# Performance Analysis of Tag Anti-collision Algorithms for RFID Systems

Cheng-Hao Quan[1], Won-Kee Hong[2] and Hie-Cheol Kim[2]

[1] RFID System Research Team,
ETRI, Daejeon, Korea
{chquan}@etri.re.kr
[2] School of Information and Communication Eng.,
Daegu University, Gyeongsan Gyeongbuk, Korea
{wkhong, hckim}@daegu.ac.kr

**Abstract.** Lately, the ISO fixed on UHF Gen2 as one of the standard protocols for RFID, called ISO 18000-6 C, along with ISO 18000-6 A/B. It means that the RFID system should provide the multi-protocol support for tag identification and a proper protocol should be chosen depending on the situation. The tag anti-collision algorithm is one of the important research issues to be on top of the protocol's performance. This paper introduces several anti-collision algorithms for tag identification in the literature and presents the performance comparison and evaluation of those algorithms based on the 96-bit EPC™ (Electronic Product Code™). The performance results show that the collision tracking tree algorithm is found to have the highest performance than any other anti-collision algorithm, identifying 749 tags per second.

## 1 Introduction

RFID (Radio Frequency IDentification) technology, which identifies electronic tags on objects using RF signal without contact, is spotlighted as a key technology in implementing ubiquitous environment. In the RFID system, tag identification is performed by the reader's query to a tag attached an object and the tag's transmission of its identifier to the reader. If there is only one tag in the reader's identification area tag identification may be simple, but if there are multiple tags in the area they respond to the reader's query at the same time and, as a result, collisions happen among the tags within the reader's communication range. Such collisions hinder the reader from accurate tag identification. Specifically, in large-scale electronic supply chain systems that process a large amount of goods in real time, anti-collision algorithm is essential to perform multiple tag identification [1].

Recent researches on RFID system are mainly made for low-cost RFID systems at UHF band that has a long recognition distance and is less influenced by surrounding environment. These researches are focused on system construction but not many of them deal with anti-collision and high-speed identification of multiple tags. They are based on different tag systems with different types of

identifiers like 8, 16 and 32-bit identifiers. In order for RFID system to be widely used, standardization of information system stored in tags should be resolved. Owing to the effort toward standardization by EPCglobal led by Auto-ID Center, the EPC™ (Electronic Product Code™), a tag information system is a de facto standard code of RFID [4]. Different from traditional bar codes EPC code allows each individual object to have its unique code system and thus makes it possible to obtain various data such as the location and the condition of the object as well as to manage and utilize the data effectively. An EPC code is composed of four fields - header, manufacturer code, product code and object code - and its size is 96 bit or 128 bit.

This paper introduces tree-based memoryless anti-collision algorithms and slot aloha-based anti-collision algorithms based on EPC code with 96-bit identifier in RFID system and evaluate their performance. According to the result of performance evaluation, collision tracking tree algorithm [9], which is developed by out research team and one of tree-based memoryless anti-collision algorithms, is superior to others in the number of queries-responses and the number of bits transmitted, showing $2 \sim 50$ times higher performance than other algorithms in the average number of tags identified per second.

This paper is composed as follows. Sect. 2 reviews previous researches on multiple tag identification for identifying multiple tags in the reader's range. Sect. 3 and 4 explains the basic concept, operating process and examples of tree-based memoryless anti-collision algorithm and slot aloha-based anti-collision algorithm, respectively. Sect. 5 evaluates the performance of existing tree-based memoryless algorithms and slot aloha-based algorithms and presents the results of analysis. Conclusions are made in Sect. 6.

