An Efficient Protocol for RFID Multigroup Threshold-based Classification

Wen Luo Yan Qiao Shigang Chen
Department of Computer & Information Science & Engineering, University of Florida

***Abstract*—RFID technology has many applications such as object tracking, automatic inventory control, and supply chain management. They can be used to identify individual objects or count the population of each type of objects in a deployment area, no matter whether the objects are passports, retail products, books or even humans. Most existing work adopts a "flat" RFID system model and performs functions of collecting tag IDs, estimating the number of tags, or detecting the missing tags. However, in practice, tags are often attached to objects of different groups, which may represent a different product type in a warehouse, a different book category in a library, etc. An interesting problem, called *multigroup threshold-based classification*, is to determine whether the number of objects in each group is above or below a prescribed threshold value. Solving this problem is important for inventory tracking applications. If the number of groups is very large, it will be inefficient to measure the groups one at a time. The best existing solution for multigroup threshold-based classification is based on generic group testing, whose design is however geared towards detecting a small number of populous groups. Its performance degrades quickly when the number of groups above the threshold become large. In this paper, we propose a new classification protocol based on *logical bitmaps*. It achieves high efficiency by measuring all groups in a mixed fashion. In the meantime, we show that the new method is able to perform threshold-based classification with an accuracy that can be pre-set to any desirable level, allowing tradeoff between time efficiency and accuracy.**

I. Introduction

Radio-frequency identification (RFID) has rich application in cyber-physical systems for object tracking, automatic inventory control, and supply chain management [1], [2], [3]. Practical RFID systems widely exist for automatic toll payment, access control to parking garages, object tracking, theft prevention, tracking and monitoring. An RFID system typically consists of three components: readers, tags and the middleware software. Small RFID tags, each with a unique ID, are attached to objects, allowing an RFID reader to quickly access the properties of each individual object or collect statistical information about a large group of objects. Much of the existing work on RFID systems is to design tag identification protocols that read the IDs from tags [4], [5], [6], [7], [8], [9], [10], [11]. Other work designs efficient protocols to estimate the number of tags in a large RFID system [4], [5], [12], [13], [14], detects missing tags [15], [16], [17], or collects useful information [18].

This paper investigates a different problem. In practice, tags are often attached to objects belonging to different groups, for instance, different brands of shoes in a large shoe store, different titles of books in a bookstore, and goods from different countries or manufacturers in a port. One challenge is to determine whether the number of tags in each group is above or below a prescribed threshold value. The threshold may be set high to identify the populous groups, it may be set to a level that triggers certain actions such as replenishing the stocks, or even multiple thresholds can be used to classify groups based on the range of their population sizes. Solving this *multigroup threshold-based classification* problem gives us a basic tool to access a large population of numerous groups.

Precise classification requires us to know the precise number of tags in each group. Tag identification protocols [4], [5], [6], [7], [8], [9], [10], [11] can do that, but it takes them significant time to complete if the number of tags is very large. One way to improve efficiency is relaxing the problem from accurate classification to approximate classification [2], where the classification accuracy can be tuned to meet a pre-defined requirement. We may use cardinality estimation protocols [4], [5], [12], [13], [14] to estimate the number of tags in each group, and classify the group based on the estimation. However, those protocols are efficient when estimating a small number of large groups, but they are not efficient when estimating a large number of small groups, because their execution time for each group is largely indifferent in group size, as we will demonstrate shortly. In [2], Sheng et al. apply group testing to approximately detect popular groups. When the number of groups above the threshold is small, their performance is good. However, the performance of the group-testing-based solution degrades quickly (in terms of the execution time) when the number of groups above threshold becomes large.

In this paper, we propose a new classification protocol that is scalable to a large number of groups. Its design is drastically different from traditional approaches that measure the size of one group at a time. It measures the sizes of all groups together at once in a mixed fashion. Yet, the new protocol is able to perform threshold-based classification with an accuracy that can be pre-set to any desirable level, allowing tradeoff between time efficiency and accuracy. Our main contributions are summarized as follows:

1. We design an iterative protocol for threshold-based classification in a multi-group RFID system based on *logical bitmaps* that share time slots uniformly at random among all groups during the process of measuring their populations. We use the maximum likelihood estimation method to extract per-group information from the shared slots. Such slot sharing greatly reduces the amount of time it takes to

978-1-4673-5946-7/13/$31.00 ©2013 IEEE

complete classification.

2. Given an accuracy requirement, we show analytically how to compute optimal system parameters that minimize the protocol execution time under the constraint of the requirement. Our estimation method based on logical bitmaps ensures that false positive/false negative ratios are bounded, where *false positive* occurs when a below-threshold group is reported as above-threshold and *false negative* occurs when an above-threshold group is not reported.

3. We comprehensively evaluate the proposed solution and compare it with existing protocols. Our simulation results match well with the analytical results, which demonstrate that the new protocol performs far better in terms of execution time than the best existing work.

The rest of the paper is organized as follows. Section II presents the system model and defines the problem to be solved. Section III discusses the related work and gives the motivation for our solution. Section IV proposes our two-phase protocol for the RFID threshold-based classification problem. Section V evaluates the new protocol through simulations. Section VI draws the conclusion.

II. Problem Definition And System Model

A. System Model

There are three types of RFID tags. Passive tags are most widely deployed today. They are cheap, but do not have internal power sources. Passive tags rely on radio waves emitted from an RFID reader to power their circuit and transmit information back to the reader through backscattering. They have short operational ranges, typically a few meters in an indoor environment. To cover a large area, arrays of RFID reader antennas must be installed. Semi-passive tags carry batteries to power their circuit, but still rely on backscattering to transmit information. Active tags use their own battery power to transmit, and consequently do not need any energy supply from the reader. Active tags operate at a much longer distance, making them particularly suitable for applications that cover a large area, where one or a few RFID readers are installed to access all tagged objects and perform management functions automatically. With richer onboard resources, active tags are likely to gain more popularity in the future, particularly when their prices drop over time as manufactural technologies are improved and markets are expanded.

