Received date month year; | Revised date month year; | Accepted date month year

DOI: xxx/xxxx

ARTICLE TYPE

A Fast Tag Identification Anti-Collision Algorithm for RFID Systems

S. K. Wijayasekara[1] | S. Nakpeerayuth[1] | R. Annur[2] | H.-Y. Hsieh[3] | T. Sanguankotchakorn[4] | K. Sandrasegaran[5] | W. Srichavengsup[6] | T. Phromsa-ard[1] | L. Wuttisittikulkij*[1]

[1]Smart Wireless Communication Ecosystem Research Group, Department of Electrical Engineering, Chulalongkorn University, Bangkok, Thailand

[2]Department of Computer and Communication Technology , Universiti Tunku Abdul Rahman, Perak Campus, Malaysia

[3]Department of Electrical Engineering and Graduate Institute of Communication Engineering, National Taiwan University, Taipei, Taiwan

[4]School of Engineering and Technology , Asian Institute of Technology, Bangkok, Thailand

[5]Electrical and Data Engineering, University of Technology Sydney, Sydney, Australia

[6]Computer Engineering, Robotics and Technology Research Laboratory, Thai Nichi Institute of Technology, Bangkok, Thailand

Correspondence
*L. Wuttisittikulkij Email: Lunchakorn.W@chula.ac.th

Present Address
Department of Electrical Engineering, Chulalongkorn University, Bangkok, Thailand.

Summary

In this work, we propose a highly efficient binary tree-based anti-collision algorithm for RFID (Radio Frequency IDentification) tag identification. The propose Binary Splitting Modified Dynamic Tree (BS-MDT) algorithm employs a binary splitting tree to achieve accurate tag estimation and a modified dynamic tree algorithm for rapid tag identification. We mathematically evaluate the performance of the BS-MDT algorithm in terms of the system efficiency and the time system efficiency based on the ISO/IEC 18000-6 Type B standard. The derived mathematical model is validated using computer simulations. Numerical results show that the proposed BS-MDT algorithm can provide the system efficiency of 46% and time system efficiency of 74%, outperforming all other well-performed algorithms.

KEYWORDS:
RFID, tree algorithm, anti-collision, tag, reader, ALOHA

1 | INTRODUCTION

Radio Frequency Identification (RFID) is a promising technology for people, animal, asset and document tracking [1,2]. An RFID system consists of a reader and multiple tags. A reader is an electronic device which wirelessly communicates with tags in its coverage range. A tag, which is a tiny microchip with an antenna, is attached to an item or a product that needs to be tracked. In RFID tag identification process, the reader first sends a query command to the tags in its interrogation zone. The corresponding tags then respond with a reply at a randomly selected timeslot. If the reader receives only one reply signal in a certain timeslot the tag is said to be successfully identified. If the reader receives several reply signals simultaneously in the same timeslot, a tag collision is said to happen and none of them can be identified. The tag collision increases the number of message signals to be

[0]**Abbreviations:** RFID, radio frequency identification; EPC, electronic product code

exchanged between the reader and tags thereby prolonging the tag identification process. If this problem is not handled properly, the tag identification time can be excessively long especially when a large number of tags are involved. As a result, an efficient anti-collision algorithm is essential for RFID systems to ensure a fast tag identification. [3,4,5,6,7,8,9].

In literature, anti-collision algorithms can be broadly classified into two categories namely ALOHA-based and tree-based. The concept of ALOHA was first introduced by Abramson in 1970 [10] as a random-access radio communications. Five years later, Roberts [11] suggested a simple yet elegant modification to pure ALOHA, where a synchronous data transmission happens within a specific time period called timeslot and if necessary, the re-transmission occurs after a random number of timeslots. This is known as slotted ALOHA. Many anti-collision algorithms proposed for RFID systems are based on a variant of slotted ALOHA called Frame Slotted ALOHA (FSA). In FSA several consecutive slots are grouped into a frame and in each frame, the tags can randomly choose one slot to transmit a reply to the reader. If tags suffer from a collision in the current frame, the re-transmissions of the respective collided tags are performed in the next frame. Frame slotted ALOHA was initially proposed with fixed frame sizes, but subsequent works revealed that delay performance can be much improved with dynamic frame sizes [12 13 14]. In the RFID EPCglobal Gen-2 [15] standard, the Q algorithm which is adopted as its anti-collision algorithm is essentially dynamic FSA with an early frame breaking policy. Recently, progress on improving the anti-collision protocols for EPC Gen2 standard has been impressive. They focus on the frame size adjustment to perform dynamic FSA. The proposed methods in [16 17 18 19,20] can achieve a system efficiency close to the theoretical limit of ALOHA-based algorithms of 36.8%. More recently, [20] introduces an early frame breaking policy in DFSA to identify a suitable frame size to currently presenting tags in the system. After identifying the best frame size, the collided tags are sub-grouped and identified using DFSA and this method achieves a system efficiency of as high as 41%.

The tree algorithm was proposed by [21] and independently by [22], two years after the invention of the slotted ALOHA. At time of invention, it was considered as a breakthrough of random access protocols for resolving collision. For RFID systems, the same principle of the tree algorithm has been applied to resolve tag collision problems. As opposed to the frame slotted ALOHA that relies on avoiding repeated collisions within a series of frames, the tree algorithm aims to resolve a collision in a strategic manner by recursively splitting the collided tags into subgroups until each subgroup contains at most one tag. The conventional tree algorithm applies the splitting of collided tags into two subgroups so called binary tree algorithm. For generalized tree algorithms, the tags are split into Q subgroups where Q is an integer number greater than or equal 2. These Q-ary tree algorithms have been comprehensively studied in [23]. An important performance metric for the tree-based anti-collision algorithms is the system efficiency which is defined as the ratio between the initial number of tags and the number of slots required to identify all the tags. For the binary and ternary tree algorithms, their system efficiency are proved to be 34.6% and 36.6% respectively. The efficiency of the basic binary tree algorithm can be enhanced to 37.5% by skipping definite colliding slots, which happens when no tag selects the left branch of the tree structure [24,25]. A further slight improvement can be achieved with biased splitting to the left branch and an efficiency of 38.1% can be obtained as suggested by the same reference. This algorithm will be hereafter referred as Modified Tree Algorithm (MTA). Tree Slotted ALOHA (TSA) is another interesting way to improve performance of tree-based algorithms [26]. Instead of using a fixed frame size, such as two or three throughout the entire collision resolution process, each colliding slot can be split into various different frame sizes, depending on the number of tags involved in the collision. Therefore, this class of algorithms requires accurate tag estimation. In [26], approximately 37% system efficiency is achievable using the Vogt's estimation technique [13]. Binary Splitting Tree Slotted ALOHA (BSTSA) which consists of a binary splitting technique to accomplish the initial tag estimation [27], as opposed to the Vogt's estimation. The system efficiency of 40% is achieved. Splitting Binary Tree Slotted ALOHA Algorithm (Splitting BTSA) [28] adopts the binary splitting technique for tag estimation, while subsequent collision resolution starts with a frame size equal to the estimated number of tags and followed by the binary tree algorithm. Splitting BTSA reaches the system efficiency of 42.5% which to our knowledge presents the highest achievable system efficiency to date. Recently, a novel tree-based Fast Splitting Algorithm based on Consecutive Slot Status detection (FSA-CSS) tag identification method has introduced in [29] with an efficiency of 41%. In order to control the consecutive collision slots and idle slots occurring in the identification process, a command based fast splitting and shrinking mechanisms are introduced.

In this paper, we propose an efficient splitting tree-based anti-collision algorithm, called Binary Splitting Modified Dynamic Tree algorithm (BS-MDT), which offers significantly higher system efficiency than existing tree-based anti-collision algorithms by at least 4% for a broad range of tags. The proposed algorithm first splits tags into two groups and recursively further splits only tags in the left branches until no more tags left to split. Then tags in the right branches are identified in turn using our proposed Modified Dynamic Tree (MDT) algorithm starting from the lowest branch upwards to the root. To explain how already highly efficient anti-collision algorithms, such as BSTSA and Splitting BTSA, can be further enhanced, we carried out the

following extensive investigations to determine their theoretical performance limits and identify their salient features. Based on our detailed examination on the TSA, in an ideal condition where number of tags can be known, i.e., perfect tag estimation, the maximum system efficiency of 43.4% for large number of tags can be attained, as opposed to approximately 37.0% with Vogt's tag estimation. There is a huge performance gap of 6.4% between the practically achievable algorithm and the theoretical limit, which reveals that an adaptive frame size policy as used in TSA throughout each level in the tree structure can be beneficial compared to fixed frame size policy as used in basic binary or ternary tree algorithms. Nevertheless, to approach such a promising performance limit, a much more accurate tag estimation technique than Vogt's algorithm is required. For BSTSA, where the binary splitting technique is applied in conjunction with the TSA algorithm, the system efficiency improves by 3% compared to the TSA counterpart. However, under perfect tag estimation condition, the BSTSA apparently offers the theoretical performance limit of 43.2%, slightly lower than that of TSA. Two key features in the BSTSA are identified as follows: in the first phase tags are recursively split into two subgroups at an exponential rate (only on the left branch of the binary tree structure) and in the second phase the TSA algorithm with appropriate frame size equal to the number tags in their corresponding left branches instead of a fixed initial frame size of 128 (as used in TSA) is applied to resolve tags on the right branches of the tree structure. For splitting BTSA, which operates in two phases the same as BSTSA and uses the adaptive frame size policy only at the first level for resolving tags on the right branches of the tree structure and the fixed frame size of two for the subsequent levels. The splitting BTSA achieves better system efficiency than TSA and BSTSA by 5.5% and 2.5% respectively. An interesting aspect of splitting BTSA that should be emphasized is that its theoretical limit is only 42.7%, i.e., lower than that of TSA and BSTSA. Based on the above discussion, we apply the following findings for designing a highly efficient anti-collision algorithm.

