Finding Popular Categories for RFID Tags

Bo Sheng, Chiu C. Tan, Qun Li, Weizhen Mao
Department of Computer Science
College of William and Mary
Williamsburg, VA 23187-8795, USA
{shengbo, cct, liqun, wm}@cs.wm.edu

ABSTRACT

As RFID tags are increasingly attached to everyday items, it quickly becomes impractical to collect data from every tag in order to extract useful information. In this paper, we consider the problem of identifying popular categories of RFID tags out of a large collection of tags, without reading all the tag data. We propose two algorithms based on the idea of group testing, which allows us to efficiently derive popular categories of tags. We evaluate our solutions using both theoretical analysis and simulation.

Categories and Subject Descriptors: C.2.1 [Network Architecture and Design]: Wireless Communication

General Terms: Algorithms, Design, Measurement, Performance, Theory

Keywords: Algorithms, ALOHA, Data Mining, Group Testing, RFID

1. INTRODUCTION

Radio Frequency Identification (RFID) technology is increasingly being deployed for many important applications, such as inventory control and supply chain management. Small RFID tags each containing a tag ID can be attached to products and scanned several meters away via RFID readers, usually in the form of either a portable handheld, or stationary gateway. The ID of each RFID tag specifies information about the item, such as production date and product classification. Manufacturers encode the information by assigning predefined bit positions on the ID [9]. An RFID reader can thus differentiate a jar of peanut butter from a can of beans by reading certain bits denoting the product category.

We envision that low-cost RFID will be attached on every object in our daily life, from clothes, books, pens, to very small objects such as pins and buttons. Annotating objects around us with tags gives us enormous advantage in connecting the physical world with the cyber-world so that people can easily obtain the information about the environment and some interesting applications, such as tracing and tracking physical objects, will be a norm. In particular, combining RFID and sensing technology for reader-activated sensing makes this vision more likely.

We believe that more powerful tags and readers in the future promise many more applications based on how we may use those tags. We may often encounter the scenario, where a reader needs to read a large amount of tags within its range. For example, in a shipping portal or warehouse, the items in pallets and cases will be read together in bulk. In such scenarios, we may wonder how to efficiently extract useful information from that many tags. This paper considers a particular problem of efficiently finding the popular categories among these numerous items. This is important when we want to track the most popular categories shipped in a day, or the least consumed types of goods in a warehouse, or the most frequent values sensed by RFID sensors when the values can be classified into categories.

However, when the collection of tags is large, reading data from every tag to extract information is very time consuming. Furthermore, in many instances, precise information is not required. Instead, the ability to quickly extract information from a large group of tags, even with some errors, is more desired. An example is the earlier research [19], which proposed efficient methods for quickly estimating the number of tags in a collection. In this paper, we aim to solve a more complicated problem, finding popular categories within a large collection of tags. We use the concept of group testing [8] in designing our algorithms. The basic idea behind group testing is that by dividing the categories into groups, we can rapidly eliminate groups that contain many unpopular categories. This allows us to focus on the groups that encompass potential popular categories.

The major contributions of this paper are summarized as follows. (1) This paper considers a complex data mining problem of finding popular categories in RFID systems and we are the first to target at a solution without collecting all tag IDs. This is a technically challenging problem and the solution will benefit many applications with efficiency concerns. (2) We propose a simple fast threshold checking scheme (TCS), which accurately answers whether the number of involved tags exceeds a threshold with high probability. (3) We design two probabilistic algorithms based on group testing and TCS to efficiently find popular categories. The first one is a generic group testing, which randomly places categories into groups. The second algorithm is a combination of group testing and divide-and-conquer. (4) We comprehensively evaluate the proposed schemes and compare them against existing solutions. Our simulation results show that our schemes significantly reduce the total scanning time measured as the number of short slots, which will be explained later.

2. RELATED WORK

For a reader to successfully receive data from multiple tags, anti-collision protocols must be designed so that replied data from multiple tags will not be garbled because of collision. In general, two approaches are used to regulate collision. The first is based on the ALOHA protocol [2, 3, 9, 10, 15, 22, 24, 28–32]. A representative protocol used in RFID systems is the framed ALOHA [24], a variation of ALOHA [1]. In this protocol, a frame is divided into multiple time slots. The communication is initialized when the reader broadcasts a frame size, i.e., the number of slots in the frame. Every RFID tag responds only in a particular slot in the current frame. The reader can successfully receive data in a certain slot if only one tag picks the slot for transmission. This process is repeated until all tag data are collected. The second approach uses the tree traversal technique [4, 5, 16, 21, 25–27, 33]. The reader broadcasts an ID prefix, and those tags whose IDs match the prefix will respond. If a collision is detected, the reader will append '0' or '1' to the prefix and send new prefixes again. It is equivalent to traversing a binary tree, where each tag's ID is a leaf node. The expansion of prefix stops if only one tag responds. The goal of the above anti-collision protocols is to collect all the IDs, which can definitely solve our problem of finding popular categories. However, as we will show in evaluation, they are not efficient. Interestingly, we do use the framed ALOHA and a tree-traversal-like method in the paper, but with a totally different purpose.

In the database community, mining RFID data has drawn considerable attention [13, 14, 17, 23]. Their problems are formulated at a high level, where all RFID data are already stored in a central database. Our paper considers the problem where none of the RFID data has been collected.

Recent research work in [19] is the closest to this paper. The authors consider the problem of estimating the number of tags without collecting the tag IDs. Based on the framed ALOHA, their algorithms analyze the numbers of empty slots, single-reply slots and collision slots to obtain approximated information. By carefully tuning the parameters for multiple iterations, their solutions can quickly estimate the number of RFID tags with high accuracy. [20] uses a similar analysis for anonymous tracking in RFID systems. In this paper, our TCS scheme is based on a similar analysis. However, we consider a more complex problem of finding popular categories. Directly applying the algorithms in [19] cannot efficiently resolve it.

Another relevant research is finding popular items in streaming data [6, 7, 11, 12, 18]. Similar ideas of group testing [8] are adopted in [6] to maintain a small set of counters to find frequent items in data streams, thus achieving memory efficiency. In this paper, our goal is to reduce the scanning time and the assumption of scanning all the data in one pass in the data streaming algorithms is impossible.

3. PROBLEM FORMULATION AND SYSTEM MODEL

We consider that, within the reading range of a reader, there are n products each of which is attached with an RFID tag, that is, n tags $(t_1, \dots, t_n)$ in total. Every RFID tag contains a unique ID represented by a bit string, which consists of several fields [9]. We assume that one of the fields specifies the category the product belongs to. The bit string in the field is called *category ID*. Depending on the applications, a category ID can be as generic as the origin of country, or as specific as a brand and model number. We assume that we know the set of distinct category IDs of the tags considered in this scenario, denoted as $C = \{C_1, \dots, C_m\}$. For each tag t_j, we use c_j to represent its category ID. We will discuss the scenario without knowing C in Section 4.5.

In this paper, popular categories are defined by an application specific threshold. Let F_i be the number of products in category C_i.

Definition 1. *Given a threshold* $\alpha \in (0, 1)$*,* C_i *is a popular category if* $F_i \geq \alpha \cdot n$*.*

Our goal is to find a category set R, which contains popular categories of products in the warehouse. To this end, we are going to design randomized algorithms. This requires us to slightly modify the problem in the randomized setting as follows. Given α, $\beta \leq \alpha$, and $\delta \in (0, 1)$, we would like to **minimize the scanning time** and find a category set R such that with probability larger than $1 - \delta$, the following two accuracy constraints are satisfied:

1. **Completeness Constraint**: $\{C_i | F_i \geq \alpha \cdot n\} \subseteq R$;
2. **Population Constraint**: $\forall C_i \in R, F_i \geq \beta \cdot n$.