Communication between readers and tags is time-slotted. Readers send out a request, which is followed by a slotted time frame during which tags transmit in their selected slots. The readers may take turns to transmit the request in order to avoid interference, or a more sophisticated scheduling algorithm may be used to allow readers that do not interfere to transmit simultaneously. When a tag transmits, as long as one reader receives the transmission correctly, the transmission will be successful. In our protocol design, we can logically treat all readers as one, which transmits a request and then listens to the tags' responses. There are different types of time slots [9], among which two types are of interest in this paper. The first type is called a tag-ID slot, whose length is denoted as T_{tag}, during which a reader is able to broadcast a 96-bit tag ID. The second type is called a short-response slot, whose length is denoted as T_{short}, which carries one-bit information: '0' for an empty slot when no tag transmits, and '1' for an non-empty slot when one or more tags transmit a signal to make the channel busy.

Significant asymmetry exists between readers and tags: Tags are supposed to be used in large quantities, and they must be cheap. The cost for a reader is less of a concern because its number in use is much fewer. Therefore, unlike tags, the reader is not limited in storage space, computation power, or energy supply. If necessary, it can be connected to a powerful server for resources. With a high-quality antenna, a reader is able to receive weak signals from tags. With low-quality antennas, although tags can receive strong signals from the reader, they cannot receive each other's weak signals. They may not even sense whether the channel is busy or idle, i.e., whether another tag is transmitting. Nor can they sense if collision has occurred when two tags transmit simultaneously. Hence, a CSMA/CA-like MAC protocol [19] cannot be assumed in an RFID system. But the reader can detect whether the channel is idle or whether collision occurs. Such asymmetry points out a design principle that we should follow: pushing the complexity to the reader while leaving the tags simple.

B. Multigroup Threshold-based Classification Problem

Consider a big warehouse with tens of thousands of items. Each item is attached with an RFID tag for communication with an RFID reader. The items are divided into different groups based on certain properties, which can be the product sub-category, production date, or production place. To support grouping, each tag ID should contain two components: a *group ID*, which identifies the group the tag belongs to, and a *member ID*, which identifies a specific tag in the group. Clearly, all tags in a group must carry the same group ID, while tags in different groups carry different group IDs. We assume that the RFID reader knows the group IDs in the system.

We define the *population* or *size* of a group as the number of tags in this group. As we have explained in the introduction, while it is possible to perform precise multigroup classification at high cost, the focus of this paper is to study efficient solutions for approximate multigroup classification. We formally define the problem as follows: Let h be the threshold and α be a large probability value. We require that any group whose population exceeds h should be reported with a probability of at least α. Let l be another integer parameter smaller than h and β be a small probability value. We also require that the probability of reporting any group with l or fewer tags should be no more than β. Let k be the population of an arbitrary group g. Our performance objectives can be expressed in terms of

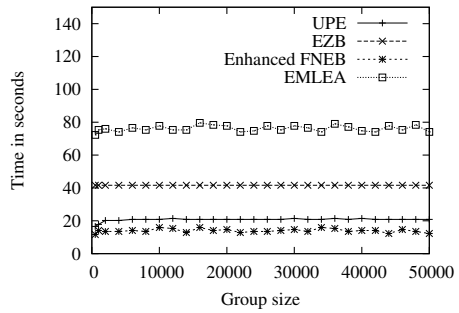

Fig. 1. The estimation time with respect to the group size for UPE, EZB, Enhanced FNEB and EMLEA, when $\alpha' = 99\%$ and $\beta' = 1\%$.

TABLE I
NOTATIONS

Symbols	Descriptions
$1-\alpha$	upper bound of false negative ratio
β	upper bound of false positive ratio
m	bit length of logical bitmap
n	number of tags
S	number of above-threshold groups
k	actual number of tags in an arbitrary group
$\hat{k}$	estimated number of tags in an arbitrary group
r_i	random number in the ith polling
f	length of the time frame each polling
$H(\cdot)$	hash function whose range is $[0, f-1]$
m_{id}	a tag's member ID
g_{id}	a tag's group ID
h	a prescribed higher bound threshold value
l	a prescribed lower bound threshold value
w	number of pollings

conditional probabilities as follows:

$$\begin{aligned} &Prob\{ \text{ group } g \text{ is reported by the reader } | k \geq h\} \geq \alpha \\ &Prob\{ \text{ group } g \text{ is reported by the reader } | k \leq l\} \leq \beta \end{aligned} \quad (1)$$

We treat the report of a group with l or fewer tags as a false positive, and the non-report of a group with h or more tags as a false negative. Hence, the above objectives can also be stated as bounding the false positive ratio by β and the false negative ratio by $1-\alpha$.

III. PRELIMINARY

A. Prior Work

One possible solution for the multigroup threshold-based classification problem is to use a reader to collect the actual tag IDs from tags, where each ID contains bits that identify the group of the tag. For a reader to successfully collect tag IDs in proximity, collision arbitration protocols must be considered so that replies from multiple tags will not be garbled due to collision. In general, collision arbitration protocols can be classified into two categories. The first category is ALOHA-based [4], [5], [6], [7]. In these protocols, communication is initialized when a reader broadcasts a polling request, followed by a slotted time frame during which tags respond. Receiving the request, each tag independently picks up a time slot to transmit its ID. To address collision, the reader has to repeat this process for a certain number of pollings before all tag IDs can be successfully received. The second category is tree-based [8], [9], [10], [11]. These protocols resolve collision by traversing a binary tree with the IDs of the tags being the leaf nodes. The reader first broadcasts an ID prefix string. The tags whose IDs match the string will respond. If collision happens, the reader will append an '0' or '1' to the prefix string and send out the new string. This process repeats until only one tag responds. Applying to the problem in this paper, these two types of protocols do not work well for large-scale RFID systems due to their long identification time.

