Efficient RFID Grouping Protocols

Jia Liu, *Member, IEEE,* Min Chen, Bin Xiao, *Senior Member, IEEE,* Feng Zhu, *Member, IEEE,* Shigang Chen, *Fellow, IEEE,* and Lijun Chen, *Member, IEEE*

***Abstract*—The grouping problem in RFID systems is to efficiently group all tags according to a given partition such that tags in the same group will have the same group ID. Unlike previous research on unicast transmission from a reader to a tag, grouping provides a fundamental mechanism for efficient multicast transmissions and aggregate queries in large RFID-enabled applications. A message can be transmitted to a group of m tags simultaneously in multicast, which improves the efficiency by m times when comparing with unicast. This paper studies this practically important but not yet thoroughly investigated grouping problem in large RFID system. We start with a straightforward solution called the Enhanced Polling Grouping (EPG) protocol. We then propose a time-efficient Filter Grouping (FIG) protocol that uses Bloom filters to remove the costly ID transmissions. We point out the limitation of the Bloom-filter based solution due to its intrinsic false positive problem, which leads to our final ConCurrent Grouping (CCG) protocol. With a drastically different design, CCG is able to outperform FIG by exploiting collisions to inform multiple tags of their group ID simultaneously and by removing any wasteful slots in its frame-based execution. We further enhance CCG to make it perform better with very large groups. Simulation results demonstrate that our best protocol CCG can reduce the execution time by a factor of 11 when comparing with a baseline polling protocol.**

***Index Terms*—RFID, grouping, time efficiency**

I. Introduction

RADIO Frequency IDentification (RFID) has been widely deployed for tagged object tracking [2]–[4], supply chain management [5]–[7], and warehouse inventory control [8]–[13]. Grouping of RFID tags can play an important role in improving the performance of RFID-enabled applications. For example, when tags belonging to the same group share a common group ID, the reader can simultaneously transmit the same data to them, greatly saving the communication overhead in comparison with the traditional unicast transmission. In another example, after grouping all tags, the reader can execute effective aggregate queries, such as cardinality estimation [14]–[18] or sensor-data collection [19]–[21], for tags in the same group, dramatically benefiting the functions of inventory management and monitoring.

Jia Liu, Feng Zhu, and Lijun Chen are with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China. E-mail:{liujia,zhufeng}@smail.nju.edu.cn, chenlj@nju.edu.cn.

Min Chen and Shigang Chen are with the Department of Computer & Information Science & Engineering, University of Florida, Gainesville, FL 32611, USA. E-mail:{min,sgchen}@cise.ufl.edu.

Bin Xiao is with the Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong. E-mail:csbxiao@comp.polyu.edu.hk.

A preliminary version [1] of this paper appeared in the proceedings of IEEE INFOCOM'15.

Manuscript received June 30, 2015; revised October 6, 2015.

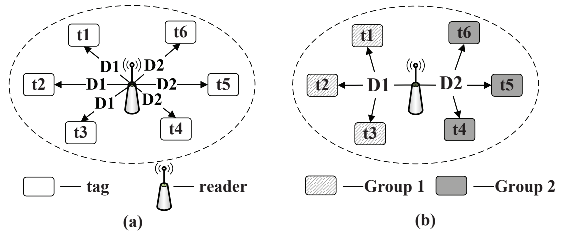


Fig. 1: Multicast transmissions with the grouping scheme.

Given the tag population $\mathcal{G}$ in an RFID system and an arbitrary group partition of $\mathcal{G}$ (i.e., non-overlapping subsets of $\mathcal{G}$), the grouping problem is to efficiently inform all tags in $\mathcal{G}$ about which groups they belong to, such that tags in the same subset will have the same group ID. We use an example to further clarify this problem and its practical significance. Consider multicast transmissions in Fig. 1. The reader in Fig. 1(a) intends to transmit data D_1 to tags t_1, t_2, and t_3, and data D_2 to tags t_4, t_5, and t_6, where the data may be shipment information about tagged objects to be recorded on tags for tracking purpose or queries for reporting different sensor information. The traditional approach is for the reader to unicast the same data to the relevant tags, one at a time. Six data transmissions are needed. In contrast, with the tags being grouped in Fig. 1(b), the reader is able to send D_1 to tags t_1, t_2, and t_3 by one transmission that carries their group ID as the destination address. Similarly, data D_2 can be transmitted to t_4, t_5, and t_6 in one transmission. Two transmissions are needed in total, reducing the overhead by a factor of 3. Therefore, grouping tags is a fundamental mechanism that may greatly improve management efficiency in RFID-enabled applications.

We give a few application examples. Consider a warehouse where goods come in and then are shipped out. Each shipment is a group, and we want to write the same information (for tracking purpose) to the tags in the group. With a grouping protocol, such writing can be made more efficient, which is desirable in order to minimize the interference with other normal operations in a busy warehouse such as moving objects in or out of the area where the writing happens. In another example, the use of access password allows us to prevent unauthorized readers from accessing the tags' information in privacy-sensitive environment. According to the C1G2 RFID standard [22], the length of the access password stored in the tag memory is 16 bits, which is too weak to guard against the brute force attack. Therefore, we may want to frequently update the passwords on tags to strengthen the privacy protection. Assigning each tag a different password is time-consuming. Alternatively, we may split the entire tag set into small groups, and assign the tags in the same group the same new password. The split can be different each time for better security. In this scenario, a grouping protocol as proposed in this paper can help further improve the efficiency in group-based password updates. Finally, consider an RFID programming system that programs the tags before shipping them out. Tags of different purposes will be programmed differently with data written to the tag memory or even code in case of WISPs (Wireless Identification and Sensing Platform) [23] or similar tags. With a large number of pre-packaged blank tags kept in the system, we can dynamically define groups as needed and program tags in each group together, saving the communication overhead of air programming.

Little work studied how to efficiently group RFID tags before. One intuitive solution is to leverage the traditional polling protocol [21]: For each tag, the reader transmits its ID together with the assigned group ID to inform the tag of its group. This protocol is inefficient for a large RFID system because it requires the reader to broadcast a large number of tag IDs and the same number of group IDs. In this paper, we propose a series of protocols to progressively improve the grouping performance. We first present an Enhanced Polling Grouping (EPG) protocol that avoids repeatedly transmitting the same group ID, improving the grouping efficiency over the traditional polling protocol. We then propose a Filter Grouping (FIG) protocol that uses Bloom filters [24] to avoid transmitting the tag IDs. We address the negative impact of the false positive problem (which is intrinsic to any Bloom filter), and determine the optimal system parameters through a joint optimization to minimize the protocol execution time. We finally propose a more scalable and efficient ConCurrent Grouping (CCG) protocol that avoids the false positive problem and can simultaneously label tags of different groups with their respective group IDs in a single time frame, which is fundamentally different from the one-group-at-a-time approach by FIG. Moreover, CCG is capable of exploiting collisions to label multiple tags in one slot. The efficiency is further improved by leveraging an ordering vector to eliminate any slot waste. We derive an upper bound for the execution time of CCG, which is equivalent to transmitting $(0.028 + 0.018\times\lceil \log_2 k\rceil)\times n$ tag IDs, much faster than transmitting n tag IDs as well as n group IDs in the traditional polling protocol, where n is the number of tags in the system, k is the number of groups, an tag ID is 96 bits long, and a group ID can be indexed by $\lceil \log_2 k\rceil$ bits. Finally, we enhance CCG to make it perform better with very large groups.

We conduct extensive simulations based on the specification of the EPC C1G2 standard [22]. The simulation results show that for grouping a total of 10,000 RFID tags in 100 groups, the execution time of the traditional polling protocol is 44.6s. EPG reduces the execution time to 39.1s. FIG further shortens the execution time to 7.4s. CCG performs best and takes only 3.9s, improving the grouping efficiency by a factor of 11 when comparing with the traditional polling protocol. It is thus more suitable for real-time RFID-enabled applications.

The rest of the paper is organized as follows. Section II defines the grouping problem and gives a straightforward solution. Section III proposes a filtering grouping protocol. Section IV presents a more efficient concurrent grouping protocol. Section V enhances the concurrent grouping protocol. Section VI generalizes the single grouping problem to multiple grouping problem. Section VII evaluates the performance of the proposed protocols. Section VIII discusses the related work. Finally, Section IX concludes this paper.

II. Problem Statement

A. System Model

An RFID system consists of one or multiple readers and a large number of tags. The readers are connected with a backend server for information storage and computation. Each tag has a unique tag ID. It can communicate with a reader directly. But tags cannot communicate amongst themselves. We can logically treat the readers as one if they are well synchronized and scheduled [25]. To simplify the description, our protocols are presented for a single reader, but they can be easily modified for multiple readers when the collision-free transmission schedule among the readers is established.

We assume that the reader has the knowledge of all tag IDs as a priori [19], [26], [27]. The tag IDs can be automatically collected through one of the numerous existing tag identification protocols [28]–[30].

In a large RFID system, tagged objects may be classified into groups by their categories (e.g., shoes or bags), properties (e.g., shoe sizes), manufacturers, arrival/departure dates, or other criteria. Grouping objects facilitates the inventory process and benefits the warehouse management because we can easily carry out operations for particular groups based on their group IDs. For example, once we have informed tags about their group IDs, we can transmit a message to tags in one group by using their group ID as the destination address, which is much more efficient than sending each tag in the group a separate message using the tag ID as the destination address.

B. Problem Definition

Consider a large RFID system of n tags. Denote the tag set as $\mathcal{G} = \{t_1, t_2, ..., t_n\}$. A partition of the set $\mathcal{G}$ is a family of disjoint sets $\mathcal{P} = \{P_1, P_2, ..., P_k\}$ such that $\bigcup_{i=1}^{k} P_i = \mathcal{G}$. We refer to P_i as a *group* and each tag in $\mathcal{G}$ exactly belongs to one group. There are k groups in $\mathcal{P}$.

The grouping problem is to efficiently label all RFID tags in $\mathcal{G}$ according to $\mathcal{P}$, such that tags in the same group will carry the same group ID. More specifically, the reader is instructed by the user with the partition $\mathcal{P}$, and it is supposed to inform all tags in the same group P_i about their group IDs g_i, $1 \le i \le k$, where different groups should have different group IDs.

In today's practice, when a tag ID is written, a portion of the prefix in the ID can serve as a static group ID. This works when we manually program tags one by one before deployment. This paper studies dynamic grouping based arbitrary partition after tags are deployed. Certainly we can still use a portion of the prefix in the tag ID as its group ID. In that case, we will have to overwrite that portion for regrouping.

