*Sensors & Transducers, Vol. 189, Issue 6, June 2015, pp. 61-70*

**Sensors & Transducers**


# A Pre-Detection Based Anti-Collision Algorithm with Adjustable Slot Size Scheme for Tag Identification

**Chiu-Kuo LIANG, Yuan-Cheng CHIEN**
Department of Computer Science and Information Engineering, Chung Hua University,
707, Sec. 2, WuFu Rd., 30012, Hsinchu, Taiwan
Tel.: +886(3)5186411, fax: +886(3)5186416
E-mail: ckliang@chu.edu.tw

*Received: 5 June 2015 /Accepted: 26 June 2015 /Published: 30 June 2015*

**Abstract:** One of the research areas in RFID systems is a tag anti-collision protocol; how to reduce identification time with a given number of tags in the field of an RFID reader. There are two types of tag anti-collision protocols for RFID systems: tree based algorithms and slotted aloha based algorithms. Many anti-collision algorithms have been proposed in recent years, especially in tree based protocols. However, there still have challenges on enhancing the system throughput and stability due to the underlying technologies had faced different limitation in system performance when network density is high. Particularly, the tree based protocols had faced the long identification delay. Recently, a Hybrid Hyper Query Tree ($H^2QT$) protocol, which is a tree based approach, was proposed and aiming to speedup tag identification in large scale RFID systems. The main idea of $H^2QT$ is to track the tag response and try to predict the distribution of tag IDs in order to reduce collisions. In this paper, we propose a pre-detection tree based algorithm, called the Adaptive Pre-Detection Broadcasting Query Tree algorithm (APDBQT), to avoid those unnecessary queries. Our proposed APDBQT protocol can reduce not only the collisions but the idle cycles as well by using pre-detection scheme and adjustable slot size mechanism. The simulation results show that our proposed technique provides superior performance in high density environments. It is shown that the APDBQT is effective in terms of increasing system throughput and minimizing identification delay. *Copyright © 2015 IFSA Publishing, S. L.*

**Keywords:** RFID, Tag anti-collision, Hybrid query tree, Pre-detection query tree.

## 1. Introduction

Radio Frequency IDentification (RFID) is an automatic technology that guarantees to advance modern industrial practices in object identification and tracking, asset management, and inventory control [1]. Recently, several identification systems such as barcodes and smart cards are incorporated for automatic identification and data collection. However, these systems have several limits in read rate, visibility, and contact. RFID systems are a matter of great concern because they provide fast and reliable communication without requiring physical sight or touching between readers and tags.

One of the areas of research is the speed with which a given number of tags in the field of RFID readers can be identified. For fast tag identification, anti-collision protocols, which reduce collisions and identify tags irrespective of occurring collisions, are required [1-7]. There are two types of collisions: reader collisions and tag collisions. Reader collisions indicate that when neighboring readers inquire a tag concurrently, so the tag cannot respond its ID to the inquiries of the readers. These collision problems can be easily solved by detecting collisions and communicating with other readers. Tag collisions occur when multi tags try to respond to a reader simultaneously and cause the reader to identify no tag. For low-cost passive RFID tags, there is nothing to do except response to the inquiry of the reader. Thus, tag anti-collision protocols are necessary for improving the cognitive faculty of RFID systems.

In general, the tag anti-collision techniques can be classified into two categories, aloha-based and tree-based protocols. Aloha-based approaches use time slot to reduce collision probability, such as Framed-Slotted aloha algorithm [1, 8], dynamic framed slotted aloha algorithm [5]. Tags randomly select a particular slot in the time frame, load and transmit its identification to the reader. Once the transmission is collided, tags will repeatedly send its id in next interval of time to make sure its id is successfully recognized. Aloha-based protocols can reduce the collision probability. However, they have the tag starvation problem that a particular tag may not be identified for a long time. For the consideration of performance, when number of RFID tag increased, the tag collision rate will be increased as well; this may result a low tag recognition rate.

The tree-based schemes use a data structure similar to a binary search algorithm, such as binary tree splitting protocol [3], query tree (QT) algorithm, and tree working algorithm [6, 9]. An RFID reader consecutively communicates with tags by sending prefix codes based on the query tree data structure. Only tags in the reader's interrogation zone and of which ID match the prefix respond. The reader can identify a tag if only one tag respond the inquiry. Otherwise the tags responses will be collided if multiple tags respond simultaneously.

Although tree based protocols deliver 100 % guaranteed read rates, but they have relatively long identification delay. Recently, a hybrid query tree protocol (HQT) [10] was proposed and aiming to reduce transmission overhead by using 4-ary search tree mechanism and slotted backoff mechanism, in order to speed up tag identification and to increase the overall read rate and throughput in large-scale RFID systems. The main idea of the HQT technique is to reduce the number of collisions during the identification phase. In the 4-ary search tree mechanism, the prefix string of a collided query will be extended by 2-bits next time, unlike of 1-bit in the QT protocol. This way, collisions can be reduced substantially. Furthermore, the HQT protocol was aiming to reduce the idle cycles by using a slotted backoff mechanism. When a tag responds to a reader, it sets its backoff timer using a part of its ID. If there is a collision (multiple tags respond), the reader can partially deduce how the IDs of tags are distributed and potentially reduce unnecessary queries.

Based on the HQT protocol, a $H^2QT$ protocol [11] was proposed and aiming to reduce the idle cycles and improve the performance of tag identification. Although the $H^2QT$ technique performs better than the HQT technique in reducing the number of idle cycles, it still has some idle cycles, which cannot be reduced during the tag identification process. In this paper, we proposed a pre-detection based protocol, called Adaptive Pre-Detection Broadcasting Query Tree (APDBQT) protocol, to eliminate those unnecessary idle cycles. To evaluate the performance of our proposed technique, we have implemented our proposed APDBQT scheme along with previous proposed HQT method. The experimental results show that the proposed technique presents significant improvement in most circumstance.

The remainder of this paper is organized as follows: Related work is discussed in Section 2. In Section 3, the tree based tag identification algorithm is introduced as preliminary of this study. In Section 4, our proposed algorithm, the APDBQT algorithm is presented. Performance comparisons and analysis of the proposed technique will be given in Section 5. Finally, in Section 6, some concluding remarks are made.

