LocP: An Efficient Localized Polling Protocol for Large-scale RFID Systems

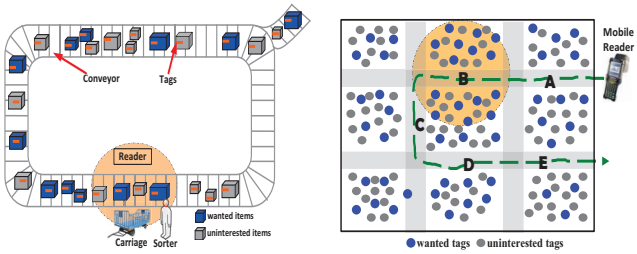
Binbin Li[†‡], Yuan He[*], Wenyuan Liu[‡§], Lin Wang[‡§], Hongyan Wang[†‡]
[†]School of Economics and Management, YanShan University, China
[‡]School of Information Science and Engineering, YanShan University, China
[*]School of Software, TNLIST, Tsinghua University, China
[§]The Key Laboratory for Computer Virtual Technology and System Integration of HeBei Province, China
ysulbb@gmail.com, he@greenorbs.com, {wyliu, wlin, why}@ysu.edu.cn

***Abstract*—RFID systems nowadays are operated at large-scale in terms of both occupied space and tag quantity. One may have prior knowledge of the complete set of tags (denoted by N) and any set of wanted tags (denoted by M) within the complete set, i.e. , $M \subseteq N$. Then here comes an open problem: when one is particularly interested in a subarea of the system, how to collect information (not simply tagIDs) from a wanted subset (denoted by d_M) of the interrogated tags (denoted by d_N) in that subarea? This issue has great significance in many practical applications but appears to be challenging when there is a stringent time constraint. In this work, we first establish the lower-bound of this problem, and show a straightforward polling solution. Then, we propose a novel polling protocol called *LocP*, which consists of two phases: the Tags-Filtering phase and the Ordering-and-Reporting phase. *LocP* employs Bloom Filter twice to significantly reduce the scale of candidate tags in the Tags-Filtering phase. In the Ordering-and-Reporting phase, tags determine their own transmission time-slots according to the allocation vectors iteratively broadcasted by the reader. *LocP* thus achieves a delicate tradeoff between time and polling accuracy. We conduct extensive simulations to evaluate the performance of *LocP*. The results demonstrate that *LocP* is highly efficient in terms of information collection time, leading to convincing applicability and scalability of large-scale RFID systems.**

I. Introduction

Radio Frequency IDentification (RFID) technology has gained popularity with many important applications, such as access control [1][2], object identification [3], inventory management [4], transportation and logistics [5], localization [6][7], and tracking [8][9], and so on. In recent years, there are two new trends in RFID area. One is a trend of applying RFID at large scale in terms of both tag quantity and occupied space. Another trend is the increasing need of management granularity of the RFID information (e.g. SenseID data) [10]. Those trends make query processing an even more complex and challenging problem, in which time efficiency is a critical factor.

It is easy to numerate many large-scale RFID applications with such information collection and management requirements. The first example is a RFID assisted baggage sortation system [8][11] in the airport, as shown in Figure 1(a). Baggages from different check-in counters are carried on a shared conveyor and mixed together. It is important to efficiently identify the baggages of a particular flight among piles of baggages and sort them out efficiently without any missing / extra item. A similar scenario appears in the distribution centers of Amazon and Alibaba, where the numbers of postal parcels are usually hundreds of thousands. During the batch processing, it is important for those tags to report their status after executing a batch command to check whether there are any failures. Also, in large food warehouses, traditional passive tags and sensor-augmented tags (such as WISP [12]) usually coexist. As shown in Figure 1(b), one may carry a mobile reader and walk through the warehouse along a predefined path to collect sensor-produced information from a specified subset of tags. If abnormal temperature is detected, the reader can identify the corresponding items and make prompt reactions to ensure food security.



(a) Baggages sortation in airport (b) Sensor data collection in warehouse

Fig. 1. Two information collection examples in large-scale RFID systems

To tackle information collection in large-scale RFID systems, a number of protocols like *MIC* [13], *ETOP* [14] and *BIC* [15][16] have been proposed. In this paper, we study a new problem, localized polling. We assume that one has prior knowledge of the complete set of tags (denoted by N) and any set of interested tags (denoted by M) within the complete set, i.e., $M \subseteq N$. When he/she is particularly interested in a subarea of the system, how to collect information (not simply tagIDs) from a wanted subset (denoted by d_M) of the interrogated tags (denoted by d_N) in that subarea? This problem is completely different from previous literature shown in Figure 2 and cannot be well resolved by existing protocols, as will be discussed in Section II.

In this paper, we propose a novel solution, called **Loc**alized **P**olling protocol (*LocP*), which consists of two phases: the

978-1-5090-3281-5/16/$31.00 ©2016 IEEE

Tags-Filtering phase and the Ordering-and-Reporting phase. The objective of the first phase is to filter out most of the unwanted tags with two Bloom Filters respectively constructed on the reader and tags. Then the reader allocates each candidate tag to an unique time slot for collecting the tag's information in the second phase. Note that there is a dilemma we have to face. That is, singly reducing the time of any one phase of the two will cause increase in time in another phase. A sophisticated design is needed to enhance the overall polling efficiency. Based on limited prior knowledge, *LocP* sets a delicate tradeoff between these two phases, so that the overall execution time is drastically reduced.

Our contributions can be summarized as follows.

- We are the first to formally define the localized polling problem in large-scale RFID systems, and propose a novel solution *LocP*, which consists of a Tags-Filtering phase and an Ordering-and-Reporting phase to resolve it.
- In the Tags-Filtering phase, when determining the length of second filter vector, the cardinality of alert tags which can pass the test of first filter vector is required. We establish an upper bound for this cardinality, so that a cardinality estimation round is not needed. Meanwhile, using this upper bound, a tradeoff between the two phases of *LocP* is obtained to enhance the overall polling efficiency.
- We conduct extensive simulations with different parameter settings, for the purpose of collecting different amounts of information. The results demonstrate the superior efficiency of *LocP* in almost all the cases, compared with other feasible protocols.

The rest of the paper is organized as follows. We discuss the related work in Section II. In Section III, we give out the system model and formally define the localized polling problem in a large-scale RFID system. We establish the lower bound of this problem, and propose the basic Polling Protocol (*BLP*) in Section IV. We present detailed description of the *LocP* protocol and analysis in Section V. And in Section VI we conduct extensive simulations to evaluate the proposed protocols. We conclude this work in Section VII.

II. Related Work

How to process large quantities of tags has been a hot topic in the RFID community. A school of existing works put their efforts on designing time efficient anti-collision protocols to collect tags' identification (tagIDs). Existing tag identification protocols can be classified into two categories: aloha-based protocols [17] and tree-based protocols [18]. In recent years, several novel identification works exploiting collision [19] [20] and physical layer information [21][22] are proposed to further improve the efficiency of identification.

