Top-k Queries for Multi-category RFID Systems

Xiulong Liu*, Keqiu Li*, Jie Wu†, Alex X. Liu‡, Xin Xie*, Chunsheng Zhu§ and Weilian Xue¶
*School of Computer Science and Technology, Dalian University of Technology, China
†Department of Computer and Information Sciences, Temple University, USA
‡State Key Laboratory for Novel Software Technology, Nanjing University, China
§Department of Electrical and Computer Engineering, The University of British Columbia, Canada
¶School of Management, Liaoning Normal University, China

***Abstract*—This paper studies the practically important problem of *top-k queries*, which is to find the top k largest categories and their corresponding sizes. In this paper, we propose a Top-k Query (TKQ) protocol and a technique that we call Segmented Perfect Hashing (SPH) for optimizing TKQ. Specifically, TKQ is based on the framed slotted Aloha protocol. Each tag responds to the reader with a Single-One Geometric (SOG) string using the ON-OFF Keying modulation. TKQ leverages the length of continuous leading 1s in the combined signal to estimate the corresponding category size. TKQ can quickly eliminate the sufficiently small categories, and only needs to focus on a limited number of large-size categories that require more accurate estimation. We conduct rigorous analysis to guarantee the predefined accuracy constraints. To further improve time-efficiency, we propose the SPH scheme, which improves the average frame utilization of TKQ from 36.8% to nearly 100% by establishing a bijective mapping between tag categories and slots. To minimize the overall time cost, we optimize the key parameter that trades off between communication cost and computation cost. Experimental results show that our TKQ+SPH protocol not only achieves the required accuracy constraints, but also achieves a 2.6∼7x faster speed than the existing protocols.**

***Index Terms*—RFID, Multi-category, Estimation, Top-k query.**

I. Introduction

A. Background and Problem Statement

Radio Frequency Identification (RFID) has been widely used in various applications such as inventory management [1], [2], localization [3], anti-counterfeiting [4], and human tracking [5], [6], etc. An RFID system typically consists of readers, tags, and a back-end server. A tag is a microchip with an antenna in a compact package that has limited computing power and communication ranges. RFID tags can be classified into two types: *active tags*, which use the internal battery to power their circuits, and *passive tags*, which do not have their own power source and are powered up by harvesting the energy from the reader's electromagnetic fields. The back-end server controls the RFID reader to send commands to query the tags, and the tags respond over a shared wireless medium.

In many RFID applications, tags are categorized into different categories. For example, a warehouse may categorize tags according to the brands or manufacturers of the items that the tags are attached to. We consider a set of tags where each tag has a unique ID that consists of two fields: a *category ID* that specifies the category of the tag, and a *member ID* that identifies the tag within its category. The number of categories and the number of tags in each category are unknown in advance. This paper studies the practically important problem of *top-k queries*, which is to use RFID readers to query the tags so that we can quickly obtain the *top-k largest categories* and *the size of each such category*. For example, a vendor may want to know the most popular categories shipped in a day, or the least consumed types of goods in its warehouse [7]. We define the *Multi-category RFID Top-k Query* problem as: given a set of tags that can be classified into ℓ categories $C_1, C_2, ..., C_\ell$, error thresholds $\varepsilon, \alpha \in (0, 1]$, and reliability requirements $\delta, \beta \in [0, 1)$, a multi-category RFID top-k query scheme outputs a set of k categories $\mathcal{K}$ and the size of each category in $\mathcal{K}$, which satisfy the following constraints. Here, n_i is the actual size of C_i and $\hat{n_i}$ is the estimated size of C_i for $1 \leq i \leq \ell$, and $\mathcal{M} = \max\{n_j | C_j \notin \mathcal{K}\}$ (*i.e.*, $\mathcal{M}$ is the size of the largest non-top-k category):

$$\begin{aligned} &\textbf{\textit{Membership Constraint:}}\ \forall C_i \in \mathcal{K}, Pr\left[n_i \geq (1-\varepsilon)\mathcal{M}\right] \geq \delta \\ &\textbf{\textit{Population Constraint:}}\ \forall C_i \in \mathcal{K}, Pr\left[|\hat{n_i} - n_i| \leq \alpha n_i\right] \geq \beta \end{aligned} \quad (1)$$

The membership constraint means that for any category C_i in $\mathcal{K}$, the probability that its size is larger than or equal to $(1-\varepsilon)\mathcal{M}$ is larger than or equal to δ. The population constraint means that for any category C_i in $\mathcal{K}$, the probability that the size difference between its actual size n_i and its estimated size $\hat{n_i}$ is less than or equal to $n_i\alpha$ is larger than or equal to β.

B. Limitations of Prior Art

A straightforward solution to the multi-category RFID top-k query problem is to use the tag identification protocols [8], [9] to read the ID of each tag. Although perfectly accurate, they are relatively slow as they have to identify each individual tag. Another straightforward solution is to use the tag estimation protocols [10]–[14] to estimate the size of each category. Although simple, they are also relatively slow as they have to individually estimate the size of each category. The ES protocol [1] can address our problem; however, it does not scale as its frame size is equal to the number of tags.

C. Proposed Approach

In this paper, we propose a Top-k Query (TKQ) protocol and a technique that we call Segmented Perfect Hashing (SPH) for optimizing TKQ.

1) TKQ: TKQ is based on the framed slotted Aloha protocol. First, the reader broadcasts the frame size f and a random seed $\mathcal{R}$ to initialize a slotted time frame. Each tag chooses a slot $sc \in [0, f-1]$ by calculating the hash

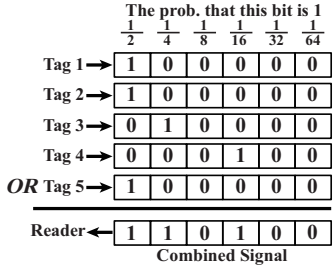

Fig. 1. Multiple SOG strings are combined based on the bitwise OR operation.

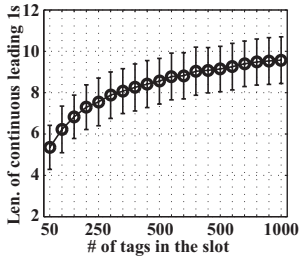

Fig. 2. Monotonous relationship between the length of continuous leading 1s and the number of tags.

function $sc = \mathcal{H}(C_i, \mathcal{R}) \mod f$. Thus, the tags from the same category always choose the same slot. For simplicity, we first assume that the category set $\mathcal{C} = \{C_1, C_2, ..C_\ell\}$ is known in advance. For each non-empty slot, if all the tags belong to the same category, then we call it a *homogeneous slot*; otherwise, we call it a *heterogeneous slot*. The reader then calculates $\mathcal{H}(C_i, \mathcal{R}) \mod f$ for $1 \leq i \leq \ell$ to predict whether the slot is a homogeneous slot, a heterogeneous slot, or an empty slot. In the slot chosen by a tag, the tag responds with a ν-bit Single-One Geometric (SOG) string, denoted by $\mathbb{SOG}[0...\nu-1]$, which satisfy two constraints: (i) only one bit is 1 and the other bits are 0s; (ii) the probability that $\mathbb{SOG}[j] = 1$ is $\frac{1}{2^{j+1}}$ for $j \in [0, \nu-1]$. The tags transmit the SOG string using the ON-OFF keying modulation [15]: a bit 1 is represented by the presence of a carrier wave; a bit 0 is represented by the absence of a carrier wave. The combined signal $\mathbb{CS}[0..\nu-1]$ is logically equal to the result of the bitwise *OR* operation on all replied strings, as illustrated in Fig. 1.

