A Novel Anti-Collision Algorithm for High-Density RFID Tags

Sarawut Makwimanloy[1], Piya Kovintavewat[2], Urachada Ketprom[3], Charturong Tantibundhit[4]

[1,4] *Electrical and Computer Engineering Department, Thammasat University, Thailand*
[2] *RFID Technology and Applications Research Unit, Nakhon Pathom Rajabhat University, Nakhon Pathom, Thailand*
[3] *RFID Program, National Electronics and Computer Technology Center (NECTEC), Thailand*

Email: [1] makwimanloy@hotmail.com, [2]piya@npru.ac.th, [3]urachada.ketprom@nectec.or.th, [4]tchartur@engr.tu.ac.th

***Abstract*— In a radio frequency identification (RFID) system, when more than one tag communicates with the reader at the same time, a collision will occur, resulting in the failure of that communication. Many anti-collision algorithms, such as Binary Tree (BT), FSA, and DFSA, have been used in ISO and EPC standards to prevent such a collision. This paper develops a new anti-collision algorithm based on the BT and the DFSA algorithms. Specifically, all tags are divided into many groups using the DSFA algorithm. Then, the tags in each group are identified using the BT algorithm. Results indicate that the proposed algorithm performs better than the existing ones in terms of the number of used time slots (the less the used time slot, the faster the algorithm).**

I. Introduction

Radio frequency identification (RFID) is a technology for automated identification. Typically, an RFID system consists of a reader and tags, which communicate with one another via radio frequency waves. Recently, RFID has been widely used in many applications, such as transport systems, electronic ticketing, access control, animal identification, logistics, and supply chain management [1].

In the application, where many tags are present in the reader's field, if more than one tag communicates with the reader at the same time, a *collision* will occur resulting in the failure of that communication. Thus, each tag has to resend all information to the reader. To prevent this problem, an anti-collision algorithm must be used. Based on the International Standards Organization (ISO) and EPCglobal (EPC), there are 3 types of anti-collision algorithms, namely, binary tree (BT) [2, 3], Framed Slotted ALOHA (FSA) [2], and Dynamic Framed Slotted ALOHA (DFSA) [2, 4] algorithms. However, these algorithms take a lot of time identify tags [2].

Many improved anti-collision algorithms have recently been proposed in the literature. For example, Cheng and Jin [2] presented the analysis and simulation of several RFID anti-collision algorithms and partitioning of tags for near-optimum RFID anti-collision performance. Shin and Kim [5] proposed a partitioning technique, which enables a faster accurate estimation on the number of contending tags, and yields much higher throughput against previous non-partitioning approaches. Cho *et al.* [6] proposed an anti-collision algorithm using parity bit (ACPB) in RFID system. The ACPB identifies tags without checking all bits in the tags. Then, the reader uses the parity bit, which is added to the tag's ID number. Clearly, ACPB can reduce the number of the requests from the reader. Thus, it can shorten the time of identifying all tags in the reader's field. In this paper, we propose a novel anti-collision algorithm, which is based on the BT algorithm. The proposed algorithm can estimate the number of tags in the reader's field and identifies all tag faster than the existing anti-collision algorithms.

The rest of this paper is organized as follows. Section II briefly describes how BT and DFSA algorithms work. A new anti-collision algorithm is explained in Section III. Section IV compares the performance of different anti-collision algorithms. Finally, Section V concludes this paper.

II. Existing Anti-Collision Algorithms

This section briefly describes how BT and DFSA perform because their performances are compared with the proposed anti-collision algorithm.

A. *Binary Tree* (BT)

The BT algorithm or the Query Tree algorithm [6] divides tags into two groups based on the most significant bit (MSB) of the tag's ID number, which consists only of bits "0" and "1". To search a tag, a dividing process continues adding up the number "0" and "1" into each group, until finding a tag [2, 7, 8]. Note that we consider only the case where the tags do not support a random generator in hardware for group selection [9], meaning that the BT algorithm operates on the tag's identification (ID) numbers.

To obtain all tags, the reader begins a search by sending a prefix bit "0" or "1" to all tags and waits for the response. If there is only one response, the reader then can identify that tag. However, if more than one tag responds back at the same time, a collision will occur. In this case, the reader will add another bit ("0" or "1") to a prefix bit and send the new prefix bits to the remaining tags until there is only one response. The reader will do this process until all tags are identified.