A probabilistic analytical model is proposed by Kodialam and Nandagopal for anonymously estimating tag population [12] . Their estimators are the Zero Estimator (ZE), Collision Estimator (CE), and the Unified Probabilistic Estimator (UPE), which collect information from tags in a series of time frames and estimates the population of tags based on the number of empty slots and/or the number of collision slots. A follow-up work by the same authors proposes the Enhanced Zero-Based Estimator (EZB) [13], which is an asymptotically unbiased estimator and makes estimation only based on the number of empty slots. Qian et al. provide a replicate-insensitive estimation algorithm called the Lottery-Frame scheme (LoF) [14]. The Enhanced First Non-Empty slots Based Estimator (Enhanced FNEB) [20] can be used to estimate tag population in both static and dynamic environments by measuring the position of the first non-empty slot in each frame. Li et al. [21] study the estimation problem for large-scale RFID systems from the energy angle based on Maximum Likelihood Estimation (MLE). They design several energy-efficient probabilistic algorithms that iteratively refine a control parameter to optimize the information carried in the transmissions from tags, such that both the number and the size of the transmissions are minimized.

B. Motivation

We first show the performance of some existing work through simulation. From the simulation results, we argue that these protocols are not time-efficient when they are applied to the multigroup threshold-based classification problem.

Figure 1 presents the execution time of four existing protocols [12], [13], [20], [21] with respect to the number of tags in a group; details about the simulation setting and parameters can be found in Section V-A. The protocols are designed to estimate the size of a single group. To perform multigroup threshold-based classification, they have to estimate one group at a time. Their estimation accuracy is specified by a confidence interval: The probability for the estimate to deviate from the true group size by β' percentage or more should not exceed α', where α' and β' are two pre-specified system parameters. They are set to 99% and 1%

respectively in our simulation. From the figure, we observe that the estimation time changes very little with respect to the number of tags. For example, UPE takes about 20 seconds to estimate the tag population in the range from 500 to 50,000. If there are two groups of 25,000 tags each, the total estimation time for the two groups will be 40 seconds. However, if there are 100 groups of 500 tags each, the total estimation time will be 2,000 seconds! Hence, these protocols are not suitable when there are numerous small groups.

Sheng, Tan and Li studied how to reduce the execution time for identifying populous groups whose sizes are larger than a threshold [2]. They start with a simple fast threshold checking scheme (TCS), which approximately answers whether the number of involved tags exceeds a threshold with high probability. Based on TCS, they propose two probabilistic protocols. The first one is based on generic group testing (GT), and the second protocol is a combination of group testing and divide-and-conquer. Simulations show that their best protocol can significantly reduce the execution time for populous group discovery. However, it is also shown in the simulation results that their execution time is approximately proportional to the number of groups above the threshold. Hence, the performance of the protocol will deteriorate if the number of groups above the threshold is large. In addition, the RFID reader must be able to distinguish three types of slots: (1) empty slot, during which no tag transmit; (2) singleton slot, during which only one tag transmits, and (3) collision slot, during which more than one tag transmits.

We follow two general design principles when designing our time-efficient classification protocol. First, we want to minimize the length of each time slot. Based on the parameters of the Philips I-Code specification [22], in order to transmit a 96-bit ID from a tag to an RFID read, we need a slot of 2.11 ms. However, if the reader is not interested in IDs but wants to distinguish collision slots from singleton or empty slots [2], tags should transmit 10-bit long responses, using slots of 0.491 ms each. Furthermore, if the reader does not need to distinguish collision slots from singleton slots but only wants to know whether the slots are empty or not, tags can transmit one-bit short responses, using slots of 0.321 ms each to carry one bit information (channel busy or idle); this is the type of slots we will use in our protocol design. Note that 0.302 ms idle time is included in each slot to separate it from neighboring slots. Second, we want to minimize the number of slots. Figure 1 clearly shows that traditional approaches of measuring one group at a time will not work well when there are a large number of groups. We take a new design that is drastically different from traditional ones: measuring the groups all at once. This design has an interesting feature that its execution time is largely insensitive to the number of groups if the total number of tags is about the same. It makes our protocol particularly suitable for situations where there are a large number of small or medium-sized groups.

IV. An Efficient Threshold-based Classification Protocol

This section presents an efficient Threshold-Based Classification (TBC) Protocol, which is a combination of *dynamic slot sharing* among groups and *maximum likelihood estimation* of group sizes.

A. Dynamic Slot Sharing

Let's begin with a single group and use a well known approach [12] of estimating the group population for our discussion: The RFID reader broadcasts a polling request, asking tags to respond in a subsequent time frame of f slots. Upon receiving the request, each tag randomly selects a slot in the frame to transmit. Listening to the channel, the reader converts the time frame into a bitmap. Each bit corresponds to a slot, '0' for an empty slot and '1' for a non-empty slot. The reader then counts the number of '0' bits. Intuitively, when there are more tags transmitting, fewer slots will be left empty, which means fewer '0' bits. A functional relationship can be established between the number of tags in the group and the number of '0' bits in the bitmap [12], and we can use this function to estimate the former from the latter. When there are multiple groups, we can repeat the above scheme, using a different time frame for each group.

Analysis has shown that '0' bits must account for a reasonable portion of the bitmap in order to ensure accuracy. Hence, to handle a large tag group, we should conservatively set the frame size to be large enough such that a reasonable number of '0' bits will remain after all tags pick their slots to transmit. But when there are a mix of large and small groups, there is no one-frame-size-fit-all: A small frame size for all groups will not do well for large groups; a large frame size is good for large groups, but it is wasteful for small groups. This is particularly true when the majority of all groups are small but there are a few large ones. Can we choose a different frame size for each group based on its population? No, because per-group population is not given but what we want to know.