## 2. Related Work

Many research results for collision avoidance have been presented in literature. Frequency Division Multiple Access (FDMA), Code Division Multiple Access (CDMA), Time Division Multiple Access (TDMA) and Carrier Sense Multiple Access (CSMA) [20] are four basic access methods to categorize MAC-level protocols. Standard collision avoidance protocols like RTS-CTS [21] cannot be applied in RFID systems since when a reader broadcasts an RTS, all tags in the read range need to send back CTS to the reader. It then requires another collision avoidance mechanism for CTS, and it will make the protocol more complicated. Techniques for resolving RFID reader collision problems are usually proposed as reader anti-collision techniques or tag anti-collision solutions. The Colorwave [22] is a scheduling-based approach to prevent RFID readers from simultaneously transmitting signal to an RFID tag. The Colorwave is used as a distributed anti-collision system based on TDMA in RFID network. The Pulse protocol [23] is referred as a beacon broadcast and CSMA mechanism. Readers periodically send a "beacon" during communication with tags in separate control channels. If a reader receives a beacon, the residual back-off timer will be stored and kept until the next coming chance. This process is expected to achieve the fairness among all readers. A coverage-based RFID reader anti-collision mechanism was proposed in [24]. Kim, *et al.* [24] presented a localized clustering coverage protocol for solving reader collision problems occurring among homogeneous RFID readers. In [25], Cha, *et al.* proposed two ALOHA-based algorithms with a Tag Estimation Method (TEM) for speedup object identification in RFID systems. Hsu, *et al.* [27] proposed a two phase dynamic modulation (TPDM) scheme to coordinate communications between multiple readers and tags. In TPDM, the scheduling is divided into two phases, the regional scheduling and the hidden terminal scheduling.

The existing tag identification approaches can be classified into two main categories, the Aloha-based anti-collision scheme [1, 4, 5, 8, 12] and the tree-based scheme [3, 6, 7, 9]. RFID readers in the former scheme create a frame with a certain number of time slots, and then add the frame length into the inquiry message sent to the tags in its vicinity. Tags response the interrogation based on a random time slot. Because collisions may happen at the time slot when two or more tag response simultaneously, making those tags could not be recognized. Therefore, the readers have to send inquiries contiguously until all tags are identified. As a result, Aloha-based scheme might have long processing latency in identifying large-scale RFID systems [4]. In [1], Vogt, *et al.* investigated how to recognize multiple RFID tags within the reader's interrogation ranges without knowing the number of tags in advance by using framed Aloha. A similar research is also presented in [13] by Zhen, *et al.* In [12], Klair, *et al.* also presented a detailed analytical methodology and an in-depth qualitative energy consumption analysis of pure and slotted Aloha anti-collision protocols. Another anti-collision algorithm called enhanced dynamic framed slotted aloha (EDFSA) is proposed in [5]. EDFSA estimates the number of unread tags first and adjusts the number of responding tags or the frame size to give the optimal system efficiency.

In tree-based scheme, such as ABS [3], Improved Bit-by-bit Binary-Tree (IBBT) [14] and IQT [15], RFID readers split the set of tags into two subsets and labeled them by binary numbers. The reader repeats such process until each subset has only one tag. Thus, the reader is able to identify all tags. The adaptive memoryless tag anti-collision protocol proposed by Myung, *et al.* [2] is an extended technique based on the query tree protocol. Choi, *et al.* [14] also proposed the IBBT algorithm in Ubiquitous ID system and evaluate the performance along three other old schemes. The IQT protocol [15] is a similar work approach by exploiting specific prefix patterns in the tags to make the entire identification process. In [16], Zhou, *et al.* consider the problem of slotted scheduled access of RFID tags in a multiple reader environment. They developed centralized algorithms in a slotted time model to read all the tags. With the fact of NP-hard [16], they also designed approximation algorithms for the single channel and heuristic algorithms for the multiple channel cases. In [26], Hsu, *et al.* presented a threshold jumping technique to improve the system throughput in large-scale RFID systems. Their main idea is to limit the number of collisions. When the number of collisions exceeds a predefined threshold, it reveals that the tag density in RF field is too high. To avoid unnecessary enquiry messages, the prefix matching will be moved to next level of the query tree, alleviating the collision problems. However, in irregular or imbalanced RFID systems, inefficient situation may happen. Liang, *et al.* [17] proposed an intelligent wrap-around scan (iWAS) jumping scheme to improve the situation. The main idea of their proposed scheme is that, instead of adding one bit for prefix matching at next level as collision occurred in query tree algorithm, they analyze the possible ways for causing the collision and take appropriate actions at next level. In their observation, there are four possible ways to cause the collision. Once the collision conditions are detected by the reader, the scanning process will be moved to the next level. As a result, the collision queries will be avoided.

Although tree based schemes have advantage of implementation simplicity and better response time compare with the Aloha based ones, they still have challenges in decreasing the identification latency. In this paper, we present a pre-detection tree based tag identification technique aims to coordinate simultaneous communications in large-scale RFID systems, to speedup minimize tag identification latency and to increase the overall read rate and throughput. Simulation results show that our proposed technique outperforms previous techniques.

## 3. Tree-based Anti-Collision Schemes

In this section, we present three tree-based anti-collision techniques, namely the Query Tree algorithm (QT) [3], the HQT algorithm [10], and the $H^2QT$ algorithm [11], that are most related to our work.

### 3.1. Query Tree Algorithm

The query tree algorithm (QT) uses binary splitting strategy to identify tags. A reader transmits the *k*-length prefix. Then tags send from $(k+1)^{th}$ bit to the end bit of tag IDs if the first *k* bits of tag IDs are the same as the prefix. Also, if the received tag IDs collide, the extended prefix attached '0' or '1' to the prefix is retransmitted. Furthermore, if there is no collision, the reader identifies one of the tags. Fig. 1 shows an example of the query tree scheme.