1. The adaptive frame size policy can be very effective for tree-based anti-collision algorithms, if the initial number of tags can be accurately estimated and the frame size are properly chosen accordingly.
2. The binary splitting technique is an elegant means to provide accurate tag estimates for each subgroup, after the recursive split of initial tags is completed.
3. When the adaptive frame size policy is properly applied at the first level of the tree structure, each of the resulting subgroups contains small number of tags, mostly between 2-5 tags.
4. A small group of 2-5 tags can be effectively resolved using our proposed MDT algorithm without any tag estimation being required.

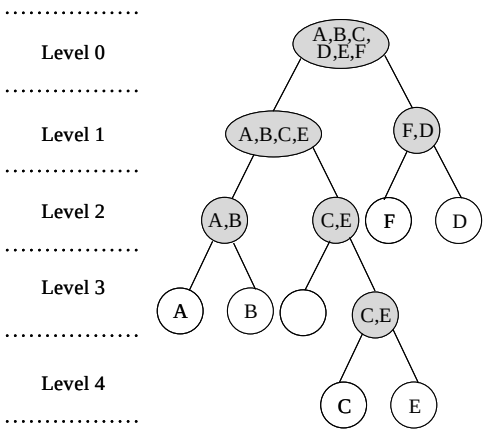
Therefore, our contributions in this paper can be summarized as follows:

1. We introduce an efficient anti-collision algorithm namely BS-MDT which outperforms the existing algorithms.
2. We analyze the proposed method analytically and derive a relationship between the number of tags and the frame size in order to achieve the highest system efficiency.
3. We carry out extensive simulations and theoretical analysis to evaluate the performance of BS-MDT algorithm with respect to system efficiency and time system efficiency for larger number of tags and the results are compared with well-known tree-based anti-collision algorithms namely, splitting BTSA, dynamic BTSA, BSTSA, TSA, MTA, BTA and TTA. Numerical results confirm that the proposed method can achieve higher system efficiency and time system efficiency following the ISO/IEC 18000-6 standard than all existing known tree-based algorithms and almost 4% higher system efficiency than that of splitting BTSA for the entire range of tags under investigation.

The rest of the paper is constructed as follows. In Section 2, the fundamental concept of tree algorithms is described with the execution examples of binary tree and modified tree algorithms. In Section 3, we present the proposed algorithm and its mathematical model. In Section 4, the performance comparison in terms of the system efficiency and the time system efficiency of the proposed method with seven other well-known algorithms is given. In Section 5 we conclude this paper.

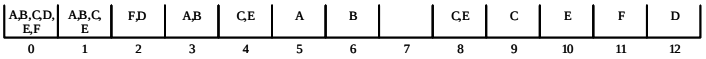
2 | FUNDAMENTAL CONCEPT OF TREE ALGORITHMS

In this section, we present the fundamental concept of conventional tree-based anti-collision algorithms and describe the execution procedure of the well-known Binary Tree Algorithm (BTA) in addition to one of its variants called Modified Tree Algorithm



(a) A possible tree structure of tree algorithm with $Q = 2$ *when six packets are initially collide.*

(b) Pre-order traversal (depth first search)



(c) Serial search

(d) Parallel search (breadth first search)

FIGURE 1 A possible collision resolution process of tree algorithm with $Q = 2$ when six packets collide in time slot axes.

(MTA). When there is a tag collision occurring at a particular timeslot, the tree algorithm uses a divide-and-conquer paradigm through a tree structure to systematically resolve the tag collision. The collision resolution process begins at the root node of the tree which corresponds to the initial collision timeslot. There are only three possible types of nodes, *i.e.*, idle, collision and success, in the tree structure. The collision node, which represents a timeslot with collided tags, will generate Q new nodes, where $Q \in \mathbb{Z}^+$ and $Q \geq 2$. That is tags involved in the collision are randomly split into Q subgroups. The idle and success nodes, which correspond to timeslots with no tag and with exactly one tag respectively, will become leaf nodes of the tree structure. Therefore, when there are no more new collision nodes generated the tree structure will cease to grow and this also means the end of collision resolution period. The number of nodes in the tree is essentially the number of timeslots required to resolve the collision.

2.1 | Binary Tree Algorithm (BTA)

In the Binary Tree Algorithm (BTA), each collided node is always split into two new nodes ($Q = 2$) in the next level. Thus the tree structure continues to grow until all the leaf nodes contain one or no tags. Figure 1(a) depicts a binary tree structure which represents an example of the collision resolution process of BTA with an initial collision of six tags. Firstly, the root node at Level 0 which is always a collided node generates two new nodes to Level 1 of the binary tree structure. The two nodes in Level 1 are collided nodes and hence each of these nodes generates two new nodes at Level 2. The two new nodes in Level 2 on the left-hand side are still collided nodes, so they split further into Level 3. On the other hand, the other two new nodes on the right

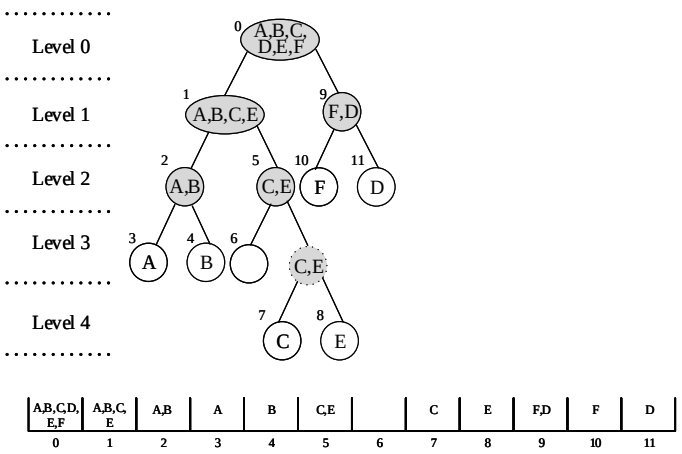


FIGURE 2 An execution example of MTA with initial six tags collision.

hand side become success nodes. This procedure is repeated until all the leaf nodes of the tree structure become idle or success nodes. In this example, there are 12 nodes (excluding the initial collision node) in the tree structure, so the number of timeslots needed to resolve six tags colliding is 12 timeslots (excluding the initial collision slot).

For a given tree structure, there are three different modes to transmit tags into time axis depending on the orders of tag transmissions, namely preorder traversal, serial and parallel searches. The preorder traversal begins at the initial collision node and explores nodes along a path all the way to a leaf before backtracking, and then along another path explores other nodes. This is equivalent to depth first search. The preorder traversal is illustrated in Figure 1(b) with respect to the binary tree structure given in Figure 1(a). As illustrated in Figure 1(c), the serial search explores a group of Q nodes together instead of one node at a time. The $Q = 2$ subgroups from the splitting process is always transmitted in the Q consecutive slots. The parallel search starts from the initial collision node and traverses the nodes in the same level before traversing the nodes in the levels further away. All subgroups in the same level will be transmitted in consecutive slots as depicted in Figure 1(d). Although in these three types the order of tree traversal is different from each other they are essentially the same with respect to the number of slots required in resolving the collision. In this example, they need the same number of slots to resolve the six tags collision *i.e.,* 12 slots.
It can be seen that Q subgroups of preorder traversal are not always transmitted in Q consecutive slots. On the other hand, the Q subgroups of serial and parallel trees are always transmitted in Q consecutive slots that exhibits a frame structure.

2.2 | Modified Tree Algorithm (MTA)

In BTA, the collided slots are further split into two subgroups and if the first subgroup is an idle, it is certain that the second subgroup is a collision. Therefore, a slot wastage can be reduced by just splitting the second subgroup into two subgroups by pretending that the collision has occurred. This mechanism has been implemented in[25,26] as Modified tree algorithm (MTA) with 37.5% system efficiency. In Figure 2 , an example of the MTA execution procedure is given. Note that in Figure 2 , the collided slot in Level 2 containing Tags C,E is followed by an idle and a definite collision slot at Level 3. This definite collision slot can be split into two subgroups without reading it. Therefore, to resolve this initial six tags collision, the MTA requires 11 slots while the BTA required 12 slots as given in Figure 1 . Further, in[25,26],it is given that a 38.1% system efficiency can be achieved in MTA with 0.418 of bias splitting probability in the left sub group.

