An Efficient Protocol for RFID Multigroup Threshold-Based Classification Based on Sampling and Logical Bitmap

Wen Luo, Yan Qiao, Shigang Chen, *Senior Member, IEEE*, and Min Chen

***Abstract*—Most existing research adopts a "flat" view of radio frequency identification (RFID) systems to perform various functions of collecting tag IDs, estimating the number of tags, detecting the missing tags, etc. However, in practice, tags are often attached to objects of different groups, which may represent different product types in a warehouse, different book categories in a library, etc. As we move from a flat view to an organized group view, there arise many interesting problems. One of them, called *multigroup threshold-based classification*, is the focus of this paper. It 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 toward detecting a small number of populous groups. Its performance degrades quickly when the number of groups above the threshold becomes large. In this paper, we propose a new classification protocol based on *tag sampling* and *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 preset to any desirable level, allowing tradeoff between time efficiency and accuracy.**

***Index Terms*—Multigroup threshold-based classification, radio frequency identification (RFID), time efficiency.**

I. Introduction

RADIO frequency identification (RFID) has rich application in cyber-physical systems for object tracking, automatic inventory control, and supply chain management [1]–[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 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 existing research work focuses on designing tag identification protocols that read the IDs from tags [4]–[12]. Other work designs efficient protocols to estimate the number of tags in a large RFID system [4], [5], [13]–[16], detects missing tags [17]–[19], identifies unknown tags [20], searches wanted tags [21], [22], or collects useful information [23].

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]–[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 predefined requirement. We may use cardinality estimation protocols [4], [5], [13]–[15] 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, as we will demonstrate shortly. In [2], Sheng *et al.* apply group testing to approximately detecting 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

Manuscript received June 05, 2013; revised December 31, 2013, June 24, 2014, and September 29, 2014; accepted October 04, 2014; approved by IEEE/ACM Transactions on Networking Editor A. X. Liu. Date of publication November 25, 2014; date of current version February 12, 2016. This work was supported in part by the National Science Foundation under Grants CNS-1115548 and CNS-1409797.

W. Luo is with Optym, Inc., Gainesville, FL 32603 USA (e-mail: wluo@cise.ufl.edu).

Y. Qiao is with Google, Inc., Mountain View, CA 94043 USA (e-mail: yqiao@cise.ufl.edu).

S. Chen and M. Chen are with the Department of Computer and Information Science and Engineering, University of Florida, Gainesville, FL 32611 USA (e-mail: sgchen@cise.ufl.edu; min@cise.ufl.edu).

Digital Object Identifier 10.1109/TNET.2014.2367520

preset to any desirable level, allowing tradeoff between time efficiency and accuracy. Our main contributions are summarized as follows.

1) We design a new protocol for threshold-based classification in a multigroup RFID system based on *tag sampling* and *logical bitmaps*, which 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 complete classification. Sampling further improves the performance of the protocol significantly.
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 sampling and 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 to 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 on-board resources, active tags are likely to gain more popularity in the future, particularly when their prices drop over time as manufacturing 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. We use two types of slots to carry tag responses. The first type is called a long-response slot, whose length is denoted as T_{long}, during which a tag transmits multiple bits, allowing the reader to tell whether there is collision in a slot. 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 nonempty slot when one or more tags transmit signal to make the channel busy.

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 subcategory, production date, or production place. To support grouping, each tag ID should contain two components: a *group ID*, which identifies the group to which the tag belongs, 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 Section I, 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 conditional probabilities as follows:

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

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 $1-\alpha$.

In practice, there may be multiple thresholds, each of which is used to classify a subset of groups. For example, in a warehouse that stores shoes, computers, and other products, the thresholds for shoes and computers may be different because their expected inventory levels may not be the same. When there are multiple thresholds, we first place tag groups in subsets, each of which corresponds to a threshold. We then perform single-threshold classification within each subset. To do so, the reader broadcasts the group IDs in the subset, and the classification is performed

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

among the tags that carry one of those IDs. Some notations used in this paper are given in Table I for quick reference.

III. Preliminary

A. Prior Work

Sheng *et al.* studied the multigroup threshold-based classification problem in [2]. They begin with a simple threshold checking scheme (TCS) to approximately answers whether the number of tags exceeds a threshold. Based on TCS, they propose two probabilistic protocols. The first one is based on generic group testing (GT), which consists of multiple rounds. In each round, the reader shuffles all groups into different categories, each of which may contain tags from multiple groups. TCS is then applied to check the number of tags in each category. The categories with sufficient tags are labeled as potential populous categories, which may include above-threshold groups. In the end, the testing history is used to classify all above-threshold groups. The second protocol is a combination of group testing and divide-and-conquer, which ignores the categories that fail to pass the TCS tests in the previous round, divides the remaining categories into multiple subcategories, and applies TCS to each subcategories in the remaining rounds.

Another possible solution for the multigroup threshold-based classification problem is to use a reader to collect the actual tag IDs from tags [4]–[11], where each ID contains bits that identify the group of the tag. Applied to the problem in this paper, these ID-collection protocols do not work well for large-scale RFID systems due to their long identification time.