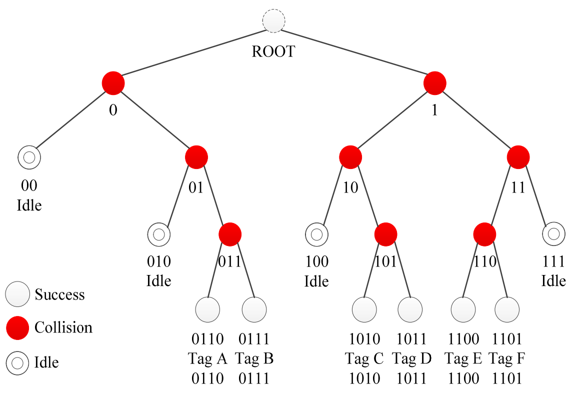


**Fig. 1.** Example of query tree algorithm.

### 3.2. Hybrid Query Tree Algorithm

In the environment with high tags density, collision may happen very frequently while using the query tree algorithm, and due to that, a lot of query time will be wasted. By using the 4-ary search query tree mechanism, HQT can enable the prefix to increase two bits at a time from 1 bit. In this way, some collisions occurred in QT protocol can be reduced in HQT protocol. However, the drawback of the 4-ary search tree mechanism is the increasing number of idle cycles. To resolve this problem, HQT protocol introduces the slotted backoff mechanism. The slotted backoff mechanism is a technique that makes tags respond to the transmit prefix after waiting a certain time, instead of immediately respond. When a tag responds to a reader, it sets its backoff timer using a part of its ID. The backoff time of each tag is determined from the 2-bits, which follow the prefix of tag ID identical to the query prefix string. For example, tags do not defer their response if it is '00'. If it is '01', '10', or '11', tags will defer 1, 2, or 3 backoff time slots until they respond to the reader, respectively. Fig. 2 shows the operation of the slotted backoff mechanism in HQT algorithm.

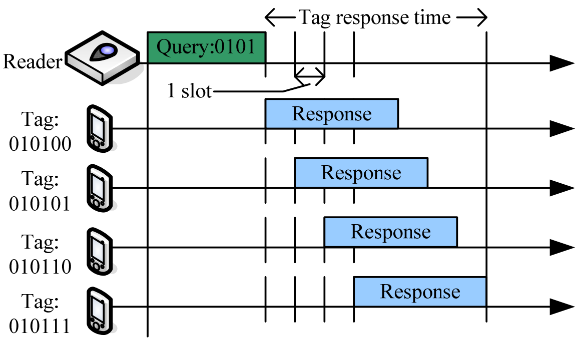


**Fig. 2.** The slotted backoff mechanism in HQT.

### 3.3. Hybrid Hyper Query Tree Algorithm

The main problem in HQT algorithm is that those idle cycles between busy slots cannot be reduced. To resolve the problem, the $H^2QT$ algorithm uses a different slotted backoff mechanism. The backoff time of each tag is determined from the 3-bits, which follows the prefix of tag ID identical to the prefix. Unlike the mechanism used in HQT, the $H^2QT$ counts the number of '1' in the following 3-bits and uses this number as the selected time slot for tags to respond. Fig. 3 shows the tag selecting its response slot based on tag ID.

Fig. 4 depicts an example of the query tree structure of identifying 5 tags with 6-bits ID length using $H^2QT$ algorithm. The process of the identification is as follows: First of all, the reader sends request command with the empty-prefix to the tags.

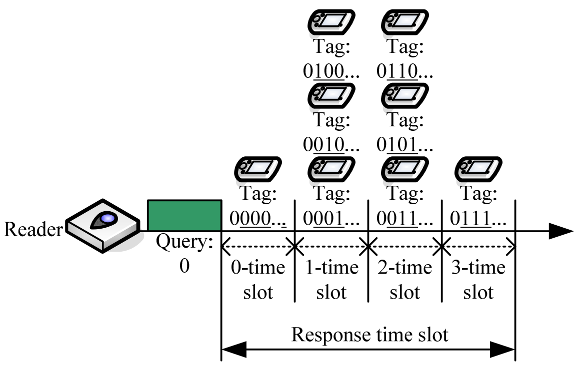


**Fig. 3.** The $H^2QT$ algorithm.

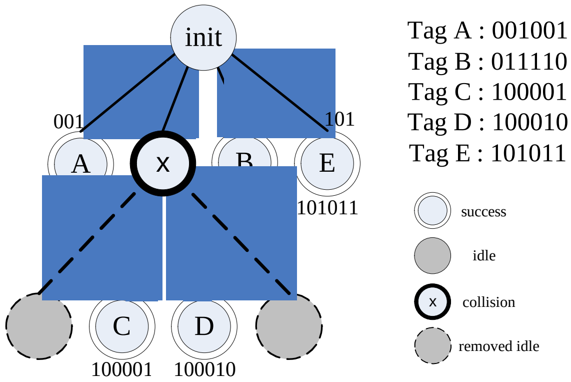


**Fig. 4.** Example of $H^2QT$ algorithm.

In this case, tags A, C and D will delay one time slot to respond since the first 3-bits of their tag IDs contain only one '1', as shown in Fig. 5. Similarly, tags B and E will delay two time slots to respond due to the number of '1' in the first 3-bits if their tag IDs is 2. In this case, since no tag responds immediately, it means that there is no tag whose first 3-bits of their tag IDs match '000'. Therefore, there is no need for reader to send the prefix string '000'. Similarly, since no tag responds after 3 time slot delay, the reader does not need to send the prefix string '111'. Therefore, the idle cycles can be eliminated.

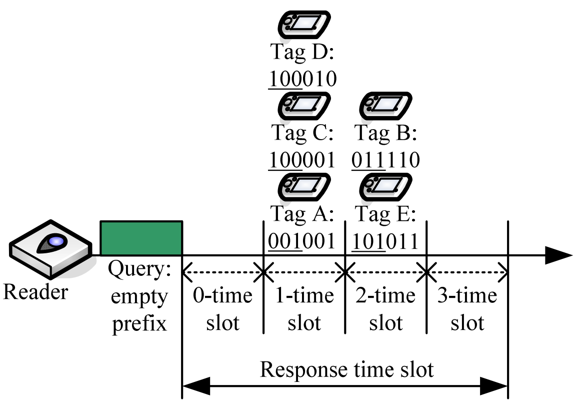


**Fig. 5.** The response of tags after reader's empty prefix request in Fig. 4.

Next, the reader receives tag IDs from tags A, C and D after one time slot delay. At this moment, the reader is aware that the pattern of the first 3-bits of tags A, C and D is ‘X0X’, in which ‘X’ represents a collision bit. Thus, the reader recognizes that the first 3-bits of tag A, C and D may be ‘001’ or ‘100’, which will be added into the queue for re-transmission. Similarly, the reader is aware that the bit pattern of the first 3-bits of tags B and E is ‘XX1’ after two time slots delay. Thus, the reader will put prefix strings ‘011’ and ‘101’ into the queue for re-transmission.