C. A Naïve Solution

In the Traditional Polling Grouping (TPG) protocol, the reader first separates a tag from others by broadcasting its ID and then transmits the corresponding group ID to label this tag. The same group ID will be repeatedly broadcast for labeling multiple tags in a group. We now present an Enhanced Polling Grouping (EPG) protocol that avoids transmitting any group ID repetitively.

EPG contains k grouping rounds. In each round, the reader polls all tags in a single group. Consider an arbitrary round for grouping P_i, $1 \leq i \leq k$. The reader broadcasts IDs of all tags belonging to P_i in turn. Each unlabeled tag keeps listening to the wireless channel. Only when the tag receives its own ID, it transitions from the *unlabeled* state to the *marked* state. After polling all tags in P_i, the reader labels the marked tags by broadcasting the group ID g_i and these tags transition to the *labeled* state. Each labeled tag then keeps silent while others stay active for participating in the subsequent rounds. Fig. 2 illustrates the state diagram of an RFID tag in the EPG protocol, with the initial state being the unlabeled state.

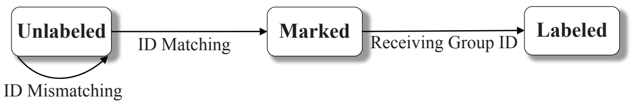


Fig. 2: State diagram of an RFID tag in EPG.

Note that the major difference between TPG and EPG is that EPG transmits each group ID only once. Let t_{id} be the length of a time slot that transmits a 96-bit tag ID [22], and t_{gid} be the length of a slot for transmitting a group ID, where a tag ID is much longer than a group ID. The total execution time of TPG is $n \times (t_{id} + t_{gid})$, where n is the number of tags. In comparison, the execution time of EPG is $n \times t_{id} + k \times t_{gid}$.

Although EPG can improve the grouping efficiency over TPG, it still has to painstakingly transmit n tag IDs, resulting in long running time under a large tag set. Hence, we seek novel grouping protocols to quickly group a large number of tags.

III. Filter Grouping Protocol

The Efficient Tag-Ordering Polling protocol (ETOP) in [21] was designed to collect sensor data from a subset of tags in a large sensor-augmented tag system. It uses Bloom filters to encode the subset and broadcast the filters to the tags for two purposes: (1) Informing the tags in the subset that they need to report their sensor data; (2) establishing an order among the tags so that they can take turns to report their data, free of collision. The primary objective is to achieve energy efficiency, assuming battery-powered active tags. ETOP effectively establishes one group. If we apply it repetitively, it can be used to establish multiple groups.

For energy efficiency, ETOP uses a large number of small Bloom filters instead of a single large one to encode a subset of tags (such that each tag reads one small filter instead of a common, large filter). However, breaking a large Bloom filter into many smaller ones will increase the overall false positive ratio. Moreover, ETOP's filters are partitioned Bloom filters designed for establishing an order among the tags in the subset. Partitioned Bloom filters have slightly higher false positive ratio than their standard counterpart.

Since this paper does not need to establish an order among tags in each group and our goal is time efficiency instead of energy efficiency, directly using ETOP for our purpose is inefficient. Consequently, we redesign a Bloom-filter based solution in this section, using a single standard Bloom filter, called the Filter Grouping protocol (FIG), to avoid most ID broadcasting. The protocol is introduced as a performance benchmark for comparison with the main contribution of this paper in the following sections.

FIG is also similar to CATS [25] in its Bloom filter construction, except that CATS encodes a set X of wanted tags (which may contain many tags not in the system) in a Bloom filter and its goal is to find the intersection of X and the set of tags currently in the system.

A. Basic Idea

The idea is to separate tags in one group at a time from other groups by using a space-efficient Bloom filter [24]. As the reader broadcasts a filter encoding one group to all tags in the system, the tags in the encoded group will be correctly marked. Some tags in other groups may also be marked mistakenly due to the false positives of Bloom filters. Because the reader has both the filter and all tag IDs, it can predict the mis-marked ones and can thus inform them to unmark by transmitting their IDs in an additional phase, which can however cause significant overhead. We may reduce the unmarking overhead by lowering the false positive ratio with a larger filter, at the expense of increasing the filtering overhead. The key is to perform a joint optimization to minimize the combined overhead of filtering and unmarking. The end result is a protocol that is far superior than EPG. Moreover, we need to consider the order of the groups in which the Bloom filters are applied, which also affects the overall execution time.

B. Protocol Overview

FIG consists of k grouping rounds, each of which deals with one group in $\mathcal{P}$ with three phases: *filtering phase*, *polling phase*, and *labeling phase*. *1)* In the filtering phase, the reader broadcasts a filter that encodes tags in a group, and only tags passing this filter will transition from the unlabeled state to the marked state. Transitions 1 and 2 in Fig. 3 depict this phase. *2)* The polling phase is to unmark all incorrectly marked tags caused by false positives. The reader broadcasts these tags' IDs. Upon receipt of their IDs, the tags move back to the unlabeled state. Transitions 3 and 4 in the figure illustrate this phase. *3)* In the labeling phase, the reader labels the remaining marked tags by broadcasting the group ID. These tags then transition from the marked state to the final labeled state. Other unlabeled tags will participate in and be grouped by subsequent rounds.

C. Protocol Details

Consider the ith round for grouping $P'_i \in \mathcal{P}$, $1 \leq i \leq k$. The reason for using P'_i instead of P_i is to show that the order

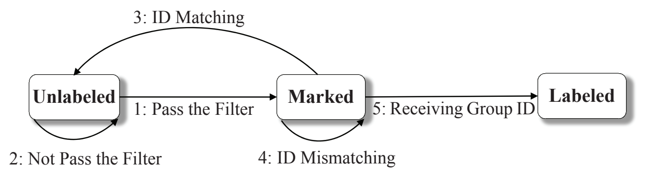


Fig. 3: State diagram of an RFID tag in FIG.

of groups in the rounds does not have to follow the order of groups defined in $\mathcal{P}$. Let G_i be the set of unlabeled tags at the beginning of this round.

1) Filtering Phase: This phase approximately marks all tags in P'_i using a Bloom filter. The reader first constructs a Bloom filter by mapping each tag in P'_i to k_i bits in an L_i-bit vector and setting those bits to ones, where mapping is done by hashing the ID of the tag. The optimal values of L_i and k_i will be determined shortly. The filter is denoted as $BF(P'_i)$. The reader broadcasts the values of L_i and k_i first, and then the filter $BF(P'_i)$. If the filter is too long, the reader can split it into 96-bit segments and transmit each of them in a time slot of length t_{id} [19]. Each unlabeled tag in G_i hashes its own ID to k_i bit positions in the filter, and thus knows which segments it needs to listen. If all those k_i bits in $BF(P'_i)$ are ones, the tag passes the filter and transitions to the marked state. Otherwise, the tag remains in the unlabeled state. We denote the set of tags in the marked state as $M_i(\subseteq G_i)$.

2) Polling Phase: A Bloom filter does not have false negatives, meaning that $P'_i \subseteq M_i$. However, it may have false positives, namely, a tag in M_i may not be in P'_i. Knowing the filter $BF(P'_i)$ and the unlabeled tag set G_i, the reader can predict the subset $M_i - P'_i$ of marked tags that should be unmarked. The reader then broadcasts the IDs of the tags in $M_i - P'_i$ one after another. When receiving their IDs, these incorrectly marked tags will transition back to the unlabeled state. After this phase, all remaining marked tags belong to P'_i.

3) Labeling Phase: In the final phase, the reader broadcasts the group ID of P'_i to notify all marked tags which group they belong to. When receiving the group ID, the marked tags move to the labeled state. The protocol then enters the next round.

D. Optimal Parameter Setting

We give the optimal values of L_i and k_i in the following theorem.

Theorem 1. *Let n_i be the number of unlabeled tags in G_i and m'_i be group size of P'_i. The optimal filter length L_i and the optimal number k_i of hash functions for the ith round, $\forall 1 \le i \le k$, are*

$$\begin{aligned} k_i &= \ln 2 \times \frac{L_i}{m'_i} \\ L_i &= \frac{m'_i}{(\ln 2)^2} \times \ln(96 \times (\ln 2)^2 \times \frac{n_i - m'_i}{m'_i}), \end{aligned} \tag{1}$$

which minimize the execution time of the ith round to

$$T(m'_i, n_i) = \frac{L_i}{96} \times t_{id} + (n_i - m'_i) \times 0.6185^{\frac{L_i}{m'_i}} \times t_{id} + t_{gid}. \tag{2}$$

Proof. Consider the ith grouping round, where $1 \le i \le k$. In the filtering phase, the reader takes $\frac{L_i}{96} \times t_{id}$ time to transmit an L_i-bit filter.[1] In the polling phase, the reader needs to poll $|G_i - P'_i| \times f_i$ improperly marked tags, where f_i is the false positive rate of the Bloom filter. Since $P'_i \subseteq G_i$, $|G_i - P'_i| = |G_i| - |P'_i| = n_i - m'_i$. The polling time in this phase is thus equal to $(n_i - m'_i) \times f_i \times t_{id}$. In the labeling phase, the reader takes a time slot of t_{gid} to transmit a group ID. We thus have the total execution time $T(m'_i, n_i)$ of this round:

$$T(m'_i, n_i) = \frac{L_i}{96} \times t_{id} + (n_i - m'_i) \times f_i \times t_{id} + t_{gid}.$$

Given L_i, n_i, and m'_i, $T(m'_i, n_i)$ increases monotonously with the false positive rate f_i. We want to decrease f_i as much as possible, so as to minimize $T(m'_i, n_i)$. It is well known that

$$f_i = \big(1 - (1 - \frac{1}{L_i})^{k_i m'_i}\big)^{k_i} \approx \big(1 - e^{\frac{-k_i m'_i}{L_i}}\big)^{k_i}.$$

Let the first-order derivative of f_i be 0. We can derive the minimal $f_i = 0.6185^{\frac{L_i}{m'_i}}$ when $k_i = \ln 2 \times \frac{L_i}{m'_i}$. We thus have the execution time of the ith round under the optimal k_i as follows.

$$T(m'_i, n_i) = \frac{L_i}{96} \times t_{id} + (n_i - m'_i) \times 0.6185^{\frac{L_i}{m'_i}} \times t_{id} + t_{gid}.$$

Given m'_i and n_i, let $\frac{dT(m'_i, n_i)}{dL_i} = 0$. We can derive the minimal execution time $T(m'_i, n_i)$ in (2) when L_i is equal to $\frac{m'_i}{(\ln 2)^2} \times \ln(96 \times (\ln 2)^2 \times \frac{n_i - m'_i}{m'_i})$. □

E. Order of Grouping

Although we can minimize the execution time of each single round according to (1) and (2), different group sequences will lead to different global execution time, where a sequence among groups in $\mathcal{P}$ gives the order in which the rounds are applied. It is however a challenging task to find the optimal sequence. A straightforward solution is to exhaustively search all possible group sequences, compute their execution time, and find out the optimal sequence. However, there exist $k!$ permutations among k groups in $\mathcal{P}$, which makes the straightforward solution unscalable to a large k.