Many methods were proposed to estimate the whole population of an RFID system. They are essentially single-group estimators. We can use them to first estimate individual group sizes (one group at a time) and then use the sizes for classification purpose. Kodialam and Nandagopal propose the first set of single-group estimators, including the Zero Estimator (ZE), the Collision Estimator (CE), and the Unified Probabilistic Estimator (UPE), which collect information from tags in a series of time frames and estimates the whole population of tags in the system based on the number of empty slots and/or the number of collision slots [13]. A follow-up work by the same authors proposes the Enhanced Zero-Based Estimator (EZB) [14], 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) [15]. The Enhanced First Non-Empty slots Based Estimator (Enhanced FNEB) [24] can be used to estimate tag population in both static and dynamic environments by measuring the position of the first nonempty slot in each frame. Li *et al.* [25] study the estimation problem for large-scale RFID systems from the energy angle based on an Enhanced Maximum Likelihood Estimation Algorithm (EMLEA). 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. The Average Run based Tag estimation (ART) scheme [16] further reduces the execution time for population estimation, based on the average run length of ones in the bit string received in the standardized frame-slotted Aloha protocol. Finally, the Zero-One Estimator (ZOE) [26] provides fast and reliable cardinality by tuning the system parameters and converging to the optimal settings through a bisection search.

B. Motivation

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

Fig. 1 presents the execution time of five existing single-group estimators [13], [14], [16], [24], [25], with respect to the number of tags in the group; details about the simulation setting and parameters can be found in Sections V-A. While these estimators are designed to measure the size of a single group, they may be applied to performing multigroup threshold-based classification by estimating 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 prespecified 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, ART takes about 10 s 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 20 s. However, if there are 100 groups of 500 tags each, the total estimation time will be 1000 s. Hence, these estimators are not suitable when there are numerous small groups.

The group testing method in GT [2] can significantly reduce the execution time for populous group discovery. However, simulation results show 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 transmits;

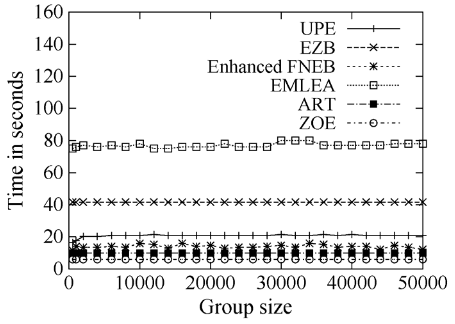


Fig. 1. Estimation time with respect to the group size for UPE, EZB, Enhanced FNEB, EMLEA, and ART, when $\alpha' = 99\%$ and $\beta' = 1\%$.

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 EPCglobal Gen-2 standard [27], in order to transmit a 96-bit ID from a tag to an RFID reader, we need a slot of 2608.8 μs. 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 470.5 μs 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 290.8 μs each to carry one bit information (channel busy or idle); this is the type of slot we will use in our protocol design. Note that 266.4 μs waiting time is included in each slot to separate it from neighboring slots.

Second, we want to minimize the number of slots. Fig. 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 and probabilistically sharing each slot by multiple groups. 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. 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

We share all slots among all groups. Notably, we abandon the approach of applying a single-group estimator [2], [13], [14], [24], [25] to measure one group at a time, but instead measure the sizes of all groups together in one time frame whose slots are shared: 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 tags from other groups introduce.

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.

Sharing slots saves time. For example, if we share each slot among 30 groups on average (as we observe in a typical simulation of Sections V), we will be able to achieve a factor of 30 reduction in execution time. However, sharing slots cause noise among groups during size estimation. Although the noise is statistically uniformly distributed, its variance requires us to repeat for additional logical bitmaps in order to average out the noise variance, which means long execution time. Fortunately, the time saved by sharing outweighs the time needed for noise removal as we will demonstrate later.

B. Overview

Our 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 four system parameters: 1) w is the number of pollings (or time frames);2) p is a sampling probability; 3) 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 4) m is the size of each logical bitmap. Clearly, $m < f$. The sampling probability is introduced so that not all tags have to participate in each polling unless $p = 1$; each tag will have a probability of p to be sampled and transmit in a polling. 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, p, 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 and the system parameters, p, 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 decides with a probability p for whether to participate in the current polling. If it decides not to, it will keep silent until the next polling request. If the tag decides to participate, it computes a hash value $H(g_{\rm id} \bigoplus F(r_i, H(t_{id}) \bmod m))$ as the index of the time-slot selected for its transmission, where $H(\cdot)$ is a hash function whose range is $[0, f-1]$, $g_{\rm id}$ is the tag's group ID, $t_{\rm id}$ is the tag's member ID, and $F(x,y)$ is a pseudo-random number generator 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 transmissions from all participating tags form a bitmap B_i.

Clearly, for tags of group g, the indices of their selected slots in the frame can only be $H(g_{\rm id} \bigoplus F(r_i, 0))$, $H(g_{\rm id} \bigoplus F(r_i, 1)), \ldots, H(g_{\rm id} \bigoplus F(r_i, m-1))$. 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 $H(t_{\rm id}) \bmod m$ gives the index of the corresponding bit in the logical bitmap. For example, if a tag selects the $H(g_{\rm id} \bigoplus F(r_i, H(t_{\rm id}) \bmod m))$th slot to transmit, the $(H(t_{\rm id}) \bmod m)$th bit in the logical bitmap will be set to "1." Essentially, we embed the logical bitmaps of all in B_i.

We point out that the complexity of our protocol is mostly placed at the RFID reader, which has to compute the optimal system parameters, initiate the protocol, receive tag transmissions, and perform classification (see Sections IV-D and IV-E). The tag's operation is relatively simple: receiving a request from the reader, performing hash, and transmitting in a time-slot. To compute $H(g_{\rm id} \bigoplus F(r_i, H(t_{\rm id}) \bmod m))$, we expect tags to implement a pseudo-random number generator F as required by [27]. A hash function may be implemented from F by using the hash input as the input to F. There are other simple ways of implementing a hash function for tags, such as [17], which uses a prestored bit ring to produce hash output.

