Fast RFID Grouping Protocols

Jia Liu[†‡], Bin Xiao[‡], Shigang Chen[§], Feng Zhu[†] and Lijun Chen[†]
[†]State Key Laboratory for Novel Software Technology, Nanjing University, China
[‡]Department of Computing, Hong Kong Polytechnic University, China
[§]Department of Computer & Information Science & Engineering, University of Florida, USA

***Abstract*—In RFID systems, the grouping problem 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 the 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. We study fast grouping protocols in large RFID systems. To the best of our knowledge, it is the first attempt to tackle this practically important yet uninvestigated problem. We start with a straightforward solution called the Enhanced Polling Grouping (EPG) protocol. We then propose a time-efficient FIltering 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. 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.**

I. Introduction

Radio Frequency IDentification (RFID) has been widely deployed in a variety of applications for monitoring and tracking tagged objects [1], [2], supply chain management [3], [4], and warehouse inventory control [5]–[7]. Grouping 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 sensor-data collection [8], [9] or cardinality estimation [10]–[12], for tags in the same group as required, dramatically benefiting monitoring functions and inventory management. 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, so that tags in the same subset will have the same group ID.

To further clarify the grouping problem and understand its practical significance, consider an example of 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 drastically improve management efficiency in RFID-enabled applications.

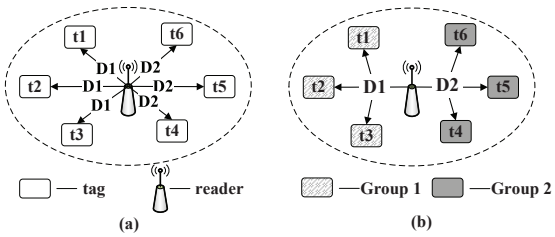


Fig. 1: Multicast transmissions with the grouping scheme

How to group RFID tags efficiently is a new problem not studied before. One intuitive solution is to leverage the traditional polling protocol [13]: 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 performance of grouping. 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 FIltering Grouping (FIG) protocol that uses Bloom filters [14] to avoid transmitting the tag IDs. We address the negative impact of the false positive problem 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

978-1-4799-8381-0/15/$31.00 ©2015 IEEE

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 and k is the number of groups.

We conduct extensive simulations based on the specification of the EPCglobal Gen-2 standard [15]. The simulation results show that to group 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, which improves the grouping efficiency by a factor of 11 when comparing with the traditional polling protocol, and is thus more suitable for real-time RFID-enabled applications.

The rest of the paper is organized as follows. Section II states the grouping problem and shows a straightforward solution. Section III proposes a filtering grouping protocol. Section IV presents a more efficient concurrent grouping protocol. Section V evaluates our protocols. Finally, Section VI concludes this paper.

II. Problem Statement

A. System Model

An RFID system typically consists of one or multiple readers and a large number of tags under coverage. 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 [16], [17]. 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 [8], [18]–[20]. The tag IDs can be automatically collected through one of the numerous existing tag identification protocols [21]–[23].

In a large RFID system, tagged objects may be classified into groups by their properties (e.g., shoes or bags), 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 are able to inform 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 containing n tags. We 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.

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 have the same group ID. More specifically, the reader is instructed by the user with the partition $\mathcal{P}$, and it is supposed to label all tags in the same group P_i with the same group ID 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. However, the state of the art does not address the problem of how to efficiently inform the tags about their individual new group IDs when all tags are now mixed together.

C. A Naive Solution

As aforementioned, 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 an identical group. We now present an Enhanced Polling Grouping (EPG) protocol that avoids transmitting group IDs redundantly.

EPG contains k grouping rounds, where k is the number of groups. In each round, the reader polls all tags in a single group. Consider an arbitrary round for grouping P_i, $1 \le i \le k$. The reader broadcasts IDs of all tags belonging to P_i in turn. Each unlabeled tag keeps listening to the wireless channel, and only the tag receiving its own ID 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 alert 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 [15], and t_{gid} be the length of a slot for transmitting 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 $n\times t_{id} + k\times t_{gid}$, where k is the number of groups.

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. Filtering Grouping Protocol

In this section, we propose an efficient FIltering Grouping (FIG) protocol that avoids most ID broadcasting.

