2010 International Conference on Distributed Computing Systems

ASAP: Scalable Identification and Counting for Contactless RFID Systems

Chen Qian†, Yunhuai Liu‡, Hoilun Ngan‡, Lionel M. Ni‡
‡*Department of Computer Science and Engineering*
Hong Kong University of Science and Technology,
Email: {yunhuai, cpeglun, ni}@cse.ust.hk
†*Department of Computer Science*
The University of Texas at Austin
Email: cqian@cs.utexas.edu

***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 mobile and with large numbers, two classical MAC layer collision-arbitration protocols, slotted ALOHA and Tree-traversal, 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 low-latency RFID identification and has stable performance for massive RFID networks. Theoretical analysis and experimental evaluation show that ASAP outperforms most existing collision-arbitration solutions. ASAP is efficient for both small and large deployment of RFID tags, in terms of time and energy cost. Hence it can benefit dynamic and large-scale RFID systems.**

***Keywords*-RFID; ALOHA protocol; Collision arbitration;**

I. Introduction

The Radio Frequency Identification (RFID) is a short-range radio communication technology that has been widely used in many applications such as identity recognition, localization/tracking [1] [2] [3] [4], and population (cardinality) counting [5] [6] [7]. 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 [8]. Contactless RFID systems are also called RFID Networks.

Restricted by the simple structure, however, RFID tags cannot run CSMA-like protocols to avoid link layer collisions. Two classical collision arbitration protocols are commonly used for RFID networks, namely slotted ALOHA [9] and Tree Traversal [10]. 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* [11] [12]. The desired identification protocol for RFID networks should be time-efficient and scalable.

Consider such an application of RFID systems: Hong Kong International Airport (HKIA), one of the largest airports with the second largest volume of annual cargo transportations in the world, is one of the earliest RFID system consumers. Every year, HKIA loses more than $20 million to rectify misplaced and mis-transported bags as every mis-location costs $100 in average to relocate the object. As early as August 2005, HKIA began to adopt the RFID technology in its logistic management system, which 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 [13] [14], which is beyond the scope of this paper.

Slotted ALOHA is an intuitive solution, but it does not appear to be scalable. On the other hand, Tree-traversal (Query Tree) has stable performance, but too many reader queries lower the system efficiency. Protocols that combine estimation and identification together are also designed to make identification more efficient [5] [6] [15] [16]. 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. A scalable arbitration protocol with stable and high efficiency is desirable for dynamic RFID systems.

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

1063-6927/10 $26.00 © 2010 IEEE
DOI 10.1109/ICDCS.2010.84

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, compared with the existing identification protocols for 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 precisely*. 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 [5] [6] [7].

The rest of this paper is organized as follows. In Section II, we discuss the system model and some existing works for RFID identification. Section III presents our basic ideas and the detailed protocol design. We show the theoretical analysis in Section IV, and present the experimental performance evaluation in Section V. We conclude our work in Section VI, and indicate possible future work directions.

II. 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.

A. 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. We denote the time cost for a query and a listening slot by t_q and t_s respectively. The time cost for switching from one mode to another is t_{switch}. During a period, let n, Q, S, N_t and W be the number of tags, queries, listening slots, tag replies and switches, respectively. We define the system time-efficiency R_t for every time unit t_u, as follows,

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

We define an energy-efficiency metric with the notations named similarly to the time metric.

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

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 [17] [18] [15]. For example, in [15] 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.

B. 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 [9]. 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 un-identified 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 can prove

Theorem 1: The highest system efficiency happens when $l = n$.

Also, 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 [10] [17] [18] [19] 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 [20], 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,

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

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 and independent from the tag cardinality.

We formally define the concept of cardinality-insensitive as follows:

Definition: *An RFID arbitration protocol is* ***cardinality-insensitive*** *if for any cardinality the time (energy) efficiency falls in the range* $[a - \delta, a + \delta]$*, where* δ *is a relatively small value compared with* a*.*

In our model, we use $\delta = 2\%$.