An Aloha-based anti-collision polling scheme has been standardized by EPCglobal, where the reader begins each interrogation round by informing all the tags about the frame size. Each tag then chooses a time-slot at random and transmits only within that time-slot. More specifically, in EPCglobal C1G2 standard [27], a reader initiates each communication round with tags. The reader transmits an operation code (e.g., Query, Write, Select, ACK, etc.) indicating the expected operation of tags, the backscatter bit rate, and tag encoding schemes (e.g., FM0 or Miller). For example, the reader may initiate communication by sending tags a Query command, which includes a field that sets the number f of slots in the round. In addition, this command also involves other parameters that can be used to negotiate with the tags about the length of each slot and the waiting time between consecutive slots. Upon receiving the Query, each tag should pick a random value in the range $[0, 2^f)$ and load this value into its slot counter. It decrements the slot counter each time when receiving a QueryRep from the reader. If its slot counter is decreased to zero, it replies to the reader; otherwise, the tag shall remain silent.

Our protocol cannot be directly supported by today's off-the-shelf C1G2-compatible tags because the current standard does not support operations on group IDs (such as hashing based on a group ID for the index of a slot). However, we believe future tags (or standards) may be enhanced to support such a protocol. In our case, a new operational code called Classification needs to be defined, group IDs need to be standardized, and operations based on group IDs (such as hashing) need to be implemented on tags.

D. Report Phase

After w pollings, the reader obtains w bitmaps, B_i, $1 \le i \le w$. 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 $H(g_{\rm id} \bigoplus F(r_i, j))$th bit in B_i, i.e., $LB_i(g)[j] = B_i[H(g_{\rm id} \bigoplus F(r_i, j))]$, where $1 \le i \le w$ and $0 \le j \le m-1$.

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 p/m to select this bit and set it to "1" because the tag is sampled with probability p, and if sampled, it only sets one of the m bits in the logical bitmap of g. Any tag in other groups has a probability of p/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

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

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

$$L_i = \left(\left(1 - \frac{p}{f}\right)^{n-k} \left(1 - \frac{p}{m}\right)^k\right)^{x_i} \times \left(1 - \left(1 - \frac{p}{f}\right)^{n-k} \left(1 - \frac{p}{m}\right)^k\right)^{m-x_i}. \tag{3}$$

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

$$L = \prod_{j=1}^{w} \left[\left(\left(1 - \frac{p}{f}\right)^{n-k} \left(1 - \frac{p}{m}\right)^k\right)^{x_i} \times \left(1 - \left(1 - \frac{p}{f}\right)^{n-k} \left(1 - \frac{p}{m}\right)^k\right)^{m-x_i}\right]. \tag{4}$$

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)

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

Differentiating both sides of the above equation, we have

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

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

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

In (7), m and f are given parameters whose values are precomputed 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 be sampled and set a certain bit in B_i to "1" is p/f. The probability for each bit to remain zero is approximately $(1-(p/f))^n$. The likelihood function for us to observe X_i zeros in B_i, $1 \le i \le w$, is

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

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

$$\sum_{i=1}^{w} X_i - wf\left(1-\frac{p}{f}\right)^n = 0 \qquad n = \frac{\ln \frac{\sum_{i=1}^{w} X_i}{wf}}{\ln\left(1-\frac{p}{f}\right)} \quad (10)$$

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 Section IV-E 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

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

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

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

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

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

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

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

According to (14), we know

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

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

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

Thus

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

The first performance objective in (1) can be translated into $\text{Prob}(\hat{k} \ge T | k \ge h) \ge \alpha$, 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 $\mu_1 = mw(1-(p/f))^{n-h}(1-(p/m))^h$ and ${\sigma_1}^2 = \mu_1(1-(1-(p/f))^{n-h}(1-(p/m))^h)$.

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

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

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

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

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

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 five system parameters p, w, f, m, and T, such that $w \times f$ is minimized. We can approximately solve (21) through searching a preset parameter space. See Sections V for the search algorithm used in our simulations and the resulting protocol performance. The value of n can be estimated through an estimation protocol [13], [16], [26]. Moreover, once the performance objectives are decided, we can precompute the system parameters (p, w, f, m, and T) for different values of n (e.g., at steps of 500). After the current number of tags in the whole system is estimated, the system parameters can be quickly looked up from the precomputed results.

V. Numerical Results

A. Setting

We evaluate the performance of TBC and compare it to the GT [2], the ART scheme [16] and the ZOE [26]. GT is the most related work. It probabilistically identifies populous groups whose sizes are larger than a threshold. We denote the proposed protocol TBC with the optimal sampling probability p^* as TBC(p^*), and TBC with $p = 100\%$ as TBC(100%), which is a special case of TBC where sampling is not applied, as is published in the conference version of this paper [28]. We want to see how much improvement can be achieved through optimal sampling. ZOE and ART 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. More specifically, they estimate the group sizes one group at a time.

TABLE II
Optimal Values for the Five System Parameters When $n = 500\,000$, $\alpha = 99.9\%$, and $\beta = 0.1\%$

	Optimal values				
	p	w	f	m	T
$l = 0.1h$	0.431	6	173,658	208	141
$l = 0.3h$	0.479	7	204,362	233	167
$l = 0.5h$	0.537	9	223,075	287	195
$l = 0.7h$	0.579	12	243,436	312	228
$l = 0.9h$	0.615	17	281,607	359	263

For each group, they progressively improve the size estimation $\hat{k}$. We will reject a group when $\hat{k} \leq ((h + l)/2)$ and the probability of $|k - \hat{k}| > h - \hat{k}$ is no greater than $1 - \alpha$. We will accept a group when $\hat{k} \geq (h + l/2)$ and the probability of $|k - \hat{k}| > \hat{k} - l$ is no greater than β.

