ASAP: Scalable Collision Arbitration for Large RFID Systems

Chen Qian, *Student Member, IEEE;* Yunhuai Liu, *Member, IEEE;*
Hoilun Ngan; Lionel M. Ni, *Fellow, IEEE*

Abstract—The growing importance of operations such as identification, location sensing and object tracking has led to increasing interests in contactless Radio Frequency Identification (RFID) systems. Enjoying the low cost of RFID tags, modern RFID systems tend to be deployed for large-scale mobile objects. Both the theoretical and experimental results suggest that when tags are in large numbers, most existing collision arbitration protocols do not satisfy the scalability and time-efficiency requirements of many applications. To address this problem, we propose *Adaptively Splitting-based Arbitration Protocol* (ASAP), a scheme that provides efficient RFID identification for both small and large deployment of RFID tags, in terms of time and energy cost. Theoretical analysis and simulation evaluation show that the performance of ASAP is better than most existing collision-arbitration solutions and the time efficiency is close to the theoretically optimal values.

Index Terms—RFID; ALOHA protocol; Collision arbitration;

1 Introduction

Radio Frequency Identification (RFID) is a short-range radio communication technology that has been widely used in many applications such as identity recognition, population (cardinality) counting [1] [2] [32], and localization/tracking [4] [5] [6]. A standard RFID system is mainly composed of two types of devices: RFID tags and readers. RFID tags, labeled with a unique serial number (ID), are used to identify objects such as human beings and items. RFID readers that carry antennas are used to collect the information of RFID tags nearby. The simple structure and low-cost of RFID systems offer promising advantages to applications of large volumes of objects in a mobile environment [7]. Contactless RFID systems are also called RFID Networks.

Restricted by the simple structure, however, RFID tags cannot use CSMA-like protocols to avoid collisions. Two classical collision arbitration protocols are commonly used for RFID networks, namely slotted ALOHA [8] and Tree Traversal [9]. If every tag can report the reader without collision, their identities can be recognized. Hence in the context of RFID networks, collision arbitration protocols are also called *identification protocols* [10] [11]. The desired identification protocol for RFID networks should be time-efficient and scalable.

- *Chen Qian is with the Department of Computer Sciences, University of Texas, Austin, TX 78712.*
  E-mail: cqian@cs.utexas.edu
- *Yunhuai Liu is with The Third Research Institute of Ministry of Public Security, Shanghai, China.*
  E-mail: yunhuai@sub.siat.ac.cn
- *Hoilun Ngan and Lionel M. Ni are with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Kowloon, Hong Kong.*
  E-mail: {cpeglun, ni}@cse.ust.hk

Consider such an application of RFID systems: Hong Kong International Airport (HKIA) began to adopt the RFID technology in its logistic management system as early as August 2005. It handles more than 3.6 million tons of cargo annually, and *thousands of objects at a snapshot*. One important application is thus to identify all bags in the system for package verification purpose.

Time efficiency is a great concern because applications are usually in mobile environments. If the identification process lasts too long, mobile objects may have left the reader's range before being recognized. Moreover, the protocol should be scalable to support large number of tags as in the HKIA cargo transportation system. Privacy-preserving is also a problem [12] [13], which is beyond the scope of this paper.

Slotted ALOHA is an intuitive solution, but it does not appear to be scalable. Consider a slotted ALOHA protocol which sets its frame to 100 slots. To identify 100 tags, the system efficiency is about 37%, which is quite efficient and known to be the theoretical bound of ALOHA protocol. In mobile environments, the tag number may change. To identify 1000 tags, the theoretical efficiency goes down to lower than 0.1%. The efficiency of the protocol is highly affected by the cardinality. In other words, slotted ALOHA is *cardinality-sensitive* and thus not scalable. On the other hand, Tree-traversal (Query Tree) is *cardinality-insensitive* because when the cardinality increases, the system efficiency keeps still. One disadvantage of this protocol is that every two response windows (called slots) should be triggered by a control message (called query) sent from the reader. Too many reader queries lower the system efficiency. A scalable arbitration protocol with stable and high efficiency is desirable for dynamic RFID systems.

Protocols that combine estimation and identification together are also designed to make identification more efficient [1] [2] [14] [15]. However, these protocols work efficiently only in a limited range of tag cardinalities. Out of this range, the time efficiency decreases sharply according to theoretical and empirical results.

In this paper, we propose a novel Adaptive Splitting-based Arbitration Protocol (ASAP). ASAP is efficient, cardinality-insensitive and scalable. ASAP adaptively groups the tags into multiple subsets and estimates the cardinality of each subset during this process. Subsets form a hierarchy. The number of subsets in each hierarchy is always around $\log n$, where n is the tag cardinality. The ALOHA frame with 32 slots is enough for up to estimate 2^{32} tags. The estimation and identification are well combined. Compared with Tree-traversal which is also cardinality-insensitive, ASAP is able to improve the time efficiency to almost the twice (the detailed value depends on particular system parameters). Elaborated analysis shows that ASAP has a remarkably shorter processing time and lower energy cost when compared with the existing identification protocols in a wide range of tag cardinality. *ASAP works efficiently for large-scale RFID systems, and it also performs no worse than other protocols in small-scale applications.* Unlike some other approaches, ASAP does not require knowing the approximate tag number of the system in advance. Hence we believe ASAP can be a possible solution for dynamic RFID systems. The efficiency of ASAP does not depend on any performance models.

RFID arbitration protocols are also used for counting tag numbers. The only difference from identification is that, the tag reply message can be a very short notification instead of its ID to tell the reader increasing the counter. For counting tasks that do not require the completely precise result, estimation schemes can be applied [1] [2] [32].

The rest of this paper is organized as follows. In Section 2, we discuss the system model and some existing works for RFID identification. Section 3 presents our basic ideas and the detailed protocol design. We also proposed the arbitration technique that does not rely on geometric hashing in Section 4. We show the theoretical analysis in Section 5, and present the experimental performance evaluation in Section 6. Finally we conclude our work in Section 7, and indicate possible future work directions.

2 System Model and Preliminary

A contactless RFID system mainly consists of two types of components: RFID readers and tags. They communicate via radio signals. The reader broadcasts the interrogation messages (also called queries) and tags give back the response, carrying the desired information such as tag IDs. Since simultaneous responses by two or more tags will lead to response collisions, the reader cannot successfully receive all information in one round of communications. Therefore, the core problem for successful identification is arbitrating tag collisions in RFID networks.

TABLE 1
Notations

n	number of tags
t_q	time cost of a query
t_s	time cost of a slot
t_{switch}	time cost for switching mode
e_q	energy cost of a query
e_s	energy cost of a slot
e_{switch}	energy cost for switching mode
e_{resp}	energy cost to reply
Q	number of queries
S	number of slots
N_t	number of tag replies
W	number of mode switches

2.1 System Model

The RFID reader can switch between two modes, i.e., sending mode and listening mode. The basic unit of the sending mode is a query. And the basic unit of the listening mode is a listening slot. Using the notations in Table 1, we define the system time-efficiency R_t for every time unit t_u, as follows,

$$R_t = \frac{n}{Q \cdot t_q + S \cdot t_s + W \cdot t_{switch}} t_u \quad (1)$$

We define an energy-efficiency metric similarly.

$$R_e = \frac{n}{Q \cdot e_q + S \cdot e_s + W \cdot e_{switch} + N_t \cdot e_{resp}} e_u \quad (2)$$

The only difference is that active tags need energy $N_t \cdot e_{resp}$ to reply.

Usually the time or energy cost of a query is larger than that of a listening slot. A reply only includes the ID of the tag, while a query includes more than that, such as the ID of the reader, the sequence number of this query, the type of targeted tags, the ALOHA frame length, and the information of tag groups for polling. Also the switching time is considered, because both the reader and tags should do some processing before the next round of reading or listening.

We try to include more factors in our efficiency model to make it more generalized. Different parameters can be applied for different RFID systems. Other models for RFID efficiency have been proposed in the literature [16] [17] [14]. For example, in [14] the query cost is not counted. This model may be suitable for those systems with low query latency. The efficiency of our proposed protocol, ASAP, does not depend on any particular performance models and parameters. Under other models, ASAP is still (or even more) efficient by our experimental results.