We name the first constraint *completeness constraint*, since it requires returning all popular categories. The second constraint is called *population constraint*, as it defines the lower bound of the population of any returned category.

Here we briefly explain the rationale of this problem formulation. Ideally, we would like to return all popular categories, i.e., $\{C_i | F_i \geq \alpha \cdot n\} \subseteq R$, and only the popular categories. However, our randomized setting may return some unpopular categories. To control what extraneous categories may be returned, we introduce another parameter $\beta \leq \alpha$, which defines a lower bound for the population of any returned category. It requires that any $C_i \in R$ must have no fewer than $\beta \cdot n$ products, i.e., $\forall C_i \in R, F_i \geq \beta \cdot n$. A strict requirement may set $\beta = \alpha$. In practice, however, applications usually tolerate a certain level of inaccuracy. For example, it is meaningful to return a category with fewer than $\alpha \cdot n$ products as a popular category. With the requirement of β, the population of each resulting category, although maybe less than $\alpha \cdot n$, is confined to be close to $\alpha \cdot n$. Furthermore, to save scanning time, the number of products in each category is estimated by a probabilistic algorithm. Thus, we can not provide deterministic guarantee for the two constraints. Instead, another parameter $\delta \in (0, 1)$ is defined as a probabilistic guarantee which specifies the maximum allowed probability that our returned results fail to satisfy the two constraints.

In this paper, our schemes often use a 'select' operation: the tags satisfying a certain condition will stay active while the others will keep silent. In a 'select' command, two types of conditions can be specified. First, the reader can broadcast a prefix bit string $mask$ and each tag t_j will check if its category ID matches the received prefix, i.e., if the first $|mask|$ bits of c_j is the same as $mask$, where $|mask|$ is the bit length of $mask$. Second, the reader can broadcast three numbers, r, u, and v, and each tag t_j will check the following condition, $h(r, c_j) \bmod u = v$, where h is a hash function. We use $h_u(r, x)$ to indicate $h(r, x) \bmod u$ in the rest of this paper. In both cases, an RFID tag will keep active only when the specified condition holds.

Our communication model is based on the framed ALOHA. We assume that an RFID reader is able to distinguish the slots with no reply, single reply, or multiple replies. We define these slots as empty slot, single-reply slot, or collision slot respectively. In the typical ALOHA scheme, the duration of a non-empty slot (single-reply or collision) is much longer than that of an empty slot, because tags transfer the whole ID with CRC (Cyclic Redundancy Check) in a non-empty slot. In our approaches, every tag does not transfer the long ID, but a short random bit string (usually < 10 bits [19]), as long as the RFID reader can detect the presence of the signal. Thus, all slots in our approaches have similar durations. In the rest of this paper, we call an empty slot or a slot transferring short bit strings as *short slot*, and a slot transferring IDs as *long slot*. We use S and L to denote the lengths of a short slot and long slot respectively. In addition, our schemes use the algorithm presented in [19] to estimate the total number of active tags. For total n' active tags, the algorithm, denoted as $\Omega(a, b)$ for $a, b \in (0, 1)$, gives an estimation of $\tilde{n}'$ for n', such that with probability larger than a, $1 - \frac{b}{2} \le \frac{\tilde{n}'}{n'} \le 1 + \frac{b}{2}$. Let $|\Omega|$ be the scanning time of Ω. As claimed in [19], $|\Omega|$ is independent of n. Table 1 lists some notations used in the following sections.

$n/\tilde{n}$	number of tags / estimation of n
$n'/\tilde{n}'$	number of active tags/ estimation of n'
C_i/F_i	category ID / number of products in C_i
t_j/c_j	RFID tag / t_j's category ID

Table 1: Summary of Notations

4. FIND THE POPULAR CATEGORIES

We propose and compare different solutions in this section. First, we describe two straightforward, but impractical solutions. Then, we introduce the Threshold Checking Scheme (TCS), which is an important component in our solutions. Finally, we propose our schemes, group testing with TCS and tree traversal with TCS.

4.1 Simple Solutions

The first simple solution is to collect all tag IDs by using the framed ALOHA. Then, we can scan the data and find all popular categories. We call this solution *identification scheme*. In this solution, we have to use long slots to correctly receive the IDs. As analyzed in the prior work [3, 10, 28], the number of slots needed is proportional to the number of tags n. It is inefficient when n is very large.

Alternatively, we can use Ω to resolve the problem. The algorithm is described in Algorithm 1. For each category, the reader broadcasts the category ID so that the tags in the category stay active while the other tags keep silent. Then, we apply Ω to estimate the number of active tags and compare the result with the threshold. Since Ω can obtain a good estimation with a certain setting, Algorithm 1 is able to find all popular categories with a very high probability and the scanning time is $m(L + |\Omega|)$. In practice, this solution is not efficient either, because we may have hundreds

Algorithm 1 Check Each Category

```
1: Run $\Omega$ to obtain $\tilde{n}$
2: for $i = 1$ to $m$ do
3:     Reader broadcasts $C_i$
4:     Tag $t_j$ stays active if $c_j = C_i$
5:     Run $\Omega$ to obtain $\tilde{n}'$
6:     if $\tilde{n}' \ge \alpha \cdot \tilde{n}$ then $R = R \cup \{C_i\}$
7: return $R$
```

of categories (large m) and $|\Omega|$ could be thousands of short slots for a certain accuracy [19]. We will compare these two simple solutions with our solutions in Section 5.

4.2 Threshold Checking Scheme (TCS)

Our algorithms are based on a scheme that estimates whether the number of currently active tags (n') exceeds a given threshold. We call this scheme Threshold Checking Scheme (TCS). The details are presented in Algorithm 2. The input includes a frame size f and other two parame-

Algorithm 2 $TCS(f, \tau_1, \tau_2)$

```
1: Reader broadcasts $f$
2: Each tag randomly picks a time slot to reply
3: Reader obtains $N_0$ and $N_c$
4: if ($N_0 \le \tau_1$) and ($N_c > \tau_2$) then return true
5: else return false
```

ters $\tau_1, \tau_2 \le f$. The reader first broadcasts the frame size f. RFID tags follow the basic framed ALOHA protocol and respond at a random time slot. During this frame, the reader keeps counting the numbers of empty slots and collision slots, recorded in N_0 and N_c respectively. In the end, the reader will compare N_0 and N_c with τ_1 and τ_2 to determine the returned value of TCS. We intentionally avoid using the number of slots for single tag reply (N_1) because N_1 is not a monotonous function of the number of tags. N_0 and N_c, however, are monotonous decreasing and increasing functions of the number of tags respectively. This gives us a simple way to check if n' is greater than the given threshold. Due to the page limit, we omit the detailed analysis here and refer the interested reader to [19].

By carefully choosing f, τ_1, and τ_2, we can have a high confidence that if the number of active tags exceeds a given threshold the protocol returns true. In the following lemmas and theorems, we give the analysis for the protocol assuming there are n' active RFID tags. More specifically, we show the results on $Suc(n')$, which is defined as the probability that $TCS(f, \tau_1, \tau_2)$ returns true when applied to n' active tags. These lemmas and theorems are crucial for the analysis of our algorithms which will be presented later.