To compare the performance of different anti-collision algorithms, we use the required total number of commands sent from the reader to the tag as a criterion. Each command is referred to as one *time slot* (or, in short, *slot*). Assuming that each slot uses the same processing time, the algorithm that requires a large number of slots will operate slow.

B. Dynamic Framed Slotted ALOHA (DFSA)

Dynamic Framed Slotted ALOHA developed from FSA is utilized in Class 1 Generation 2 of EPC [4]. It divides tags into many groups according to the number of slots specified by a reader. All tags will random the slot number between 0 to the number of slots, and the tags having the same number will be in the same group.

First, the reader sends a command with a "slot_number." Note that the "slot_number" will be set to 0 at the first time, and it will then increase by 1 for every round. If the tag has a group number equal to the "slot_number," that tag will respond to the reader. Then, if there is only one response at this time, the reader will identify that tag. If there is a collision, the reader will increase the "slot_number" by 1 and send it to all remaining tags. The reader repeats this process until the "slot_number" is equal to the number of slots.

When the reader finishes sending a command with the "slot_number" between 0 to the number of slots, we assume that the operation time is one frame. If the reader cannot identify all tags in the reader's filed, the reader will begin the new frame. The reader can adjust the number of slots in the new frame based on a Q-parameter [4 – 5]. The reader will do this process until it can identify all tags in the reader's filed.

III. Proposed Anti-Collision Algorithm

The simulation in [10] showed that the BT algorithm is more efficient than FSA and DFSA. This is because the BT algorithm uses a less number of slots when the number of tags in the system is small. Practically, when the system has a large number of tags, the BT algorithm tends to perform worse because it uses a lot of slots to identify all tags if compared to DFSA [10].

The proposed algorithm is developed based on the BT and the DFSA algorithms. We first divide tags into many groups using the DFSA algorithm as illustrated in Fig. 1. Then, all tags in each group are identified using the BT algorithm. To achieve this, we assume that the tag can generate a 9-bit uniform random number and has a function to select a group according to that random number. To make the proposed algorithm more efficient, the number of groups must coincide with the number of tags. Specifically, the less the number of tags, the less the number of groups. Therefore, we must first estimate the number of tags in the reader's field so as to determine the number of groups used in the proposed algorithm. To do this, we use the number of tags in each group to estimate the total number of tags in the reader's field since each group should have an equal probability to have the same number of tags.

Figure 2 shows how the proposed anti-collision algorithm works. First, we determine the number of groups from the estimated total number of tags in the reader's field. Based on the simulation with maximum of 1,000 tags, the number of groups suitable for the proposed algorithm is 32 groups. Next, we randomly pick three groups in order to identify tags based

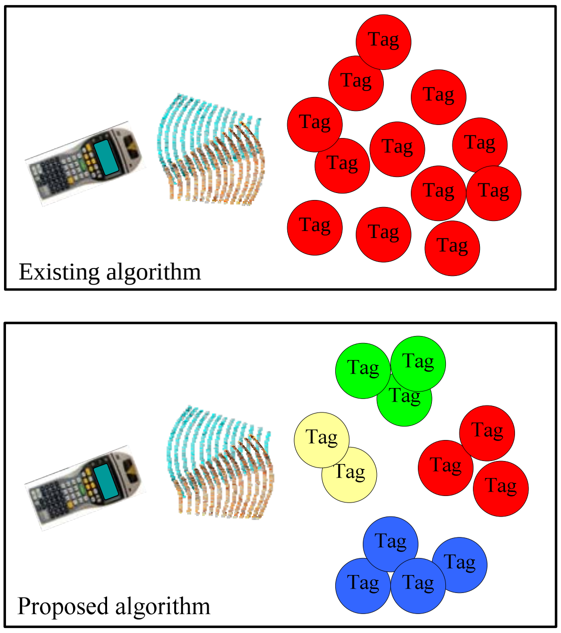

Figure 1. How the proposed algorithm works.

