INTERNATIONAL JOURNAL OF INTEGRATED ENGINEERING
ISSN: 2229-838X e-ISSN: 2600-7916

IJIE

Vol. 16 No. 3 (2024) 257-272
https://publisher.uthm.edu.my/ojs/index.php/ijie

Group Collision Tracking Tree for Passive Multi-Tags RFID Systems

Marizan Yaacob[1]*, Noraimi Shafie[2], Norliza Mohamed[2], Azizul Azizan[2]*

[1] *Pusat Pengajian Diploma SPACE,*
Universiti Teknologi Malaysia, Kuala Lumpur, 54100, MALAYSIA
[2] *Faculty of Artificial Intelligence,*
Universiti Teknologi Malaysia, Kuala Lumpur, 54100, MALAYSIA

*Corresponding Author: marizan.kl@utm.my
DOI: https://doi.org/10.30880/ijie.2024.16.03.023

Article Info

Received: 20 August 2024
Accepted: 24 September 2024
Available online: 9 November 2024

Keywords

Radio Frequency Identification (RFID), tag collisions, collision tracking tree

Abstract

Radio Frequency Identification (RFID) systems are progressively replacing traditional barcode systems by utilizing cost-effective passive RFID tags, which are powered by electromagnetic waves emitted by readers. A significant challenge in multi-tag RFID systems is managing tag collisions, where multiple tags may respond simultaneously to a reader's query. This paper introduces the Group-Collision Tracking Tree (GCTT) algorithm, an advanced tree-based method designed to tackle this issue while maintaining the principles of memoryless algorithms. The GCTT algorithm categorizes tags into two groups based on the Most Significant Bit (MSB) of their IDs and employs Manchester coding combined with a recursive approach to efficiently manage and resolve collisions. This technique effectively organizes tags into segmented groups, minimizing the likelihood of collisions and optimizing the tag-reading process. Simulation results demonstrate that GCTT significantly outperforms existing Collision-Tracking (CT) and Bi-response Collision Tree (BCT) algorithms in terms of response time, while maintaining low time complexity and efficient memory utilization. The recursive functionality of GCTT allows it to manage large volumes of tags with minimal memory overhead, thus preserving the memoryless nature of the algorithm. Consequently, GCTT proves to be highly effective for large-scale RFID systems.

1. Introduction

Various technologies have been proposed to address tag collision issues in RFID systems, and they typically fall into two main categories: ALOHA-based protocols and tree-based protocols. ALOHA-based protocols aim to reduce tag collisions but often suffer from the problem of tag starvation, where some tags cannot be identified for extended periods. In contrast, tree-based protocols are considered the primary and evolving anti-collision technology due to their high throughput and adaptable algorithms. Deterministic or tree-based protocols can be further categorized into Query Tree, Binary Search, Tree Splitting, and Bit Arbitration (BTA) protocols.

Some protocols, like Query Tree (QT)[1], do not require additional memory and are referred to as memoryless protocols. In a memoryless tree-based algorithm, a reader sends a prefix of the Electronic Product Code (EPC) to query tags, and tags that match the prefix respond. By progressively extending the prefixes until only one tag's ID matches, successful tag identification is achieved. Memoryless tree-based algorithms have minimal tag circuitry requirements, storing only the tag's ID and a prefix matching circuit to match it with reader queries. These algorithms do not involve counters or additional circuitry to remember states or record paths.

Query Tree (QT) is an example of a memoryless algorithm with minimum tag circuitry specifications. Variants of Query Tree (QT), Query Tree Improved [1] [2] have attempted to maintain the memoryless aspect, with most improvements made on the reader side by enhancing query methods. However, deterministic anti-collision protocols still face identification delays, which increase as the size of the ID and the number of tags increase. While many proposed protocols aim to enhance RFID tag identification performance by improving efficiency and reducing energy consumption, they often come at the cost of increased protocol complexity. This complexity can limit the practicality of these protocols, especially in passive RFID tag systems.

The time required to identify an RFID tag can vary based on several factors, including the RFID technology used, tag response time, the anti-collision algorithm, and the system configuration. Generally, the tag identification process involves several steps, including query transmission, tag response time, collision detection and resolution and data extraction from the identified tag.

Here is a breakdown of the steps contributing to tag identification time: Query Transmission: The RFID reader sends a query signal to the tags within its reading zone, typically containing reader identification and a response command. Tag Response Time: Tags receiving the query respond within a specific time frame, which varies based on tag type and technology. For instance, passive UHF RFID tags can respond in a fraction of a millisecond. Collision Detection and Resolution: When multiple tags respond simultaneously, collisions can occur. Anti-collision algorithms manage and resolve these collisions, with the time required depending on the algorithm's efficiency. Data Extraction: Once a tag's response is successfully received and resolved, the RFID reader extracts the tag's encoded data by processing the received signal.

The tag identification time can significantly vary based on factors like the number of tags present, RFID technology (passive or active), anti-collision algorithm efficiency, and the complexity of tag data. In modern RFID systems, single tag identification is often rapid, typically taking fractions of a second. However, in scenarios involving numerous tags or complex data, identification times can increase due to collision resolution and data extraction processes.

Time complexity of an RFID anti-collision algorithm involves how the algorithm's performance scales with an increasing number of tags [3]. Time complexity provides insight into how the computational workload grows relative to the input size (in this case, the number of tags). Commonly, the Big O notation is used to express time and space complexity, indicating an upper bound on the algorithm's execution time growth rate.

2. Radio Frequency Identification (RFID) System

The three main components of an RFID system are tags, RFID readers and back-end server. A tag is an identification device attached to an item, which use radio frequency (RF) to communicate. The reader is a device that can recognize the presence of RFID tags and read information supply by them. The readers query tags by broadcasting an RF signal, and the tag response to the reader with a number or other identifying information. The reader forwards the tags response to the back-end server. The server has a database of tags and can retrieve detailed information regarding the tag from the tag response.

RFID tags can be classified into three types: passive tag, active tag, and semi-passive tag. Passive tags have no internal power source. They have to rely solely on the reader's energy in order to energize and transmit. It relies the continuous wave (CW) power emitted by reader to perform the identification process. The communication from passive tag to reader is usually backscatter (high frequency passive tags) or load modulation (low frequency passive tags). Because of power constraint, the communication range for passive tags is from 2.5 cm up to 10 cm. Typically operate at frequencies of 128 kHz, 13.5 MHz, 915 MHz, or 2.45 GHz. Passive tags are the cheapest, thinnest and most flexible among the three which makes them very popular in the supply chain and retail industry [4].

A passive tag has constraints in functionality, but it has the distinct feature to the active tag. It is enough small to attach to an object easily. There could be multiple tags a reader should identify. All the functionality tags should do is that they response with the data corresponding to the signal received from a reader. The communication between tags is impossible. Passive tags cannot make a decision of whether the channel is busy or not. A collision is occurred at the reader's side when two more tags get transmitted simultaneously. Therefore, the arbitrational mechanism is required. The protocol aiming to avoid collisions between a reader and tags anti-collision protocol. The anti-collision protocol should have the following characteristics. A reader should identify all the tags within its range. The anti-collision algorithm should have a mechanism which is capable of verifying that all the tags are identified and it should minimize the time elapsed for the identification of tags. It lies on same line as reducing collisions.