Next, the reader sends the request command with prefix string ‘001’. At this moment, only tag A responds after 1 time slot delay. In this case, tag A is identified by the reader. Table 1 summarizes the detail steps of communication between the reader and the tags with the example shown in Fig. 4.

**Table 1.** Communication steps of Fig. 4.

| Step | HQT | | $H^2QT$ | |
|---|---|---|---|---|
| | **Broadcast** | **Status** | **Broadcast** | **Status** |
| **1** | empty | Collision | empty | Collision |
| **2** | 00 | Identify Tag A | 001 | Identify Tag A |
| **3** | 01 | Identify Tag B | 100 | Collision |
| **4** | 10 | Collision | 011 | Identify Tag B |
| **5** | 1000 | Collision | 101 | Identify Tag E |
| **6** | 1001 | Idle | 100001 | Identify Tag C |
| **7** | 1010 | Identify Tag E | 100010 | Identify Tag D |
| **8** | 100001 | Identify Tag C | | |
| **9** | 100010 | Identify Tag D | | |

## 4. Proposed Schemes

Recall that, in $H^2QT$ algorithm, the idle cycles can be reduced substantially. However, there still have some collision time slots. As a result, the reader has to spend more time slots to resolve the collisions. Due to that, it will take more time to complete the tag identification process. In this paper, we first introduced our previously proposed Pre-Detection based Broadcasting Query Tree (PDBQT) algorithm [18-19] which uses a pre-allocated time slots for collecting the distribution of tags’ id to eliminate the collision time slots and idle cycles. Then, we dynamically change the size of the pre-allocated time slots. As a result, the performance of pre-detection scheme can be improved significantly.

### 4.1. Pre-Detection Based Scheme

The main idea of our scheme is to realize the precise distribution of tag IDs. Once the distribution of tag IDs has been obtained, the reader broadcasts such message to tags and each tag is aware of the exact time slot to response. As a result, tags respond to the reader in different time slots and collisions can be avoided. Furthermore, since each tag realizes its corresponding time slot to respond, no empty time slot exists. To do so, we therefore propose a pre-detection based scheme to collect the distribution from tag IDs.

In our proposed pre-detection based algorithm, after the reader sends the request command to tags, the operations during the tag response period can be partitioned into three phases: the pre-detection phase, the broadcasting phase and the tag response phase, as shown in Fig. 6. The purposes of the three-phase design can be explained as follows: In pre-detection phase, the reader can realize the distribution of tag IDs by collecting the responses from tags. Then, in broadcasting phase, the reader can send the distribution information to tags so that each tag is aware of the time slot to respond its ID to reader. Finally, in the tag response phase, the responses from tags are arranged into a sequence of time slots so that collisions and empty slots can be avoided. It should be noticed that, in our proposed pre-detection scheme, a pre-defined parameter *m* has been used to indicate the size of the pre-detection time slot.

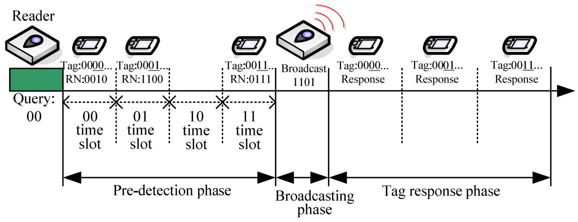


**Fig. 6.** The tag response cycle of our proposed scheme.

After the reader sends a query string (i.e. a prefix string) and a parameter *m* to tags and then waits for tags to response, in the pre-detection phase, each tag whose tag ID matches with the prefix string sent from the reader will respond on the pre-detection time slot according to its following *m*-bits of its tag ID. In order to collect the response information from tags, we allocate $2^m$ short time slots in pre-detection phase for tags to respond and the time slots can be numbered as the binary representation of *m* bits respectively. Fig. 6 shows an example with $m = 2$. The pre-detection phase in Fig. 6 consists of four time slots, namely the ‘00’, ‘01’, ‘10’, and ‘11’ time slots respectively. We also adapt the *m*-array search tree mechanism such that each tag whose tag ID matches with the prefix string sent from the reader will respond on the corresponding time slot depending on the following *m*-bits of its tag ID. Then each tag responds a 4-bits random number (RN) to reader instead of the whole tag ID. The reasons for tags of using 4-bits random numbers to respond are as follows: First, it can reduce the time for reader to realize the distribution of tag IDs, compared with the response of whole tag IDs. Second, the status of each time slot can be precisely identified with high probability. If no tag responds in a time slot, then the reader can correctly identify such time slot as an idle cycle, which can be eliminated during the tag response phase. If only one tag responds, the reader can also correctly identify such time slot as a successful cycle. Therefore, the reader will allocate a time slot to receive the response from that tag in the tag response phase. If more than one tag respond, since the tags respond 4-bits random numbers, a collision cycle can be identified by the reader by checking the received different random numbers. Although, there still has some chance for a reader to receive the same random number from different tags, however, the probability of successful collision detection is very high. Therefore, by using our pre-detection mechanism, the distribution of tag IDs can be correctly obtained with high accuracy.

Meanwhile, the reader monitors and records the response status from tags in each time slot during the pre-detection phase. The reader uses a ‘0’-bit to represent the time slot when no tag responds or more than one tag respond. On the other hand, the reader uses a ‘1’-bit to represent the time slot when only one tag responds. Therefore, after the pre-detection phase, the reader can use an *m*-bits string to represent the status of $2^m$ time slots in the pre-detection phase. Then, during the broadcasting phase, the reader broadcasts the *m*-bits string to tags and by receiving the binary bit string, each tag can realize the exact time slot to respond by counting the number of ‘1’ in the received binary bit string from the start bit to its corresponding bit. Then, the tag can respond its tag ID to reader in the tag response time slot by finding the correct time slot to respond. For example, in Fig. 6, the tag which responds on the ‘11’ time slot in the pre-detection phase can realize that it can only send its tag ID on the third time slot in the tag response phase since it receives the binary bit string ‘1101’ sent from the reader and there are three ‘1’s from the beginning to its corresponding ‘1’.