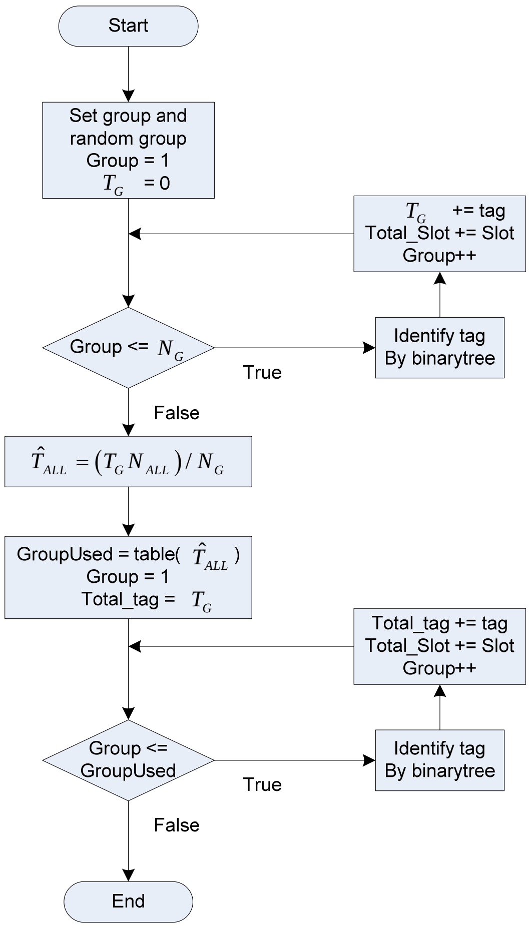

Figure 2. A flowchart of the proposed anti-collision algorithm.

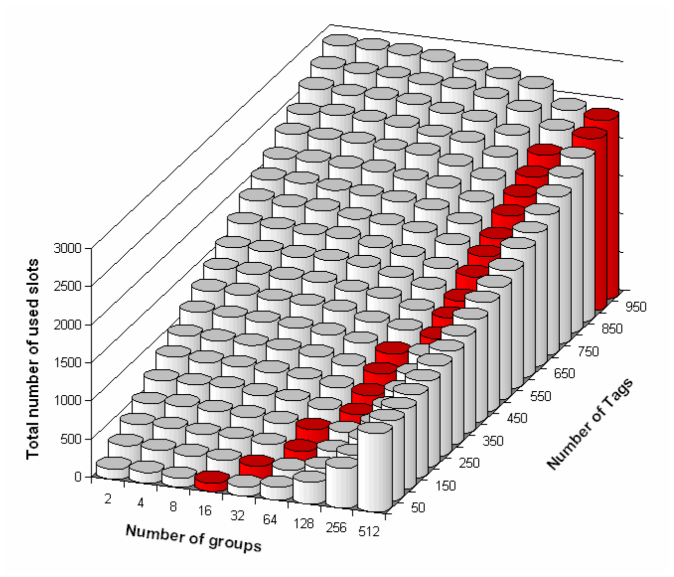

Figure 3. The number of used slots for different number of tags and groups.

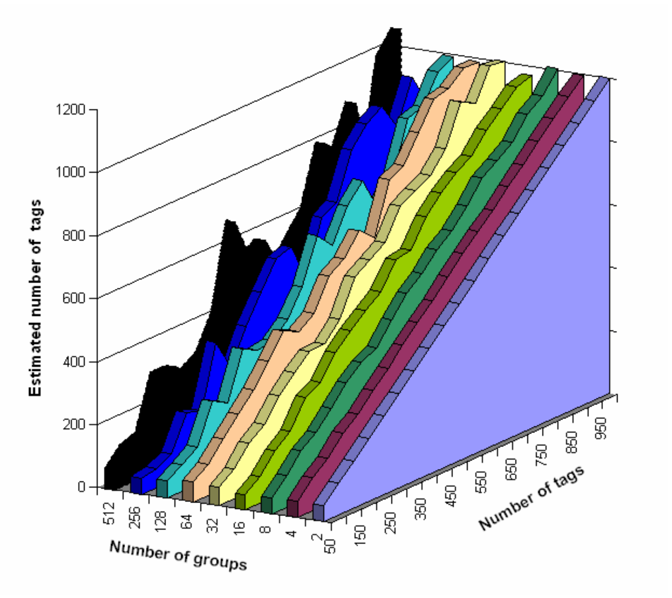

Figure 4. The estimated number of tags for different number of tags and groups (for N_G = 3).

on the BT algorithm. Then, the total number of tags in the reader's fields can be estimated according to

$$\hat{T}_{ALL} = \left(T_G N_{ALL}\right) / N_G \qquad (1)$$

where $\hat{T}_{ALL}$ is the estimated total number of tags in the reader's field, T_G is the number of identified tags in the selected three groups, N_G is the number of selected groups used to find $\hat{T}_{ALL}$ (e.g., N_G = 3), and N_{ALL} is the total number of groups in the reader's field (e.g., N_{ALL} = 32).