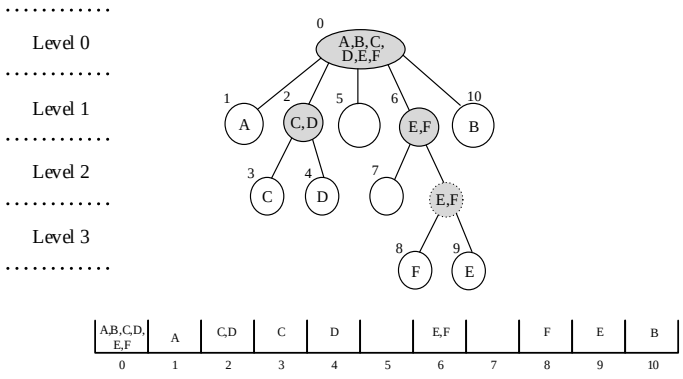


FIGURE 3 An execution example of MDT with initial six packets collision.

3 | THE PROPOSED ANTI-COLLISION ALGORITHM

This section will discuss the system description of the Binary Splitting Modified Dynamic Tree (BS-MDT) algorithm. Firstly, with the aid of an example the proposed Modified Dynamic Tree (MDT) the algorithm is explained. Next, the concept behind the proposed BS-MDT algorithm is defined using an example, a flow chart, and pseudocode which shows the communication between the RFID reader and tag. Finally, an analytical derivation of BS-MDT algorithm is given. The proposed BS-MDT algorithm contains the binary splitting tag estimation and dynamic frame length adjustment. It first performs tag estimation by using the binary splitting through a binary tree structure as proposed by[27] and subsequently adopted by[28] until the left branch contains zero or one tag. In this binary splitting approach, each node approximately contains half of the tags from its parent node at a given level of the tree. When the splitting step is finished, tags in each node on the right hand side of the tree structure are identified by the MDT algorithm with the initial frame size equal to β times the number of collided tags in the left-hand side, where β is the multiplicative factor which will be discussed in Section3.3. Thus, in the BS-MDT, the tags in the right branches of the binary tree structure are resolved from the lowest level to the top.

3.1 | Modified Dynamic Tree Algorithm (MDT)

In the Modified Dynamic Tree (MDT) algorithm, the initial users are grouped into smaller groups and the collided groups among them are resolved by applying the MTA algorithm. Figure 3 shows an execution processes of the MDT algorithm for six competing Tags (A,B,C,D,E,F) with an initial frame size of five slots. After splitting the six tags into five slots, the slot 2 and 6 are become collided slots. Slot 2 is followed by slot 3 and 4 where each slot contains a single tag. The collided slot 6 is followed by an idle slot 7 and a definite collision slots where the reader further split this definite collided slot into two slots without reading it. Following this example, it requires 10 slots to identify all the six tags.

3.2 | BS-MDT Algorithm

In this section, the system description of BS-MDT algorithm is discussed in detail with the aid of reader and tag pseudocodes and flow chart. Figure 4 presents the binary splitting procedure, followed by our proposed initial frame size assignment method for the MDT. As we can see, tags are split into two subgroups, *i.e.*, left and right, repeatedly in the depth first search manner, until reaching the slot or the k^{th} level which contains either zero or one tag. For an initial size of N tags, the value of k can be estimated as $log_2 N$, indicating that this process can finish very rapidly. If at the k^{th} level, the slot contains no tag, *i.e.*, $NLeft_k = 0$, it is certain that the number of tags on the right branch of the k^{th} level must contain at least 2 tags, *i.e.*, $NRight_k \geq 2$. Therefore, the initial frame size (L_k) should be set to at least two slots and in the proposed method, L_k is set to 2 for maximizing the system efficiency. On the other hand, if $NLeft_k = 1$, L_k is set to 1 for the same reason. Once all tags in the right branch of the k^{th} level are resolved by the MDT method described below, the actual value of $NRight_k$ can be known, and thus the actual number of colliding tags at the $(k-1)^{th}$ level on the left branch can be readily obtained as $NLeft_{k-1} = NRight_k + NLeft_k$. The next step

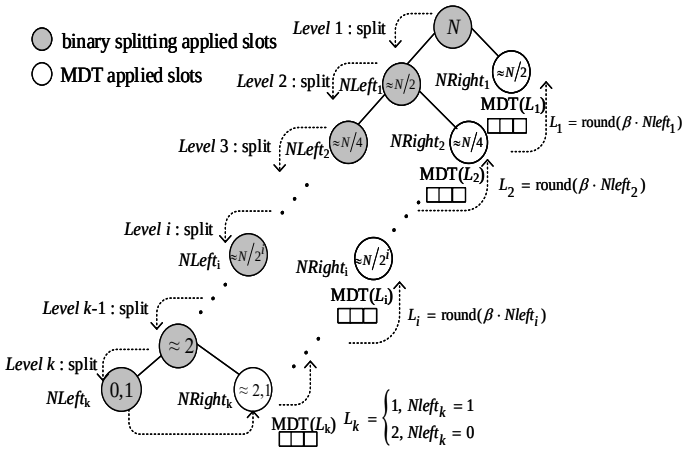


FIGURE 4 BS-MDT algorithm.

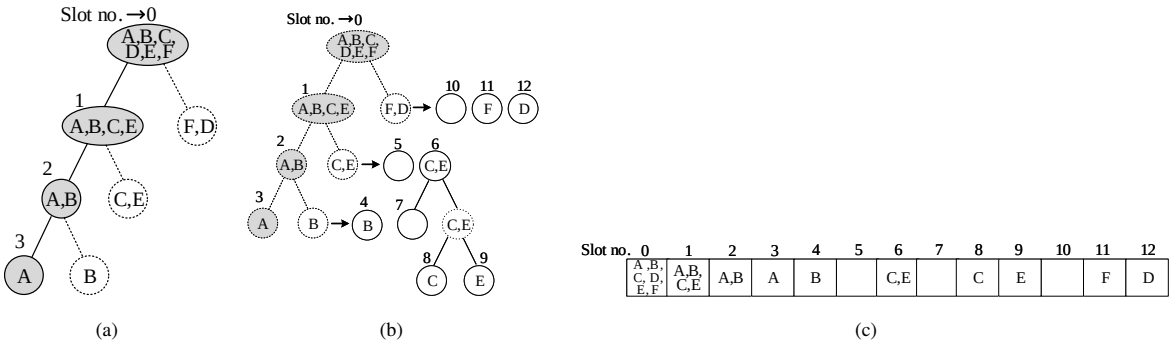


FIGURE 5 Example of transmission process in the BS-MDT algorithm with six initial collided tags.

is to resolve tag collision on the right branch of each level starting from the $(k-1)^{th}$ level upward to the 1^{st} level using MDT. Unlike most algorithms that usually assign the initial frame size equal to the estimated number of tags, we show that the system efficiency can increase by 1% by assigning $L = round(\beta N)$, where $\beta < 1$. The MDT algorithm executes based on two steps. In the first step, the competing tags randomly select a timeslot from a frame, which is made up of L slots. In the second step, the collided slots within the frame are resolved using the modified tree algorithm (MTA)[25,26] until all the collided tags are identified.

Figure 5 presents an example of the tag identification process of BS-MDT for six colliding Tags (A-F) in Slot 0. Figure 5(a) shows details of the first step, where binary splitting is performed three times and terminates at Slot 3 with one tag in it, *i.e.*, tag A. Figure 5(b) shows details of the second step. Tag B is first resolved by MTA with an initial frame size of $L_3 = 1$ and thus the number of colliding tags in Slot 2 is known to be two. Therefore, the initial frame size for resolving tags on the right hand side of the 2^{nd} level containing Tags C and E is $L_2 = 2$, assuming $\beta = 0.79$. Notice that Slot 7 is idle, implying that the next slot will be a sure collision, and hence it is skipped and immediately split into two slots, *i.e.*, Slots 8 and 9. Since we can know that Slot 1 contains 4 colliding tags, the initial frame size for resolving the right-hand-side slot of the 1^{st} level is $L_1 = 3$. A total of twelve slots are required to resolve the six tag collisions as shown in Figure 5(c).