Our simulation parameters are set based on the typical setting of the EPCglobal Gen-2 standard [27]. Any two consecutive transmissions (from a reader to tags or from a tag to the reader) are separated by a waiting time of 266.4 μs. According to the specification, the transmission rate from a tag to the reader is the same as the transmission rate from the reader to a tag. The rate from a tag to the reader is 40.97 kb/s; it takes 24.4 μs 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. The type of slots used by TBC, ART, and ZOE, denoted as T_{short}, contains only one bit. Its length is 290.8 μs. The type of slots used by GT needs to detect collision and contains 10 bits. Its slot length is $T_{\text{long}} = 470.5\mu$s. The slot length for carrying a 96-bit ID is $T_{\text{ID}} = 2,608.8\mu$s. Compared to the total amount of time used by all tags to transmit to the reader, the time used by the reader to broadcast information to the tags is negligible in all three protocols.

For TBC(p^*), we approximately compute (21) with five loops for p from 0 to 1 at steps of 0.001, T from l to h, f from 0 to $10n$, w from 0 to $10n/f$, and m from 0 to f. Note that w and m are inner loops after f. The range of f is determined as follows: It does not make sense to run our protocol if its frame size is larger than the straightforward solution of collecting all tag IDs, which will give the exact size of each group. The optimal ALOHA ID-collection solution takes $2.72n(T_{\text{ID}} + T_{\text{short}})$, where T_{short} is used for acknowledgment. This is about $10nT_{\text{short}}$ slots based on the parameters in the previous paragraph. Hence, we can set the range of f from 0 to $10n$. There is no point to consider f beyond $10n$. The reason is that if the optimal value of f is equal to or greater than $10n$, we should not use the proposed protocol because a straightforward solution will perform better. Next, we consider w and m. The execution time of our protocol is $w \times f$ slots. Hence, we should set $w \times f < 10n$. The range of w is thus from 0 to $10n/f$. We know that $m \leq f$. Hence, the range of m is from 0 to f. An example of the computed optimal parameters is given in Table II.[1]

Once we find the best value of $w \times f$, the execution time is known, which is $T_{\text{short}} \times w \times f$ plus the time of estimating

[1]In our simulations, the optimal value of w is consistently below 50, and the optimal value of m is consistently below 1000.

TABLE III
ESTIMATION TIME COMPARISON WHEN $\alpha = 99.9\%$ AND $\beta = 0.1\%$

	Estimation Time in minutes(′), seconds(″)				
	TBC(100%)	TBC(p^*)	GT	ART	ZOE
$l = 0.1h$	$7'23''$	$4'43''$	$20'23''$	$57'54''$	$48'10''$
$l = 0.3h$	$8'59''$	$6'56''$	$25'57''$	$68'18''$	$57'19''$
$l = 0.5h$	$10'15''$	$9'19''$	$33'43''$	$81'50''$	$69'13''$
$l = 0.7h$	$14'23''$	$12'12''$	$42'12''$	$96'5''$	$80'30''$
$l = 0.9h$	$20'34''$	$17'57''$	$61'39''$	$114'19''$	$95'1''$

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

	Estimation Time in minutes(′), seconds(″)				
	TBC(100%)	TBC(p^*)	GT	ART	ZOE
$l = 0.1h$	$1'14''$	$39''$	$3'49''$	$39'25''$	$26'14''$
$l = 0.3h$	$1'42''$	$57''$	$4'26''$	$46'31''$	$28'05''$
$l = 0.5h$	$3'05''$	$1'44''$	$6'15''$	$61'50''$	$35'56''$
$l = 0.7h$	$4'51''$	$2'52''$	$9'12''$	$84'34''$	$47'15''$
$l = 0.9h$	$6'21''$	$4'05''$	$11'58''$	$100'24''$	$59'23''$

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

	Estimation Time in minutes(′), seconds(″)				
	TBC(100%)	TBC(p^*)	GT	ART	ZOE
$l = 0.1h$	$44''$	$18''$	$1'55''$	$9'31''$	$7'47''$
$l = 0.3h$	$1'6''$	$29''$	$2'9''$	$11'12''$	$8'23''$
$l = 0.5h$	$1'47''$	$1'01''$	$2'57''$	$12'10''$	$9'26''$
$l = 0.7h$	$2'39''$	$1'43''$	$4'11''$	$14'21''$	$11'14''$
$l = 0.9h$	$4'4''$	$2'56''$	$6'36''$	$17'11''$	$14'32''$

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

	Estimation Time in minutes(′), seconds(″)				
	TBC(100%)	TBC(p^*)	GT	ART	ZOE
$l = 0.1h$	$38''$	$15''$	$1'33''$	$5'37''$	$5'38''$
$l = 0.3h$	$52''$	$20''$	$1'51''$	$6'19''$	$5'47''$
$l = 0.5h$	$1'17''$	$48''$	$2'18''$	$7'56''$	$6'33''$
$l = 0.7h$	$1'56''$	$1'14''$	$3'6''$	$8'43''$	$8'5''$
$l = 0.9h$	$2'50''$	$2'25''$	$4'55''$	$10'24''$	$10'11''$

the value of n using ZOE. Note that even though n is predetermined for each simulation, we assume that the reader does not know this value beforehand, and it runs ZOE once to estimate n with $\pm 5\%$ error at 95% confidence level. There is a special case: If there is only a single group to be classified, n will be the group size, and no further action will be needed for the purpose of classification. In this case, our protocol becomes ZOE, and we will set the accuracy requirements of ZOE in the same way as previously described in the first paragraph of Section V-A.

