An Adaptive Combination Query Tree Protocol for Tag Identification in RFID Systems

Yi Jiang and Ruonan Zhang

***Abstract*—A reader must be able to identify tags as quickly as possible, so tag anti-collision is a significant issue for the RFID system. In this paper, we propose a novel tag anti-collision protocol (ACQT), which is suitable for a large mobile tags environment. Based on a 3-ary tree, it has the optimal capacity to identify for tree-based protocols. Using a combination query tree, it can solve the problem of not being able to generate a 3-ary tree when the length of a tag's ID is not a multiple of 3. The joining and leaving strategies can efficiently be applied for tag mobility. The simulation results show that the protocol we present can achieve better performance than previous protocols by decreasing identification delay, collision cycles and idle cycles.**

***Index Terms*—3-ary tree, adaptive query tree, tag identification, RFID.**

I. Introduction

THE RFID system is a contactless automatic identification system attracting more attention recently. A reader recognizes objects through wireless communications with tags, which are attached to objects and have a unique ID [1]. When a reader sends a query, it has multiple tags to respond to in its communication range, and a collision has occurred. The reader must be able to identify tags as quickly as possible. Considered to be the limit of tags, the collision issue needs urgently to be solved.

To settle the collision issue successfully, several tag anti-collision protocols have been proposed, which can be grouped into 2 types: deterministic methods or probabilistic methods. The deterministic methods are based on a tree-based protocol, and the probabilistic methods are based on the ALOHA protocol whose performance is degraded with a large number of tags, so we mainly studied the tree based protocols. The QT protocol [2], based on a binary tree, has a memoryless character, which does not use the history of the prior queries. The HQT protocol [3] is based on a 4-ary tree which has better performance than QT, but it is not an optimal choice when compared to a q-ary tree. The literature [4] is the best tree-based protocol, but not all of the lengths of a tag's ID can be multiples of 3.

In this paper, we propose a novel tag anti-collision protocol ACQT (the Adaptive Combination Query Tree), which is suitable for large mobile tags with a 3-ary tree. It has the best capacity to identify tags. To deal with binary tag IDs, we use a conversion scheme to convert binary to 3-ary. We consider the anti-collision protocol from two aspects: If the length of a tag's ID is the multiple of 3, we use a query tree based 3-ary tree, if not, we use a combination query tree. To account for the mobility of tags, we propose the joining and leaving strategy to reduce the idle and collision cycles. The simulation results show that the protocol we present outperforms previous protocols by decreasing identification delay, collision cycles and idle cycles.

The remainder of the paper is organized as follows. We describe our proposed protocol in Section II. Section III analyzes the performance of our protocol. Finally, our concluding remarks are stated in section IV.

Manuscript received February 18, 2012. The associate editor coordinating the review of this letter and approving it for publication was C. Mitchell.

The authors are with the School of Electronics and Information, Northwestern Polytechnical University, Xi'an, Shanxi province, 710072 China (e-mail: peipeiv88nwpu@hotmail.com).

This work was supported by the Nature Science Foundation of Shanxi Province of China (2012JQ8005), the Basic Research Foundation of NWPU (JC20100214), and the E-star Foundation of the School of Electronics and Information in NWPU.

Digital Object Identifier 10.1109/LCOMM.2012.060112.120345

II. The Proposed ACQT Protocol

In a previous paper [5], the authors showed that a 3-ary tree is optimal and that in a q-ary tree, if q is greater than 3, the performance is decreased as q is increased. Thus, if the length of tag's ID is a multiple of 3, we can choose a 3-ary tree to deal with collisions; however, if it is not a multiple of 3, we select a combination of a binary tree and a 3-ary tree.

A. The Conversion of Binary to 3-ary

In this section, we give the formula for the conversion of binary to 3-ary. It is an extension of the protocol [4] which is based only on the specific bits. Our formula is suitable for converting any number of bits, except for those that are indivisible by 3.

