Research Journal of Applied Sciences, Engineering and Technology 5(8): 2494-2498, 2013
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: July 27, 2012 Accepted: September 12, 2012 Published: March 15, 2013

An Improved Query Tree Anti-Collision Algorithm Using Collision Location

Keli Chen and Bing Li
School of Mathematics and Computer, Xihua University, Chendu, 610039, P.R. China

Abstract: In order to reduce the identification delay and the energy consumption of the Query Tree Anti-collision Algorithm, the disadvantage of the current query tree anti-collision algorithms is analyzed and an improved query tree anti-collision algorithm is proposed, which is called collision Location based Hybrid Query Tree (LHQT) algorithm. In the algorithm, Manchester code is used and collision location is detected when a collision occurs. Then collision location is used to update quickly and accurately the query prefix in Query Tree (QT) algorithm and Hybrid Query Tree (HQT) Algorithm. Theoretical analysis and simulation show that the proposed algorithm can efficiently decrease the identification cycles and transmitted bits, which reduce the identification delay and the energy consumption.

Keywords: Anti-collision algorithm, collision location, query tree

INTRODUCTION

Radio Frequency Identification (RFID) technology is the most crucial to the Internet of Things, which is called as the "third information revolution" after the computer and the Internet. With a large number of applications of RFID technology in the Internet of things, there are a large number of identification tags in the range of a reader and the reader should be able to accurately identify all tags in a timely manner. However, when multiple tags transmit their IDs simultaneously, data collision resulted from the data transaction between more tags and the reader at the same time occurs. Anti-collision algorithms for passive RFID tags are critical to identify tags correctly and efficiently.

Numerous existing anti-collision algorithms can be divided into two categories: probability algorithm and deterministic algorithm (Klair *et al*., 2010). In probability algorithm such as Pure Aloha (PA), Slotted Aloha (SA) and Framed Slotted Aloha (FSA), tag generates a random time delay to respond to readers when collisions. However, there is a “tag starvation” problem in probability algorithm. It means that a specific tag cannot be identified for a long time, which cannot guarantee 100% identification ratio. In order to solve the problem, a series of deterministic algorithms such as Tree Splitting (TS), Query Tree (QT), Binary Search (BS) and Bitwise Arbitration (BTA), has been proposed. In deterministic algorithm, readers continue to send the query prefix of the tag ID and tags which prefix of tag ID is matched to the query prefix is response to reader, reader detects the collision and the reader splits tags into smaller subsets recursively through extending the query prefix until only one tag responds in each round of the tag-reader communication. Deterministic algorithm can provide 100% recognition success rate and has been widely used. However, due to the continuously splitting procedure, they have the relatively longer identification delay and higher power consumption. In TS variants, tags require random number generator and a counter to track their tree position, thus making them costly and computationally. QT algorithms overcome these problems by storing tree construction information at the reader and tags only need to have a prefix matching circuit.

LITERATURE REVIEW

Before presenting the proposed LQT algorithms, we first present the related previous work about query tree algorithm, because it is the bases of the LQT and necessary for understanding LQT.

Query Tree algorithm (QT): Query Tree (QT) (Law *et al*., 2000) is considered to be a milestone in the development of binary tree-based algorithms (Haosong and Younghwan, 2011). The reader sends out a query prefix, the tag in the range of the reader compares the prefix with its ID and transmits its ID to the reader if its ID matches with the prefix. Collision occurs when multiple tags have the same prefix. If a collision happens, the reader extends the previous prefix by adding bit 0 or 1 and starts the next query. In a recent query, previous responding tags are divided into two subsets. Reader can identify the tag immediately until there is only one tag matching with the prefix in a

Corresponding Author: Keli Chen, School of Mathematics and Computer, Xihua University, Chendu, 610039, P.R. China

subset. The query procedure will not stop until all tags are identified. Although the QT algorithm can guarantee reliable performance, it needs a long time to converge the identification process. There are some reasons resulted in the long time as follows:

- Only one bit is added to the previous prefix in extending the query prefix. This may result in more querying procedure.
- There are many Idle Cycles in extending the query prefix. There are numerous extensions to the QT such as shortcutting, Aggressive enhancement, AQT, HQT and Unified Q-ary Tree (UQT) (Prapassara and Bela, 2009) etc.

