Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 6638936, 10 pages
https://doi.org/10.1155/2021/6638936

Research Article

Time Slot Detection-Based *M*-ary Tree Anticollision Identification Protocol for RFID Tags in the Internet of Things

Xiaojiao Yang,[1] **Bizao Wu**,[2] **Shixun Wu**,[1] **Xinxin Liu**,[1] **and W. G. Will Zhao**[3]

[1]*School of Information science and Engineering, Chongqing Jiaotong University, Chongqing 400074, China*
[2]*Solution Center, China Mobile IoT Company Limited, Chongqing 401121, China*
[3]*Business Administration, Lakehead University, Thunder Bay, ON, Canada P7B 5E1*

Correspondence should be addressed to Xiaojiao Yang; yxj@cqjtu.edu.cn

Received 22 December 2020; Revised 8 January 2021; Accepted 15 January 2021; Published 4 February 2021

Academic Editor: Zhili Zhou

Recently, a number of articles have proposed query tree algorithms based on bit tracking to solve the multitag collision problem in radio frequency identification systems. However, these algorithms still have problems such as idle slots and redundant prefixes. In this paper, a time slot detection-based *M*-ary tree (Time Slot Detection based *M*-ary tree, TSDM) tag anticollision algorithm has been proposed. When a collision occurs, the reader sends a predetection command to detect the distribution of the *m*-bit ID in the 2^m subslots; then, the time slot after predetection is processed according to the format of the frame-like. The idle time slots have been eliminate through the detection. Using a frame-like mode, only the frame start command carries parameters, and the other time slot start commands do not carry any parameters, thereby reducing the communication of each interaction. Firstly, the research status of the anticollision algorithm is summarized, and then the TSDM algorithm is explained in detail. Finally, through theoretical analysis and simulation, it is proved that the time cost of the TSDM algorithm proposed in this paper is reduced by 12.57%, the energy cost is reduced by 12.65%, and the key performance outperforms the other anticollision algorithms.

1. Introduction

Radiofrequency identification is a vital noncontact automatic identification technology in the sensing layer of the Internet of Things. In a typical RFID system, multiple tags are read by the reader simultaneously through a channel. When this channel is used by multiple tags simultaneously, it will cause signal overlap and interference problems: multitag collision problems. It will cause a waste of channel bandwidth, communication energy cost, and communication time cost. Therefore, developing effective anticollision strategies to improve the performance of RFID systems has an important significance.

Regarding the anticollision algorithm, massive research has been done and formed many valuable articles. Based on fundamentals and implementation, the anticollision algorithm can be divided into three main categories: ALOHA-based, hybrid-based, and tree-based algorithms. Based on the *M* -ary tree algorithm, we proposed a TSDM (Time Slot Detection based *M*-ary Tree) anticollision algorithm. Therefore, in the introduction, the current research status of the ALOHA-based algorithm and the hybrid-based algorithm is briefed, and the current research status of the tree algorithm is detailed.

These articles focus on the improvement of ALOHA-based algorithm [1–6], and the basic idea behind this kind of algorithm is that the reader estimates the number of tags according to the collision, success, and idle time slots first and then adjusts the frame length dynamically based on the estimation results. When the number of tags is equal to the number of time slots, the ALOHA-based algorithm can achieve the highest throughput rate of 36.8%. Ease of implementation is an advantage of the algorithm based on ALOHA. The randomness of the identification process and the starvation problem are the disadvantages of the ALOHA-based algorithm.

Those articles [7–11] proposed the hybrid-based algorithm, which combines part of the ALOHA-based algorithm and the tree-based algorithm. In the paper [11], based on the dynamic frame slot ALOHA algorithm, the binary tree algorithm is used to identify the collision slot. Compared with the ALOHA algorithm, the hybrid-based algorithm has a more considerable improvement in the throughput rate. However,

the starvation problem is still to be solved, and its implementation complexity is higher than the ALOHA-based algorithm.