We define L, l is the length of tag's ID with binary and ternary expression respectively, and L is the multiple of 3. Thus $l/2 = L/3$. We must find a way to convert L bits $(EPC)_2$ to l bits $(EPC)_3$. The $(EPC)_2$ is denoted by $x = \{x_1, x_2, ...x_i, ..., x_L\}$, $x_i \in \{0, 1\}$, and the $(EPC)_3$ is denoted by $y = \{y_1, y_2, ...y_j..., y_l\}$, $y_j \in \{0, 1, 2\}$. The relationship of $(EPC)_2$ and $(EPC)_3$ is described by (1):

$$x_i \times 2^2 + x_{i+1} \times 2^1 + x_{i+3} \times 2^0 = y_j \times 3^1 + y_{j+1} \times 3^0 \quad (1)$$

$$i = \{1, 4, 9...3k + 1\}, k \in \{1, 2, ...L/3 - 1\} and$$

$$j = \{1, 3, 5...2t + 1\}, t \in \{1, 2, ...l/2 - 1\}$$

From the above discussion, a conversion table can be created as shown in Table I.

TABLE I
THE CONVERSION OF BINARY TO 3-ARY

binary	000	001	010	011	100	101	110	111
3-ary	00	01	02	10	11	12	20	21

TABLE II
THE IDENTIFICATION PROCEDURE OF 6 TAGS WITH 6BITS

step	Reader query	Tag respond	Q	yQ1	yQ2	yQ3
1		Collision:1-6	0,1,2			
2	0	Collision:1-3	1,2,00,01,02		0	
3	1	Collision:4-5	2,00,01,02,10,11,12		0,1	
4	2	idetify:6	00,01,02,10,11,12	2	0,1	
5	00	idetify:1	01,02,10,11,12	2,00	0,1	
6	01	Collision:2-3	02,10,11,12,010,011,012	2,00	0,1,01	
7	02	idle	10,11,12,010,011,012	2,00	0,1,01	02
8	10	idetify:4	11,12,010,011,012	2,00,10	0,1,01	02
9	11	idetify:5	12,010,011,012	2,00,10,11	0,1,01	02
10	12	idle	010,011,012	2,00,10,11	0,1,01	02,12
11	010	idetify:2	011,012	2,00,10,11,010	0,1,01	02,12
12	011	idle	012	2,00,10,11,010	0,1,01	02,12,011
13	012	idetify:3		2,00,10,11,010,012	0,1,01	02,12,011

B. The Query Tree based 3-ary Tree

Based on the idea of the QT [2] and using the formula we presented in section A, we propose an identification process applying to 3-ary query trees. It will provide available information for the study of the adaptive protocol in the next section by recording the corresponding content in different variables.

A reader has a queue Q, which maintains queries for the current identification process. At the beginning of the process, the reader sends queries from Q to ask tags in order to recognize them by their responds, which includes three cases on communication between tags and reader.

1) Identify: Exactly one tag responds to the reader. If there are enough identification query, it can recognize all tags. The reader notes the query to a queue yQ_1.

2) Collision: Number of tags that respond to the reader is more than one. The reader is unable to recognize any tags. The reader notes the query to a queue yQ_2, then adds $\{0, 1, 2\}$ to the queries respectively, which can compose the new queries to execute next identification process, and bring them to the queue Q.

3) Idle: No tag responds. It is a waste that should be reduced. The reader notes the query to a queue yQ_3.

The reader deletes a query from Q if it receives the result as identify or idle. If it receives the result as a collision, it will add three new queries to Q, then delete the old one and transmit them. The tag identification process continues until Q is empty.

For example, we assume that there are six tags which have a tag ID with a length of 6bits, which are (000000), (001010), (001110), (011000), (100111), and (110111).The conversion of binary to 3-ary is executed. There are six tags with ternary, which are (0000), (0102), (0120), (1000), (1121), (2021). Table II lists the details of this identification procedure. Fig.1 is its corresponding ternary tree.

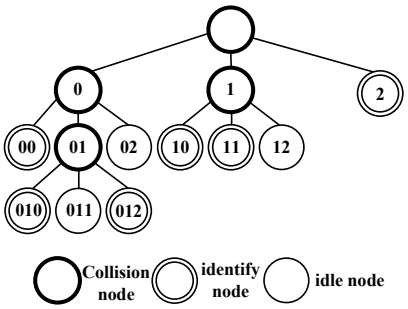


Fig. 1. The 3-ary tree to the identification procedure of six tags with six bits.