Driven by the need of utilizing the increasingly rich information from RFID tags, recent studies focus more on information collection, a process to collect not tagIDs but dynamic, real-time information associated with tags, e.g. sensor readings from sensor-augmented tags. To this end, S. Chen *et al.* first put forward this problem in [13] and tackled the issue of collecting information from all the N tags in the interrogation region, with N known in advance, as shown in Figure 2(a). Their proposal called Multi-hash Information Collection protocol (*MIC*) lets the reader schedule tags transmission by broadcasting hash-selection vectors. Aiming at reducing per-tag energy consumption, Y. Qiao *et al.* [14] extend the problem to collect information of M tags out of total N tags as shown in Figure 2(b), and propose two Tag Ordering Polling protocol, *TOP* and *ETOP*. *TOP* and *ETOP* require both N and M be known in advance. Both *MIC* and *ETOP* are only applicable in single reader scenarios. To cope with multi-reader scenarios, H. Yue *et al.* propose *BIC* [15][16], which can efficiently collect information from unknown-subset tags of total N tags as shown in Figure 2(c). In order to improve the time efficiency, a Bloom Filter which is distributively constructed by tags are transmitted to the reader to identify the interrogated tags from N. *BIC* does not assume to know anything about the target tags, and so only can collect information of all the tags in the communication range.

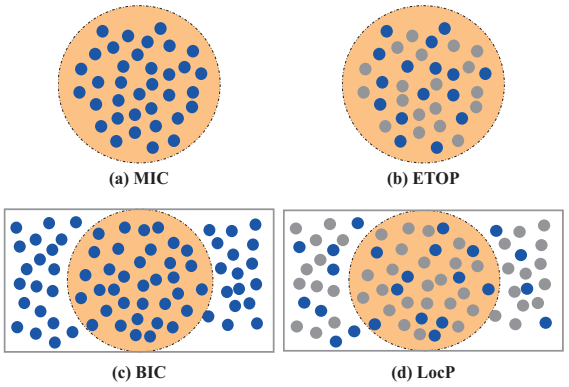


Fig. 2. Problem difference among *MIC*, *ETOP*, *BIC* and *LocP*, where blue circles represent the wanted tags and gray circles represent the unwanted / uninterested tags

In order to efficiently collect information from a subset of wanted tags in a subarea of the whole system, it is essential to approximatively know which wanted tags are located in the subarea. While it is not so easy to achieve this objective with limited priori knowledge of N and M, especially when the quantity of tags is usually large in large-scale RFID systems. Existing protocols cannot effectively address the localized polling problem well. This is indeed an open problem in the community that has great significance in practice. Our work in this paper is to release the assumption of prior knowledge of d_N and d_M, and realize time efficient information collection in large-scale RFID systems.

III. Preliminaries

A. System Model

Consider a large-scale RFID system serves in a large physical space, all the tags can be classified into two categories, namely wanted tags whose information needs to be collected and uninterested tags which can be ignored temporarily. We

assume that all the tagIDs (N) including all the wanted tagIDs (M) in the whole system have been obtained and stored in the backend database in advance [15]. Because of the dynamic, these two kinds of tags may be mixed together and distributed across the whole space, and it is impossible to get a description of the tags' distribution. On the other hand, due to the limited communication range, a reader monitors only a limited subarea of the whole system. Hence, in any subarea, the reader does not have any prior knowledge about the interrogated tags (both d_N and d_M) in the communication range.

The communication model between the reader and tags is *Reader-Talks-First* and bit- or time-slotted, which follows the EPCglobal C1G2 standard [23] in general. The reader initializes the communication by sending out a request message and several parameters, such as frame size, number of hash functions, random seeds, etc. Once receiving a request, each tag selects one or several slots to transmit its data in the following bit- or time-slotted frame, according to the results of hash functions. The clocks of tags are synchronized by the reader's signal.

B. Problem Definition

Assuming there are total n tags in the whole RFID system, denoted by N, i.e. $n = |N|$. Let M be the wanted subset of N, which needs to collect information in the whole system, and $m = |M|$. So we have $M \subseteq N$ and $m \leq n$. As aforementioned, both N, M and n, m can be got easily according to the backend database. However, when coming to a particular subarea of the whole system, neither tags in the reader's communication range (d_N) nor the wanted tags whose information needs to be collected in current subarea (d_M) are unknown. Similarly, we let $d_n = |d_N|$, $d_m = |d_M|$ and $d_M \subseteq d_N$. So there must be $d_M \subseteq d_N \subseteq N$, $d_M \subseteq M \subseteq N$, $d_m \leq d_n \leq n$ and $d_m \leq m \leq n$.

Our objective is to design a time efficient localized polling protocol to collect information from d_m wanted tags (d_M) out of d_n interrogated tags (d_N) in a subarea of the whole system, where both d_M and d_N are unknown, but all the tags (N) and all the wanted tags (M) in the whole system are known in advance. Note that the information collection here means not only getting the set of information simply, but also mapping all the information to corresponding tags, so that we can associate the information with corresponding items quickly.

The time efficiency performance, i.e. the minimization of protocol execution time is the most primitive objective in this paper. There are two reasons as follows: First, for a large-scale RFID system, the number of RFID tags may be come to millions scale, such as 2 millions tags in a large supermarket. So the polling protocol should guarantee both efficiency and scalability. Second, the communication rate between reader and tags is low, so the communication overhead between the reader and tags, namely the overall protocol execution time should be minimized.

IV. Baseline Solutions

A. Lower Bound

Intuitively, the lower bound to collect information of d_m wanted tags in a subarea is $d_m \times t_{inf}$, where t_{inf} represents the time for a tag to transmit its information. However, this bound can never be achieved because of two reasons. First, the reader does not know d_M or d_m at all. Second, RFID network is a typically centralized network, tags cannot hear each other's transmission, so there must be collisions at the reader side. To avoid these collisions, the only method is to schedule the tags' transmission at the reader side, and let the reader broadcast the scheduling result to all the tags. So the overall execution time of localized polling must be larger than the lower bound. Nevertheless, it offers a benchmark to evaluate the performance of our protocols.

B. Basic Localized Polling Protocol (BLP)

The straightforward way to resolve the localized polling problem is to let the reader broadcast the tagIDs in d_M one by one. After it broadcasts a tagID, it waits for a time-slot of t_{inf} to receive the information. When receiving a tagID, each tag compares the received tagID with its own tagID. If it is matched, the tag transmits its information and does not participate in the rest of protocol again. Otherwise, it keeps silent. However, the reader does not have any priori knowledge about d_M. A feasible inferior scheme is to let the reader broadcast the tagIDs in M one by one rather than d_M.