Int. Journal of Integrated Engineering Vol. 16 No. 3 (2024) p. 257-272

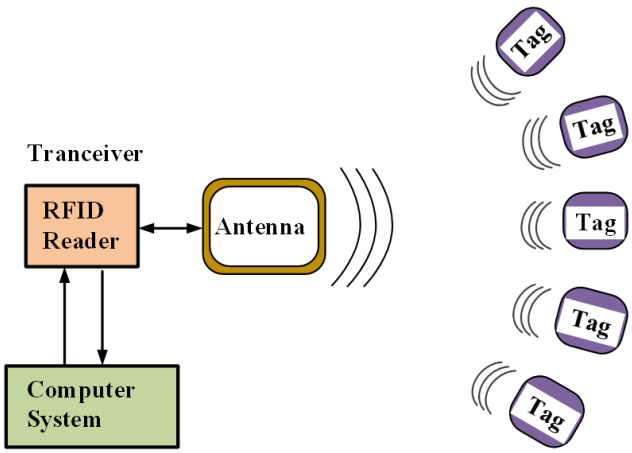

Fig. 1 *RFID system*

2.1 Anti-Collision Algorithm

The mainstream anti-collision protocols can be divided into Aloha-based and tree-based protocols. Tree–based is a type of deterministic algorithm. Fig. 2 shows the RFID anti-collision protocols. Collision detection and resolution are critical processes in RFID systems, especially in scenarios where multiple RFID tags are present within the reading range of an RFID reader and respond to a query simultaneously. Tag collisions can occur when the signals from multiple tags interfere with each other, making it difficult for the reader to distinguish individual tag responses. Collision detection and resolution algorithms are employed to manage these situations and ensure accurate data retrieval. RFID reader sends out a query signal to initiate communication with nearby RFID tags. RFID tags within the reading zone receive the query and respond by ending their unique identification numbers or other data encoded in their memory. When multiple tags respond at the same time, their signals can overlap and interfere with each other. This results in a collision, where the reader cannot clearly identify the responses of individual tags. The RFID reader detects that a collision has occurred by observing inconsistencies or interference patterns in the signals received from the tags. Anti-Collision Algorithm is to resolve collisions and identify tags individually, anti-collision algorithms are employed. These algorithms manage the way tags are queried and responded to in a controlled manner.

The anti-collision algorithm iterates through multiple rounds of querying and tag responses. In each iteration, the algorithm identifies a subset of tags that can respond without causing a collision. This process continues until all tags are successfully identified. Tags that have been identified are usually acknowledged and excluded from subsequent queries. This ensures that only the remaining unidentified tags are queried in subsequent rounds. After successful identification, the reader extracts the data or information stored on each identified tag. Efficiency in collision detection and resolution algorithms is crucial, in term of the performance of an RFID system especially in scenarios with a high density of tags. The goal is to minimize the number of iterations required to identify all tags accurately. Different anti-collision algorithms, such as the Binary Tree Protocol and the Aloha protocol, are designed to optimize the identification process based on the specific characteristics of the RFID system and its application.

The principle of Aloha-based protocols is based on random back off mechanism. Thus, as a representative of a probabilistic protocol, Aloha-based protocol can be further divided into Pure Aloha (PA), Slotted Aloha, Frame Slotted Aloha (FSA)[5][6][7] and Dynamic Frame Slotted Aloha (DFSA) [8] [9] [10]. In Aloha-based protocols, each tag responds to the reader with ID information at a randomly assigned slot. If a collision is detected, a tag should be required to retransmit its ID. The implementation process of Aloha-based protocols is straight and easy, but its system efficiency is usually low especially as the number of tags increases. In addition, due to the randomness of Aloha-based protocols, they suffer from the tag starvation problem that some tags may not be successfully identified for a long time period.

A Tree-based method can be classified into a Memory-based algorithm and Memoryless-based algorithm [7]. In memory-based algorithm, which can be grouped into a splitting tree and bit arbitration algorithm. Memoryless is dedicated to Query Tree (QT) algorithm. Query Tree (QT), Tree Splitting (TS), Binary Search (BS) and Bitwise Arbitration (BTA) protocols [11][12[13] are tree-based algorithm of RFID system. The binary search (BS) algorithm is a collision detection mechanism based on the Manchester coding technique and its variants have improved the binary search algorithm [14],[15],[16]. However, some methods use a 4-ary dynamic search, which creates an idle cycle if the number of tags is less or has random ID bits. The 4-ary query is useful for a group of large tag numbers that have some similarity in the tag ID bits. The BSF1 and BSF2 algorithms [17] are improvements from Binary Search (BS). These algorithms implement a grouping method and it is very good for

robust applications, however it creates many idle cycles due to the backtracking process to the parent node after successfully reading the tag. However, the backtracking procedure allows the algorithm to adapt to new arriving tags.

Collision tracking tree (CT) [18] is the variant of QT, but it used Manchester coding to track the collision bit. The major improvement is the elimination of idle cycle which contribute the system efficiency of 50%, however it used a lot of stack and memory to record the path and node. The variant of CT is ICT [19] and BCT [20] which having some improvement. ICT adapts a duality technique where the reader will read two tags at the same time if there is only one collision bit detected from the Manchester-coding mapping, however this adds to the complexity of the algorithm. Meanwhile for BCT, in each node, the two group of tags answer the reader's same query in two sub-cycles: R0-cycle and R1-cycle, respectively by transmitting their remainder IDs if their IDs match the received prefix. This method requires another mechanism to isolate and read R0 and R1 cycles on the same response wave. This tends to increase the complexity in the actual system implementation.

System efficiency and system time efficiency [21] have been used to measure the performance of anticollision algorithms. However, this measurement is only subject to the number of collision round, idle round and successful round which does not reflect the overall performance of the algorithm. This research measures performance by measuring the response time, which is the time taken for reading all tags and also with Big O Notation to verify the algorithm time and space complexity.