Many papers have generated large amounts of valuable research about the tree-based algorithm [12–29]. The key technology of this algorithm is bit identification and bit tracking. The reader can group a set of tags based on collision bits. Moreover, repeat the step until there is only one tag in a group. The article [14] proposed the CwT algorithm. The tag replied that it is not the remaining ID but the partial ID with a length of w bits, which effectively reduces the average amount of data transferred between reader and tag. An MCT algorithm is proposed in the paper [15]. The data frame format is used in the MCT algorithm to process m-bit collision bits simultaneously, and the tag selects the response time slot based on the collision bits. Since only the first command of the frame is carrying the parameter, this will reduce the amount of the transferred data. An MQT algorithm based on the M-ary algorithm has been proposed [22]. The tag first mapped the m-bit ID from the highest collision bit according to rules [22]. Then, the specific mapped part is sent to the reader with the remaining ID. In the STT algorithm [27], the reader recorded continuous collision slots and idle slots in the identification process. The length of the prefix will make adjustments in agreement with the recorded value. The reader calculates the actual m-bit ID based on the collision bits of the received mapping part. In summary, the main problem of MCT is that there are too many idle slots. The redundant prefix problem still needs to be solved in the MQT, CwT, and STT algorithm. Therefore, the above algorithms still have room for improvement.

We propose time slot detection based on the M-ary tree anticollision algorithm (TSDM). The core idea of the TSDM algorithm is to detect first and identify second. The anticollision identification process is divided into two steps. The first step is to detect the m-bit ID$'$ value using a detection command. Then, synchronization results by sending a start command of the frame. Tags select the corresponding slot based on the calculation. In this way, we can eliminate idle slots through the proposed TSDM algorithm. In the typically M-ary tree algorithm, each command takes at least a parameter, and each reply includes the entire remaining ID. Therefore, redundant prefixes exist in the tag identification process. The TSDM algorithm is adopting a frame-like mode to deal with collision tags. Only the start command takes parameters, and the others take no parameters. Therefore, the TSDM algorithm reduces the average communication load between the tag and the reader. The simulation and theoretical analysis proved that the communication load, tag cost, time cost, and energy cost of the TSDM algorithm are better than the MCT, MQT, CwT, and STT algorithms compared with those articles [14, 15, 22, 27]. In a noisy environment, that is, take capture effects and path loss into the simulation, the TSDM algorithm still performs better than the others.

The writing framework of this article is as follows: Section 2 details the proposed TSDM algorithm. Section 3 is the theoretical analysis part, which analyses the performance of the TSDM algorithm. Section 4 is the simulation verification. First, the communication load, time cost, and energy cost of STT, CwT, MCT, MQT, and TSDM algorithms are compared by simulation in an ideal environment. Then, the impact of the environmental noise on the time cost is simulated. Section 5 is the conclusion. The performance of the proposed TSDM algorithm is evaluated and summarized.

2. The Proposed Time Slot Detection Protocol

The first part of this section details the registers and commands used in the TSDM algorithm. The second part introduces the implementation process of the TSDM algorithm in detail. The third part gives an example of the TSDM algorithm.

2.1. Related Commands and Registers. This article uses three special commands and a register for manipulating collisions. The register is for the tag, with the rest commands used for the reader.

BQ (*pre*): command BQ (*pre*) is a command for time slot detection. Only tags that match the prefix *pre* will reply to the command BQ.

FQ (*slot*): the command FQ (*slot*) is a command to start a frame. The parameter *slot* is the time slot distribution of the collision tags. Expressly, set the collision bit of the string received by the command BQ to 1 and set the remaining bits to 0. The value obtained is the *slot*.

SQ: command SQ is a command to start a time slot. Command SQ does not carry any parameters.

Register *RegM*: the M-bits register in the tag used for storing binary objects. *RegM* will be updated when the command BQ is received.

2.2. Algorithm Description. This section details the TSDM algorithm, illustrated by pseudocode and an example. Table 1 briefly explains the symbols used in the paper and the meanings they represent.

The pseudocode in Algorithm 1 details the implementation of the TSDM algorithm. *Qu* in step 1-1 is a queue for temporarily storing prefixes in the identification process. Steps 1-1 to 1-4 call function Function_AntiCol() to deal with the multitag anticollision problem.