Our intuition is that for any slot, the more tags choose it, the longer the sequence of continuous leading 1s in the combined signal is. Fig. 2 illustrates this fact. Hence, we use the length of the continuous leading 1s observed in the combined signal to estimate the number of tags in the current slot. Consider the example in Fig. 1, the length of continuous leading 1s in the combined signal is 2. Using the estimator given in Lemma 1, we can estimate the number of tags that choose this slot by calculating $1.2897 \times 2^2 \approx 5.16$, which is very close to the real tag number. Since the combined signal in a heterogeneous slot can only estimate the sum of the sizes of multiple categories, TKQ uses the homogeneous slots. We use multiple independent frames so that each category can occupy sufficient homogeneous slots to achieve accurate estimate. After each round of estimation, we sort the estimated category sizes to find the top-k ones.

2) SPH: The homogeneous slots are useful for estimating category sizes. However, the ratio of homogeneous slots in the frame is up to 36.8% on average, when the frame size equals the number of categories. The low frame utilization becomes a key bottleneck of TKQ. Recall that each tag uses $\mathcal{H}(C_i, \mathcal{R}) \mod f$ to choose a slot. Essentially, the seed $\mathcal{R}$ determines the mapping between categories and slots. A straightforward method is to generate a series of seeds, from which we find the best seed $\mathcal{R}$ that establishes a *full bijective mapping* between all categories and the slots. However, as shown in Table II, finding such a full bijective mapping seed is time-consuming when the number of categories ℓ is large. For example, it takes even 9.1×10^{13} years on average when $\ell = 64$.

To improve the frame utilization of TKQ, we propose an optimization scheme called *Segmented Perfect Hashing (SPH)*. At the beginning of each round of estimation, we put all category IDs that are under estimation into a pending set $\mathcal{P}$. Instead of finding a full bijective mapping seed, SPH just needs to find a random seed $\mathcal{R}$ that establishes a bijective mapping between m categories and the first m slots in the frame. Let $\mathcal{S}$ represent the set of these m categories. Then, the reader broadcasts $\mathcal{R}$ and f to the tags to issue a frame. After executing the first m homogeneous slots, the reader sends commands to terminate the current frame immediately. Then, we set $\mathcal{P} = \mathcal{P} - \mathcal{S}$. We repeat the above process until $\mathcal{P} = \emptyset$. Logically, a long frame is divided into multiple frame segments. The slots in each segment are homogeneous slots.

D. Technical Challenges and Solutions

The first key challenge is to guarantee the accuracy constraints in (1). We propose an unbiased estimator that leverages the average length of continuous leading 1s in the homogeneous slots to estimate the category sizes. We also calculate the variance of the estimator to measure its deviation from the real category size. To satisfy the membership constraint, we propose a fundamental theorem to recognize the larger one among two arbitrary categories with a predefined reliability δ. We determine that $C_i \in \mathcal{K}$ if C_i is larger than $\ell - k$ small categories, and $C_j \notin \mathcal{K}$ if C_j is smaller than k large categories. To satisfy the population constraint, we calculate the number of homogeneous slots required for any $C_i \in \mathcal{K}$ so that the deviation of the estimated category size is small enough to satisfy the (α, β) accuracy.

The second key challenge is to optimize the frame segment size m, which is the key factor that significantly impacts the time-efficiency of SPH. Intuitively, if the frame segment size m is too large, the computation cost on the server side will be large accordingly, and will dominate the total time cost. On the contrary, if the frame segment size m is too small, the current round of estimation will contain many frame segments with each requiring an extra transmission cost for transmitting parameters $\langle \mathcal{R}, f \rangle$. Essentially, the segment size m trades off between the computation cost and the communication cost. We define an *efficiency function* $\mathcal{F}_{\langle m, |\mathcal{P}| \rangle}$ for measuring the time-efficiency of TKQ. We calculate the optimal value of m by maximizing $\mathcal{F}_{\langle m, |\mathcal{P}| \rangle}$.

E. Novelty and Advantage over Prior Art

The key novelty of this paper is twofold. First, we propose the *Top-k Query (TKQ)* protocol to find the top k largest categories and their sizes with guaranteed accuracy constraints. Second, we propose the *Segmented Perfect Hashing (SPH)* scheme to improve the average frame utilization of TKQ from 36.8% to nearly 100%. Compared with ES [1], TKQ uses a short frame size that is equal to the number of categories, whereas ES uses a frame size that is equal to the number of

tags. This leads to much faster execution speed. Compared with RFID identification protocols [8], RFID estimation protocols [11], and multi-category RFID top-k query protocol ES [1], our TKQ+SPH protocol achieves 2.6$\sim$7x speedup.

The rest of this paper is organized as follows. In Sections II and III, we present the detailed design of TKQ and SPH, respectively. In Section IV, we review the related work. In Section V, we conduct simulations to evaluate the performance of the proposed protocols. Section VI concludes this paper.

II. The Basic Protocol: TKQ

In this section, we first present the detailed design of the *Top-k Query (TKQ)* protocol. Then, we propose an unbiased estimator for estimating each category size, as well as the estimation variance. Finally, we explain how TKQ dynamically determines the top k largest categories and their sizes, meanwhile satisfying the constraints in (1).

TKQ includes several independent rounds of estimation. At the beginning of an arbitrary round of estimation, the reader broadcasts the parameters $\langle \Re, \nu \rangle$ to notify each tag to randomly generate a ν-bit SOGs string [14], [16]. Specifically, the tag first generates a ν-bit random binary string $\mathbb{B}[0...\nu-1]$ by returning the first ν bits of the hash value $\mathcal{H}(ID, \Re)$. Let μ represent the length of continuous leading 0s in $\mathbb{B}[0...\nu-1]$. The tag sets $\mathbb{SOG}[\mu] = 1$ and $\mathbb{SOG}[i \neq \mu] = 0$. Qian *et al.* indicated that $\nu = 32$ is large enough [14]. Hence, we set $\nu = 32$ in the rest of this paper. For example, a tag generates a 32-bit random string $\mathbb{B}[0...31]$ as '00101$\cdots$'. Since the length of continuous leading 0s in $\mathbb{B}[0...31]$ is 2, the tag sets $\mathbb{SOG}[2] = 1$ and $\mathbb{SOG}[i \neq 2] = 0$, *i.e.*, the generated SOG string is '00100$\cdots$'. An extreme case is that every bit in $\mathbb{B}[0...31]$ is 0, then, $\mathbb{SOG}[0...31]$ will be all 0s accordingly, which does not comply with the property of a SOG string. Fortunately, the probability of this case is as small as $\frac{1}{2^{32}}$.

Then, the reader issues a slotted time frame by broadcasting the parameters $\langle \mathcal{R}, f \rangle$, where $\mathcal{R}$ is a random seed and f is the frame size. Upon receiving these parameters, each tag initializes its slot counter sc by calculating $sc = \mathcal{H}(C_i, \mathcal{R}) \mod f$, where C_i is its category *ID*. Clearly, the tags from the same category should have the same slot counters. The reader broadcasts `QueryRep` command at the end of each slot to notify each tag to decrement its slot counter sc by one [17]. A tag will respond to the reader once its slot counter sc becomes 0. The tags from the same category C_i necessarily choose the same slot, because each tag chooses the slot by calculating $\mathcal{H}(C_i, \mathcal{R}) \mod f$. There are three types of slots: the *homogeneous slot* in which the tags are from the same category; the *heterogeneous slot* in which the tags are from different categories; the *empty slot* that no tag chooses. We are able to predict whether a slot is homogeneous, heterogeneous, or empty, by calculating $\mathcal{H}(C_i, \mathcal{R}) \mod f$ for $1 \leq i \leq \ell$. TKQ uses the combined signals in the homogeneous slots to estimate the corresponding category sizes. We set the frame size f to the number of categories because this setting results in the highest ratio of homogeneous slots in the frame on average [8]. The categories that pick the heterogeneous slots cannot be estimated in this round. Therefore, multiple frames with different seeds are required to let each category have a chance to occupy sufficient homogeneous slots for estimation.