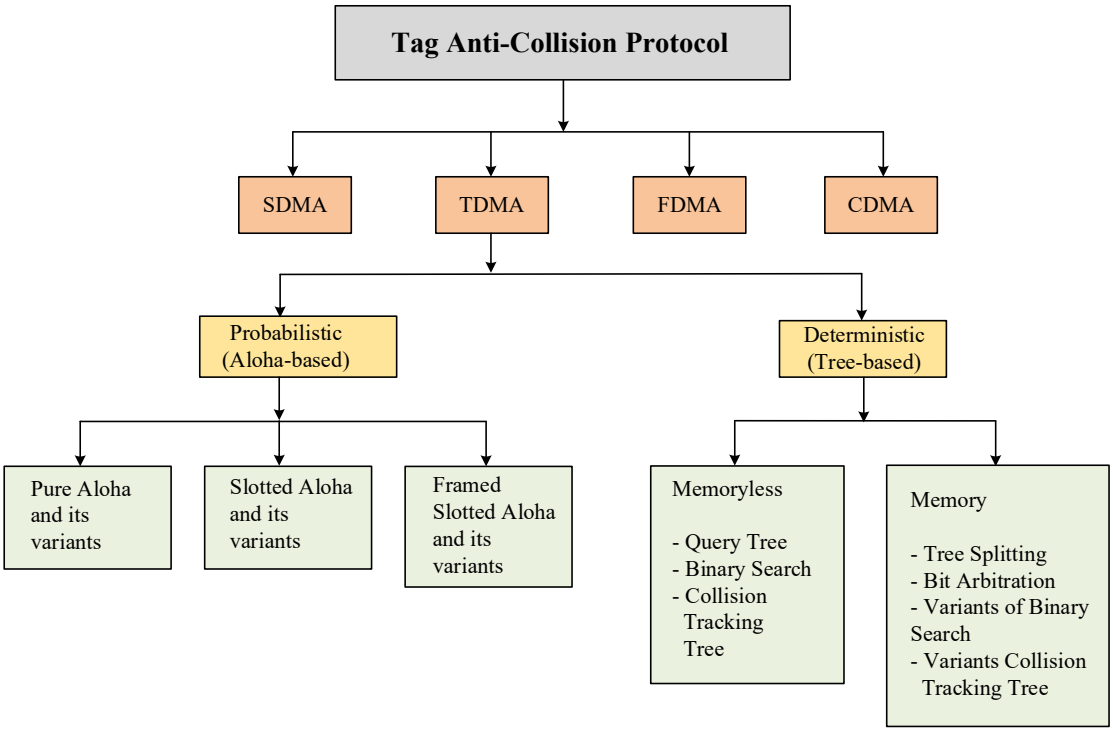

Fig. 2 *RFID anti-collision protocols*

3. Methodology

The overall aim of the research is to propose an improve anti-collision algorithm for passive RFID system. Implementation of the operational framework as shown in Fig. 3. An operational framework describes the sequence of steps to achieve the research objectives. This framework consists of three phases: Phase 1 – literature review, Phase 2 - development of the proposed algorithm and Phase 3 - evaluation of the proposed algorithm.

In Phase 1, Several memoryless methods have been extensively researched in the domain of RFID anti-collision algorithms, encompassing both classic Aloha-based methods and tree-based algorithms such as Query-tree, binary-search, and collision-tracking tree, along with their various iterations and enhancements. These algorithms have been meticulously studied and compared in terms of their collision resolution techniques, spanning from their foundational concepts to their modern variants aimed at improving performance and efficiency in RFID systems.

Aloha-based algorithms operate on a probabilistic basis, where tags respond to queries independently, making them simple yet prone to collisions and inefficiencies, especially in dense tag environments. On the other hand, tree-based algorithms introduce hierarchical or binary partitioning strategies to streamline the tag identification process and minimize collision occurrences. For instance, Query-tree algorithms utilize prefix-based querying,

Int. Journal of Integrated Engineering Vol. 16 No. 3 (2024) p. 257-272

while binary-search algorithms employ iterative partitioning to swiftly isolate responding tags. More advanced variants, like collision-tracking tree algorithms, focus on enhancing collision resolution by employing sophisticated techniques such as Manchester coding. Manchester coding plays a pivotal role in enhancing collision detection accuracy by discerning bit transitions, thereby reducing false detections and optimizing identification efficiency.

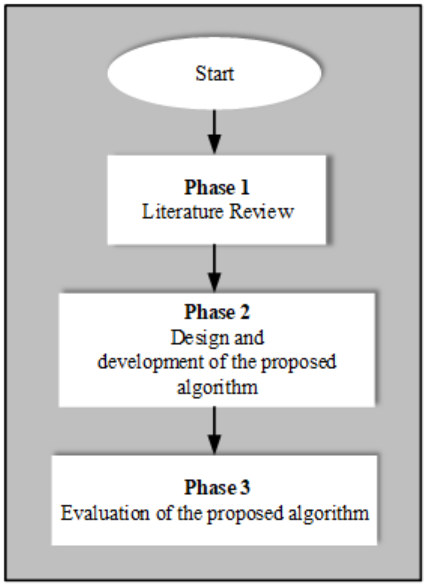

Fig. 3 *Operational framework*

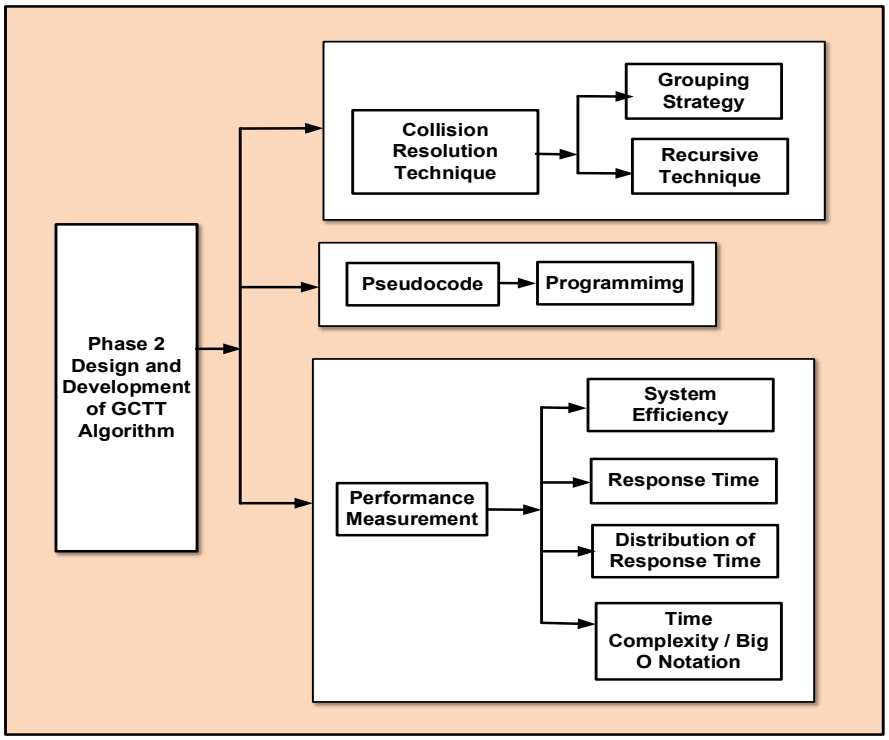

Fig. 4 *Operational framework of phase 2*

The scope of the study was then narrowed down to focus on grouping techniques and collision detection methods, particularly those utilizing Manchester coding due to its effectiveness in detecting bit collisions. The emphasizes is the implementation to tackle the multi-tag problem. The approach for the unique constraints of passive tags, which have limited circuitry and memory capacity. By leveraging the precise bit transition detection capabilities of Manchester coding, the study seeks to optimize the collision resolution process. This optimization aims to enhance the system's efficiency in identifying multiple tags simultaneously, ensuring reliable and rapid

tag identification even in dense RFID environments. The emphasis not only bridges the gap between theory and application but also addresses the real-world challenges posed by the minimalistic design of passive RFID tags, ultimately contributing to more effective and scalable RFID systems.

In phase 2, the study involves a comprehensive discussion of the methodology behind the Group Collision Tracking Tree (GCTT) algorithm as highlighted in Fig.4. The process begins by dividing RFID tags into two distinct groups based on the Most Significant Bit (MSB) of their identification (ID): one group includes IDs starting with 0X, while the other includes IDs starting with 1X. This initial segregation facilitates more efficient processing and is followed by the detection of collision bits using Manchester coding, which accurately identifies bit transitions and thereby enhances collision detection accuracy.

The detailed steps of this methodology are carefully translated into pseudocode to ensure a clear and structured implementation framework. This pseudocode is then written and simulated in Python, a language selected for its versatility and powerful computational capabilities. To further optimize the algorithm's performance, recursive function techniques are employed. These techniques streamline execution by breaking down complex tasks into simpler, repetitive ones, significantly accelerating processing time. This phase ensures that the GCTT algorithm not only effectively identifies and resolves collisions but also operates efficiently within the constraints of large-scale RFID systems, particularly those utilizing passive tags.