TBC(100%) works just like TBC(p^*), except that it is operated under the sampling probability $p = 1$. GT will also compute its optimal system parameters, including the time frame size f, the number R of rounds, and the number of shuffled groups W. The execution time required by GT is $T_{\text{long}} \times f \times W \times R$. The parameters of ART and ZOE are set based on their original papers.

In our simulations, $n = 500\,000$, the range of group sizes is (0, 500], $h = 250$, and the value of l varies. There are 2000 groups. The number x of above-threshold groups (whose sizes are greater than h) may vary in some simulations, but its default value is 1000. Besides randomly choosing the size of each above-threshold group from [250, 500] and that of each below-threshold group from [1, 249] in one simulation, our default way of determining group sizes is given as follows: We first 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 $\min\{249, (M/2000 - x)\}$ to be its population, which is denoted as s_1. For the second below-threshold group, we select a random value between 1 and $\min\{249, (M - s_1/1999 - x)\}$ as its population, which is denoted as s_2. Similarly, we assign a random value between 1 and $\min\{249, (M-s_1-s_2/1998-x)\}$ 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.

B. TBC in Single Threshold Scenario

1) Execution Time Required With Respect to α, β, and l/h: We compare TBC(100%), TBC(p^*), GT, ART, and ZOE in terms of execution time. Tables III–VI present our simulation results under different values of l/h, α, and β. As an example, Table II gives the optimal values for the five system parameters when $\alpha = 99.9\%$ and $\beta = 0.1\%$.

Table III shows the execution time required when $\alpha = 99.9\%$ and $\beta = 0.1\%$. From the table, we can see that TBC(100%) has a much smaller execution time than GT, ART, and ZOE. For example, GT takes $33'43''$ when $l = 0.5h$, which is about triple of the time taken by TBC(100%). When sampling is introduced, the performance of TBC improves, i.e., TBC(p^*) takes less time to classify the above-threshold groups under the same setting—only 27.2% of the time taken by GT. ART and ZOE consume an order of magnitude or more time than TBC(p^*).

When l/h becomes larger, TBC(100%), TBC(p^*), GT, ART, and ZOE need more time to classify the above-threshold groups. This is because a larger ratio of l/h means a higher accuracy requirement for classification. The performance gain by TBC(p^*) over GT, ART, ZOE, and TBC(100%) shrinks as l/h increases, but remains significant. For example, when $l = 0.7h$, the execution time required by TBC(p^*) is 31.2% of the time by GT. When $l = 0.9h$, the time by TBC(p^*) is 34.1% of that by GT.

GT uses a simple, fast threshold checking scheme to probabilistically identify populous groups with size larger than a threshold. However, it incurs a large variance in its estimated result. To satisfy a high accuracy requirement, a large number of executions is required, which lengthens execution time. In addition, GT has to identify whether a slot is empty, singleton, or collision, resulting in longer slots. TBC(100%) estimates all group sizes together and shares slots among all groups. In addition, it only needs to know whether each slot is empty or not. Hence, its execution time is shorter. Sampling is turned on in TBC(p^*), which requires only a fraction of tags to participate in a protocol execution. When the sampling probability p is chosen to be a small value, the number of participating tags is largely

TABLE VII
FALSE NEGATIVE RATIO AND FALSE POSITIVE RATIO WHEN $\alpha = 99\%$ AND $\beta = 1\%$

	FNR			FPR		
	TBC(100%)	TBC(p^*)	GT	TBC(100%)	TBC(p^*)	GT
$l = 0.1h$	0.0084	0.0095	0.0098	0.0075	0.0087	0.0086
$l = 0.3h$	0.0084	0.0095	0.0098	0.0075	0.0089	0.0090
$l = 0.5h$	0.0084	0.0095	0.0098	0.0079	0.0089	0.0090
$l = 0.7h$	0.0084	0.0095	0.0098	0.0081	0.0090	0.0090
$l = 0.9h$	0.0084	0.0095	0.0098	0.0089	0.0091	0.0093

TABLE VIII
FALSE NEGATIVE RATIO AND FALSE POSITIVE RATIO WHEN $\alpha = 95\%$ AND $\beta = 5\%$

	FNR			FPR		
	TBC(100%)	TBC(p^*)	GT	TBC(100%)	TBC(p^*)	GT
$l = 0.1h$	0.037	0.042	0.046	0.04	0.039	0.041
$l = 0.3h$	0.037	0.042	0.046	0.041	0.042	0.042
$l = 0.5h$	0.037	0.042	0.046	0.043	0.043	0.042
$l = 0.7h$	0.037	0.042	0.046	0.043	0.046	0.042
$l = 0.9h$	0.037	0.042	0.046	0.044	0.047	0.045

reduced, which in turn reduces the frame size for each polling. It is not efficient to invoke ART and ZOE to estimate the size of each group one at a time.

Tables IV and VI compare the execution times of the five protocols when $\alpha = 99\%$, $\beta = 1\%$; $\alpha = 95\%$, $\beta = 5\%$; and $\alpha = 90\%$, $\beta = 10\%$, respectively. These three tables show that TBC(p^*) outperforms other protocols under different parameter settings. When compared to Table III, 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. However, the performance gain by TBC(p^*) remains significant.