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 [9] [10]. 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.

Based on Theorem 1, an estimation of tag cardinality can help to improve the efficiency of collision arbitration. Recently several tag estimation schemes are proposed [5] [6]. Bonuccelli *et al.* designed a protocol Tree Slotted ALOHA (TSA) [15] 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 128 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 [16] enhances the stability of TSA by dynamically apply the previous estimated value to adjust the next estimation. It, however, still requires there are several identifiable slots. In cases like n equal to several thousand, the estimation converges very slowly too. Therefore TSA and DTSA are both cardinality-sensitive.

MSS, designed by Namboodiri and Gao [18], 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 [17] 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 [19] is proposed to improve the efficiency for RFID tags with different ID distributions.

III. 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.

A. Geometric Splitting-based Estimation

The *Geometric Splitting-based Estimation* (GSE) has first been proposed in [6]. 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,

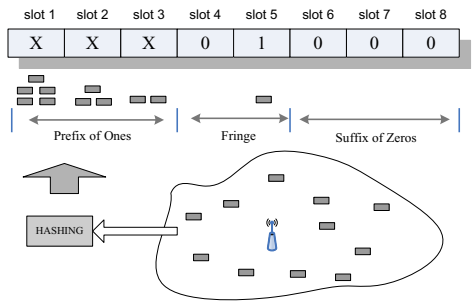

Figure 1. An example of GSE

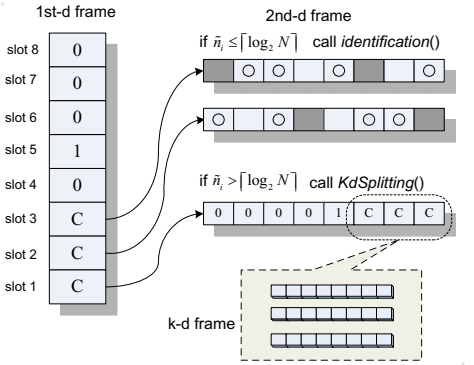

Figure 2. An example of ASAP

we have n tags that are placed into l slots with 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. Thus, the k-th bit in the merged 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 [21] and [6].

B. 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. With 32-slot frames, ASAP is stable for up to $2^{32} = 4,294,967,296$ 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 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. 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 [15] [16] 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 affects the stability.

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.

C. Enhanced Estimation in ASAP

Note that the GSE proposed in [6] 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 present an enhanced version of GSE in this section.

We first extend GSE to estimate the numbers of tags for subsets $s_1, s_2, ...$, 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.

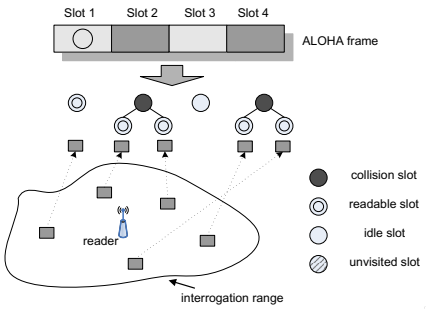

Figure 3. An example of identification in ASAP

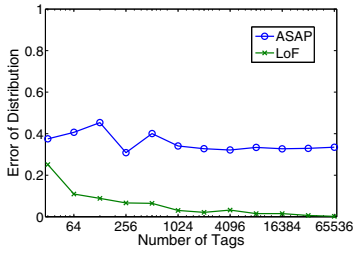

Figure 4. The Error of Geometric Distribution in Real Applications

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 [6], 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 (leftmost 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 $\log n \ll 32$.

D. 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 [22]. 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 ASAP, we set the threshold as 32. 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.

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 [6]. 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., Message-Digest algorithm 5 (MD5) or Secure Hash Algorithm (SHA-1), 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.

In LoF, the version of GSE proposed in [6], the estimation accuracy highly depends on tag ID distribution. Non-uniform distributed tags will affect the performance of LoF. Our Kd-Splitting() scheme alleviates the influence of ID distribution to the accuracy of GSE, because all IDs are re-distributed by each MD5 hash function. Figure 4 shows the (empirical) error of geometric distribution for LoF and ASAP.