We propose a greedy group ordering scheme that finds a near-optimal group sequence (see Section VII-B1). This scheme takes the candidate group with minimal grouping overhead as the next to be grouped. More formally, the greedy scheme is to form an ordered grouping sequence $P'_1, P'_2, ..., P'_k$ satisfying $T(m'_i, n_i) \le T(m'_j, n_i), 1 \le i \le j \le k$, where $m'_1, m'_2, ..., m'_k$ are the group sizes. Recall that $T(m'_j, n_i)$ denotes the execution time to label the tags belonging to P'_j in the ith round and $T(m'_i, n_i) \le T(m'_j, n_i)$ means that the group P'_i is the best choice for the current round since the grouping overhead is minimum among all unlabeled groups. Consider the ith grouping round. There are $(k-i+1)$ unlabeled groups left since the reader has labeled $(i-1)$ groups in the previous $(i-1)$ rounds. Suppose that the group size

[1]For the purpose of clarity, we ignore the negligible communication overhead of transmitting L_i and k_i, since they generally take only a couple of bytes to encode [25].

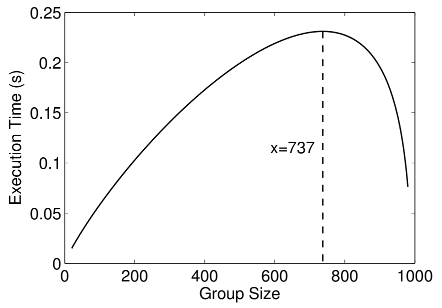


Fig. 4: Execution time with respect to the group size of an unlabeled group.

of each unlabeled group is $q_1, q_2, ..., q_{k-i+1}$. Without loss of generality, we assume that $q_1 \leq q_2 \leq ... \leq q_{k-i+1}$. We then have the following Theorem.

Theorem 2. *The next group to be labeled must be one of the two groups whose group sizes are q_1 and q_{k-i+1}.*

Proof. Consider $T(x, n_i)$ as a function of group size x. It is defined by (2) with m'_i replaced by x. From $\frac{dT(x,n_i)}{dx} = 0$, we can derive that the maximum (or minimum) execution time is reached when

$$x = \frac{lambertw(0, e^{5\times \ln 2 + \ln 3 + 2\times \ln(\ln 2)})}{lambertw(0, e^{5\times \ln 2 + \ln 3 + 2\times \ln(\ln 2)}) + 1} \times n_i \approx 0.7369 n_i, \tag{3}$$

where $lambertw(0, x)$ indicates the main branch[2] of the Lambert W function [31] at the elements of x. Since $lim_{x\to 0} T(x, n_i) = t_{gid}$ and $T(0.7369\times n_i, n_i) = 0.0607\times n_i \times t_{id} + t_{gid}$, $T(0.7369\times n_i, n_i)$ is greater than $lim_{x\to 0} T(x, n_i)$ when $n_i \geq 1$, suggesting that $T(0.7369\times n_i, n_i)$ is the function $T(x, n_i)$'s maximum value for a given $n_i \geq 1$. In other words, $T(x, n_i)$ first increases with x. After peaking at $0.7369\times n_i$, it monotonously declines. Since $q_1 \leq q_2 \leq ... \leq q_{k-i+1}$, the minimal grouping overhead $T(x, n_i)$ in the ith round must be $T(q_1, n_i)$ or $T(q_{k-i+1}, n_i)$. □

Fig. 4 illustrates the execution time with respect to the group size of an unlabeled group, where $k = 10$, $n_i = 1000$, $t_{id} = 3.8ms$, and $t_{gid} = 0.4ms$ (the parameter setting follows the EPC C1G2 standard [22], see Section VII-A). We can clearly see that the execution time $T(x, 1000)$ increases as the group size x increases. After reaching the maximum when $x = 0.7369\times n_i \approx 737$, the execution time decreases with x.

Algorithm 1 gives the pseudo code for ordering groups of sizes $m'_1, m'_2, ..., m'_k$ in a near-optimal sequence. We will show that the performance of such an ordered group sequence is close to the optimal in Section VII-B1.

IV. Concurrent Grouping Protocol

A. Motivation

Although FIG improves the grouping performance by using Bloom filters, it has to separately deal with one group at a time. A filter can successfully label all tags in the group P'_i that it encodes, but some tags in other groups may be mistakenly marked due to false positives, which can be logically considered as *collision* in the filter between tags outside P'_i and tags in P'_i. The problem is that there may be a lot more tags outside P'_i than those inside, which means that the number of incorrectly marked tags could be even larger than the number in P'_i, causing significant polling (unmarking) overhead, unless we make the false positive ratio of the filter sufficiently small. Lowering the false positive ratio is not free; it increases the size of the filter.

[2]In mathematics, the Lambert W function is the inverse relation of the function $f(x) = xe^x$. Since the mapping of $x \mapsto xe^x$ is not injective, the Lambert W function consists of a set of branches. In particular, the main branch is defined for $x \in [-e^{-1}, \infty]$.

Algorithm 1: Group Ordering Scheme

Input: M: sizes of all groups in $\mathcal{P}$; each element in M corresponds to a group
Output: an ordered sequence of groups

```
1:  while $M \neq \emptyset$ do
2:      $x^- = min(M)$;
3:      $x^+ = max(M)$;
4:      $n_i = sum(M)$;
5:      if $T(x^-, n_i) \leq T(x^+, n_i)$ then
6:          $m'_i = x^-$;
7:      else
8:          $m'_i = x^+$;
9:      end if
10:     $M = M - \{m'_i\}$;
11:     place the corresponding group as the next in the
        sequence;
12: end while
```

The above dilemma is fundamentally caused by the choice of Bloom filters and the one-group-at-a-time strategy in our protocol design. To further improve the performance, we need to explore other radically different ideas: labeling tags in all groups together, making some collisions useful so that multiple tags can be labeled together in one slot, and identifying unusable collisions beforehand so that they can be avoided without incurring actual overhead. These ideas form the basis of our next protocol, called the ConCurrent Grouping (CCG) protocol.

B. Protocol Description

CCG also consists of multiple grouping rounds, each of which has an *ordering phase* and a *labeling phase*. The ordering phase tells tags whether and when they will be labeled in the current round, and the labeling phase transmits group IDs to label the tags. Details are given below.

1) Ordering Phase: The reader first broadcasts a request with parameters $\langle f, r\rangle$, where f is the number of slots in a *virtual frame* and r is a random seed. Note that the virtual frame will never be actually played out. It only serves as a vehicle for finding useful slots, and later an actual frame of only the useful slots will be carried out.

Upon receiving this request, each unlabeled tag randomly picks a slot whose index is $H(id, r) \mod (f + 1)$, where id is the tag's ID and $H(\cdot)$ is a hash function. Slots picked by no tag, exactly one tag, and multiple tags are called *empty slot*, *singleton slot*, and *collision slot*, respectively. For our protocol, useful slots are singletons or collision slots that are picked by tags from the same group. These slots are also called *homogeneous slots*. Collision slots picked by tags from different groups are called *heterogeneous slots*, which are not useful. Empty slots are certainly not useful, either.

Take Fig. 5 for example. The second slot is homogeneous since it is chosen by t_1 and t_4, which belong to the same group P_1. So does the sixth slot. In contrast, the fourth slot is heterogeneous as its two tags, t_2 and t_5, are from two different groups. The remaining slots are empty slots.

The tags that pick homogeneous slots are called *homogeneous tags*. Each tag does not know whether it has picked a homogeneous slot or not, but the reader does. With the tag ID information, the reader can predict which slots in the virtual frame are empty, homogeneous, or heterogeneous. It will remove the empty and heterogeneous slots before initiating the frame to tell the tags their group IDs. More specifically, the reader informs the tags which slots are removed by broadcasting an *ordering vector* V of f bits [21], with one bit for each slot in the virtual frame, 0 for empty or heterogeneous and 1 for homogeneous. If V is too long, the reader can split it into 96-bit segments and transmit each segment in a time slot of length t_{id} [19]. For instance, the ordering vector V for the example in Fig. 5 is '010001'. The actual frame to be carried out contains only two homogeneous slots, the second slot and the sixth slot, indicating by the two 1s.

From a tag's perspective, the ordering vector V carries two pieces of information. For one, the tag can learn whether the slot it picks is homogeneous or not by examining the corresponding bit in V. Only if it is, the tag transitions from the unlabeled state to the marked state. For the other, V tells the index of a homogeneous slot in the actual frame to be carried out. If a marked tag finds that there are i ones in V preceding its bit, the tag knows that it picks the $(i + 1)$th homogeneous slot.

2) Labeling Phase: Only the marked tags participate in this phase. Let h be the number of homogeneous slots. The reader initiates an actual labeling frame of h slots, and transmits a group ID in each slot for the tag(s) that pick the slot. Because the tag(s) in each slot are from the same group, the reader can label them simultaneously. From the ordering vector, each marked tag knows which slot it picks in the frame and thus receives its group ID from that slot. For example, in Fig. 5, the actual frame contains only two homogeneous slots. In the first slot, the reader broadcasts P_1's group ID to label t_1 and t_4. In the second slot, the reader broadcasts P_2's group ID to label t_3. No slot is wasted in the actual frame.

After the labeling phase, the current grouping round terminates and the above two phases repeat round after round until all tags are labeled.

C. Parameter Setting and Performance Analysis

We determine the optimal value of f. Consider an arbitrary grouping round. Recall that h is the number of homogeneous slots. Let ψ be the number of homogeneous tags (which have picked the homogeneous slots). The execution time t of this round is:

$$t = \frac{f}{96}\times t_{id} + h\times t_{gid}. \tag{4}$$

We define the *grouping efficiency*, denoted as λ, as the ratio of the number of homogeneous tags to the execution time of this round:

$$\lambda = \frac{\psi}{t} = \frac{\psi}{\frac{f}{96}\times t_{id} + h\times t_{gid}}. \tag{5}$$

Clearly, the bigger the value of λ is, the more the tags will be labeled in each unit of execution time. We thus need to find the optimal f that maximizes λ.