In this way, the reader totally needs to broadcast m tagIDs. No matter whether the request tag is in the communication range, the reader has to waiting for a time-slot to receive information. So the overall time cost is $m \times (t_{tag} + t_{inf})$. This can be time-consuming, especially when m is large but d_m is small. We regard it as a Basic Localized Polling protocol (*BLP*), a much more efficient polling protocol should take less time than *BLP*.

V. Efficient Localized Polling Protocol Design

In this section, we present the detail of *LocP*, which employs Bloom Filters twice to filter out most uninterested tags and allocation vectors to schedule the candidate tags' information transmission in a proper order.

A. Protocol Overview

When coming to a particular subarea, the reader does not have any prior knowledge about the interrogated tags (d_N) and the wanted tags (d_M) which need to be collected information. Fortunately, the reader can get all the tags (N) and all the wanted tags (M) from the backend database easily. It is time-consuming to get the accurate d_M by identifying tags in d_N one by one, due to the inefficiency of identification protocols. Even the BLP which broadcasts the tagIDs in M one by one is not a good choice as verified later. Similar to existing approaches [13][14][15], to reduce the overall execution time, we tend to let the reader schedule the tags' transmission order after filtering out most uninterested tags. The whole procedure of *LocP* is showed in Figure 3.

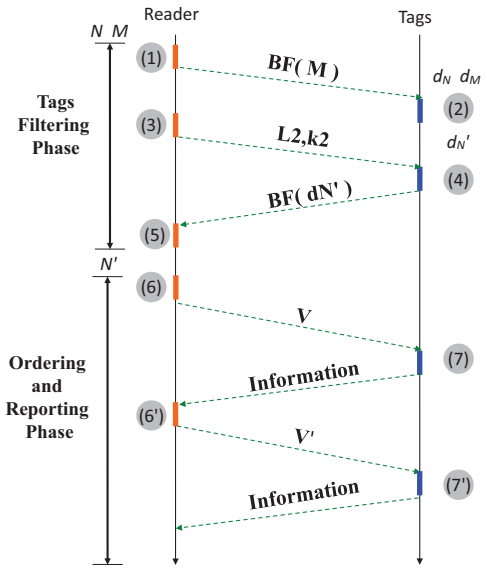


Fig. 3. The overview of our *LocP*

LocP consists of two phase, a Tags-Filtering phase and an Ordering-and-Reporting phase. The purpose of former phase is to filter out most uninterested tags and reduce the number of candidate tags to a much smaller scale with two Bloom Filters which are constructed at reader and tags side respectively. According to the property of Bloom Filter, we can guarantee that all the wanted tags in d_M must be in the final candidate tags. In the second ordering and reporting phase, the reader allocates each candidate tag to an unique time-slot by broadcasting allocation vectors, information of candidate tags is collected round by round until all the information has been collected.

Obviously, the overall execution time of *LocP* is the summation of Tags-Filtering phase and Ordering-and-Reporting phase. And the procedure of first phase is quite similar with *CATS* [4] and *ITSP* [24], which aim at minimizing the length of filter with the guarantee of tolerable false positive. If we directly employ these tag search protocols in the first phase, the time of first phase can be minimized, but there will be more candidate tags and thus longer execution time in the second phase, so the overall execution time does not necessarily be minimized. On the other hand, increasing the length of filter can reversely decrease the false positive probability [25], and then reduce the number of candidate tags. While a continual increasing may has insignificant contribution in reducing the overall execution time of *LocP* when the length has been large enough. Hence, there is a dilemma as showed in Table I.

Therefore, a time tradeoff between the two phase of *LocP* is needed for the purpose of reducing the overall execution time. Next, we first show the basic *LocP* without optimization (i.e. calculate the length of two filters with known m and n respectively). Then we try to establish an upper bound of the cardinality of tags which can pass the test of first filter $BF(M)$. Then we can calculate the length of second filter in

TABLE I
THE DILEMMA IN LOCALIZED POLLING

First Phase		Second Phase		Execution Time
filter	time	candidate tags	time	
shorter	shorter	more	longer	longer
longer	longer	less	shorter	longer

the Tags-Filtering phase with this upper bounder, rather than the large n in *LocP* with optimization. In such a way, not only the cardinality estimation round before the construction of the second filter can be canceled, but also can improve the polling efficiency by significantly reducing the length of second filter, even if this may lead to a slightly increase of the number of candidate tags in the Ordering-and-Reporting phase.

B. Tags Filtering Phase

Since the wanted tags in d_M which need to transmit information in current subarea must belong to M, i.e. there must be $d_M \subseteq M$, we could use the Bloom Filter $BF(M)$ which is constructed with the known M to filter out the most tags which do not belong to M, and can guarantee all the wanted tags d_M are included.

To get $BF(M)$, the reader maps all the wanted tags in M into an L_1 bits vector using K_1 independent hash functions with random seeds S_1. Then the reader broadcasts the $BF(M)$, together with L_1, K_1 and S_1 to all the tags in d_N.

When receiving $BF(M)$, L_1, K_1 and S_1, each tag in d_N uses the same K_1 hash functions, the random seeds S_1 and their own tagIDs to check whether the all K_1 bits in $BF(M)$ are '1's. If any bit of the K_1 bits is '0', the tag cannot pass the test of $BF(M)$, it means that this tag must not be an element of M. Hence, it also must not be an element of d_M because $d_M \subseteq M$. So it should keep silent and does not participate in the rest of protocol. Otherwise, if all the K_1 bits are '1's, the tag passes the test of $BF(M)$, and keeps alert to continue participate in the following steps.

Because $d_M \subseteq M$, according to the property of Bloom Filter, all the tags in d_M can pass the test of $BF(M)$. On the other hand, there may be false positives in Bloom Filter, so a tag in d_N but not in d_M may also pass the test. In a word, the tags which can pass the test of $BF(M)$ include all the tags in d_M and the tags which can also pass the test in d_N but not in d_M, we denote all these tags as d'_N. So we have $d_M \subseteq d'_N \subseteq d_N$.

According to [25], given the number of all wanted tags m and the false positive probability p, the optimal length L_1 of Bloom Filter $BF(M)$ and the number of hash functions K_1 can be calculated as

$$L_1 = -\frac{m \times \ln p}{(\ln 2)^2} \tag{1}$$

$$K_1 = \frac{L_1}{m} \ln 2. \tag{2}$$

But there is still a problem, as the tags in d'_N merely know they may be a member of M locally, the reader still know

nothing about the d'_N. Obviously, it is not a good idea to collect all the tagIDs with an identification process. So we employ the Bloom Filter again, the only difference is that previous $BF(M)$ is constructed at the reader side, while the new one is distributively constructed by the tags in d'_N.