The implementation process of Function_AntiCol() is described in steps 2-1 to 2-17 in Algorithm 1. Next, we will introduce steps 2-1 to 2-17 in detail. Step 2-2 means popping a string from the queue *Qu* as the parameter pre. Step 2-3 represents broadcast the command BQ (pre). Step 2-4, from the received string, computes the parameter *slot* and N. The parameter *slot* is calculated by setting the collision bit of the received string to 1 and setting the remaining bits to 0. The parameter N is the number of collision bits in the received string. Loop in steps 2-5 to 2-17 uses a frame-like mode for time slot detection. Steps 2-6 to 2-10 determine whether it is the first time slot of this frame. The reader sends the command FQ only in the first time slot, while the rest time slots send command SQ. Steps 2-12 to 2-16 judge whether the received data is colliding. A collision occurs when it queues the prefix *pre* into the queue *Qu*; otherwise, the reader directly identifies a tag.

The three commands in the proposed TSDM algorithm are BQ (*pre*), FQ (*slot*), and SQ. Next, we will introduce in

Table 1: Meaning of symbols.

Symbol	Meaning
L	The length of the tag's ID
ε	The parameter of command BQ, which means empty and occupies 1 bit
$\|\|$	Connection operator. For example, 0001‖1100 = 00011100
&	Bitwise AND operator. For example, 1101&0100 = 0100
<<	Left shift operator. For example, 0001 < <2 = 0100
$\|S\|$	Calculate the length of the string S. For example, $\|0001\| = 4$
ID $[i, j]$	Get the i-jth string of ID, where $i \le j$. For example, if ID = 11101010 then ID $[3, 5] = 101$
Count (x)	Count the number of 1 in the binary number x. For example, count $(1101) = 3$

```
1   Reader Operation
1-1:  Qu={ε};
1-2:  while(Qu≠NULL) do
1-3:     Function _AntiCol()
1-4:  end while
2   Anti-Collision processing
2-1:  Function _AntiCol()
2-2:  {pre=Deque(Qu)
2-3:     Broadcast command BQ(pre);
2-4:        (slot, N)=ConvertResponse(Received strings);
2-5:        for (i=1; i≤N; i++) do
2-6:           ifi==1 do
2-7:              Broadcast command FQ(slot)
2-8:           else
2-9:              Broadcast command SQ
2-10:          end if
2-11:          left_ID=received string
2-12:          ifleft_ID collision do
2-13:             Generate the parameter pre and insert it into Qu
2-14:          else
2-15:             Obtain the corresponding tag ID
2-16:          end if
2-17:       end for}
```

Algorithm 1: Pseudocode of the TSDM algorithm.

detail the process of the tags after received the above three commands.

Received command BQ(*pre*): if ID[1:|*pre*|] equals the parameter pre, the tag calculates the value of *RegM* first and then sends it to the reader. Otherwise, set *RegM* to NULL and will not reply. Algorithm 2 shows the computation procedure of *RegM*. Initialize register in Algorithm 2 means that *RegM* is a binary number with a length of 2^m bits and set its initial value to 1. For instance, $RegM = 00000001$ when $m = 3$. The first step assigns the $|pre| + 1$ to $|pre| + m$ bits of the tag′ID to the variable $B(i)$. The second step converts binary to decimal. The third step left shifts *RegM* by $m(i)$ bits and assigns the result to *RegM*.

Received command FQ(*slot*) or SQ: if *RegM*==NULL, the label will not reply. On the contrary, if *RegM*≠NULL, the tag sends ID[|*pre*| + *m* + 1:*L*] to the reader in the *Td*-th time slot. The calculation formula of *Td* is $Td = \text{Count}((RegM\text{-}1)\&slot) + 1$.

```
Initialize register:|RegM|=2^m,and RegM = 1
1:  B(i)=ID[|pre|+1:|pre|+m]
2:  m(i)=Binary_to_ Decimal(B(i))
3:  RegM=RegM<< m(i)
```

Algorithm 2. Pseudocode of update the register *RegM*.

2.3. An Example of the Proposed TSDM Algorithm. To describe our proposed algorithm intuitively, in this part, we will use the example in Table 2 and the structure figure in Figure 1 to illustrate the implementation.

When $m = 3$, Table 2 illustrates the detailed process for identifying seven tags.

Step1. The reader broadcasts the command BQ(ε), where the parameter ε means empty. After receiving the command

Table 2: An example of the TSDM algorithm.

