A New Anti-Collision Based on A-Priori Information

Sarawut Makwimanloy[1], Piya Kovintavewat[2], Urachada Ketprom[3], Charturong Tantibundhit[4], Chaichana Mitrpant[5]

[1,4] *Electrical and Computer Engineering Department, Thammasat University, Thailand*
[2] *RFID Technology and Applications Research Unit, Nakhon Pathom Rajabhat University, Nakhon Pathom, Thailand*
[3,5] *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, [5]chaichana.mitrpant@nectec.or.th

***Abstract*— A collision occurs when more than two tags present in the reader's field of a radio frequency identification (RFID) system. Many anti-collision algorithms (e.g., Binary Tree, FSA, and DFSA) have been employed in ISO and EPC standards to prevent such a collision. This paper proposes a new anti-collision algorithm based on a-priori information about the manufacturer code. Results indicate that the proposed anti-collision 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). Specifically, the proposed algorithm uses less number of slots than the existing ones, approximately 50%.**

I. Introduction

Radio-frequency identification (RFID) system has been introduced to uniquely identify the object of interest. The RFID system consists of a reader and a tag, which communicate between each other via radio frequency waves. Recently, the RFID system has been employed in a variety of applications, such as, transport systems, ticketing, access control, animal identification, and so forth.

When more than one tag in the reader's field communicates with the reader at the same time, a *collision* will occur resulting in the failure of that communication. In this case, 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) [1, 2], Framed Slotted ALOHA (FSA) [1], and Dynamic Framed Slotted ALOHA (DFSA) [1, 3] algorithms.

Many improved anti-collision algorithms have recently been proposed in the literature. For example, reference [1] presents the analysis and simulation of several RFID anti-collision algorithms and partitioning of tags for near-optimum RFID anti-collision performance. Partitioning technique enabling a faster accurate estimation on the number of contending tags, which yields much higher throughput against previous non-partitioning approaches, was proposed in [4]. However, for a special case where a-priori information about the manufacturer code is known, there is no anti-collision algorithm that exploits such information so as to improve its performance. In this paper, we propose a new anti-collision algorithm based on a-priori information, which shows better performance in terms of the number of used time slots (or speed) than other anti-collision algorithms. Specifically, the less the number of used time slots, the faster the algorithm. The performance comparison of different anti-collision algorithms used in ISO and EPC standards is also provided to serve as a guideline for users to decide which algorithm should be utilized for a given condition.

The rest of this paper is organized as follows: Section II briefly describes the anti-collision algorithms used in ISO and EPC standards. A new anti-collision algorithm based on a-priori information 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 the anti-collision algorithms (i.e., BT, FSA, and DFSA) work.

A. Binary Tree Algorithm

A Binary Tree (BT) algorithm is employed in Type B of ISO 18000-6 and Class 1 of EPC [2]. It basically divides tags into two groups based on the most significant bit (MSB) of the tag's ID number, denoted as MSBID, 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 [1, 5, 6]. Note that we consider only the case where the tags do not support a random generator in hardware for group selection [7], meaning that the BT algorithm operates on the tag's identification (ID) numbers.

Fig. 1 shows how the BT algorithm works. Suppose there are 3 tags in the reader's field, namely, "011," "101," and "110," where the first digit is MSB. To obtain all tags, the reader begins a search by sending bit "0" (step 1) to all tags and waits for the response. There is one response sent to the reader because there is only one tag beginning with bit "0." Now, the reader recognizes Tag 1. Next, the reader sends bit "1" (step 2) to the other two tags, i.e., "101" and "110". In this case, a collision occurs because two tags respond back at the same time. Therefore, the reader sends another bit "0" (step 3) to these two tags. In this time, the reader can recognize Tag 2 because the reader receives only one response. Then, the reader sends another bit "1" (step 4) to the remaining tag, which results in only one response from

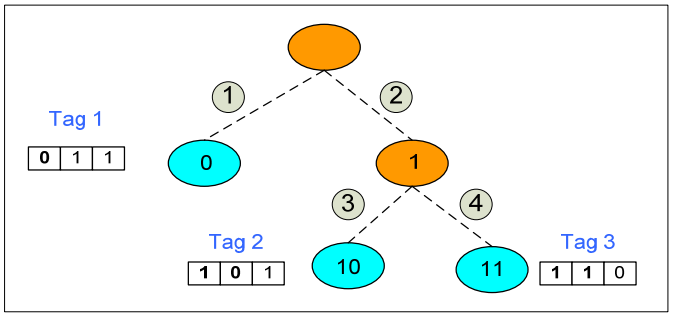

Fig. 1. How Binary Tree algorithm works.

Tag 3 sent to the reader. This means there is no other tags in the reader's field, which implies the end process of the BT algorithm.