Algorithm 1 and Algorithm 2 show the reader's and tag's pseudocode of proposed BS-MDT algorithm respectively. In Algorithm 1, the function SPLITTING () starts the initial binary splitting phase by broadcasting the SplitQuery command as given in line number 4. When the initial splitting phase is finished, the number of levels in the binary tree structure which is $level$ and the final feedback type f, can be identified. Using these two parameters, function BSMDT() is called the MDT algorithm for each right subgroup set which were created at the initial binary splitting phase. At the beginning of the algorithm,

Algorithm 1 Binary Splitting MDT: reader operation

```
[level, f] = SPLITTING()
BSMDT(level, f)
procedure SPLITTING()
    Broadcast SplitQuery, and level = 0
    while f = collision do
        Receive tag response and detect a collision
        if receives several responses then
            f = collision, and level = level+1
        else if receives no responses then
            f = idle
        else if receives only one response then
            Receive ID and store it
            f = successful
        end if
        Transmit f
    end while
    return level and f
end procedure
procedure BSMDT(level, f)
    if f = idle then
        nLeft = 0
        L = 2
    else if f = successful then
        nLeft = 1
        L = 1
    end if
    while level > 0 do
        numberOfsuccess = MDT(L)
        nRight= nRight + numberOfsuccess
        nLeft = nLeft + nRight
        L = round(nLeft * 0.79)
        level = level - 1
    end while
end procedure
procedure MDT(L)
    Broadcast Query with L, and SC = 0
    while SC < L do
        Receive tag response and detect a collision
        if receives several responses then
            Transmit f = collision, and tags = MTA()
            numberOfsuccess = numberOfsuccess + tags
        else if receives no responses then
            Transmit f = idle
        else if receives only one response then
            Receive ID and Transmit f = successful
            numberOfsuccess = numberOfsuccess + 1
        end if
        SC = SC + 1
    end while
    return numberOfsuccess
end procedure
```

```
52: procedure MTA()
53:     Counter = 2
54:     while Counter > 0 do
55:         Receive tag response and detect a collision
56:         if receives several responses then
57:             f = collision, and Counter = Counter+1
58:         else if receives no responses then
59:             f = idle, and Counter = Counter-1
60:         else if receives only one response then
61:             Receive ID and transmit f = successful Counter = Counter-1 and tags = tags + 1
62:         end if
63:     end while
64:     return tags
65: end procedure
```

the initial frame length L for MDT is identified based on last feedback type f derived in the initial binary splitting phase If f is idle the L is set to 2 or if f is success the L is equal to 1. In Algorithm 1, from line number 29 to 31, it shows the procedure of setting the initial frame length to the subsequent right sub group set. The function MDT() implements the MDT algorithm. In MDT(), the reader broadcast the Query command with the frame size L and in the meanwhile, the reader maintains a Slot Counter (SC) which is useful to identify the termination point of the MDT algorithm when the SC value is equal to frame size L. Initially the value of the SC is zero and it is incremented by one at the end of each slot. In each slot, the reader transmits the feedback types of idle, collision and success to the tags based on the receive responses from the tags. If the reader receives only a single response, the reader can identify the tag and sends f as successful, and when there are no responses the slot become an idle and reader sends f as idle. When the reader receives several responses in a given slot, it is a collided slot, and function MTA() is used to identify these collided tags and reader sends f as collision. In MTA(), let $Counter$ initialize 2, and when $Counter$ becomes zero the reader knows that the MTA has finished.

As given in Algorithm 2, a tag functions on initial splitting phase or MDT algorithm based on the SplitQuery and Query commands. When the tag whose $Counter$ value is zero receives the Query command, the $Counter$ value will be updated by adding a random number between 0 to L-1 to it. And, when a tag, whose $Counter$ value is not zero receives the query command, it's $Counter$ changes by adding L-1 to it. The MTA procedure is formed using the MTA() in reader's Algorithm 1 and the TagIdentify() in tag's Algorithm 2. In TAGIDENTIFY(), in order to identify the definite collisional slot the $previous f$ variable is maintained.

Further, to summarize the above mentioned procedures, the flow chart of proposed BS-MDT is shown in Figure 6 . If the reader broadcasts the command of SplittingQuery with initial frame size of $Q = 2$, the binary splitting phase will be started. The binary splitting step finishes when the reader identifies a success or an idle response from the current reading slot. The levels number and the response of last reading slot can be obtained. Using these obtained values , the reader then broadcasts the Query command with the initial frame size (L). The frame size (L) is chosen by rounding the obtaining values from total identified tags at each $level$ multiply by 0.79 to the nearest integer. Then the reader stays for the tag responses from L slots and the collided slots are further resolved using MTA.

3.3 | Analysis of the BS-MDT Algorithm

In this section, we analyze the average number of timeslots required by the MTA, MDT and BS-MDT algorithms to resolve all N tags. The system efficiency can be determined as

$$\eta(N) = \frac{N}{T(N)}, \tag{1}$$

where $T(N)$ is the required average number of timeslots to identify N collided tags.

Algorithm 2 Binary Splitting MDT: tag operation

```
1: if receive Reader's SplitQuery then
2:     Counter = a random binary number 0 and 1
3:     while f = collision do
4:         [f, Counter] = TAGIDENTIFY(Counter)
5:     end while
6: end if
7: while Counter >= 0 do
8:     if receive Reader's Query with L then
9:         if Counter = 0 then
10:            Counter = a random number from 0 to L - 1
11:        else
12:            Counter = Counter + L - 1
13:        end if
14:    else if receive Reader's f then
15:        Counter = TAGIDENTIFY(Counter)
16:    end if
17: end while
18: procedure TAGIDENTIFY(Counter)
19:    if Counter = 0 then
20:        Send ID and receive f from reader
21:        if f = successful then
22:            identified, and previousf = successful
23:        else if f = collision then
24:            Counter = Counter + a random binary number 0 or 1
25:            previousf = collision
26:        end if
27:    else
28:        if f = collision then
29:            Counter = Counter + 1
30:            previousf = collision
31:        else if f = idle then
32:            if previousf = collision and Counter = 1 then
33:                Counter = a random binary number 0 or 1
34:            else
35:                Counter = Counter - 1
36:                previousf = idle
37:            end if
38:        end if
39:    end if
40:    return Counter and f
41: end procedure
```

The average number of timeslots used in the MTA algorithm with biased splitting probability T_{MTA} can be stated as given in (2). This is derived by extending (3.13) in[23] with p as the splitting probability of the left branch in a binary tree structure:

$$T_{MTAbias}(N) = 1 - (1-p)^N + \sum_{i=0}^{N} \binom{N}{i} p^i (1-p)^i \left[p^{N-2i} + (1-p)^{N-2i}\right] T_{MTAbias}(i). \tag{2}$$

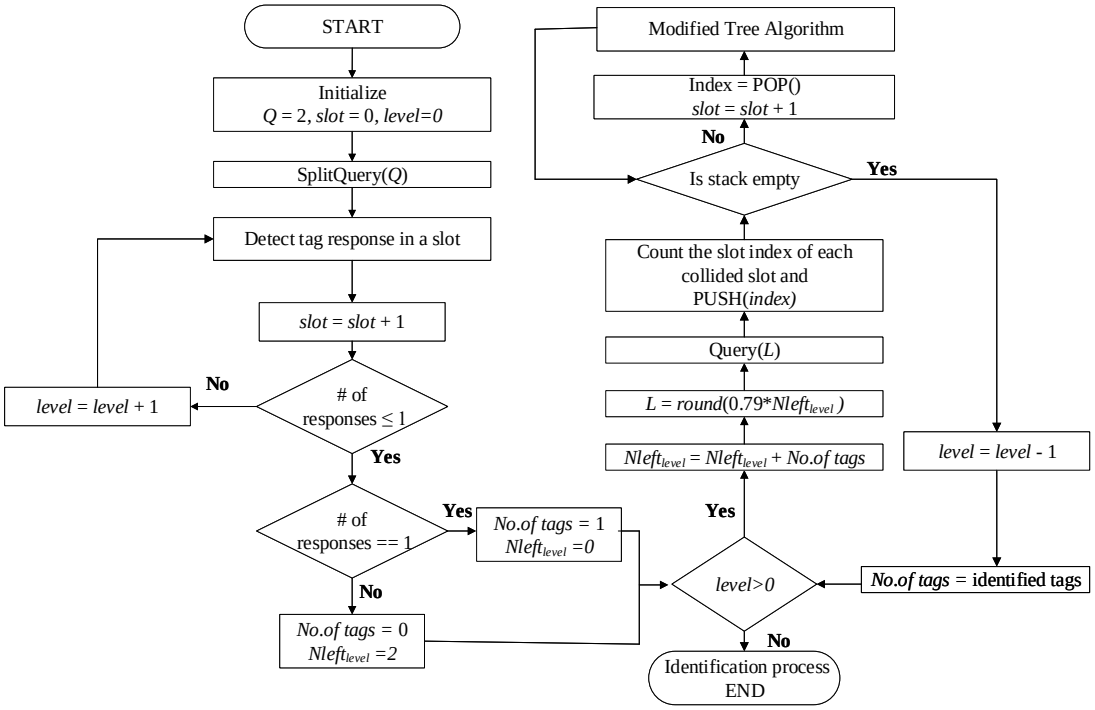


FIGURE 6 Flow chart of proposed BS-MDT algorithm.

Next, let $T_{MDT}(N, L)$ represent the average number of slots needed in the MDT algorithm to resolve the N colliding tags using an initial frame size of L and it can be mathematically given as

$$T_{MDT}(N, L) = L\left[B(N, 1/L, 0) + B(N, 1/L, 1) + \sum_{n=2}^{N} B(N, 1/L, n) \cdot T_{MTAbias}(n)\right], \tag{3}$$

where $B(n, p, i) = \binom{n}{i} p^i (1-p)^{n-i}$.