## 2 Related Work

Anti-collision algorithms for multiple tag identification are largely divided into tree-based deterministic algorithm and slot aloha-based probabilistic algorithms. Deterministic algorithms form a binary tree with tag identifiers expressed in binary bits and identify tags through browsing the nodes in the tree. In this type of algorithms, we can predict the process of tag identification. This type of algorithm is again divided into memory algorithms and memoryless ones. In memory algorithms, the response of a tag is determined by the query to the tag and the current state of the tag and thus each tag must store and manage its state information. Representative memory algorithms are splitting tree algorithm [3] and bit-arbitration algorithm [5]. In memoryless algorithms, on the other hand, the response of a tag is determined only by the query to the tag. This type of algorithms is a good approach for simple implementation of tags as well as for low cost, low power and small size. Representative memoryless algorithms are tree-walking algorithm [6], query tree algorithm [8] and memoryless collision tracking tree algorithm [9]. Probabilistic algorithms are based on aloha protocol. Each of tags in a reader's identification area selects one of given $N$ slots to transmit tag information and sends its identifier. Thus, tag collision can be avoided by time

difference among the slots. However, because it is not easy to count the exact number of tags in the identification area, the optimal number of slots and the time that transmission of tag information are completed should be determined probabilistically. Probabilistic algorithms are again divided into ID-slot algorithms and Bit-Slot ones. In ID-slot algorithms each tag puts its identifier to a slot and sends the slot while in Bit-Slot algorithms each tag creates information composed of special bits, fills a slot with the information and sends it to the reader. Representative ID-slot algorithms are I-Code algorithm [10] and STAC (Slotted Terminating Adaptive Collection) algorithm [1] and a representative Bit-Slot algorithm is anti-collision algorithm [7] using the Bit-Slot mechanism.

## 3 Tree-based Memoryless Anti-collision Algorithm

Tree based memoryless anti-collision algorithms can implement tags at a low cost because tags do not need to maintain their state information. Still, these algorithms need a memory to store some bits of an identifier in the reader during the process of tag identification and, for this, they have a memory structure of stack or queue. Here, we have a brief review of representative tree-based algorithms - tree-walking, query tree and collision tracking tree algorithm.

In tree-walking algorithm, the reader begins a query to tags using $k$-bit prefix ($B_{(0,k)}$), which is a bit string from the 0th bit ($b_0$) to the $k$th bit ($b_k$) of a tag identifier for tag-reader communication. Each tag in the area checks the received prefix against its identifier and, if they match with each other, the tag sends the $k+1$th bit ($b_{k+1}$) of the tag identifier to the reader. Here, tag responses can be either of two types. First, it is the case that all bits received from tags within reader's area are '0' or '1'. In this case, the reader creates a new prefix ($B_{(0,k+1)}$) by adding the received bit value to the existing prefix. Second, received bits contain both '0' and '1'. This means that a collision has happened. The reader stores the prefix that had a collision into the stack and at the same time creates a new prefix ($B_{(0,k+1)}$) by adding '0'. The new prefix is send to the tag in the next query-response process. This process is repeated as many times as the number of bits of the tag identifier until a tag is identified. If a tag is identified, the prefix stored in the stack is retrieved and a new prefix ($B_{(0,k'+1)}$) is created by adding '1' to the prefix, and using the new prefix a new query-response process is performed. If the stack is empty, it means that the whole tag identification process has been completed and all tags in the area have been identified.

In query tree algorithms, tags matching with $k$-bit prefix ($B_{(0,k)}$) sent by the reader send bit strings from the $k+1$th to the last of their identifier ($B_{(k+1,l)}$) to the reader in all. Here, tag responses can be either of two types. First, only one tag or no tag responds. In this case, a new prefix ($B_{(0,k')}$) is created using a prefix stored in the queue. Second, multiple tags respond together and collisions occur among them. In this case, '0' and '1' are added to the existing prefix and the prefixes are stored in the queue respectively. In addition, a new prefix($B_{(0,k')}$) is retrieved from the queue and is used in the next query-response. This process is continued until all tags in the area are identified. If the queue is empty, it means

that the whole tag identification process has been completed and all tags in the area have been identified.