TABLE III
THE IDENTIFICATION PROCEDURE OF 6TAGS WITH 5BITS

step	Reader query	Tag respond	Q	yQ1	yQ2	yQ3
1		Collision:1-6	0,1,2			
2	0	Collision:1-3	1,2,00,01,02		0	
3	1	Collision:4-5	2,00,01,02,10,11,12		0,1	
4	2	idetify:6	00,01,02,10,11,12	2	0,1	
5	00	idetify:1	01,02,10,11,12	2,00	0,1	
6	01	Collision:2-3	02,10,11,12,010,011	2,00	0,1,01	
7	02	idle	10,11,12,010,011	2,00	0,1,01	02
8	10	idetify:4	11,12,010,011	2,00,10	0,1,01	02
9	11	idetify:5	12,010,011	2,00,10,11	0,1,01	02
10	12	idle	010,011	2,00,10,11	0,1,01	02,12
11	010	idle	011	2,00,10,11	0,1,01	02,12,010
12	011	Collision:2-3	0110,0111	2,00,10,11	0,1,01,011	02,12,010
13	0110	idetify:3	0111	2,00,10,11,0110	0,1,01,011	02,12,010
14	0111	idetify:4		2,00,10,11,0110,0111	0,1,01,011	02,12,010

C. The Combination Query Tree (CQT)

If L is not a multiple of 3, we select a combination query tree with binary tree and 3-ary tree. In such a circumstance, it can be divided into two parts to study.

- If the remainder after L is divided by 3 is 1 bit (binary), it can be ignored, and the method of query tree based on 3-ary tree can be used to identify all tags. After the above procedure, the one remainder tag to the same prefix, we can identify directly. Then, the two remainder tags to the same prefix, we can judge them by its last bit as 0 or 1.
- If the remainder after L is divided by 3 is 2 bits (binary), it can be dealt with by binary tree, after using the method of query tree based on 3-ary tree to identify all of the tags. If the parts of a multiple of 3 belonging to tags are different, the tags can be recognized without the other process, if not, the remainder bits corresponding to the tree has two branches to deal with.

In a second case, there are six tags which have tag ID lengths of 5bits, which are (00000), (00110), (00111), (01100), (10011), and (11001). In the first 3bits, we apply the method of 3-ary tree, and the remaining 2bits can use the method of binary tree.

The conversion of binary to 3-ary is executed. There are six tags with ternary and binary, which are (00 00), (01 10), (01 11), (10 00), (11 11), (20 01). Table III lists the details of this identification procedure. Fig.2 is its corresponding combination tree.

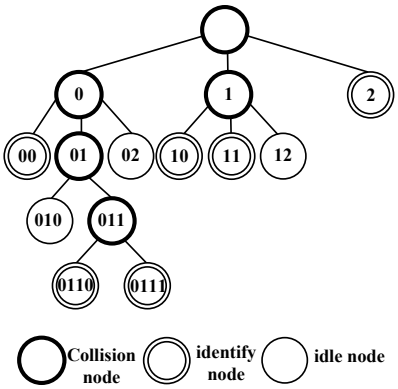


Fig. 2. The combination tree to the identification procedure of six tags with five bits.

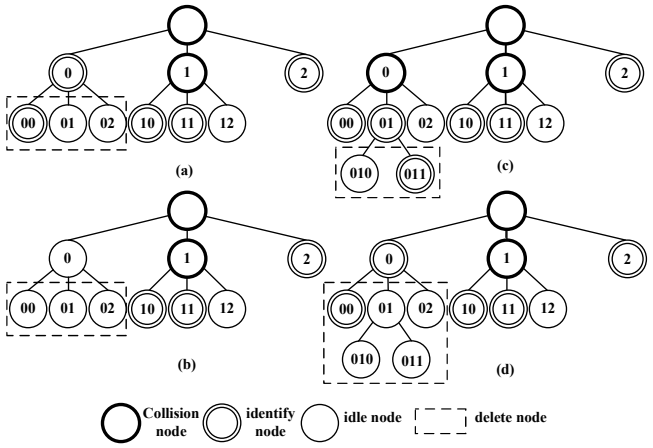


Fig. 3. The tree corresponding to the four cases.

D. The Adaptive Combination Query Tree (ACQT)

To consider the mobility of tags, we classify tags into staying tags, arriving tags, and leaving tags. All query procedures which are used to identify tags can be divided into identification cycles, collision cycles, and idle cycles.