Once we have an estimate of the total number of tags in the reader's field, we can now choose the suitable number of groups to identify tags. Then, we use a regular BT algorithm to identify tags in each group.

IV. Simulation Result

Assuming that the tag's ID number consists of 64 bits (all random bits). Our proposed method to estimate the total number of tags in the reader's field is efficient when the number of tags is varying. Note that we use the BT algorithm to identify tags in each group. Figure 3 shows the total number of used slots to identify all tags for different number of tags and groups, where the x-axis represents the number of groups, the y-axis indicates the number of tags, and the z-axis represents the number of used slots. Practically, the less the number of used slots, the faster the algorithm. It is apparent that for a given number of tags, there is the suitable number of groups (i.e., the shaded columns) that yields the lowest number of used slots. Therefore, the proposed algorithm must first estimate the total number of tags in the reader's field so as to determine the suitable number of groups.

Figure 4 illustrates the estimated number of tags for different number of tags and groups, where the x-axis represents the number of groups, the y-axis indicates the number of tags, and the z-axis represents the estimated number of tags. Clearly, the less number of groups will result in a better estimation of the total number of tags. For example, the number of groups of 2 will give 100% accuracy of the estimated total number of tags. However, based on exhaustive search, we found that the number of groups of 32 is the maximum number of groups, which yields minimum error of the estimation under specified condition. For example, for N_G = 3 and N_{ALL} = 32, the total number of tags from 0 to 200 tags will give an error of 31% - 37%, but for N_G = 31 and N_{ALL} = 32, the total number of tags from 0 to 200 tags will give an error of 0.05% - 0.15%. Thus, the chosen parameter for N_G will depend strongly on the error threshold requirement.

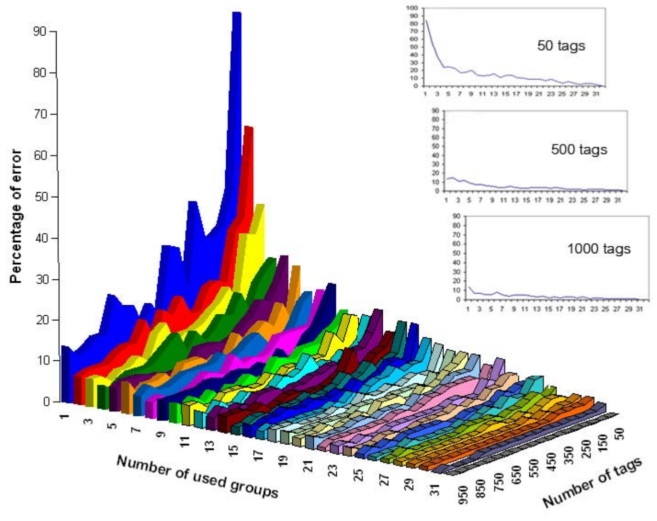

Figure 5. The percentage of error between the actual number of tags and the estimated number of tags (for N_{ALL} = 32).

Figure 5 compares the percentage of error between the actual number of tags and the estimated number of tags obtained from our proposed method, where the x-axis represents the number of used groups for estimating tags, the number of used groups for estimating tags, the y-axis indicates the number of actual tags, and the z-axis represents the percentage of error. We first set the total number of groups of 32 (i.e., $N_{ALL} = 32$). Then, we vary the number of used groups from 1 to 32 (i.e., $N_G = 1$ to 32) so as to estimate the total number of tags in the reader's field. If we use a large number of used groups, the estimation error will be small, but the proposed algorithm will require a lot of number of used slots, which implies low efficiency. Conversely, if we use a small number of used groups, the estimation error will be large, resulting in unacceptable estimate. Based on Fig. 5, we set the number of used groups to be 3 because if the larger number of group is utilized, the number of used slots will increase to an unacceptable level even though the percentage of error between the estimated tags and the actual tags is decreased.

In this paper, we compare the performance of the four algorithms, namely, Binary Tree, Binary Tree 3 bits, DFSA, and the proposed algorithm (with 32 groups), assuming that the tag's ID number consists of 64 bits (all random bits).