In order to get the information of d'_N, the reader first constructs an L_2 bits vector $BF(d'_N)$, all the bits are initialized to be '0's. Then the reader broadcasts a request message to start a new frame, which contains three parameters L_2, K_2, and S_2. L_2 is the length of the Bloom Filter, K_2 is the number of hash functions, which can be calculated using the known n as [15] did. So,

$$L_2 = -\frac{n \times \ln p}{(\ln 2)^2}, \tag{3}$$

K_2 is the number of hash functions can be calculated as

$$K_2 = \frac{L_2}{n} \ln 2. \tag{4}$$

And S_2 is a set of random seeds used with the hash functions.

Once receiving the request message, each tag in d'_N also constructs a L_2 length vector which is also initialized to '0's. Each tag calculates K_2 hash values with random seeds in S_2 and its own tagID using K_2 hash functions independently. And assigns the corresponding K_2 bits in the vector to '1's. After having generated the vectors, all the tags in d'_N start to transmit their vector simultaneously. If the binary is '0', the tag does not transmit anything, and if it is a '1', the tag transmits a short signal in the physical layer. The reader only needs to sense the physical channel, in the ith bit-slot, if the channel is idle, it means that there are no tag transmit in this bit-slot, the reader sets the ith bit to '0' in the $BF(d'_N)$. On the contrary, if the channel is busy, which means more than one tags transmit signals, the reader sets the ith bit to '1'. After L_2 bit-slots, the reader can get a new Bloom Filter vector $BF(d'_N)$.

Then the reader uses the $BF(d'_N)$ to test all the tags in N. As the reader knows all the parameters including L_2, K_2 and S_2, for each tag in N, the reader calculates K_2 hash values as the tags do. If any bit of the K_2 bits is '0', the tag cannot pass the test of $BF(d'_N)$. Otherwise, if all the bits are '1's, this tag passes the test. We take all the tags which can pass the test of $BF(d'_N)$ as the candidate tags, denoted as N'. According to property of Bloom Filter, we have $d_M \subseteq d'_N \subseteq N'$. Combining with $d_M \subseteq M$, we can get that $d_M \subseteq (M \cap N')$.

C. Ordering and Reporting Phase

So far, the reader has obtained the candidate tags N', we can start to collect information from them. To avoid collisions, the whole process consists of several rounds. Algorithm 1 regulates the behavior of the reader in this phase.

In each round, we let U be the subset wanted tags which have not completed their information transmission, and initially $U = M \cap N'$. And let CN be the subset tags of N' which are allocated to current round, CU be the wanted tags allocated to this round. The reader first maps all the tags in N' to an allocation vector V whose length is set to $|N'|$. If multiple tags allocate to the same bit, it means that these tags will cause collisions, so the reader keeps the corresponding bit to '0', and let these tags keep silent in this round, and participate in the next rounds. On the contrary, if only one tag hashes to a particular bit, which means this tag is allowed to transmit its information to the reader in this round, so set the bit to '1' and adds it from CN, particularly, if it also is a member of M, then add it to CU.

Algorithm 1 Ordering and Reporting Algorithm for reader

```
R ← φ
U ← M ∩ N'
while |U| > 0
    Generate V and get CN
    CU ← M ∩ CN
    Broadcast V and the random seed s
    while |CU| > 0
        if there is a response in a time-slot then
            t ← Identification the corresponding tag in CN
            Save the information
            Delete t from U and CU
            Add t to R
        end if
    end while
    Send out a command to terminate this round
    Update N' ← N' − CN
    U ← M ∩ N'
end while
```

After getting V, the reader then sends out a request message together with the allocation vector V and a random seed s, and then waits for the response in the next time-slotted frame. If the length of V exceeds the length of tagID, the reader can divide V into several segments and transmit the segments one by one in a time-slot of t_{id} to all the tags.

When receiving the request message, each tag calculates the position of its indicator bit in V using the random seeds s and the same hash function exploited by the reader. If the bit is '0', the tag realizes that it should not participate in this round to avoid collisions. While if the bit is '1', the tag realizes that it is the right round it should transmit its information. To know which time-slot to respond, the tag examines how many '1's preceding to its own indicate bit. If there are total i '1's preceding its indicate bit in V, the tag will reporting its information in $(i+1)$th time-slot of current round.

Once receiving information in any time-slot, the reader can identify the corresponding tag according to the transmission slot quickly, because the reader knows the expected transmission order according to V. After saving the information, we can delete it from U and CU, and add it to R, which turns out to be d_M at the end of *LocP*. When CU turns out to be empty, which means all the wanted tags allocated to this round have completed their information transmission, the reader sends out a command to terminate this round. The process repeat until the U is empty, which means all the wanted tags have completed their information transmission. In such way, we not

only can get all the information of d_m wanted tags in current subarea, but also can get the d_M efficiently.

D. Parameters Optimization

With a given false positive probability p, the parameters of L_1 and K_1 have been optimized according to Equation 1 and 2, because the reader knows the accurate value of m. While the L_2 and K_2 calculated with a large n according to Equation 3 and 4, have the gap to be improved. As aforementioned, although a longer L_2 can reduce the number of candidate tags and thus the execution time of the second phase of *LocP*, while it does not necessarily reduce the overall execution time of *LocP*. To reduce the overall execution time, a feasible method is letting the reader perform a cardinality estimation for d'_N, while this may also be time-consuming. Instead of a time-consuming cardinality estimation round, we try to establish an upper bound of the cardinality of d'_N (denoted as $\hat{n}$) with the known $BF(M)$.

In order to illustrate the principles, we assume that there is a rough estimation round, which is unneeded actually. The purpose of the estimation round is to get a rough estimation about $\hat{n}$ with a new Bloom Filter vector BF. And the construction procedures of BF is same as $BF(d'_N)$, the length of BF is set to $w = L_1$, the number of hash functions is $k = K_1$.

THEOREM 1. *In BF, which is constructed by all the tags in N using w and k, the probability arbitrary ith bit to be '0' is*

$$Pr\{BF(i) = 0 | i \in [1, w]\} = e^{-\lambda}, \tag{5}$$

and correspondingly,

$$Pr\{BF(i) = 1 | i \in [1, w]\} = 1 - e^{-\lambda}, \tag{6}$$

where $\lambda = \frac{k\hat{n}}{w}$.