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 [14]. As the reader broadcasts a filter encoding one group to all tags in the system, the tags in the encoded group will be correctly labeled. Some tags in other groups may also be marked mistakenly due to the false positive 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 $P_i' \in P$, $1 \le i \le k$.[1] Let G_i be the set of unlabeled tags at the beginning of this round.

1) Filtering Phase: This phase roughly yet quickly marks all tags in P_i' by leveraging a Bloom filter. The reader first constructs a Bloom filter by hashing the ID of each tag in P_i' to an L_i-bit vector with k_i hash functions, where 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} [8]. Each unlabelred tag in G_i hashes its own ID to k_i bit positions in the filter, and thus knows which segments it needs to record. If all those k_i bits in $BF(P_i')$ are ones, the tag passes the filter and thus transitions to the marked state. Otherwise, the tag remains in the unlabeled state. We denote the set of tags in the marked state as $M_i (\subseteq G_i)$.

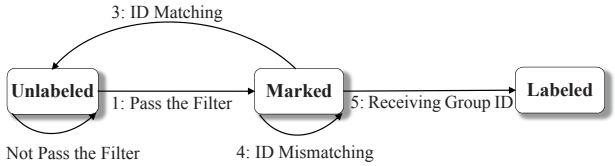


Fig. 3: State diagram of an RFID tag in FIG

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 the number of tags in 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 $\frac{L_i}{96} \times t_{id}$ to transmit an L_i-bit filter[2]. 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

[1]The reason for using P_i' instead of P_i is to show that the order of groups in the rounds does not have to follow the order of groups defined in $\mathcal{P}$.

[2]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 [16].

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 are thus supposed to decrease f_i as much as possible, so as to minimize $T(m'_i, n_i)$.

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

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 optimal k_i.

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

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.

We propose a greedy group ordering scheme that finds a near-optimal group sequence (see Section V-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 $T(m'_i, n_i) \leq T(m'_j, n_i), 1 \leq i \leq j \leq k$, where $m'_1, m'_2, ..., m'_k$ are the tag size for each group. Note that $T(m'_j, n_i)$ denotes the execution time to label the tags belonging to P'_j in the ith round and $T(m'_i, n_i) \leq T(m'_j, n_i)$ depicts 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. It is clear that there are $(k - i + 1)$ unlabeled groups left as the reader has labeled $(i - 1)$ groups in the previous $(i - 1)$ rounds. Suppose that the tag size for 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 with the tag sizes q_1 and q_{k-i+1}.

Proof: Suppose that $x(> 0)$ is a real number that indicates the tag size of an unlabeled group. Let $\frac{dT(x, n_i)}{dx} = 0$, we have the maximum/minimum execution time when

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

where $lambertw(0, x)$ indicates the main branch[3] of Lambert W function [24] at the elements of x. According to (3), we have $x = 0.7369 \times n_i$. 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 $n_i \geq 1$, demonstrating 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 tag 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 EPCglobal Gen-2 standard [15], see Section V-A). We can clearly see that the execution time $T(x, 1000)$ first experiences an increase trend over the tag size x. After reaching the maximum when $x = 0.7369 \times n_i = 736.9$, the execution time sharply decreases with x. Algorithm 1 depicts how to get ordered groups whose tag size are orderly $m'_1, m'_2, ..., m'_k$ according to our greedy scheme. We will show that such ordered group sequence approaches to the optimal one in Section V-B1.

Algorithm 1: Group Ordering Scheme

Input: M: the tag size set for each group in $\mathcal{P}$
Output: M': the tag size set for the ordered groups

```
M' = ∅;
for (i = 1; i ≤ |M|; i++) do
    x⁻ = min(M);
    x⁺ = max(M);
    n_i = sum(M);
    if T(x⁻, n_i) ≤ T(x⁺, n_i) then
        m'_i = x⁻;
    else
        m'_i = x⁺;
    end if
    M = M − {m'_i}; M' = M' ∪ {m'_i};
end for
```

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

[3]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, Lambert W function consists of a set of branches, specifically the main branch is defined for $x \in [-e^{-1}, \infty]$.