Assume $\mathcal{N}$ rounds of estimation have been executed. We use a boolean variable $\theta_{i,x}$ to indicate whether a category C_i chooses a homogeneous slot in the x-th round of estimation. If so, $\theta_{i,x} = 1$; otherwise, $\theta_{i,x} = 0$. We now zoom in the slot that category C_i chooses in the x-th round of estimation. Let $\mathcal{L}_{i,x}$ represent the length of continuous leading 1s of the combined signal in this slot. We use s_i to denote the number of homogeneous slots that the category C_i occupies among $\mathcal{N}$ rounds of estimation, then, we have $s_i = \sum_{x=1}^{\mathcal{N}} \theta_{i,x}$. Let $\overline{\mathcal{L}}_{i,\mathcal{N}}$ represent the average length of continuous leading 1s of the combined signals in these s_i homogeneous slots. Then, we have $\overline{\mathcal{L}}_{i,\mathcal{N}} = \frac{1}{s_i} \sum_{x=1}^{\mathcal{N}} (\mathcal{L}_{i,x} \times \theta_{i,x})$. Next, Lemma 1 gives an unbiased estimator that leverages the variable $\overline{\mathcal{L}}_{i,\mathcal{N}}$ to estimate the size of category C_i. Lemma 2 calculates the variance of the estimator. The used notations are summarized in Table I.

Lemma 1. *Let s_i represent the number of homogeneous slots occupied by category C_i among $\mathcal{N}$ rounds of estimation, and $\overline{\mathcal{L}}_{i,\mathcal{N}}$ represent the average length of continuous leading '1s' of the combined signals in these slots. $\hat{n_i} = \omega 2^{\overline{\mathcal{L}}_{i,\mathcal{N}}}$ is an unbiased estimate of the category size n_i, where $\omega = 1.2897$.*

Proof. The detailed proof can be referred to in [14]. □

Lemma 2. *Let s_i represent the number of homogeneous slots occupied by category C_i among $\mathcal{N}$ rounds of estimation, and $\overline{\mathcal{L}}_{i,\mathcal{N}}$ represent the average length of continuous leading '1s' of the combined signals in these slots. The variance of $\hat{n_i} = \omega 2^{\overline{\mathcal{L}}_{i,\mathcal{N}}}$ is $Var(\hat{n_i}) = (\ln 2)^2 n_i^2 \varrho^2 / s_i$, where $\varrho = 1.1213$.*

Proof. We represent $\hat{n_i} = \omega 2^{\overline{\mathcal{L}}_{i,\mathcal{N}}}$ by $\phi\left(\overline{\mathcal{L}}_{i,\mathcal{N}}\right)$. Then, we calculate the Taylor's series expansion of $\phi\left(\overline{\mathcal{L}}_{i,\mathcal{N}}\right)$ around $\mathcal{E} = E\left(\overline{\mathcal{L}}_{i,\mathcal{N}}\right)$ as follows:

$$\begin{aligned} \hat{n_i} &= \phi\left(\overline{\mathcal{L}}_{i,\mathcal{N}}\right) = \phi\left(\mathcal{E}\right) + \left(\overline{\mathcal{L}}_{i,\mathcal{N}} - \mathcal{E}\right) \times \left\{ \frac{\partial \phi\left(\overline{\mathcal{L}}_{i,\mathcal{N}}\right)}{\partial \overline{\mathcal{L}}_{i,\mathcal{N}}} \Big|_{\overline{\mathcal{L}}_{i,\mathcal{N}} = \mathcal{E}} \right\} \\ &= n_i + \ln 2 \cdot n_i \cdot \left(\overline{\mathcal{L}}_{i,\mathcal{N}} - \mathcal{E}\right) \end{aligned} \quad (2)$$

Then, we can calculate the variance $Var(\hat{n_i})$ as follows.

$$\begin{aligned} Var(\hat{n_i}) &= E\left\{\hat{n_i} - E(\hat{n_i})\right\}^2 \\ &= (\ln 2)^2 n_i^2 E\left\{\overline{\mathcal{L}}_{i,\mathcal{N}} - E\left(\overline{\mathcal{L}}_{i,\mathcal{N}}\right)\right\}^2 \\ &= (\ln 2)^2 n_i^2 Var\left(\overline{\mathcal{L}}_{i,\mathcal{N}}\right) \end{aligned} \quad (3)$$

We know from [14] that $Var\left(\overline{\mathcal{L}}_{i,\mathcal{N}}\right) = \varrho^2 / s_i$, where $\varrho = 1.1213$. Therefore, we have $Var(\hat{n_i}) = (\ln 2)^2 n_i^2 \varrho^2 / s_i$. □

Lemma 2 infers that, as the estimation is repeated round by round, the estimator variance for each category is expected to decrease gradually. After each round of estimation, we compare the estimated category sizes to dynamically determine whether a category belongs to the top-k set $\mathcal{K}$. It is nontrivial to investigate how to correctly compare two categories according to their estimated sizes. For two categories C_i and C_j, if the estimated sizes satisfy $\hat{n_i} < \hat{n_j}$, can we assert that the actual cardinality n_i is smaller than n_j? The answer is no, due to the inherent estimation variance: a small-size category may be

overestimated, while a large-size category may be underestimated. Next, Theorem 1 tells us how to correctly compare two category sizes with a predefined reliability $\delta \in (0,1)$.

Theorem 1. *Assume that two categories C_i and C_j occupy s_i and s_j homogeneous slots, respectively. Their estimated category sizes satisfy $\hat{n_i} > \hat{n_j}$; We have $Pr\{n_i \geq (1-\varepsilon)n_j\} \geq \delta$, if the following inequality holds.*

$$\frac{\hat{n_i} - \hat{n_j} + n_j\varepsilon}{\ln 2 \varrho \sqrt{n_i^2/s_i + n_j^2/s_j}} \geq \Phi^{-1}(\delta) \tag{4}$$

Proof. Since the estimated category sizes $\hat{n_i}$ and $\hat{n_j}$ are independent to each other, we have $E(\hat{n_i} - \hat{n_j}) = E(\hat{n_i}) - E(\hat{n_j}) = n_i - n_j$ and $Var(\hat{n_i} - \hat{n_j}) = Var(\hat{n_i}) + Var(\hat{n_j})$. According to the central limit theorem, we know $\mathcal{Z} = \frac{(\hat{n_i}-\hat{n_j})-(n_i-n_j)}{\sqrt{Var(\hat{n_i})+Var(\hat{n_j})}}$ asymptotically follows the standard normal distribution [18]. Then, $Pr\{n_i \geq (1-\varepsilon)n_j\}$ can be transformed as follows:

$$Pr\left\{\mathcal{Z} \leq \frac{\hat{n_i}-\hat{n_j}+n_j\varepsilon}{\sqrt{Var(\hat{n_i})+Var(\hat{n_j})}}\right\} = \Phi\left(\frac{\hat{n_i}-\hat{n_j}+n_j\varepsilon}{\sqrt{Var(\hat{n_i})+Var(\hat{n_j})}}\right)$$

To ensure the above probability is larger than the required reliability δ, we should guarantee $\frac{\hat{n_i}-\hat{n_j}+n_j\varepsilon}{\sqrt{Var(\hat{n_i})+Var(\hat{n_j})}} \geq \Phi^{-1}(\delta)$. Here, $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution, and $\Phi^{-1}(\cdot)$ is its inverse function. Substituting the expressions of $Var(\hat{n_i})$ and $Var(\hat{n_j})$ into the above inequality, we obtain Eq. (4). □

To satisfy the membership constraint, we determine that $C_i \in \mathcal{K}$ if we can find at least $\ell - k$ small categories, says C_j, such that C_i and each C_j satisfy the inequality in Eq. (4); $C_j \notin \mathcal{K}$ if we can find at least k large categories, says C_i, such that each C_i and C_j satisfy the inequality in Eq. (4). Next, Theorem 2 indicates how to ensure the population constraint.