2) FPR and FNR with respect to α, β, and l/h: We call a group whose size is no more than l (no less than h) as a below-l (above-h) group. The *false positive ratio* (FPR) is defined as the fraction of below-l groups that are mistakenly reported. The *false negative ratio* (FNR) is defined as the fraction of above-h groups that are not reported. Tables VII and VIII present our simulation results of FNR and FPR under different values of α, β, and l/h. For Table VII, $\alpha = 99\%$ and $\beta = 1\%$. The FNR values of TBC(100%), TBC(p^*) and GT are consistently smaller than $1-\alpha$, and their FPR values are consistently smaller than β, which means that these protocols meet the performance objectives in (1). In addition, we observe that our protocol has smaller FNR and FPR than GT. The results for $\alpha = 95\%$ and $\beta = 5\%$ are in Table VIII, where the values of FPR and FNR also meet the objectives.

3) Execution Time Required With Respect to the Number of Above-Threshold Groups: In the previous comparison, the number of above-threshold groups is set at the default value 1000. We further compare TBC(100%), TBC(p^*), 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 number of above-threshold groups from 250 to 1250. From the figure, the execution time of GT is linear in S, but TBC(100%) and TBC(p^*) are different. Not only do they outperform GT, but also their execution times are both insensitive to S. As long as the total number of tags in the system is the same, their execution times can be approximately viewed as constants even when the number of groups is different. Such an observation agrees with (21), which does not

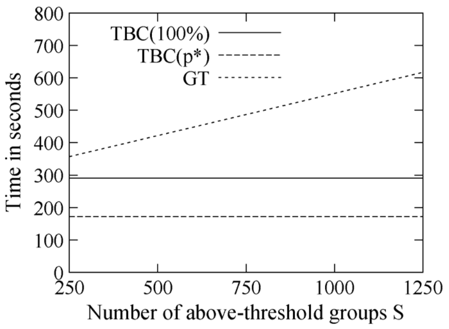


Fig. 2. Execution time with respect to the number of groups S that are supposed to be reported when $\alpha = 99\%$, $\beta = 1\%$, and $l = 0.7$ h. The total number of tags n is fixed to be 500 000 at each point.

include S in its formulation. Furthermore, thanks to sampling, it takes TBC(p^*) less time to classify the above-threshold groups than TBC(100%), for any value of S.

In Fig. 3, we fix the number of groups to 2000, while allowing 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 execution times of TBC(100%), TBC(p^*), and GT are approximately proportional to the number of above-threshold groups. However, the lines of TBC(p^*) and TBC(100%) have somewhat smaller slopes than the line of GT.

4) Execution Time Required With Respect to the Number of Groups: We let the number of groups in the system increase from 250 to 2000, with their group sizes randomly selected from $(0, 500]$. Fig. 4 shows the execution times of the five protocols with respect to the number of groups. All the protocols take longer time to classify more groups. However, the execution times of TBC($p*$) and TBC(100%) increase with smaller slopes, and therefore they are more scalable than ZOE, ART, and GT. For example, when there are 4000 groups, the time for ART is 27 974 s, the time for ZOE is 15 582 s, the time for GT is 753 s, and the time for TBC(100%) is 451 s, and the time for TBC($p*$) is 315 s.

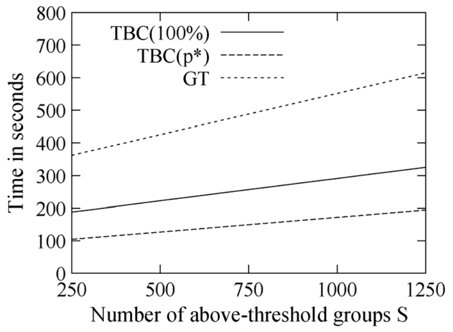


Fig. 3. Execution time with respect to the number of groups S that are supposed to be reported when $\alpha = 99\%$, $\beta = 1\%$, and $l = 0.7h$. The total number of tags n for all the groups increases along with S.

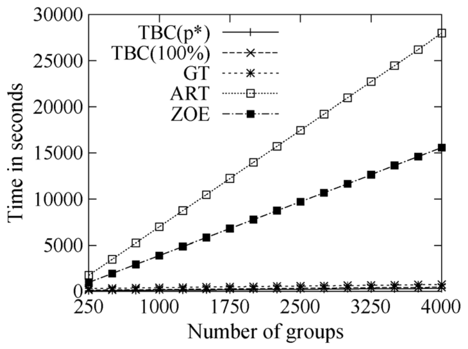


Fig. 4. Execution time with respect to the number of groups when $\alpha = 99\%$, $\beta = 1\%$, $h = 250$, $l = 0.5h$, and n increases with the number of groups.

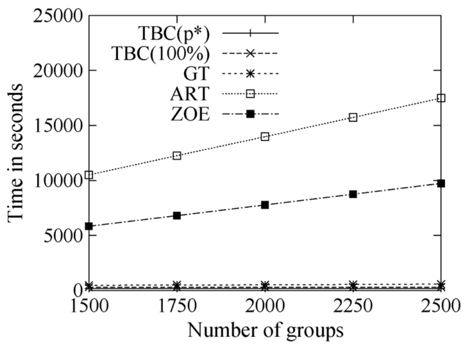


Fig. 5. Execution time with respect to the number of groups when $\alpha = 99\%$, $\beta = 1\%$, $h = 250$, $l = 0.5h$, and n is fixed at 500 000.