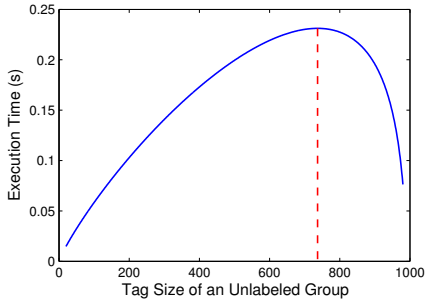


Fig. 4: The execution time with respect to the tag size of an unlabeled group

incorrectly marked tags could be even more 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.

The above dilemma is fundamentally due to the choice of Bloom filters and the one-group-at-a-time strategy in our protocol design. To further improve the performance, we make attempt to explore other radically different ideas without using Bloom filters: labeling tags in all groups together, making some collisions useful so that multiple tags can be labeled together in one slot, and identifying unusable collision 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 designated 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 in the frame, and later an actual frame of the useful slots only will be carried out.

Upon receiving this request, each unlabeled tag randomly picks a slot whose index is $H(id,r) \mod f$, where id is the tag's ID and $H(\cdot)$ is a hash function. Slots picked by no tag, exact 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, neither.

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

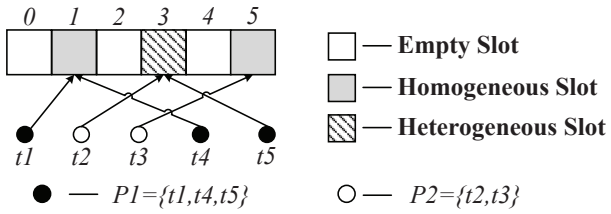


Fig. 5: Three kinds of slots in the ordering phase

The tags picking 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 carrying out the frame to inform the tags of their group IDs. Before doing so, it must inform the tags which slots are removed. For this purpose, the reader broadcasts an *ordering vector* V of f bits [13], 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} [8]. For instance, the ordering vector V for the example in Fig. 5 is '010001'. The actual frame to be carried out contains only two slots.

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 the 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 want to 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:

$$\lambda = \frac{\psi}{t} = \frac{\psi}{\frac{f}{96}\times t_{id} + h\times t_{gid}} \quad (5)$$

Clearly, the bigger the value of λ is, the more the tags will be labeled 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.

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

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

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 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

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

Proof: Given j tags belonging to P_i, the probability that a slot is mapped by only these tags is $(\frac{1}{f})^j\times(1-\frac{1}{f})^{n'-j}$. Because there are $\binom{m'_i}{j}$ kinds of 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 $\sum_{j=0}^{m'_i} j\times\binom{m'_i}{j}\times(\frac{1}{f})^j\times(1-\frac{1}{f})^{m'_i-j}$ that is the expected value of the variable X following 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 k groups, we have the expected number of homogeneous tags in a slot $\sum_{i=1}^{k}\left((1-\frac{1}{f})^{n'-m'_i}\times\frac{m'_i}{f}\right)$. With f slots in the ordering phase, we finally have the expected number of homogeneous tags in this phase, that is $\sum_{i=1}^{k}\left(m'_i\times(1-\frac{1}{f})^{n'-m'_i}\right)$. ■

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

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

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

Theorem 5: When λ attains the maximum value, f must be in the interval $[1,e(n'+1)]$, where e is the natural constant.

Proof: Consider the group efficiency λ as defined by (5). For 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, and h_1 and h_2 be the expected numbers of homogeneous slots. We then have

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

Let $t_{gid}=\beta\times\frac{t_{id}}{96}$ and $\lambda(f_1)-\lambda(f_2)\geq 0$, we have:

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

Clearly, f_2 increases with ψ_2 and h_2, but decreases with ψ_1 and h_1. For a given frame size f_1, the expected number of singleton slots is $n'\times(1-\frac{1}{f_1})^{n'-1}$ [10], [25]. As aforementioned, a singleton slot must be a homogeneous slot, we thus have $h_1\geq n'\times(1-\frac{1}{f_1})^{n'-1}$. Since there is at least one tag in a homogeneous slot, the expected number of homogeneous tags $\psi_1\geq h_1\geq n'\times(1-\frac{1}{f_1})^{n'-1}$. On the other hand, for the frame size f_2, we have $h_2\leq\psi_2\leq n'$ as there are totally n' tags. By substituting above lower bounds and upper bounds, we have $f_2\geq\frac{f_1}{(1-\frac{1}{f_1})^{n'-1}}$, such that $\lambda(f_1)-\lambda(f_2)\geq 0$ always holds. We then derive the minimal $f_2\approx e(n'+1)$ when $f_1=n'+1$. That means, when $f_2\geq e(n'+1)$, there must be a frame size $f_1=n'+1$ ensuring that $\lambda(f_1)-\lambda(f_2)\geq 0$. ■

Based on Theorem 5, we can numerically compute the optimal value of f from the range $[1,e(n'+1)]$ by finding which value maximizes λ in (8). As an example, Fig. 6 shows the grouping efficiency with respect to f, where $t_{id}=3.8ms$, $t_{gid}=0.4ms$ (see Section V-A), the number of groups is 10, and the number n' of unlabeled tags is 1,000, evenly distributed among the groups. The grouping efficiency λ attains its maximum value when f is in [1,2721].

Once we compute the optimal value of f, we are able to compute the expected value of h from (6), and then compute the expected execution time from (4). We cannot give a formula for execution time in closed form. However, we can derive an upper bound below, which demonstrates good performance.

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

$$T_{upper} = n\times(\frac{e}{96}\times t_{id}+(e-1)\times t_{gid}). \quad (9)$$

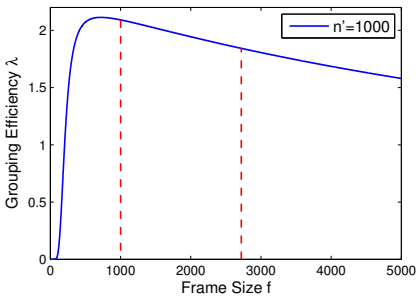


Fig. 6: The grouping efficiency λ with respect to the frame size f

Proof: According to Theorem 5, there must be an optimal $f \in [1, e(n'+1)]$ ensuring that $\lambda(f) \geq \lambda(f_1)$, where $f_1 = n'+1$. Since $\lambda(f_1)$ increases with ψ_1 and decreases with h_1, we have:

$$\lambda(f_1) \geq \frac{min(\psi_1)}{\frac{f_1}{96} \times t_{id} + max(h_1) \times t_{gid}}$$

As previously mentioned, a singleton slot is definitely a homogeneous slot that has one homogeneous tag at least. The number ψ_1 of homogeneous slots is thus no less than the number of singleton slots, that is, $min(\psi_1) = n'(1-\frac{1}{n'+1})^{n'-1} \approx e^{-1}n'$. Meanwhile, the number h_1 of homogeneous slots must be no more than the number of non-empty slots (homogeneous slots and heterogeneous slots). Because the expected number of empty slots is $(n'+1)(1-\frac{1}{n'+1})^{n'} \approx e^{-1} \times n'$, the number of non-empty slots is $(1-e^{-1})n'$ and $max(h_1) = (1-e^{-1})n'$. We thus have

$$\lambda(f_1) \geq \frac{e^{-1}n'}{\frac{n'+1}{96} \times t_{id} + (1-e^{-1})n' \times t_{gid}} \approx \frac{e^{-1}}{\frac{t_{id}}{96} + (1-e^{-1})t_{gid}}$$

With n tags in $\mathcal{G}$, the total execution time is no more than $\frac{n}{\lambda(f_1)} = n \times (\frac{e}{96} \times t_{id} + (e-1) \times t_{gid}) \approx n \times (0.028 \times t_{id} + 1.718 \times t_{gid})$. ■

D. Performance Improvement

Consider the labeling phase. The reader needs to broadcast a group ID in each labeling slot, it is time-consuming when the group ID is too long, e.g., the size of tag IDs. In this case, we make a minor modification for the labeling phase, so that it is insensitive to the length of group IDs. The key idea is to transmit the index of each group instead of the group ID. 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. For further improvement, we orderly concatenate all indexes 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 slip it into 96-bit segments and transmit each of them in t_{id} [8]. The tags in the ith labeling slot are thus labeled by the ith index that is from the $((i-1) \times \lceil \log_2 k \rceil)$th bit to $((i \times \lceil \log_2 k \rceil) - 1)$th bit in the labeling vector. In this way, t_{gid} in (5) and (9) 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. Evaluation