In phase 3, the testing and evaluation of the GCTT algorithm involved comparing it with the BCT and CT algorithms, which were selected as benchmarks due to their use of Manchester coding for collision tracking. The simulations focused on system performance, specifically measuring response time, which indicates the time required to read all tags, and Big O notation, which was used to assess the algorithm's complexity.

3.1 GCTT Algorithm

The operation of the GCTT algorithm can be explained by understanding the tagging method in the RFID system. Each tag is given a unique ID whose bit length for today's applications is 96 bits EPC (Electronic Product Code). To understand the tagging method, it can be started with a small number of bits. Table 1 shows the truth table for a 3-bit ID. The selection of a small number of bits is to make it easier to understand the overall operation of the GCTT algorithm. The relationship between the number of bits and the number of tags can be related by the equation, $n = 2^N$, where *N* is the number of bits and *n* is the number of tags. If using 3-bit, it gives 8 unique IDs for a total of 8 tags. Each tag is assigned a unique ID as in Table 1 so that each tag has a different ID.

Table 1 *3-bit ID*

ID	Tag Name
000	Tag A
001	Tag B
010	Tag C
011	Tag D
100	Tag E
101	Tag F
110	Tag G
111	Tag H

Fig. 5 shows the interrogation between the reader and the tag. Tags labeled A, B, C and D have MSB 0 while Tags labeled D, E, F and G have MSB 1. GCTT algorithm will sort group by group by starting with the group of tags that have MSB 0 then MSB 1 until all tags are read.

Scenario 1 is the situation when only one tag is in the interrogation zone as in Fig. 6. The tag has MSB 0. First the reader queries with 0. The tag replies to the reader. The reader scans the returned ID via Manchester coding, since there is only one tag in the slot, no tag collision occurs and no bit collision (X). The reader also checks the total number of bits to verify it is 3 bits. The reader sends a SELECT command on the tag. After reading the ID, the reader sends an UNSELECT command and puts the tag in 'silence' status. After that the reader continues the query with 1. No tag reply. The reading process has been completed. The time complexity (*Tc*) is defined by the number of time slots/round allocated to read the tag. For scenario 1, the time complexity is given by 2n where *n* is the number of tags. The total number of slots for reading tags is equal to 2.

Int. Journal of Integrated Engineering Vol. 16 No. 3 (2024) p. 257-272

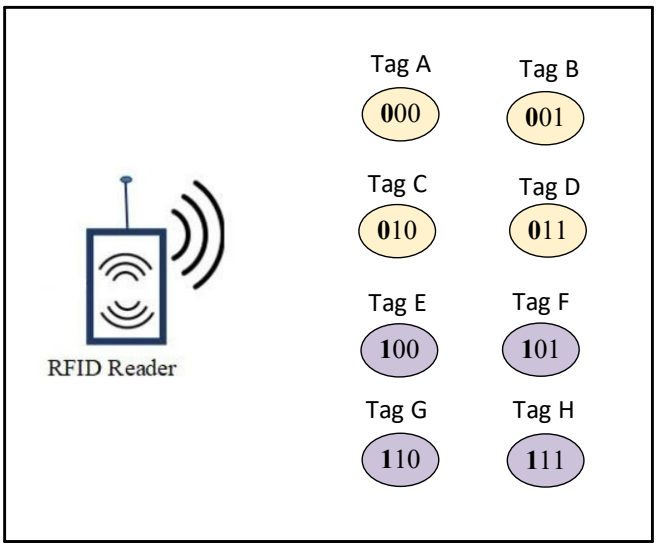

Fig. 5 *Interrogation between the reader and the tags*

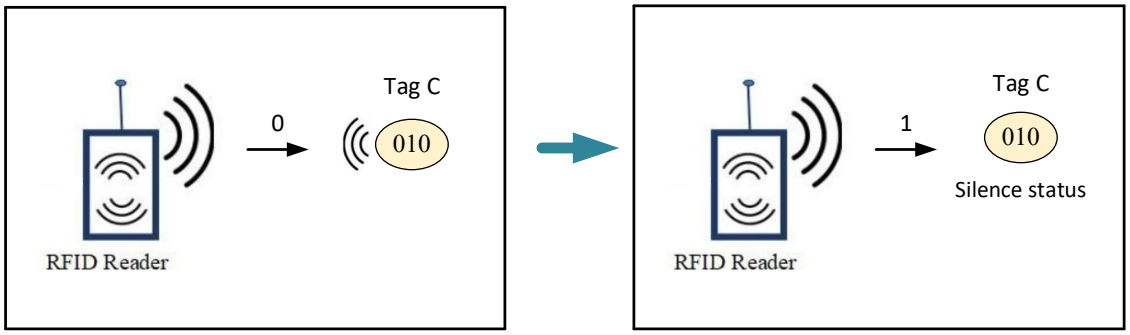

Fig. 6 *Query and reading of a tag of MSB 0*

Scenario 2 is the situation when only one tag with MSB 1 is in the interrogation zone as in Fig. 7. First the reader queries with 0. There is no tag reply. After that the reader continues the query with 1. The tag replies to the reader. The reader scans the returned ID via Manchester coding, since there is only one tag in the slot, no tag collisions occur and no bit collisions (X). The reader also checks the total number of bits to verify it is 3 bits. The reader sends a SELECT command on the tag. After reading the ID, the reader sends an UNSELECT command and puts the tag in 'silence' status. The reading process has been completed. For scenario 1, the time complexity is given by 2n where *n* is the number of tags. The total number of slots for reading tag is equal to 2.

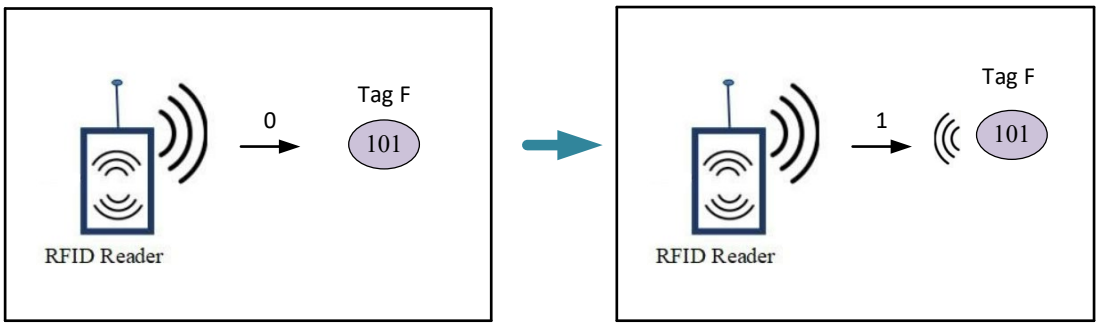

Fig. 7 *Query and reading of a tag of MSB 1*

Scenario 3 is a situation when several tags with MSB 0 are in the interrogation zone, Table 2 shows how GCTT sorts the tags. The time complexity is given by 2n where *n* is the number of tags. In this case, n = 4, the number of slots for reading tags is equal to 8.