2.2 Existing Protocols

Two main kinds of protocols are used in arbitrating collisions in ALOHA networks. In the context of RFID networks, collision arbitration protocols are also called *identification protocols*.

The first type of identification is slotted ALOHA [8]. The reader creates an ALOHA frame with a fixed number of time slots. Each tag randomly picks up a slot to transmit back. If a tag responds to a slot without collision, it is successfully identified by the reader. At the end of the frame, the reader acknowledges the successfully identified tags to keep silent in the next round. In the next round, a new frame will be created for those unidentified tags until all the tags are identified.

Suppose the frame length is l and the number of tags is n, and k queries are sent for synchronization. For each particular slot, the probability of success is

$$\Pr(sucess) = n \cdot \frac{1}{l} \cdot (1 - \frac{1}{l})^{n-1} \tag{3}$$

The efficiency is

$$R = \frac{l \cdot \Pr(sucess) t_u}{k \cdot t_q + l \cdot t_s + 2k \cdot t_{switch}} = \frac{n(1-\frac{1}{l})^{n-1} t_u}{k(t_q + 2t_{switch}) + l \cdot t_s} \tag{4}$$

Assume n and l are so large that we can omit the constant k. Also assume $t_s = t_u$. We have a well-known result:

Theorem 2.1: The highest system efficiency happens when $l = n$.

The proof is straightforward and skipped here. When $l = n$,

$$R_{max} = (1 - 1/n)^{n-1} \approx 1/e = 36.8\%$$

The value 36.8% is a theoretical upper-bound, since we have not included the query and synchronization cost.

In real applications, n is dynamic and could become very large. If $n = 500$ and $l = 100$, R decreases to 3.3%. If $n = 1000$ and $l = 100$, R is only 0.4%, which means in average the reader cannot hear one successful transmission during 100 slots. The efficiency of Slotted ALOHA is *cardinality-sensitive*, so that makes the system difficult to scale to large tag cardinalities.

The second type, Tree-traversal [9] [16] [17] [18] works as follows: The reader sends out a query to ask the tags with IDs prefixed by 0 and 1 to respond to two slots respectively. If there are collisions, the reader increases the prefix length further to 00, 01, 10 and 11, and sends queries asking tags with these prefixes to reply.

This process works like a depth-first (or breadth-first) traversal on a binary tree.

We analyze the performance of Tree-traversal under our model. According to [19], the slot cost of tree algorithm is $S = 2.885n$. The number of query can be approximated by $Q = S - n$ in *ideal cases*, since a query is needed by every non-leaf node in the tree.

Thus,

$$R = \frac{n}{1.89n(t_q + 2t_{switch}) + 2.89nt_s} t_u \tag{5}$$

Supposing in an RFID system, $t_s = t_{switch} = t_u$ and $t_q = 2t_s$, we have $R \approx 10\%$. Though the efficiency of Tree-traversal is lower than the maximum rate of slotted ALOHA, it is *cardinality-insensitive*. The performance is stable for different tag cardinalities.

In this paper, we call an identification protocol cardinality-insensitive, if for different tag cardinalities the time (energy) efficiency has stable values.

Some other models only consider the number of the listening slots, and use its inverse to represent the efficiency. If the query cost is ignored, then the efficiency of Tree-traversal is 35%, which is similar to the maximum rate of slotted ALOHA. Therefore we should use Tree-traversal in any cases because it is much more stable. This conclusion contradicts the experiences of ALOHA networks [8] [9]. Our model provides a reasonable explanation: slotted ALOHA enjoys a high efficiency in ideal cases, but loses the stability; Tree-traversal wins the stability but the efficiency is lower.

A desired identification protocol for large-scale RFID networks should have a stable system efficiency close to the maximum rate of slotted ALOHA.

2.3 Other Related Work

In recent years, RFID identification and estimation become hot research topics, due to the increasing deployment of RFID systems [25].

Besides the two classic arbitration protocols, slotted ALOHA and Tree Traversal, Simplot *et al.* first propose to use tag estimation to improve the throughput of RFID identification [26]. Bonuccelli *et al.* designed a protocol Tree Slotted ALOHA (TSA) [14] that combines estimation and identification together, so that it processes faster than Tree algorithm. TSA has a good performance when the cardinality is close to the frame length so that the Chebyshev's inequality-based estimation works. However, if the tag cardinality is dynamic and could be relatively large, estimation results of TSA will eventually be the same for most large cardinalities, and converge very slowly. DTSA [15] enhances the stability of TSA by dynamically apply the previous estimated value to adjust the next estimation. It still requires there are several identifiable slots. In cases like the number of tags, n, equal to several thousand, the estimation converges very slowly too. MSS [17] reduces the queries of Tree-traversal by opening a multi-slotted response window for each query. It provide significant energy savings compared with classical Tree protocol. ABS [16] is an enhanced tree-based protocol which reduce the number of collisions by exploiting information obtained from the last process of identification. ABS is an inter-process optimization. In this paper, we mainly focus on a single process. STT [18] is proposed to improve the efficiency for RFID tags with different ID distributions. Yeh *et al.* [27] suggest using pre-signaling to improve the RFID identification process. In [28], Yang *et al.* design a Quick Collision Detection (QCD) scheme based on the bitwise complement function plus collision preamble, which significantly simplifies the IC design of RFID tags. Very recently, Season [29] is proposed as a joint identification protocol to adjacent readers collaboratively identify the contentious tags, and solve the cross-tag-collision. Also several approaches are proposed to identify missing tags [30] [31].

Kodialam and Nandagopal [1] designed two early tag estimation algorithms: Unified Simple Estimator (USE) and Unified Probabilistic Estimator (UPE). Qian *et al.* propose LoF [2], which is the first geometric estimation protocol. FNEB [32] uses the position of the first reply from a group of tags for cardinality estimation. Li *et al.* address the energy efficiency problem for RFID estimation. Very recently, Zheng *et al.* [34] present another geometric estimation protocol that further improves the time efficiency.

Previous estimation-based arbitration protocols assume that the number of tags is in a relatively small range. When the tag number exceeds the range, the efficiency decreases. In this paper, the proposed ASAP scheme is highly scalable to a much larger range of tag numbers (up to 2^{32} in theory), as stated in Section 5. It is because ASAP incorporates estimation and arbitration into a recursive process. When the tag number is unknown, ASAP splits the entire tag set while estimating the cardinality of each subset. When the estimation of a subset is accurate enough, ASAP runs arbitration for the subset. Detailed protocol specification is presented in Section 3.

3 Adaptively Splitting-based Arbitration Protocol (ASAP)

In this section, we describe our identification protocol ASAP. ASAP is a combined estimation-identification protocol. We first introduce the Geometric Splitting-based Estimation (GSE), followed by the detailed design of ASAP.

We denote the number of tags and frame size by n and l, respectively. We do not consider other sources of identification errors such as poor read rates, ghost tags and occlusions. We also assume every tag can be probed by the single reader. The identification problem in multi-reader scenarios is left for future research.

3.1 Geometric Splitting-based Estimation

The *Geometric Splitting-based Estimation* (GSE) has first been proposed in [2]. Instead of randomly picking a slot to respond, the tag applies a geometric distributed hash function H to its ID. Here geometric distribution means that $1/2^t$ of the IDs have the hash value t. Then each tag responds in the time slot that matches its hash value, and tags in one time slot form a tag subset.

In a hash function, let the tags represent the keys and the slots represent the hash values. As illustrated in Fig. 1, we have n tags that are placed into l slots with

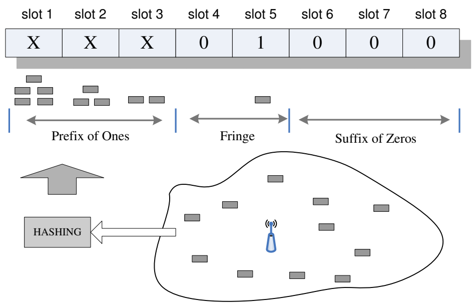

Fig. 1. An example of GSE

geometric distribution, i.e., $1/2^t$ of the tags are in the t-th slot. Therefore, approximately $n/2$ responses are in the time slot 1, $n/4$ are in the time slot 2...and $n/2^t$ are in the time slot t. Let $BM[]$ be the bitmap that records the responses to a frame, where 0 denotes a empty slot and 1 denotes a collision or single reply. Thus, the k-th bit in the bitmap $BM[k]$ will almost certainly be zero if $k \gg \log_2 n$, and be one if $k \ll \log_2 n$. The fringe consists zeros and ones for the k whose value is near $\log_2 n$.