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. For example, in Fig. 1, the total number of slots that the reader requires to recognize all three tags is four slots. This means that the number of slots is increased one slot every time when the reader sends out each one bit, i.e., "0" or "1."

For the BT algorithm used in EPC Class 1, the searching procedure is similar to the BT algorithm used in ISO 18000-6 Type B, but the BT algorithm in EPC Class 1 will divide a group into 8 subgroups based on 3 bits at each step [2]. There are both advantages and disadvantages between the BT algorithm used in ISO and EPC as illustrated in Section IV.

B. Framed Slotted ALOHA (FSA)

This algorithm developed from the Slotted Aloha algorithm is used in Type A of ISO 18000-6 [7]. It divides tags into many groups according to the number of slots specified by a reader. All tags will random the slot number, and the tags having the same number will be in the same group.

First, the reader sends an "Init_round" command to tags for setting the number of slots within one frame. Next, tags randomly pick a slot number between 0 to "slot_number," and record it into a "slot_count." If the "slot_count" equals to the required "slot_number," the tag will respond to the reader. Then, three possible outcomes could happen:

1) No Tag respond
   Reader will send a "Close_slot" command to all tags to increase "slot_count."
2) One Tag respond
   Reader will pass a "Next_slot" command to the responded tag so as not to respond the reader in next frames.
3) Multiple Tags respond
   Reader recognizes a collision and will send "Close_slot" to collided tags so as to increase "slot_count."

This procedure repeats until the reader can identify all tags completely [6]. In FSA, the total number of slots is equal to all slots used in the FSA algorithm.

C. Dynamic Framed Slotted ALOHA (DFSA)

This algorithm developed from FSA is utilized in Class 1 Generation 2 of EPC. It works similar to FSA except that the number of slots in each frame can be adjusted based on a Q-parameter [3, 4].

In DFSA, a reader sends a command to tags for specifying a Q-parameter. Next, tags randomly select and record values between 0 and Q-parameter into "slot_counter." The tag with "slot_counter" equal to 0 will respond back to the reader. Then, the reader sends a "Query" command to decrease the value of "slot_counter", and also sends a "QueryAdjust" command to adjust the value of Q-parameter. However, if there are empty or collided slots more than the number of accepted slots, tags will repeat all steps until the reader can identify all tags.

III. Proposed Anti-Collision Algorithm

When we know some a-priori information about the tags, we will be able to improve the performance of an existing anti-collision algorithm. In this paper, three a-priori information are considered in this paper, i.e.,

1) Suppose a-priori information about the total number of tag's manufacturers is known. We found that for each application, if possible, it is preferable to employ all tags from one manufacturer in the system.
2) Suppose the total number of tags needed to identify is known. In this case, we found that there is no significant performance improvement when we use this information in an anti-collision algorithm.
3) Suppose the manufacturer code of tags is known. In this case, we can use this information to improve the performance of an anti-collision algorithm. Specifically, the manufacturer code helps reduce the time to identify all tags.

Figure 2 shows a structure of tag's ID number. In a searching process, all anti-collision algorithms begins with the MSBID (i.e., the first bit in the right-hand side of Fig. 2), and continues to the 64-th bit. The proposed anti-collision algorithm is the existing anti-collision algorithm that exploits a-priori information. This means that if we know a manufacturer code (i.e., IC manufacturer serial number in Fig.2), the proposed anti-collision algorithm can start the searching process at the 33-th bit, instead of the first bit. Clearly, this will reduce the time to identify all tags. As shown in simulation, the proposed anti-collision algorithm identifies all tags much faster than other algorithms.

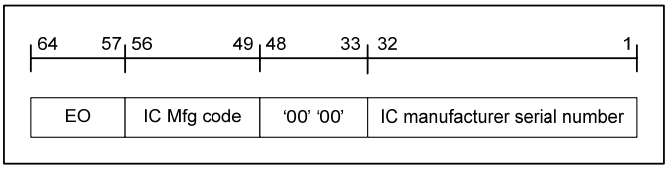

Fig. 2. A structure of tag's ID number used in ISO 18000-6 [7].

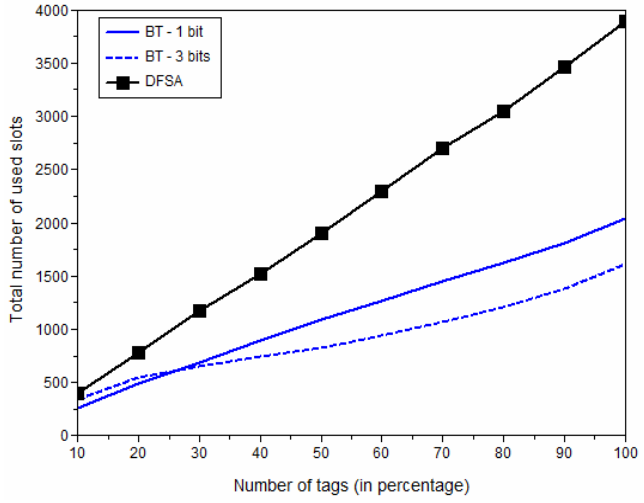

Fig. 3. Performance comparison of BT 1-bit, BT 3-bit, and DFSA.