Table 2 *Steps to sort 4 tags of MSB 0*

find_tags (Reader Query)	Collision Tracking (X = Collision bit)	Is length of ID equal 3 and no 'X'?	Status	Tag Respond	Slot
0	0X	No	Collision	Tag A (000) Tag B (001) Tag C (010) Tag D (011)	1
00	00X	No	Collision	Tag A (000) Tag B (001)	2
000	000	Yes	**Read**	**Tag A (000)**	3
001	001	Yes	**Read**	**Tag B (001)**	4
01	01X	No	Collision	Tag C (010) Tag D (011)	5
010	010	Yes	**Read**	**Tag C (010)**	6
011	011	Yes	**Read**	**Tag D (011)**	7
1	-	-	-	None	8

Scenario 4 is a situation when several tags with MSB 1 are in the interrogation zone, Table 3 shows how GCTT sorts the tags. The time complexity is given by 2n where *n* is the number of tags. In this case, n = 4, the number of slots for reading tags is equal to 8.

Table 3 *Steps to sort 4 tags of MSB 1*

find_tags (Reader Query)	Collision Tracking (X = Collision bit)	Is length of ID equal 3 and no 'X'?	Status	Tag Respond	Slot
0	-	-	-	None	1
1	1X	No	Collision	Tag E (100) Tag F (101) Tag G (110) Tag H (111)	2
10	10X	No	Collision	Tag E (100) Tag F (101)	3
100	100	Yes	**Read**	**Tag E (100)**	4
101	101	Yes	**Read**	**Tag F (101)**	5
11	11X	No	Collision	Tag G (110) Tag H (111)	6
110	110	Yes	**Read**	**Tag G (110)**	7
111	111	Yes	**Read**	**Tag H (111)**	8

Scenario 5 is a situation when several tags with MSB 0 and MSB 1 are in the interrogation zone, Table 4 shows how GCTT sorts the tags. The time complexity is given by 2(n-1) where *n* is the number of tags. In this case, n = 8, the number of slots for reading tags is equal to 14.

Int. Journal of Integrated Engineering Vol. 16 No. 3 (2024) p. 257-272

Table 4 *Steps to sort 8 tags of MSB 0 and MSB 1*

find_tags (Reader Query)	Collision Tracking (X = Collision bit)	Is length of ID equal 3 and no 'X'?	Status	Tag Respond	Slot
0	0X	No	Collision	Tag A (000) Tag B (001) Tag C (010) Tag D (011)	1
00	00X	No	Collision	Tag A (000) Tag B (001)	2
000	000	Yes	**Read**	**Tag A (000)**	3
001	001	Yes	**Read**	**Tag B (001)**	4
01	01X	No	Collision	Tag C (010) Tag D (011)	5
010	010	Yes	**Read**	**Tag C (010)**	6
011	011	Yes	**Read**	**Tag D (011)**	7
1	1X	No	Collision	Tag E (100) Tag F (101) Tag G (110) Tag H (111)	8
10	10X	No	Collision	Tag E (100) Tag F (101)	9
100	100	Yes	**Read**	**Tag E (100)**	10
101	101	Yes	**Read**	**Tag F (101)**	11
11	11X	No	Collision	Tag G (110) Tag H (111)	12
110	110	Yes	**Read**	**Tag G (110)**	13
111	111	Yes	**Read**	**Tag H (111)**	14

Table 5 is a summary of GCTT algorithm scenarios in solving tag collision problems in passive RFID systems. By referring to scenario 5 which is the situation where all tags between MSB 0 and MSB 1 are in the interrogation zone, the CT algorithm has a time complexity of 2n-1 [22] while GCTT has a time complexity of 2(n-1). It shows the improvement of GCTT by reducing the slot when reading all the tags of all possible ID distribution. This shows that GCTT has less time complexity and is suitable for applications in multi-tag and large-scale systems.

Table 5 *Summary of the GCTT algorithm scenario*

Scenario	Setting	Number of Tag (n)	Number of Bit / length (l)	Number of Slot (s)	Time Complexity (T_c)
Scenario 1	MSB/ Header 0	1	3	2	$T_c = 2n$
Scenario 2	MSB/ Header 1	1	3	2	$T_c = 2n$
Scenario 3	MSB/ Header 0	4	3	8	$T_c = 2n$
Scenario 4	MSB/ Header 1	4	3	8	$T_c = 2n$
Scenario 5	Mixed MSB/Header 0 and 1	8	3	14	$T_c = 2(n-1)$

The resolution steps are described as in pseudocode in Fig. 8, and then written and simulated in Python. During the simulation, the number of tags is increased up to 1000 and the number of bits is increased up to 128 bits. This situation is to test the performance of GCTT by increasing the bit and increasing the number of tags. In terms of

programming techniques, it uses recursive function techniques to minimize the use of space and for more efficient execution time with the increased of input size. The combination of this method with MSB grouping and collision tracking results in faster resolution in resolving tag collisions.

Simulation Environment, Python.

Computer specification:
CPU: AMD Ryzen 5 3600
Motherboard: B550M DS3H
RAM: 48 GB 3200 MHz
Graphic Card: RTX 3060 12 GB VRAM

```
function find_tags(header):

        // matching_tags is the response of multiple (collision) or single tag
        matching_tags = read_tags(header)

        if length of matching_tags equal to number of bit and no 'x' in tag then
                return matching_tags
        else if length of matching_tags equal to 0 then
                if header starts with 0 then
                        return find_tags("1")
                else
                        return 0
                end if
        else
                //initialize tags with empty array
                tags = []
                // example: 0x -> 00
                set end of matching_tags to 0
                append tags with find_tags(matching_tags)
                // example: 0x -> 01
                set end of matching_tags to 1
                append tags with find_tags(matching_tags)

                if length of header == 1 and header starts with 0 then
                        append tags with find_tags("1")
                end if

                return tags
        end if
end function
```

Fig. 8 *GCTT Pseudocode*

4. Experiment

In phase 3, as shown in Fig. 9, the GCTT algorithm was evaluated by comparing it with the BCT and CT algorithms, which were selected as benchmarks due to their use of collision tracking methods with Manchester coding and the memoryless nature of CT. The evaluation focused on response time, measuring the duration required to read all tags, and Big O notation, assessing the complexity of the algorithms. The performance assessment was conducted in three stages: response time analysis, distribution of response time, and Big O notation evaluation.

Experiment 1 and Experiment 2 are to get the response time of the GCTT algorithm by changing the length of the ID and the number of tags. Experiment 2 is to test the size or length of the standard ID used today with a 96-bit ID and a total of 1000 tags. For future use, the ID will be extended to 128 bits. Experiment 1 and 2 are to find the response time which is the time taken to read all the tags. The algorithm is to sort the tags according to the MSB header. The way Reader queries tag by sending the query which is indirectly divides the large group of tags into two groups, one group with MSB/header 0 and another group with MSB/header 1.

Experiment 3, Experiment 4 and Experiment 5 are to get the response time distribution of the GCTT algorithm by changing the number of tags and MSB header. Most of the headers show some meaning such as the company name, for example tags attached to products of the same manufacturer could have the same header. This research

Int. Journal of Integrated Engineering Vol. 16 No. 3 (2024) p. 257-272

also tests the situation where a group of tags are tagged on products from the same brand and tests on products in supermarkets, for example having various products and manufacturers.