Consider an example as shown in Fig. 7. In Fig. 7, there are 5 tags with 6-bits of tag IDs, ‘001001’, ’011110’, ‘100001’, ‘100010’ and ‘101011’, respectively. The identification process of our pre-detection scheme with parameter $m = 2$ is as follows: First of all, the reader sends the request command with the empty-prefix to the tags. In this case, all tags respond to this request command and the time slot for a tag to respond is depending on the first 2-bits of its tag ID. In this example, tag A will respond in ‘00’ time slot, tag B will respond in ‘01’ time slot, tags C, D, and E will respond in ‘10’ time slot, and no tag responds in ‘11’ time slot, as shown in Fig. 7(a). Suppose that the random numbers for tags A, B, C, D, and E to respond are ‘1110’, ‘1100’, ‘0011’, ‘1010’, and ‘0111’, respectively. It can easily be seen that, since there is only one tag response for both ‘00’ and ‘01’ time slots, the reader will mark both time slots as ‘1’ to indicate the successful status. Furthermore, since it has more than one tag responses in ‘10’ time slot, the reader will mark ‘10’ time slot as ‘0’ to indicate the collision status. It should be noticed that as the reader recognize the collision time slot, the corresponding prefix bit string will be added into a queue for further requesting. In this example, the ‘10’ bit string will be added into the queue. After the pre-detection phase, the reader will mark all time slots as ‘1100’ and broadcast it to tags. After tags receive the message, tags A and B realize their own time slot to respond. Therefore, tags A and B will be identified subsequently. In the meantime, tags C, D, and E recognize that the status of their time slot is ‘0’, which means that they do not need to send their tag IDs to reader at that time slot, as shown in Fig. 7(a). After identifying tags A and B, the reader sends another request command from queue, which is the ‘10’ bit string as shown in Fig. 7(b). In this cycle, tag E is identified and bit string ‘1000’ is added into queue. In the last round, as shown in Fig. 7(c), tags C and D can be identified.

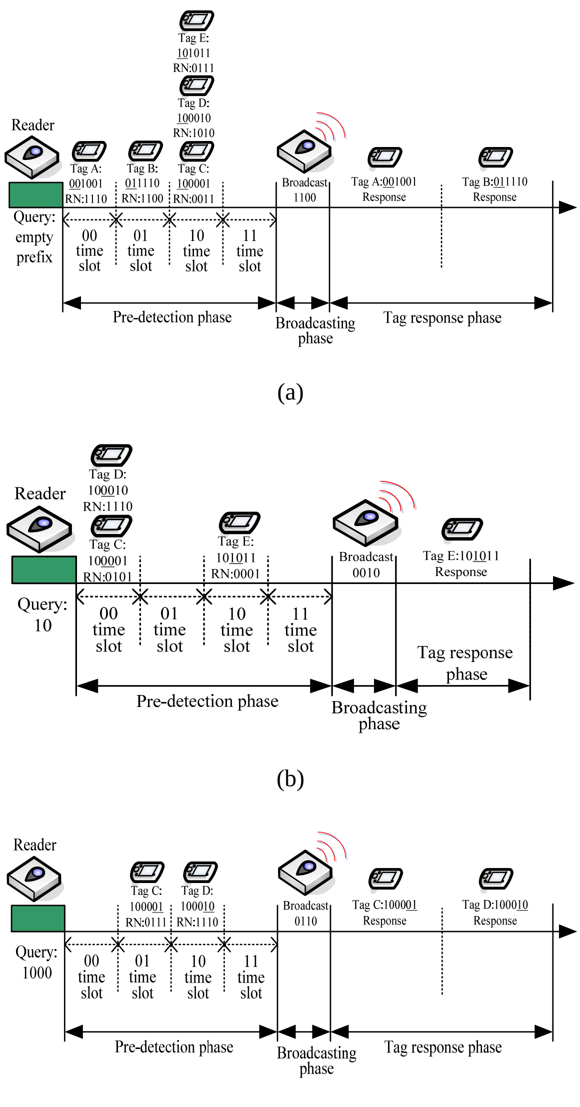


**Fig. 7.** Example of pre-detection based algorithm.

### 4.2. Adaptive Pre-Detection Based Scheme

Consider the situation that the tag density is low. Then, by using previous pre-detection scheme, it can be seen that although the collision time slots is reduced, but the number of idle time slot is increased as well in the pre-detection phase. On the other hand, in the situation with high tag density, the number of collision time slot is increased in the pre-detection phase while the number of idle time slot is decreased. This is due to the fixed size of the time slots in pre-detection phase.

From the above observation, we proposed an improved pre-detection based scheme, namely the Adaptive Pre-Detection Broadcasting Query Tree (APDBQT) scheme. The idea of APDBQT scheme is dynamically changing the size of the pre-detection time slots by adjusting the parameter *m* after each query cycle. More specifically, after the tag response phase, the reader can re-calculate the value of *m* by knowing the idle slot ratio in the previous pre-detection phase. The idle slot ratio is defined as the number of idle time slots to the total number of time slots in the previous pre-detection phase. Thus, as the idle slot ratio gets high, the density of tag distribution gets low and vice versa. Therefore, the reader may allocate more time slots in pre-detection phase by letting *m* larger in the next cycle of identification when the reader is aware of that the density of tags is getting higher. Similarly, the reader may allocate less time slots in pre-detection phase by letting *m* smaller in the next cycle of identification when the reader is aware of that the density of tags is getting lower. In APDBQT scheme, we define the density of tags to be high if the idle slot ratio is smaller than 50 % and low otherwise.

