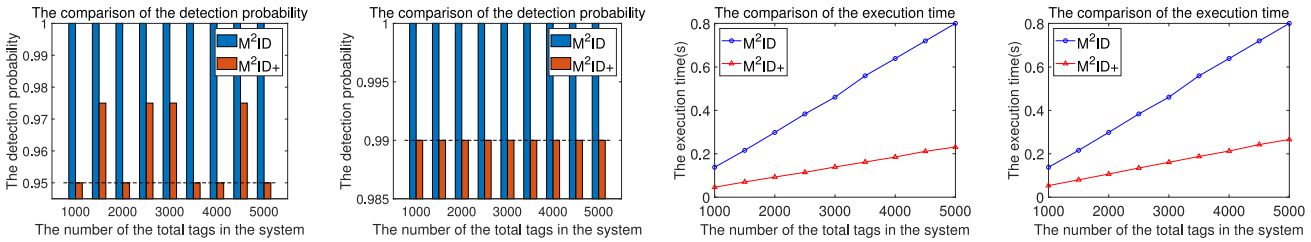

Fig. 6. Detection probability and the execution time versus the number of the total tags with $\alpha = 95\%$ and $\alpha = 99\%$. (a) $\alpha = 95\%$. (b) $\alpha = 99\%$. (c) $\alpha = 95\%$. (d) $\alpha = 99\%$.

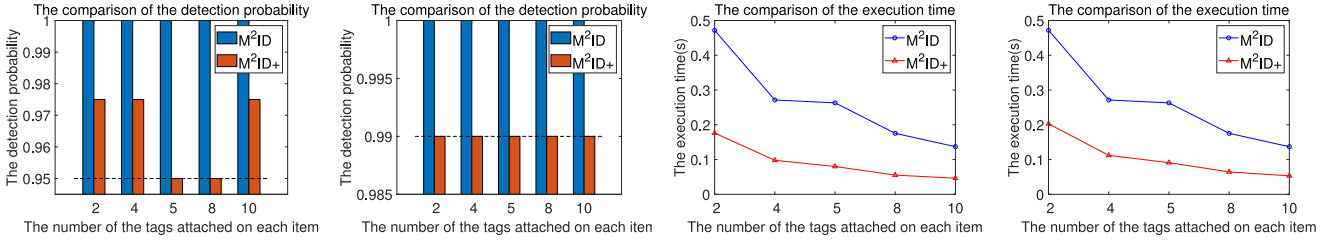

Fig. 7. Detection probability and the execution time versus. The number of the tags on each item with $\alpha = 95\%$ and $\alpha = 99\%$. (a) $\alpha = 95\%$. (b) $\alpha = 99\%$. (c) $\alpha = 95\%$. (d) $\alpha = 99\%$.

VI. Performance Evaluation

We evaluate the performance of the proposed M^{2}ID and M^{2}ID+ in terms of the detection probability and the execution time, and compare their performance with the most related work named BPI [26] that uses bloom filter to identify the tags in the multitagged RFID systems. The timing parameters in the simulation follow the EPC-global Gen-2 standard [8]. Specifically, the transmission rate is 40.97 kb/s, and a broadcast slot and a response slot are $T_d = 24.4$ μs and $T_{\text{tag}} = 290.8$ μs, respectively. Furthermore, the number of the rounds accomplishing once hash function is $N_c = 344$ [29] and the period of the CPU's clock is $T_{\text{CPU}} = 0.27$ ns. The parameters like the sampling ratio and the number of the representative tags in each segment are set from the analysis. The results are obtained from 1000 independent runs.

Performance Verification: We evaluate the proposed protocols under five scenarios. In the simulation, the threshold of the missing items is set to $M_a = 2$ and the required detection reliability varies from $\alpha = 95\%$ to $\alpha = 99\%$ in the first two scenarios and is fixed to $\alpha = 95\%$ in the third scenario. In the fourth scenario, we set $\alpha = 100\%$ and $p_{\text{sample}} = 100\%$ to enable the tag identification of M^{2}ID+, and we show the superior time efficiency of the proposed protocols compared with the state-of-the-art BPI [26]. The above simulations work in the ideal channel, i.e., $p_e = 0$. Therefore, the detection probability equals to the ratio of the correct detection. In the last simulation, we verify the effectiveness of the proposed protocols under the unstable channel and the ratio of the correct detection is set as $\alpha = 95\%$.