Note that $B(N, 1/L, 0)$ and $B(N, 1/L, 1)$ give the probabilities of idle and success slots respectively within an initial frame size of L. $B(N, 1/L, n) \cdot T_{MTAbias}(n)$ gives the number of slots required to resolve the collision happened when n tags out of N are collided in the frame with L slots using the MTA algorithm with biased splitting probability. In MDT, the relationship between the initial frame size L and the number of tags N can be obtained as $L = 0.79N$.

Finally, the average timeslots used in the proposed BS-MDT$T_{BSMDT}(N)$ can be derived as

$$T_{BSMDT}(N) = \sum_{j=0}^{N-1} \frac{1}{2^N} \binom{N}{j} \left[1 + T_{MDT}(j, q) + T_{BSMDT}(N-j)\right] + \frac{1}{2^N}\left[1 + T_{MDT}(N, 2)\right], \tag{4}$$

where $q = round(\beta(N-j))$ (*i.e.*, rounds $(\beta(N-j))$ to the nearest integer) and $T_{BSMDT}(0) = T_{BSMDT}(1) = 0$.

4 | SIMULATION PARAMETERS AND RESULTS

The performance of the proposed BS-MDT is evaluated and compared with BTA [22], TTA [23], MTA [24], TSA [26], BSTSA [27], Dynamic BTSA [28] and Splitting BTSA [28] in terms of system efficiency, time system efficiency, average number of collisions and idle slots. Although, the system efficiency has been the most common measure of performance for anti-collision protocols, recent studies for RFID systems aim at maximizing the time system efficiency as it takes into account the fact that idle slots are shorter than successful and collision slots. Various recommended time parameters as defined in the ISO/IEC 18000-6 Type B standard [30]play

a part in the calculation of actual time required for tag identification.
As given in [17], the system efficiency (η_{SE}) and time system efficiency (η_{TSE}) are defined by equation (5) and (6) respectively.

$$\eta_{SE} = \frac{N}{I(N) + S(N) + C(N)}, \tag{5}$$

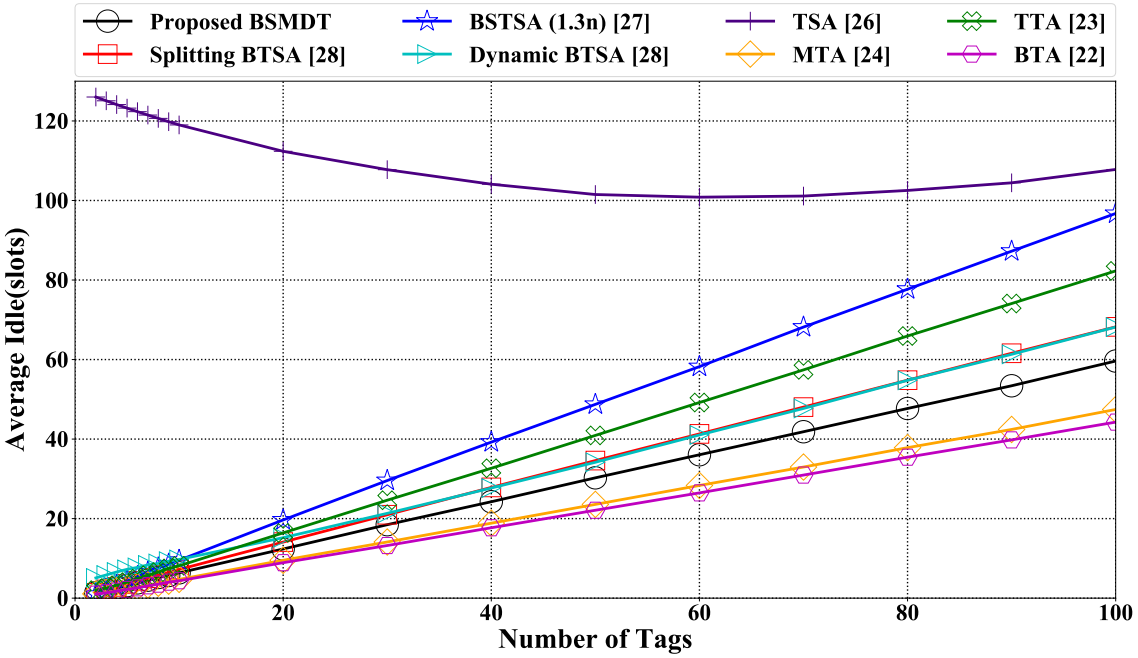
$$\eta_{TSE} = \frac{NT_S}{I(N)T_I + S(N)T_S + C(N)T_C}, \tag{6}$$

where $I(N), S(N)$ and $C(N)$ are the expected number of idle, successful and collision slots respectively, while T_I, T_S and T_C are their corresponding time duration according to ISO/IEC 18000-6 Type B standard [30]. We can calculate the T_I, T_S and T_C by multiplying the one-bit duration with the total number of bits required to transfer between a reader and a tag for the corresponding commands based on the ISO/IEC 18000-6 Type B standard [30]. To obtain the time system efficiency by following (6), we use the timing values given in [31] and the values are shown in Table 1. These values represent the time to process a node when it is in an idle T_I, a success T_S and a collision T_C states.

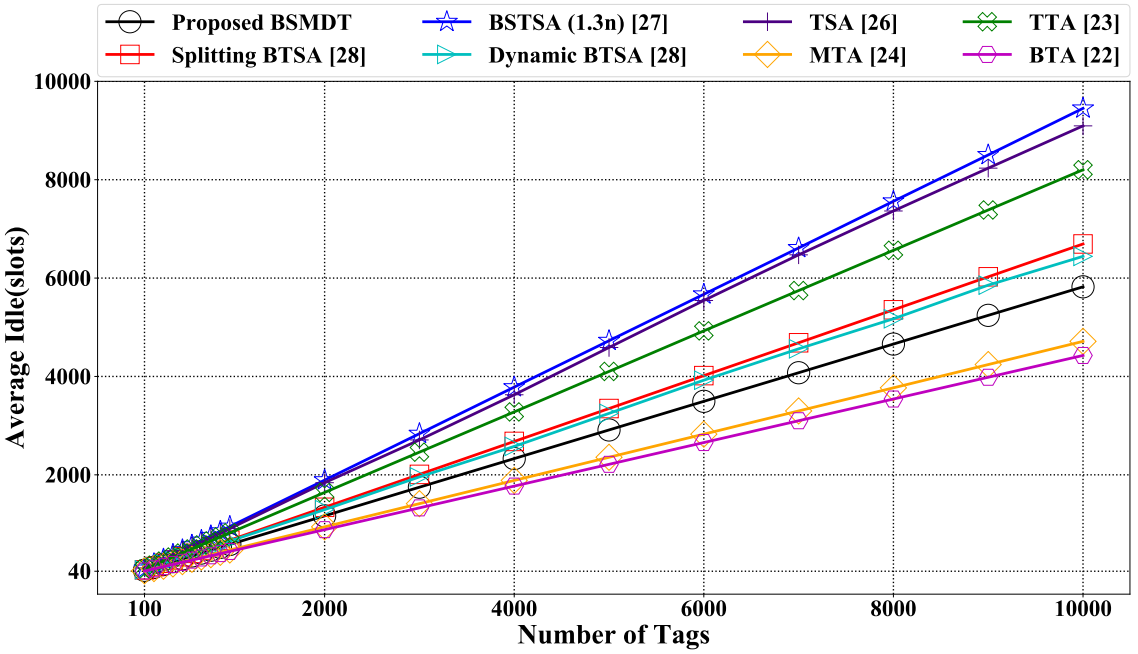
TABLE 1 Time for processing one node based on ISO/IEC 18000-6 Type B standard [31].

	Collision	**One tag reply**	**Idle**
Time for processing one node (ms)	4.025	9.975	2.025