We can estimate the tag number by:

Estimator 1.

$$\tilde{n} = \lambda \times 2^{P_0} = 1.2897 \times 2^{P_0} \tag{6}$$

is an estimator of the tag number n, where P_0 is the position of the first idle slot heard by the reader.

This estimator is suggested by [20] and [2].

In GSE the number of collision slots only increases logarithmically with the cardinality. Hence as long as the cardinality is not close to 2^l (l is the frame length), the estimation in ASAP is always cardinality-insensitive. Estimations used in TSA and DTSA [14] [15] split the entire tag set into 128 slots evenly, and estimate each subset by the number of collision slots. When tag cardinality increases, almost all slots become collisions, which affect the scalability. The simplest geometric hash function to implement is “the position of least-significant bit of zero in binary representation of tag ID”, assuming tag IDs are uniformly distributed. We will present how to construct multiple geometric hash functions when IDs are not uniformly distributed, in Section 3.4.

3.2 General Architecture of ASAP

The design principle of ASAP is based on the simple observation that slotted ALOHA is very inefficient when the tag cardinality increases, while Tree-traversal introduces too much control overhead. ASAP adaptively splits the entire tag set into multiple subsets, and estimates the cardinalities during the splitting. Unlike TSA and DTSA [14] [15], ASAP is stable for up to 2^{32} tags.

Essentially, ASAP uses a combined estimation-identification algorithm. Initially, the reader creates an ALOHA frame, named 1st-d frame, and asks all tags to reply in the slots. If collisions happen (it is almost for sure), ASAP splits the whole tag set into multiple

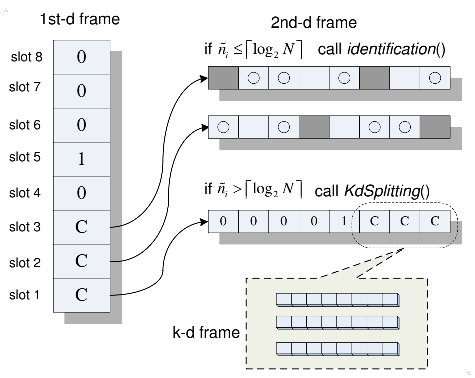
1st-d frame
2nd-d frame
slot 8
slot 7
slot 6
slot 5
slot 4
slot 3
slot 2
slot 1
0
0
0
1
0
C
C
C
if $\tilde{n}_i \le \lceil \log_2 N \rceil$ call *identification*()
if $\tilde{n}_i > \lceil \log_2 N \rceil$ call *KdSplitting*()
0 0 0 0 1 C C C
k-d frame

Fig. 2. An example of ASAP

subsets. Each subset contains tags that reply to the same collision slot. We push the collision slots follow a particular pattern, i.e., geometric distribution. The first slot contains about half of the n tags. The second slot contains about 1/4 of the n tags. The k-th slot contains about $1/2^k$ tags. Thus if the reader hears an idle or identifiable slot at position t. The collision slot prior to it (the $(t-1)$-th slot) probably contains 1 or 2 tags. The $(t-2)$-th slot has about 3 or 4 tag replies (the previous value multiply by 2). The cardinalities of all collision slots as well as the entire tag number can be estimated by the help of GSE.

Fig. 2 gives an example of the ASAP operation. Slot 4, 6, 7 and 8 are idle slots. Slot 5 only has one response and therefore the tag in slot 5 is successfully identified. Slot 1, 2 and 3 are collisions slots. Each of them corresponds to one tag subset. These subsets should be further identified. Suppose the estimated cardinality of subset S_i is $\tilde{n}_i$. If $\tilde{n}_i$ is smaller than a threshold T like slot 2 and 3, the reader employ a random ALOHA scheme framed by $\tilde{n}_i$ slots to identify the subset S_i. According to Theorem 1, it can achieve a high efficiency. If $\tilde{n}_i > T$ like slot 1, the subset should be splitted further by the scheme described in the last paragraph. Numbered by the recursion levels, the frames are named 1st-d frame, 2nd-d frame, ..., kth-d frame. The recursive process is called k-dimensional Splitting.

3.3 Enhanced Estimation in ASAP

Note that the GSE proposed in [2] cannot directly applied to ASAP. For example it is only able to estimate the entire tag cardinality. Also different hashes are required for k-dimensional Splitting. We describe the detailed design of ASAP and present an enhanced version of GSE in this section.

We first extend GSE to estimate the numbers of tags for subsets $s_0, s_1, ...$, where the set s_i contains tags that respond to slot i. If slot i is idle or with a single reply, the cardinality of s_i will be 0 and 1. To estimate the number of tags in a collision slot s_i, we have,

Estimator 2. If the ith slot is a collision slot, the estimated cardinality of tags that reply in this slot, i.e., the cardinality of s_i, is

$$\tilde{n_i} = 1.2897 \times 2^{P_0 - i} \tag{7}$$

where P_0 is the position of the first idle slot heard by the reader.

Estimator 2 can be derived by the property of geometric distribution.

We can expect that 32 slots are sufficient for the estimation algorithm because few applications could have a cardinality of 2^{32}. Therefore, in ASAP, we fix the frame length as 32.

Unlike the protocol in [2], the GSE phase in ASAP does not always finish the entire time-slotted frame. In the other words, for most frames, 32 slots is sufficient but not necessary, because obviously in most cases tag numbers are much lower than 2^{32}. According to Estimator 2, we know that the useful information in GSE is the first idle slot (left-most zero). Therefore as soon as the reader hears an idle, it immediately probes a “STOP” message and opens one listening slot. On receiving a “STOP”’, all tags that have not replied yet should report to the listening slot. In the example of Fig. 2, all tags after slot 5 (if any) should reply to slot 5. We can expect that the number of these tags will not be very large. This technique compresses the listening slots from 32 to about $\log n + 1$. It benefits the identifications for which $\log n \ll 32$.

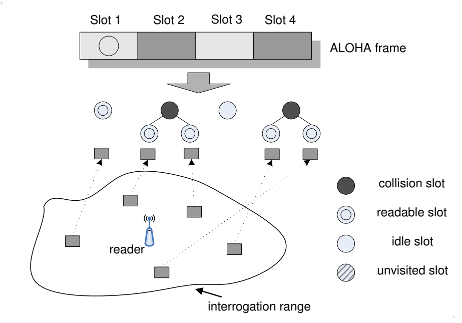

Fig. 3. An example of identification in ASAP

```
KdSplitting()

PROCEDURE
1. wait_for_message();
2. if there is a splitting message from a reader
3.   l ← the frame length in the message;
4.   Select a slot k from [0, l - 1] in geometric distribution;
5.   Transmit a message in time slot k.
6. end if
```

(a) KdSplitting algorithm on tags

```
KdSplitting(S)

INPUT
The tag set S.

PROCEDURE
1.  Broadcast a splitting message to tags in S;
2.  for i = 0 to l - 1
3.    wait_for_response();
4.  end for
5.  for i = 0 to l - 1
6.    if time slot i has a single response
7.      Recognize the tag;
8.    else if time slot i is a collision
9.      n ← estimated cardinality of tags in slot i;
10.     if n ≤ T
11.       Put Identification(S_i) into the execution queue;
12.     else
13.       Put KdSplitting(S_i) into the execution queue;
14.     end if
15.   end if
16. end for
```

(b) KdSplitting algorithm on readers

Fig. 4. Pseudocode of KdSplitting()

```
Identification()

PROCEDURE
1. wait_for_message();
2. if there is an identification message from a reader
3.     $l \leftarrow$ the frame length in the message;
4.     Select a slot $k$ from [0, $l$ - 1] in uniformly random;
5.     Transmit a message in time slot $k$.
6. end if
```

(a) Identification algorithm on tags

```
Identification(S)

INPUT
The tag set $S$.

PROCEDURE
1.  Broadcast an identification message to tags in $S$;
2.  for $i$ = 0 to $l$ - 1
3.      wait_for_response();
4.  end for
5.  for $i$ = 0 to $l$ - 1
6.      if time slot $i$ has a single response
7.          Recognize the tag;
8.      else if time slot $i$ is a collision
9.          TreeTraversal($S_i$);
10.     end if
11. end for
```

(b) Identification algorithm on readers

Fig. 5. Pseudocode of Identification()

3.4 k-dimensional Splitting

The simple version of ASAP protocol immediately applies slotted ALOHA using the estimated values as frame lengths to identify each tag subsets. However, it is suggested that ALOHA should not use long frames due to the time synchronization problem [21] and interference between readers in a multi-reader environment [24]. Therefore, for some s_i with relatively large size, the protocol splits it further by the kth-d frames using different geometric hash functions which is called *k-dimensional splitting*.