Proof. Assuming that all the hash functions used by the $\hat{n}$ tags are following a uniform distribution, so the probability a hash function H of a tag selects the ith bit-slot is

$$Pr\{H(\cdot) = i | i \in [1, w]\} = \frac{1}{w}.$$

Then, the probability of arbitrary bit i in BF to be '0' is

$$Pr\{BF(i) = 0\} = (1 - \frac{1}{w})^{k\hat{n}},$$

which means all the $k\hat{n}$ hash functions of $\hat{n}$ tags have not selected the ith bit-slot. Using the approximation of

$$\lim_{x \to \infty} (1 - \frac{1}{x})^x = e^{-1},$$

the above equation can be simplified as

$$Pr\{BF(i) = 0\} = (1 - \frac{1}{w})^{k\hat{n}} \approx e^{-\frac{k\hat{n}}{w}} = e^{-\lambda}.$$

Correspondingly, the probability $BF(i)$ to be '1' which means more than one tag response in the ith bit-slot is

$$Pr\{BF(i) = 1\} = 1 - Pr\{BF(i) = 0\} \approx 1 - e^{-\lambda}.$$

□

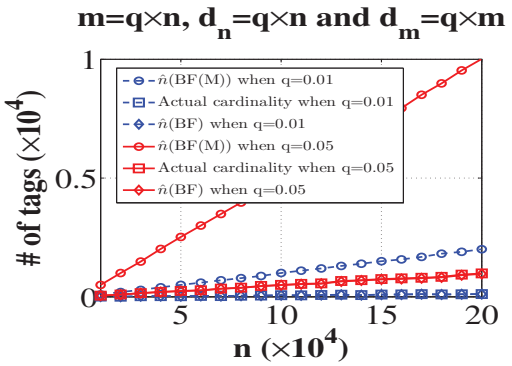


Fig. 4. The comparison between Estimation result and Actual cardinality in d'_N

THEOREM 2. *Let $\rho = \frac{1}{w}\sum_{n=1}^{w} X(i)$ be the ratio of '1's in the vector BF, where $X(i)$ denotes the ith observation in BF, then the cardinality of d'_N can be calculated as*

$$\hat{n} = -\frac{w \ln(1 - \rho)}{k}. \tag{7}$$

Proof. We define X as a random variable which takes value 0 with probability $e^{-\lambda}$ and value 1 with probability $1 - e^{-\lambda}$, namely,

$$Pr\{X = 0\} \approx e^{-\lambda}, Pr\{X = 1\} \approx 1 - e^{-\lambda}.$$

Obviously, the random variable X follows the Bernoulli distribution. Therefore, the expectation of X is as follows

$$E(X) = 1 - e^{-\lambda}.$$

Besides, assuming that the trails of $X_i (1 \leq i \leq w)$ are i.i.d, then we have $E(\rho) = E(X)$. According to the law of large numbers, when w is large enough we have

$$\rho = E(\rho) = E(X) = 1 - e^{-\lambda}.$$

So we can estimate λ as follows

$$\lambda = -\ln(1 - \rho).$$

So that, the observation of ρ in BF can be used to estimate the tag cardinality as follows

$$\hat{n} = -\frac{w \ln(1 - \rho)}{k}.$$

Obviously, Equation 7 is a monotonically increasing function with ρ. □

THEOREM 3. *Let $BF(M)$ be the Bloom Filter vector constructed with M, and BF be the Bloom Filter vector constructed by the tags in d'_N which can pass the test of $BF(M)$ using the same parameters as $BF(M)$'s, then $\rho(BF) \leq \rho(BF(M))$.*

Proof. This theorem can be proved using contradiction easily. □

THEOREM 4. *Let $\hat{n}(BF)$ denote the cardinality estimated using the received BF, which is a projection of the actual cardinality of d'_N, and $\hat{n}(BF(M))$ denotes the cardinality estimated using the known $BF(M)$, then $\hat{n}(BF) \leq \hat{n}(BF(M))$.*

Proof. This theorem can be proved according to Equation 7 and Theorem 3. □

In other words, we have $\hat{n} \approx \hat{n}(BF) \leq \hat{n}(BF(M))$. We then conduct a simple simulation to verify the relation between the actual cardinality of d'_N, $\hat{n}(BF)$ and $\hat{n}(BF(M))$ with eight different settings, Figure 4 plots two of them. From Figure 4, we can get that the estimation results with BF could reflect the actual cardinality well, and the estimation results with $BF(M)$ are always larger than both the actual number of tags and the estimation with BF in the all cases.

Based on the above analysis, for the purpose of reducing the overall execution time, a time tradeoff between the two phases of *LocP* is introduced, we use the rough estimation of $\hat{n}(BF(M))$ to calculate the length of $BF(d'_N)$ and the number of hash functions, rather than $\hat{n}(BF)$, even if $\hat{n}(BF)$ is much closer to the actual $\hat{n}$. Thus, the L_2 and K_2 of $BF(d'_N)$ can be calculated as follows,

$$L_2 = -\frac{\hat{n}(BF(M)) \times \ln p}{(\ln 2)^2}, \tag{8}$$

and

$$K_2 = \frac{L_2}{\hat{n}(BF(M))} \ln 2. \tag{9}$$

In this way, we can not only cut down the communication overhead to estimate the cardinality of d'_N, but also reduce the actual false positive probability (denoted as p' and $p' \leq p$) of $BF(d'_N)$ to a much smaller level by slightly increasing the length of Bloom Filter vector L_2.

E. Time Analysis

The overall execution time of *LocP* is the summation of the time for tags-filtering and ordering-and-reporting. For the purpose of clarity, we ignore the transmission cost of the configuration parameters including L_1, K_1, S_1, L_2, K_2 , S_2 in the first phase and s in the second phase, which normally take several bytes to encode. So execution time mainly consists of four parts, i.e. the time for the reader to broadcast $BF(M)$ (denoted by T_1) and the time for the tags to construct $BF(d'_N)$ (denoted by T_2) in the first phase, the time for the reader to broadcast the allocation vectors (denoted by T_3) and the time for the candidate tags to transmit their information (denoted by T_4) in the second phase.

According to Equation 1, $T_1 = -\frac{m \times \ln p}{(\ln 2)^2} \times Tbit_{R \to T} = -\frac{m \times \ln p}{96 \times (\ln 2)^2} \times t_{id}$, where $Tbit_{R \to T}$ is the time for the reader to transmit a bit, and t_{id} is the time for the reader to transmit a tagID. For T_2, as the expected value of $\hat{n}(BF(M))$ is m, so the expected value of $T_2 = -\frac{m \times \ln p}{(\ln 2)^2} \times Tbit_{T \to R} = -\frac{m \times \ln p}{32 \times (\ln 2)^2} \times t_{inf32}$, where $Tbit_{T \to R}$ is the time for the tag to transmit a bit, and t_{inf32} is the length of a time-slot which allows a tag to transmit a 32 bits information.

The expected cardinality of the set d'_N can be calculated as

$$d_m + p \times (d_n - d_m).$$

So the expected cardinality of N' (denoted by $E(N')$) is

$$E(N') = d_m + p \times (d_n - d_m) + p' \times [n - (d_m + p \times (d_n - d_m))].$$

As $p' \leq p$, so the upper bound of $E(N')$ can be calculated as

$$\begin{aligned} E(N') =& d_m + p \times (d_n - d_m) \\ &+ p' \times [n - (d_m + p \times (d_n - d_m))] \\ \leq& d_m + p \times (d_n - d_m) \\ &+ p \times [n - (d_m + p \times (d_n - d_m))] \\ =& (1-p)^2 \times d_m + p(1-p) \times d_n + p \times n. \end{aligned}$$

According to [14], the expected number of indicator bits in the all allocation vectors for each tag is e. Thus the expected time for the reader to broadcast all the allocation vector $T_3 = \frac{e \times E(N')}{96} \times t_{id}$. Since each tag in N' is allocated an unique time-slot to report its information to the reader, the total number of time slots is $E(N')$. Then, $T_4 = E(N') \times t_{inf32}$.