The Figures 7, 8, 9 and 10 show the comparison of average number of idle slots, average number of collision slots, system efficiency, and and comparison of time system efficiency parameter values for above stated eight algorithms respectively. Figure 7(a), 8(a), 9(a) and 10(a) show resultant graphs for 2 to 100 range of tags and the Figure 7(b), 8(b), 9(b) and 10(b) show the resultant graphs for larger number of tags range from 100 to 10000 tags. Numerical results are obtained through our derived analytical model and verified through computer simulations. The parameters β and p are optimized through extensive simulations and the system efficiency is maximized when using $\beta = 0.79$ and $p = 0.418$. In our work, the initial Q value for Dynamic BTSA is considered as 4.0, while in the TSA, the initial frame length is 128. In order to estimate the number of tags in the TSA, the Vogt estimation method is used which is given in [13].
The Eq. (5) confirms that the number of idle and collision slots give a considerable effect to system efficiency where higher the number of idle and collision slots lower the system efficiency and vice versa. Traditionally in tree algorithm, the number of sub-groups that each collision slot is split has a direct impact on the number of idle and collision slots occurring in the system. Smaller is the sub-group size, higher are the number of collision with less idle slots. And it is true even in the opposite order. According to Figures 7 and 8 the lowest idle slots and the largest number of collision slots are given by BTA and MTA where both algorithms split the collided tags into two sub-groups throughout their splitting process. Further, it can be seen that the TTA which uses the splitting sub-groups size as three gives a higher number of idle and collision slots than BTA and MTA. Therefore, it verifies that when the collision tags are split into a small number of sub-groups, lower number of idle slots and higher the number of collision slots can be experienced. The MTA has experienced a higher number of idle slots as compared to BTA and this is expected to be happened after the splitting of definite collision slots which contribute more number of idle slots. The proposed algorithm gives a slightly higher number of idle slots than that of MTA as a reason of the empty sub-groups happen at the initial sub-grouping phase of the MDT algorithm.
As shown in Figure 8, the proposed BS-MDT algorithm and BSTSA offer comparatively low number of collision slots while the Splitting BTSA gives slightly higher collision slots than these two algorithms. As we know, these three algorithms have two main execution phases in their algorithms. In the first phase, they follow the binary splitting tag estimation method until the leftmost node in the binary tree structure becomes empty or contains a single tag. This mechanism points out that each sub-group contains a half the portion of the collided tags from the previous level of binary tree structure. In the second phase, the rightmost sub-group in each level of the binary tree structure is split into the sub-groups with the size equal to the estimated number of collided tags. The Splitting BTSA and BSTSA consider the estimated number of tags in the rightmost sub-group is equal to the number of identified tags at the leftmost subgroup and split the rightmost sub-group tags accordingly. In our proposed algorithm, we obtained an optimal sub-group size using β x identified the number of tags at the leftmost sub-group.



(a)

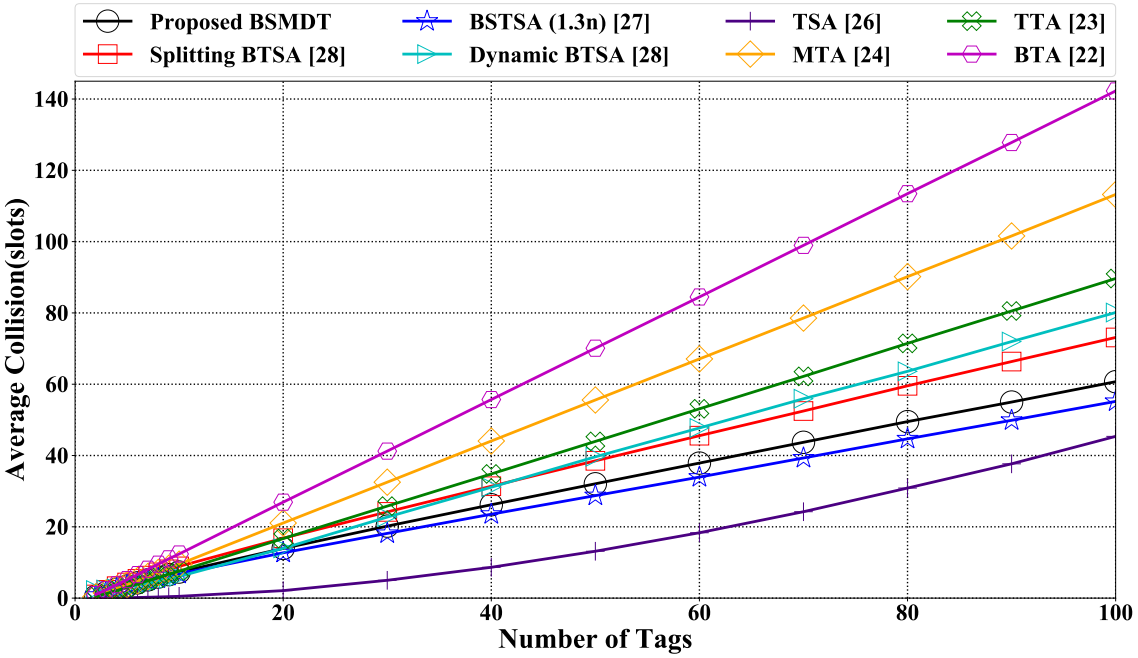


(b)

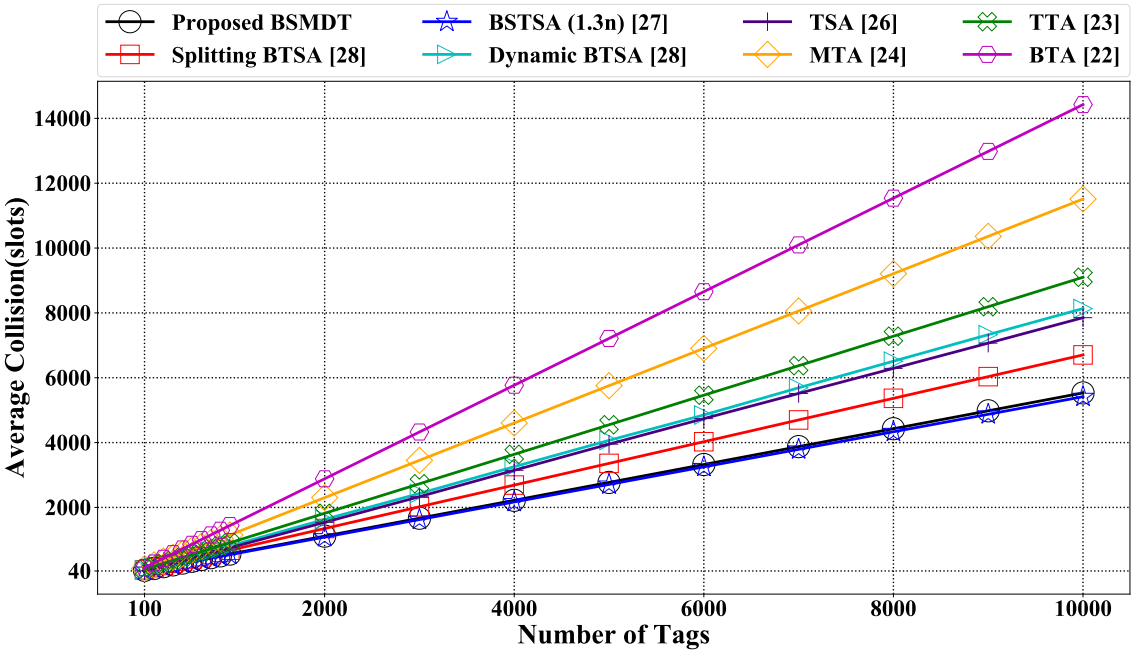
FIGURE 7 Average number of idle slots for various tree-based algorithms.

From Figures 7 and 8, it can be confirmed that this derived optimal frame size effectively works and control the number of idle and collision slots in the proposed method.

The TSA gives highest number of idle slots especially for the small number of tags range. In TSA, initially, the tags are split into 128 sub-groups and this results into a greater number of idle slots for the small number of tags. Furthermore, the estimation error happens from Vogt's estimator introduce a greater number of idle and collision slots in TSA. The TSA uses the Vogt's estimator to estimate the collided tags in the collided sub-groups where this estimated value decides the next sub-group size that the collided tags are split into. Though BSTSA uses TSA as the tag identification algorithm in its second phase, the



(a)

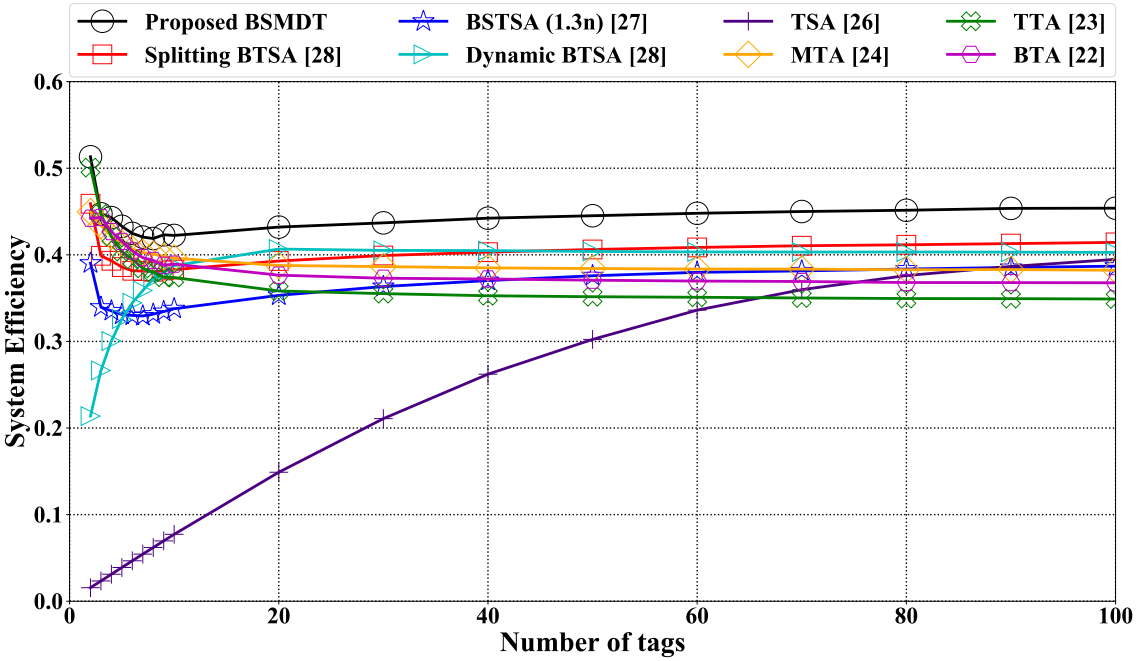


(b)