LEMMA 1. *When n' and f are large*[1], *N_0 and N_c approximately follow a normal distribution , $N_0 \sim N(\mu_0, \sigma_0)$, and $N_c \sim N(\mu_c, \sigma_c)$, where μ_0, σ_0, μ_c and σ_c are defined in Appendix A.*

PROOF. Refer to [19]. □

THEOREM 1. *When n' and f are large,*

$$Suc(n') = \frac{1}{4}(1 + erf(\frac{\tau_1 - \mu_0}{\sqrt{2}\sigma_0})) \cdot (1 - erf(\frac{\tau_2 - \mu_c}{\sqrt{2}\sigma_c})),$$

[1]We consider general rules of thumb for approximating a binomial distribution to a normal distribution.

where erf is the error function of the standard normal distribution[2]*, and variables μ_0, σ_0, μ_c and σ_c are defined in Appendix A.*

Proof. Based on the properties of normal distributions,

$$Pr(N_0 \le \tau_1) = \Phi(\frac{\tau_1 - \mu_0}{\sigma_0}) = \frac{1}{2}(1 + erf(\frac{\tau_1 - \mu_0}{\sqrt{2}\sigma_0}));$$
$$Pr(N_c > \tau_2) = 1 - \Phi(\frac{\tau_2 - \mu_c}{\sigma_c}) = \frac{1}{2}(1 - erf(\frac{\tau_2 - \mu_c}{\sqrt{2}\sigma_c})).$$

Therefore,

$$Suc(n') = Pr(N_0 \le \tau_1) \cdot Pr(N_c \ge \tau_2)$$
$$= \frac{1}{4}(1 + erf(\frac{\tau_1 - \mu_0}{\sqrt{2}\sigma_0})) \cdot (1 - erf(\frac{\tau_2 - \mu_c}{\sqrt{2}\sigma_c})).$$

□

Theorem 2. *$Suc(n')$ is an increasing function of n', i.e., if $n'_1 \ge n'_2$, $Suc(n'_1) \ge Suc(n'_2)$.*

Proof. Obviously, compared with a group with n'_2 tags, a group with n'_1 tags tends to have less empty slots and more collision slots. □

Theorem 3. *Given a list $\{u_1, \ldots, u_q\}$ and a number $v > 0$, if $\sum u_i = z$, then*

$$\sum Suc(u_i) \le \frac{z}{v} + (q - \frac{z}{v})Suc(v).$$

Proof. We divide the list into two sets, $S_1 = \{i | u_i \ge v\}$ and $S_2 = \{i | u_i < v\}$. Obviously, at most $\frac{z}{v}$ elements belong to S_1. Therefore,

$$\begin{aligned}\sum Suc(u_i) &= \sum_{i \in S_1} Suc(u_i) + \sum_{i \in S_2} Suc(u_i) \\ &\le |S_1| \cdot 1 + (q - |S_1|) \cdot Suc(v) \\ &= |S_1| \cdot (1 - Suc(v)) + q \cdot Suc(v) \\ &\le \frac{z}{v} + (q - \frac{z}{v})Suc(v).\end{aligned}$$

□

4.3 Group Testing with TCS

In this section, we propose a solution based on group testing with TCS. We first divide the tags into groups according to their category IDs. The tags with the same category ID belong to the same group and each group may contain the tags in multiple categories. We then apply TCS to check the number of tags in each group. The intuition is that many categories with few tags may be grouped together and thus can be easily identified as unpopular categories in a simple group test. The groups with sufficient tags are labeled as potential popular groups, which may include popular categories or have no popular categories (when a certain number of unpopular categories contribute adequate number of tags). Our algorithm continues to shuffle all categories into different groups and apply the TCS tests to the new groups again. This process is repeated for a prescribed number of rounds and in the end, the testing history is able to reveal all popular categories.

The details of our protocol are illustrated in Algorithm 3. The whole process consists of T rounds (line 3) and in each round all tags are distributed into W groups by a hash function $h(r, C)$, where r is a random seed and C is a category ID. A tag t_j is in group g if $h_W(r, c_j) = g$ (recall $h_W(r, c_j)$ denotes $h(r, c_j)$ mod W). We use a different random seed to shuffle the categories in each round. Thus, Algorithm 3 totally generates T random seeds, denoted by $\{r_1, r_2, \ldots, r_T\}$. Throughout the algorithm, all tags form $T \times W$ groups, labeled as $G(k, g)$ for $k \in [1, T]$ and $g \in [0, W - 1]$, such that

$$G(k, g) = \{C_i | \; h_W(r_k, C_i) = g\}.$$

Algorithm 3 Group Testing

```
1: Run $\Omega$ to obtain $\tilde{n}$
2: Calculate parameters $T, W, f, \tau_1$, and $\tau_2$
3: for $k = 1$ to $T$ do
4:     for $g = 0$ to $W - 1$ do
5:         Reader broadcasts a random seed $r_k$, $W$, and $g$
6:         Tag $t_j$ stays active if $h_W(r_k, c_j) = g$
7:         $M[k, g] = TCS(f, \tau_1, \tau_2)$
8: for $C_i \in C$ do
9:     check=true
10:    for $k = 1$ to $T$ do
11:        if (not $M[k, h_W(r_k, C_i)]$) then
12:            check=false
13:    if check then
14:        $R = R \cup \{C_i\}$
15: return $R$
```

In the rest of this paper, we use $|G(k, g)|$ to denote the number of the tags whose category IDs belong to $G(k, g)$. In round k, the reader broadcasts r_k, W, and g (line 5) to select the RFID tags mapping to group $G(k, g)$. We then run $TCS(f, \tau_1, \tau_2)$ to examine the number of RFID tags in $G(k, g)$. We record the results in a matrix M: $M[k, g] = true$ means that there might be popular categories in group $G(k, g)$. Otherwise, if $M[k, g] = false$, all the categories in $G(k, g)$ are considered as unpopular categories. Thus, as shown in lines 8-15, a category will be returned, only if the group it belongs to in every round passes the test. Fig. 1 illustrates an example of group testing with 10 categories.

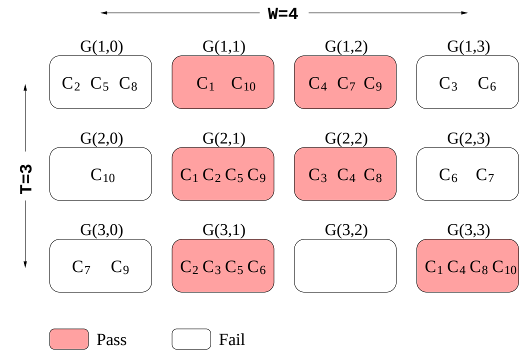

Figure 1: There are 10 category IDs, with parameters $W = 4$ and $T = 3$. Based on the test results, C_1 and C_4 will be returned as popular categories.

In the following, we show how to choose these parameters to minimize the scanning time while the constraints are satisfied. Theorem 4 and Theorem 5 give the conditions that provide the probabilistic guarantee for the completeness constraint and population constraint (stated in Section 3) respectively. Theorem 6 expresses the scanning time by the parameters. Combining them, we can find the optimal

[2]In our implementations, continuity correction is applied.

parameters with the minimum scanning time while satisfying the two constraints.

Specify the constraints: Since TCS is probabilistic and group testing is essentially a randomized algorithm, a popular category may be filtered out of the resulting set and an unpopular category may survive all tests and be present in R. The following two theorems specify the conditions for the parameters to satisfy the accuracy constraints.