One way to solve the above dilemma is to share all slots among all groups. To do so, we have to abandon the traditional approach of one time frame per group [2], [12], [13], [20], [21], and instead use a single time frame for all groups — a new approach that measures the sizes of all groups together: Each group ID is pseudo-randomly hashed to a certain number of slots in the time frame. Each tag in the group will probabilistically pick one of these slots to transmit. Listening to the channel, the reader converts the time frame into a bitmap. For each group, it extracts the bits that the group ID is hashed to. Those bits form the *logical bitmap* of the group, from which the group size is estimated. In this approach, each bit and the corresponding slot may be shared by more than one group. This sharing introduces noise; the logical bitmap of one group may carry some bits that are set to '1' not by transmissions of tags in this group, but by transmissions of tags from other groups that happen to be hashed to the same time slots. Fortunately, in a bird's-eye

view, all slots are shared by all groups uniformly at random (through independent hashing), which means the noise is uniformly distributed in the whole time frame. Such uniform noise is measurable. To estimate the size of a group, we will use the logical bitmap of that group, but subtract the noise that other groups introduce into the bitmap.

To further improve performance, the reader repeats the above approach multiple times to gather multiple independent logical bitmaps for each group, and estimation based on multiple logical bitmaps reduces the variance of the result.

B. Overview

Our threshold-based classification (TBC) protocol consists of three phases: the parameter-precomputing phase, the frame phase, and the report phase. The parameter-precomputing phase computes system parameters for optimal performance of the protocol. Using these parameters, the frame phase makes $w(\geq 1)$ polling requests, each of them followed by a time frame, during which tags of all groups transmit in selected slots. The reader converts each time frame into a bitmap, from which logical bitmaps are extracted. Using these logical bitmaps, the report phase employs the Maximum Likelihood Estimation (MLE) method to report the above-threshold groups.

There are three system parameters: 1) w is the number of pollings (or time frames), 2) f is the size of each time frame, i.e., the number of slots in a frame or the number of bits in the bitmap that the frame is converted to, and 3) m is the size of each logical bitmap. Clearly, $m < f$. The execution time of TBC is dominated by the w time frames, which have $w \times f$ time slots in total. We want to find the optimal values for w, f and m such that the constraints in (1) are met and the value of $w \times f$ is minimized.

Before presenting the parameter-precomputing phase for how the optimal system parameters are computed, we will first describe the frame phase and the report phase because deriving the formulas for optimal system parameters relies on the knowledge of how the frame phase works.

C. Frame Phase

The frame phase is composed of w pollings, which are performed in a similar way: In the ith polling (where $1 \leq i \leq w$), an RFID reader first broadcasts a request message, including a random number r_i and the system parameters, f and m. The request message also serves the purpose of synchronizing the clocks of all tags for starting a time frame of f slots right after the request.

Consider an arbitrary tag t in an arbitrary group g. The tag computes a hash value $H(g_{id} \bigoplus F(r_i, H(t_{id}) \mod m))$ as the index of the time slot selected for its transmission, where $H(\cdot)$ is a hash function whose range is $[0, f-1]$, g_{id} is the tag's group ID, t_{id} is the tag's member ID, and $F(x, y)$ is a pseudo-random number function that takes two input parameters: x and y. $F(x, y)$ uses x as the seed, generates y random numbers, and outputs the yth number. The transmission from all participating tags forms a bitmap B_i.

Clearly, for tags of group g, the indices of their selected slots in the frame can only be $H(g_{id} \bigoplus F(r_i, 0))$, $H(g_{id} \bigoplus F(r_i, 1))$, ... , $H(g_{id} \bigoplus F(r_i, m-1))$. These slots or more precisely, the bits converted from these slots, form the logical bitmap of g, denoted as $LB_i(g)$.

Note that the value of $H(t_{id}) \mod m$ gives the index of the corresponding bit in the logical bitmap. For example, if a tag selects the $H(g_{id} \bigoplus F(r_i, H(t_{id}) \mod m))$th slot to transmit, the $(H(t_{id}) \mod m)$th bit in the logical bitmap will be set to '1'. Essentially, we embed the logical bitmaps of all in B_i.

D. Report Phase

After w pollings, the reader obtains w bitmaps, B_i, $1 \leq i \leq w$. It sends the bitmaps to an offline data processing module. There, the logical bitmaps of each group is extracted. For an arbitrary group g, we extract a logical bitmap $LB_i(g)$ from B_i as follows: Set the jth bit of $LB_i(g)$ to be the $H(g_{id} \bigoplus F(r_i, j))$th bit in B_i, i.e., $LB_i(g)[j] = B_i[H(g_{id} \bigoplus F(r_i, j))]$, where $1 \leq i \leq w$ and $0 \leq j \leq m-1$.

Let x_i be the number of zeros observed in $LB_i(g)$. Let n be the total number of tags in the system and k be the actual population of group g. Below we derive the formula to compute an estimate $\hat{k}$ of the population.

Consider the ith polling in the frame phase and an arbitrary bit b in $LB_i(g)$. A tag in group g has a probability of $\frac{1}{m}$ to select this bit and set it to '1' because the tag only sets one of the m bits in the logical bitmap of g. Any tag in other groups has a probability of $\frac{1}{f}$ to set this bit to '1' due to dynamic slot sharing across the whole frame. Hence, the probability for b to remain zero is

$$q = (1 - \frac{1}{f})^{n-k}(1 - \frac{1}{m})^{k}. \tag{2}$$

Hence, the likelihood function L_i for us to observe x_i bits of zeros in $LB_i(g)$ is

$$\begin{aligned} L_i = &((1 - \frac{1}{f})^{n-k}(1 - \frac{1}{m})^{k})^{x_i} \\ &\times (1 - (1 - \frac{1}{f})^{n-k}(1 - \frac{1}{m})^{k})^{m-x_i}. \end{aligned} \tag{3}$$

The likelihood function L for us to observe all x_i values, $1 \leq i \leq w$, in the w logical bitmaps is $L = \prod_{i=1}^{w} L_i$. That is,

$$\begin{aligned} L = \prod_{j=1}^{w} \Big[&((1 - \frac{1}{f})^{n-k}(1 - \frac{1}{m})^{k})^{x_i} \\ &\times (1 - (1 - \frac{1}{f})^{n-k}(1 - \frac{1}{m})^{k})^{m-x_i} \Big]. \end{aligned} \tag{4}$$