Theorem 2. *Let s_i represent the number of homogeneous slots that category C_i occupies, $\alpha \in (0,1)$ be the confidence interval, $\beta \in (0,1)$ be the required reliability. To guarantee $Pr\{|\hat{n_i} - n_i| \leq n_i\alpha\} \geq \beta$, we should ensure the inequality:*

$$s_i \geq \frac{(\ln 2)^2 \varrho^2}{\alpha^2} \left\{\Phi^{-1}\left(\frac{1+\beta}{2}\right)\right\}^2 \tag{5}$$

Proof. $Pr\{|\hat{n_i} - n_i| \leq n_i \cdot \alpha\} \geq \beta$ can be transformed into:

$$Pr\left\{\frac{-\alpha n_i}{\sqrt{Var(\hat{n_i})}} \leq \mathcal{W} \leq \frac{\alpha n_i}{\sqrt{Var(\hat{n_i})}}\right\} \geq \beta$$

Here, $\mathcal{W} = \frac{\hat{n_i} - E(\hat{n_i})}{Var(\hat{n_i})}$ asymptotically follows the standard normal distribution [18]. To ensure the above inequality, we only need to guarantee $\frac{\alpha n_i}{\sqrt{Var(\hat{n_i})}} \geq \Phi^{-1}(\frac{1+\beta}{2})$. Substituting the expression $Var(\hat{n_i})$ into this inequality, we obtain Eq. (5). □

Let $\mathcal{U}$ represent the set of categories that are under estimation, which is initialized as $\mathcal{C}$ at the beginning. After each round of estimation, we delete the small categories that are out of the top-k set from $\mathcal{U}$. Additionally, we remove the large categories that should be in the top-k set, meanwhile satisfying the population constraint from $\mathcal{U}$ to $\mathcal{K}$. As the estimation process goes on, the set $\mathcal{U}$ will shrink gradually while $\mathcal{K}$ grows. The estimation process of TKQ is repeated round by round until $|\mathcal{K}| = k$. Then, we obtain the k largest categories in $\mathcal{K}$ that satisfy the constraints in Eq. (1).

TABLE I
NOTATIONS USED IN THE PAPER.

Notation	Description
ℓ	number of tag categories.
C_i / $\mathcal{C}$	C_i is category ID, $1 \leq i \leq \ell$; $\mathcal{C} = \{C_1, C_2, ..C_\ell\}$.
n_i / $\hat{n_i}$	size of category C_i; estimated size of category C_i.
$\mathbb{SOG}$[..]	Single-One Geometric (SOG) string.
$\mathbb{CS}$[..]	combined signal of multiple SOG strings.
$\mathcal{N}$	number of rounds of estimation repeated by TKQ.
$\theta_{i,x}$	$\theta_{i,x} = 1$ when C_i chooses a homogeneous slot in the x-th round of estimation; $\theta_{i,x} = 0$ for otherwise.
s_i	number of homogeneous slots occupied by C_i.
$\mathcal{L}_{i,x}$	length of continuous leading 1s in the slot that C_i chooses in the x-th round of estimation.
$\overline{\mathcal{L}}_{i,\mathcal{N}}$	average length of continuous leading 1s in the s_i homogeneous slots, of category C_i, among $\mathcal{N}$ rounds.
ε / α	error thresholds.
δ / β	reliability requirements.
$\Phi(\cdot)$ / $\Phi^{-1}(\cdot)$	cumulative distribution function of the standard normal distribution; inverse function of $\Phi(\cdot)$.
$\mathcal{H}(\cdot)$	hash function with uniform distribution.
$\mathcal{R}$	random seed.
f	frame size, *i.e.*, the number of slots in the frame.

III. The Supplementary Protocol: SPH

In this section, we first explain the motivation of proposing the *Segmented Perfect Hashing (SPH)* protocol. Then, we use a case study to elaborate on the basic idea of SPH, which is followed by the detailed protocol design. We also provide theoretical analyses to optimize the key parameters to maximize the time-efficiency of SPH. Finally, we will discuss some practical issues such as multi-reader deployment, and channel error, etc.

A. Motivation and Challenge

The homogeneous slots are useful in the TKQ protocol. Hence, the ratio of homogeneous slots in the frame can be interpreted as the frame utilization, which is given by $\binom{\ell}{1}(\frac{1}{f})(1-\frac{1}{f})^{\ell-1} \approx \rho e^{-\rho}$. Here, $\rho = \ell/f$, ℓ is the number of categories, and f is the number of slots in the frame. It is easy to find that the average frame utilization is just up to 36.8% when $\rho = 1$. The low frame utilization is a major bottleneck of TKQ. Some previous work, *e.g.*, [19], uses a bit-vector to guide the tags to skip the useless slots, thereby improving the frame utilization. However, it requires the tags to be able to interpret the bit-vector, such a functionality is hard to be implemented in the passive RFID tags.

Obviously, the ideal case is that all the slots in the frame are homogenous slots, *i.e.*, there is a bijective mapping between the tag categories and slots. Recall that, given an arbitrary seed $\mathcal{R}$, the server is able to predict the mapping between tag categories and slots even before actually executing the frame. Hence, a straightforward method is to generate a series of seeds $\mathcal{R}$, and the server tests them to choose the best one that establishes a bijective mapping between tag categories

TABLE II
THE COMPUTATION COST OF THE STRAIGHTFORWARD METHOD.

# of categories (ℓ)	# of tested seeds	Time cost on average
1	1	1.5×10^{-7}s
2	2	5.8×10^{-7}s
4	10.7	6.2×10^{-6}s
8	416.1	4.8×10^{-4}s
16	8.8×10^{5}	2.4s
32	5.6×10^{12}	297.6 days
64	3.1×10^{26}	9.1×10^{13} years

and slots. Using such a seed to issue a frame, each slot in the actual frame will be the homogeneous slot. Next, we analyze the computation complexity of this method.

For an arbitrary seed $\mathcal{R}$, the probability that it establishes a full bijective mapping between ℓ categories and $f = \ell$ slots is $P_b = \ell!/\ell^\ell$. Therefore, we need to test $1/P_b = \ell^\ell/\ell!$ hash seeds on average, so that one of them is a full bijective mapping seed. When testing a seed $\mathcal{R}$, the server first calculates the hash function $\mathcal{H}(C_i, \mathcal{R}) \mod f$ to map each category C_i to a slot in the virtual frame for each $i \in [1, \ell]$. Then, the reader checks whether all slots in the virtual frame are homogeneous slots. Let η represent the number of clock cycles required by the server to calculate the hash function, λ be the number of clock cycles that it takes to check the status of each slot in the virtual frame, and t_c be the duration of a clock cycle that depends on the CPU frequency of the server. Then, the average time of finding such a full bijective mapping seed should be $1/P_b \times (\ell\eta + \ell\lambda)t_c$. The numerical results in Table II reveal that the required computation cost grows sharply as the number of categories ℓ increases. For example, when $\ell = 64$, it takes even 9.1×10^{13} years on average to find a bijective random seed, using a single-threaded program. Therefore, establishing a bijective mapping between the categories and slots is a nontrivial issue. Note that, in Table II, we set $\eta = 344$, $\lambda = 3$ according to [20], and the clock cycle $t_c = 4.17 \times 10^{-10}s$ with the 2.4GHz CPU.