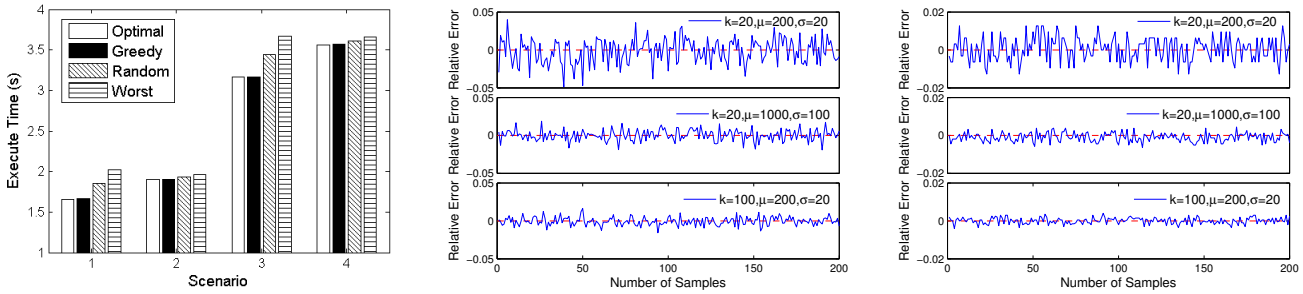
In this section, we evaluate the performance of EPG, FIG, and CCG by simulations. We first verify the efficiency of the greedy group ordering scheme in Algorithm 1 and the derived execution time of FIG and CCG. Since there is no prior work studying the grouping problem in RFID systems, we then compare the execution time of our protocols with the baseline protocol TPG.

A. Simulation Setting

In our simulation, there are totally k groups and we randomly generate the tag size of each group according to the normal distribution $N(\mu, \sigma)$, where μ is the mean and σ is the standard variance. The communication parameter settings follow the specification of EPCglobal Gen-2 standard [15]. 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 with the lower bound 26.7 kbps (the similar conclusion can also be drawn under other parameter configurations). That is, it takes the reader 37.45 μs to transmit one bit. We have $t_{id} = (37.45 \times 96 + 302) = 3897.2$ μs. The group ID g_i in our simulation is set to the index i, $1 \leq i \leq k$, with the length of $\lceil \log_2 k \rceil$ bits. t_{gid} is thus equal to $(37.45 \times \lceil \log_2 k \rceil + 302)$ μs. All presented results are the average of 200 independent trials.

B. Protocol Verification

1) The Greedy Group Ordering Scheme: As discussed in Section III, different group sequences lead to different global execution time of FIG. In Fig. 7(a), we quantify the performance gap between the greedy group sequence in Algorithm 1 and the optimal group sequence. We compare the execution time of the optimal, greedy, random, and worst group sequences under four different scenarios. In scenarios 1 and 2, the number of groups k is 5; the tag size in each group follows $N(1000, 800)$ and $N(1000, 100)$ respectively. In scenarios 3 and 4, the tag size in each group still follows the same normal distribution $N(1000, 800)$ and $N(1000, 100)$, but the group size k is 8. For each scenario, we randomly generate a raw group sequence that is treated as the random sequence. Taking such raw sequence as input, we get the greedy group sequence according to Algorithm 1. For the optimal and worst group sequences, we exclusively traverse $k!$ kinds of group sequences and find out the minimal execution time and maximal execution time. We observe 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. The execution time in the scenarios with small



(a) The efficiency of the greedy scheme of FIG (b) Verification of execution time of FIG (c) Verification of execution time of CCG

Fig. 7: Protocol verification for FIG and CCG

standard variances (scenario 2 and 4) relies little on the group sequence since there are no significant differences among the tag sizes of different groups, diminishing the influence of group sequences.