We want to find an estimate $\hat{k}$ that maximizes L, namely,

$$\hat{k} = \arg\max_{k} \{L\}. \tag{5}$$

Since the maximum is not affected by monotone transformations, we take the logarithm of both sides

of (4):

$$\ln(L) = \sum_{i=1}^{w} \Big[x_i((n-k)\ln(1-\frac{1}{f}) + k\ln(1-\frac{1}{m})) + (m-x_i)\ln(1-(1-\frac{1}{f})^{n-k}(1-\frac{1}{m})^k)) \Big]. \quad (6)$$

Differentiating both sides of the above equation, we have

$$\frac{\partial \ln L}{\partial k} = \sum_{i=1}^{w} \Big[(\frac{x_i - m(1-\frac{1}{f})^{n-k}(1-\frac{1}{m})^k}{1-(1-\frac{1}{f})^{n-k}(1-\frac{1}{m})^k}) \times (\ln(1-\frac{1}{m}) - \ln(1-\frac{1}{f})) \Big]. \quad (7)$$

After setting the right side to zero and simplifying it, we have the following estimator,

$$\hat{k} = \frac{\ln(\frac{\sum_{i=1}^{w} x_i}{mw(1-\frac{1}{f})^n})}{\ln(\frac{1-\frac{1}{m}}{1-\frac{1}{f}})}. \quad (8)$$

In (7), m and f are given parameters whose values are pre-computed by the reader. The values of x_i, $1 \le i \le m$, are obtained from $LB_i(g)$. The total number n of tags can be estimated from the bitmaps, B_i, $1 \le i \le w$. Let X_i be the number of zeros in B_i. The probability for each tag to set a bit in B_i to '1' is $\frac{1}{f}$. The probability for each bit to remain zero is approximately $(1-\frac{1}{f})^n$. The likelihood function for us to observe X_i zeros in B_i, $1 \le i \le w$, is

$$\mathcal{L} = \prod_{i=1}^{w} (1-\frac{1}{f})^{nX_i}(1-(1-\frac{1}{f})^n)^{f-X_i} \quad (9)$$

Using the maximum likelihood estimation, we take the logarithm of both sides, differentiate it, and then let the right side be zero. We have

$$\sum_{i=1}^{w} X_i - wf(1-\frac{1}{f})^n = 0$$

$$n = \frac{\ln \frac{\sum_{i=1}^{w} X_i}{wf}}{\ln(1-\frac{1}{f})}, \quad (10)$$

where X_i, $1 \le i \le w$, are obtained from B_i.

For each group g, after we estimate its population $\hat{k}$ based on (8), we report the group (as an above-threshold group) if $\hat{k} \ge T$, where T is another system parameter that will be determined in the next subsection based on the probabilistic performance objectives (1).

E. Parameter-Precomputing Phase

We first develop the constraints that the system parameters must satisfy in order to achieve the probabilistic performance objectives. Based on the constraints, we determine the optimal values for the length m of logical bitmaps, the number w of pollings, the frame size f, and the parameter T.

A group g whose estimated population is $\hat{k}$ will be reported if

$$\hat{k} \ge T. \quad (11)$$

That is,

$$\frac{\ln(\frac{\sum_{i=1}^{w} x_i}{mw(1-\frac{1}{f})^n})}{\ln(\frac{1-\frac{1}{m}}{1-\frac{1}{f}})} \ge T$$

$$\sum_{i=1}^{w} x_i \le wm(\frac{1-\frac{1}{m}}{1-\frac{1}{f}})^T(1-\frac{1}{f})^n. \quad (12)$$

Let $C = wm(\frac{1-\frac{1}{m}}{1-\frac{1}{f}})^T(1-\frac{1}{f})^n$. Therefore, the probability for the reader to report a group is $Prob(\hat{k} \ge T) = Prob(\sum_{i=1}^{w} x_i \le C)$.

From (2), we know that x_i follows the binomial distribution with parameters m and $(1-\frac{1}{f})^{n-k}(1-\frac{1}{m})^k$:

$$x_i \sim Bino(m, (1-\frac{1}{f})^{n-k}(1-\frac{1}{m})^k). \quad (13)$$

Since a binomial distribution $Bino(a,b)$ can be excellently approximated by a normal distribution $Norm(ab, ab(1-b))$ when a is large enough (which is the case for m), (13) can be approximately written as

$$x_i \sim Norm(m(1-\frac{1}{f})^{n-k}(1-\frac{1}{m})^k, m(1-\frac{1}{f})^{n-k}(1-\frac{1}{m})^k(1-(1-\frac{1}{f})^{n-k}(1-\frac{1}{m})^k)). \quad (14)$$

According to (14), we know

$$\sum_{i=1}^{w} x_i \sim Norm(mw(1-\frac{1}{f})^{n-k}(1-\frac{1}{m})^k, mw(1-\frac{1}{f})^{n-k}(1-\frac{1}{m})^k(1-(1-\frac{1}{f})^{n-k}(1-\frac{1}{m})^k)). \quad (15)$$

Let $\mu = mw(1-\frac{1}{f})^{n-k}(1-\frac{1}{m})^k$ and $\sigma^2 = mw(1-\frac{1}{f})^{n-k}(1-\frac{1}{m})^k(1-(1-\frac{1}{f})^{n-k}(1-\frac{1}{m})^k)$, then we have

$$Prob(\sum_{i=1}^{w} x_i = j) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(j-\mu)^2}{2\sigma^2}} \quad (16)$$

Thus,

$$Prob(\hat{k} \ge T) = Pro(\sum_{i=1}^{w} x_i \le C) = \sum_{j=0}^{C} \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(j-\mu)^2}{2\sigma^2}} \quad (17)$$

The first performance objective in (1) can be translated into $Prob(\hat{k} \ge T | k \ge h) \ge \alpha$, which is

$$\sum_{j=0}^{C} \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(j-\mu)^2}{2\sigma^2}} \ge \alpha \quad (18)$$

where $k \ge h$. Since the left side of the inequality is an increasing function of k, we can replace the term k with h. Then, we have the first constraint for the system parameters.

$$\sum_{j=0}^{C} \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{\frac{-(j-\mu_1)^2}{2\sigma_1^2}} \ge \alpha, \quad (19)$$

where $\mu_1 = mw(1-\frac{1}{f})^{n-h}(1-\frac{1}{m})^h$ and ${\sigma_1}^2 = \mu_1(1-(1-\frac{1}{f})^{n-h}(1-\frac{1}{m})^h)$.