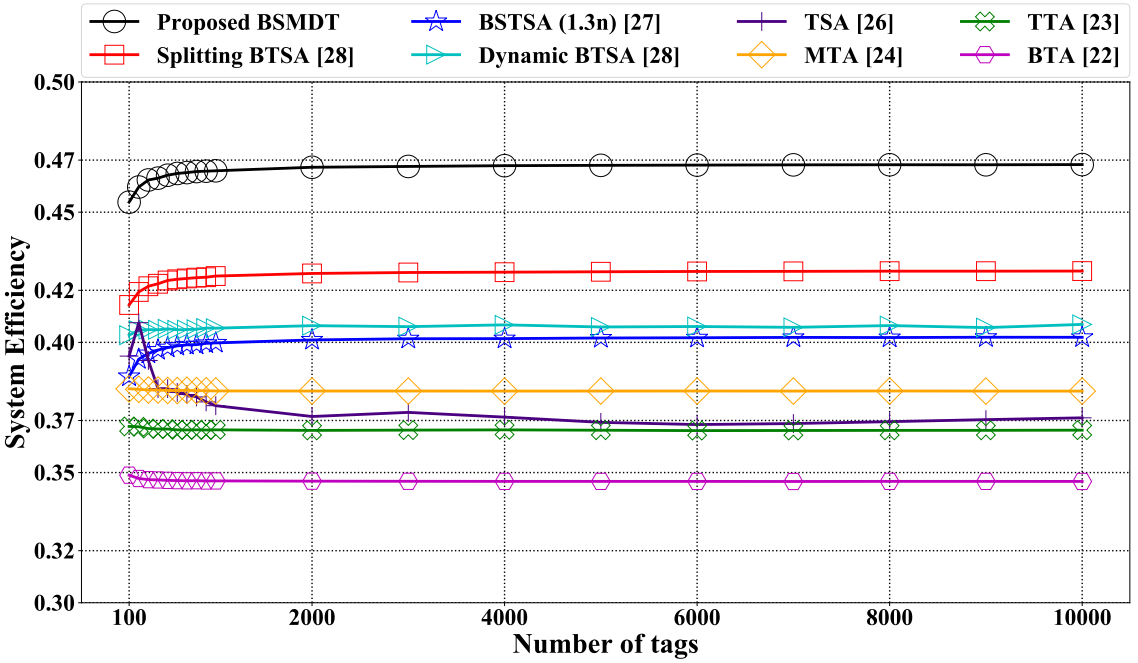
FIGURE 8 Average number of collision slots for various tree-based algorithms.

BSTSA experiences a lower number of idle and collision slots than TSA due to the effectiveness of the initial binary splitting tag estimation technique introduced in it.

The Dynamic BTSA and Splitting BTSA have slightly the same number of idle slots which are higher than that of the proposed algorithm. The dynamic BTSA also has two main phases like the proposed algorithm, BSTSA and splitting BTSA. Based on the received feedbacks of idle and collision slots, the dynamic BTSA can adjust the frame length closer to the number of tags in its first phase. In the second phase, the unidentified tags are initially grouped into 16 sub-groups and the collided groups among the initial subgroups are identified using BTA. This fixed initial frame size is highly affected by the performances of a



(a)

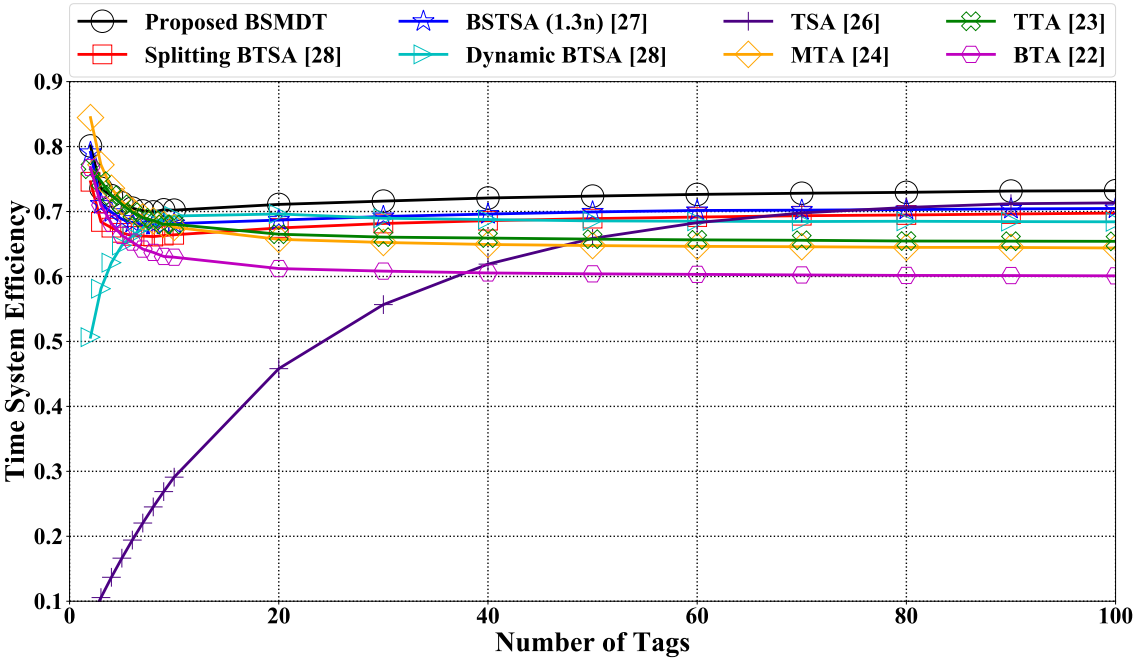


(b)

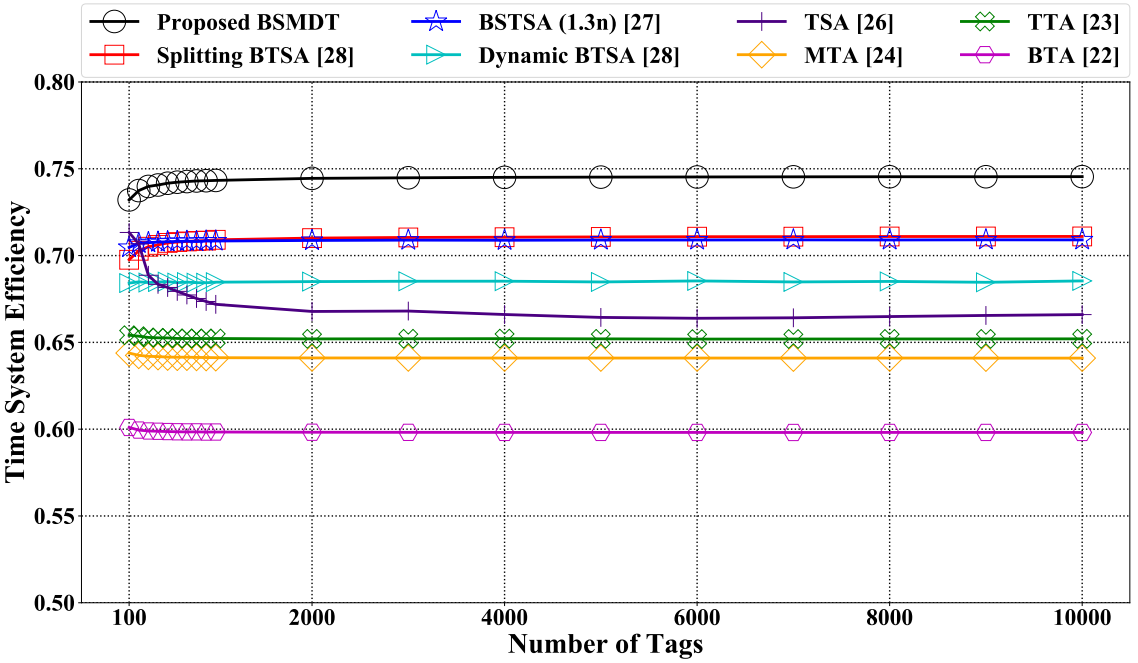
FIGURE 9 Comparison of system efficiency for various tree-based algorithms.

smaller number of tags such as less than 20 tags.