2) Verification of Execution Time: In Fig. 7(b) and Fig. 7(c), we conduct simulations to verify the correctness of derived execution time of FIG and CCG. There are three kinds of scenarios. In the first scenario, the number of groups k is 20, the average tag size μ is 200, and the standard variance σ is 20. In the second scenario, we fix k, but alter the normal distribution, i.e., $\mu = 1000$ and $\sigma = 100$. In the third scenario, we fix μ and σ, but set $k = 100$. We sample 200 simulation results in each scenario and plot the relative error by computing $\frac{v_{sim}-v_{theo}}{v_{theo}}$, where v_{sim} is the simulated time and v_{theo} is the theoretical time. In Fig. 7(b), we observe that the relative error of FIG mainly vibrates between the interval [-0.05, 0.05] in the first scenario. With the increase of the tag size in each group (scenario 2) and the number of groups (scenario 3), the relative error can further decrease. In Fig. 7(c), the relative error of CCG is within the smaller interval [-0.02, 0.02] in the first scenario. The same conclusions can also be drawn in the second and third scenarios. The tightness between the simulated value and theoretical value demonstrates that the derived execution time can well depict the actual execution time of FIG and CCG.

C. Protocol Performance

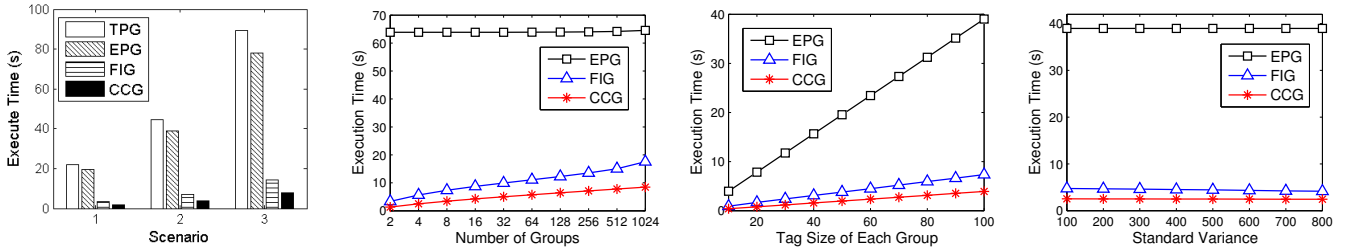
In this subsection, we evaluate the performance of our protocols under various parameter settings. In Fig. 8(a), we compare the execution time of EPG, FIG, and CCG with the baseline protocol TPG under three various scenarios. In scenario 1 we set $k = 50$, $\mu = 100$, and $\sigma = 40$. In scenario 2, we double the number k of groups without changing others, i.e., $k = 100$, $\mu = 100$, and $\sigma = 40$. In scenario 3, we increase the tag size of each group, i.e., $k = 100$, $\mu = 200$, and $\sigma = 80$. We take the scenario 2 as an example to examine the execution time of our protocols, where the number n of tags is 10,000. The execution time of TPG is 44.6s, which is the most time-consuming amongst four protocols. EPG reduces the execution time to 39.1s since it avoids transmitting redundant group IDs. FIG further shortens the execution time by 83% to 7.4s. CCG works best, which takes only 3.9s, no more than one eleventh of the time needed by TPG. Similar conclusions can also be drawn in other two scenarios: CCG performs best, FIG follows, then EPG, and finally TPG.

Note that different scenarios in Fig. 8(a) affect the execution time of our protocols. We now study the impact of various parameters, i.e., the number k of groups, the average tag size μ, and the standard variance σ, on our protocol performance.

In Fig. 8(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. EPG almost remains stable since the transmission overhead for sending group IDs is negligible compared with broadcasting n tedious tag IDs. In contrast, FIG and CCG see a logarithmic growth 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 tread 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. 8(c), we study the impact of the average tag 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 EPG sees a sharp rise over μ and it approaches to 40s when $\mu = 100$. By contrast, CCG spends the minimal execution time under various μ. It takes no more than 4s to achieve the same grouping task, producing a $10\times$ performance gain. Although FIG is far superior than EPG, it takes longer time to achieve the same grouping task in comparison with CCG. 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 thus can assert that n has the similar impact on the execution time.