Hybrid Query Tree algorithm (HQT): Ryu *et al*. (2007) extends the QT protocol with aggressive enhancement and a slotted random back-off mechanism. In a new query, the query prefix is appended with two bits (even more bits), instead of a single bit. The extension reduces the collision cycles but increases the idle cycles. In order to reduce collision cycles and eliminate idle cycles, HQT combined the QT with a slotted random back-off mechanism. When a new query prefix is received, the tags matching the prefix do not respond immediately but respond after a back-off delay. The duration of the back-off timer is determined by the last two bits of the prefix. The extension may reduce the idle cycles resulted from aggressive enhancement. There are some extensions to the HQT (Haosong and Younghwan, 2011; Zhou and Cai, 2012; Sun and Chen, 2011; Jiang and Ma, 2012).

Collision location: By using Manchester Encode (Finkenzeller, 2003), if two (or more) transponders simultaneously transmit bits of different values, then the positive and negative transitions of the received bits cancel out each other and lead to an error. When the collision occurred in the RFID system, the error can be used to locate the position of the collision bit. By using the collision position information, the reader may extend the query prefix by adding more bits to the previous prefix and reduce idle cycles in the identification process (Haosong and Younghwan, 2011; Jiang and Ma, 2012; Choi *et al*., 2006).

THE PROPOSED ALGORITHM

In order to reduce collision cycles and idle cycles and to minimize total identification delay, we propose an enhanced algorithm with integration of the present algorithms in section II. The proposed algorithm is named as collision Location based Hybrid Query Tree (LHQT) algorithm.

The idea description of LHQT: We assume that:

- The Manchester Encode is used in the algorithm for locating the collision positions.
- The reader uses a prefix stack to maintain the query prefix.
- The memory-less tag has a circuit matching the query prefix and all the tags in the range of reader can respond simultaneously.

The algorithm is combined with QT and a slotted random back-off mechanism. The following optimizations are made in LHQT:

- The reader stops to receiving the bits transmitted by tags when it locates two collision bits.
- The reader updates the query prefix by adding the received bit to it when no collision occurs.
- After that, the responding tags are split into four subsets by updating the two collision bits.
- With the difference of HQT, the duration of the back-off timer is determined not by the last two bits of the query prefix but by the last two bits of the tag ID. The reason is that the last two bits of the query prefix may be the same for the query prefix may mean the category of the identification objects in EPC. Duration of the back-off timer is similar and a slotted random back-off mechanism has no effect to reduce the idle cycles. However the last two bits of tag ID are changed randomly for these bits means the numbers of objects.
- The duration of the back-off timer is determined not by the value of last two bits of the tag ID but by the count of bit 1 in the last two bits of the tag ID. In the case, there only three slots: slot 0 means that the last two bits are 00, slot 1 means that the last two bits are 01 or 10, slot 2 means that last two bits are 11. It reduces the total slots and reduces the idle slots.

Procedure description of LHQT: The LHQT consists of the following steps:

- The reader initializes a null query prefix string and a null prefix stack and pushes the query prefix into prefix stack
- The reader fetches a query prefix from prefix stack and broadcasts it to all tags and each tag sends back a response to the reader. However, a tag gives response after the back-off timer which is decided by the count of last two bits and a collision may occur
- The reader locates the collision positions and generates the new query prefixes which are pushed into prefix stack. There are some cases in generating the new query prefix as following:
  - **Idle cycle:** No tag responds; idle cycle should be as little as possible, until zero
  - **Identification cycle:** Only one tag responds the reader in which the reader may identify the tag; the number of identification cycles should be equal to

Table 1: LHQT algorithm example

Step	Query prefix	Respond tags	Updated prefix	Query stack	Identified tags
1	Null	A: 001110	*0*	100	
		B: 000011		001	
		C: 100110		000	
		D: 100011			
2	100	C: 100110	100*1*	001	C
		D: 100011		000	D
3	001	A: 001110	001110	000	A
4	000	B: 000011	000011	Null	B

or less than the number of tags. Less identification cycles are, the better the performance of the algorithm is

- **Collision cycle:** Collision occurs. In the case, the reader tracks collisions and then updates the query prefix. If received bit is not the collision bit, the reader updates the query prefix with the received bit. If received bit is the collision bit, the reader updates the query prefix with the char '*'. Then, the reader counts the number and position of collision bits in query prefix until the number of collision bits is two or the length of the updated query prefix is equal to the length of tag ID. It also has several cases:
- **Only one collision:** It means that there are two tags which collision bits of ID are 0 or 1. So the reader updates the QP by replacing the char '*' with bit 0 or 1 and identifies the two tags
- **More than one collision:** The reader updates the collision bit of query prefix with 0 or 1 and pushes them into the prefix stack. As HQT algorithm, the reader updates two collision bits of query prefix not only one. For example, the updated query prefix is "1*00*1", then generates the four query prefixes: 100001, 100011, 110001 and 110011

- **Query stacks checking:** If the prefix stack is not empty, continue to step b; if the prefix stack is empty, the identification process is terminated

Example: In order to understand our proposed LHQT algorithm, we walk through the identification process in Table 1 with the assumption that there are four tags.