Figure 6 illustrates the performance comparison as the plot between the number of tags (x-axis) and the total number of used slots (y-axis). The smaller the number of used slots, the faster the algorithm. The proposed algorithm outperforms the other algorithms, i.e., at the considering total number of used slots, the proposed algorithm uses a smaller number of tags. The advantage of the proposed algorithm is more visible as the increase of the number of tags and could be explained as follow. The DFSA divides groups of tags into slots randomly. Thus, tags are more likely to collide especially when a large number of tags are presented in the reader's field. While in the case of BT and BT 3-bit, the more numbers of tags presented in the reader's field, the more identical of the most significant bit ID of the tags. Therefore, more collisions occur resulting in higher used slots.

V. Conclusions

The anti-collision algorithms are crucial to the application that uses a large number of tags. In general, the BT algorithm performs faster than the DFSA algorithm when the number of tags is small. The proposed algorithm exploits the advantage of both the BT and the DFSA algorithms. Specifically, all tags are divided into many groups based on the DFSA algorithm, and the tags in each group are identified using the BT algorithm. It is clear from simulation that the proposed anti-collision algorithm performs better than the existing ones in terms of the number of used time slots, which implies fast identification process.

Acknowledgment

This work is supported by National Science and Technology Development Agency (NSTDA) and the RFID Program, National Electronics and Computer Technology Center (NECTEC), Thailand, under grant TG-44-21-50-098M.

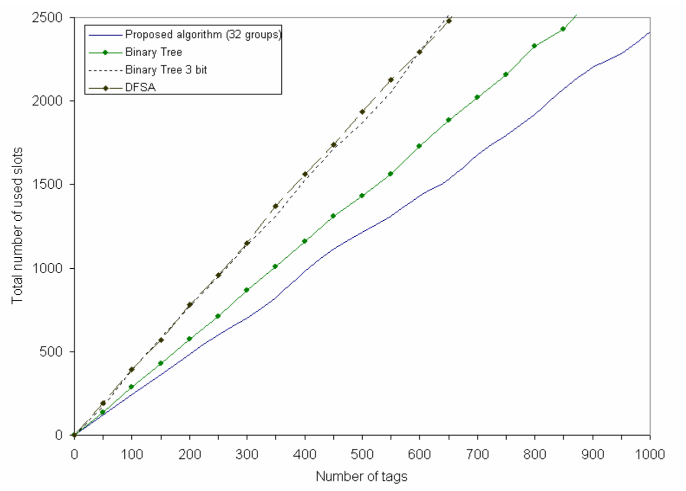

Figure 6. Performance comparison of different anti-collision algorithms.

References

[1] K. Finkenzeller, *RFID handbook*, John Wiley & Sons, West Sussex, 2003.

[2] T. Cheng and L. Jin, "Analysis and Simulation of RFID Anti-collision Algorithm," *IEEE Advanced Communication Technology*, vol. 1, pp. 697 – 701, Mar. 2007.

[3] EPC Global. 860MHz~930MHz Class I Radio Frequency Identification Tag Radio Frequency & Logical Communication Interface Specification Candidate Recommendation, Version 1.0.1.

[4] EPC Global. EPC^{TM} Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz~960MHz, Version 1.0.9.

[5] W. J. Shin and J. G. Kim, "Partitioning of Tags for Near-Optimum RFID Anti-collision Performance," *IEEE Wireless communications and Networking Conference*, pp. 1673-1678, Mar. 2007.

[6] J. S. Cho, J. D. Shin and S. K. Kim, "RFID Tag Anti-Collision Protocol: Query Tree with Reversed IDs," *ICACT*, pp. 225-230, Mar. 2008.

[7] C. Abraham, V. Ahuja, A. K. Ghosh, and P. Pakanati, "Inventory Management using Passive RFID Tags: A Survey," Department of Computer Science thesis, University of Texas at Dallas, Richardson, Texas.

[8] R. Ahmed, "Performance Comparison of RFID Tag Anti-collision Algorithm using Simulation and Real Testing Based," M. Eng. thesis, Asian Institute of Technology, Thailand, May.2007.

[9] ISO/IEC 18000-6:2003(E), Part 6: Parameters for air inter-face communications at 860-960 MHz, Nov. 26, 2003.

[10] S. Makwimanloy, P. Kovintavewat, U. Ketprom, C. Tantibundhit and C. Mitrpant, "A New Anti-Collision Based on A-Priori Information," in *Proc. of ECTI-CON* 2008, Krabi, Thailand, vol. II, pp. 733 – 736, May 14 – 16, 2008.