In Fig. 8(d), we evaluate the standard variance σ with respect to 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. EPG and CCG remain stable because the total number of tags as well as the number of groups stays the same regardless of various σ. FIG experiences a slight decrease over σ. That is because the smaller σ is, the more similar tag sizes are, which makes it more difficult for the reader to



(a) Execution time of different protocols (b) Execution time over various groups (c) Execution time over various tag sizes of each group (d) Execution time over various standard variances

Fig. 8: Performance comparison

sift out tags belonging to the same group. However, with the increase of k, the influence of the standard variance weakens since the gaps between different tag sizes narrow. According to above simulation results in Fig. 8, we say that the average tag size μ or the total number n of tags has the most impact on the execution time, the number k of groups follows, and the standard variance σ influences only FIG's execution time.

VI. Conclusion

This paper investigates a new problem of how to fast group a large number of tags in RFID systems. Grouping tags plays a fundamental role in improving the inventory and management efficiency in RFID-enabled applications. We present three protocols tailored to such grouping problem. The first one called Enhanced Polling Grouping (EPG) protocol avoids repeatedly transmitting the same group ID compared with the Traditional Polling Grouping (TPG) protocol. The second one is called FIltering Grouping (FIG) protocol that uses Bloom filters rather than tedious polling to label tags group by group. The joint optimization together with a greedy group ordering scheme is exploited to minimize the execution time of FIG. We finally propose a more scalable and efficient ConCurrent Grouping (CCG) protocol that avoids the false positive problem intrinsic to Bloom filters and simultaneously labels tags of different groups in a single time frame. Simulation results show that CCG performs best, which takes about half of the execution time of FIG, one tenth of the execution time of EPG, and only one eleventh of the execution time of the baseline protocol TPG.

Acknowledgment

This work is supported in part by the National Natural Science Foundation of China (No. 60873026 and 61272418), the National Science and Technology Support Program of China (No.2012BAK26B02), and the Future Network Prospective Research Program of Jiangsu Province (No.BY2013095-5-02), HK RGC PolyU 5281/13E, and US National Science Foundation grant NeTS-1409797.

References

[1] 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.

[2] J. Han, C. Qian, X. Wang, D. Ma, J. Zhao, P. Zhang, W. Xi, and Z. Jiang, "Twins: Device-free object tracking using passive tags," in *Proc. of IEEE INFOCOM*, 2014, pp. 469–476.

[3] 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. (TKDE)*, vol. 23, no. 5, pp. 742–758, 2011.

[4] J. Liu, B. Xiao, K. Bu, and L. Chen, "Efficient distributed query processing in large RFID-enabled supply chains," in *Proc. of IEEE INFOCOM*, 2014, pp. 163–171.

[5] K. Bu, B. Xiao, Q. Xiao, and S. Chen, "Efficient misplaced-tag pinpointing in large RFID systems," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 23, no. 11, pp. 2094–2106, 2012.

[6] 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.

[7] X. Liu, H. Qi, K. Li, Y. Shen, A. Liu, and W. Qu, "Time- and energy-efficient detection of unknown tags in large-scale RFID systems," in *Proc. of IEEE MASS*, 2013, pp. 95–103.

[8] 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.

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

[10] M. Shahzad and A. X. Liu, "Every bit counts: Fast and scalable RFID estimation," in *Proc. of ACM MobiCom*, 2012, pp. 365–376.

[11] B. Chen, Z. Zhou, and H. Yu, "Understanding RFID counting protocols," in *Proc. of ACM MobiCom*, 2013, pp. 291–302.

[12] 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.

[13] 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.

[14] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[15] *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.

[16] Y. Zheng and M. Li, "Fast tag searching protocol for large-scale RFID systems," in *Proc. of IEEE ICNP*, 2011, pp. 363–372.

[17] 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.

[18] C. C. Tan, B. Sheng, and Q. Li, "How to monitor for missing RFID tags," in *Proc. of IEEE ICDCS*, 2008, pp. 295–302.

[19] 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.

[20] 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 Transactions on Computers (TC)*, vol. 64, no. 1, pp. 87–96, 2015.

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

[22] 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.

[23] 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.

[24] 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.

[25] M. Kodialam and T. Nandagopal, "Fast and reliable estimation schemes in rfid systems," in *Proc. of ACM MobiCom*, 2006, pp. 322–333.