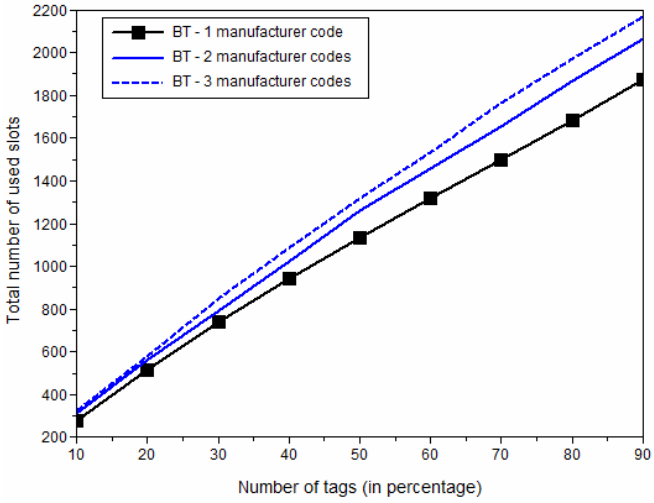

Fig. 4. Performance of BT with multiple manufacturer codes.

IV. Simulation

Performance comparison of anti-collision algorithms has been investigated in [1, 6]. In this paper, we compare the performance of the proposed anti-collision algorithm with the existing algorithms in different aspects as follows.

A. DFSA and Binary Tree

Here, we compare the performance of three algorithms, i.e., Binary Tree 1 bit (BT 1-bit), Binary Tree 3 bits (BT 3-bit), and DFSA, we assume that the tag's ID consists of 10 bits (all random bits). Note that we cannot simulate 64-bit tag's ID because of the limitation of memory requirement. Figure 3 compares the performance of different algorithms, where the x-axis represents the number of used tags in percentage, and the y-axis is the total number of used slots. The less the total numbers of used slots, the faster the algorithm's speed. It is clear that the BT performs better than the DSFA, especially when the number of tags is large. This is because the DFSA divides groups of tags randomly into slots. Thus, tags are more likely to collide, especially when a large number of tags present in the reader's field. Furthermore, the BT 1-bit performs better than the BT 3-bit when the number of used tags is less than 25%, but worse than the BT 3-bit when the number of used tags is larger than 25%. Therefore, the selected algorithm depends on the number of used tags for a given application.

B. Binary Tree with multiple manufacturer codes

In Figure 3, we assume that the tag's ID consists of 20 bits. Here, we consider the case where the IC manufacturer code is known and can be divided into one, two, and three groups (i.e. the first 10 bits are the same for each group, the last 10 bits are random numbers). We expected that the number of groups affects the performance of algorithms. Figure 4 compares the performance of BT with 1, 2, and 3 manufacturer codes, where each point is averaged by 10 data sets.

It is apparent from Fig. 4 that the BT with 1 manufacturer code performs better than that with 2 and 3 manufacturer codes. As expected, the results confirm that the more the difference in the IC manufacturer codes, the more the number of slots required to identify all tags. Therefore, for one application, it is preferable to use all tags from one manufacturer if possible.

C. Smart Binary Tree algorithm

In this section, we compare the performance of our proposed anti-collision algorithm with the existing ones. We consider two cases of a-priori information, i.e., when the total number of tags is known and when the manufacturer codes are known.

The proposed algorithm that knows when the total number of tags needed to identify is the normal anti-collision algorithm, but it will stop the searching process when all tags are identified. We observed that there is no significant performance improvement (not shown here) when the reader knows the total number of tags needed to identify. This is because the normal algorithm will also stop the searching processing when no tag responds when querying.

However, if a-priori information about manufacturer codes is known, we can then improve the performance of anti-collision algorithms. Let us denote "Smart BT *n*-bit" as the BT *n*-bit algorithm that exploits such a-priori information. We also assume that the tag's ID number consists of 19 bits (the first 9 bits represent a manufacturer code and the last 10 bits represent a random ID number (again, we cannot simulate a 64-bit tag's ID number because of the limitation of memory requirement). Then, with the Smart BT algorithm, the searching process skips the 9-bit manufacturer code, and starts the normal BT algorithm at the first bit of the 10-bit ID number.