Similarly, the second performance objective in (1) can be translated into the following constraint,

$$\sum_{j=0}^{C} \frac{1}{\sqrt{2\pi{\sigma_2}^2}} e^{\frac{-(j-\mu_2)^2}{2{\sigma_2}^2}} \leq \beta, \tag{20}$$

where $\mu_2 = mw(1-\frac{1}{f})^{n-l}(1-\frac{1}{m})^l$ and ${\sigma_2}^2 = \mu_2(1-(1-\frac{1}{f})^{n-l}(1-\frac{1}{m})^l)$.

Our goal is to find optimal system parameters that minimize the execution time required by TBC, i.e., $w \times f$, subject to the above two constraints.

$$\begin{aligned}
&\text{Minimize } w \times f, \qquad (21)\\
&\text{subject to } \sum_{j=0}^{C} \frac{1}{\sqrt{2\pi{\sigma_1}^2}} e^{\frac{-(j-\mu_1)^2}{2{\sigma_1}^2}} \geq \alpha\\
&\sum_{j=0}^{C} \frac{1}{\sqrt{2\pi{\sigma_2}^2}} e^{\frac{-(j-\mu_2)^2}{2{\sigma_2}^2}} \leq \beta\\
&C = mw(\frac{1-\frac{1}{m}}{1-\frac{1}{f}})^T(1-\frac{1}{f})^n\\
&\mu_1 = mw(1-\frac{1}{f})^{n-h}(1-\frac{1}{m})^h\\
&{\sigma_1}^2 = \mu_1(1-(1-\frac{1}{f})^{n-h}(1-\frac{1}{m})^h)\\
&\mu_2 = mw(1-\frac{1}{f})^{n-l}(1-\frac{1}{m})^l\\
&{\sigma_2}^2 = \mu_2(1-(1-\frac{1}{f})^{n-l}(1-\frac{1}{m})^l).
\end{aligned}$$

The parameters h, l, α and β are given by the performance objectives (1). To solve the above constrained optimization problem, we need to determine the optimal values of the remaining four system parameters w, f, m and T, such that $w \times f$ is minimized. Since all these parameters are bounded integers, we may find the optimal set of parameters by exhaustive search, which occurs offline before the TBC protocol is executed. The computation process includes four loops to enumerate all possible discrete values for the four parameters.

V. Numerical Results

A. Setting

We evaluate the performance of TBC and compare it with the existing work, including the Unified Probabilistic Estimator (UPE) [12], the Enhanced First Non-Empty slots Based Estimator (Enhanced FNEB) [20], and the Group Testing (GT) [2]. UPE and Enhanced FNEB are designed for RFID population estimation, not for satisfying the probability performance objectives in (1). However, the estimation results from these two estimators can be used for classification by reporting those groups whose estimated sizes are above a threshold. GT is the most related work. It probabilistically identifies populous groups whose sizes are larger than a threshold.

Our simulation setting is based on the Philips I-Code specification [22]. Any two consecutive transmissions (from a reader to tags or from a tag to the reader) are separated by a waiting time of 0.302 ms. According to the specification, the transmission rate from a tag to the reader is different from the transmission rate from the reader to a tag. The rate from a tag to the reader is 53 Kb/sec; it takes 0.018 ms for a tag to transmit one bit. The length of a slot is calculated as the sum of a waiting time and the time for the tag to transmit a certain number of bits. Since the type of slots used by TBC and Enhanced FNEB, T_{short}, contains only one bit, the slot length is 0.321 ms. The type of slots used by UPE and GT needs to detect collision and contains 10 bits. Their slot length is $T_{long} = 0.49ms$. Comparing with the time used by tags to transmit to the reader, the time used by the reader to transmit to the tags are negligible in all four protocols.

In our simulation, $n = 500,000$, the range of group sizes is (0, 500], $h = 250$, and the value of l varies. There are 2,000 groups, and the number x of above-threshold(h) groups may vary in simulation, but its default value is 1,000. First, we randomly choose the sizes for the above-threshold groups from [250, 500]. After that, we distribute the remaining M tags into the below-threshold groups. For the first below-threshold group, we generate a random number between 1 and $\min\{249, \frac{M}{2000-x}\}$ to be its population, which is denoted as s_1. For the second below-threshold group, we select a random value between 1 and $\min\{249, \frac{M-s_1}{1999-x}\}$ as its population, which is denoted as s_2. Similarly, we assign a random value between 1 and $\min\{249, \frac{M-s_1-s_2}{1998-x}\}$ as the population for the third below-threshold group. This process is repeated for all remaining below-threshold groups. If there are still tags left unassigned, we assign them arbitrarily to below-threshold groups as long as their sizes are below 250.

For each simulation, TBC will compute the optimal value of $w \times f$ (together with the optimal system parameters). Once $w \times f$ is determined, the execution time is known, which is $T_{short} \times w \times f$ plus the time for broadcasting polling requests. GT will also compute its optimal system parameters, including the time frame size f, the number R of rounds, and the number W of shuffled groups W. The execution time required by GT is $T_{long} \times f \times W \times R$. UPE and Enhanced FNEB work under the settings based on their original papers. Their execution times are the sum of the times for measuring the populations of individual groups.

B. Execution time required in terms of α, β and l/h

We compare TBC, GT, UPE and Enhanced FNEB in terms of execution time. Tables II – IV show our simulation results under different values of l/h, α and β.