THEOREM 4. *The completeness constraint is satisfied with more than $1-\delta$ probability if $(1-\delta\cdot\alpha)\le Suc(\alpha\cdot n)^T$.*

PROOF. Consider a popular category C_i, assume C_i belongs to $G(k,g)$. Let $t=|G(k,g)|\ge F_i\ge\alpha\cdot n$. $G(k,g)$ will pass the TCS test with probability of $Pr(M[k,g]=true)=Suc(t)$. According to Theorem 2,

$$Pr(M[k,g]=true)\ge Suc(\alpha\cdot n).$$

The probability that any of the T groups that C_i belongs to will fail in the TCS test is at most $1-Suc(\alpha\cdot n)^T\le\delta\cdot\alpha$. Based on the definition of a popular category, there are at most $\frac{1}{\alpha}$ popular categories. Thus, by union bound, the probability that no popular category is missing (all popular categories pass all the T tests) is greater than $1-\delta\cdot\alpha\cdot\frac{1}{\alpha}=1-\delta$. $\square$

THEOREM 5. *The population constraint is satisfied with more than $1-\delta$ probability if there exists u, such that*

$$\left(\frac{n-\beta\cdot n}{W(u-\beta\cdot n)}(1-Suc(u))+Suc(u)\right)^T\le\delta.$$

PROOF. We prove the theorem by showing that for any unpopular category C_i, i.e., $F_i<\beta\cdot n$, the probability to be returned in R is less than δ. Assume in a certain round, C_i belongs to a group G and let $t=|G|$. The probability that group G passes a TCS test is $Suc(t)$. For any given u,

$$\begin{aligned} Suc(t) &= Pr(t\ge u)Suc(t)+Pr(t<u)Suc(t)\\ &\le Pr(t\ge u)+(1-Pr(t\ge u))Suc(u). \end{aligned}$$

Let X denote the number of tags in group G which do not belong to category C_i, i.e., $X=t-F_i$. The expectation of X is $E(X)=\frac{n-F_i}{W}$. According to Markov's inequality,

$$\begin{aligned} Pr(t\ge u) &= Pr(X\ge u-F_i)\le\frac{n-F_i}{W(u-F_i)}\\ &= \frac{1}{W}(1+\frac{n-u}{u-F_i})<\frac{1}{W}(1+\frac{n-u}{u-\beta\cdot n}). \end{aligned}$$

Therefore,

$$\begin{aligned} Suc(t) &\le Pr(t\ge u)(1-Suc(u))+Suc(u)\\ &< \frac{n-\beta\cdot n}{W(u-\beta\cdot n)}(1-Suc(u))+Suc(u). \end{aligned}$$

Considering T rounds of tests, C_i will be returned in R with probability of $Suc(t)^T<\delta$. $\square$

Express the scanning time: Here we express the scanning time used in Algorithm 3. In a simple estimation, we need test $T\cdot W$ groups and each test consumes one long slot and f short slots. Thus, in total, Algorithm 3 takes $T\cdot W\cdot(L+f\cdot S)$. We find, however, that it is not necessary to check all groups. In every round, we recognize some unpopular categories, thus the remaining possible popular categories become fewer and fewer. If one group contains only known unpopular categories, we can skip the TCS test for it. We analyze the scanning time in the following series of theorems and lemmas. Theorem 6 bounds the expected scanning time utilizing the result from Lemma 3. Lemma 2 is an auxiliary lemma that helps prove Lemma 3.

LEMMA 2. *Given $a\in(0,1)$, $x<b<n$, and $c\ge1$, $(a+(1-a)\frac{n-x}{W\cdot(b-x)})^c$ is a convex function of x.*

PROOF. See Appendix B. $\square$

LEMMA 3. *Let m_k be the expected number of possible popular categories after the k-th iteration in line 3 of Algorithm 3 and $m_0=m$. Given $u>0$ and $v\le\frac{W\cdot u-n}{W-1}$, then $\forall k\in[1,T]$, m_k is bounded by*

$$\frac{n}{v}+(m-\frac{n}{v})(Suc(u)+(1-Suc(u))\frac{n-1}{W(u-1)})^k.$$

PROOF. For a category C_i, let $p_{i,k}$ be the probability that C_i will still be considered as a possible popular category after the k-th iterations, $m_k=\sum_i p_{i,k}$. Similar to Theorem 5, for any given u,

$$p_{i,k}\le(Suc(u)+(1-Suc(u))\frac{n-F_i}{W(u-F_i)})^k.$$

We divide all categories into two sets, $S_1=\{C_i|F_i>v\}$ and $S_2=\{C_i|F_i\le v\}$. We have,

$$\begin{aligned} \sum p_{i,k} &= \sum_{C_i\in S_1}p_{i,k}+\sum_{C_i\in S_2}p_{i,k}\\ &\le |S_1|+\sum_{C_i\in S_2}(Suc(u)+(1-Suc(u))\frac{n-F_i}{W(u-F_i)})^k. \end{aligned}$$

According to Lemma 2, the right side of the above inequality is a convex function of F_i. To maximize the right hand side, for each category $C_i\in S_2$, F_i takes value of either 1 or v, by the property of a convex function. Suppose $t_1=|\{C_i|F_i=v\}|$ and $t_2=|\{C_i|F_i=1\}|$ when the maximization is achieved. Therefore, $\sum p_{i,k}$ is bounded by

$$\begin{aligned} & |S_1|+t_1\cdot(Suc(u)+(1-Suc(u))\frac{n-v}{W(u-v)})^k\\ + &\ t_2\cdot(Suc(u)+(1-Suc(u))\frac{n-1}{W(u-1)})^k\\ = &\ |S_1|+t_1+t_2\cdot(Suc(u)+(1-Suc(u))\frac{n-1}{W(u-1)})^k. \end{aligned}$$

Let $\lambda=(Suc(u)+(1-Suc(u))\frac{n-1}{W(u-1)})^k\le1$, we have

$$\begin{aligned} \sum p_{i,k} &\le |S_1|+t_1+t_2\cdot\lambda\\ &= |S_1|+t_1+(m-|S_1|+t_1)\cdot\lambda\\ &= m\cdot\lambda+(|S_1|+t_1)\cdot(1-\lambda). \end{aligned}$$

Since the right side of the above inequality is an increasing function of $|S_1|+t_1$ (the number of categories with no less than v tags) and $|S_1|+t_1$ is at most $\frac{n}{v}$, we have

$$\sum p_{i,k}\le\frac{n}{v}+(m-\frac{n}{v})(Suc(u)+(1-Suc(u))\frac{n-1}{W(u-1)})^k.$$

$\square$

THEOREM 6. *The expected scanning time is bounded by*

$$ST = (L + f \cdot S) \cdot W \cdot \sum_{k=1}^{T} (1 - (1 - \frac{1}{W})^{m_{k-1}}), \quad (1)$$

where m_{k-1} is expressed by the bound derived in Lemma 3, replacing k with $k-1$.