In the experiments of this work, we set the threshold as 32, although the threshold completely depends on the particular RFID systems. More specifically, the reader performs a splitting by calling the function *KdSplitting()* when $\tilde{n}_i > 32$. The algorithm *KdSplitting()* recursively applies GSE to split the tag sets like Fig. 2. When otherwise, i.e., a subset has an estimated cardinality smaller than 32, the identification scheme is called. *KdSplitting()* can either be depth-first or breadth-first. Here we use the depth-first one.

When an un-identified subsets s_i whose estimated cardinality is smaller than 32, we call the algorithm *Identification()*. It first apply random slotted ALOHA. If collisions still exist, the reader uses Tree-traversal to deal with the collisions. An example is depicted in Fig. 3. In Fig. 3, as we know in advance that there are approximately four tags in the subset (but the exact number is five), an ALOHA frame with four slots will likely be sufficient to give each tag a dedicated slot when each tag randomly picks one to transmit. If collisions still happen, each collision slot can be considered as a tree root to continue a Tree-traversal. It stops when every tag is identified.

The pseudocode specification of *KdSplitting()* and *Identification()* is presented in Figures 4 and 5.

In *KdSplitting()*, the protocol may require a tag to apply several different hash functions for different K=1, 2, ... It is not specified how to obtain multiple different geometric hash functions in [2]. Here we provide a solution. The position of least-significant bit of zero in binary representation of tag ID is geometric distributed. Let us denote this hash as H'. Then we also employ a group of uniform distributed hash functions, e.g., Secure Hash Algorithm 2 (SHA-2) or Message-Digest Algorithm 5 (MD5) [35], denoted by $H_1, ..., H_k$. It is obvious that $H'(H_1(ID)), ..., H'(H_k(ID))$ are all geometric distributed hash functions, because $H_i(ID)$'s rightmost zero also has a probability of $1/2^t$ to be in bit $t-1$. The hash values of MD5 are 128-bit, but it is not difficult converting them to the length we want. Note that *ASAP does not require the hash computation on each tag*. To avoid computing hashes on tags, we just pre-compute the hash values for all tags during the manufacture and write the hash values onto them as hard state. Each tag will select one of the values under the request of the reader. During the splitting process, tags just need to compare its last-used hash value and the hash value in the current query. They do not require extra memory to identify their groups.

Another method to generate geometric distribution is to do sequential coin-flipping. It is known that the first head (or tail) appearing in sequential coin-flipping follows geometric distribution. Recent progress in RFID hardware show that coin-flipping can be done in RFID tags on some bits which predictably power up randomly [22].

4 ASAP without Geometric Hashing

Using geometric hashing for ASAP increases either computational cost, or storage. In this section, we proposed a method that helps ASAP to perform the same functions while requiring no geometric hashing.

Besides the unique ID, we further assume that each tag has another number with the same digits as the ID in binary representation, called *ASAP code*, in short, *A-code*. To generate a A-code for each tag, the manufacturer or the RFID system can use a uniform hash function H, and use $H(ID)$ as the A-code. For each tag, only one A-code is needed during lifetime. From Section 3.4 and the analysis in [23], although the tag IDs may have special distribution and pattern, their A-codes are re-distributed to almost uniform for large-scale RFID system by the uniform hashing H.

Let L be the number of digits of A-code in binary representation. When the system starts ASAP, instead of asking tags to reply in geometric distribution, the reader issues a random number r with L bits, and includes the number into the reader query. After hearing the query message, each tag replies to the $(k+1)$-th slot if the suffix of its A-code matches k bits to the random number. For example, if the random number $r = 010011$, then tags whose A-codes are *****0 reply to the first slot, because their suffixes match no bit of r. They account for about 50% of the entire tag set, since the A-codes are almost uniform for large-scale RFID system. Similarly, tags whose A-code are ****01 reply to the second slot, and so on. In this way, tags in slots are also in approximately geometric distribution. Hence, GSE can be performed without geometric hashing for the 1st dimensional frame.

For the 2nd dimensional frame, tags in each collision slot need to be further splitted if $\tilde{n}_i > T$, where T is the threshold between splitting and identification. At this stage, the reader knows a certain length of the tag suffixes. In the above example, tags in slot 2 are known to have suffixes 01. Hence for the 2nd dimensional splitting, the reader issues a new random number with $L-d$ bits, where d is the length of the known suffix, for tags in each collision slot. Tags match the random number with their *first $L-d$ bits* and select a slot in the 2nd frame to reply. Using the above example again, the reader issues 1001 for tags in slot 2 of the 1st dimensional frame, and asks them to reply to the 2nd dimensional frame. Tags whose A-code are ***001 reply to slot 1 in the 2nd dimensional frame, because their first $L-d$ bits ***0 match no bit in the suffixes to the random number; tag whose A-code are **1101 reply to slot 2, and so on. Therefore, ASAP can perform the recursive splitting without geometric hashing.

There is a little difference between ASAP using geometric hashing and using random number matching as introduced above. Note that in the example, the reader only need to set the 2nd dimensional frame to be $L-d+1$ slots, because the maximum number of bits for tags to match is the random number is $L-d$. Hence the kth-d frame length is deterministic. If ASAP uses geometric hashing, the frame length is nondeterministic, because every time a different hash function is used. Original ASAP can terminate a frame whenever the remaining slots are of no use. By this reason, ASAP may potentially achieve a little higher efficiency than ASAP without geometric hashing does. This fact is validated by our experimental results in Section 6.

5 Analysis and Discussion

The most significant performance metrics of identification protocols include the processing time (latency) and the energy consumed. Here we analyze them respectively by developing several analytical models. The time and energy cost models are both based on the sequential (SEQ) operation. The sequential operation is a half-duplex operation where the reader sends out the query message for a specific period, and then changes its mode to listen to the responses until the end of the ALOHA frame. We consider both reader queries and tag responses in the analysis model. Our analysis only focuses on average cases.

5.1 Time Cost Analysis

Consider the model described in Section 2.1. We present the total number of queries, slots, switches and responses as functions of n. The total time cost can be expressed as $T(n) = Q(n)t_q + S(n)t_s + W(n)t_{switch}$, where $Q(n)$ is the number of queries and $S(n)$ is the number of total time slots required for identifying n tags.