1) Joining strategy: In the next process, there are some tags arriving. The collision cycles can be judged only once, because a collision has happened with staying tags, so it does not need to be judged again. Then, the idle cycles and the identification cycles are used for recognizing new tags, except for the identification cycles at the end of the tree, which have been used by staying tags. It does not need to judge from the root node of the tree as does the HQT [3]. The cases are categorized as follows:

- Identify: the query corresponding to the two kinds of cycles which sends to tags can obtain a readable response.
- Collision: the query corresponding to the two kinds of cycles which sends to tags can obtain more than one response. We use CQT to deal with it.

2) Leaving strategy: In the next process, there are some tags leaving, it makes abnormal queries as follows:

- A collision query has three child nodes belonging to a 3-ary tree, in which it has an identification node and two idle nodes. So, we can judge their father node as an identification node, and delete the three queries corresponding to the three child nodes.
- A collision query has three child nodes belonging to a 3-ary tree, in which it has three idle nodes. So we can judge their father node as also being an idle node and delete the three queries corresponding to the three child nodes.
- A collision query has two child nodes belonging to a binary tree, in which it has an identification node and an idle node. So, we can judge their father node as being an identification node, and delete the two queries corresponding to the two child nodes.
- A collision query has two child nodes belonging to a binary tree, in which it has two idle nodes. So we can judge their father node as also being an idle node and delete the two queries corresponding to the two child nodes. Based on father's brother node, we can judge whether to delete the father or not.

The tree corresponding to the four cases is depicted in Fig.3

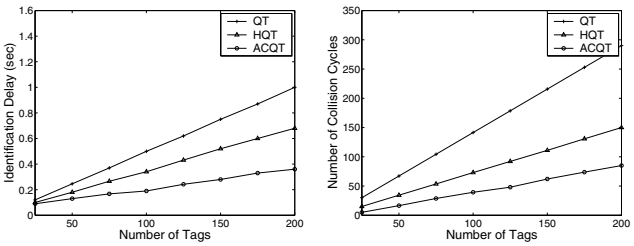
If the tags have the same prefix, each tag can choose to delay a given time slot based on their next bits. This idea is already introduced in the HQT [3], but HQT only records the values of a busy duration and a busy starting time for a channel. If a cycle does not respond in this process, it will not be observed, so the scheme cannot detect all idle cycles. We extend this scheme by recording the values every time the tag responses are received. Based on the number of time slots, we can discard the idle cycles and the corresponding subtrees, which will decrease the times of the inquire process. To distinguish between a 3-ary tree and a binary tree, our method uses different time slots. Using the scheme we presented, the unnecessary idle cycles can be reduced.

III. Simulation And Analysis

We evaluated the performance of the adaptive combination query tree protocol as compared to the QT protocol and the HQT protocol. The simulation consists of one reader and n tags to recognize, whose area is $10m \times 10m$. The reader is located in the center of the simulation area and its reading range is $3m$, n varies from 25 to 200. Each tag has a unique ID length of 128 bits. Both the query and tag's response are transmitted at the same transmission rate: $128Kbps$. Tag mobility follows the random walk model [6] with a maximum speed of $2m$/process. We assume that the back off time slot is $20\mu s$. We compare three performance metrics in 2 different scenarios: average identification delay, number of collision cycles, and number of idle cycles. The tag's ID is randomly selected.

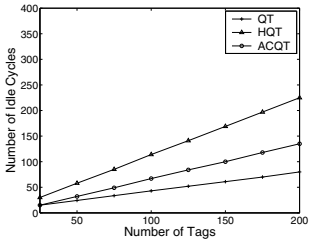
A. All Tags in the Scenario without Mobility

We assume that the number of tags is fixed, and there is no arriving or leaving tags within this query process. The reader should identify all tags starting with the initial query string and expanding the query string step by step. As shown in Fig.4, each result is the average from 20 simulations. The three performance metrics of ACQT are lower with increasing number of tags. In Fig.4 (a) and (b), the identification delay and the collision cycles are lowest for the ACQT, HQT is in the middle, and QT is highest. This is because the literature