Fig. 8 depicts another example of the process of identifying 9 tags with 6-bits of tag IDs, namely A:'000010', B:'001001', C:'001010', D:'011001', E:'100001', F:'100110', G:'101011', H:'110011', and I:'110111', respectively, by using APDBQT protocol. First of all, the reader sends the request command with the empty-prefix and *m* = 2 to the tags. In this case, all tags respond to this request command and the time slot for a tag to respond is depending on the first 2-bits of its tag ID. In this example, tags A, B and C will respond in '00' time slot, tag D will respond in '01' time slot, tags E, F, and G will respond in '10' time slot, and tags H and I will respond in '11' time slot, as shown in Fig. 8(a). Suppose that the random numbers for tags to respond are all different. It can easily be seen that, since there is only one tag response for '01' time slot, the reader will mark the time slot as '1'. Furthermore, since it has more than one tag responses in '00', '10', and '11' time slots, the reader will mark all of these time slots as '0'. As a result, there is no idle slot in the pre-detection phase. This means that the value of *m* will be increased to be 3 as the reader realizes that the tag density is high. In the next query cycle, since *m* = 3, the reader will allocate $2^3$ = 8 time slots in the pre-detection phase as shown in Fig. 8(b). After finishing the pre-detection phase, the reader realizes that the idle slot ratio is greater than 50 %. This means that the value of *m* will be decreased as the tag density is low. Therefore, in the next query cycle, the value of *m* will be changed into 2 and four time slots will be allocated in the pre-detection phase as shown in Fig. 8(c). Finally, in the last round, the value of m will be changed into 3 and eight time slots will be allocated in the pre-detection phase as shown in Fig. 8(d).

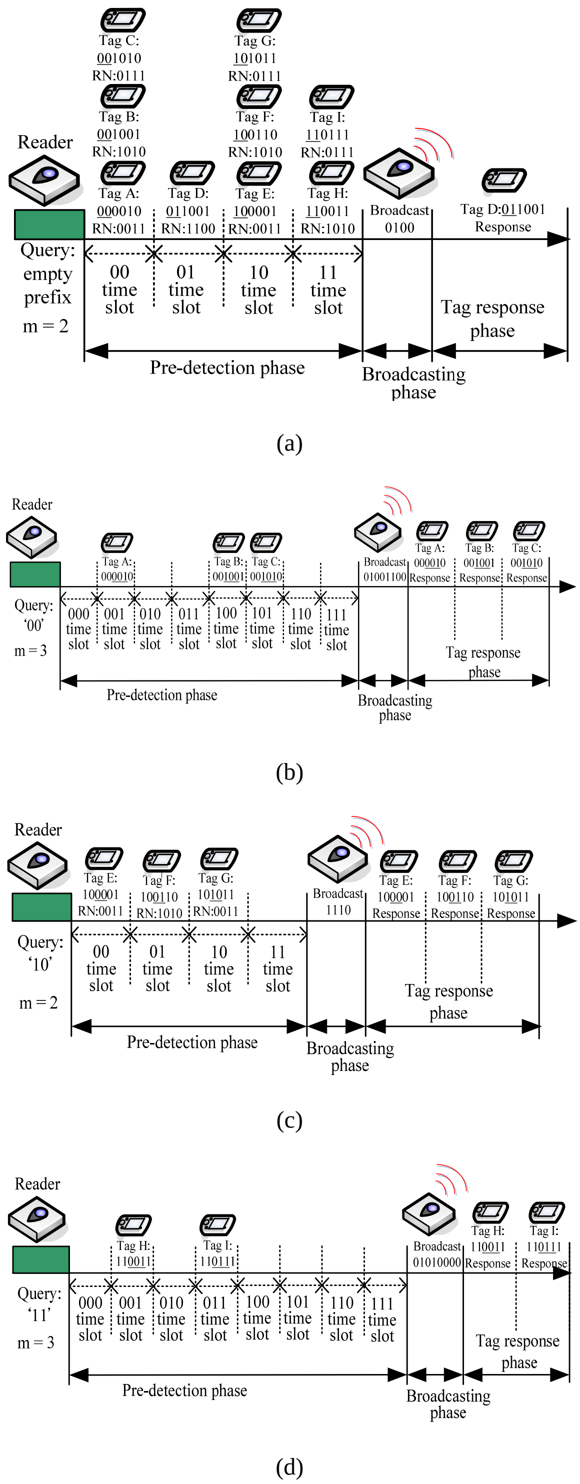


**Fig. 8.** Example of our APDBQT algorithm.

### 4.3. Comparison of Identification Methods

To facilitate the understanding of the performance of our proposed algorithm, we compare the identification process between previous $H^2QT$ and our proposed APDBQT algorithms by using the example in Fig. 8.

Table 2 shows the required prefixes and steps for identifying all 9 tags by using different methods. In Table 2, the $H^2QT$ scheme needs 10 steps to complete the identification process while in our proposed APDBQT scheme, only 4 steps are needed. Thus, our proposed APDBQT protocol reduces identification overhead efficiently and achieves better performance than $H^2QT$ scheme.

**Table 2.** Communication steps of Fig. 8.

| Step | $H^2QT$ | | APDBQT | |
|---|---|---|---|---|
| | **Broadcast** | **Status** | **Broadcast** | **Status** |
| **1** | empty | Identify Tag A | empty | Identify Tag D |
| **2** | 001 | Collision | 00 | Identify Tags A, B and C |
| **3** | 100 | Collision | 10 | Identify Tags E, F and G |
| **4** | 011 | Identify Tag D | 11 | Identify Tags H and I |
| **5** | 101 | Identify Tag G | | |
| **6** | 110 | Identify Tags H and I | | |
| **7** | 001001 | Identify Tag B | | |
| 8 | 001010 | Identify Tag C | | |
| 9 | 100001 | Identify Tag E | | |
| 10 | 100110 | Identify Tag F | | |

## 5. Performance Evaluation

To evaluate the performance of the proposed technique, we have implemented the PDBQT and APDBQT schemes along with the hybrid query tree protocol (HQT). Fig. 9 compares the number of queries to identify different number of RFID tags. Because the tag id is set 16 bits length, density = 10 % means that there are $2^{16} \times 10$ % = 6554 tags and density=20 % means that there are $2^{16} \times 20$ % = 13107 tags, and so on. All tags are randomly generated in a uniform distribution manner. Basically, the proposed techniques PDBQT and APDBQT can reduce the amount of inquiry messages. As we expected, the APDBQT outperforms the PDBQT. Although the PDBQT and APDBQT methods do not improve the inquiry messages very much in low density network, when density increasing, both of the proposed methods present significant improvements to the hybrid query tree protocol.