We first determine the value of the number of queries $Q(n)$. We can express $Q(n)$ as $Q_s(n)+Q_i(n)$, where $Q_s(n)$ is the number of queries sent in *KdSplitting()* and $Q_i(n)$ is that of *Identification()*. $Q_s(n)$ can further be consider to equal to the times of calling *KdSplitting()* plus the times of calling *Identification()* in *KdSplitting()*. As described in our protocol, after LoF estimation, tag sets whose cardinality is less than or equal to $M = 32$ are have to call *KdSplitting()* and others call *Identification()*. Let T represent the average threshold position of these two kinds of sets. We have $\frac{n}{2^T} \approx M$ because the splitting stops when the set's cardinality is less than or equal to M. We derive,

$$T \approx \log_2\left(\frac{n}{M}\right) < \log_2 n \tag{8}$$

The number of sets that need to call *Identification()* is less than $\log_2 n - T = \log_2 M$. Thus, the value $Q(n)$ can be expressed as,

$$Q_s(n) = \log_2 M + T + \sum_{i=1}^{T} Q_s\left(n/2^i\right) \tag{9}$$

Similarly,

$$\begin{cases} Q_s(n/2) = \log_2 M + T - 1 + \sum\limits_{i=1}^{T-1} Q_s\left(\frac{n}{2^{i+1}}\right) \\ \ldots\ldots \\ 2^{j-1}Q_s(n/2^j) = 2^{j-1}\log_2 M + 2^{j-1}T \\ \qquad\qquad -j2^{j-1} + 2^{j-1}\sum\limits_{i=1}^{T-j} Q_s\left(n/2^{i+j}\right) \end{cases}$$

where $j < T$. Adding the left and right sides of all these inequalities, we obtain,

$$Q_s(n) = \sum_{j=1}^{T} 2^{j-1}\log_2 M + \sum_{j=1}^{T} 2^{j-1}(T-j).$$

Since $\sum\limits_{j=1}^{T} j2^j = (T-1)2^{T+1} - 1$,

$$Q_s(n) = \frac{n\log_2\log_2 n}{M} + \frac{n}{M}$$

Since $\log_2 n < M = 32$, the complexity of $Q_s(n)$ is much less than $O(n)$. The value of $Q_i(n)$ will be determined later.

We then compute the number of total time slots required, $S(n)$. We can express $S(n)$ as $S_s(n)+S_i(n)$, where $S_s(n)$ is the number of time slots required in *KdSplitting()* and $S_i(n)$ is that of *Identification()*. The total number of queries for *KdSplitting()* is the second term of (11). Four additional time units to send the “Stop” messages should also be included. It is easy to get,

$$S_s(n) < \frac{n(\log_2 n + 4)}{M} < n \tag{10}$$

The complexity of $S_i(n)$ needs more consideration. In *Identification()* the length of the frame is the estimation value $\tilde{n}$. If collisions still occur, each collision slot can be considered as a tree root to continue tree-based splitting.

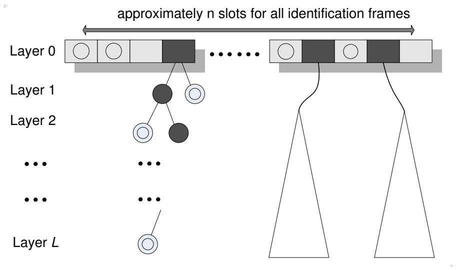

Fig. 6. Identification Tree

As illustrated in Fig. 6, for each tag, the identificaion can be considered as a tree-based searching, and the search tree now has n roots. All n tags are uniformly distributed in the interval [0,1) at every layer of the search tree. Therefore, for one tag searching, we have the following equations to express the probabilities that the slot in layer l of this searching is an idle slot, a successful slot or a collision slot respectively,

$$\Pr_{idle}(l) = \left(1 - 2^{-l}/n\right)^n$$

$$\Pr_{readable}(l) = n \times \frac{2^{-l}}{n}\left(1 - 2^{-l}/n\right)^{n-1} = 2^{-l}\left(1 - 2^{-l}/n\right)^{n-1}$$

$$\begin{aligned}\Pr_{collision}(l) &= 1 - \Pr_{readable} - \Pr_{idle} \\ &= 1 - 2^{-l}\left(1 - 2^{-l}/n\right)^{n-1} - \left(1 - 2^{-l}/n\right)^n\end{aligned}$$

If a node at layer l is visited by at least one tree-based searching, the parent of this node must be a collision slot. Since the tags are uniformly distributed, the probability that a node at layer l is visited by at least one searching is,

$$\begin{aligned}\Pr_{visited}(l) &= \Pr_{collision}(l-1) \\ &= 1 - 2^{-l+1}\left(1 - 2^{-l+1}/n\right)^{n-1} \\ &\quad - \left(1 - 2^{-l+1}/n\right)^n\end{aligned}$$

The number of time slots required on average for identification is equal to the number of nodes that are visited by the tag searching. Let L denote the deepest layer in the n-root search tree. Then we are able to compute $S_i(n)$ by summing the visiting probabilities of all nodes in all layers,

$$\begin{aligned}S_i(n) &= n + \sum_{l=1}^{L}\sum_{i=0}^{2^l-1}\Pr_{visited}(l) \\ &= n + \sum_{l=1}^{L}\sum_{i=0}^{2^l-1}\left[1 - 2^{1-l}\left(1 - 2^{1-l}/n\right)^{n-1} - \left(1 - 2^{1-l}/n\right)^n\right] \\ &= n + \sum_{l=1}^{L}\left[2^l - 2\left(1 - 2^{1-l}/n\right)^{n-1} - 2^l\left(1 - 2^{1-l}/n\right)^n\right]\end{aligned}$$

For any expression $(1 - h/n)^n$ where $h < 1$ and n is a large number, using the binomial expansion approximation, we have[1]

$$\begin{aligned}(1 - h/n)^n &= 1 - h + \frac{h^2(n-1)}{2n} - \frac{h^3(n-1)(n-2)}{6n^2} + \ldots \\ &> 1 - h.\end{aligned}$$

Thus we derive,

$$\begin{aligned}&S_i(n) \\ &< n + \sum_{l=1}^{L}\left\{2^l - 2\left[1 - \frac{2^{-l+1}(n-1)}{n}\right] - 2^l\left(1 - 2^{-l+1}\right)\right\} \\ &= n + \sum_{l=1}^{L}\left[\frac{2^{-l+2}(n-1)}{n}\right] \\ &< n + \sum_{l=0}^{L} 1 = n + L\end{aligned}$$

Since $2^L \ll n$,

$$S_i(n) = n + o(\log_2 n) \tag{11}$$

Considering that the estimation will introduce errors which affect the value of $S_i(n)$, we use $S_i(n) < 2n$.

Also $Q_i(n) < L + 1 < \log_2 n$.

In summary,

$$Q(n) = Q_s(n) + Q_i(n) < \frac{n\log_2\log_2 n + n}{M} + \log_2 n \tag{12}$$

$$S(n) = S_s(n) + S_i(n) < 2n + \frac{n(\log_2 n + 4)}{M} \tag{13}$$

It is obvious that we have $W(n) = 2Q(n)$.

By (14) and (15), assuming $t_u = t_s = t_{switch} = t_q/2$, [2] we have

$$\begin{aligned}T_{ASAP}(n) &= Q(n)t_q + S(n)t_s + W(n)t_{switch} \\ &= \left(\frac{n\log_2\log_2 n + n}{M} + \log_2 n\right)(t_q + 2t_{switch}) \\ &\quad + \left(2n + \frac{n(\log_2 n + 4)}{M}\right)t_s\end{aligned} \tag{14}$$

Since $n < 2^{32}$, $T_{ASAP}(n)$ is upper-bounded by $4.125n$. Thus the efficiency $R_t = n/T > 24.2\%$. Since this value of R is only a lower-bound, the actual efficiency is higher (26.8% based on the experimental evaluation). This bound is valid for up to billions (2^{32}) of tags.

The efficiency of Tree algorithms is about 10% by Section 2.2. ASAP has a remarkably higher efficiency, which is also cardinality-insensitive as we will show in Section 6. Based on the analysis in [14], for TSA working in ideal cases (n is close to 128), the time cost $T(n) \approx \frac{3}{4}n(t_q + 2t_{switch}) + 2.3nt_s = 5.3nt_u$. Hence the efficiency for ideal cases is around 18.8%. When n becomes larger, TSA will degenerate to a Tree-traversal-like algorithm with a efficiency value about 14%. Note if we use other models ASAP still outperforms Tree-traversal and TSA.

1. Actually we may directly get the inequality from *Bernoulli's inequality* in real analysis.

2. Please note that this assumption is just presenting an example RFID system to get direct-viewing values. Different systems may have different relationships of t_u, t_s, t_{switch} and t_q. ASAP will have qualitatively same performance for them. It can be seen by substituting other settings into (14) and (15). Also we will validate this fact in the evaluation section.