As shown in Figure 9, the BTA, TTA, MTA, and TSA offer the system efficiency less than 40%. The BTA, MTA and TTA, use a fixed sub-group size of 2 and 3 in their resolution process and this results in a higher number of idle and collision slots. In TSA, although it has a low number of collision slots, it achieves 37.5% of system efficiency since it is experiencing a higher number of idle slots. The BSTSA gives a system efficiency around 40% which is slightly higher than that of TSA as a result of the proposed elegant and powerful binary splitting tag estimation technique. As shown in Figures 7 and Fig.8, the dynamic BTSA maintains a lower number of collision and idle slots than BSTSA and TSA and this causes it to perform better than



(a)



(b)

FIGURE 10 Comparison of time system efficiency for various tree-based algorithms.

BSTSA and TSA and achieve 40% of system efficiency. The splitting BTSA achieves 42% of system efficiency by adopting the binary splitting tag estimation as an estimation process in its first phase and the BTSA as the tag identification technique in its second phase. Therefore, the splitting BTSA is improving the performance of dynamic BTSA which gives the system efficiency of 40%. The proposed BS-MDT algorithm tends to have a lower number of idle and collision slots due to the proposed procedure of identifying the initial frame length for MDT algorithm and the way of handling the definite collision slots in BS-MDT algorithm. As a result, it can grant the system efficiency of 46%.

We apply the above mentioned eight algorithms to the RFID system according to the ISO/IEC 18000-6 Type B standard[30] at a

data rate of 40 kbps with the timing values given in Table 1. The Figure 10 shows that the proposed BS-MDT algorithm can achieve better time system efficiency around 74% than all the other algorithms. This is even better than BSTSA which was primarily designed and optimized for maximum time system efficiency. It is possible that some protocols which have good system efficiency may not be as effective with the time system efficiency, such as MTA. The reverse is also true for instance, BSTSA. It is justified in Figure 10, that BTA, TTA, MTA, TSA and Dynamic TSA which have higher number of collisions gives lower time efficiency (less than 70%). Additionally, the BSTSA and splitting TSA offers nearly the same time efficiency of 71% since they have a considerable number of idles and collisions. The proposed algorithm provides the time efficiency of 74% as it has the lowest number of idle and collision slots as compared to other stated algorithms.

5 | CONCLUSIONS

In this paper, we proposed a highly efficient anti-collision algorithm, which gives better system efficiency and time system efficiency than all known protocols for a broad range of number of tags. The proposed BS-MDT can identify tags for RFID systems faster than other known tree-based algorithms by using the combination of binary search estimation method and modified dynamic tree algorithm. We also presented a new and complete mathematical analysis to derive the average number of timeslots required by several tree-based algorithms such as MTA, MDT and proposed BS-MDT algorithm to resolve a group of colliding tags. The mathematically derived results are verified by simulations. In our investigation, several other aspects of performance, namely average number of collision and idle slots required in the tag identification process are also evaluated and compared to stated seven well-known methods. Numerical results confirm that the BS-MDT offers the average system efficiency of 42% for small numbers of tags i.e, 2 to 100 and slightly above 46% for larger numbers of tags i.e, 100 to 10000. When applying the BS-MDT algorithm to the RFID system using ISO/IEC 18000-6 Type B standard[30], the BS-MDT algorithm gives the time system efficiency around 72% for small numbers of tags i.e, 2 to 100 and at least 74% for larger numbers of tags i.e, 100 to 10000. This is higher than that of all other existing algorithms.

ACKNOWLEDGEMENT

The first author would like to thank the Graduate School of Chulalongkorn University for the 100th anniversary Chulalongkorn University Fund for Doctoral Scholarship, Overseas Research Experience Scholarship and 90th Anniversary Chulalongkorn University Fund (Ratchdaphiseksomphot Endowment Fund).

References

1. Finkenzeller K. *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification*. Wiley Publishing; 2nd ed.2003.

2. Want R. An Introduction to RFID Technology. *IEEE Pervasive Comput.*. 2006;5(1):25–33.

3. Jia X, Feng Q, Fan T, Lei Q. RFID technology and its applications in Internet of Things (IoT). In: 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet):1282-1285; April, 2012.

4. Popovski P. Tree-Based Anti-Collision Protocols for RFID Tags:203-229. United Kingdom: John Wiley and Sons 1st ed.2010.

5. Zheng F, Kaiser T. *Anti-Collision of Multiple-Tag RFID Systems,*. John Wiley and Sons; 1st ed.2016.

6. Yang A, Hancke GP. RFID and Contactless Technology:351–385. Cham: Springer International Publishing 2017.

7. DK Klair, Chin Kwan-Wu, R Raad. A Survey and Tutorial of RFID Anti-Collision Protocols. *Commun. Surveys Tuts.* 2010;12(3):400–421.

8. DH Shih, Sun PL, DC Yen, SM Huang. Taxonomy and survey of RFID anti-collision protocols. *Computer Communications.* 2010;29(11):2150-2166.

9. Su J, Sheng Z, Leung VCM, Chen Y. Energy Efficient Tag Identification Algorithms For RFID: Survey, Motivation And New Design. *IEEE Wireless Communications.* 2019;:1-7.

10. Abramson N. THE ALOHA SYSTEM: Another Alternative for Computer Communications. In: AFIPS '70 (Fall):281–285ACM; 1970; New York, USA.

11. Roberts LG. ALOHA Packet System with and Without Slots and Capture. *SIGCOMM Comput. Commun. Rev.*. 1975;5(2):28–42.

12. Schoute FC. Dynamic frame length ALOHA. *IEEE Trans Commun.* 1983;31(4):565-568.

13. Vogt H. Efficient Object Identification with Passive RFID Tags. In: Pervasive '02:98–113Springer-Verlag; 2002; London, UK.

14. Wijayasekara SK, Annur R, Sasithong P, Vanichchanunt P, Nakpeerayuth S, Wuttisittikulkij L. A Reduced Complexity of Vahedi's Tag Estimation Method for DFSA. *Engineering Journal.* 2017;21(6):111-125.

15. EPCglobal . EPC^{TM}Radio-Frequency Identity Protocols Generation-2 UHF RFID Specification for RFID Air Interface Protocol for Communications at 860 MHz - 960 MHz. 2015;2.0.1.

16. Chen WT. An efficient RFID anticollision algorithm using early adjustment of frame length. *International Journal of Communication Systems.* 2016;29(18):2568-2579.

17. Šolić P, Radić J, Rožić N. Energy Efficient Tag Estimation Method for ALOHA-Based RFID Systems. *IEEE Sensors Journal.* 2014;14(10):3637-3647.

18. Chen WT. A Feasible and Easy-to-Implement Anticollision Algorithm for the EPCglobal UHF Class-1 Generation-2 RFID Protocol. *IEEE Trans Autom Sci Eng.* 2014;11(2):485-491.

19. Su J, Sheng Z, Xie L. A Collision-Tolerant-Based Anti-Collision Algorithm for Large Scale RFID System. *IEEE Commun Lett.* 2017;21(7):1517-1520.

20. Su J, Sheng Z, Hong D, Wen G. An Effective Frame Breaking Policy for Dynamic Framed Slotted Aloha in RFID. *IEEE Commun Lett.* 2016;20:692-695.

21. Capetanakis J. Tree Algorithms for Packet Broadcast Channels. *IEEE Trans. Inf. Theor.*. 1979;25(5):505–515.

22. Capetanakis J. Generalized TDMA: The Multi-Accessing Tree Protocol. *IEEE Transactions on Communications.* 1979;27(10):1476-1484.

23. Mathys P, Flajolet P. Q-ary Collision Resolution Algorithms in Random-access Systems with Free or Blocked Channel Access. *IEEE Trans Inf Theor.* 1985;31(2):217–243.

24. Massey JL. Collision-Resolution Algorithms and Random-Access Communications:73–137. Vienna: Springer Vienna 1981.

25. Yingqun Y, Giannakis GB. SICTA:a 0.693 Contention Tree Algorithm using Successive Interferenc Cancellation. In: INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies, 13-17 March 2005, Miami, FL, USA:1908–1916; 2005.

26. Bonuccelli MA, Lonetti F, Martelli F. Tree slotted aloha: a new protocol for tag identification in RFID networks. In: 2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks(WoWMoM'06):6 pp.-608; 2006.

27. La Porta TF, Maselli G, Petrioli C. Anticollision Protocols for Single-Reader RFID Systems: Temporal Analysis and Optimization. *IEEE Transactions on Mobile Computing.* 2011;10(2):267-279.

28. Wu H, Zeng Y, Feng J, Gu Y. Binary Tree Slotted ALOHA for Passive RFID Tag Anticollision. *IEEE Transactions on Parallel and Distributed Systems.* 2013;24(1):19-31.

29. Su J, Sheng Z, Xie L, Li G, AXLiu . Fast Splitting-Based Tag Identification Algorithm For Anti-Collision in UHF RFID System. *IEEE Transactions on Communications.* 2019;67(3):2527-2538.

30. 18000-6 ISO-IEC. Information technology Radio-frequency identification for item management-Part 6: Parameters for air interface communications at 860 MHz to 960 MHz. 2004;.

31. Kwon Dae-Ken, Kim Wan-Jin, Kim Hyoung-Nam. Improvement of Anti-Collision Performance for the ISO 18000-6 Type B RFID System. *IEICE Transactions.* 2007;90-B:2120-2125.