In collision tracking tree algorithms, the reader makes a query to tags using a $k$bit prefix($B_{(0,k)}$) as a parameter. Each tag in the area checks the received prefix against its identifier and, if they match with each other, the tag sends the reader a bit string ($B_{k+1,l}$) from the $k+1$th bit ($b_{k+1}$) to the last bit ($b_l$) of the tag identifier in order. On receiving identifier information from tags, the reader determines if there is a collision in the received bits. If a collision occurs as '0' and '1' are received at the same time, the reader stops receiving bits and orders the tag to stop the transmission of identifier. If all bits are '0' or '1' the reader continues to receive the remaining bits. If the last bit of identifier information is received without collision, a tag is identified. If a collision occurs, different from tree-walking algorithms or query tree algorithms that add '0' or '1' to the existing prefix, a new prefix is created by adding '0' or '1' to all bits received without collision and is saved to be used as a parameter in the next query-response. If a tag is identified, a new prefix is stored in the stack or the queue to identify another tag in the area. The process is repeated until all tags in the area are identified. If the stack or the queue is empty, it means that the whole tag identification process has been completed and all tags in the area have been identified.

## 4 Slot Aloha-based Anti-collision Algorithm

Slot aloha-based anti-collision algorithms are based on aloha protocol, and some of them are I-Code algorithm, STAC algorithm and Bit-Slot algorithm. In I-Code algorithm, a reader cycle, namely, a query-response process is progressed using a frame composed of a number of slots. For tag identification, the reader sends $< I, rnd, N >$ information to tags ($I$: the range of tag identifier, $rnd$: seed value for creating a random value, $N$: the number of slots in the frame). Here, each tag selects a slot from the frame at random, loads its identifier into the slot and sends it to the reader. The reader identifies tags using identifiers loaded into the slots of the frame. This process is repeated until all tags in the area are supposed to have been identified. The identification process has two problems related to the determination of frame size ($N$), and the exact guess of completion time of identification.

First, if $N$ is too large it causes the waste of time slots, and if it is too small it causes collisions among tags. I-Code algorithm uses the following method to determine $N$. In each reader cycle, slots in a frame containing responses from tags can be: 1) empty; 2) loaded with one tag identifier; or 3) loaded with multiple tag identifiers. Given frames received by a reader, a slot distribution of those frames based on the classification can be expressed as follows: $< c_0, c_1, c_k >$. Here, $c_0$ is the number of empty slots, $c_1$ is the number of slots loaded with one tag identifier, and $c_k$ is the number of slots loaded with multiple tag identifiers. In a reader cycle, the minimum bound of the number of tags in the identification area denoted by $n$ can be calculated by the equation '$n = c_1 + 2c_k$'. Depending

on the calculated $n$ [10], a new frame size ($N$) to be used in the next query-response process is determined. Second, probability-based I-Code algorithm is not easy to know the point of time when tag identification is completed. I-Code algorithm solves the problem by introducing a model based on the homogeneous markov process to tag identification process.

In case of STAC algorithm, if an empty slot or a collision slot is detected, the reader stops the transmission of the slot and sends tags the command 'close slot sequence' that triggers the transmission of a new slot. This reduces unnecessary overhead and improves performance.

In Bit-Slot algorithm, a frame is composed of special bits and the operating process of the algorithm is as follows. In response to the reader's query, each tag in the area generates a random value of the same size as that of tag identifier and sends it to the reader. The created value has '1' only in one bit and '0' in all the other bits. The reader inspects the bits of the received frame in order. If there is no bit with '1' in the corresponding position, it means that there is no response. Transmission of two or more bits with '1' means a collision. If there is only one bit with '1', the tag is identifiable and the received random value is sent to the tags in the reader's area. Only the tag that sent the corresponding random value sends its identifier to the reader and the tag is identified. In tag identification, the process of selecting one out of multiple tags in the area is called tag singulation. If tag singulation is finished, the tag sends its identifier to the reader. Different from I-Code or STAC algorithm, Bit-Slot algorithm divides tag identification process into tag singulation and tag identifier transmission, and in tag singulation frame size is the same as the bit length of a tag identifier and a frame is composed of bits.Its tag identification speed is fast because the frame size is small.