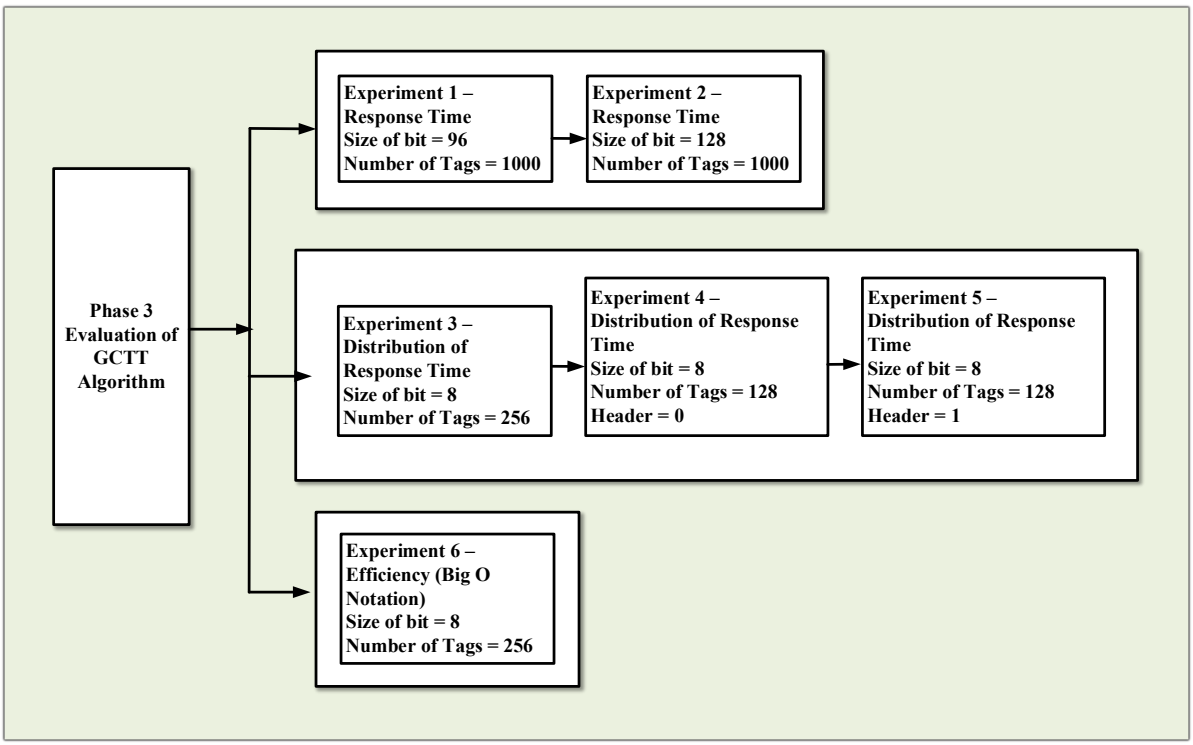

Fig. 9 *Evaluation of GCTT algorithm*

Experiment 3 is to find the distribution of response time which is distribution of the time taken to read all the tags. The size of ID tag is 8 bits and the total number of tags are 256. For an 8-bit ID, the maximum number of tags are 256 tags. Algorithm is tested to sort tags according to the MSB header. The way Reader queries tag by sending query logic 0 first is that it indirectly divides a large group of tags into two groups, one group with header 0 and another group with header 1. Each group has a total of 128 tags. Group partitioning is to reduce the branches of the binary tree and speed up the reader to sort tags and read IDs.

Experiment 4 is to find the distribution of response time which is the distribution of time taken to read all the tags. The tag ID size is 8 bits and the total number of tags is 128. The algorithm is tested to sort the tags according to the MSB 0 header. Because the tags only have the MSB 0 header, only one group is accessed by the reader.

Experiment 5 is to find the distribution of response time which is the distribution of time taken to read all the tags. The tag ID size is 8 bits and the total number of tags is 128. The algorithm is tested to sort the tags according to the MSB 1 header. Because the tags only have the MSB 1 header, only one group is accessed by the reader.

Experiment 6 is Big O notation which expressed a mathematical concept used in computer science to describe the efficiency of algorithms, specifically in terms of their time or space complexity. It provides an upper bound on the growth rate of an algorithm's runtime or memory usage as the input size increases. This helps in understanding the worst-case scenario for how an algorithm performs as the size of the input data grows.

5. Result

Fig. 10 shows response time versus number of tags. In this Experiment, the ID size is set to 96-bit and the number of tags is increased up to 1000. A comparison is made between BCT, CT and GCTT Algorithms. The simulation results show that the response time of GCTT is better than BCT and CT. The grouping method in GCTT speeds up the time to sort collisions and read all tags because this method reduces tree branches. In addition, the programming method uses a recursive function that also contributes to achieving a fast response time in the process of reading all the tags. GCTT is the best in the context of scalability where it is able to achieve fast times even with a high number of tags.

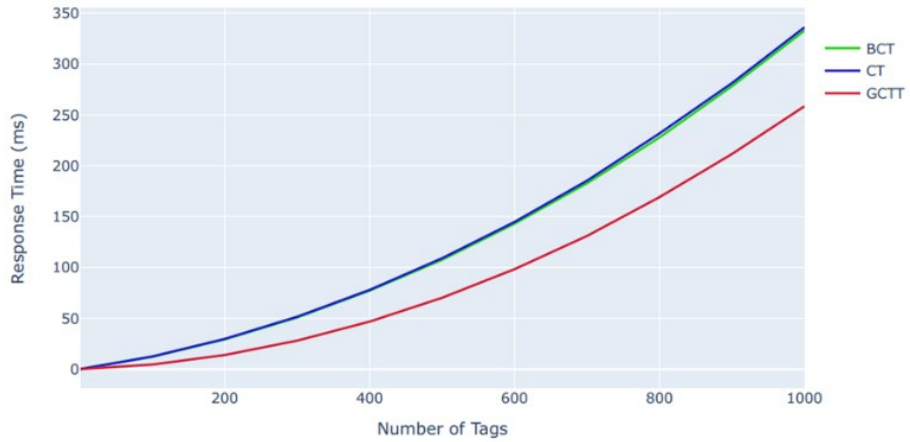

Fig. 10 *Response time of BCT, CT, GCTT versus number of tags (1000 tags, 96 bits)*

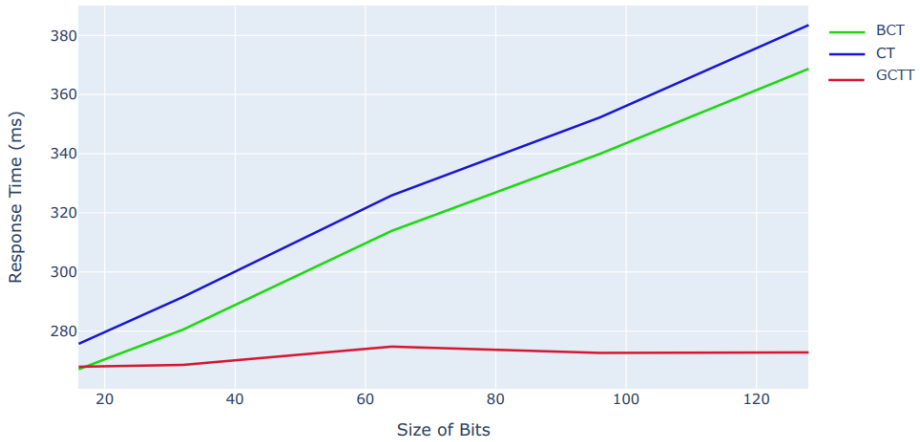

Fig. 11 *Response time of BCT, CT, GCTT versus number of tags (1000 tags, 128 bits)*

Fig. 11 shows response time versus number of bits. In this Experiment the number of tags set to 1000 and the ID size is increase to 128-bit. A comparison is made between BCT, CT and GCTT Algorithms. The simulation results show that the response time of GCTT is better than BCT and CT. The grouping method in GCTT speeds up the time to sort collisions and read all tags because this method reduces the branch of the tree. The number of bits is also a factor that affects the time taken to read all the tags. GCTT shows the achievement of very fast response times as the ID size increases. GCTT can adapt to the increase in ID size in the future.