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 [23].

IV. 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.

A. Time Cost Analysis

Consider the model described in Section II.A. 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,

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

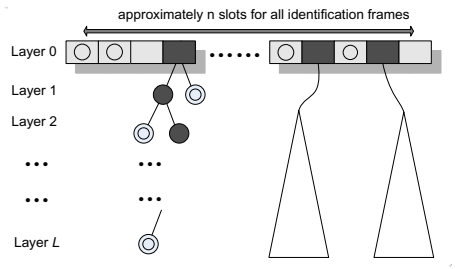

Figure 5. Identification Tree

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

$$Q_s(n) = \frac{n\log_2\log_2 n}{M} + \frac{n}{M} \tag{10}$$

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,

$$S_s(n) < \frac{n(\log_2 n + 4)}{M} < n \tag{11}$$

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. As illustrated in Fig. 5, 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,

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

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

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

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]

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

Thus we derive,

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

Since $2^L \ll n$,

$$S_i(n) = n+o(\log_2 n) \tag{12}$$

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,

$$Q(n) = Q_s(n)+Q_i(n) < \frac{n\log_2\log_2 n+n}{M}+\log_2 n \tag{13}$$

$$S(n) = S_s(n)+S_i(n) < 2n+\frac{n(\log_2 n+4)}{M} \tag{14}$$

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

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

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

$$\begin{aligned}T_{ASAP}(n) &= Q(n)t_q+S(n)t_s+W(n)t_{switch}\\ &= \left(\frac{n\log_2\log_2 n+n}{M}+\log_2 n\right)(t_q+2t_{switch})\\ &\quad +\left(2n+\frac{n(\log_2 n+4)}{M}\right)t_s\end{aligned} \tag{15}$$

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 II.C. ASAP has a remarkably higher efficiency, which is also cardinality-insensitive as we will show in Section V. Based on the analysis in [15], 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.

B. Energy Cost Analysis

Energy efficiency is another important metric [18]. 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 skip the detailed derivation. Similar to the time cost, ASAP has stable and low energy cost.

C. 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 IV.A, 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.

[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.