PROOF. Let X_k be the number of groups we need check in the k-th iteration. For a certain group, the probability that all the tags in it belong to known unpopular categories is $(1-\frac{1}{W})^{m_{k-1}}$. Thus, the expected value of X_k is $E(X_k) = W(1-(1-\frac{1}{W})^{m_{k-1}})$. Obviously, it is an increasing function of m_{k-1}. Thus, ST bounds the expected scanning time when we express it with the upper bound of m_{k-1}. $\square$

Solve the optimization problem: In summary, given α, β, δ, n and m, our problem is to determine the values of T, W, f, τ_1 and τ_2 in the following optimization problem.

$$\begin{aligned} & \text{minimize } ST \text{ (Eq.(1))} \\ s.t. \quad & (1 - \delta \cdot \alpha) \leq Suc(\alpha \cdot n)^T; \\ & \exists u, (\frac{n - \beta \cdot n}{W(u - \beta \cdot n)}(1 - Suc(u)) + Suc(u))^T \leq \delta. \end{aligned}$$

Since all these parameters are bounded integers, we can find the optimal set of parameters by discretizing them and enumerating all possible values. The process basically includes five loops to enumerate all possible discrete values for the five parameters. We also apply some optimization strategy to speed up the process.

4.4 Tree Traversal

Group testing can be applied differently. In this subsection, we combine group testing with divide-and-conquer. We first divide all tags into W groups based on their category IDs and run TCS for each group, which is the same as the first round in the previous solution. However, in this scheme, we do not shuffle all categories into groups in each of the remaining rounds. We ignore those groups that fail to pass the TCS tests and suppose there are no popular categories in them. Each of the groups which pass the test is further divided into W sub-groups and we apply TCS to each sub-group. This dividing process is repeated recursively until TCS test fails or there is only one category in the group, in which case that category will be returned as a popular category. Fig. 2 illustrates an example.

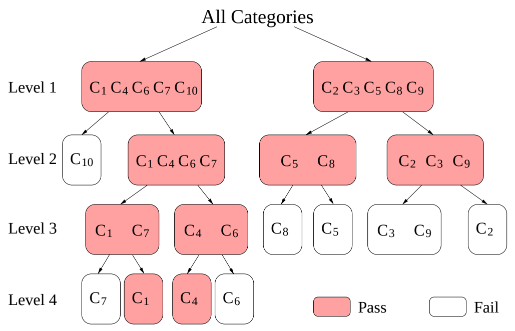

Figure 2: In this example, there are 10 categories with parameter $W = 2$. Based on the test results, C_1 and C_4 will be returned as popular categories.

Conceptually, this scheme is equivalent to a depth-first tree traversal on a W-ary tree, where each leaf is a category and each non-leaf node represents a group of categories that appear as leaves of the subtree rooted at it. Different from the previous scheme, this scheme uses multiple random seeds and group indices to define a group. For example, a node at level 1 (a direct child of the root) is defined by a pair composed of a random seed and a group index as in the previous scheme. However, to select a group represented by a level 2 node, we need first select the tags belonging to its parent node, and then divide them into W sub-groups by another random seed. Thus, we need two pairs of random seeds and group indices to define a level 2 node. Inductively, for a node at level l, the group it represents is defined by l pairs of random seeds and group indices. Thus, we denote a node by a vector of random seeds $\{r_k\}$ and a vector of group indices $\{v_k\}$,

$$\text{Node } (\{r_k\}, \{v_k\}) = \{C_i | \ \forall k, h_W(r_k, C_i) = v_k\}.$$

Since passive RFID tags are memoryless devices, when visiting a node on the tree, the reader has to provide all random seeds and group indices to select the corresponding group. Algorithm 4 presents the details of traversing a node. The first call is to traverse the root (level 0), where both $\{r_k\}$ and $\{v_k\}$ are empty.

Algorithm 4 Traverse Node ($\{r_k\}, \{v_k\}$) at Level l

```
1: for $k = 1$ to $l$ do
2:     Reader broadcasts $W$, $v_k$, and $r_k$
3:     Each tag $t_j$ stays active if $h_W(r_k, c_j) = v_k$
4: if $TCS(f, \tau_1, \tau_2) = true$ then
5:     Reader generates a new random seed $r$
6:     for $v = 0$ to $W - 1$ do
7:         Traverse Node ($\{r_k\} \cup \{r\}, \{v_k\} \cup \{v\}$).
```

Specify the constraints: Similar to the previous subsection, the following Theorem 7 and Theorem 8 give the conditions that guarantee the completeness constraint and population constraint. Lemma 4 is needed by the proof of Theorem 7.

LEMMA 4. *Consider a leaf node at level l. Given $u \geq 1$,*

$$Pr(l \leq u) = (1 - \frac{1}{W^u})^{m-1}.$$

PROOF. For a certain category C_i, the probability that a different category falls in the same group at level l is $\frac{1}{W^l}$. The probability that none of the other $m-1$ categories share the same hashed values is $(1-\frac{1}{W^l})^{m-1}$. $\square$

THEOREM 7. *The completeness constraint is satisfied with more than $1-\delta$ probability if there exists u, such that*

$$1 - (1 - \frac{1}{W^u})^{m-1} Suc(\alpha \cdot n)^u \leq \delta \cdot \alpha.$$

PROOF. Assume a popular category is represented by a leaf node at level l. It must pass l TCS tests to be returned, which has a probability of at least $Suc(\alpha \cdot n)^l$. Given a parameter $u \geq 1$, the probability that a popular category will be returned is more than $Pr(l \leq u) \cdot Suc(\alpha \cdot n)^u$. Applying Lemma 4 and union bound, this theorem can guarantee the accuracy requirement. $\square$

THEOREM 8. *The population constraint is satisfied with more than $1-\delta$ probability if $Suc(\beta \cdot n) < \delta$.*

Proof. Any returned category in this scheme must pass the test as a leaf node, i.e., without tags in any other category in the same group. Therefore, $Suc(\beta \cdot n) < \delta$ guarantees that with more than $1-\delta$ probability, an unpopular category will not pass the test by its own. $\Box$

Express the scanning time: In this tree traversing process, when visiting a node at level l, we need l long slots to transmit the random seeds and group indices which define the node. Then we need f short slots for each TCS test. Theorem 9 bounds the expected scanning time.

Theorem 9. *Given u, the expected scanning time of the tree traversal scheme is bounded by*

$$ST = W \cdot \sum_{l=0}^{\log_W m-1} ((l+1) \cdot L + f \cdot S) \cdot (\frac{n}{u} + (W^l - \frac{n}{u}) Suc(u)). \quad (2)$$

Proof. Assume a node i is at level $l+1$. Let N_i be the number of tags whose category IDs belong to the group represented by i. The probability that i is visited is less than $Suc(N_j)$, where j is i's parent at level l.

Let us consider a balanced W-ary tree, with W^l nodes at level l. The expected number of nodes visited at level $l+1$ is at most $W \cdot \sum_j Suc(N_j)$. According to Theorem 3

$$\sum Suc(N_j) \le \frac{n}{u} + (W^l - \frac{n}{u}) Suc(u).$$

Visiting node i requires $l+1$ long slots for the reader to broadcast random numbers and group indices and f short slots for the TCS test. Thus, considering all levels, the expected scanning time is bounded by ST. $\Box$

Therefore, our goal is to find the optimal parameters to

$$\begin{aligned} & \text{minimize } ST \text{ (Eq.(2))} \\ s.t. \quad & \exists u, 1 - (1 - \frac{1}{W^u})^{m-1} Suc(\alpha \cdot n)^u \le \delta \cdot \alpha; \\ & Suc(\beta \cdot n) < \delta. \end{aligned}$$

Similar to the previous scheme, all the involved parameters are integers and bounded. Thus, we are able to enumerate all possible values and find the optimal parameters.