PERFORMANCE EVALUATION

In this study, all the IDs are randomly generated. Like all the previous works, the time delay to identify all the tags is decided by the number of queries sent by the reader and the communication overhead. The communication overhead is measured by the number of bits transmitted by tags and the number of bits transmitted by the reader. Performance of LHQT is compared with QT and HQT and LQT which is updated only collision bit in every round. Performance was evaluated with different number of tags. The number of tags is set up from 20 to 250 in step of 20. Each data point shown in the figures is the average of 100 runs.

It is supposed firstly that the length of tag ID is eight statically to analyze the numbers of tags on the performance of the algorithm.

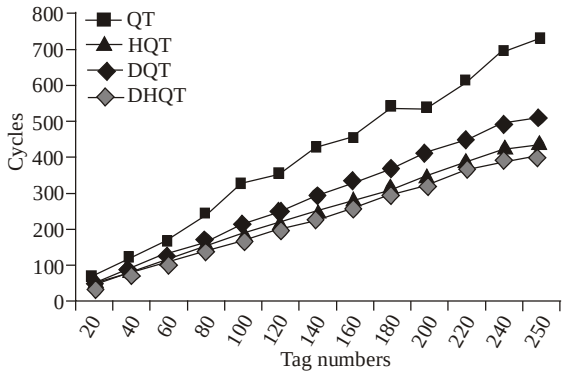

Fig. 1: Comparison of query cycles

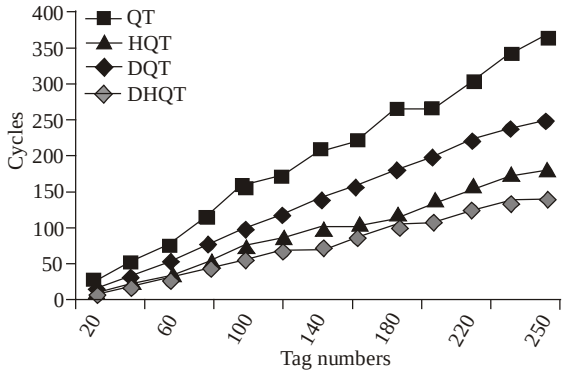

Fig. 2: Comparison of collision cycles

Query cycles: Query cycles are divided into three types such as idle cycles, collision cycles and identified cycles. Figure 1 and 2 compares query cycles and collision cycles for identification against different numbers of tags. In these figures, it shows that the LHQT protocol outperforms the others in respect of the number of query cycles. The reason is: in LHQT and LQT, the collision location is introduced to update the non-collision bit of query prefix and avoid idle queries. With the difference of LQT, extending two collision bits in a query is adopted in LHQT as HQT in order to minimize the idle cycles. So the number of query cycles and idle cycles is less than LQT. In HQT, it employs slotted back-off mechanism to reduce the prefixes which resulting in idle cycles. In QT, the reader detects only the collision occurs but does not make full use of collision position information and extends only one bit to the prefix, so there are more idle cycles and query cycles. It is worst one.

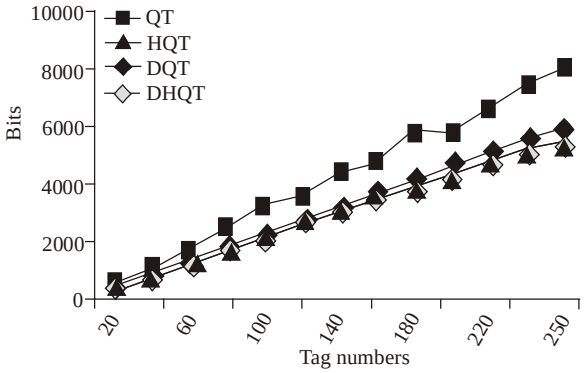

Fig. 3: Comparison of transmitted bits

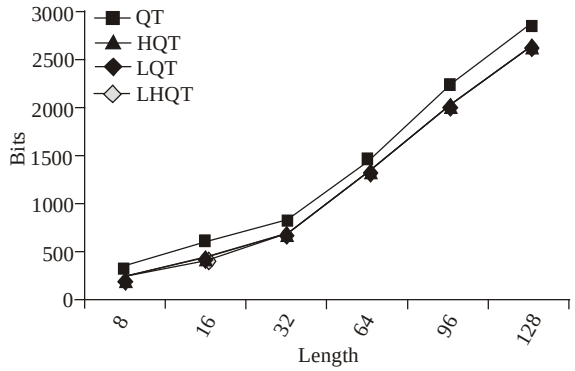

Fig. 4: Comparison of ID length

Transmitted bits: Figure 3 compares the transmitted bits for identification against different numbers of tags. It can be seen from the figure that transmitted bits in LHQT is the least. The reasons are:

- Transmitted bits for identification are composed of bits sent by the reader and bits transmitted by tags. Although the bits sent by the reader in a query cycle are same in four algorithms, bits sent by the reader for identification in LHQT are least comparing with others as it has the least query cycles.
- In LHQT and LQT, the reader stops receiving a bit of responding tag's ID once it has tracked one or two collision bit. But in QT and HQT, the reader receives a few of bits from k+1 to N (k is length of prefix and N is length of ID). The greater the value of N is, the better the performance of LHQT algorithm.

Length of tag ID: In order to evaluate the length of ID on the performance of algorithms, the length of ID is changed from 8 to 128 bits. And the number of tags is 200 statically. Figure 4 compares transmitted bits for identification 200 tags against different length of ID. It can be seen from the figure that transmitted bits in LHQT is almost the same as HQT and LQT, but is less than QT. With the increasing of the length, the transmitted bits are more. It is because the query prefix sent by reader and bits transmitted by a tag in a query is more.

CONCLUSION

Tag anti-collision is a crucial technique for RFID system. The approach of using collision location in HQT algorithms to shorten the identification process is presented to provide for energy-aware RFID tag identification. Simulation results show that LHQT can achieve a significant reduction in processing time for tag identification. The algorithm, like the existing Hybrid Query Tree algorithm, is memory-less requiring the tags to store no state of the identification process and offer guarantees on the time required to read all tags. LHQT does not only employ the individual collision location but also adopt the relevant information to update quickly the query prefix. Then the 4-ary tree is used to extend the two collision bits in updated query prefix. It may reduce the collision cycles but increase the idle cycles. LHQT combined the QT with a slotted random back-off mechanism. The duration of the back-off timer is determined by the last two bits of the tag ID which changes faster and can really play a random effect.

ACKNOWLEDGMENT

This study was supported by the Research Fund of Key Laboratory of Xihua University (No. XDZ0818-09), China. We also thank the anonymous reviewers for giving valuable suggestions to further improve on this work.

REFERENCES

Choi, J.H., D. Lee, Y. Youn, H. Jeon and H. Lee, 2006. Scanning based pre-processing for enhanced RFID tag anti collision protocols. International Symposium on Communications and Information Technologies (ISCIT), Bangkok, Thailand, pp: 1207-1211.

Finkenzeller, K., 2003. RFID Handbook. Carl Hanser Verlag, Munich, FRG, pp: 200-219.

Haosong, G. and Y. Younghwan, 2011. A bit collision detection based hybrid query tree protocol for anti-collision in RFID system. 11th IEEE International Conference on Computer and Information Technology, South Korea, pp: 158-163.

Jiang, Y. and M. Ma, 2012. RFID bit match anti-collision algorithm in Internet of things. Appl. Res. Comput., 29(1): 88-91.

Klair, D.K., C. Kwan-Wu and R. Raad, 2010. A survey and tutorial of RFID anti-collision protocols. IEEE Commun. Surveys Tutorial, 12(3): 400-421.

Law, C., K. Lee and K.Y. Siu, 2000. Efficient memoryless protocol for tag identification (extended abstract). Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Computing and Communications, Toronto, CA, pp: 75-84.

Prapassara, P. and S. Bela, 2009. Unified Q-ary tree for RFID tag anti-collision resolution. 20th Australasian Database Conference (ADC 2009), Wellington, New Zealand, pp: 49-58.

Ryu, J., H. Lee, Y. Seok, T. Kwon and Y. Choi, 2007. A hybrid query tree protocol for tag collision arbitration in RFID systems. IEEE International Conference on Communications (ICC), Scotland, pp: 5981-5986.

Sun, W. and A. Chen, 2011. An effective tag anticollision algorithm in RFID System. Appl. Res. Comput., 28(1): 3717-3719.

Zhou, Q. and M. Cai, 2012. Improved hybrid query tree anti-collision algorithm in RFID system. Comput. Eng. Design, 33(1): 209-213.