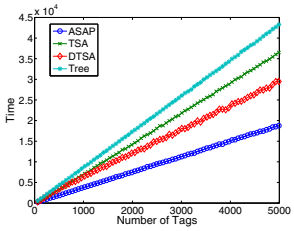

Figure 6. Comparison in time units, $T(n)$

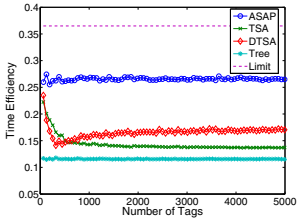

Figure 7. Comparison in time efficiency

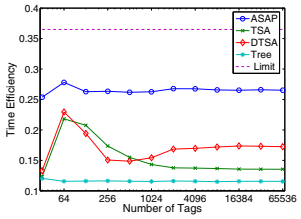

Figure 8. Time efficiency with larger range of cardinalities

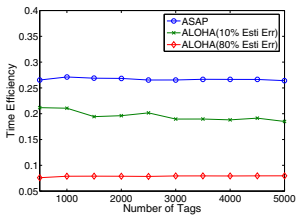

Figure 9. ASAP vs. slotted ALOHA and tag no. known with error

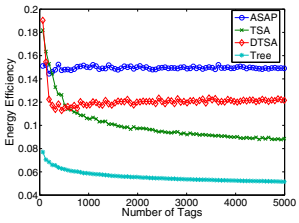

Figure 10. Comparison in energy efficiency

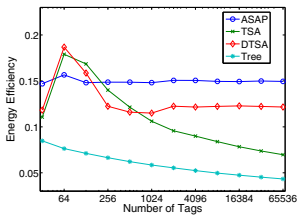

Figure 11. Energy efficiency with larger range of cardinalities

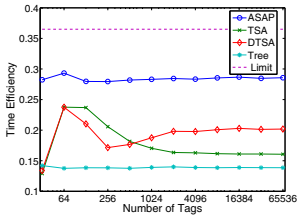

Figure 12. Time efficiency under Model B

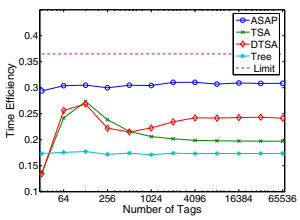

Figure 13. Time efficiency under Model C

V. Experimental Evaluation

We have conducted extensive experiments to verify the performance of ASAP. 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 hardware, 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 [15] and DTSA [16], 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 II and IV. We also tried different performance models in Section V. C. Accuracy is defined as the fraction of tags to be recognized by the reader.

A. Simulation Setup

Different amounts (32 to 65536) of tags will be deployed in a rectangular region with dimension 10m×10m. Due to the limited computation ability, we are not able to simulate billions of tags. Note that the arbitration itself is very efficient, but to simulate billions 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. The tag IDs are 32-bits wide. 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 II.

B. Performance of ASAP

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

Figures 6 and 7 plot the time cost and time efficiency for identifying different amounts of tags (from 64 to 4992). In Fig. 7 we also add the upper bound efficiency of slotted ALOHA as the limit of $R_t(n)$, which is impossible to achieve unless one can know the precise cardinality in advance and discard all query cost. The figures indicate that the performance of ASAP is always better than those of existing approaches. To better characterize these protocols, we also plot the time efficiency in a larger cardinality range in Fig. 8. From Fig. 7 and 8, 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 sharply 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. 9, ASAP is more efficiency than simple ALOHA.

[3]The values of TSA and DTSA shown here are different from those in the original papers [15] [16], because the query cost was not considered there.

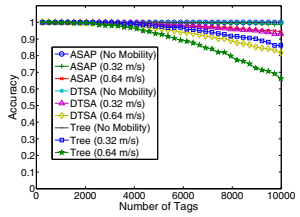

Figure 14. Accuracy comparison with tag mobility

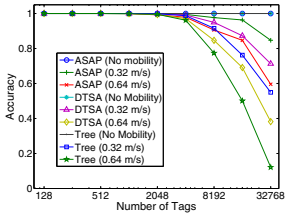

Figure 15. Accuracy comparison with tag mobility and exponentially increasing cardinalities

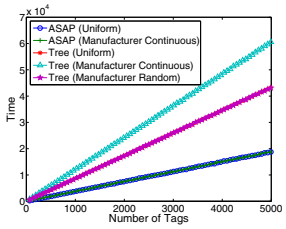

Figure 16. The Impact of ID Distribution

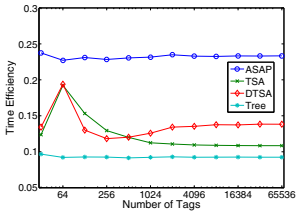

Figure 17. Time efficiency for lossy links

The comparison of energy efficiency is plotted in Fig. 10 and 11. 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.

C. 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 12 and 13 show the results of model B and C respectively. ASAP still performs stably and more efficiently than other approaches. Due to the space constraints we do not show the results for energy cost, which is similar to those in Fig. 10 and 11.

D. 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. 14 and 15.

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.

E. Impact of Tag ID Distribution

In this section, we will consider possible scenarios that tag IDs are not uniformly distributed. We simulate these scenarios by proposing two tag ID distributions, namely manufacturer-random and manufacturer-continuous. Tag IDs with the former distribution have a common prefix and their suffixes are uniformly distributed among the tags. Tag IDs with the later distribution will also have a common prefix but their suffixes are continuous assigned. In this distribution, the actual length of the common prefix could be much longer depending on the tag size. Figure 16 show the total operating time. Manufacture-continuous distribution affect the performance of Tree algorithm as queries are issued based on tag IDs. ASAP and DTSA, however, are stable for any ID distributions.