Table II shows the execution time required when $\alpha = 99\%$ and $\beta = 1\%$. From the table, we can see that TBC has a much smaller execution time than GT, UPE and Enhanced FNEB. GT takes $1'14''$ when $l = 0.1h$, which is 3.09 times of TBC. UPE and Enhanced FNEB consume an order of magnitude or more time than GT and TBC.

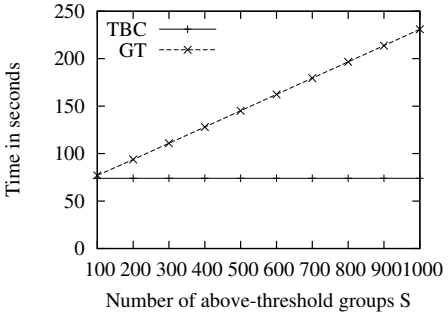

Fig. 2. Execution time with respect to the number of groups S which are supposed to be reported when $\alpha = 99\%$, $\beta = 1\%$ and $l = 0.1h$. The total number of tags n is fixed to be 500,000 at each point.

GT uses a simple, fast threshold checking scheme (TCS) to probabilistically identify populous groups with size larger than a threshold. However, TCS incurs a large variance in its estimated result. To satisfy a high accuracy requirement, a large number of TCS executions are required, which lengthens execution time. In addition, GT has to identify whether a slot is empty, singleton or collision, resulting in longer slots. It is not efficient to invoke UPE and Enhanced FNEB to estimate the size of each group one at a time. In addition, UPE requires tags to transmit 10-bit responses to distinguish singleton slots from collision slots. TBC estimate all group sizes together and share slots among all groups. It only needs to know whether each slot is empty or not. Hence, its execution time is the shortest.

In Table II, when l/h becomes larger, both TBC and GT need more time to classify the above-threshold groups. This is because a larger ratio of l/h means a higher accuracy requirement for above-threshold classification. The performance gain by TBC over GT shrinks as l/h increases, but remains significant. For example, when $l = 0.1h$, the execution time required by GT is 3.09 times that of TBC. When $l = 0.9h$, the time by GT becomes 1.9 times that of TBC.

Tables III and IV compare the execution time of the four protocols when $\alpha = 95\%$, $\beta = 5\%$, and $\alpha = 90\%$, $\beta = 10\%$, respectively. These two tables show that TBC outperforms other protocols under different parameter settings. When comparing with Table II, we see that given the same values of h and l, the execution times of all protocols are reduced when α decreases or β increases.

C. Execution time required in terms of the number of above-threshold groups

In the previous comparison, the number of above-threshold groups is set at the default value 1,000. We further compare TBC and GT by varying the number of above-threshold groups, denoted as S. Let $\alpha = 99\%$ and $\beta = 1\%$. In Fig. 2, we keep $n = 500,000$ and vary the the number of above-threshold groups from 100 to 1,000. As we see in the figure, TBC outperforms GT, and the execution time of TBC is insensitive to S. As long as the total number of tags in the system is the same, the execution time can be approximately viewed as a constant even when the number of groups is different. Such an observation agrees with (21), which does not include S in its formulation. In Fig. 3, we allow the total number of tags to change. Each below-threshold group takes a random population in the range of $(0, 250)$ and each above-threshold group takes a random population in the range of $[250, 500]$. From the figure, we observe that the classification times of TBC and GT are approximately proportional to the number of above-threshold groups. However, the line of GT

TABLE II
ESTIMATION TIME COMPARISON WHEN $\alpha = 99\%$ AND $\beta = 1\%$

	Estimation Time in minutes($'$), seconds($''$)			
	TBC	GT	UPE	Enhanced FNEB
$l = 0.1h$	$1'14''$	$3'49''$	$556'50''$	$336'36''$
$l = 0.3h$	$1'42''$	$4'26''$	$556'50''$	$336'36''$
$l = 0.5h$	$3'05''$	$6'15''$	$556'50''$	$336'36''$
$l = 0.7h$	$4'51''$	$9'12''$	$556'50''$	$336'36''$
$l = 0.9h$	$6'21''$	$11'58''$	$556'50''$	$336'36''$

TABLE III
ESTIMATION TIME COMPARISON WHEN $\alpha = 95\%$ AND $\beta = 5\%$

	Estimation Time in minutes($'$), seconds($''$)			
	TBC	GT	UPE	Enhanced FNEB
$l = 0.1h$	$44''$	$1'55''$	$17'31''$	$10'27''$
$l = 0.3h$	$1'6''$	$2'9''$	$17'31''$	$10'27''$
$l = 0.5h$	$1'47''$	$2'57''$	$556'50''$	$336'36''$
$l = 0.7h$	$2'39''$	$4'11''$	$556'50''$	$336'36''$
$l = 0.9h$	$4'4''$	$6'36''$	$556'50''$	$336'36''$

TABLE IV
ESTIMATION TIME COMPARISON WHEN $\alpha = 90\%$ AND $\beta = 10\%$

	Estimation Time in minutes($'$), seconds($''$)			
	TBC	GT	UPE	Enhanced FNEB
$l = 0.1h$	$38''$	$1'33''$	$10'37''$	$7'5''$
$l = 0.3h$	$52''$	$1'51''$	$10'37''$	$7'5''$
$l = 0.5h$	$1'17''$	$2'18''$	$556'50''$	$336'36''$
$l = 0.7h$	$1'56''$	$3'6''$	$556'50''$	$336'36''$
$l = 0.9h$	$2'50''$	$4'55''$	$556'50''$	$336'36''$

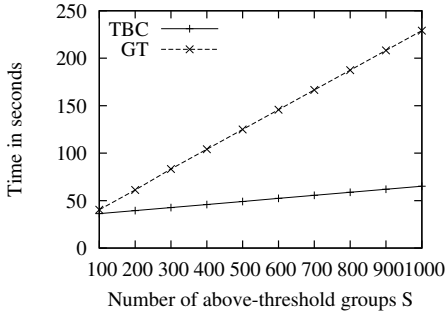

Fig. 3. Execution time with respect to the number of groups S which are supposed to be reported when $\alpha = 99\%$, $\beta = 1\%$ and $l = 0.1h$. The total number of tags n for all the groups increases along with S.