B. Case Study

Here, we give a case study, as exemplified in Fig. 3, to explain the basic idea of the *Segmented Perfect Hashing (SPH)* protocol. Assume that there are 9 tag categories $C_1 \sim C_9$ in the RFID system. At the beginning, a pending category set $\mathcal{P}$ is initialized as $\{C_1, C2, .., C_9\}$. SPH includes several mapping processes. In the first mapping, we find a random seed $\mathcal{R}_1$ that establishes a bijective mapping between 3 categories (*i.e.*, C_1, C_2, and C_4) and the first 3 slots. After executing the first 3 slots, the reader terminates the frame, and sets $\mathcal{P} = \mathcal{P} - \{C_1, C_2, C_4\}$. Here, the first 3 slots can be interpreted as a frame segment. In the second mapping, we find another random seed $\mathcal{R}_2$ that establishes a bijective mapping between 3 categories (*i.e.*, C_3, C_6, and C_7) in $\mathcal{P}$ and the first 3 slots. After executing the first 3 slots, the reader terminates the frame and sets $\mathcal{P} = \mathcal{P} - \{C_3, C_6, C_7\}$. In the third mapping, we find a random seed $\mathcal{R}_3$ that establishes a bijective mapping between

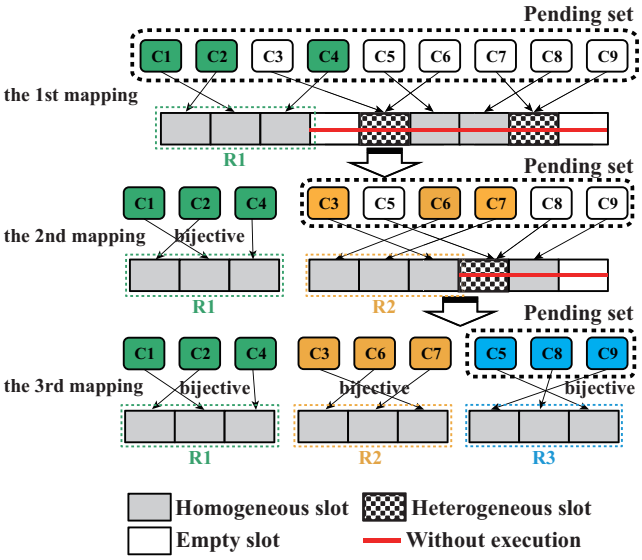

Fig. 3. A case study of Segmented Perfect Hashing (SPH).

$\mathcal{P} = \{C_5, C_8, C_9\}$ and 3 slots. After executing the last frame segment, the pending set $\mathcal{P}$ becomes $\emptyset$ and SPH terminates. Logically, we obtain a bijective mapping between 9 categories and 9 slots through 3 frame segments.

To show the advantage of SPH, we first calculate its computation cost in the example. Given an arbitrary seed $\mathcal{R}$, each category in the pending set $\mathcal{P}$ is randomly hashed to one of the $f = |\mathcal{P}|$ slots in the frame. We refer to the seed $\mathcal{R}$ as an $\langle m, |\mathcal{P}| \rangle$ *bijective seed*, if it establishes a bijective mapping between m categories and the first m slots in the frame. Let $P_{\langle m,|\mathcal{P}|\rangle}$ represent the probability that $\mathcal{R}$ is such a seed. We calculate its expression as follows:

$$P_{\langle m,|\mathcal{P}|\rangle} = \frac{\binom{|\mathcal{P}|}{m} \times m! \times (|\mathcal{P}| - m)^{|\mathcal{P}|-m}}{|\mathcal{P}|^{|\mathcal{P}|}} \quad (6)$$

To obtain an $\langle m, |\mathcal{P}| \rangle$ *bijective seed*, the server needs to test $1/P_{\langle m,|\mathcal{P}|\rangle}$ random seeds on average. The expected computation cost, denoted as $\mathcal{O}_{\langle m,|\mathcal{P}|\rangle}$, is given as follows:

$$\mathcal{O}_{\langle m,|\mathcal{P}|\rangle} = \frac{|\mathcal{P}|\eta t_c + m\lambda t_c}{P_{\langle m,|\mathcal{P}|\rangle}} = \frac{|\mathcal{P}|^{|\mathcal{P}|} \times \{|\mathcal{P}|\eta t_c + m\lambda t_c\}}{\binom{|\mathcal{P}|}{m} \times m! \times (|\mathcal{P}| - m)^{|\mathcal{P}|-m}}, \quad (7)$$

where $\{|\mathcal{P}|\eta t_c + m\lambda t_c\}$ is the computation cost of testing each seed. Now, we consider the example in Fig. 3. The overall computation cost on the server side will be $\mathcal{O}_{\langle 3,9\rangle} + \mathcal{O}_{\langle 3,6\rangle} + \mathcal{O}_{\langle 3,3\rangle}$. In fact, the straightforward method is a special case of SPH, where $m = \ell$. In this example, the computation cost of the straightforward method is $\mathcal{O}_{\langle 9,9\rangle}$. We calculate $\frac{\mathcal{O}_{\langle 9,9\rangle}}{\mathcal{O}_{\langle 3,9\rangle} + \mathcal{O}_{\langle 3,6\rangle} + \mathcal{O}_{\langle 3,3\rangle}} \approx 38.4$, which infers that SPH is about 38.4 times faster than the straightforward method in this example. Note that, SPH can show more performance improvement if larger number of tag categories are involved.

C. Detailed Design of SPH

Before presenting the detailed design of SPH, we need to re-clarify the relationship between TKQ and SPH as follows. Generally, TKQ includes multiple rounds of estimation. In

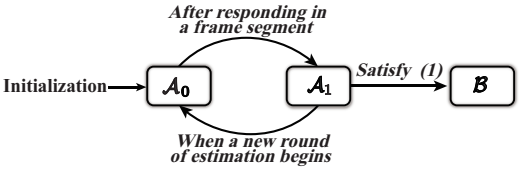

Fig. 4. State diagram of an RFID tag in SPH.

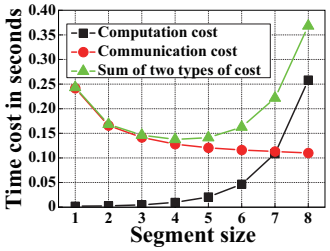

Fig. 5. Tradeoff between computation and communication costs. ℓ=50.

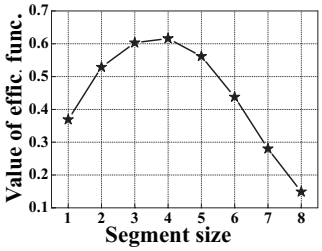

Fig. 6. Impact of segment size on the efficiency function. $|\mathcal{P}|$=50.

each round of estimation of TKQ, we use SPH to improve the frame utilization. SPH includes multiple mapping processes, each corresponds to a frame segment. In SPH, each tag needs to keep a flag that has three possible values. A tag with the flag of $\mathcal{A}0$ is active and will participate in the next mapping process; a tag with the flag of $\mathcal{A}1$ will not participate in the rest of mapping processes in current round of estimation, but it will still participate in the next round of estimation; a tag with the flag of $\mathcal{B}$ is inactive and will no longer participate in the remaining rounds of estimation. Next, we will present the SPH protocol in detail, meanwhile explaining the state transition of the RFID tags shown in Fig. 4.

At the beginning of an arbitrary round of estimation, we use $\mathcal{U}$ to represent the set of categories that are under estimation. The reader sends commands to reset the tags whose categories are within $\mathcal{U}$ to be with the flag of $\mathcal{A}0$. The pending category set $\mathcal{P}$ is initialized as $\mathcal{U}$. The server finds an $\langle m, |\mathcal{P}|\rangle$ bijective seed $\mathcal{R}$ that establishes a bijective mapping between m categories in $\mathcal{P}$ and the first m slots in the frame. We use $\mathcal{S}$ to represent the set of these m categories, clearly, $\mathcal{S} \subseteq \mathcal{P}$. The reader uses the binary parameters $\langle \mathcal{R}, f\rangle$ to issue a frame. Each tag determines a slot by calculating $sc = \mathcal{H}(\mathcal{R}, ID) \bmod f$. At the end of each slot, the reader broadcasts the command `QueryRep` to notify each tag with the flag of $\mathcal{A}0$ to decrement its slot counter sc by one [17]. Upon finding its slot counter $sc = 0$, a tag will respond to the reader with the generated SOG string, and turns its flag from $\mathcal{A}0$ to $\mathcal{A}1$. Using the combined signals received in the homogeneous slots, the reader updates the estimated category sizes as well as the estimation variance based on Lemmas 1 and 2. If a category has already satisfied the constraints in (1), the reader will sends an `ACK` command to turn the flag of the tags confined in this slot to $\mathcal{B}$. The tags with the flag of $\mathcal{B}$ will keep inactive and will no longer participate in the remaining rounds of estimation. After the execution of the first m slots, the reader terminates the current frame segment immediately, and the server updates the pending set by $\mathcal{P} = \mathcal{P} - \mathcal{S}$. Note that, the tags that responded in previous frame segments should have the flag $\mathcal{A}1$. Only the tags with flag $\mathcal{A}0$ will participate in the next mapping process. Similar processes are repeated until the pending set $\mathcal{P} = \emptyset$, which also means that the current round of estimation is finished.