Let $m'_1, m'_2, ..., m'_k$ be the numbers of unlabeled tags in groups $P_1, P_2, ..., P_k$ respectively at the beginning of the round. Let n' be their sum, i.e., $n' = \sum_{i=1}^{k} m'_i$. We give the expected number h of homogeneous slots and the expected number ψ of homogeneous tags in the following two theorems.

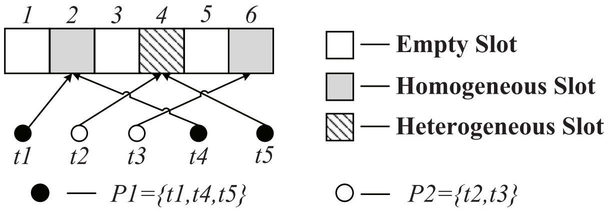


Fig. 5: Three kinds of slots in the ordering phase.

Theorem 3. *With a virtual frame of f slots, the expected number of homogeneous slots is*

$$h = f\times \sum_{i=1}^{k} \left((1-\frac{1}{f})^{n'-m'_i} \times (1-(1-\frac{1}{f})^{m'_i})\right). \tag{6}$$

Proof. Consider the group P_i, $1\leq i\leq k$. The probability that all m'_i tags in P_i do not pick a given slot is $(1-\frac{1}{f})^{m'_i}$. Hence, the probability that at least one tag in P_i picks this slot is $(1-(1-\frac{1}{f})^{m'_i})$. To make this slot homogeneous, the other $(n'-m'_i)$ tags are not supposed to pick this slot, that is $(1-\frac{1}{f})^{n'-m'_i}$. Therefore, the probability that this slot is chosen by only tags coming from P_i is $(1-\frac{1}{f})^{n'-m'_i}(1-(1-\frac{1}{f})^{m'_i})$. With k groups, the probability that a slot is homogeneous is $\sum_{i=1}^{k}\left((1-\frac{1}{f})^{n'-m'_i}(1-(1-\frac{1}{f})^{m'_i})\right)$. There are f slots, and we thus have the final expression of h as shown in (6). □

Theorem 4. *With a virtual frame of f slots, the expected number of homogeneous tags is*

$$\begin{aligned}\psi &= f\times \sum_{i=1}^{k}\sum_{j=0}^{m'_i}(j\times \binom{m'_i}{j}\times (\frac{1}{f})^{j}\times (1-\frac{1}{f})^{n'-j}) \qquad (7)\\ &= f\times \sum_{i=1}^{k}(1-\frac{1}{f})^{n'-m'_i}\sum_{j=0}^{m'_i}(j\times \binom{m'_i}{j}\times (\frac{1}{f})^{j}\times (1-\frac{1}{f})^{m'_i-j})\\ &= \sum_{i=1}^{k}(m'_i\times (1-\frac{1}{f})^{n'-m'_i}).\end{aligned}$$

Proof. Given j tags belonging to P_i, the probability that a slot is picked by only these tags is $(\frac{1}{f})^{j}\times(1-\frac{1}{f})^{n'-j}$. Because there are $\binom{m'_i}{j}$ possible combinations for j tags, the probability that a certain slot is exactly mapped by j tags from P_i is

$\binom{m'_i}{j}\times(\frac{1}{f})^j\times(1-\frac{1}{f})^{n'-j}$. With j ranging from 0 to m'_i, the expected number of tags in P_i (excluding tags outside P_i) mapping to this slot is $\sum_{j=0}^{m'_i} j\binom{m'_i}{j}(\frac{1}{f})^j(1-\frac{1}{f})^{n'-j}$. By extracting the common factor $(1-\frac{1}{f})^{n'-m'_i}$, we have the expression $\sum_{j=0}^{m'_i} j\times\binom{m'_i}{j}\times(\frac{1}{f})^j\times(1-\frac{1}{f})^{m'_i-j}$. Hence, the expected value of the variable X follows the binomial distribution with parameters m'_i and $\frac{1}{f}$, i.e., $X\sim B(m'_i,\frac{1}{f})$. Since $E(X)=\frac{m'_i}{f}$, the expected number of homogeneous tags belonging to P_i in a slot is $\frac{m'_i}{f}\times(1-\frac{1}{f})^{n'-m'_i}$. Considering all k groups, the expected number of homogeneous tags in a slot is $\sum_{i=1}^{k}\big((1-\frac{1}{f})^{n'-m'_i}\times\frac{m'_i}{f}\big)$. With f slots in the ordering phase, the total expected number of homogeneous tags in this phase is $\sum_{i=1}^{k}\big(m'_i\times(1-\frac{1}{f})^{n'-m'_i}\big)$. □

Substituting h and ψ in (5) with (6) and (7), we have the grouping efficiency λ in this round.

$$\lambda=\frac{\sum_{i=1}^{k}\big((1-\frac{1}{f})^{n'-m'_i}\times\frac{m'_i}{f}\big)}{\frac{t_{id}}{96}+t_{gid}\times\sum_{i=1}^{k}(1-\frac{1}{f})^{n'-m'_i}(1-(1-\frac{1}{f})^{m'_i})}. \quad (8)$$

It is challenging to directly derive the maximal λ from (8). We instead find an interval of f where the grouping efficiency λ is maximized, and we then search the optimal f in this interval.

Theorem 5. *When λ attains the maximum, f must be in the interval [1,$n'(e+1.09\beta)$], where e is the natural constant, n' is the number of unlabeled tags, and $\beta=\frac{96\times t_{gid}}{t_{id}}$.*

Proof. Consider any two frame sizes f_1 and f_2, $f_1\leq f_2$. Let ψ_1 and ψ_2 be the corresponding expected numbers of homogeneous tags respectively; h_1 and h_2 be the expected numbers of homogeneous slots respectively. We have the group efficiency λ from (5):

$$\begin{cases}\lambda(f_1)=\frac{\psi_1}{\frac{f_1}{96}\times t_{id}+h_1\times t_{gid}}\\ \lambda(f_2)=\frac{\psi_2}{\frac{f_2}{96}\times t_{id}+h_2\times t_{gid}}.\end{cases}$$

Let $\lambda(f_1)-\lambda(f_2)\geq 0$, we have:

$$f_2\geq\frac{\psi_2}{\psi_1}(f_1+\beta h_1)-\beta h_2. \quad (9)$$

Given a frame size f, the expected number of singleton slots is $n'\times(1-\frac{1}{f})^{n'-1}\approx n'\times e^{-\frac{n'}{f}}$ [16], [32]. As aforementioned, a singleton slot must be a homogeneous slot, and thus we have

$$n'\times e^{-\frac{n'}{f_1}}\leq h_1. \quad (10)$$

Since at least one tag resides in a homogeneous slot, the number of homogeneous tags is no less than that of homogeneous slots for sure. On the other hand, the number of homogeneous tags cannot exceed the total number of unlabeled tags. We have

$$\begin{cases}h_1\leq\psi_1\leq n'\\ h_2\leq\psi_2\leq n'.\end{cases} \quad (11)$$

Consider the slots in the virtual frame. The number of homogeneous slots must be no greater than the number of non-empty slots (chosen by one or more tags). Given a frame size f, the expected number of empty slots is $f\times(1-\frac{1}{f})^{n'}\approx f\times e^{-\frac{n'}{f}}$, and the number of non-empty slots is $f-f\times e^{-\frac{n'}{f}}$. Hence, we have

$$h_1\leq f_1\times(1-e^{-\frac{n'}{f_1}}). \quad (12)$$

Intuitively, the number of homogeneous slots increases as the frame size increases. This conclusion can be easily proved if we can give the proof that the number of homogeneous slots chosen by a single group increases with the frame size. Consider an arbitrary group with m unlabeled tags. According to (6), we have the number $h(m,f)$ of homogeneous slots chosen by this group:

$$h(m,f)=f\times((1-\frac{1}{f})^{n'-m}-(1-\frac{1}{f})^{n'}).$$

Based on this equation, we have

$$\begin{aligned}\frac{dh(m,f)}{df}&=(\tfrac{f-1+n'-m}{f})(1-\tfrac{1}{f})^{n'-m-1}\\&\quad-(\tfrac{f-1+n'}{f})(1-\tfrac{1}{f})^{n'-1}.\end{aligned}$$

Let $g(x)=(\frac{f-1+x}{f})(1-\frac{1}{f})^{x-1}$. We have

$$\frac{dg(x)}{dx}=(1-\frac{1}{f})^{x-1}(\frac{1}{f}+(1+\frac{x-1}{f})\ln(1-\frac{1}{f})).$$

According to the Taylor series for $\ln(1-\frac{1}{f})$, we have

$$\ln(1-\frac{1}{f})=\sum_{i=1}^{\infty}\frac{-1}{i\times f^i},\quad\text{for }|\frac{1}{f}|<1.$$

Hence,

$$\frac{dg(x)}{dx}<(1-\frac{1}{f})^{x-1}(\frac{1}{f}+(1+\frac{x-1}{f})\times\frac{-1}{f})\leq 0.$$

Therefore, the function $g(x)$ monotonically decreases with x. Since $(n'-m)$ is less than n', $\frac{dh(m,f)}{df}=g(n'-m)-g(n')>0$. That means $h(m,f)$ monotonically increases with the frame size f. As $h=\sum_{i=1}^{k}h(m_i,f)$, we have

$$h_1\leq h_2. \quad (13)$$

Let $\Gamma=\frac{\psi_2}{\psi_1}(f_1+\beta h_1)-\beta h_2$. Clearly, Γ increases with ψ_2 and f_1, while decreasing with ψ_1 and h_2. From (10), (11), (12), and (13), we have:

$$\begin{aligned}\Gamma&\leq\frac{\psi_2}{\psi_1}(f_1+\beta h_1)-\beta h_1\\&=\frac{\psi_2}{\psi_1}f_1+\beta h_1(\frac{\psi_2}{\psi_1}-1)\\&\leq\frac{n'}{n'e^{-\frac{n'}{f_1}}}f_1+\beta f_1(1-e^{-\frac{n'}{f_1}})(\frac{n'}{n'e^{-\frac{n'}{f_1}}}-1)\\&=f_1e^{\frac{n'}{f_1}}+\beta f_1(e^{\frac{n'}{f_1}}+e^{-\frac{n'}{f_1}}-2). \quad (14)\end{aligned}$$

Let $\gamma(f)=fe^{\frac{n'}{f}}+\beta f(e^{\frac{n'}{f}}+e^{-\frac{n'}{f}}-2)$. According to (9) and (14), the grouping efficiency $\lambda(f_1)$ is greater than $\lambda(f_2)$ if $f_2\geq\gamma(f_1)\geq\Gamma$. We set $f_1=n'$ and derive $\gamma(f_1)=n'(e+1.09\beta)$. In other words, when the frame size f is no less than $n'(e+1.09\beta)$, there must be a frame size $f_1=n'$ making $\lambda(f_1)>\lambda(f)$ always hold. □

Based on Theorem 5, we can numerically compute the optimal f that maximizes λ in (8). As an example, Fig. 6

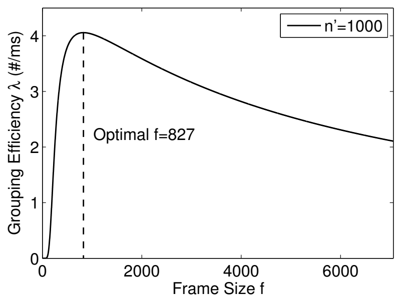


Fig. 6: Grouping efficiency λ with respect to the frame size f.

shows the grouping efficiency with respect to f. In this figure, $\beta = 4$, $t_{id} = 3.8ms$, and the number n' of unlabeled tags is 1,000, which are evenly distributed among 10 groups, i.e., each group has 100 tags. As we can see, the grouping efficiency λ attains the maximum when f is equal to 827, which is in the interval [1,7078].