Based on the above analysis, the expected execution time of *LocP*, denoted as T, can be calculated as follows:

$$\begin{aligned} T =& -\frac{m \times \ln p}{96 \times (\ln 2)^2} \times t_{id} \\ &- \frac{m \times \ln p}{32 \times (\ln 2)^2} \times t_{inf32} \\ &+ \frac{e \times E(N')}{96} \times t_{id} \\ &+ E(N') \times t_{inf32}. \end{aligned}$$

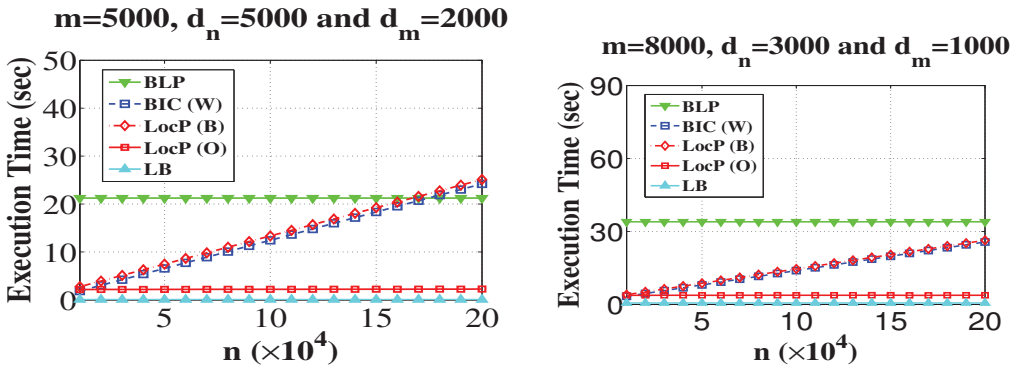
When $n = 10,000$, $m = 8,000$, $d_n = 3,000$, $d_m = 1,000$, and $t_{id} = 3,927\mu$s, $t_{inf32} = 604\mu$s according to Section VI, so the lower bound of execution time is 0.604s, and the execution time of BLP is 33.98s. If the false positive probability of Bloom Filter p is set to 0.05, the upper bound of $E(N')$ is 6,045, so the upper bound of LoP's execution time is 9.6017s, while the actual execution time as showed in Section VI is only 3.79s. This demonstrates a high efficiency of the proposed *LocP*.

VI. PERFORMANCE EVALUATION

In this section, we evaluate the performance of *LocP*. We take the *LocP* without optimization as *LocP(B)*, the *LocP* with optimization as *LocP(O)*. As *BIC* may be available in the worst case when treats all the interrogated tags as wanted tags which have information to transmit. So we compare the execution time of *LocP(B)*, *LocP(O)* with *BLP*, *BIC (worst case)* as *BIC(W)* and the *Lower Bound* as *LB* in this paper.

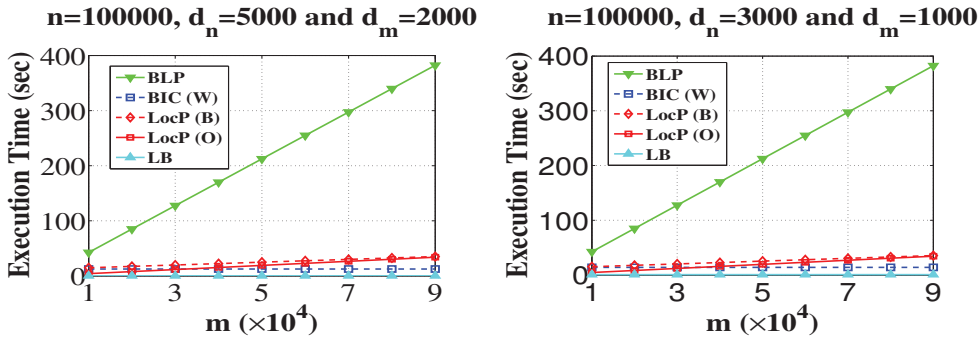
A. Simulation Setting

The simulation setting is following the C1G2 standard [23]. Any two consecutive transmission from the reader to tags or vice versa are separated by a waiting time of 302μs, i.e. $t_{ver} = 302\mu$s. The transmission rate from the reader to tags is 26.5Kb/s, it takes 37.76μs to transmit 1 bit. If the length of tagID is 96 bits, it totally takes $3{,}927\mu$s for the reader to



(a) Information collection with 1 bit information (b) Information collection with 32 bits information

Fig. 5. Performance Comparison under different N



(a) Information collection with 1 bit information (b) Information collection with 32 bits information

Fig. 6. Performance Comparison under different M

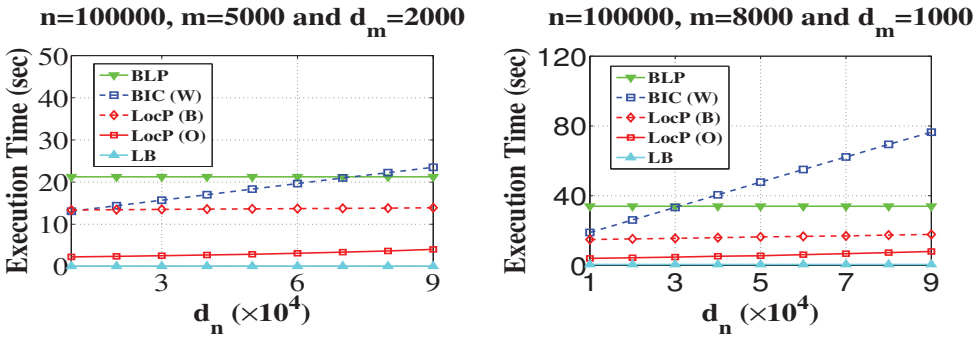
broadcast a tagID, i.e., $t_{id} = 3,927\mu$s. The rate from a tag to the reader is 53Kb/s, it takes 18.88μs for a tag to transmit 1 bit, $t_{inf1} \approx 18.88\mu$s. So the time to transmit 32 bits information $t_{inf32} \approx 604\mu$s. The false positive probability of Bloom Filter is set to 0.05 in all simulations, unless otherwise specified. And for each simulation, we set two different scenarios for the purpose of transmitting 1 bit and 32 bits information respectively.