1) In the first scenario, there exists ten tags on each item and the number of the tags varies from 1000 to 5000. The simulation results of the detection probability and the execution time are depicted in Fig. 6. The results show that the proposed M^{2}ID and M^{2}ID+ can satisfy the requirement of the detection reliability. Yet, these two protocols have to spend more time on finding a missing item event as the number of the tags in the system increases when there would be more representative tags leading to the longer time cost of the seed

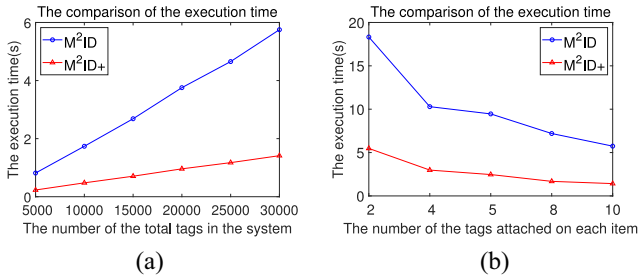

Fig. 8. Time efficiency versus the system scale under $\alpha = 95\%$. (a) Case 1 with the number of the total tags varied from 5000 to 30 000 and ten tags on each item. (b) Case 2 with the number of the tags on each item varied from 2 to 10 and 30 000 tags in total.

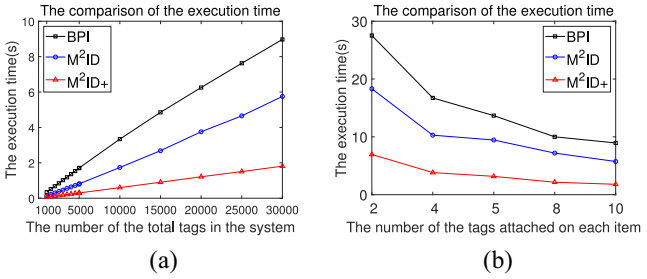

Fig. 9. (a) Execution time with the number of the total tags varied from 1000 to 30 000 and the number of the tags on each item is set to 10. (b) Execution time with the number of the tags on each item varied from 2 to 10 and the number of the total tags is set to 30 000.

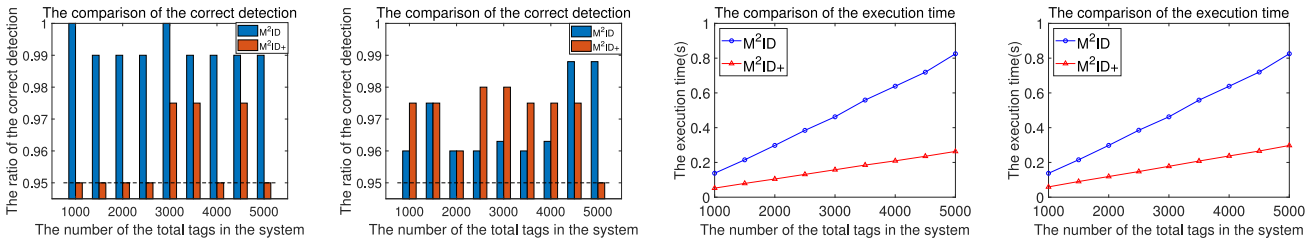

Fig. 10. Ratio of the correct detection and the execution time versus the error probability of the uplink with $p_e = 0.1$ and $p_e = 0.2$. (a) $p_e = 0.1$. (b) $p_e = 0.2$. (c) $p_e = 0.1$. (d) $p_e = 0.2$.

searching and the parameters broadcasting. We can also observe that M^{2}ID+ is more time-efficient. As shown in Fig. 6(c), M^{2}ID+ is faster 3× than M^{2}ID when the number of the total tags is 5000.