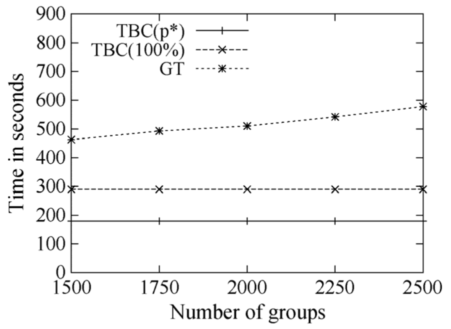


Fig. 6. Zoom-in of Fig. 5 for TBC and GT.

TABLE IX
ESTIMATION TIME COMPARISON WHEN $\alpha = 99.9\%$ AND $\beta = 0.1\%$

	Estimation Time in minutes($'$), seconds($''$)				
	TBC(100%)	TBC(p^*)	GT	ART	ZOE
$g_s = 10$	$6'31''$	$3'12''$	$14'54''$	$41'56''$	$37'35''$
$g_s = 20$	$9'32''$	$4'38''$	$16'42''$	$43'12''$	$38'41''$
$g_s = 50$	$14'14''$	$7'29''$	$21'43''$	$44'27''$	$40'1''$
$g_s = 100$	$23'50''$	$15'57''$	$29'46''$	$45'3''$	$41'38''$
$g_s = 200$	$36'28''$	$29'21''$	$40'39''$	$48'19''$	$44'12''$
$g_s = 1,000$	$45'39''$	$43'15''$	$51'33''$	$50'59''$	$48'4''$
$g_s = 2,000$	$51'46''$	$51'46''$	$56'17''$	$52'16''$	$51'46''$

Next, we show an interesting property of TBC when we fix the total number of tags in the system at 500 000 but vary the number of groups. Fig. 5 shows the execution times of the five protocols with respect to the number of groups. The times of ART and ZOE still grow roughly linearly with the number of groups. We replot the curves of GT, TBC($p*$), and TBC(100%) in Fig. 6 for a better view. The execution time of GT grows with respect to the number of groups. However, the times of TBC($p*$) and TBC(100%) stay flat, insensitive to the number of groups when the total number of tags is unchanged. The reason is simple: Each group is assigned a certain number of slots, and each tag in other groups may cause noise in one of those slots due to sharing. For TBC, the overall noise only depends on the total number of tags in other groups, not the number of other groups.

C. TBC With Multiple Thresholds

Finally, we consider the case of multiple thresholds. The 2000 tag groups are randomly divided into g_s subsets, each of which is assigned a different threshold (h, l), where l and h are randomly picked from $100 \leq l \leq h \leq 400$. The proposed protocol will be executed for each subset of groups at a time.

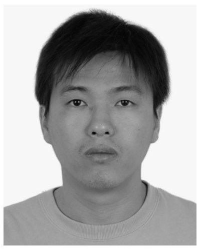
Table IX compares the execution times of the five protocols when $\alpha = 99.9\%$ and $\beta = 0.1\%$. The first column shows the different values of g_s. When $g_s = 2,000$, each subset has a single group. That is, each group has a distinct threshold. As we have explained previously, TBC becomes ZOE in this case. When g_s is smaller than 2000 and thus there are subsets with more than one group, TBC(100%) and TBC(p^*) perform better than others. The smaller the value of g_s is, the larger the average subset per threshold will be, and the more the performance gain by TBC over others will be.

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 estimating 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. Furthermore, sampling is introduced to reduce the number of participating tags (and thus collision), which in turn cuts down the execution time. Reducing the number of participating tags also saves the overall energy expenditure by the tags if battery-powered active tags are used. Our new protocol is able to perform multigroup classification with any preset accuracy. We evaluate the proposed solution and demonstrate that it compares favorably to the best existing work. We also present the method of computing the optimal system parameters.