Fig. 12 *Distribution response time of BCT, CT and GCTT*

Fig. 12 shows response time distribution of GCTT is better than BCT and CT. The graph for GCTT is on the far left showing the fastest response time compared to BCT and CT.

Int. Journal of Integrated Engineering Vol. 16 No. 3 (2024) p. 257-272

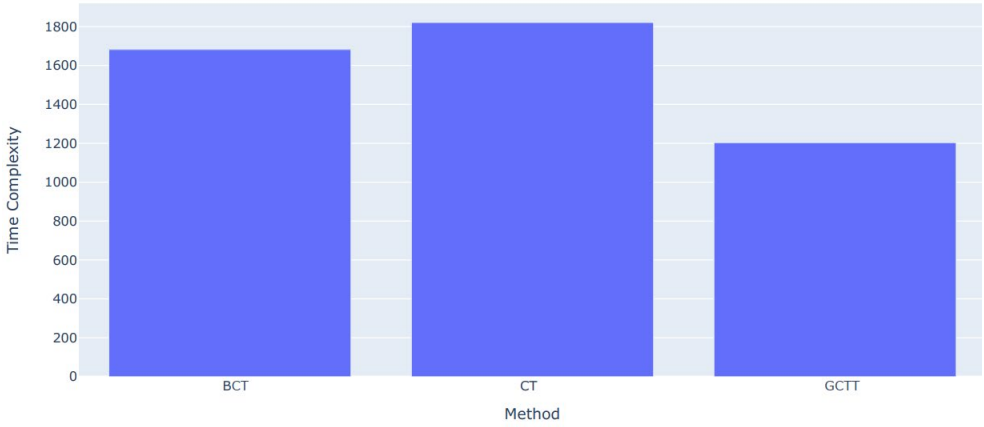

Fig. 13 *Big O notation of BCT, CT and GCTT (256 samples/tags, 8 bit)*

Big O Notation is a metric for determining the efficiency of an algorithm. It allows to estimate how long the program will run on different sets of inputs and measure how effectively it scales as the size of the input increases. Big O, also known as Big O notation, represents an algorithm's worst-case complexity. It uses algebraic terms to describe the complexity of an algorithm. Big O defines the runtime required to execute an algorithm by identifying how the performance of the algorithm will change as the input size grows. It measures the efficiency and performance of algorithm using time and space complexity [23]. Fig. 13 shows that GCTT having less value of Big O compare to BCT and CT. The design of GCTT takes into account the complexity of the algorithm. The programming uses a recursive function technique that reduces time and space complexity which contributes to achieving a fast time in the process of reading all the tags.

5.1 Discussion

System efficiency (SE) is a commonly used metric for evaluating anticollision protocol performance, measuring how efficiently the protocol uses slots. It is defined as the ratio between the minimum number of slots/rounds needed to read all tags which are equivalent to the number of tags within the reader's range and the actual number of rounds taken by the protocol. This total includes identification rounds (R_{ident}), collision rounds (R_{coll}), and idle rounds (R_{idle}) (24].

$$SE = \frac{R_{ident}}{R_{tot}} = \frac{R_{ident}}{R_{idle} + R_{ident} + R_{coll}} \quad [1]$$

System efficiency (SE) does not account for the varying durations of different rounds, which led to the introduction of Time System Efficiency (Time SE)(24). Time SE considers the different lengths of each round, recognizing that idle rounds, which do not transmit tag IDs, are shorter and thus have a smaller impact on protocol performance compared to identification and collision rounds. To accurately assess Time_SE, idle rounds are normalized to the length of identification and collision rounds, both of which have the same duration using a multiplicative factor that represents their relative weight. Time System Efficiency (Time_SE) is defined by

$$Time\_SE = \frac{R_{ident}}{\beta R_{idle} + R_{ident} + R_{coll}} = \frac{R_{ident}}{R_{tot} + (\beta - 1)R_{idle}} \quad [2]$$

Time System Efficiency (Time_SE) is a measure of the percentage of time successfully spent in identifying tags. Specifically, it is the ratio between the time taken to read all tags, if there were no collisions, and the total actual time taken to read the tags. Time_SE gives a clear idea of how much time is devoted to tag identification and how much is wasted in collisions or in idle rounds. A protocol that has Time_SE = 0.6, spends the 60 percent of time in identifying tags, while 40 percent of time is wasted in collision and idle rounds. The Time_SE is complemented with latency, which is a measure of the total time taken to resolve queries, and reflects the performance experienced by the user (24).

Collision Tracking (CT) enhanced system performance by removing idle round (R_{idle}), leading to system efficiency of 50%. The analysis and formulation for system efficiency of CT and its varints just rely on parameters R_{iden} and R_{coll}.

BCT, the variant of CT improves system efficiency by reducing R_{coll} but this improvement introduces additional complexity to the circuitry. As a result, their classification could change, as they are no longer considered

memoryless algorithms. This trade-off involves balancing the performance gains with the increased complexity associated with implementing and managing these enhancements.

Table 6 *Time complexity and system efficiency [18][20][24]*

Anti-collision algorithm	Time complexity	System efficiency
CT	$2n-1$	$1/(2-1/n)$
BCT	n-1	n/ n -1

$$SE\ of\ CT = \frac{R_{ident}}{R_{tot}} = \frac{R_{ident}}{R_{ident} + R_{coll}} = \frac{n}{2n-1} = \frac{1}{\left(2-\frac{1}{n}\right)} \quad [3]$$

$$SE\ of\ BCT = \frac{R_{ident}}{R_{tot}} = \frac{R_{ident}}{R_{ident} + R_{coll}} = \frac{n}{n-1} \quad [4]$$

Table 6 shows a formulation and analysis focused solely on the number of rounds required to read tags. *n* is the number of tags. This approach simplifies the evaluation by considering only the round count, without accounting for the variations in the time each round might entail. Although the Collision Tree (CT) algorithm and its variants have effectively eliminated idle time, known as R_{idle}, this analysis does not fully address temporal aspects. The duration of each round can still vary significantly based on the tag-sorting method and the overall complexity of the tag-reading process. Therefore, a more detailed temporal analysis is needed to fully assess the performance implications.

The study addresses a significant limitation in previous research [18][20][24], which primarily focused on the number of rounds needed to read tags without a thorough analysis of temporality. To address this issue, the research employs simulation techniques to evaluate the response time, or the time taken by the algorithm to resolve tag collisions.

Table 7 presents a comparative analysis of the CT, BCT, and GCTT algorithms based on simulation results, highlighting the superior performance of GCTT in terms of response time and scalability. Specifically, GCTT achieves the fastest response time of 259 ms when processing 1,000 tags with a 96-bit ID, outperforming CT and BCT, which recorded response times of 336 ms and 333 ms, respectively. This performance underscores GCTT's effectiveness in large-scale RFID implementations. Furthermore, when dealing with 128-bit IDs and 1000 tags, GCTT maintained a response time of 273 ms, compared to 383 ms for CT and 367 ms for BCT, demonstrating GCTT's ability to adapt to scalability challenges effectively.