Once we get the optimal value of f, we are able to compute h via (6), and the expected execution time via (4). We now give an upper bound of CCG's execution time for a rough glance at the protocol performance.

Theorem 6. *An upper bound of CCG's execution time is*

$$T_{upper} = n\times(\frac{e}{96}\times t_{id} + (e-1)\times t_{gid}), \tag{15}$$

where n is the number of tags in G.

Proof. Consider an arbitrary round, in which the number of unlabeled tags is n'. From (5), we have:

$$\lambda(f) = \frac{\psi}{\frac{f}{96}\times t_{id} + h\times t_{gid}} \geq \frac{min(\psi)}{\frac{f}{96}\times t_{id} + max(h)\times t_{gid}}.$$

As previously mentioned, since a singleton slot must be a homogeneous slot, the number ψ of homogeneous tags is no less than that of singleton slots, i.e., $min(\psi) = n'\times e^{-\frac{n'}{f}}$. Besides, the number h of homogeneous slots is no more than that of non-empty slots (homogeneous slots and heterogeneous slots), i.e., $max(h) = f\times(1-e^{-\frac{n'}{f}})$. Let $f = n'$. We have

$$\lambda(f) \geq \frac{e^{-1}n'}{\frac{n'}{96}\times t_{id} + (1-e^{-1})n'\times t_{gid}} = \frac{e^{-1}}{\frac{t_{id}}{96} + (1-e^{-1})t_{gid}}.$$

With n tags in $\mathcal{G}$, the total execution time of CCG is $\frac{n}{\lambda(f)} \leq n\times(\frac{e}{96}\times t_{id} + (e-1)\times t_{gid}) \approx n\times(0.028\times t_{id} + 1.718\times t_{gid})$. □

D. Performance Improvement

Consider the labeling phase. The reader needs to broadcast a group ID in each labeling slot, which is time-consuming when the group ID is long, e.g., 32 bits. In this case, we make a minor modification to the labeling phase, so that the overhead is insensitive to the length of group IDs. The key idea is to transmit the index of each group (instead of the group ID) in the time frame of the labeling phase. That means, for each group P_i, $1 \leq i \leq k$, we just need to transmit the index i rather than g_i. Since there are k groups, $\lceil \log_2 k \rceil$ bits for each index are enough to distinguish each group. For example, only $\lceil \log_2 2 \rceil = 1$ bit is needed for a group in Fig. 5. The reader can respectively broadcast '1' and '0' in the first and second labeling slot to label corresponding tags. After the frame, the reader will transmit the sequence of group IDs in order of their indices.

For further improvement, we concatenate all group indices to form a *labeling vector* of $\lceil \log_2 k \rceil \times h$ bits, and then broadcast it, where h is the number of homogeneous slots. If the vector is too long, we can split it into 96-bit segments and transmit each of them in t_{id} [19]. The tags in the ith labeling slot are thus labeled by the ith index that is from the $((i-1)\times\lceil \log_2 k \rceil + 1)$th bit to $(i\times\lceil \log_2 k \rceil)$th bit in the labeling vector. In this way, t_{gid} in (5) and (15) is equal to $\frac{\lceil \log_2 k \rceil}{96}\times t_{id}$. The upper bound of the total execution time of CCG is $(0.028 + 0.018\times\lceil \log_2 k \rceil)\times t_{id}\times n$.

V. Enhanced CCG

A. Motivation

Although CCG achieves high grouping efficiency by simultaneously labeling tags from different groups, it needs to assign each homogeneous slot a group ID of $\lceil \log_2 k \rceil$ bits. When there is a very large group whose tags pick numerous (homogeneous) slots, the same group ID will be transmitted in all these slots. This high-level redundancy presents opportunity for optimization.

Our idea is to deal with each large group separately. For any given large group P', we will treat the tags in all other unlabeled groups *logically as another group*. The reader applies CCG on these two groups, P' and all other unlabeled tags. In the labeling phase, each slot needs only one-bit index (instead of $\lceil \log_2 k \rceil$ bits): '1' for P' and '0' for all others. After the time frame, the group ID for P' is transmitted, which will be received by all tags in P' that have been labeled with index '1'.

The above modified CCG for labeling one large group is denoted as CCG$_1$. Note that there may be multiple large groups in the system, and each of them requires a separate execution of CCG$_1$. In the following, we design an enhanced version of CCG (denoted as E-CCG), which first invokes CCG$_1$ to handle large groups one at a time and then invokes CCG (as described in the previous section) for the remaining small groups.

B. Protocol Description

The basic idea of E-CCG is to separately label the large groups and then concurrently label all remaining groups. Consider the initial partition $\mathcal{P} = \{P_1, P_2, ..., P_k\}$. Without loss of generality, we assume that $|P_1| \geq |P_2| \geq ... \geq |P_k|$. Algorithm 2 gives the pseudo code of the protocol. In line 2, we compare the execution time of using CCG to label all remaining groups with that of first using CCG$_1$ to label P_i, $1 \leq i \leq k$, and then using CCG to label the rest concurrently. The term $T_c(\cdot)$ is the execution time of using CCG to group tags according to a given partition. This time can be predicted by the reader because it has all tag IDs and precomputes all system parameters. More specifically, CCG consists of multiple rounds. The reader can determine which tags will be

Algorithm 2: Enhanced CCG Protocol

```
Input: $\mathcal{P}$: $\{P_1, P_2, ..., P_k\}$, where $|P_i| \geq |P_j|, 1 \leq i \leq j \leq k$
1: for $(i = 1;\ i \leq k;\ i{+}{+})$ do
2:     if $T_c(\mathcal{P}) < T_1(P_i, \mathcal{P}) + T_c(\mathcal{P} - \{P_i\})$ then
3:         break;
4:     end if
5:     CCG$_1$($P_i, \mathcal{P}$); /* separate $P_i$ from $\mathcal{P}$ using CCG$_1$ */
6:     $\mathcal{P} = \mathcal{P} - \{P_i\}$;
7: end for
8: CCG($\mathcal{P}$); /* group all tags left using CCG */
```

labeled at each round and the per-round execution time is given by (4). $T_1(P_i, \mathcal{P})$ is the execution time of using CCG$_1$ to label tags in P_i from a partition $\mathcal{P}$, which can also be predicted. Lines 2-6 state that if applying CCG$_1$ takes less time, then we will use it to label the next large group; otherwise we use CCG to label all remaining groups concurrently.

C. Performance Analysis

The execution of E-CCG consists of CCG$_1$ for the large groups and CCG for other groups. The performance of using CCG to group all unlabeled tags has been discussed in Section IV-C. Here we will focus on the performance analysis of CCG$_1$. Consider the current set $\mathcal{G}'$ of unlabeled tags and a group P' to be labeled next. The partition is logically treated as $\{P', \mathcal{G}' - P'\}$. Let $n' = |\mathcal{G}'|$ and $m' = |P'|$. According to CCG$_1$, there may be several grouping rounds to separate P' from $\mathcal{G}'$. For the ith round, we let n'_i be the number of unlabeled tags in $\mathcal{G}'$ and m'_i be the number of unlabeled tags in P' before this round. Initially, $n'_1 = n'$ and $m'_1 = m'$. The grouping time t_i and the grouping efficiency λ_i in this round are

$$\begin{aligned} t_i &= \frac{f_i + h_i}{96} \times t_{id} \\ \lambda_i &= \frac{\psi_i}{t_i}, \end{aligned} \tag{16}$$

where ψ_i is the expected number of homogeneous tags in this round, f_i is the length of the virtual frame in the ordering phase, and h_i is the expected number of homogeneous slots in the labeling phase (which will not be actually carried out as we have explained in Section V-A). Note that $f_i + h_i$ is the total number of bits transmitted by the reader in this round. From (6), the expected number h_i of homogeneous slots in this round is

$$h_i = f_i \times ((1 - \frac{1}{f_i})^{m'_i} + (1 - \frac{1}{f_i})^{n'_i - m'_i} - 2(1 - \frac{1}{f_i})^{n'_i}). \tag{17}$$

From (7), the expected number ψ_i of homogeneous tags in this round is

$$\psi_i = m'_i \times (1 - \frac{1}{f_i})^{n'_i - m'_i} + (n'_i - m'_i) \times (1 - \frac{1}{f_i})^{m'_i}. \tag{18}$$

Substituting h_i and ψ_i in (16) with (17) and (18), we derive the grouping efficiency λ_i in this round:

$$\begin{aligned} \lambda_i &= \frac{\frac{m'_i}{f_i}(1 - \frac{1}{f_i})^{n'_i - m'_i} + \frac{n'_i - m'_i}{f_i}(1 - \frac{1}{f_i})^{m'_i}}{\frac{t_{id}}{96}(1 + (1 - \frac{1}{f_i})^{m'_i} + (1 - \frac{1}{f_i})^{n'_i - m'_i} - 2(1 - \frac{1}{f_i})^{n'_i})} \\ &\approx \frac{96}{t_{id}} \times \frac{\frac{m'_i}{f_i} e^{-\frac{n'_i - m'_i}{f_i}} + \frac{n'_i - m'_i}{f_i} e^{-\frac{m'_i}{f_i}}}{1 + e^{-\frac{m'_i}{f_i}} + e^{-\frac{n'_i - m'_i}{f_i}} - 2e^{-\frac{n'_i}{f_i}}}. \end{aligned} \tag{19}$$

Once m'_i and n'_i are given, we can derive the optimal f_i that maximizes λ_i by computing $\frac{d\lambda_i(f_i)}{df_i} = 0$. Besides, based on Theorem 5, we can also derive the interval $[1, 3.8n'_i]$ in which the optimal f_i resides by letting $\beta = 1$. Enumerating all possible candidates in this interval can easily find out the optimal f_i.