REFERENCES

[1] L. Ni, Y. Liu, and Y. C. Lau, "LANDMARC: Indoor location sensing using active RFID," in *Proc. IEEE PerCom*, 2003, pp. 407–415.
[2] B. Sheng, C. C. Tan, Q. Li, and W. Mao, "Finding popular categories for RFID tags," in *Proc. ACM MobiHoc*, May 2008, pp. 159–168.
[3] Q. Yao *et al.*, "Randomizing RFID private authentication," in *Proc. IEEE PerCom*, 2009, pp. 1–10.
[4] H. Vogt, "Efficient object identification with passive RFID tags," in *Proc. Pervasive*, 2002, pp. 98–113.
[5] J. Zhai and G. N. Wang, "An anti-collision algorithm using two-functioned estimation for RFID tags," in *Proc. ICCSA*, 2005, pp. 702–711.
[6] J. Cha and J. Kim, "Novel anti-collision algorithms for fast object identification in RFID system," in *Proc. IEEE ICPADS*, 2005, vol. 2, pp. 63–67.
[7] D. Klair, K. Chin, and R. Raad, "On the energy consumption of pure and slotted Aloha based RFID anti-collision protocols," *Comput. Commun.*, vol. 32, no. 5, pp. 961–973, 2009.
[8] D. Hush and C. Wood, "Analysis of tree algorithm for RFID arbitration," in *Proc. IEEE ISIT*, 1998, p. 107.
[9] J. Myung and W. Lee, "An adaptive memoryless tag anti-collision protocol for RFID networks," in *Proc. IEEE ICC*, 2005.
[10] H. Choi, J. Cha, and J. Kim, "Fast wireless anti-collision algorithm in ubiquitous ID system," in *Proc. IEEE VTC*, Sep 2004, vol. 6, pp. 4589–4592.
[11] V. Namboodiri and L. Gao, "Energy-aware tag anti-collision protocols for RFID systems," in *Proc. IEEE PerCom*, 2007, pp. 23–36.
[12] L. Yang *et al.*, "Season: Shelving interference and joint identification in large-scale RFID systems," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 3092–3100.
[13] M. Kodialam and T. Nandagopal, "Fast and reliable estimation schemes in RFID systems," in *Proc. ACM MobiCom*, Los Angeles, CA, USA, 2006, pp. 322–333.
[14] M. Kodialam, T. Nandagopal, and W. Lau, "Anonymous tracking using RFID tags," in *Proc. IEEE INFOCOM*, 2007, pp. 1217–1225.
[15] C. Qian, H. Ngan, and Y. Liu, "Cardinality estimation for large-scale RFID systems," in *Proc. IEEE PerCom*, 2008, pp. 30–39.
[16] M. Shahzad and A. Liu, "Every bit counts: Fast and scalable RFID estimation," in *Proc. ACM MobiCom*, Aug. 2012, pp. 365–376.
[17] S. C. T. Li and Y. Ling, "Identifying the missing tags in a large RFID system," in *Proc. ACM MobiHoc*, 2010, pp. 1–10.
[18] B. Sheng, Q. Li, and W. Mao, "Efficient continuous scanning in RFID systems," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.
[19] C. Tan, B. Sheng, and Q. Li, "How to monitor for missing RFID tags," in *Proc. IEEE ICDCS*, Jun. 2008, pp. 295–302.
[20] X. Liu *et al.*, "A fast approach to unknown tag identification in large scale RFID systems," in *Proc. ICCCN*, 2013, pp. 1–7.
[21] M. Chen, S. C. W. Luo, Z. Mo, and Y. Fang, "An efficient tag search protocol in large-scale RFID systems," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 899–907.
[22] Y. Zheng and M. Li, "Fast tag searching protocol for large-scale RFID systems," *IEEE/ACM Trans. Netw.*, vol. 21, no. 3, pp. 924–934, Jun. 2012.
[23] S. Chen, M. Zhang, and B. Xiao, "Efficient information collection protocols for Sensor-augmented RFID networks," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 3101–3109.
[24] H. Han *et al.*, "Counting RFID tags efficiently and anonymously," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.
[25] T. Li, S. Wu, S. Chen, and M. Yang, "Energy efficient algorithms for the RFID estimation problem," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.
[26] Y. Zheng and M. Li, "ZOE: Fast cardinality estimation for large-scale RFID systems," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 908–916.
[27] EPCglobal, "EPC radio-frequency identity protocols Class-1 generation-2 UHF RFID protocol for communications at 860 MHz–960 MHz version 1.0.9," 2005 [Online]. Available: http://www.epcglobalinc.org/standards/uhfc1g2/uhfc1g2_{1}_0_9-standard-20050126.pdf
[28] W. Luo, Y. Qiao, and S. Chen, "An efficient protocol for RFID multi-group threshold-based classification," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 890–898.

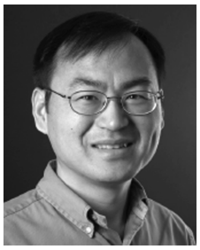


Wen Luo received the B.S. degree in computer science and technology from the University of Science and Technology of China, Hefei, China, in 2008, and the Ph.D. degree in computer and information science and engineering from the University of Florida, Gainesville, FL, USA, in 2014.

He has since worked for Optym, Inc., Gainesville, FL, USA. His research interests include RFID technologies and Internet traffic measurement.

Yan Qiao received the B.S. degree in computer science and technology from Shanghai Jiao Tong University, Shanghai, China, in 2009, and is currently pursuing the Ph.D. degree in computer and information science and engineering at the University of Florida, Gainesville, FL, USA.

Her advisor is Dr. Shigang Chen. Her research interests include network measurement, algorithms, and RFID protocols.



Shigang Chen (A'03–M'04–SM'12) received the B.S. degree from the University of Science and Technology of China, Hefei, China, in 1993, and the M.S. and Ph.D. degrees from the University of Illinois at Urbana-Champaign, Urbana, IL, USA, in 1996 and 1999, respectively, all in computer science.

After graduation, he worked with Cisco Systems, San Jose, CA, USA, for three years before joining the University of Florida, Gainesville, FL, USA, in 2002, where he is currently an Associate Professor with the Department of Computer and Information Science and Engineering. He served on the technical advisory board for Protego Networks from 2002 to 2003. He published more than 100 peer-reviewed journal/conference papers. He holds 11 US patents. His research interests include computer networks, Internet security, wireless communications, and distributed computing.

Dr. Chen is an Associate Editor for the IEEE/ACM TRANSACTIONS ON NETWORKING, *Computer Networks*, and the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY. He served in the steering committee of IEEE IWQoS from 2010 to 2013. He received IEEE Communications Society Best Tutorial Paper Award in 1999 and an NSF CAREER Award in 2007.

Min Chen received the B.E. degree in information security from the University of Science and Technology of China, Hefei, China, in 2011, and is currently pursuing the Ph.D. degree in computer and information science and engineering at the University of Florida, Gainesville, FL, USA.

His advisor is Dr. Shigang Chen, and his research interests include RFID technologies and network security.