4.5 Extension

4.5.1 Without Knowledge of C

All previous solutions are based on the assumption that the set of present category IDs is known. In fact, with minor modifications, our schemes are also suitable for the scenario where category IDs are unknown.

Obtain m: In our schemes, m is an important factor in setting other parameters. In this extension, our first step is to use Ω to estimate m. We can let the reader send a random seed r and a frame size f as usual and have each tag t_j respond at slot $h_f(r, c_j)$. In this way, all the tags in a group will reply at the same slot, acting as a single tag. Thus, we can count the number of empty slots and use Ω to estimate the number of distinct categories.

Group Testing: If we use group testing, the analysis of the scanning time will be different. Without the category ID information, we have to exam all $T \cdot W$ group. We can easily find the optimal parameter setting with this modified objective. For each group that passes a TCS test, we need use a simple query tree scheme to find the category IDs in the group. For each category ID, we check the other groups it belongs to. If all of them pass the tests, we return this category as a popular category.

Tree Traversal: We can also use the tree traversal scheme in this extension. Without the category ID information, however, we have to determine if the traversing process reaches leaf nodes. An effective way is to observe the number of empty sub-groups of a node. If all sub-groups but one are empty, then with more than $1 - \frac{1}{W}$ probability, the node is a leaf node. If this scenario has occurred for several times (k times) while we keep dividing the non-empty sub-group, then with probability more than $1 - \frac{1}{W^k}$, the node is a leaf node. With a heuristic value of k, we can confirm a leaf node with high probability in this means. After locating a leaf node, we can easily obtain the category ID by using a prefix mask to query each bit. Assume the category ID is represented by B bits. We can locate it in B slots.

4.5.2 Continuous Monitoring

A unique advantage of group testing method is that it can be used for continuous online popular categories discovery. For example, in a shipping port monitoring system, goods may come through the monitoring gate in bulk and bursty fashion, or in a large warehouse, a reader cannot reach all the tags in stock. In both scenarios, finding the popular categories is different from the case that all tags are within the range of a reader, in which case the tag information can be retrieved any time. Group testing approach can conform to this dynamic environment so that the popular categories can be found by only estimating the number of tags that fall in each of the predetermined number of groups. Our algorithm can be slightly modified to suit this case.

5. PERFORMANCE EVALUATION

We evaluate the performance of our schemes via simulations. By default, we set $n = 10000$, $m = 100$, $\alpha = 0.1$, $\beta = 0.05$, and $\delta = 0.01$. In addition, $|\Omega(a, b)|$ is estimated as 2000 short slots for $a = 0.99$ and $b = 0.05\%$ according to [19], and we assume that the duration of a long slot is 5 times that of a short slot, i.e., $L = 5S$.

We begin by presenting the performance of the simple solutions mentioned in Section 4.1. For the first identification scheme, we conduct 1000 simulations with an initial frame size $f = 10000$. At the end of each frame, the new frame size is set to the number of the tags which have not been collected. With the default setting, the time consumed in our simulations is about 122k short slots on average and the deviation is less than 2k short slots. For the other simple scheme (Algorithm 1), the scanning time is estimated based on $|\Omega| = 2000$. Checking each category needs 2000 short slots to finish Ω. Thus, with the default setting, Algorithm 1 requires $100 \times 2000 = 200$k short slots. These two simple solutions are both very costly, as we will show later when comparing with our schemes.

For the rest of the evaluation, we denote group testing with TCS as GT, and tree traversal with TCS as TT. All results are the averaged results of 1000 independent trials.

5.1 Distribution Models for Data Sets

The performance of our schemes is heavily dependent on the product distribution in all categories. The following distribution models are considered in our evaluation.

- Uniform Distribution: In this distribution, we intentionally introduce some popular categories, and uniformly distribute the remaining tags to the other unpopular categories. We use $UD(k)$ to denote the uniform distribution with exactly k popular categories. For this distribution, each popular category is assigned $\alpha \cdot n$ tags, and other $m-k$ categories have $\frac{n-k\cdot\alpha\cdot n}{m-k}$ tags.

- Max/1 Distribution: We denote this distribution as $M1(X)$, where X is the maximum number of tags in one category. In this distribution, each category has either X tags or only 1 tag. Since the total number of tags is n, there are $\lfloor\frac{n-m}{X-1}\rfloor$ categories with X tags and $m - \lfloor\frac{n-m}{X-1}\rfloor$ categories with 1 tag.

- Zipf Distribution: We also consider the Zipf distribution, which is commonly found in the real world. This distribution, denoted as $ZD(n, Z)$, is specified by two parameters. The first parameter is the total number of tags and the second parameter Z defines the upper bound of the population for each category, i.e., the number of tags in each category ranges from 1 to Z. For each category, the probability of having $i \in [1, Z]$ tags is $\frac{c}{i^\theta}$, where c is the normalization constant and θ characterizes the distribution. In our data set $ZD(n, Z)$, we tune the value of θ such that the total number of tags is n.

5.2 Scanning Time

5.2.1 *Varying Number of Tags*

We first evaluate our schemes by varying the number of RFID tags n. Fig. 3, Fig. 4 and Fig. 5 present the performance of GT and TT under the uniform, Max/1 and Zipf distributions respectively.

We observe that, when n increases, TCS tests in both GT and TT require larger frame sizes. This is because the number of tags involved in TCS test cases for GT and TT increases, i.e., each group in GT and each node in TT contain more tags. It is intuitive that, for TCS to achieve the same accuracy, a test case with more tags requires a larger frame size. If the frame size remains the same, the increased number of tags will overwhelm most slots in the frame with collisions engendering an inaccurate estimation.

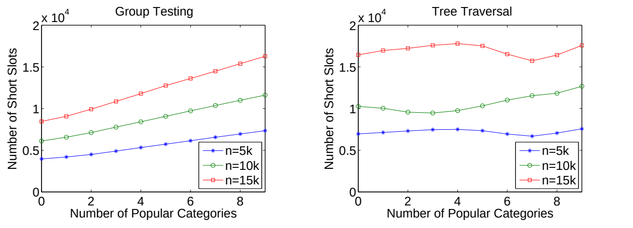

Figure 3: Scanning time for the uniform distribution with varying n

Under the uniform distribution (Fig. 3), the average number of tags in one category (< 15) is far less than the threshold ($\alpha \cdot n = 500, 1000$ and 1500). Both schemes can efficiently identify the groups with popular categories. In the GT scheme, the scanning time is approximately proportional to the number of popular categories. However, the scanning time of the TT scheme does not change much along axis x. In both schemes, a larger n yields more scanning time primarily due to the increase of the frame size in TCS.

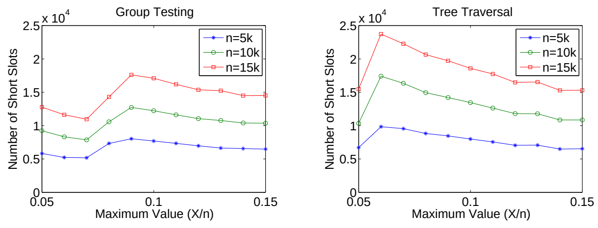

Figure 4: Scanning time for the $M1$ distribution with varying n

For the $M1(X)$ distribution (Fig. 4), we vary the maximum value X from $0.05n$ to $0.15n$. Let us call a category with X tags a *large category*, and a group containing at least 1 large category a *large group*. Basically, a large group has a higher probability to pass the TCS tests. The value of X has two impacts on the performance. On the one hand, the growth of X increases the probability that a large group can pass the TCS tests. The consequence is that we have to apply more TCS tests to eliminate the unpopular categories. On the other hand, when X increases, there are fewer large categories and groups in the protocol, which helps filter out the unpopular categories quickly. In Fig. 4, both schemes are fast at the starting phase, because when X is small, all categories (even large categories) are unpopular and every group has a small probability to pass the TCS tests. Thus, both schemes quickly eliminate all categories and return no popular category. When X grows, the first impact becomes visible, and a sharp increase appears for both schemes, though the peak values are reached at different values of X. We also observe there is a slight decline for GT before the peak value due to the second impact. When X keeps increasing, the second impact becomes dominant and both schemes show a decreasing scanning time after the peak values. For a fixed value of X/n, the scanning time is nearly proportional to n.