2) In the second scenario, we investigate the impact of the number of the tags on one item on the detection probability and the execution time. To this end, we set the total number of the tags in the system as 1000, and vary the number of the tags on each item from 2 to 10. We can draw from Fig. 7 the similar conclusions as in the first scenario that both M^{2}ID and M^{2}ID+ can achieve the required reliability, but the latter is more time-efficient and the gain is 2× at least.
3) In the third scenario, we show the impact of the system scale on the time efficiency. The experiment consists of two cases: The first case witnesses ten tags on each item while the number of the total tags varies from 5000 to 30 000; The second case has 30 000 tags while the number of the tags on each item changes from 2 to 10. They indicate the change of the number of the items. We can observe from Fig. 8 that both M^{2}ID and M^{2}ID+ spend more time as the system scales up, but the increasing speed of M^{2}ID+ is significantly slower.
4) In the fourth scenario, we compare the proposed protocols with the state-of-the-art solution BIP. To this end, we set the sample ratio in M^{2}ID+ to 100%. Fig. 9(a) shows that M^{2}ID+ is most time-efficient and M^{2}ID+ is faster than BIP when the number of the tags with the change of the item population. Similarly, Fig. 9(b) depicts the execution time when the total number of the tags is 30 000 and the number of the tags on each item varies from 2 to 10. Similarly, M^{2}ID+ spends least time among these three protocols. From Fig. 9, M^{2}ID+ is still most effective among them and M^{2}ID+ achieves at least 4x performance gain compared with BPI.
5) In the fifth scenario, we verify the effectiveness of our proposed protocols in the unstable channel where the downlink works in error-free and the probability of bit inversion in the uplink is from $p_e = 0.1$ to $p_e = 0.2$. Fig. 10 illustrates that ten tags on each item while the number of the total tags varies from 1000 to 5000 with different p_e. The ratio of the correct detection degrades in M^{2}ID but still satisfies the requirement of the correct detection. The execution time of M^{2}ID+ grows with the increasing p_e since M^{2}ID+ has to increase the sample ratio to guarantee the correct detection.

VII. Conclusion

This article has addressed a variation on the missing item event detection problem arising from multitagged item in RFID systems. The application of the prior works to the new problem suffers low time efficiency due to repeated checks of one item. To overcome this drawback, we have provided two solutions, namely M^{2}ID and M^{2}ID+, from the perspective of tradeoff between the computation and the communications. They have used the seed selection to ask a subset of the tags in systems to report their presence while M^{2}ID+ can achieve both probabilistic detection and deterministic identification. We have also derived the optimum parameters under the unstable channels. The simulation results have confirmed the superiority in terms of time efficiency under the required detection reliability to the existing state-of-the-art solution.

References

[1] J. Liu, X. Chen, X. Liu, X. Zhang, X. Wang, and L. Chen, "On improving write throughput in commodity RFID systems," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2019, pp. 1522–1530.
[2] J. Yu, W. Gong, J. Liu, L. Chen, K. Wang, and R. Zhang, "Missing tag identification in COTS RFID systems: Bridging the gap between theory and practice," *IEEE Trans. Mobile Comput.*, vol. 19, no. 1, pp. 130–141, Jan. 2020.
[3] L. Zhang, W. Xiang, X. Tang, Q. Li, and Q. Yan, "A time-and energy-aware collision tree protocol for efficient large-scale RFID tag identification," *IEEE Trans. Ind. Informat.*, vol. 14, no. 6, pp. 2406–2417, Jun. 2018.
[4] J. Yu, J. Liu, R. Zhang, L. Chen, W. Gong, and S. Zhang, "Multi-seed group labeling in RFID systems," *IEEE Trans. Mobile Comput.*, vol. 19, no. 12, pp. 2850–2862, Dec. 2020.
[5] J. Zhang *et al.*, "RFHUI: An RFID based human-unmanned aerial vehicle interaction system in an indoor environment," *Digit. Commun. Netw.*, vol. 6, no. 1, pp. 14–22, 2020.
[6] J. Liu, S. Chen, M. Chen, Q. Xiao, and L. Chen, "Pose sensing with a single RFID tag," *IEEE/ACM Trans. Netw.*, vol. 28, no. 5, pp. 2023–2036, Oct. 2020.
[7] X. Liu *et al.*, "Accurate localization of tagged objects using mobile RFID-augmented robots," *IEEE Trans. Mobile Comput.*, vol. 20, no. 4, pp. 1273–1284, Apr. 2021.
[8] *EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communication at 860 MHz–960 MHz*. EPC, El Segundo, CA, USA, 2015.
[9] H. Aftab, K. Gilani, J. Lee, L. Nkenyereye, S. Jeong, and J. Song, "Analysis of identifiers in IoT platforms," *Digit. Commun. Netw.*, vol. 6, no. 3, pp. 333–340, 2020.
[10] N. R. Federation. (2020). *National Retail Security Survey*. [Online]. Available: https://nrf.com
[11] Crime and Tech. (2019). *Retail Security in Europe. Going Beyond Shrinkage*. [Online]. Available: https://checkpointsystems.com/uk/detail/369/uk-retailers-suffer-most-from-shrinkage
[12] L. Bolotnyy and G. Robins, "Multi-tag RFID systems," *Int. J. Internet Protocol Technol.*, vol. 2, no. 3, pp. 218–231, 2007.