VI. Multiple Grouping

So far, we have discussed the protocol design with each tag belonging to exactly one category. In practice, however, a tagged product may belong to multiple groups due to different classification criteria. For example, milk is both food and liquid produce. In this section, we discuss the *multiple grouping* problem that assigns each tag one or more group IDs as needed. Formally, let $\mathcal{C} = \{C_1, C_2, ..., C_k\}$ be the set of all groups, where k is the number of categories. Each C_i, $1 \leq i \leq k$, represents a category as well as the tag set belonging to this category. A tag in $\mathcal{G}$ may reside in one or more groups. Hence, the intersection of C_i and C_j, $1 \leq i < j \leq k$, may be non-empty. The multiple grouping problem is to inform all tags to which group(s) they belong. By contrast, we refer to the previous case where each tag belongs to exactly one group as *single grouping*.

A. Multiple Grouping With TPG

TPG in multiple grouping consists of n grouping rounds, where n is the number of tags. For an arbitrary tag, the reader first separates it from others by broadcasting its tag ID and then transmits the corresponding group IDs to label it. Unlike single grouping, the tag may belong to one or more groups in multiple grouping. To reduce the communication overhead, we concatenate all group IDs together to form a bit string and broadcast it in one transmission. The tag matching the tag ID can decode its group IDs by segmenting the bit string according to the length of group IDs.

B. Multiple Grouping With EPG

Compared with TPG, EPG in multiple grouping aims to reduce the redundant transmissions of group IDs. EPG partitions the entire tag set $\mathcal{G}$ into disjoint subsets according to the group set $\mathcal{C}$. Any tags are classified into the same subset only when they belong to the same groups (including the number of groups and what exactly the groups are). After that, EPG labels each subset in sequence. Namely, the reader first transmits the IDs of tags in a subset and then broadcasts all group IDs that these tags belong to. All unlabeled tags keep listening, and those that receive their IDs will be grouped by the follow-up group IDs. This process repeats until all tags are grouped.

C. Multiple Grouping With FIG

FIG labels one group at a time. Consider the ith grouping round for labeling C_i, $1 \leq i \leq k$. The reader takes C_i as the input to encode a Bloom filter $BF(C_i)$ and then broadcasts it. Upon receiving and checking the filter, all tags in C_i are correctly marked but some tags outside C_i may also be marked due to false positives. Knowing all tag IDs, the reader is able to predict the set W_i of mistakenly marked tags, and unmark them in the polling phase. The remaining marked tags will be labeled by the reader in the labeling phase.

The difference is that for single grouping, the labeled tags will stay silent in subsequent rounds, but in case of multiple grouping, all tags must be active throughout protocol execution, regardless of whether they have been labeled. Moreover, when predicting false positives, the reader should consider all tags outside C_i, i.e., $\mathcal{G} - C_i$, in case of multiple grouping, in contrast to $\mathcal{G} - \bigcup_{j=1}^{i} C_j$ in single grouping. Besides, we do not need to consider the issue of grouping order (Algorithm 1) since all groups participate in each round.

D. Multiple Grouping With CCG

CCG simultaneously labels tags from different groups. It may have multiple grouping rounds. Consider an arbitrary round. In the ordering phase, an ordering vector V is broadcast to tell each tag whether the slot that it picks is homogeneous and what the index of its homogeneous slot is in the actual frame to be carried out. For multiple grouping, a significant difference is that a slot is homogeneous if and only if the tag(s) in this slot belong to exactly the same set of groups (as each tag may now belong to more than one group). In case that tags of a homogeneous slot belong to multiple groups, the corresponding slot in the labeling phase will need to carry more than one group ID (or index).

In the labeling phase, the reader broadcasts the labeling vector to label the tags. The number of group IDs (or indices) may be different in each labeling slot. To address this issue, we add a 1-bit flag preceding each group ID to be broadcast, where flag '1' signals the start of a new slot and flag '0' means that the slot has more group ID(s), as shown in Fig. 8. Assume a group ID is l_v bits long. With this flag, each group ID in a slot requires $(l_v + 1)$ bits. Each time after receiving $(l_v + 1)$ bits for a group ID from the reader, each tag checks the first bit to see whether a new slot starts or not. By keeping track of the number of '1's, the tag knows the number of slots and will learn its own group ID(s) upon the arrival of its slot.

VII. Evaluation

In this section, we evaluate the performance of EPG, FIG, CCG, and E-CCG via simulations. Since there is no prior work studying the grouping problem in RFID systems, we compare the execution time of our protocols with the baseline protocol TPG.

A. Simulation Setting

We randomly generate the group size of each group according to the normal distribution $N(\mu, \sigma^2)$, where μ is the mean

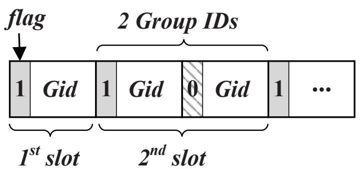


Fig. 8: Illustration for the labeling vector with flags, where each slot may carry more than one group IDs (or indices). Flag '1' signals the start of a new slot, and '0' means that at least one more group ID (or index) will flow in the slot.

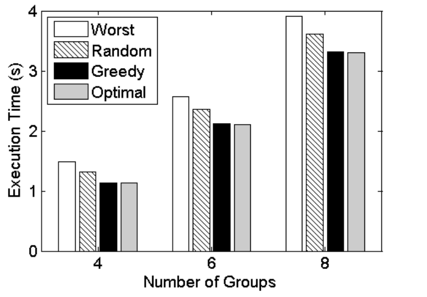
and σ is the standard variance. Our simulations will vary the values of μ, σ and k to show the performance of the protocols under different scenarios, where k is the number of groups.

The communication parameter setting follows the specification of the EPC C1G2 standard [22]. Any two consecutive communications, from the reader to tags or vice versa, are separated by a time interval of 302 μs. The data rate from the reader to tags is 26.7 kbps to 128 kbps. We set the data rate to 26.7 kbps. Note that other rates will change the absolute execution time of the protocols but there will be similar conclusions on relative comparison. It takes the reader 37.45 μs to transmit one bit. Hence, $t_{id} = (37.45 \times 96 + 302) = 3897.2$ μs. The group ID g_i in our simulation is set to the corresponding index i, $1 \leq i \leq k$, which is $\lceil \log_2 k \rceil$ bits long. We also have $t_{gid} = (37.45 \times \lceil \log_2 k \rceil + 302)$ μs. All presented results are the average of 200 independent trials.

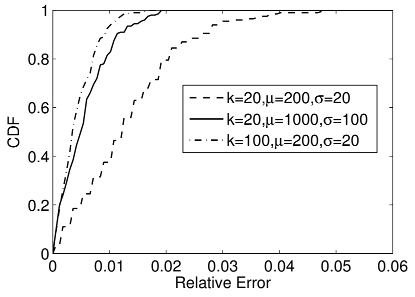
B. Confirmation of Theoretical Results

1) The Greedy Group Ordering Scheme: As discussed in Section III, different group sequences lead to different execution time of FIG. We compare the execution time of the optimal, greedy, random, and worst group sequences under three different scenarios with different number of groups. In the three scenarios, the group size follows $N(1000, 1000^2)$, and the number k of groups are 4, 6, and 8, respectively. For each scenario, we arbitrarily generate a group sequence as the *random sequence*. We then produce the *greedy group sequence* using Algorithm 1. For the optimal and worst group sequences, we exhaustively try out all $k!$ possible group sequences and find the minimal execution time and maximal execution time. The results are shown in Fig. 7(a). We can see that the optimal group sequence performs the best, the greedy sequence follows, then the random sequence, and finally the worst sequence. The negligible difference between the greedy group sequence and the optimal group sequence suggests that our greedy group ordering scheme can determine a nearly optimal group sequence.

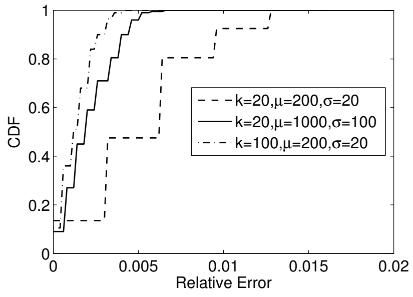
2) Verification of Execution Time: In Fig. 7(b) and Fig. 7(c), we conduct simulations to verify the correctness of the derived execution-time formulas for FIG and CCG respectively under three scenarios. In the first scenario, the number k of groups is 20, the average group size μ is 200, and the standard variance σ is 20. In the second scenario, we keep the same value for k, but change the distribution of group sizes to $\mu = 1000$ and $\sigma = 100$. In the third scenario, we keep the same values for μ and σ, but change k to 100. We perform 200



(a) Performance of FIG under different group sequences



(b) Execution time of FIG



(c) Execution time of CCG

Fig. 7: Confirmation of theoretical results through simulations for (1) the effectiveness of the greedy group ordering scheme and (2) the execution-time formulas.

independent simulation runs in each scenario and plot the CDF of relative error. The relative error is computed as $\frac{|t_{sim}-t_{theo}|}{t_{theo}}$, where t_{sim} is the simulation time and t_{theo} is the theoretical time. In Fig. 7(b), we observe that the relative error of FIG is less than 0.05 in the first scenario. With the increase of the group size (scenario 2) or the number of groups (scenario 3), the relative error further decreases. In Fig. 7(c), the relative error of CCG is within 0.015, much smaller than FIG in the first scenario. The same conclusions can also be drawn for the second and third scenarios. The tightness between the simulation value and theoretical value demonstrates that the derived execution time can well depict the actual execution time of FIG and CCG.

C. Protocol Performance

We evaluate the performance of our protocols under various parameter settings. We compare the execution time of EPG, FIG, and CCG with the baseline protocol TPG under three scenarios. In scenario 1, we set $k = 50$, $\mu = 100$, and $\sigma = 40$. In scenario 2, we double the number k of groups, i.e., $k = 100$, $\mu = 100$, and $\sigma = 40$. In scenario 3, we increase the group size of each group, i.e., $k = 100$, $\mu = 200$, and $\sigma = 80$. The execution time of the protocols are presented in Fig. 9(a). Take scenario 2 as example. The total number n of tags is 10,000. The figure shows that the execution time of TPG is 44.6s, which is the largest amongst the four protocols. EPG reduces the execution time to 39.1s since it avoids transmitting redundant group IDs. FIG further reduces the execution time by 83% to 7.4s. CCG works best, taking only 3.9s, no more than one eleventh of the time needed by TPG. Similar conclusions can also be drawn from the other two scenarios: CCG performs best, FIG next, then EPG, and finally TPG.