D. Parameter Optimization

The time cost of SPH contains two aspects: (i) the *computation cost* on the server side for finding the $\langle m, |\mathcal{P}|\rangle$ bijective seeds; (ii) the *communication cost* for transmitting the parameters $\langle \mathcal{R}, f\rangle$ (from reader to tags) and transmitting SOG strings (from tags to reader). The numerical results in Fig. 5 reveal that the frame segment size m is a key factor that controls the tradeoff between the computation cost and the communication cost. The underlying reason is elaborated as follows. If the frame segment size m is too large, the computation cost on the server side will be large accordingly, and will dominate the total time cost. On the contrary, if the frame segment size m is too small, the current round of estimation will contain too many frame segments, each requiring the non-negligible transmission cost for sending the parameters $\langle \mathcal{R}, f\rangle$. Therefore, it is important to optimize the frame segment size m to minimize the total time cost.

Given a pending category set $\mathcal{P}$, Eq. (7) has given the computation cost for finding an $\langle m, |\mathcal{P}|\rangle$ bijective seed on the server side. Let $\mathcal{T}_{act}$ represent the actual communication cost of executing a frame segment with size m; $\mathcal{T}_{idl}$ be the ideal communication cost of executing a frame segment with size m. We have $\mathcal{T}_{act} = m\times\tau_u^\nu + \tau_d^\vartheta$ and $\mathcal{T}_{idl} = m\times\tau_u^\nu$. Here, τ_u^ν is the time to transmit the ν-bit SOG strings from tags to the reader (uplink). τ_d^ϑ is the time to transmit the ϑ-bit parameters $\langle \mathcal{R}, f\rangle$ from reader to tags (downlink) for initializing the frame. We define an *efficiency function* $\mathcal{F}_{\langle m,|\mathcal{P}|\rangle} = \frac{\mathcal{T}_{idl}}{\mathcal{T}_{act}+\mathcal{O}_{\langle m,|\mathcal{P}|\rangle}}$. Its physical meaning is that: the numerator indicates the ideal time for executing a frame segment containing m slots; the denominator indicates the actual time for executing such a frame segment. Obviously, the value of $\mathcal{F}_{\langle m,|\mathcal{P}|\rangle}$ is no more than 1. In the ideal case, where it does not involve the transmission of parameters and the computation cost on the server side, the value of $\mathcal{F}_{\langle m,|\mathcal{P}|\rangle}$ equals 1. However, such an ideal case cannot be achieved because these two costs are inevitable. We calculate the expression of $\mathcal{F}_{\langle m,|\mathcal{P}|\rangle}$ as follows:

$$\mathcal{F}_{\langle m,|\mathcal{P}|\rangle} = \frac{m(m!)\tau_u^\nu \binom{|\mathcal{P}|}{m} (|\mathcal{P}|-m)^{|\mathcal{P}|-m}}{(m\tau_u^\nu+\tau_d^\vartheta)\binom{|\mathcal{P}|}{m} m!\,(|\mathcal{P}|-m)^{|\mathcal{P}|-m} + |\mathcal{P}|^{|\mathcal{P}|}\,(|\mathcal{P}|\eta+m\lambda)\,t_c}$$

As exemplified in Fig. 6, we could find an optimal value of m by maximizing $\mathcal{F}_{\langle m,|\mathcal{P}|\rangle}$. We have observed from simulations that the optimal value of m is typically less than 15. Therefore, we can quickly find its optimal value even by the exhaustive searching, which occurs offline before running our protocol.

E. Discussion on Some Practical Issues

1) Identification of Category IDs: For the case that the category set $\mathcal{C}$ is unknown in advance, we propose a fast approach to identify the category IDs. The reader queries the tags by broadcasting parameters $\langle R, f\rangle$, and each tag determines a slot

in the frame by calculating $\mathcal{H}(C_i, \mathcal{R}) \mod f$. In the picked slot, a tag responds with its category ID C_i as well as the checksum of C_i. In a homogeneous slot, the combined signal in the category ID field will match the combined signal in the checksum field, thus, the reader is able to successfully identify the corresponding category ID. Multiple frames are repeated until all categories are identified.

2) Deployment of Multiple Readers: In large-scale application scenarios, a single reader is usually unable to cover the whole application area. Therefore, multiple readers are required to be deployed. Many excellent reader-scheduling schemes were proposed to efficiently synchronize the readers [21]. We let the query commands across all the readers be consistent, and the readers return the received data to the server. Thus, these readers cooperate like a powerful reader that can cover the whole area [11]. Then, we can migrate the proposed protocols to the multi-reader scenarios seamlessly.

IV. Related Work

Through comprehensive review of previous work, we summarize and classify the related work into four fields below.

Tag identification is to identify the exact tag IDs within the vicinity of the reader. There are two types of solutions: Aloha-based protocols [8] and tree-based protocols [9]. In the Aloha-based protocols, the tags content for slots in the frame to respond with their IDs. In the tree-based protocols, the reader broadcasts a 0/1 string to query the tags. A tag responds with its ID once it finds that the querying string is the prefix of its ID. A reader identifies a tag when one tag responds.

Probabilistic estimation is to estimate the cardinality of a tag set with a predefined accuracy constraint [10]–[14], [22]. Kodialam *et al.* proposed the first set of tag estimation protocols, USE and UPE, which use the number of empty or collision slots to estimate population sizes [23]. Similarly, Zheng *et al.* proposed the PET protocol for tree-based RFID systems [10]. Qian *et al.* first proposed using geometric distribution hash to estimate the tag cardinality of a single set [14]. Shahzad *et al.* proposed ART, which uses the average run length of non-empty slots for cardinality estimation [11]. Li *et al.* proposed the Maximum Likelihood Estimator (MLE), which looks at the energy aspect [24]. Liu *et al.*studied the problem of RFID estimation with blocker tag [22].

Iceberg query is to identify the categories whose cardinality is above the given threshold for multi-category RFID systems. Sheng *et al.* proposed Group Testing (GT) scheme to rapidly eliminate the groups that contain multiple small-size categories [7]. Luo *et al.* proposed a Threshold-Based Classification (TBC) Protocol, which can obtain multiple logical bitmaps from a single time frame. Each bitmap can be used to estimate the tag cardinality of a category. The categories whose sizes are obviously above the threshold can be quickly removed. The iceberg query protocols cannot be borrowed to solve the problem of top-k query, because the threshold (*i.e.*, the size of the k-th largest category) is unknown previously and hard to obtain in some applications.

Top-k query is to pinpoint the k largest categories within a multi-category RFID system. Xie *et al.* made the first and only dedicated effort to address the top-k query problem [1]. In the Ensemble Sampling (ES) protocol [1], all the tags contend for a common slotted frame, and each responds with the category ID in a random slot. ES leverages the ratio of singleton slots carrying category ID C_i to the total singleton slots to estimate the category size n_i, and dynamically finds the top k largest categories. In Section I-B, we have discussed the limitations of the related work.

V. Performance Evaluation

We implemented the proposed protocols in Matlab on a ThinkPad X230 desktop with an Intel 2.4GHz CPU. The simulated RFID system contains ℓ tag categories. We randomly generate the category sizes following the normal distribution $Norm(\mu, \sigma)$ [1]. Recall that our protocols can seamlessly work in both multi-reader and single reader scenarios. Same as [1], [8], [11], we simulate a single reader that has sufficient power to probe all tags. The transmission rate between the reader and tags is asymmetric. The uplink (tags to reader) rate is $53Kb/s$, while the downlink rate is $26.5Kb/s$. Between any two consecutive data transmissions, there is a waiting time $\tau_w = 302us$ [11]. The clock cycle t_c of the computer is $4.17 \times 10^{-10}s$. For clarity, the programs for finding the bijective seeds required by SPH is in a single-threaded manner. We compare TKQ+SPH with three representative protocols: the *Ensemble Sampling (ES)* protocol [1], which is the only dedicated solution to the top-k query problem; the *Enhanced Dynamic Framed Slotted ALOHA (EDFSA)* protocol [8], which is the identification protocol specified in the C1G2 standard; the *Average Run-based Tag estimation (ART)* protocol [11], which is an excellent tag estimation protocol. Extensive simulations under various settings are conducted to evaluate the time-efficiency and the reliability of these protocols. Each simulation result is averaged from 500 independent trials.