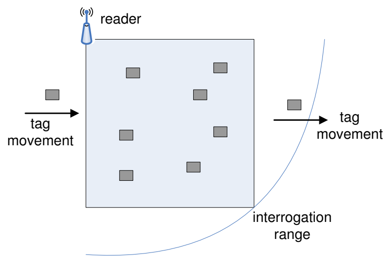

Fig. 7. Simulation environment

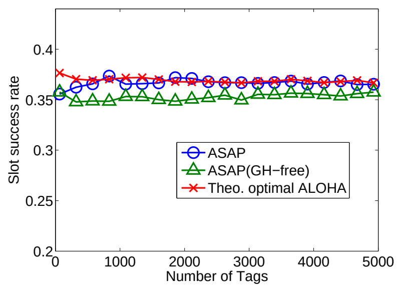

Fig. 8. Slot success rate vs. number of tags

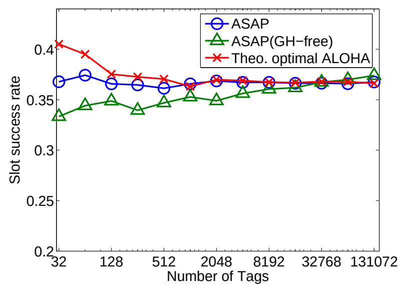

Fig. 9. Slot success rate vs. number of tags with larger range of cardinalities

5.2 Energy Cost Analysis

Energy efficiency is another important metric [17]. For passive-tag RFID systems, the energy is only consumed on the readers. Thus the analysis of energy cost is similar to that of time cost. For active-tag RFID systems, we have to consider the energy cost on both the reader side and tag side. Due to the space constraint, we put the detailed derivation in the supplementary material. Similar to the time cost, ASAP has stable and low energy cost.

5.3 The Trade-off in ASAP

The design of ASAP requires tags either storing several hash values in advance, or having the random power up bits. Since the binary string of $H(ID)$ is much shorter than that of ID, the extra memory cost in each tag is not very much. As analyzed in Section 3, after T times of splitting, the cardinality of a set must be less than M. Plus a uniform hashing needed for the identification frame, the total number of hash values a tag has to store is $T+1 < \log_2 n < M$. Since the string length of $H(ID)$ is a constant number and the length of ID is M, the extra memory cost is close to the memory used to store tag ID. Besides, our protocol only requires tags to spend little extra memory to store its state and identify the group it belongs to. ASAP trades the storage for time and energy efficiency and performance stability.

6 Performance Evaluation

We have conducted extensive simulations to verify the performance of ASAP and Geometric Hashing-free ASAP, short as ASAP (GH-free). In order to precisely evaluate RFID arbitration protocols, we build a packet-level simulator including tag mobility. We choose simulation instead of real implementation because of the two reasons: 1. ASAP (and some other protocols for comparison) requires some change of the tag storage, which can only be done by the manufacturer. 2. Simulation enables us to study large-scale systems and explore the potential of the protocols.

ASAP is compared with the general Tree-traversal algorithm, TSA [14] and DTSA [15], with varies number of tags. Time Efficiency, $R_t(n)$, energy efficiency, $R_e(n)$, and accuracy are essential performance metrics. The Time and energy efficiencies metrics follow Section 2 and 5. We also tried different performance models in Section 6.3. Accuracy is defined as the fraction of tags to be recognized by the reader. When tags are mobile, some of them can move out of the interrogation region before being identified.

6.1 Simulation Setup

Different amounts (32 to 131072) of tags will be deployed in a rectangular region with dimension 10m$\times$10m. Due to the limited computation ability, we are not able to simulate over millions of tags. Note that the arbitration itself is very efficient, but to simulate millions of mobile tags is resource-consuming. The readers are located at the top-left hand corner of the region with a communication radius of 14.4m. Tags move from the left to the right. Only tags inside the rectangular region can be identified. The tag IDs are 32-bits wide. The environment is shown in Figure 7. The applications of such environment can be the merchandise in a factory, cargo in an airport, visitors to a conference, people in a stadium, etc. The simulation is repeated for 100 times and the average results are presented below. In each figure, the unit of time or energy cost is unified as Section 2.

6.2 Performance of ASAP

In this section, we show the performance gain of ASAP over other approaches.

We first show the *slot success rate* of ASAP. Slot success rate is defined as the number of successful slots over the number of total slots. This metric implies the time efficiency. More importantly, it is model-independent. We also add the theoretically optimal ALOHA into comparison. In this protocol, the RFID reader is assumed to know the exact number of tags in advance for every

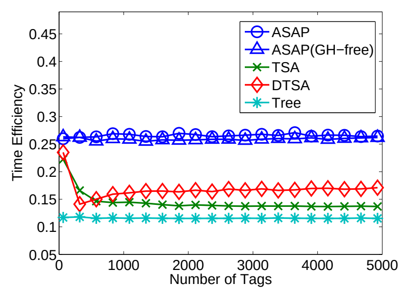

Fig. 10. Comparison in time efficiency

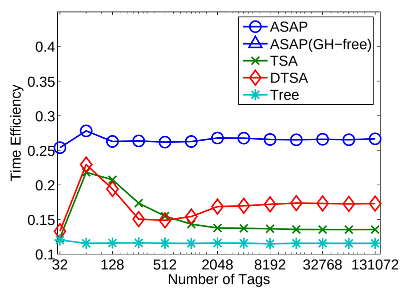

Fig. 11. Time efficiency with larger range of cardinalities

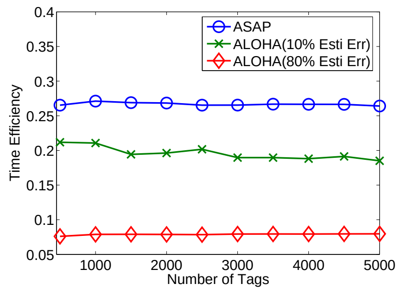

Fig. 12. ASAP vs. slotted ALOHA and tag no. known with error

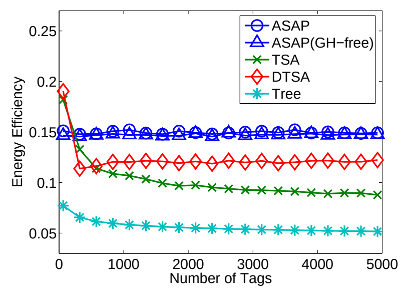

Fig. 13. Comparison in energy efficiency

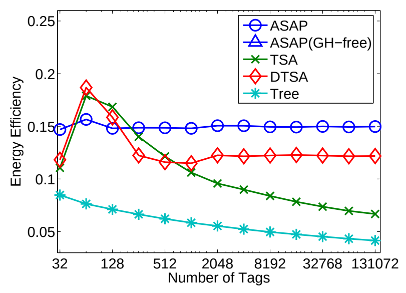

Fig. 14. Energy efficiency with larger range of cardinalities

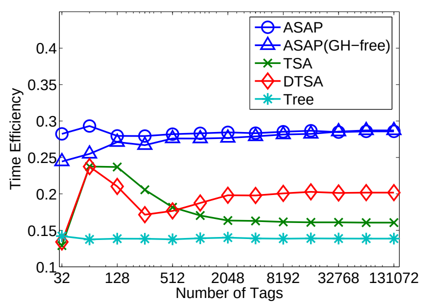

Fig. 15. Time efficiency under Model *B*

ALOHA frame, until all tags are identified. Since no estimation protocol can report the exact number, such ALOHA protocol only exists in theoretical aspect.

Figure 8 shows the slot success rates of ASAP and ASAP (GH-free). The values of this rate have no great changes from 64 to about 5000 tags. ASAP (GH-free) has lower success slot rate because its frame length is deterministic, which may be longer than the optimal length. Theoretically optimal ALOHA is better than ASAP for about 2-3% when the tag number is less than 500. When tag number becomes larger, there is no visible difference between the slot success rates of ASAP and theoretically optimal ALOHA. To validate the scalability of ASAP, we vary the number of tags from 32 to 65536 in Figure 9. The two ASAP curves are flat and close to the optimal curve. The slot success rate of ASAP (GH-free) actually increases when the cardinality is large.