## 5 Performance Evaluation

This section evaluates the performance of tree-based anti-collision algorithms and slot aloha-based anti-collision algorithms examined above. The performance of slot aloha-based anti-collision algorithms is determined by the number of tags in the reader's area and frame size and is nothing to do with the value of tag identifiers. On the contrary, the tag identification process of tree-based algorithms is determined by the value of tag identifiers. Moreover, the bit length of tag identifier has a significant effect on the performance of the algorithms. Thus, only identifier bit length of 96 bits for slot aloha anti-collision algorithms is considered. Different from existing 8, 16 and 32-bit identifiers, 96-bit EPC is currently promoted as an international standard by EPCglobal. Thus, 96-bit tag identifiers to analyze algorithm performance is used.

In this section, we first compare the performance of tree-based anti-collision algorithms for 8, 16 and 32-bit identifier to examine the effect of identifier's bit length on the performance. The performance of tree-based anti-collision algorithm and slot aloha-based anti-collision algorithm are analyzed based on the following premise. The maximum number of tags within a reader's area assumed

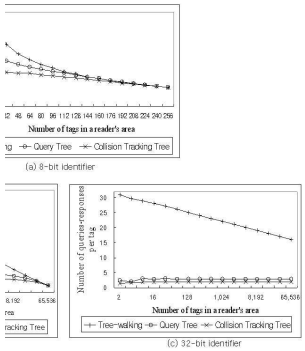


**Fig. 1.** Comparison of the number of queries-responses per tag in tree-based memoryless anti-collision algorithm

for tree-based anti-collision algorithm is 65,536($2^{16}$), although this is larger than the practically possible highest level of around 4,000 $\sim$ 8,000. The number of tags is increased by two times from 2 to 65,536 and, for each number of tags, the number of queries-responses, the number of bits transmitted and the number of tags identified per second are analyzed. The maximum number of tags assumed for slot aloha-based anti-collision algorithm is 2,048. We set the maximum frame size at 512 slots because the number of tags is related to frame size and the available bandwidth is limited in slot aloha-based anti-collision algorithm. The number of tags is also increased by two times from 2 to 2,048. Bit transmission rate used in measuring the number of tags identified per second is assumed to be 80 Kbps. In addition, each query command is assumed to be 8 bit and time to detect and process no-response and collision assumed to be 3 bit unit time [1], [2], [4]. Values used in performance evaluation is the averages of data obtained from experiment repeated 10 times.

### 5.1 Performance Analysis of Tree-based Anti-collision Algorithms

Fig. 1 shows the number of queries-responses per tag and the number of bits transmitted per tag in tree-based anti-collision algorithms when the identifier is 8, 16 and 32 bit long. The change in the number of queries-responses per tag shows the following two facts. First, with the increase of identifier bit length, difference in the number of queries-responses per tag grows larger among query tree algorithm, collision tracking tree algorithm and tree-walking algorithm. This suggests that a long tag identifier is inefficient for tree-walking algorithm. Second, with the increase of the number of tags, the number of times of query-response per tag decreased significantly in tree-walking algorithm but not in query tree algorithm and collision tracking tree algorithm. This suggests that tree-walking algorithm is more efficient when the number of tags in a reader's area is large.

Fig. 2 shows a change in the number of bits transmitted per tag. In the figure, collision tracking tree algorithm is superior to the other cases because it does

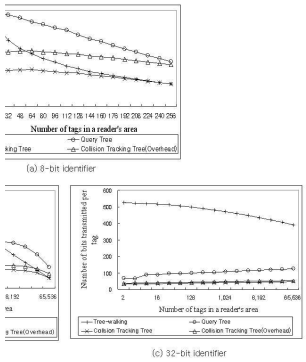


**Fig. 2.** Comparison of the number of bits transmitted in tree-based memoryless anti-collision algorithm

not consider the overhead of collision tracking. Collision tracking tree algorithm that takes overhead into accountassumes that it takes 3 bit unit time to detect and process collision. According to the result of Fig. 2, an appropriate algorithm should be chosen according to the bit length of tag identifier and the number of tags in the reader's area, but if the bit length of tag identifier is over 32, the collision tracking tree algorithm should be considered first.

The performance of tree-based anti-collision algorithms is mainly determined by the number of queries-responses and the number of bits transmitted. We examine a change in the number of queries-responses and the number of bits transmitted according to the increase in the number of tags when tag identifier is 96 bit long. The number of queries-responses means the number of times of communication between the reader and tags, and each query-response process is counted as one.