F. 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 17 shows the result where the average link quality is 0.8. The efficiency of ASAP is still close to 0.25.

G. 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. 6 to 15, 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. Though the simple Tree-traversal is not as efficient as ASAP and DTSA, it is still necessary because all of ASAP, TSA and DTSA use Tree algorithm as a function block.

VI. 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 work can also benefit other ALOHA-based networks such as Satellite communications networks.

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 Grant HKUST617908, China NSFC Grant 60933011, the National Basic Research Program of China (973 Program) under Grant No. 2006CB303000, the National Science and Technology Major Project of China under Grant No. 2009ZX03006-001, and the Science and Technology Planning Project of Guangdong Province, China under Grant No. 2009A080207002. Chen Qian is currently supported by National Science Foundation grant 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] L. M. Ni, Y. Liu, Y. C. Lau, and A. Patil, "LANDMARC: Indoor Location Sensing Using Active RFID," in *Proc. of IEEE PerCom*, 2003.

[2] D. Zhang, J. Ma, Q. Chen, and L. M. Ni, "An RF-based System for Tracking Transceiver-free Objects," in *Proc. of IEEE PerCom*, 2007.

[3] 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.

[4] M. Zuniga and M. Hauswirth, "Hansel: Distributed Localization in Passive RFID Environments," in *Proc. of IEEE SECON* 2009.

[5] M. Kodialam and T. Nandagopal, "Fast and Reliable Estimation Schemes in RFID Systems," in *Proc. of ACM Mobicom*, 2006.

[6] C. Qian, H.-L. Ngan, and Y. Liu, "Cardinality Estimation for Large-scale RFID Systems," in *Proc. of IEEE PerCom*, 2008.

[7] 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

[8] 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.

[9] L. G. Roberts, "Aloha Packet System with and without Slots and Capture," *ACM SIGCOMM Computer Communication Review*, vol. 5, pp. 28-42, 1975.

[10] J. I. Capetanakis, "Tree algorithms for packet broadcast channels," *IEEE Trans. on Information Theory*, vol. IT-25, pp. 505-515, 1979.

[11] D. Simplot-Ryl, I. Stojmenovic, A. Micic and A. Nayak, "A Hybrid Randomized Protocol for RFID Tag Identification", *Sensor Review* 2006.

[12] 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

[13] 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.

[14] Q. Yao, Y. Qi, J. Han, J. Zhao, X. Li, and Y. Liu, "Randomizing RFID Private Authentication", in *Proc. of IEEE PerCom* 2009.

[15] M. A. Bonuccelli, F. Lonetti, F. Martelli, "Tree Slotted Aloha: a New Protocol for Tag Identification in RFID Networks," *Elsevier Ad Hoc Networks* 2007.

[16] G. Maselli, C. Petrioli, C. Vicari, "Dynamic Tag Estimation for Optimizing Tree Slotted Aloha in RFID Networks,", in *Proc. of ACM MSWIM* 2008.

[17] J. Myung and W. Lee, "Adaptive Splitting Protocols for RFID Tag Collision Arbitration," in *Proc. of ACM MobiHoc*, 2006.

[18] V. Namboodiri and L. Gao, "Energy-Aware Tag Anti-Collision Protocols for RFID Systems," in *Proc. of IEEE PerCom*, 2007.

[19] 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.

[20] D. R. Hush and C. Wood, "Analysis of Tree Algorithms for RFID Arbitration," in *Proc. of IEEE ISIT*, 1998.

[21] P. Flajolet, G. N. Martin, "Probabilistic Counting Algorithms for Data Base Applications," *Journal of Computer and System Science*, 1985.

[22] Information Technology Automatic Identification and Data Capture Techniques: Radio Frequency Identification for Item Management Air Interface, *International Standard ISO 18000-6*, Nov. 2003.

[23] D. Holcomb, W. Burleson, K. Fu, "Power-up SRAM State as an Identifying Fingerprint and Source of True Random Numbers," to appear at *IEEE Transactions on Computers*.