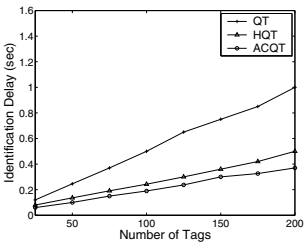
(a) Identification Delay

(b) Number of Collision Cycles

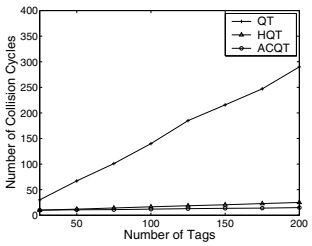


(c) Number of Idle Cycles

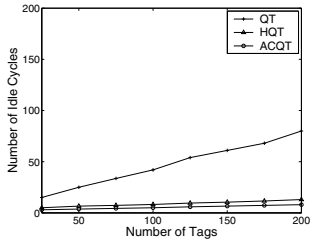
Fig. 4. The performances of three different protocols in the scenario without mobility.



(a) Identification Delay



(b) Number of Collision Cycles



(c) Number of Idle Cycles

Fig. 5. The performances of three different protocols in the scenario with 30% tags mobility.

[5] showed that the 3-ary tree is optimal, and in a q-ary tree, if q is greater than 3, the performance is decreased as q is increased. Another reason is our protocol is a combination query tree with binary tree and 3-ary tree which can improve performances further. From Fig.4 (c), we can see that the idle cycles of our protocol is lower than HQT which is based on a 4-ary query tree, due to using the combination model and back off time scheme, which can decrease the idle cycles by deciding the time slot of a tag's response. QT is the lowest because every bit can be decided to dividing tag, so the idle nodes are so small.

B. All Tags in the Scenario with 30% *Tags Mobility*

In this scenario, there are some moving tags, in which the number of tags moving in is equal to the number of tags moving out. Each result is the average of the simulations for 20 times. Fig.5 shows the results of each protocol with 30% tags mobility, in which the QT protocol was not affected since it does not use query string obtained by prior query processes. The performance of ACQT is better than other protocols, because it has joining and leaving strategies. The leaving strategy can reduce the unnecessary idle nodes. If there are some tags leaving, it will cause some nodes to be deleted. The joining strategy can reduce the unnecessary collision nodes. If there are some tags arriving, the prior idle nodes can be used to deal with the collision, and prior collision cycles are only used one time. The HQT which has a 4-ary query tree is in the middle. The ACQT has a combination query tree with binary tree and 3-ary tree, so the identification, collision and idle measurement results are better.

IV. CONCLUSION

We describe a novel tag anti-collision protocol ACQT, which is suitable for large scale tags RFID systems. From the conversion scheme, we can use a 3-ary tree instead of a binary tree. Using the method of combination query tree, it can solve the problem of the length of the tag's ID to build a 3-ary tree. The joining and leaving strategies are presented to adapt to the mobility of tags. It can reduce the idle and collision cycles. The simulation results show that our protocol can achieve better performance than previous protocols. They include shorter identification delays and fewer collision and idle cycles. Based on the comprehensive analysis and comparison, our protocol is an optimal one.

REFERENCES

[1] K. Finkenzeller, *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification*, 2nd edition. Wiley, 2003.

[2] C. Law, K. Lee, and K. Y. Siu, "Efficient memoryless protocol for tag identification," in *Proc. 2000 Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pp. 75–84.

[3] J. Ryu, H. Lee, Y. Seok, *et al.*, "A hybrid query tree protocol for tag collision arbitration in RFID systems," in *Proc. 2007 IEEE International Conference on Communications*, pp. 5981–5986.

[4] C.-N. Yang, Y.-C. Kun, J.-Y. He, and C.-C. Wu, "A practical implementation of ternary query tree for RFID tag anti-collision," in *Proc. 2010 IEEE International Conference on Information Theory and Information Security*, pp. 283–286.

[5] P. Mathys and P. Flajolet, "Q-ary collision resolution algorithms in random-access systems with free or blocked channel access," *IEEE Trans. Inf. Theory*, vol. 31, no. 2, pp. 217–243, Mar. 1985.

[6] R. A. Guerin, "Channel occupancy time distribution in a cellular radio system," *IEEE Trans. Veh. Technol.*, vol. VT-35, no. 3. pp. 89–99, Aug. 1987.