B. Performance comparison

1) Performance comparison under Different N: We first investigate the performance of *LocP* under different number of N. We set the first scenario with the parameters $m = 5,000$, $d_n = 5,000$ and $d_m = 2,000$. The second scenario with a set of different parameters, $m = 8,000$, $d_n = 3,000$ and $d_m = 1,000$. In both scenarios, we let the n vary from 10,000 to 200,000. Figure 5 plots the performance of all the protocols in the first scenario. In Figure 5(a), because both m and d_m are fixed, so the execution times of *BLP* and the *LB* also keep unchanged. And the execution times of *BIC(W)* and *LocP(B)* increase synchronously with the n increase, and even exceed the time of *BLP* in the most cases. However, the execution times of *LocP(O)* are always within 13 seconds, and almost do not change with different input of n. The same patten also appears in the second scenario as shown in Figure 5(b), when the information length is 32 bits.

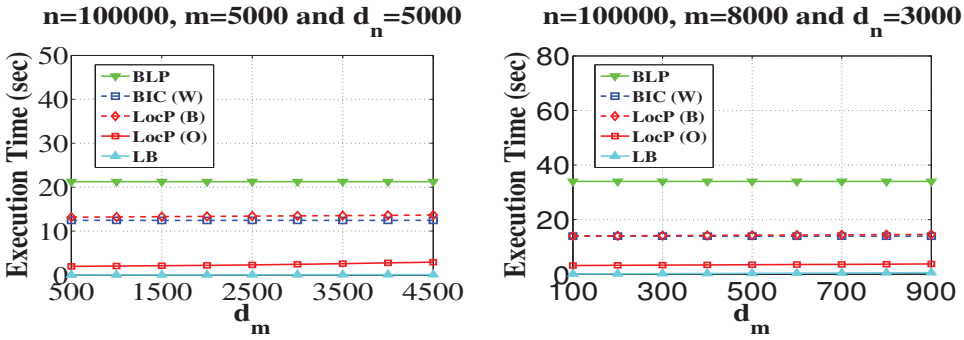
2) Performance comparison under Different M: Then we compare the performance of all the protocols with different number of all wanted tags M. The parameters of the two scenario are set to $n = 100,000$, $d_n = 5,000$ $d_m = 2,000$ and $n = 100,000$, $d_n = 3,000$ $d_m = 1,000$ respectively, and let the value of m vary from $10,000$ to $90,000$. The simulation results are shown in Figure 6. As the increase of m, both the execution time of *LocP(B)* and *LocP(O)* increase. And sometimes, the execution times of *LocP(B)* and *LocP(O)* are even slightly greater than *BIC(W)*'s. However, the time difference is little. The reason is both the length of $BF(M)$ and $BF(d'_N)$ have the relationship with m, it is time-consuming to transmit these two vectors when m is large while both d_n and d_m keep unchanged.

3) Performance comparison under Different d_N: We then investigate the performance of *LocP* under different number of interrogated tags d_N at current subarea. Figure 7 plots the simulation results of two scenarios, where $n = 100,000$, $m = 5,000$, $d_m = 2,000$ and $n = 100,000$, $m = 8,000$, $d_m = 1,000$ respectively. The *BIC(W)* performs poorly, and the execution times even exceed *BLP*'s in most cases. While the *LocP(B)*, especially the *LocP(O)* works well in all the cases, and the growth of execution time is fairly flat.



(a) Information collection with 1 bit information (b) Information collection with 32 bits information

Fig. 7. Performance Comparison under different d_N



(a) Information collection with 1 bit information (b) Information collection with 32 bits information

Fig. 8. Performance comparison under different d_M

4) Performance comparison under Different d_M: Furthermore, we feed *LocP* with different number of wanted tags d_M, and fix $n = 100,000$, $m = 5,000$, $d_n = 5,000$ and $n = 100,000$, $m = 8,000$, $d_n = 3,000$ in the two scenario respectively. As showed in Figure 8(a), when the number of wanted tags d_M is 4,500, the execution time of *BIC(W)* is 202.32s, and the time of *LocP(B)* is 203.38s which is slightly larger than *BIC(W)*. The execution time of *LocP(O)* is 14.78s, and the *LB* is about 1.44s. When the d_m is smaller, *LocP(O)* gets a better performance as showed in Figure 8(b).

C. Protocol Insights

The above results demonstrate a high efficiency of the proposed *LocP*. In the next, we investigate in details the impact of N, M, d_N and d_M. We adopt one of previous simulation settings to obtain the number of tags d'_N and N' both in *LocP(B)* and *LocP(O)*, and the length of two Bloom Filter vectors of $BF(M)$ and $BF(d'_N)$.

Figure 9 compares the number of tags d'_N which can pass the test of $BF(M)$, and the number of candidate tags N' which can pass the test of $BF(d'_N)$ in *LocP(B)* and *LocP(O)*. As showed, the number of d'_N and N' in *LocP(B)* fit well with little deviation in all the cases. The reason is we calculate the length of $BF(d'_N)$ with n in *LocP(B)*, so the false positive probability is small. With a much smaller parameter input in *LocP(O)*, the length of $BF(d'_N)$ will also be smaller, and there will be more tags in candidate tags N' than *LocP(B)*. However, the gap of N' between *LocP(B)* and *LocP(O)* is small as showed in Figure 9. When coming to the length of Bloom Filter vector as showed in Figure 10, the length of $BF(d'_N)$ in *LocP(B)* is always much larger than the one in *LocP(O)*.

Taken together, we can get that comparing with *LocP(B)*, the *LocP(O)* improves the time efficiency performance by significantly reducing the length of response Bloom Filter vector $BF(d'_N)$, which may lead to a slightly increase of the number of candidate tags N'.

VII. Conclusions

This paper presents *LocP*, an efficient localized polling protocol to collect information from a wanted subset of interrogated tags in large-scale RFID systems. *LocP* first uses Bloom Filter twice to significantly reduce the number of candidate tags in the Tags-Filtering phase. Then in the Ordering-and-Reporting phase, *LocP* let the reader schedule tags' transmission by broadcasting allocation vectors round by round until all the information has been collected. The extensive simulation shows that *LocP* is highly efficient in terms of both time efficiency and transmission overhead.