Additionally, the Big O notation analysis further validates GCTT's efficiency, with a result of 1.2k, significantly lower than the 1.8k and 1.7k observed for CT and BCT, respectively. This indicates that GCTT not only excels in response time but also offers low time and space complexity, making it a highly scalable solution for managing extensive RFID tag populations in large-scale applications. The combination of rapid response time, adaptability to increased tag and bit volumes, and efficient resource utilization positions GCTT as a leading algorithm for optimizing RFID system performance.

Table 7 *Comparison between CT, BCT and GCTT*

Collision Tracking Tree based Algorithm	**Collision Tracking (CT)**	**Bi-response Collision Tree (BCT)**	**Group-Collision Tracking Tree (GCTT)**
Response Time 96 Bits, 1000 Tags	336 ms	333 ms	259 ms
Response Time 128 Bits, 1000 Tags	383 ms	367 ms	273 ms
Algorithm Complexity (Big O Notation)	1.8k	1.7k	1.2k

6. Conclusion

In conclusion, the Group-Collision Tracking Tree (GCTT) algorithm represents a significant advancement in tag sorting technology by integrating grouping based on the Most Significant Bit (MSB), collision tracking, and recursive techniques. These advancements collectively enhance the speed and efficiency of reading all tags while maintaining the principle of a memoryless algorithm with minimal tag circuitry. The GCTT achieves this by effectively managing tag collisions and organizing tags into manageable subsets, which reduces the need for complex circuitry and extensive memory in the tags themselves. The simulation results provide a thorough evaluation of RFID system performance, highlighting both the time needed to read all tags and the algorithm's execution time. GCTT delivers outstanding performance with rapid response times and efficient collision management, while its low time complexity, confirmed by Big O notation, ensures quick tag sorting and minimal memory usage. Its recursive function adeptly handles large volumes of tags with minimal memory overhead, adhering to the memoryless design approach and making GCTT particularly well-suited for large-scale RFID systems. Overall, GCTT's efficient memory management and rapid execution underscore its effectiveness for high-performance RFID applications, offering a reliable and scalable solution for managing extensive tag populations while adhering to minimal circuitry requirements.

Acknowledgement

The work in this paper is supported by Universiti Teknologi Malaysia.

Conflict of Interest

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

The authors confirm contribution to the paper as follows: ***study conception and design:*** *Marizan Yaacob, Noraimi Shafie;* ***data collection:*** *Marizan Yaacob;* ***analysis and interpretation of results:*** *Marizan Yaacob, Noraimi Shafie, Norliza Mohamed, Azizul Azizan;* ***draft manuscript preparation:*** *Marizan Yaacob, Noraimi Shafie, Norliza Mohamed, Azizul Azizan.*

References

[1] C. Law, K. Lee, and K. Y. Siu (2000) Efficient memoryless protocol for tag identification, *Workshop Discrete Algorithms Methods Mobile Computer Communication*, 75–84.
[2] F. Zhou, C. Chen, D. Jin, C. Huang, and H. Min. (2004) Evaluating and optimizing power consumption of anti-collision protocols for applications in RFID systems, *In Proc. Int. Symp. Low Power Electron. Design, 357–362.*
[3] Xiaohao, S., & Baolong, L. (2017) An investigation on tree-based tags anti-collision algorithms in RFID, *International conference on computer network, electronic and automation (ICCNEA)*.
[4] Finkenzeller, K. (2010) *RFID handbook: Fundamentals and applications in contactless smart cards,* radio frequency identification and near-field communication.
[5] Wang Yong, Liu Qing, Wang Lei, Shen Hao (2017) Research on Anti-collision Algorithm in Radio Frequency Identification Technology, *International Conference on Intelligent Human-Machine Systems and Cybernetics*
[6] Cmiljanic N., Landaluce H, Perallos A. (2018) A Comparison of RFID Anti-Collision Protocols for Tag Identification, *MDPI*.
[7] L. Wang, Z. Luo, R. Guo and Y. Li (2023) A Review of Tags Anti-Collision Identification Methods Used in RFID Technology, *MDPI*.
[8] H. Qian (2021) Research on RFID Anticollision Algorithms in Industrial Internet of Things, *Wireless Communications and Mobile Computing*
[9] P. Kamalendu, (2021) A Novel Frame-Slotted ALOHA Algorithm for Radio Frequency Identification System in Supply Chain Management, *Science Direct, Procedia Computer Science*.
[10] Jian. S, Alex X. L and Zhengguo S. (2020) A Partitioning Approach to RFID Identification, *IEEE Transactions on Networking.*
[11] Tripathi S. and Jain V. K. (2019) Performance analysis of adaptive tree-based anti-collision protocol using M-ary splitting in RFID, *10th International Conference on Computing, Communication and Networking Technologies (ICCCNT).*
[12] Yu F., Xue W., Zhihong Q. (2017) A bit arbitration tree anti-collision protocol in radio frequency identification systems, *International Journal of Distributed Sensor Networks.*
[13] Yaacob M., Daud S. M. and Azizan A. (2019) A Review of Deterministic Anti-Collision Algorithm of Passive RFID Systems, *Open International Journal of Informatics (OIJI)*.

[14] Youjing B., Guoxing Z., Lvqing Y. and Yezi X. (2017) An Improved Binary Search RFID Anti-collision Algorithm, *12th International Conference on Computer Science & Education*.
[15] Meryle Mvoulabolo1, Tebello N.D. Mathaba, Marcel O. Odhiambo (2019) Performance analysis of binary search based RFID anti-collision algorithms, *International Multidisciplinary Information Technology and Engineering Conference (IMITEC).*
[16] Guozhong D, Weizhe Z., Sichang X., Feng Q., and Haowen T. (2020) An Improved Binary Search Anti-Collision Protocol for RFID Tag Identification, *Computers, Materials & Continua*, 1855-1868.
[17] Yaacob M. and Daud S. M. (2012). Novel Binary Search Algorithm for Fast Tags detection in Robust and Secure RFID systems. *Communications in Computer and Information Science*, 575-582.
[18] Xiaolin J, Quan Y. F. and Chengzhen M. (2010) An efficient Anti-Collision Protocol for RFID Tag Identification, *IEEE Communication Letters*.
[19] Xiaolin J. and Quan Y. F. (2015) An improved anti-collision protocol for radio frequency identification tag, *International Journal of Communication System.*
[20] Xiaolin J., Yuhao F. and Yajun G. (2022) A Simple and Fast Anti-collision Protocol for Large-scale RFID Tags Identification, *KSII Transaction on Internet and Information Systems*.
[21] Thomas F. La Porta, Maselli G. and Petrioli C. (2010) Anti-collision Protocols for Single-Reader RFID Systems: Temporal Analysis and Optimization, *IEEE Transactions on Mobile Computing*.
[22] Xiaohao, S. and Baolong, L. (2017) An Investigation on Tree-Based Tags Anti-Collision Algorithms in RFID, *International Journal of Advanced Network Monitoring and Controls.*
[23] Devi, S.G., Selvam, K. and Rajagopalan, S.P. (2011) An Abstract to Calculate Big O Factors of Time and Space Complexity of Machine Code, *International Conference on Sustainable Energy and Intelligent Systems.*
[24] Xiaolin J., Quan Y. F. and Lishan Y (2012) Stability Analysis of an Efficient Anti-Collision Protocol for RFID Tag Identification, *IEEE Transactions on Communication.*