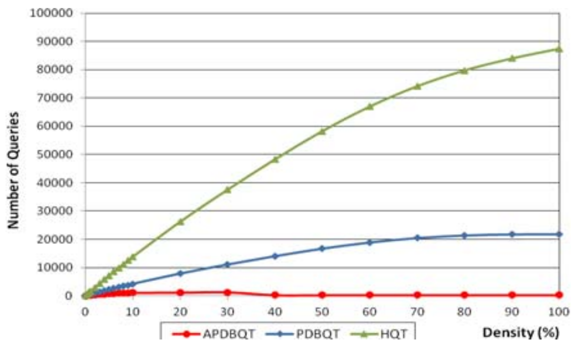


**Fig. 9.** Performance comparison of the APDBQT, PDBQT and HQT in terms of the no. of queries required.

Fig. 10 examines the number of idle slots generated to identify different number of tags. Both of the PDBQT and APDBQT methods outperforms the hybrid query tree protocol (HQT). The hybrid query tree protocol (HQT) has some idle slots during identification process. Especially, HQT will generate the most amount of idles as the density is around 50 %. It is obvious that PDBQT and APDBQT will eliminate all idle slots regardless the density of RFID network.

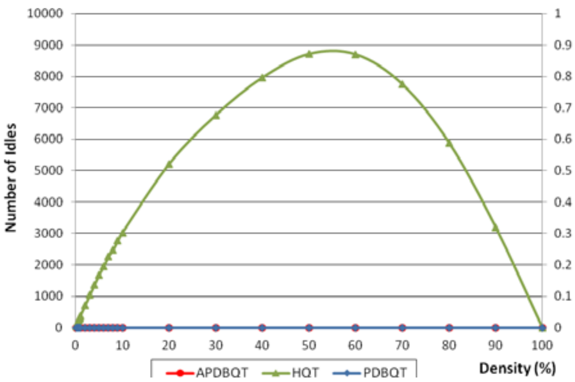


**Fig. 10.** Performance comparison of the APDBQT, PDBQT and HQT in terms of the no. of idle slots generated.

Fig. 11 compares the number of collision slots generated by each protocol during the tag identification process. Basically, the proposed techniques PDBQT and APDBQT can reduce the amount of collision slots. It is obvious that lower collision slots resulting less inquiry cost. Both of the PDBQT and APDBQT methods outperforms the hybrid query tree protocol (HQT). As we expected, the APDBQT outperforms the PDBQT. It can be observed that both of the PDBQT and APDBQT methods can avoid a lot of collision slots due to the pre-detection mechanism. As a result, when density increasing, the APDBQT scheme could present noticeable improvement as compare to the HQT.

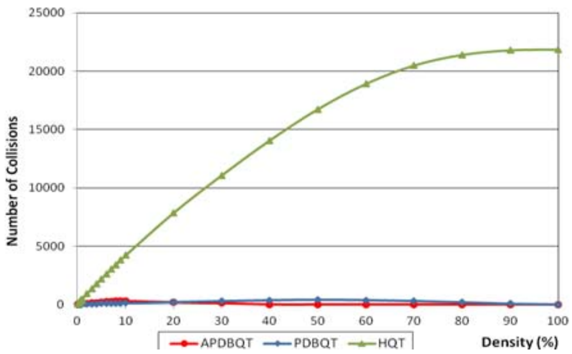


**Fig. 11.** Performance comparison of the APDBQT, PDBQT and HQT in terms of the no. of collision slots generated.

In Fig. 12, the last experiment was conducted to show the average delay time needed for one-tag identification to identify different number of RFID tags. Both of the PDBQT and APDBQT methods outperform the hybrid query tree protocol (HQT). The average delay time of HQT protocol requires more than 1 *ms* for both low and high density of RFID tags. It is obvious that PDBQT and APDBQT will reach less than 1 *ms* average delay time for one-tag identification regardless the density of tags. In particular, when density increasing, the proposed APDBQT method presents significant improvement to hybrid query tree protocol. The average delay time of APDBQT protocol requires less than 0.2 *ms* when the density of tags exceeds 50 %. As we expected, the APDBQT outperforms the PDBQT.

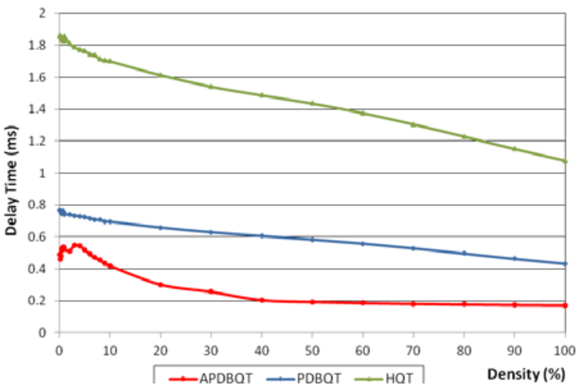


**Fig. 12.** Performance comparison of the APDBQT, PDBQT and HQT in terms of the average delay time for one-tag identification.

## 6. Conclusions

With the emergence of wireless RFID technologies, identifying high density RFID tags is a crucial task in developing large scale RFID systems. Due to the nature of large scale RFID systems, many collisions may occur during the process of tag identification. In this paper, we have presented a previously proposed pre-detection tree-based tag identification technique along with a pre-detection time slot adjustable technique for minimizing tag identification cost. By adjusting the size of pre-detection time slot and using the random numbers for tags to respond in the pre-detection phase, not only the length of pre-detection phase can be minimized but also the collision slots can be reduced. As a result, the efficiency of tag identification can be significantly improved. To evaluate the performance of proposed techniques, we have implemented the PDBQT and APDBQT techniques along with the hybrid query tree protocol (HQT). The experimental results show that the proposed techniques provide considerable improvements on the latency of tag identification. It is shown that the APDBQT technique is a nearly collision-free tag identification scheme such that the iteration overhead can be efficiently reduced. It is also shown that the APDBQT is effective in terms of increasing system throughput and efficiency.

## References

[1]. H. Vogt, Efficient object identification with passive RFID tags, in *Proceedings of the 1st International Conference on Pervasive Computing* (*Pervasive'02)*, 2002, pp. 98–113.

[2]. J. Myung, W. Lee, An adaptive memoryless tag anticollision protocol for RFID networks, in *Proceedings of the IEEE INFOCOM'05*, March 2005, Poster Session.