[13] S. Dhal and I. Sengupta, "Protocol to authenticate the objects attached with multiple RFID tags," in *Emerging Trends in Computing and Communication*. New Delhi, India: Springer, 2014, pp. 149–156.
[14] L. Shangguan, Z. Yang, A. X. Liu, Z. Zhou, and Y. Liu, "Relative localization of RFID tags using spatial–temporal phase profiling," in *Proc. NSDI*, 2015, pp. 251–263.
[15] J. Liu, H. Dai, Y. Yan, X. Zhang, X. Chen, and L. Chen, "Is this side up? Detecting upside-down exception with passive RFID," in *Proc. IEEE SMARTCOMP*, 2017, pp. 1–2.
[16] D. Hochhalter, D. Bigelow, N. J. Witchey, and C. Milam, "RFID-based rack inventory management systems," U.S. Patent App. 15 725 638, 2018.
[17] C. C. Tan, B. Sheng, and Q. Li, "How to monitor for missing RFID tags," in *Proc. IEEE ICDCS*, 2008, pp. 295–302.
[18] W. Luo, S. Chen, T. Li, and Y. Qiao, "Probabilistic missing-tag detection and energy-time tradeoff in large-scale RFID systems," in *Proc. ACM MobiHoc*, 2012, pp. 95–104.
[19] W. Luo, S. Chen, Y. Qiao, and T. Li, "Missing-tag detection and energy-time tradeoff in large-scale RFID systems with unreliable channels," *IEEE/ACM Trans. Netw.*, vol. 22, no. 4, pp. 1079–1091, Aug. 2014.
[20] M. Shahzad and A. X. Liu, "Expecting the unexpected: Fast and reliable detection of missing RFID tags in the wild," in *Proc. IEEE INFOCOM*, 2015, pp. 1939–1947.
[21] J. Yu, L. Chen, R. Zhang, and K. Wang, "Finding needles in a haystack: Missing tag detection in large RFID systems," *IEEE TCOM*, vol. 65, no. 5, pp. 2036–2047, May 2017.
[22] J. Yu, L. Chen, R. Zhang, and K. Wang, "On missing tag detection in multiple-group multiple-region RFID systems," *IEEE Trans. Mobile Comput.*, vol. 16, no. 5, pp. 1371–1381, May 2017.
[23] T. Li, S. Chen, and Y. Ling, "Identifying the missing tags in a large RFID system," in *Proc. 11th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2010, pp. 1–10.
[24] R. Zhang, Y. Liu, Y. Zhang, and J. Sun, "Fast identification of the missing tags in a large RFID system," in *Proc. IEEE 8th Annu. Commun. Soc. Conf. Sensor Mesh Ad Hoc Commun. Netw.*, 2011, pp. 278–286.
[25] X. Liu, K. Li, G. Min, Y. Shen, A. X. Liu, and W. Qu, "Completely pinpointing the missing RFID tags in a time-efficient way," *IEEE Trans. Comput.*, vol. 64, no. 1, pp. 87–96, Jan. 2015.
[26] X. Xie, X. Liu, H. Qi, and K. Li, "Fast identification of multi-tagged objects for large-scale RFID systems," *IEEE Wireless Commun. Lett.*, vol. 8, no. 4, pp. 992–995, Aug. 2019.
[27] J. Yu, W. Gong, J. Liu, L. Chen, and K. Wang, "On efficient tree-based tag search in large-scale RFID systems," *IEEE/ACM Trans. Netw.*, vol. 27, no. 1, pp. 42–55, Feb. 2019.
[28] J. Yu *et al.*, "Stabilizing frame slotted aloha based IoT systems: A geometric ergodicity perspective," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 3, pp. 714–725, Mar. 2021.
[29] M. O'Neill, "Low-cost SHA-1 hash function architecture for RFID tags," *RFIDSec*, vol. 8, pp. 41–51, Jul. 2008.