Next we study in greater details the impact of different parameters, including the number k of groups, the average group size μ, and the standard variance σ. In Fig. 9(b), we show how k influences the execution time of EPG, FIG, and CCG. We set $n = 2^{14} = 16,384$, $\sigma = 0$, and $\mu = \frac{n}{k}$, where n is the number of all tags. We vary k from 2 to 1,024 and observe the execution time of each protocol. The execution time of EPG remains stable since the transmission overhead for sending group IDs is negligible compared with broadcasting n tag IDs. In contrast, FIG and CCG see a logarithmic growth in their execution time over k. In FIG, the number of grouping rounds increases as k increases, leading to more execution time. In CCG, the reader needs to transmit $\lceil \log_2 k \rceil$ bits in a homogeneous slot in the labeling phase. More transmission bits thus consume longer execution time. Although both FIG and CCG experience the rise trend over k, CCG increases more slowly than FIG. It demonstrates that CCG is less sensitive to the number k of groups than FIG.

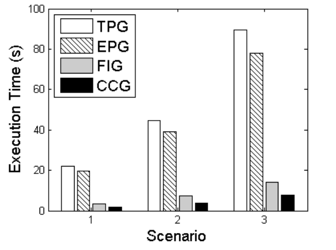
In Fig. 9(c), we study the impact of the average group size μ on the execution time of EPG, FIG, and CCG. We set $k = 100$, $\sigma = 0$, and vary μ from 10 to 100. We observe that the execution time of EPG rises quickly with respect to μ and it approaches to 40s when $\mu = 100$. By contrast, CCG spends the minimal execution time under various μ. It takes less than 4s to achieve the same grouping task, producing a 10-fold performance gain. Although FIG is far superior than EPG, it takes longer time than CCG to achieve the same grouping task. We thus conclude that the execution time of EPG, FIG, and CCG increases with μ and CCG is the most insensitive to μ. It is worth mentioning that, in this simulation, the total number n of tags is equal to $k \times \mu$ and we can thus assert that n has the similar impact on the execution time.

In Fig. 9(d), we evaluate the impact of the standard variance σ in group size on the execution time of EPG, FIG and CCG. In this simulation, we set $k = 10$, $\mu = 1,000$, and vary σ from 100 to 800. The execution time of all protocols largely stay the same under different values of σ. Although the variance varies in the simulation, the total number of tags as well as the number of groups stay the same.

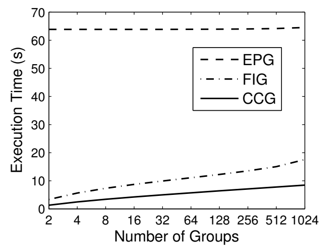
Hence, we conclude that the average group size μ and the total number n of tags have most significant impact on the protocol execution time, the number k of groups has modest impact, and the standard variance σ has little impact.

D. Performance of E-CCG

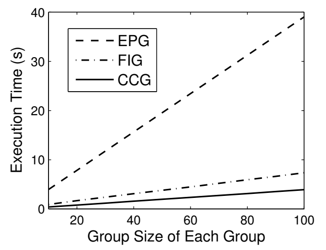
Fig. 10 presents the performance comparison of FIG, CCG, and E-CCG. In this simulation, we set $k = 101$. The group size of the first group P_1 ranges from 50 to 5,000 and the sizes of the remaining 100 groups follow $N(50, 0)$. Fig. 10(a) shows that (1) both E-CCG and CCG outperform FIG, and (2) when the size of P_1 is large, E-CCG significantly outperforms CCG; for example, when the size of P_1 is 5000, E-CCG achieves more than 40% reduction in execution time over CCG. The



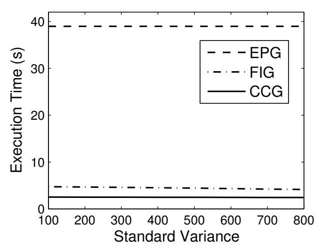
(a) Execution time of different protocols



(b) Execution time with respect to the number of groups

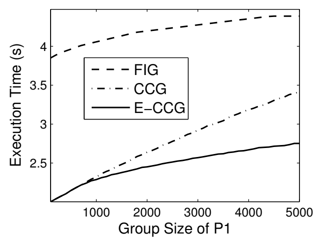


(c) Execution time with respect to group sizes

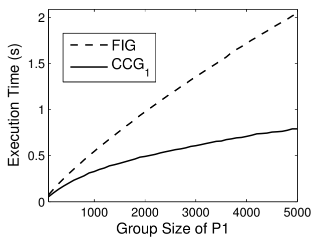


(d) Execution time with respect to standard variances

Fig. 9: Performance comparison.



(a)



(b)

Fig. 10: Grouping overhead of FIG, CCG, and E-CCG. (a) Total execution time to label all tags, (b) Execution time to label tags in group P_1.

performance gap diminishes as the size of P_1 decreases. When P_1 is 700 or fewer, E-CCG becomes the same as CCG and consequently its execution time is also the same as that of CCG. Recall from Lines 2-5 in Algorithm 2 that CCG$_1$ is adopted only when the group size is sufficiently large such that it will save time by using an added round to separate the group from all other tags. However, when the group size is small and CCG$_1$ is not applied, E-CCG simply calls CCG for grouping.

When designing E-CCG, we need to separate the tags in a large group (e.g., P_1) from all other tags. In our design, we have used CCG$_1$ which is modified from CCG. Alternatively we may also use FIG for this purpose. Fig. 10(b) compares these two choices. As we can see, FIG takes significantly more time to label tags in P_1 than CCG$_1$, which validates our choice of CCG$_1$ in the design of E-CCG.

E. Performance of Multiple Grouping

Multiple grouping assigns each tag one or more group IDs. In this subsection, we evaluate the performance of our multiple grouping protocols. We set $k = 50$ and $n = 10,000$. There are three sets of simulations, where each tag is randomly classified into up to 2, 3, and 4 groups, respectively. Fig. 11 compares the versions of TPG, EPG, FIG, and CCG that are modified for multiple grouping in Section VI. The results show that CCG considerably outperforms FIG in terms of execution time and FIG in turn outperforms EPG that outperforms TPG. For example, when each tag belongs to no more than 2 groups, TPG takes 45.4s to perform the task of multiple grouping. In comparison, EPG reduces the time to 39.8s, due to the less transmissions of group IDs. FIG further reduces the time to 10.6s, about one fifth that of TPG, thanks to its use of

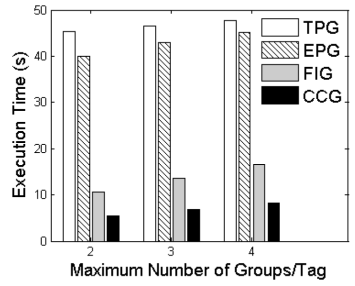


Fig. 11: Performance of multiple grouping.

filter for marking tags in each group and thus avoiding the transmissions of most tag IDs. CCG works best and takes only 5.4s, less than one eighth of the time needed by TPG. Similar conclusions can also be drawn from the other two cases with each tag belonging to no more than 3 and 4 groups, respectively: CCG performs best, FIG follows, then EPG, and TPG performs worst.

VIII. Related Work

In RFID systems, much prior work concentrates on the tag identification problem that collects IDs from all tags in an interrogation zone [29], [30], [33], [34]. Due to limited on-chip resources, tags cannot self-regulate their radio transmissions. The main design principle of tag identification is thus to remove tag-to-tag collisions in open wireless channels. Existing tag identification protocols fall into two categories: Aloha-based protocols [29], [33] and tree-based protocols [30], [34]. The key idea of Aloha-based protocols is to let tags transmit data in different slots. Each tag randomly selects a slot and only the slots chosen by exactly one tag can be used to collect ID information, effectively avoiding tag-to-tag collisions. The tree-based protocols apply a dynamic ID prefix of tag IDs to progressively split a tag set into ever smaller subsets until only one tag is left in each subset. This process is iteratively executed until all tags are successfully identified.

In recent years, the research moves on to mining functional RFID data. For example, cardinality estimation [15]–[17] is to count the number of tags in a large-scale RFID system; missing tag identification [26], [27], [35] aims to identify whether and which tags are absent; searching problems [8], [25], [36] try to find a group of interested tags from the existing tag set; unknown tag detection [37]–[40] proposes to quickly detect whether new tags exist or not in the current system. To efficiently tackle these problems, many advanced solutions have been proposed, without collecting all tag IDs. However, little previous work studied the grouping problem. Traditional polling [41], as a widely used anti-collision technique, provides a request-response way to interrogate RFID tags. It can be tailored to the grouping problem (Section II-C), but the performance is poor. Some recent polling protocols [21], [35] use advanced data structures, e.g., Bloom filters, to reduce the transmission of tag IDs. In the polling phases, however, these protocols still adopt the traditional polling. These work targets at some specific applications, such as tag information collection [21] and missing tag identification [35].

IX. Conclusion

This paper investigates a new problem of how to quickly group a large number of tags in RFID systems according to a pre-specified tag partition. This function plays a fundamental role in improving the inventory and management efficiency in various RFID-enabled applications. We present three grouping protocols. The first one is called Enhanced Polling Grouping (EPG) protocol, which avoids repeatedly transmitting the same group ID as the baseline polling protocol does. The second one is called Filter Grouping (FIG) protocol, which uses Bloom filters rather than polling to label tags group by group. Joint optimization together with a greedy group ordering scheme is proposed to minimize the protocol's execution time. The third protocol is called ConCurrent Grouping (CCG) protocol. It simultaneously labels tags of different groups in a single time frame. An enhanced CCG is designed to improve the performance in handling large groups. Simulation results show that CCG performs best, taking about half the execution time of FIG, one tenth the execution time of EPG, and only one eleventh the execution time of the baseline protocol TPG.

Acknowledgment

This work is supported in part by the National Natural Science Foundation of China (No.61272418, 61373181), the National Science and Technology Support Program of China (No.2012BAK26B02), the Future Network Prospective Research Program of Jiangsu Province (No.BY2013095-5-02), the Lianyungang City Science and technology project (Industry Development) (No.CG1420), the Fundamental Research Funds for the Central Universities, HK PolyU G-SB40, US National Science Foundation grant NeTS-1409797, and CNS-1115548. Lijun Chen is the corresponding author.