Fig. 5 presents the performance under the Zipf distribution. In our data sets, there are usually one or two popular categories. Most categories are unpopular with the number of tags scattered between 1 and $\alpha \cdot n$. Since a considerable number of unpopular categories have tags close to the threshold, our schemes take more time to identify them as unpopular compared to the uniform distribution ($UD(1)$ or $UD(2)$), in which the sizes of the unpopular and popular categories diverge dramatically.

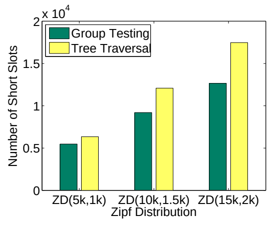

Figure 5: Scanning time for the Zipf distribution with varying n

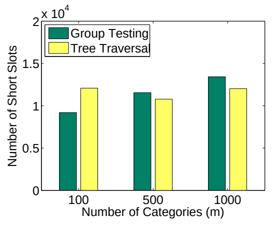

Figure 6: Scanning time for the Zipf distribution with varying m

5.2.2 *Varying Number of Categories*

We also evaluate the performance of the GT and TT schemes with a varying number of categories m. The results are illustrated in Fig. 7, Fig. 8 and Fig. 6.

In Fig. 7 and Fig. 8, we find that with other parameters fixed, the scanning time is increasing when m increases. However, the curves for $m = 500$ and $m = 1000$ are quite close. In Fig. 6, the performance of TT for varying m is almost the same, and the scanning time of GT is slightly increased when m increases.

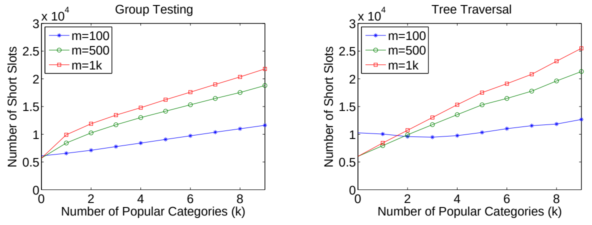

Figure 7: Scanning time for the uniform distribution with varying m

In all three distributions, the number of popular categories in each tested case is primarily determined by other parameters rather than m. Thus, with all other parameters fixed, the case with a larger m has almost the same number of popular categories and more unpopular categories which have to be filtered out. Thus, our schemes need run more TCS tests to identify these unpopular categories. However, unlike n, the impact of m is not proportional to the value of m.

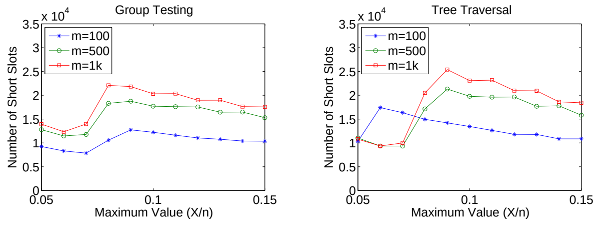

Figure 8: Scanning time for the $M1$ distribution with varying m

5.2.3 *Comparing with Simple Solutions*

Both GT and TT are very efficient in finding the popular categories. Recall that simple solutions in Section 4.1 need at least 122k short slots with our default setting. We use 122k as a baseline to compare with our schemes. In most of the tested cases, the scanning time of our schemes with the default setting is less than 15k short slots, which is about 12% of the baseline. In the scenario that only a few popular categories exist, e.g., $UD(1)$, $UD(2)$, our schemes only require $< 4\%$ of the baseline to finish. We also observe that the group testing scheme is superior to the tree traversal scheme in most cases, especially when the number of tags in some unpopular categories is close to the threshold.

5.3 Tightness of Bounds

Our analysis in Theorem 5 uses Markov inequality, a loose bound that holds for arbitrary random variables. Theorem 5 is further referred in Lemma 3 and Theorem 6 to derive a upper bound of the expected scanning time. Thus, inherently the bound in Theorem 6 is relatively loose for any specific case. To understand how well the theoretical bound matches the reality, we compare our estimated scanning time with the simulation results in this subsection. .

In the default setting, our algorithm estimates that the expected scanning time of the GT scheme is fewer than 14516 short slots. We compare this estimation with the results (mean scanning time) found in our simulations in the following table. For each distribution, we select the worst observed performance. According to the results, our estimation is very close to the actual performance (the worst case is $UD(9)$ with 12734 short slots).

	Our Bound	UD	$M1$	ZD
Number of short slots	14516	12734	11615	9196

5.4 Other Issues

This subsection covers some other issues whose details are omitted due to the page limit:

1. **Accuracy requirements:** In all our simulations, both the completeness constraint and population constraint always hold with more than $1 - \delta$ probability.

2. **Other varying parameters:** When examining the scanning time, we also vary the parameters α and β, and find two basic trends. First, if α and β become closer, our schemes need more time to find popular categories. Second, if we keep their difference constantly, increasing one of them reduces the scanning time.

3. **Compare TCS with Ω:** Group testing can also be combined with algorithm Ω, because Ω obtains more accurate estimation than our TCS test. However, in the tested cases, the frame size for TCS is between 115 to 247 slots, much less than $|\Omega| = 2000$. Based on the results in [6], group testing with Ω will use smaller parameters T and W. The scanning time, however, is still much larger than that in our schemes with TCS.

6. ADDITIONAL DISCUSSION

6.1 Signal Loss

In our algorithms, TCS makes observation based on the numbers of empty/collision slots presented in a frame. In practice, when the link quality is poor, these numbers may be inaccurate due to signal loss, i.e., the reader is not able to detect the signal sent by RFID tags. As a result, we may observe more empty slots and less collision slots.

To resolve this problem, we may use a learning phase to test the link quality between the reader and RFID tags. As long as the signal loss can be characterized by a certain model, we can easily adopt it into our analysis.

6.2 Frame Size

In some RFID standards, frame size cannot be arbitrary, but is constrained to powers of 2, i.e., f can be only set as a power of 2. Our scheme can be easily adopted without any other changes. In our simulation, the frame size is usually less than 256. Thus, the performance with this frame size constraint is similar to the results shown in Section 5.

7. CONCLUSION

In this paper, we consider the problem of efficiently finding popular categories in a large scale RFID system with many categories involved. We design two algorithms based on group testing. Our evaluation shows that group testing can reduce the scanning time for popular category discovery dramatically. We notice that the approach used in this paper can be applied to other interesting RFID estimation problems. For example, our approach can be easily extended to find the popular categories in a different setting with online continuous RFID monitoring. We believe this work gives inspiration for more efficient estimation problems in a system composed of massive RFID tags.