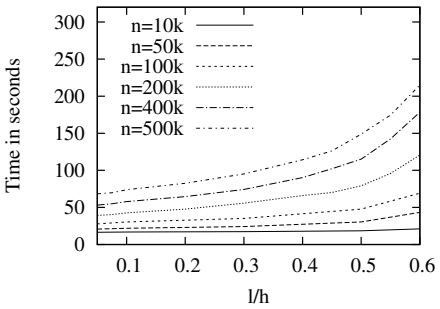

Fig. 4. Execution time with respect to the value of l/h when $\alpha = 99\%$, $\beta = 1\%$

has a larger slope than that of TBC.

D. Execution time required in terms of the total tag number

Finally, we evaluate the performance of TBC under different numbers of tags. The results are shown in Fig. 4, where $\alpha = 99\%$, $\beta = 1\%$, l/h varies from 0.1 to 0.6, and n varies from 10k to 500k. Each value of n corresponds to a curve in the figure. The execution time of TBC increases as n increases, which is expected. For all values of n, we observe that the execution time of TBC increases as l/h increases, confirming the results in Tables II-IV.

VI. Conclusion

This paper proposes a new solution for multigroup threshold-based classification in a large RFID system. While much of the prior work focuses on estimating the total number of tags in a system, it is inefficient to apply those solutions to sequentially estimate the size of each tag group and see if it is above a threshold. In this paper, we propose a new protocol based on logical bitmaps that allow the sizes of all groups to be estimated together for classification. Slot sharing is exploited to reduce the execution time. The new method is able to perform tag-group classification with any pre-set accuracy. Our protocol can be configured for tradeoff between time efficiency and accuracy. We evaluate the proposed solution and compare it with existing protocols through simulations, which demonstrate that the new protocol performs better in terms of execution time than the best existing work.

VII. Acknowledgement

This work was supported in part by the National Science Foundation under grant CNS-1115548.

References

[1] L. Ni, Y. Liu, and Y. C. Lau, "Landmarc: Indoor Location Sensing using Active RFID," *Proc. of IEEE PerCom*, 2003.
[2] B. Sheng, Chiu C. Tan, Q. Li, and W. Mao, "Finding Popular Categories for RFID Tags," *Proc. ACM MOBIHOC*, May 2008.
[3] Q. Yao, Y. Qi, J. Han, J. Zhao, X. Li, and Y. Liu, "Randomizing RFID Private Authentication," *Proc. of IEEE PerCom*, 2009.
[4] H. Vogt, "Efficient Object Identification with Passive RFID Tags," *Proc. of IEEE PerCom*, 2002.
[5] J. Zhai and G. N. Wang, "An Anti-Collision Algorithm Using Two-functioned Estimation for RFID Tags," *Proc. of ICCSA*, 2005.
[6] J. Cha and J. Kim, "Novel Anti-collision Algorithms for Fast Object Identification in RFID System," *Proc. IEEE ICPADS*, 2005.
[7] D. Klair, K. Chin, and R. Raad, "On the Energy Consumption of Pure and Slotted Aloha based RFID Anti-Collision Protocols," *Computer Communications*, 2008.
[8] D. Hush and C. Wood, "Analysis of Tree Algorithm for RFID Arbitration," *Proc. of IEEE ISIT*, 1998.
[9] J. Myung and W. Lee, "An adaptive memoryless tag anti-collision protocol for RFID networks," *Proc. IEEE ICC*, 2005.
[10] H. Choi, J. Cha, and J. Kim, "Fast Wireless Anti-collision Algorithm in Ubiquitous ID System," *Proc. IEEE VTC*, Sep 2004.
[11] V. Namboodiri and L. Gao, "Energy-Aware Tag Anti-Collision Protocols for RFID Systems," *Proc. of IEEE PerCom*, 2007.
[12] M. Kodialam and T. Nandagopal, "Fast and Reliable Estimation Schemes in RFID Systems," *Proc. of ACM MOBICOM, Los Angeles*, 2006.
[13] M. Kodialam, T. Nandagopal, and W. Lau, "Anonymous Tracking using RFID tags," *Proc. of IEEE INFOCOM*, 2007.
[14] C. Qian, H. Ngan, and Y. Liu, "Cardinality Estimation for Large-scale RFID Systems," *Proc. of IEEE PerCom*, 2008.
[15] S. Chen T. Li and Y. Ling, "Identifying the Missing Tags in a Large RFID System," *Proc. of ACM Mobihoc*, 2010.
[16] B. Sheng, Q. Li, and W. Mao, "Efficient Continuous Scanning in RFID Systems," *Proc. of IEEE INFOCOM*, March 2010.
[17] C. Tan, B. Sheng, and Q. Li, "How to Monitor for Missing RFID Tags," *Proc. of IEEE ICDCS*, June 2008.
[18] S. Chen, M. Zhang, and B. Xiao, "Efficient Information Collection Protocols for Sensor-augmented RFID Networks," *Proc. of IEEE INFOCOM*, April 2011.
[19] B. Bianchi, L. Fratta, and M. Oliveri, "Performance Evaluation and Enhancement of the CSMA/CA MAC Protocol for 802.11 Wireless LANs," *Personal, Indoor and Mobile Radio Communications, IEEE International Symposium on*, vol. 2, pp. 392–396, 1996.
[20] H. Han, B. Sheng, Chiu C. Tan, Q. Li, W. Mao, and S. Lu, "Counting RFID Tags Efficiently and Anonymously," *Proc. IEEE INFOCOM*, March 2010.
[21] T. Li, S. Wu, S. Chen, and M. Yang, "Energy Efficient Algorithms for the RFID Estimation Problem," *Proc. IEEE INFOCOM*, March 2010.
[22] Philips Semiconductors, "I-CODE Smart Label RFID Tags," *http://www.nxp.com/acrobat_download/other/identification/SL092030.pdf*, Jan 2004.