Fig. 3(a) shows the number of queries-responses per tag in each algorithm. In identifying 96-bit tags in the reader's area, it is 1.9 in collision tracking tree algorithm and 2.9 in query tree algorithm. It is 88 in tree-walking algorithm, much more than the other algorithms. This is because, in tree-walking algorithm, the query-response process is repeated for each bit of tag identifier and, as a result, the number of times of query-response increases in proportion to the bit length of tag identifier. As in Table 1, 98.92 % of responses does not have

**Table 1.** Percentage of collision, no-collision, and no-response

| | Tree Walking | Query Tree | Collision Tracking |
|---|---|---|---|
| Collision | 1.08 % | 49 % | 48.10 % |
| No-collision | 98.92 % | 35 % | 51.90 % |
| No-response | - | 16 % | - |

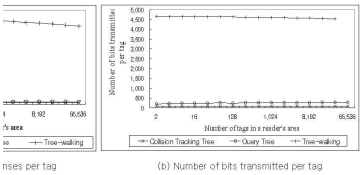


**Fig. 3.** Performance evaluation of tree-based memoryless anti-collision algorithms

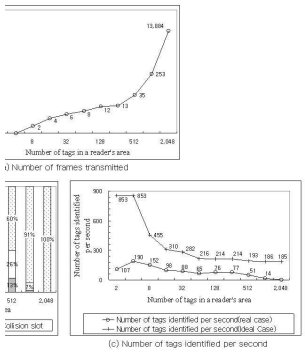


**Fig. 4.** Performance evaluation of slot aloha-based anti-collision algorithms

collision in tree-walking algorithm. This is because the number of tags in the reader's area is small compared to the size of the bit space and query-response process is performed for each bit until the tag is identified. On the other hand, query tree algorithm sends the entire tag identifier to the reader but it causes frequent no-responses and collisions. Compared to query tree algorithm, collision tracking tree algorithm reduces the number of queries-responses with collision by 1.54 times and removes no-response queries and, as a result, shows higher performance in the number of queries-responses than query tree algorithm.

The number of bits transmitted is calculated by the sum of the number of query bits and response bits as mentioned in Sect. 3. Fig. 3(b) shows the number of bits transmitted per tag in each algorithm. In tree-walking algorithm, it is much larger than that in other algorithms. This is because tree-walking algorithm has a large number of queries-responses and identifies a tag by bit basis through increasing the prefix sent to the tag bit by bit until the tag is identified.

### 5.2 Performance Analysis of Slot Aloha-based Anti-collision Algorithm

Fig. 4 shows the result of performance evaluation on slot aloha-based anti-collision algorithm. Fig. 4(a) shows a change in the number of frames transmitted according to the increase in the number of tags. As in the Fig. 4(a), the number of frames transmitted increases exponentially when the number of

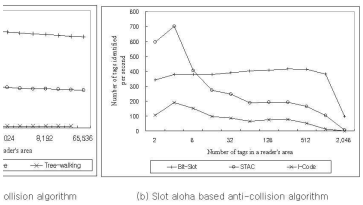


**Fig. 5.** Comparison of the number of tags identified per second in anti-collision algorithms

tags is over 512. As in Fig. 4(b), the percentage of collision slots in transmitted frames is 60 % when the number of tags is 512, 91 % when 1,024 and almost 100 % when 2,048. In addition, the percentage of empty slots is over 55 % when the number of tags is less than 64, and the maximum percentage of identified tags is less than 36 % regardless of the number of tags, suggesting that most slots are wasted away. Fig. 4(c) shows the number of tags identified per second in a worst case and in an ideal case. The worst case means that the entire frame is transmitted regardless of whether there are empty slots or collisions, and the ideal case means that, out of a frame, only slots loaded with a single tag identifier, with which the tag is identified, are transmitted. The number of tags identified per second is calculated by dividing the number of bits transmitted by bit transmission rate. Excluding cases that collision slots occupy over 60 %, the average number of tags identified per second ranges from 77 up to 190.