A. Time-efficiency

1) Impact of k: TKQ+SPH is the fastest protocol with varying number k of concerned categories. Without otherwise specified, we set both the membership accuracy (ε, δ) and the population accuracy (α, β) to $(0.05, 95\%)$ by default. The number ℓ of total categories is fixed to 100, and the cardinality of each category is randomly generated following normal distribution $Norm(\mu, \sigma)$, where $\mu = 500$ and $\sigma = 400$. Fig. 7(a) shows the execution time of each protocol with varying k. The execution time of EDFSA is stable because it has to exactly identify all the tags, regardless of the number of concerned categories. The performance of ART is also independent of k because it needs to estimate the population of each category in a separate manner. On the contrary, the execution time of ES, TKQ and TKQ+SPH increases against k. The underlying reason is as follows. A larger k means that more large-size categories have to be kept in the estimation process until meeting the population constraint, which will incur more execution time. TKQ+SPH significantly

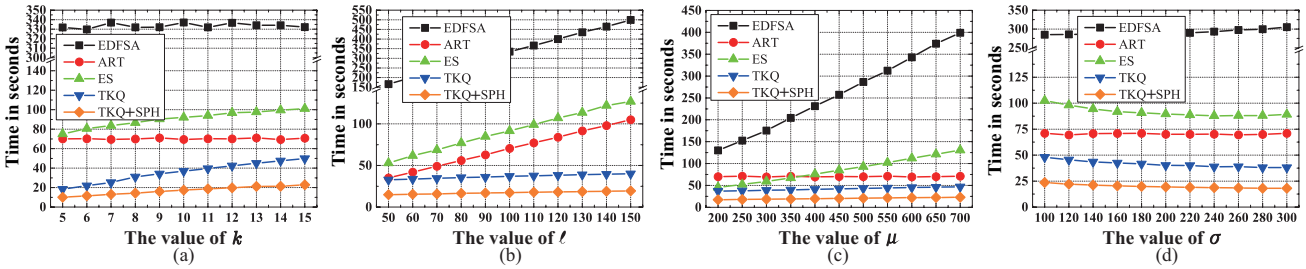

Fig. 7. Investigating the time-efficiency of protocols with different parameter settings. (a) Varying k; (b) Varying ℓ; (c) Varying μ; (d) Varying σ.

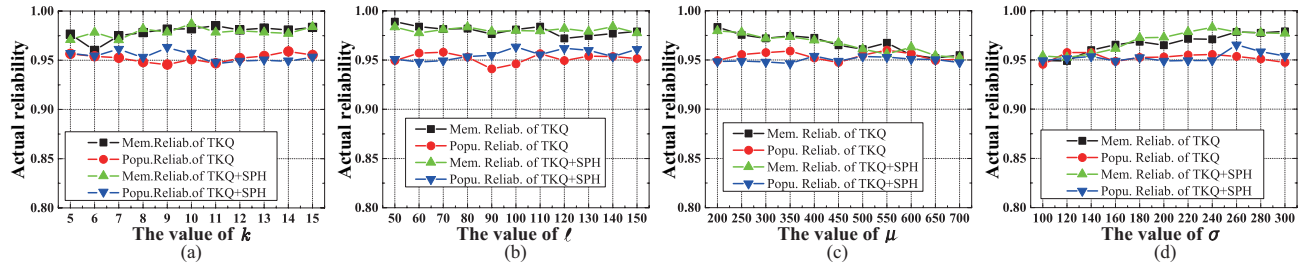

Fig. 8. Investigating the reliability of TKQ and TKQ+SPH with different parameter settings. (a) Varying k; (b) Varying ℓ; (c) Varying μ; (d) Varying σ.

outperforms the benchmark protocols, particularly when k is small. For example, TKQ+SPH runs 7x faster than the fastest existing protocol when $k = 5$.

2) Impact of ℓ: TKQ and TKQ+SPH are the only protocols that have good scalability against the number ℓ of categories. Moreover, TKQ+SPH is the fastest protocol with different settings of ℓ. In this set of simulations, we fixed the number of concerned categories k to 10, and varied the number of all categories ℓ from 50 to 150. Each category size follows the normal distribution $Norm(500, 400)$. As the number ℓ of categories increases, the number of tackled tags also greatly grows. The numerical results in Fig. 7(b) reveal that the execution of each protocol increases more or less with respect to the number ℓ of categories, but the execution time of TKQ and TKQ+SPH increases with the smallest rate among all protocols. The larger the number ℓ of total categories is, the better the time-efficiency TKQ and TKQ+SPH achieve, as compared with other protocols. Since k is fixed, increasing the number of total categories is equivalent to increase the number of unconcerned categories. Fortunately, the unconcerned categories do not require accurate estimation and can be filtered out quickly. Therefore, TKQ and TKQ+SPH are not sensitive to the number of tag categories. We observe that TKQ+SPH is the fastest protocol with different settings of ℓ, *e.g.*, it runs 5x faster than the fastest benchmark protocol when $\ell = 150$.

3) Impact of μ: TKQ+SPH is the fastest protocol with different settings of μ. In this set of simulations, we fixed the number k of concerned categories to 10, and the number ℓ of total categories to 100. Each category size is randomly generated following the normal distribution $Norm(\mu, 150)$, where μ varies from 200 to 700. The simulation results in Fig. 7(c) reveal that the execution time of ART is independent of the category size, and the execution time of TKQ and TKQ+SPH increases slightly as μ increases. The underlying reason is as follows. If we increase μ while fixing the value of σ, the relative variance of category sizes becomes small. Intuitively, it is difficult to determine which one is larger when two categories are of similar sizes. Hence, as μ increases, the execution time of TKQ and TKQ+SPH slightly grows. The execution time of EDFSA increases sharply as μ increases, because much more tags are required to be identified. The execution time of ES also increases sharply as μ increases because the used frame size should be proportional to the total tag population. We observe that ES is suitable for the RFID system that contains a large number of categories, each with a very small category size, *e.g.*, ES runs faster than EDFSA and ART when $\mu = 200$. As μ increases, the performance of ES deteriorates. We observed that even when μ is as small as 200, TKQ+SPH still runs 2.6x faster than ES. Note that, such an improvement will be more significant when μ is larger.

4) Impact of σ: TKQ+SPH is the fastest protocol with different settings of σ. In this set of simulations, we fixed the number of concerned categories k to 10, and the number of total categories ℓ to 100. Each category size is randomly generated following the normal distribution $Norm(500, \sigma)$, where σ varies from 100 to 300. As illustrated in Fig. 7(d), the execution time of Es, TKQ and TKQ+SPH decreases as σ increases. This is because the larger σ is, the larger the variance of category sizes is. Intuitively, it is relatively easy to compare two categories with significantly different sizes. Hence, as σ increases, the execution time of Es, TKQ and TKQ+SPH decreases. The execution time of ART is stable and regardless of the category sizes. And TKQ+SPH is about 3x faster than the fastest benchmark protocol with varying σ.