Hao Liu received the B.E. degree in information engineering from the Beijing Institute of Technology, Beijing, China, in 2014, where he is currently pursuing the Ph.D. degree.

His research interests include RFID technology.

Rongrong Zhang received the B.E. and M.E. degrees in communication and information systems from the Chongqing University of Posts and Telecommunications, Chongqing, China, in 2010 and 2013, respectively, and the Ph.D. degree in computer science from the University of Paris Descartes, Paris, France, in 2017.

She was a Research Fellow with the School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, ON, Canada. She is an Associate Professor with Capital Normal University, Beijing, China. Her research interests focus on wireless body area networks and Internet of Things.

Lin Chen (Member, IEEE) received the B.E. degree in radio engineering from Southeast University, Nanjing, China, in 2002, the Engineer Diploma degree from Telecom ParisTech, Paris, France, in 2005, and the M.S. degree in networking from the University of Paris, Paris, in 2005.

He currently works as Professor with the School of Computer Science and Technology at Sun Yat-sen University, Guangzhou, China and also an Associate professor with the department of Computer Science of the University of Paris-Sud, Paris. His main research interests include modeling and control for wireless networks, distributed algorithm design, and game theory.

Mr. Chen serves as the Chair of IEEE Special Interest Group on Green and Sustainable Networking and Computing with Cognition and Cooperation, IEEE Technical Committee on Green Communications and Computing.

Jihong Yu (Member, IEEE) received the B.E. degree in communication engineering and the M.E. degree in communication and information systems from the Chongqing University of Posts and Telecommunications, Chongqing, China, in 2010 and 2013, respectively, and the Ph.D. degree in computer science from the University of Paris-Sud, Orsay, France, in 2016.

He was a Postdoctoral Fellow with the School of Computing Science, Simon Fraser University, Burnaby, BC, Canada. He is currently a Professor with the School of Information and Electronics, Beijing Institute of Technology, Beijing, China. His research interests include backscatter networking, Internet of Things, and space-air communications.

Prof. Yu is an Associate Editor of IEEE INTERNET OF THINGS JOURNAL and *Compute Communications* (Elsevier).

Jiangchuan Liu (Fellow, IEEE) received the B.Eng. (*Cum Laude*) from Tsinghua University, Beijing, China, in 1999, and the Ph.D. degree from the Hong Kong University of Science and Technology, Hong Kong, in 2003.

He is currently a Full Professor with the School of Computing Science, Simon Fraser University, Burnaby, BC, Canada.

Prof. Liu is a co-recipient of the Test of Time Paper Award of IEEE INFOCOM in 2015, the ACM TOMCCAP Nicolas D. Georganas Best Paper Award in 2013, and the ACM Multimedia Best Paper Award in 2012. He is a Steering Committee Member of IEEE TRANSACTIONS ON MOBILE COMPUTING, and an Associate Editor of IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON BIG DATA, and IEEE TRANSACTIONS ON MULTIMEDIA. He is a Fellow of the NSERC E.W.R. Steacie Memorial and the Canadian Academy of Engineering.

Jianping An (Member, IEEE) received the Ph.D. degree from the Beijing Institute of Technology, Beijing, China, in 1996.

In 1995, he joined the School of Information and Electronics, Beijing Institute of Technology, where he is currently a Full Professor. He is currently the Dean of the School of Information and Electronics, Beijing Institute of Technology. His research interests are in the field of digital signal processing, wireless networks, and high-dynamic broadband wireless transmission technology.

Qianbin Chen (Senior Member, IEEE) received the Ph.D. degree in communication and information system from the University of Electronic Science and Technology of China, Chengdu, China, in 2002.

He is currently a Professor with the School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing, China, where he is the Director of the Chongqing Key Laboratory of Mobile Communication Technology. He has authored or coauthored over 100 papers in journals and peer-reviewed conference proceedings, and has coauthored seven books. He holds 47 granted national patents.