### 5.3 Number of Identified Tags per Second

Fig. 5 shows the number of tags identified per second in each algorithm according to the number of tags in the reader's area. Fig. 5(a) shows the number of tags identified per second in tree based anti-collision algorithms. In the whole range of the number of tags it is better in collision tracking tree algorithm showing average 749 than any other tree-based algorithm. Fig. 5(b) shows the number of tags identified per second in slot aloha-based anti-collision algorithms. Bit-Slot algorithm shows 362 on average, the largest number of tags identified among slot aloha-based algorithms. Accordingly, among the anti-collision algorithms analyzed above, collision tracking tree algorithm is found to have the highest performance.

## 6 Conclusions

This paper introduces several anti-collision algorithms based on EPC code with 96-bit identifier in RFID system, and evaluates their performance. According to the result of performance evaluation, tree-walking algorithm shows the slowdown of performance as the number of queries-responses and the number of bits transmitted increases excessively with the increase in the bit length of tag

identifier. In case of query tree algorithm, despite the transmission of the entire tag identifier information, high performance can not be expected because it uses a collision detection technique. I-Code algorithm and STAC algorithm, the tag identification performance of which depends on frame size, are usable only when the number of tags in the area is small but they cannot produce high performance either because of communication overhead resulting from tag collisions or empty slots.

Bit-Slot algorithm is superior in performance to I-Code algorithm or STAC algorithm because of its small frame size, but it also has a limitation because the tag identification process is divided into tag singulation and tag identifier transmission. Lastly, collision tracking tree algorithm, which is one of tree-based memoryless algorithms, performs query-response by tracking the exact location of collision and, as a result, shows much higher performance in the number of queries-responses and the number of bits transmitted than other algorithms. According to the result of simulation, collision tracking tree algorithm identified 749 tags on the average per second in identifying a maximum of 65,536 96-bit tags, showing $2 \sim 50$ times higher performance than other algorithms.

## References


1. Auto-ID Center (ed.), "13.56MHz ISM Band Class 1 Radio Frequency Identification Tag Interface Specification: Candidate Recommendation", Auto-ID Center, 2003
2. EPCglobal (ed.), "EPC$^{TM}$ Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz $\sim$ 960 MHz Version 1.0.9.", EPCglobal, 2005
3. Hush, D. R., Wood, C., "Analysis of Tree Algorithms for RFID Arbitration", in Proc. of Int. Symp. on Information Theory, pp. 107–114, 1998
4. ISO/IEC (ed.), " Information Technology – Radio-Frequency Identification for Item Management – Part 6: Parameters for Air Interface Communications at 860 MHz to 960 MHz", ISO/IEC, 2004
5. Jacomet, M., Ehrsam, A., Gehrig, U., "Contactless identification device with anti-collision algorithm", in Proc. of IEEE Conf. on Circuits, Systems, Computers and Communications, pp.4–8, 1999
6. Juels, A., Rivest, R., Szydlo, M., "The Blocker Tag: Selective Blocking of RFID Tags for Consumer Privacy", in Proc. of ACM Conf. on Computer and Communication Security, pp.103–111, 2003
7. Kim, C.-S., Park, K.-L., Kim, H.-C., Kim, S.-D., "An Efficient Stochastic Anti-collision Algorithm using Bit-Slot Mechanism", in Proc. of Int. Conf. on Parallel and Distributed Processing Techniques and Applications. 2004
8. Law, C., Lee, K.,Siu, K.-Y., "Efficient Memoryless protocol for Tag Identification", in Proc. of Int. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, pp.75–84, 2000
9. Quan, C.-H., Hong, W.-K., Lee, Y.-D., Kim, H.-C., "A Study on the Tree basesd Memoryless Anti-Collision Algorithm for RFID Systems", The KIPS Transactions. Vol. 11. Korean Informantion and Processing Society, Korea, pp.851–862, 2004
10. Vogt, H., "Efficient Object Identification with Passive RFID Tags", in Proc. of Int. Conf. on Pervasive Computing, 2002