Figure 10 plot the time efficiency for identifying different amounts of tags (from 64 to 4992). The figure indicates that the performance of ASAP is always better than those of existing approaches. Followed by ASAP (GH-free) with a little difference. To better characterize these protocols, we also plot the time efficiency in a larger cardinality range in Fig. 11. From Fig. 10 and 11, the efficiency of ASAP is always within the range $[26\%-2\%, 26\%+2\%]$, which means ASAP is cardinality-insensitive. It shows that when tag cardinality is close to the frame length (128) of TSA and DTSA, their performances are good too [3]. However their efficiencies drop when the cardinality goes away from the frame length. We also compare ASAP with slotted ALOHA under the assumption that the tag number is known with a certain error. We include the synchronization problem in the simulator. As shown in Fig. 12, ASAP is more efficiency than simple slotted ALOHA.

The comparison of energy efficiency is plotted in Fig. 13 and 14. ASAP is the most efficient except for the ideal cases of TSA and DTSA. Note that ASAP is the only cardinality-insensitive scheme in terms of energy efficiency. Even the performance of Tree algorithm decreases as the cardinality grows. TSA and DTSA is more efficient than ASAP in energy cost from the tag number 64-128, because these values are close to the default frame length of TSA and DTSA (100). When the frame length is close to the tag number, the protocols perform similarly to theoretically optimal ALOHA. Therefore TSA and DTSA win ASAP by saving the cost for recursively estimating

3. The values of TSA and DTSA shown here are different from those in the original papers [14] [15], because the query cost was not considered there.

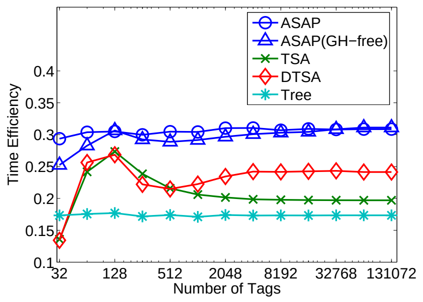

Fig. 16. Time efficiency under Model C

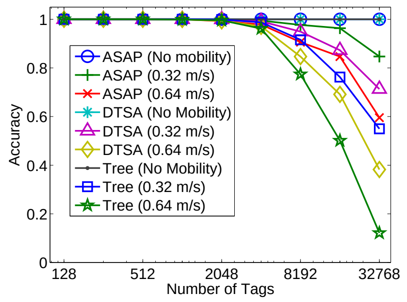

Fig. 17. Accuracy comparison with tag mobility and exponentially increasing cardinalities

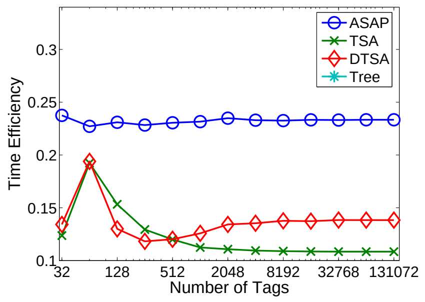

Fig. 18. Time efficiency for lossy links

and splitting the tag sets.

We found, in the results, ASAP achieves a little higher efficiency than ASAP without geometric hashing does. It is because original ASAP can terminate a frame whenever the remaining slots are of no use, as discussed in Section 4. By this reason, ASAP may have shorter processing time. On the other hand, ASAP (GH-free) gains the advantage that storing hash functions is not necessary.

6.3 Impact of Model Selection

The advantages of ASAP do not depend on our model. We also evaluate the protocols in two other models. Model B uses $t_q = t_s$, where t_q is the time for a query and t_s is that for a slot. Model C does not consider the switching time. Figure 15 and 16 show the results of model B and C respectively. ASAP still performs stably and more efficiently than other approaches. The results for energy cost are similar to those in Fig. 13 and 14 and not shown here.

6.4 Impact of Tag Mobility

It is not uncommon to have mobile tags. We simulate tag mobility by allowing the tags to move in a random direction at different speed. The result is shown in Fig. 17.

The accuracy is less than 1 because some of the tags are moved to a location outside the reading range of the readers before they have been recognized. Initially, all the tags are within the reading range of the readers. The higher the speed of the tag movement, the faster the tags are beyond the reading range of the reader. For all algorithms, the accuracy drops with increasing number or moving speed of tags. However, a more severe dropping rate is observed in the Tree-traversal approach. ASAP is capable to maintain an accuracy of around 95% with 10000 tags and high tag mobility.

6.5 Impact of Tag ID Distribution

Due to the space limitation, this section is presented in the supplementary material.

6.6 Impact of Lossy Links

Practical RFID networks might contain lossy communication links, i.e., the transmission from the reader to tags (or in the other direction) only succeed in a probability. We also test ASAP under lossy links. Figure 18 shows the result where the average link quality is 0.8. The efficiency of ASAP is still close to 0.25.

TABLE 2
Summary of Properties

	R_t, fixed n	R_t, small n	R_t, large n	extra storage
ALOHA	+	-	-	No
Tree	-	-	-	No
TSA/DTSA	+	+	-	No
ASAP	+	+	+	Yes
ASAP GH-free	+	+	+	No

6.7 Suggestion of Protocol Selection

RFID network administrators choose protocols which fit best to their particular application requirements. If the system has the precise knowledge of tag cardinality, slotted ALOHA is always the best choice. However it is rare in reality because there is no scheme that can count the tag number precisely without arbitrating collisions. Also mobile tags dynamically change the cardinality. By Fig. 10 to 17, ASAP is preferred in general cases no matter the cardinality is unknown or know approximately. For a passive-tag system, if the approximate tag number is known and the system is energy-aware, DTSA is also a good choice because it has the least energy cost in such case. We summarize the properties of these protocols in

Table 2. The table compared the time efficiency R_t of different protocols for fixed n, small n and large n.

The results for the energy efficiency R_e are similar to those of R_t.

7 Conclusion and Future Work

Efficient and scalable RFID identification protocol is on demand of many real-world applications. Slotted ALOHA scheme loses the stability, and Tree-traversal does not process efficiently. Motivated by these limitations, we design a new identification protocol called Adaptive Splitting-based Arbitration Protocol (ASAP). Analysis and experiments show that ASAP dramatically outperforms existing approaches considering both efficiency and scalability.

Our future work will be carried on along following directions. First, we only consider a single reader. In practice, there could be multiple readers functioning in overlapped regions. New problems will arise and further improvement spaces are available as well. How to benefit from such multiple readers and tackle the new problem is still unclear. Second, we assume a single, omni-direction antenna for each reader. This assumption is not always true as the practical readers may have multiple antennas. In the last, we consider a low mobility environment where the target objects have limited mobility. Concerning the practice, we believe there could be much more applications when high mobility scenarios are supported.

Acknowledgment

This research was supported in part by Hong Kong RGC Grants HKUST617710 and HKUST617811, National High Technology Research and Development (863) Program of China under Grant No. 2011AA010500, China NSFC Grant 60933011, 61073180 and 61170247, China National Science and Technology Supporting Program 2012BAH07B00, and US National Science Foundation grant NSF CNS-0830939. We thank Simon Lam for the valuable comments and discussion. We also thank the anonymous reviewers for helping us to improve this paper.