[3]. J. Myung, W. Lee, J. Srivastava, Adaptive binary splitting for efficient RFID tag anti-collision, *IEEE Communication Letter*, Vol. 10, No. 3, 2006, pp. 144-146.

[4]. C. Law, K. Lee, K. Y. Siu, Efficient Memoryless Protocol for Tag Identification, in *Proceedings of the International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, 2000, pp. 75-84.

[5]. S. Lee, S. Joo, C. Lee, An enhanced dynamic framed slotted ALOHA algorithm for RFID tag identification, in *Proceedings of the 2nd Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'05)*, July 2005, pp. 166-172.

[6]. J. I. Capetanakis, Tree algorithms for packet broadcast channels, *IEEE Trans. Information Theory*, Vol. 25, Sep. 1979, pp. 505-515.

[7]. F. Zhou, D. Jin, C. Huang, M. Hao, Optimize the Power Consumption of Passive Electronic Tags for Anti-collision Schemes, in *Proceedings of the 5th International Conference on ASIC*, Vol. 2, October 2003, pp. 1213-1217.

[8]. J. Park, M. Chung, T.-J. Lee, Identification of RFID Tags in Framed-Slotted ALOHA with Robust Estimation and Binary Selection, *IEEE Communications Letters*, Vol. 11, No.5, 2007, pp. 452-454.

[9]. B. Feng, L. J. Tao, G. J. Bo, D. Z. Hua, Id-binary tree stack anti-collision algorithm for RFID, *IEEE Transactions on Computers*, 2006, pp. 207–212.

[10]. J. Ryu, H. Lee, Y. Seok, T. Kwon, Y. Choi, A Hybrid Query Tree Protocol for Tag Collision Arbitration in RFID systems, in *Proceedings of the IEEE International Conference on Communications (ICC'07)*, 2007, pp. 24–28.

[11]. T. H. Kim, S. J. Lee, A Hybrid Hyper Tag Anti-Collision Algorithm in RFID System, *ICACT*, 2009, pp. 1276-1281.

[12]. D. K. Klair, K. W. Chin, R. Raad, An investigation into the energy efficiency of pure and slotted aloha based RFID anticollision protocols, in *Proceedings of the IEEE WoWMoM'07*, Finland, 18–21 June 2007, pp. 1-4.

[13]. B. Zhen, M. Kobayashi, M. Shimizui, Framed aloha for multiple RFID objects Identification, *IEICE Trans. on Comm., E88-B(3)*, 2005, pp. 991–999.

[14]. H. S. Choi, J. R. Cha, J. H. Kim, Improved Bit-by-bit Binary Tree Algorithm in Ubiquitous ID System, *Advances in Multimedia Information Processing - PCM 2004, Lecture Notes in Computer Science,* Vol. 3332, 2005, pp 696-703.

[15]. A. Sahoo, S. Iyer, N. Bhandari, Improving RFID System to Read Tags Efficiently, IIT Bombay, KRSIT Technical Report, June 2006.

[16]. Z. Zhou, H. Gupta, S. R. Das, X. Zhu, Slotted Scheduled Tag Access in Multi-Reader RFID Systems, in *Proceedings of the IEEE International Conference on Networks Protocols (ICNP)*, 2007, pp. 61–70.

[17]. C. K. Liang, Chih-Kang Tseng, Hsin-Mo Lin, An Efficient Tag Identification Scheme in RFID Systems, in *Proceedings of the 6th International Conference on Sensor Technologies and Applications (SENSORCOMM'12)*, Rome, Italy, 19-24 August 2012, pp. 77-82.

[18]. C. K. Liang, Yuan-Cheng Chien, Chih-Hung Tsai, A Pre-Detection Query Tree Tag Anti-Collision Scheme in RFID Systems, in *Proceedings of the Seventh International Conference on Sensor Technologies and Applications (SENSORCOMM'13)*, Barcelona, Spain, 26-29 August 2013, pp. 51-56.

[19]. C. K. Liang, Y. C. Chien, An Adaptive Pre-Detection Based RFID Tag Anti-Collision Scheme, in *Proceedings of the Conference on SensorNet (SENSORNETS'15)*, Paris, France, 11-13 February 2015, pp. 69-76.

[20]. S. Jain, S. R. Das, Collision avoidance in multi-reader RFID networks, in *Proceedings of the WiNTECH 2006*, Los Angeles, USA, 2006.

[21]. J. L. Sobrinho, R. de Haan, J. M. Brazio, Why RTS-CTS is not your ideal wireless LAN multiple access protocol, in *Proceedings of the IEEE Wireless Communications and Networking Conference*, 2005, pp. 81–87.

[22]. J. Waldrop, D. W. Engels, S. E. Sarma, Colorwave: an anticollision algorithm for the reader collision problem, in *Proceedings of the IEEE International Conference on Communications,* 2003, pp. 1206–1210.

[23]. M. B. Shailesh, I. Sridhar, PULSE: A Mac Protocol for RFID Networks, in *Proceedings of the International Workshop on Embedded and Ubiquitous Computing (EUC'05)*, Nagasaki, Japan, 2005.

[24]. J. Kim, S. Kim, D. Kim, W. Lee, E. Kim, Low-Energy Localized Clustering: An Adaptive Cluster Radius Configuration Scheme for Topology Control in Wireless Sensor Networks, in *Proceedings of the IEEE Vehicular Technology Conference (VTC)*, 2005, pp. 2546–2550.

[25]. J. R. Cha, J. H. Kim, Dynamic Framed Slotted ALOHA Algorithm using Fast Tag Estimation method for RFID System, in *Proceedings of the 3rd IEEE Consumer Communications and Networking Conference (CCNC'06)*, Vol. 2, Las Vegas, USA, 2009, pp. 768-772.

[26]. C. H. Hsu, H. C. Chao, J. H. Park, Threshold jumping and wrap-around scan techniques toward efficient tag identification in high density RFID systems, *Information Systems Frontiers*, Volume 13, Issue 4, September 2011, pp. 471-480.

[27]. C. H. Hsu, J. H. Park, C. H. Yu, Adaptive Scheduling Mechanisms toward Intelligent Services in Mobile and Wireless RFID Environments, in *Proceedings of the 22nd International Conference on Advanced Information Networking and Application (AINA'08)*, Okinawa, Japan, 25-28 March 2008, pp. 1567-1572.