References

[1] J. Liu, B. Xiao, S. Chen, F. Zhu, and L. Chen, "Fast RFID grouping protocols," in *Proc. of IEEE INFOCOM*, 2015, pp. 1948–1956.
[2] L. Ni, Y. Liu, Y. C. Lau, and A. Patil, "LANDMARC: Indoor location sensing using active RFID," in *Proc. of IEEE PerCom*, 2003, pp. 407–415.
[3] Y. Liu, Z. Yang, X. Wang, and L. Jian, "Location, localization, and localizability," *Journal of Computer Science and Technology*, vol. 25, no. 2, pp. 274–297, 2010.
[4] 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, pp. 237–248.
[5] C.-H. Lee and C.-W. Chung, "RFID data processing in supply chain management using a path encoding scheme," *IEEE Trans. on Knowl. and Data Eng.*, vol. 23, no. 5, pp. 742–758, 2011.
[6] S. Qi, Y. Zheng, M. Li, Y. Liu, and J. Qiu, "Scalable data access control in RFID-enabled supply chain," in *Proc. of IEEE ICNP*, 2014, pp. 71–82.
[7] 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, pp. 163–171.
[8] M. Chen, W. Luo, Z. Mo, S. Chen, and Y. Fang, "An efficient tag search protocol in large-scale RFID systems," in *Proc. of IEEE INFOCOM*, 2013, pp. 899–907.
[9] M. Chen, S. Chen, and Q. Xiao, "Pandaka: A lightweight cipher for RFID systems," in *Proc. of IEEE INFOCOM*, 2014, pp. 172–180.
[10] X. Liu, K. Li, G. Min, Y. Shen, A. Liu, and W. Qu, "A multiple hashing approach to complete identification of missing RFID tags," *IEEE Transactions on Communications*, vol. 62, no. 3, pp. 1046–1057, 2014.
[11] K. Bu, B. Xiao, Q. Xiao, and S. Chen, "Efficient misplaced-tag pinpointing in large RFID systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 11, pp. 2094–2106, 2012.
[12] J. Liu and L. Chen, "Placement of multiple RFID reader antennas to alleviate the negative effect of tag orientation," in *Proc. of IEEE ICPADS*, 2012, pp. 432–439.
[13] L. Xie, Q. Li, C. Wang, X. Chen, and S. Lu, "Exploring the gap between ideal and reality: An experimental study on continuous scanning with mobile reader in RFID systems," *IEEE Transactions on Mobile Computing*, vol. 14, no. 11, pp. 2272–2285, 2015.
[14] L. Xie, H. Han, Q. Li, J. Wu, and S. Lu, "Efficiently collectiing histograms over RFID tags," in *Proc. of IEEE INFOCOM*, 2014, pp. 145–153.
[15] C. Qian, H. Ngan, Y. Liu, and L. Ni, "Cardinality estimation for large-scale RFID systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 9, pp. 1441–1454, 2011.
[16] M. Shahzad and A. X. Liu, "Every bit counts: Fast and scalable RFID estimation," in *Proc. of ACM MobiCom*, 2012, pp. 365–376.
[17] Y. Zheng and M. Li, "Towards more efficient cardinality estimation for large-scale RFID systems," *IEEE/ACM Transactions on Networking*, vol. 22, no. 6, pp. 1886–1896, 2014.
[18] X. Liu, B. Xiao, K. Li, J. Wu, A. Liu, H. Qi, and X. Xie, "RFID cardinality estimation with blocker tags," in *Proc. of IEEE INFOCOM*, 2015, pp. 1679–1687.
[19] S. Chen, M. Zhang, and B. Xiao, "Efficient information collection protocols for sensor-augmented RFID networks," in *Proc. of IEEE INFOCOM*, 2011, pp. 3101–3109.
[20] Y. Zheng and M. Li, "Read bulk data from computational RFIDs," in *Proc. of IEEE INFOCOM*, 2014, pp. 495–503.
[21] Y. Qiao, S. Chen, T. Li, and S. Chen, "Energy-efficient polling protocols in RFID systems," in *Proc. of ACM MobiHoc*, 2011, pp. 25:1–25:9.
[22] *Epcglobal. EPC radio-frequency identity protocols class-1 generation-2 UHF RFID protocol for communications at 860 mhz-960mhz version 1.2.0*, Tech. Rep., 2008.
[23] WISP, "http://wisp.wikispaces.com/."
[24] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, 1970.
[25] Y. Zheng and M. Li, "Fast tag searching protocol for large-scale RFID systems," *IEEE/ACM Transactions on Networking*, vol. 21, no. 3, pp. 924–934, 2013.
[26] ——, "P-MTI: Physical-layer missing tag identification via compressive sensing," in *Proc. of IEEE INFOCOM*, 2013, pp. 917–925.
[27] W. Luo, S. Chen, T. Li, and S. Chen, "Efficient missing tag detection in RFID systems," in *Proc. of IEEE INFOCOM*, 2011, pp. 356–360.
[28] H. Vogt, "Efficient object identification with passive RFID tags," in *Proc. of IEEE PerCom*, 2002, pp. 98–113.
[29] S.-R. Lee, S.-D. Joo, and C.-W. Lee, "An enhanced dynamic framed slotted ALOHA algorithm for RFID tag identification," in *Proc. of MobiQuitous*, 2005, pp. 166–172.
[30] 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, pp. 2571–2575.
[31] R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth, "On the lambert W function," in *ADVANCES IN COMPUTATIONAL MATHEMATICS*, 1996, pp. 329–359.
[32] M. Kodialam and T. Nandagopal, "Fast and reliable estimation schemes in RFID systems," in *Proc. of ACM MobiCom*, 2006, pp. 322–333.
[33] L. Xie, B. Sheng, C. Tan, H. Han, Q. Li, and D. Chen, "Efficient tag identification in mobile RFID systems," in *Proc. of IEEE INFOCOM*, 2010, pp. 1–9.

[34] C. Qian, Y. Liu, R. Ngan, and L. Ni, "ASAP: Scalable collision arbitration for large RFID systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 7, pp. 1277–1288, 2013.

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

[36] X. Liu, B. Xiao, S. Zhang, K. Bu, and A. Chan, "STEP: A time-efficient tag searching protocol in large RFID systems," *IEEE Transactions on Computers*, vol. 64, no. 11, pp. 3265–3277, 2015.

[37] X. Liu, B. Xiao, S. Zhang, and K. Bu, "Unknown tag identification in large RFID systems: An efficient and complete solution," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 6, pp. 1775–1788, 2015.

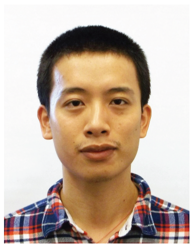
[38] F. Zhu, J. Liu, and L. Chen, "Fast physical-layer unknown tag identification in large-scale RFID systems," in *Proc. of IEEE GLOBECOM*, 2014, pp. 511–516.

[39] X. Liu, K. Li, G. Min, K. Lin, B. Xiao, Y. Shen, and W. Qu, "Efficient unknown tag identification protocols in large-scale RFID systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 12, pp. 3145–3155, 2014.

[40] X. Liu, H. Qi, K. Li, I. Stojmenovic, A. 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.

[41] Y. Qiao, S. Chen, and T. Li, *RFID as an Infrastructure*. Springer New York, 2013.

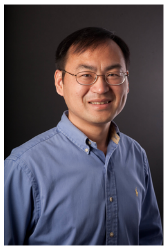
Jia Liu (liujia@smail.nju.edu.cn) received his B.E. degree in Software Engineering from Xidian University, Xi'an, China, in 2010. He is currently a Ph.D. student with the Department of Computer Science and Technology at Nanjing University of China. His research interests include RFID technologies and wireless sensor networks. He is a member of IEEE.



Min Chen (min@cise.ufl.edu) received his B.E. degree in Information Security from the University of Science and Technology of China in 2011. He is currently a Ph.D. student with the Department of Computer and Information Science and Engineering at the University of Florida. His advisor is Dr. Shigang Chen, and his research interests include RFID technologies and network security.

Dr. Bin Xiao (csbxiao@comp.polyu.edu.hk) received the B.Sc and M.Sc degrees in Electronics Engineering from Fudan University, China, and Ph.D. degree in computer science from University of Texas at Dallas, USA. After his Ph.D. graduation, he joined the Department of Computing of the Hong Kong Polytechnic University as an Assistant Professor. Now he is an associate professor and the director of the Mobile Cloud Computing Lab. His research is mainly on mobile cloud computing, smart phone technology, network security, and wireless networks. He is the editor of 3 books and has published more than 100 technical papers in international journals and conferences. Currently, he is the associate editor of the International Journal of Parallel, Emergent and Distributed Systems. He is the IEEE Senior member.

Feng Zhu (zhufeng@smail.nju.edu.cn) received his B.E. degree in Information Security from Harbin Institute of Technology, in 2011. He is currently a Ph.D. student with the Department of Computer Science and Technology at Nanjing University of China. His research interests include RFID technologies. He is a member of IEEE.



Dr. Shigang Chen (sgchen@cise.ufl.edu) is a professor with Department of Computer and Information Science and Engineering at University of Florida. He received his B.S. degree in computer science from University of Science and Technology of China in 1993. He received M.S. and Ph.D. degrees in computer science from University of Illinois at Urbana-Champaign in 1996 and 1999, respectively. After graduation, he had worked with Cisco Systems for three years before joining University of Florida in 2002. He served on the technical advisory board for Protego Networks Inc. in 2002-2003 and as CTO for Chance Media Inc. during 2012-2014. His research interests include computer networks, Internet security, wireless communications, and distributed computing. He published more than 140 peer-reviewed journal/conference papers. He received IEEE Communications Society Best Tutorial Paper Award and NSF CAREER Award. He holds 12 US patents. He is an associate editor for IEEE/ACM Transactions on Networking, and served as editors for a number of other journals. He served in various chair positions or as committee members for numerous conferences. He is a fellow of IEEE.

Dr. Lijun Chen (chenlj@nju.edu.cn) received his B.S. degree in Electrical Engineering from the Xi'an University of Science and Technology, P. R. China, in 1982 and his M.S. degree and Ph.D. degree, both in Automatic Control from China University of Mining and Technology, P. R. China, in 1993 and 1998 respectively. He was a Post Doctoral Follow at the Nanjing University (1998 -2000), P. R. China and the Michigan State University (2001-2002), U.S.A, and a Visiting Scholar at The Hong Kong Polytechnic University in 2007. His current research interests include distributed computing and ubiquitous network.