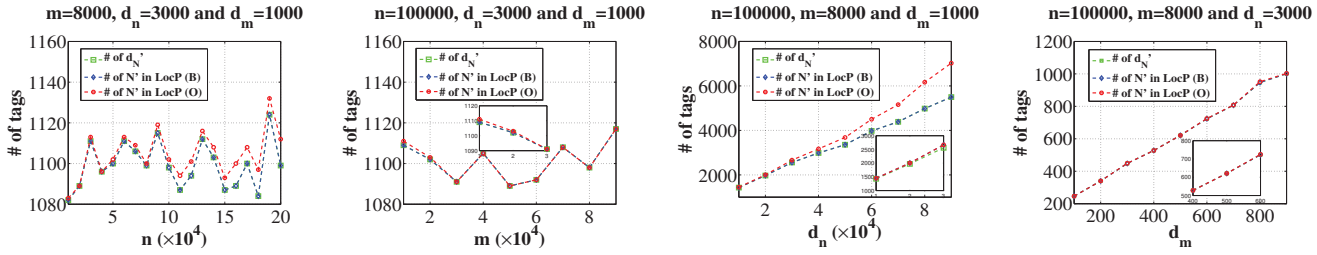


Fig. 9. Comparison of the number of tags under different parameters settings in 32 bits information collection

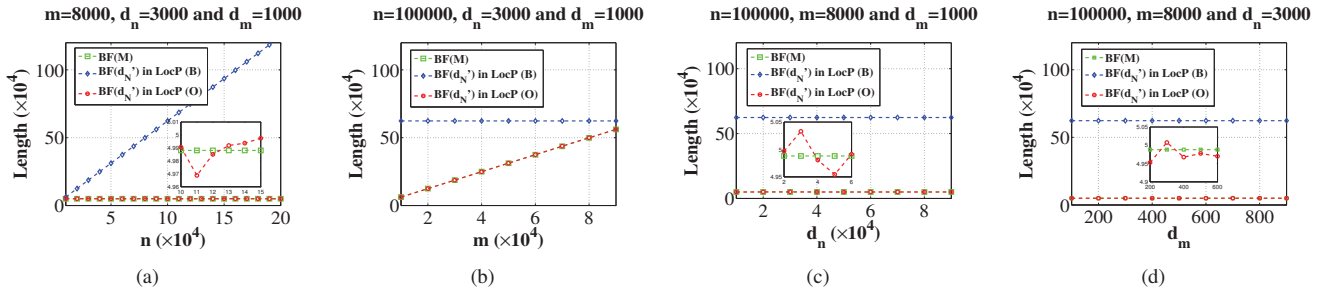


Fig. 10. Comparison of the length of Bloom Filter vector under different parameters settings in 32 bits information collection

ACKNOWLEDGMENT

This work is supported in part by National Natural Science Foundation of China (NSFC) (No.61272466, No.61303233, No. 61373181 and No. 61672448), National Science Fund for Excellent Young Scientist (No. 61422207) and Natural Science Foundation of HeBei Province (No. F2014203062 and No. G2015203242).

REFERENCES

[1] W. Gong, K. Liu, X. Miao, Q. Ma, Z. Yang, and Y. Liu, "Informative counting: fine-grained batch authentication for large-scale rfid systems," in *Proceedings of ACM MobiHoc*, 2013.

[2] W. Gong, Y. Liu, A. Nayak, and C. Wang, "Wise counting: fast and efficient batch authentication for large-scale rfid systems," in *Proceedings of ACM MobiHoc*, 2014.

[3] H. Vogt, "Efficient object identification with passive rfid tags," in *Pervasive Computing*, pp. 98–113, Springer, 2002.

[4] Y. Zheng and M. Li, "Fast tag searching protocol for large-scale rfid systems," *IEEE/ACM Transactions on Networking (TON)*, vol. 21, no. 3, pp. 924–934, 2013.

[5] S. Qi, Y. Zheng, M. Li, L. Lu, and Y. Liu, "Collector: A secure rfid-enabled batch recall protocol," in *Proceedings of IEEE INFOCOM*, 2014.

[6] L. M. Ni, Y. Liu, Y. C. Lau, and A. P. Patil, "Landmarc: indoor location sensing using active rfid," *Wireless networks*, vol. 10, no. 6, pp. 701–710, 2004.

[7] J. Wang and D. Katabi, "Dude, where's my card?: Rfid positioning that works with multipath and non-line of sight," in *Proceedings of ACM SIGCOMM*, 2013.

[8] L. Yang, Y. Chen, X.-Y. Li, C. Xiao, M. Li, and Y. Liu, "Tagoram: Real-time tracking of mobile rfid tags to high precision using cots devices," in *Proceedings of ACM MobiCom*, 2014.

[9] L. Yang, Q. Lin, X. Li, T. Liu, and Y. Liu, "See through walls with cots rfid system!," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, pp. 487–499, ACM, 2015.

[10] L. Xie, Y. Yin, A. V. Vasilakos, and S. Lu, "Managing rfid data: Challenges, opportunities and solutions," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 3, pp. 1294–1311, 2014.

[11] L. Shangguan, Z. Li, Z. Yang, M. Li, Y. Liu, and J. Han, "Otrack: Towards order tracking for tags in mobile rfid system," in *Proceedings of IEEE INFOCOM*, 2013.

[12] "Wisp." http://wisp.wikispaces.com.

[13] S. Chen, M. Zhang, and B. Xiao, "Efficient information collection protocols for sensor-augmented rfid networks," in *Proceedings of IEEE INFOCOM*, 2011.

[14] Y. Qiao, S. Chen, T. Li, and S. Chen, "Energy-efficient polling protocols in rfid systems," in *Proceedings of ACM MobiHoc*, 2011.

[15] H. Yue, C. Zhang, M. Pan, Y. Fang, and S. Chen, "A time-efficient information collection protocol for large-scale rfid systems," in *Proceedings of IEEE INFOCOM*, 2012.

[16] H. Yue, C. Zhang, M. Pan, Y. Fang, and S. Chen, "Unknown-target information collection in sensor-enabled rfid systems," *IEEE/ACM Transactions on Networking (TON)*, vol. 22, no. 4, pp. 1164–1175, 2014.

[17] P.-J. Wan, O. Frieder, X. Jia, F. Yao, X. Xu, and S. Tang, "Wireless link scheduling under physical interference model," in *Proceedings of IEEE INFOCOM*, 2011.

[18] M. Shahzad and A. X. Liu, "Probabilistic optimal tree hopping for rfid identification," in *Proceedings of ACM SIGMETRICS*, 2013.

[19] J. Wang, H. Hassanieh, D. Katabi, and P. Indyk, "Efficient and reliable low-power backscatter networks," in *Proceedings of ACM SIGCOMM*, 2012.

[20] L. Kong, L. He, Y. Gu, M.-Y. Wu, and T. He, "A parallel identification protocol for rfid systems," in *Proceedings of IEEE INFOCOM*, 2014.

[21] D. Zanetti, B. Danev, *et al.*, "Physical-layer identification of uhf rfid tags," in *Proceedings of ACM MobiCom*, 2010.

[22] J. Ou, M. Li, and Y. Zheng, "Come and be served: Parallel decoding for cots rfid tags," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, pp. 500–511, ACM, 2015.

[23] EPCglobal, "Epc radio-frequency identity protocols class-1 generation-2 uhf rfid protocol for communications at 860 mhz - 960 mhz version 1.2.0," tech. rep., 2008.

[24] M. Chen, W. Luo, Z. Mo, S. Chen, and Y. Fang, "An efficient tag search protocol in large-scale rfid systems," in *Proceedings of IEEE INFOCOM*, 2013.

[25] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," *Internet mathematics*, vol. 1, no. 4, pp. 485–509, 2004.