Figure 5 compares the performance of BT and Smart BT algorithms with one manufacturer code. Clearly, the Smart BT performs better than the BT. For the Smart BT algorithm, the decision point to decide whether or not 1-bit or 3-bit searching process should be used is approximately at 50% of the number of used tags, whereas for the BT algorithm, the decision point is at 26% of the number of used tags.

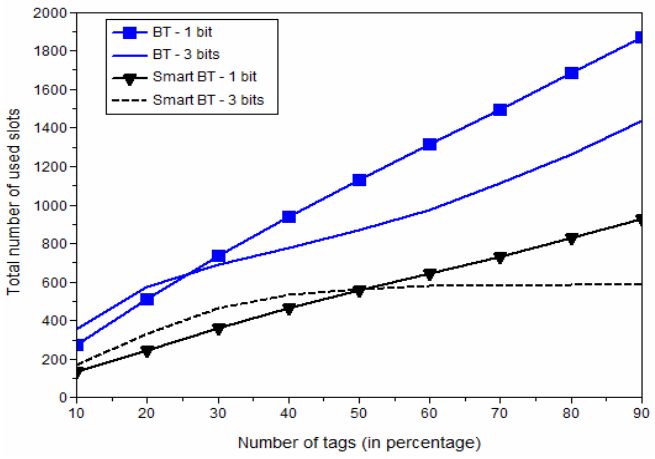

Fig. 5. Performance of BT and Smart BT with one manufacturer code.

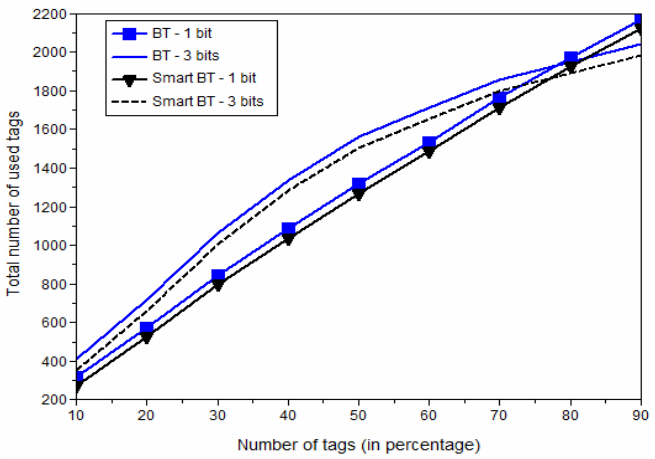

Fig. 6. Performance of BT and Smart BT with three manufacturer codes.

TABLE I
COMPARISON OF THE NUMBER OF USED SLOTS

% of tags	One manufacturer code (total slots)				Percentage of slot reduction (%)	
	Normal		Smart			
	BT	BT 3 bit	BT	BT 3 bit	BT	BT 3 bit
30%	736	844	362	462	50.815	45.26
50%	1133	1316	557	564	50.838	57.142
70%	1497	1762	733	581	51.035	67.026

TABLE I
DETAILS OF EACH ALGORITHM

Type / Details	FSA	DFSA	BT 1-bit	BT 3-bit
1) Speed	slow	normal	fast	normal
2) Ability to add tags while working	√	√	X	X
3) Complexity	normal	highest	low	low
4) Security of tag's IDs	√	√	X	X

Table I shows the total number of slots used in the Smart and *Normal* BT algorithm (extracted from Fig. 5). We found that the Smart BT algorithm requires the number of slots less than the BT algorithm, approximately 50%. We also compare the performance of BT and Smart BT algorithms with three manufacturer codes as depicted in Fig. 6. Clearly, the performance improvement is not significant. In this case, the Smart BT algorithm performs well when the numbers of tags are known prior to the communication, but the manufacturer codes of three companies have no role in time slot reduction. It is not possible for the reader to know beforehand which tags of three companies will be first read and thus keep sending the new command until no collision occurs. However, the smart BT algorithm will in general perform better than the normal BT algorithm.

V. CONCLUSIONS

Based on simulation results, we can summarize the performance comparison among existing anti-collision algorithms as illustrated in Table II. The speed is referred to as the operation time used in each algorithm. The complexity is referred to as the system request memory, computation, and another function on tags.

The anti-collision algorithms are crucial to the application that uses a lot of tags. In general, the Binary Tree algorithm performs faster than the DFSA algorithm as shown in Fig. 3. Furthermore, one should employ tags with one manufacture code in each application to expedite the identification process. In this paper, the anti-collision algorithm that exploits a-priori information about the manufacturer code is proposed. Clearly, the proposed algorithm performs better than any existing anti-collision algorithm in terms of the number of used time slots.

ACKNOWLEDGMENT

This work was 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.

REFERENCES

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

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

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

[4] 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.

[5] 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.

[6] 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.

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

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