References

[1] M. Kodialam and T. Nandagopal, "Fast and Reliable Estimation Schemes in RFID Systems," in *Proc. of ACM Mobicom*, 2006.
[2] C. Qian, H.-L. Ngan, and Y. Liu, "Cardinality Estimation for Large-scale RFID Systems," in *Proc. of IEEE PerCom*, 2008.
[3] H. Han, B. Sheng, C. C. Tan, Q. Li, W. Mao, and S. Lu, "Counting RFID Tags Efficiently and Anonymously," in *Proc. of IEEE Infocom*, 2010
[4] L. M. Ni, Y. Liu, Y. C. Lau, and A. Patil, "LANDMARC: Indoor Location Sensing Using Active RFID," in *Proc. of IEEE PerCom*, 2003.
[5] Y. Liu, L. Chen, J. Pei, Q. Chen, and Y. Zhao, "Mining Frequent Trajectory Patterns for Activity Monitoring Using Radio Frequency Tag Arrays," in *Proc. of IEEE PerCom*, 2007.
[6] M. Zuniga and M. Hauswirth, "Hansel: Distributed Localization in Passive RFID Environments," in *Proc. of IEEE SECON* 2009.
[7] S.-J. Tang, J. Yuan, X.-Y. Li, G. Chen, Y. Liu, and J. Zhao "RASPberry: A Stable Reader Activation Scheduling Protocol in Multi-Reader RFID Systems." in *Proc. of IEEE ICNP* 2009.
[8] L. G. Roberts, "Aloha Packet System with and without Slots and Capture," *ACM SIGCOMM Computer Communication Review*, vol. 5, pp. 28-42, 1975.
[9] J. I. Capetanakis, "Tree algorithms for packet broadcast channels," *IEEE Trans. on Information Theory*, vol. IT-25, pp. 505-515, 1979.
[10] D. Simplot-Ryl, I. Stojmenovic, A. Micic and A. Nayak, "A Hybrid Randomized Protocol for RFID Tag Identification", *Sensor Review* 2006.
[11] L. Xie, B. Sheng, C. C. Tan, Q. Li, and D. Chen, "Efficient Tag Identification in Mobile RFID Systems," in *Proc. of IEEE Infocom*, 2010
[12] G. Tsudik, M. Burmester, A. Juels, A. Kobsa, D. Molnar, R. Di Pietro, M. R. Rieback, "RFID security and privacy: long-term research or short-term tinkering?" in *Proc. of WISEC* 2008.
[13] Q. Yao, Y. Qi, J. Han, J. Zhao, X. Li, and Y. Liu, "Randomizing RFID Private Authentication", in *Proc. of IEEE PerCom* 2009.
[14] M. A. Bonuccelli, F. Lonetti, F. Martelli, "Tree Slotted Aloha: a New Protocol for Tag Identification in RFID Networks," *Elsevier Ad Hoc Networks* 2007.
[15] G. Maselli, C. Petrioli, C. Vicari, "Dynamic Tag Estimation for Optimizing Tree Slotted Aloha in RFID Networks," in *Proc. of ACM MSWIM* 2008.
[16] J. Myung and W. Lee, "Adaptive Splitting Protocols for RFID Tag Collision Arbitration," in *Proc. of ACM MobiHoc*, 2006.
[17] V. Namboodiri and L. Gao, "Energy-Aware Tag Anti-Collision Protocols for RFID Systems," in *Proc. of IEEE PerCom*, 2007.
[18] L. Pan and H. Wu, "Smart Trend-Traversal: A Low Delay and Energy Tag Arbitration Protocol for RFID Systems," in *Proc. of IEEE Infocom* 2009.
[19] D. R. Hush and C. Wood, "Analysis of Tree Algorithms for RFID Arbitration," in *Proc. of IEEE ISIT*, 1998.
[20] P. Flajolet, G. N. Martin, "Probabilistic Counting Algorithms for Data Base Applications," *Journal of Computer and System Science*, 1985.
[21] Information Technology Automatic Identification and Data Capture Techniques: Radio Frequency Identification for Item Management Air Interface, *International Standard ISO 18000-6*, Nov. 2003.
[22] D. Holcomb, W. Burleson, K. Fu, "Power-up SRAM State as an Identifying Fingerprint and Source of True Random Numbers," *IEEE Transactions on Computers*, Accepted for future publication.
[23] C. Qian, H. Ngan, Y. Liu, L. M. Ni, "Cardinality Estimation for Large-scale RFID Systems", in *IEEE Transactions on Parallel and Distributed Systems*, Sep, 2011.
[24] V. Sarangan, M. R. Devarapalli, S. Radhakrishnan, "A Framework for Fast RFID Tag Reading in Static and Mobile Environments," in *Elsevier Computer Networks*, 2008.
[25] M. Bolic, D. Simplot-Ryl, I. Stojmenovic, *RFID Systems: Research Trends and Challenges*, Wiley, 2010.
[26] D. Simplot, M. Latteux, and R. Kalinowski, "An Adaptive Anti-Collision Protocol for Smart Labels," *LIFL/Gemplus contribution to Joint ISO/IETC 18000-3 Work Group*, 2001
[27] M.-K. Yeh, J.-R. Jiang and S.-T. Huang, "Adaptive Splitting and Pre-Signaling for RFID Tag Anti-Collision," *Computer Communications Journal*, 2009.
[28] L. Yang, J. Han, Y. Qi, C. Wang, Y. Liu, Y. Chen, and X. Zhong, "Revisiting Tag Collision Problem in RFID Systems," in *Proc. of IEEE ICPP* 2010.
[29] L. Yang, J. Han, Y. Qi, C. Wang, T. Gu, and Y. Liu, "Season: Shevling Inteference and Joint Identification in Large-scale RFID Systems," in *Proc. of IEEE Infocom* 2011,
[30] C. C. Tan, B. Sheng, and Q. Li, "How to Monitor for Missing RFID Tags," in *Proc. of IEEE ICDCS*, 2008
[31] T. Li, S. Chen, Y. Ling, "Identifying the Missing Tags in a Large RFID System," in *Proc. of ACM MobiHoc*, 2010.
[32] H. Han, B. Sheng, C. C. Tan, Q. Li, W. Mao, and S. Lu, "Counting RFID Tags Efficiently and Anonymously," in *Proc. of IEEE Infocom*, 2010.
[33] T. Li, S. Wu, S. Chen, and M. Yang, "Energy Efficient Algorithms for the RFID Estimation Problem," in *Proc. of IEEE Infocom*, 2010.
[34] Y. Zheng, M. Li, and C. Qian, "PET: Probabilistic Estimating Tree for Large-Scale RFID Estimation," in *Proc. of IEEE ICDCS*, 2011
[35] R. Rivest, "The MD5 Message-Digest Algorithm," in *RFC 1321*, 1992

PLACE PHOTO HERE

Chen Qian Chen Qian is a Ph.D. candidate at Department of Computer Science, the University of Texas at Austin. He received the BS degree in computer science from Nanjing University, China, in 2006, the MPhil degree in computer science and engineering from the Hong Kong University of Science and Technology, in 2008. His research interests include computer networking, distributed systems, and pervasive computing. He has published over 10 research papers in a number of conferences and journals including ACM SIGMETRICS, IEEE ICDCS, IEEE PerCom, and IEEE Transactions on Parallel and Distributed Systems. He is a student member of IEEE and ACM.

PLACE PHOTO HERE

Yunhuai Liu Dr. Yunhuai Liu received his PhD degree in Computer Science and Engineering from Hong Kong University of Science and Technology in 2008. From 2008 to 2010 he worked in Hong Kong University of Science and Technology as a research assistant professor. In the year 2010, he joined Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences as an associate professor. And in the year 2011, he joined the Research Center of Internet of Things, Third Research Institute of Ministry of Public Security located in Shanghai, China. His research interests include wireless sensor networks, cognitive radio networks, and extreme-scale datacenter and data networks. His research papers have been published in many prestigious conferences and journals such as ACM Mobicom, IEEE INFOCOM, IEEE ICDCS, IEEE ICPADS, and IEEE TPDS, IEEE TMC. The paper titled ”opportunity-based topology control in wireless sensor networks” obtained the only Best Paper Award in ICDCS 2008 (1 out of 638). He is an IEEE and ACM member.

PLACE PHOTO HERE

Hoilun Ngan Raymond Hoilun Ngan got his Ph.D. from the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. He received his BEng degree in Computer Engineering (First Class Honors) and MPhil degree in Computer Science, also from HKUST in 2003 and 2005 respectively.

PLACE PHOTO HERE

Lionel M. Ni Lionel M. Ni is Chair Professor in the Department of Computer Science and Engineering at the Hong Kong University of Science and Technology (HKUST). He also serves as the Special Assistant to the President of HKUST and Dean of HKUST Fok Ying Tung Graduate School. A fellow of IEEE, Dr. Ni has chaired over 30 professional conferences and has received six awards for authoring outstanding papers.