B. Reliability

Both TKQ and TKQ+SPH are able to satisfy the required reliability in various settings. Besides the time-efficiency, the

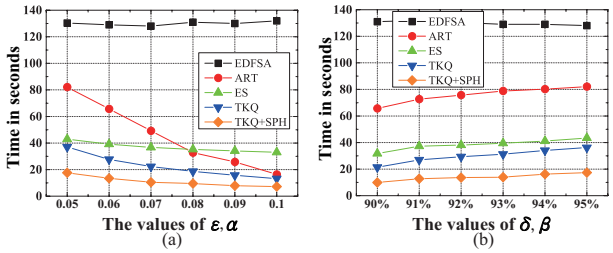

Fig. 9. Time cost vs. query accuracy. (a) Varying ε, α; (b) Varying δ, β.

actual reliability of the proposed protocols is another important performance metric. In this set of simulations, we conduct extensive simulations to evaluate the membership reliability as well as the population reliability of TKQ and TKQ+SPH. Figs. 8(a)∼(d) are plotting using different settings of k, ℓ, μ, or σ to show the impact of these key factors on the actual reliability of TKQ and TKQ+SPH. The simulation results demonstrate that the proposed protocols can meet the required membership and population constraints. Note that, the actual reliability is sometimes a bit lower than the required level. This is a normal phenomenon due to the simulation variance.

C. Time-efficiency vs. Accuracy

TKQ+SPH is the fastest protocol with different accuracy requirements. Figs. 9(a)(b) are plotted using $k = 10$, $\ell = 100$, $\mu = 200$, $\sigma = 150$. In Fig. 9(a), we fixed the error tolerance δ and β to 95%, then varied the reliability ε and α from 0.05 to 0.1. In Fig. 9(b), we fixed ε and α to 0.05, then varied δ and β from 90% to 95%. We observed that the execution time of EDFSA is stable regardless of the required accuracy. Except for EDFSA, the time cost of the other protocols is positively correlated to the error tolerance, and is negatively correlated to the reliability. TKQ+SPH is consistently faster than all the other protocols with various settings of ε, α, δ, and β.

VI. Conclusion

This paper studies the problem of top-k queries in multi-category RFID systems, and makes three key contributions. First, we propose a *Top-k Query (TKQ)* protocol. The key idea is to use the combined signals in homogeneous slots to estimate category sizes. TKQ can quickly eliminate the sufficiently small categories, and only a limited number of large-size categories need accurate estimation. Second, we propose the *Segmented Perfect Hashing (SPH)* scheme to improve the frame utilization of TKQ from 36.8% to nearly 100%. We theoretically maximize the time-efficiency of the proposed protocols. Third, we conduct extensive simulations to evaluate the protocol performance. The simulation results show that TKQ+SPH achieves not only the required accuracy constraints but also a 2.6∼7x speedup over prior protocols.

Acknowledgment

This work is partially supported by the National Science Foundation for Distinguished Young Scholars of China (Grant no. 61225010); the State Key Program of National Natural Science of China (Grant no. 61432002); NSF under Grant no. CNS-1318563, NSFC under Grant nos. of 61173161, 61173162, 61272417, 61472184, 61321491, 61370198 and 61370199, the Jiangsu Future Internet Program under Grant no. BY2013095-4-08, the Jiangsu High-level Innovation and Entrepreneurship (Shuangchuang) Program, and the Fundamental Research Funds for the Central Universities (Grant. DUT15QY20). NSF grants CNS 1461932, CNS 1460971, CNS 1449860, IIP 1439672, and CNS 1156574. Alex X. Liu is also affiliated with the Department of Computer Science and Engineering, Michigan State University, USA.

References

[1] L. Xie, H. Han, Q. Li, J. Wu, and S. Lu, "Efficiently Collecting Histograms Over RFID Tags," *Proc. of IEEE INFOCOM*, 2014.
[2] Y. Zheng and M. Li, "Fast Tag Searching Protocol for Large-Scale RFID Systems," *IEEE/ACM Transactions on Networking*, vol. 21, no. 3, pp. 924–934, 2013.
[3] J. Liu, B. Xiao, K. Bu, and L. Chen, "Efficient Distributed Query Processing in Large RFID-enabled Supply Chains," *Proc. of IEEE INFOCOM*, 2014.
[4] L. Yang, P. Pai, F. Dang, C. Wang, X.-Y. Li, and Y. Liu, "Anti-counterfeiting via Federated RFID Tags' Fingerprints and Geometric Relationships," *Proc. of IEEE INFOCOM*, 2015.
[5] X. Liu, K. Li, H. Qi, B. Xiao, and X. Xie, "Fast Counting the Key Tags in Anonymous RFID Systems," *Proc. of IEEE ICNP*, 2014.
[6] J. Liu, B. Xiao, S. Chen, F. Zhu, and L. Chen, "Fast RFID Grouping Protocols," *Proc. of IEEE INFOCOM*, 2015.
[7] B. Sheng, C. C. Tan, Q. Li, and W. Mao, "Finding Popular Categories for RFID Tags," *Proc. of ACM Mobihoc*, 2008.
[8] S. Lee, S. Joo, and C. Lee, "An Enhanced Dynamic Framed Slotted ALOHA Algorithm for RFID Tag Identification," *Proc. of IEEE MobiQuitous*, 2005.
[9] M. Shahzad and A. X. Liu, "Probabilistic Optimal Tree Hopping for RFID Identification," *Proc. of ACM SIGMETRICS*, 2013.
[10] Y. Zheng, M. Li, and C. Qian, "PET: Probabilistic Estimating Tree for Large-Scale RFID Estimation," *Proc. of IEEE ICDCS*, 2011.
[11] M. Shahzad and A. X. Liu, "Every Bit Counts: Fast and Scalable RFID Estimation," *Proc. of ACM MobiCom*, 2012.
[12] B. Chen, Z. Zhou, and H. Yu, "Understanding RFID Counting Protocols," *Proc. of ACM MobiCom*, 2013.
[13] Y. Hou, J. Ou, Y. Zheng, and M. Li, "PLACE: Physical Layer Cardinality Estimation for Large-Scale RFID Systems," *Proc. of IEEE INFOCOM*, 2015.
[14] C. Qian, H. Ngan, Y. Liu, and L. M. Ni, "Cardinality Estimation for Large-scale RFID Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 9, pp. 1441–1454, 2011.
[15] L. Kong, L. He, Y. Gu, M.-Y. Wu, and T. He, "A Parallel Identification Protocol for RFID Systems," *Proc. of IEEE INFOCOM*, 2014.
[16] W. Gong, K. Liu, X. Miao, Q. Ma, Z. Yang, and Y. Liu, "Informative Counting: Fine-grained Batch Authentication for Large-Scale RFID Systems," *Proc. of ACM MobiHoc*, 2013.
[17] E. Inc, "Radio-frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860 mhz-960 mhz," *EPCGlobal*, Inc, 1.2.0 ed., 2008.
[18] H. Han, B. Sheng, C. C. Tan, Q. Li, W. Mao, and S. Lu, "Counting RFID Tags Efficiently and Anonymously," *Proc. of IEEE INFOCOM*, 2010.
[19] Y. Qiao, S. Chen, T. Li, and S. Chen, "Energy-efficient Polling Protocols in RFID Systems," *Proc. of ACM Mobihoc*, 2011.
[20] O. N. Maire, "Low-cost SHA-1 Hash Function Architecture for RFID Tags," *RFIDSec*, vol. 8, pp. 41–51, 2008.
[21] L. Yang, J. Han, Y. Qi, C. Wang, T. Gux, and Y. Liu, "Season: Shelving Interference and Joint Identification in Large-Scale RFID Systems," *Proc. of IEEE INFOCOM*, 2011.
[22] X. Liu, B. Xiao, K. Li, J. Wu, A. X. Liu, H. Qi, and X. Xie, "RFID Cardinality Estimation with Blocker Tags," *Proc. of IEEE INFOCOM*, 2015.
[23] M. Kodialam and T. Nandagopal, "Fast and Reliable Estimation Schemes in RFID Systems," *Proc. of ACM Mobicom*, 2006.
[24] T. Li, S. Wu, S. Chen, and M. Yang, "Energy Efficient Algorithms for the RFID Estimation Problem," *Proc. of IEEE INFOCOM*, 2010.