8. ACKNOWLEDGMENTS

We would like to thank all the reviewers for their helpful comments. This project was supported in part by US National Science Foundation grants CCF-0514985, CNS-0721443, and CAREER Award CNS-0747108.

9. REFERENCES

[1] N. Abramson. The ALOHA system - another alternative for computer communications. In *Proceedings of the AFIPS Conference*, volume 37, pages 295–298, 1970.

[2] M. A. Bonuccelli, F. Lonetti, and F. Martelli. Tree slotted ALOHA: a new protocol for tag identification in RFID networks. In *WOWMOM '06*, 2006.

[3] J.-R. Cha and J.-H. Kim. Novel anti-collision algorithms for fast object identification in RFID system. In *ICPADS '05*.

[4] H.-S. Choi, J.-R. Cha, and J.-H. Kim. Fast wireless anti-collision algorithm in ubiquitous id system. In *Vehicular Technology Conference*, pages 4589–4592, 2004.

[5] I. Cidon and M. Sidi. Conflict multiplicity estimation and batch resolution algorithms. *IEEE Trans. Inf. Theor.*, 34(1):101–110, 1988.

[6] G. Cormode and S. Muthukrishnan. What's hot and what's not: tracking most frequent items dynamically. *ACM Trans. Database Syst.*, 30(1):249–278, 2005.

[7] E. D. Demaine, A. Leortiz, and J. I. Munro. Frequency estimation of internet packet streams with limited space. In *ESA '02*, 2002.

[8] D. Du and F. Hwang. *Combinatorial Group Testing and Its Applications*. World Scientific Publishing Company, 1993.

[9] EPCglobal. Class 1 generation 2 UHF air interface protocol standard version 1.0.9.

[10] C. Floerkemeier and M. Wille. Comparison of transmission schemes for framed ALOHA based RFID protocols. In *SAINT-W '06*, 2006.

[11] A. C. Gilbert, S. Guha, P. Indyk, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Fast, small-space algorithms for approximate histogram maintenance. In *STOC*, pages 389–398, 2002.

[12] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. How to summarize the universe: Dynamic maintenance of quantiles. In *VLDB '02*, 2002.

[13] H. Gonzalez, J. Han, and X. Li. Flowcube: constructing RFID flowcubes for multi-dimensional analysis of commodity flows. In *VLDB'2006*, 2006.

[14] H. Gonzalez, J. Han, and X. Li. Mining compressed commodity workflows from massive RFID data sets. In *CIKM '06*, 2006.

[15] P. Hernandez, J. Sandoval, F. Puente, and F. Perez. Mathematical model for a multiread anticollision protocol. In *IEEE Pacific Rim Conference on Communications, Computers and signal Processing*, Aug. 2001.

[16] D. Hush and C. Wood. Analysis of tree algorithms for RFID arbitration. In *ISIT*, 1998.

[17] S. R. Jeffery, M. Garofalakis, and M. J. Franklin. Adaptive cleaning for RFID data streams. In *VLDB'2006*, 2006.

[18] R. M. Karp, S. Shenker, and C. H. Papadimitriou. A simple algorithm for finding frequent elements in streams and bags. *ACM Trans. Database Syst.*, 28(1):51–55, 2003.

[19] M. Kodialam and T. Nandagopal. Fast and reliable estimation schemes in RFID systems. In *MobiCom '06*.

[20] M. Kodialam, T. Nandagopal, and W. C. Lau. Anonymous tracking using RFID tags. In *INFOCOM '07*, 2007.

[21] C. Law, K. Lee, and K.-Y. Siu. Efficient memoryless protocol for tag identification. In *Proceedings of the 1th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 75–84. ACM, August 2000.

[22] S.-R. Lee, S.-D. Joo, and C.-W. Lee. An enhanced dynamic framed slotted ALOHA algorithm for RFID tag identification. In *MOBIQUITOUS '05*, 2005.

[23] Y. Liu, L. Chen, J. Pei, Q. Chen, and Y. Zhao. Mining frequent trajectory patterns for activity monitoring using radio frequency tag arrays. In *PerCom '07*, 2007.

[24] B. Metcalfe. Steady-state analysis of a slotted and controlled ALOHA system with blocking. *SIGCOMM Comput. Commun. Rev.*, 5(1):24–31, 1975.

[25] A. Micic, A. Nayak, D. Simplot-Ryl, and I. Stojmenovic. A hybrid randomized protocol for RFID tag identification. In *IEEE International Workshop on Next Generation Wireless Networks*, 2005.

[26] J. Myung and W. Lee. An adaptive memoryless tag anti-collision protocol for RFID networks. In *IEEE ICC*, 2005.

[27] J. Myung and W. Lee. Adaptive splitting protocols for RFID tag collision arbitration. In *MobiHoc '06*, 2006.

[28] F. C. Schoute. Dynamic frame length ALOHA. *IEEE Transactions on Communications*, 31:565–568, Apr. 1983.

[29] H. Vogt. Efficient Object Identification with Passive RFID Tags. In *International Conference on Pervasive Computing*, LNCS. Springer-Verlag, 2002.

[30] J. Wieselthier, A. Ephremides, and L. Michaels. An exact analysis and performance evaluation of framed ALOHA with capture. *IEEE Transactions on Communications*, pages 125–137, Feb. 1989.

[31] J. Zhai and G.-N. Wang. An anti-collision algorithm using two-functioned estimation for RFID tags. In *ICCSA (4)*, pages 702–711. Springer, 2005.

[32] B. Zhen, M. Kobayashi, and M. Shimizu. Framed ALOHA for multiple RFID objects identification. In *IEICE TRANSACTIONS on Communications*, 2005.

[33] F. Zhou, C. Chen, D. Jin, C. Huang, and H. Min. Evaluating and optimizing power consumption of anti-collision protocols for applications in RFID systems. In *ISLPED '04*, 2004.

APPENDIX

A. VARIABLES IN LEMMA 1

The variables used in Lemma 1 are defined as follows:

$$\begin{aligned}
\mu_0(n', f) &= f \cdot e^{-\frac{n'}{f}}; \\
\sigma_0^2(n', f) &= f \cdot e^{-\frac{n'}{f}}(1 - (1 + \frac{n'}{f})e^{-\frac{n'}{f}}); \\
\mu_c(n', f) &= f(1 - (1 + \frac{n'}{f})e^{-\frac{n'}{f}}); \\
\sigma_c^2(n', f) &= f \cdot e^{-\frac{n'}{f}}((1 + \frac{n'}{f}) \\
&\quad -(1 + \frac{2n'}{f} + (\frac{n'}{f})^2 + (\frac{n'}{f})^3)e^{-\frac{n'}{f}}).
\end{aligned}$$

B. PROOF OF LEMMA 2

Let $g = (a + (1-a)\frac{n-x}{W\cdot(b-x)})^c$. The lemma is proved if the second derivative of g is positive. Let $h = \frac{n-x}{W\cdot(b-x)} > 0$. We have

$$h' = \frac{n-b}{W\cdot(b-x)^2} > 0, h'' = \frac{2(n-b)}{W\cdot(b-x)^3} > 0.$$

The first derivative of g is $g' = c\cdot(1-a)\cdot(a+(1-a)h)^{c-1}h'$, and the second derivative is

$$\begin{aligned}
g'' = c\cdot(1-a)\cdot(((c-1)(1-a)(a+(1-a)h)^{c-2}h')\cdot h' \\
+(a+(1-a)h)^{c-1}h'') > 0.
\end{aligned}$$