	Commands	Reply	Identified tags
1.	BQ(Ɛ)	X0X00X0X	Null
2.	FQ(10100101)	10001	00010001 (*T1*)
3.	SQ	X0XXX	Null
4.	SQ	01011	10101011 (*T5*)
5.	SQ	000XX	Null
6.	BQ(010)	0X00XX00	Null
7.	FQ(01001100)	11	01000111 (*T2*)
8.	SQ	11	01010011 (*T3*)
9.	SQ	00	01010100 (*T4*)
10.	BQ(111000)	X00X	11100000(*T6*) 11100011 (*T7*)

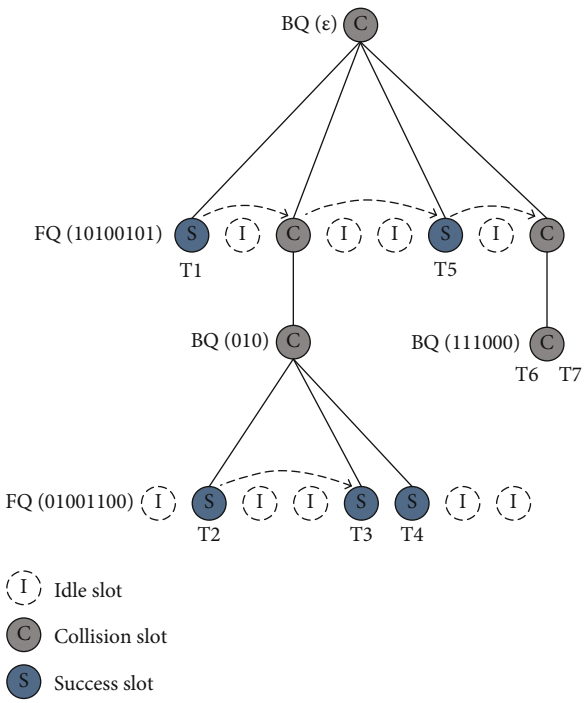

Figure 1: Structure of the identification process.

BQ(Ɛ), the tag maps the highest 3-bit ID according to the rule of Algorithm 2, then saves it to the register RegM and replies, without the need to match the prefix.

Step 2~step5. The reader broadcasts the command FQ(10100101), where the parameter 10100101 represents the distribution of tags. The tag chooses the corresponding slot depending on the value of *Td* after receiving FQ(10100101). Taking T1 as an example, after receiving the command FQ(10100101), the calculation process of *Td* is as follows: $Td = \text{Count}((RegM - 1)\&slot) + 1 = \text{Count}((00000001 - 1)\ \&10100101) + 1 = \text{Count}(00000000) + 1 = 1$. Therefore, *T1* will reply to the reader in the first and current time slot. Similarly, *Td*'s value for *T2*, *T3*, *T4*, *T5*, *T6*, and *T7* are 2, 2, 2, 3, 4, and 4, respectively. Therefore, the tags *T2*, *T3*, and *T4* will send the remaining ID in the second time slot (i.e., step 3 in Table 2). Tag *T5* will send the remaining ID in the third time slot (i.e., step 4 in Table 2). Tags *T5*, *T6*, and *T7* will send the remaining IDs in the fourth time slot (i.e., step 5 in Table 2).

Step 6. The reader broadcasts the command BQ(010), where parameter 010 indicates the prefix needs to match. The tag first compares the prefix 010 and the highest 3 bits of the ID after receiving the command BQ(010). If they are the same, the tag maps the 3-bit data starting from the highest bit of the remaining ID according to Algorithm 2, then saves it to the register *RegM* and replies. Otherwise, the tag sets *RegM* to NULL and keeps silent, waiting for the next command BQ. In step 6, only the first three bits of *T2*, *T3*, and *T4* are the same as 010, so only they will reply.

Step7~step9. Only T2, T3, and T4 will reply to the commands since the *RegM* of other unidentified tags is all NULL. The identification process has been described in detail before, and we will no cover it here.

Step 10. The reader broadcasts the command BQ(111000). After receiving the command, only *T6* and *T7* will reply. Since the remaining part of the ID has 2 bits (less than 3), the tags map and reply only 2 bits. After the reader receives X00X, it decodes the collision bits, and the final two bits of ID are 00 and 11, respectively. Due to the uniqueness of the tag ID, T6 and *T7* are directly identified.

We can conclude that the proposed TSDM algorithm only needs ten commands to identify seven tags. The structure of the TSDM algorithm in Figure 1 looks similar to the *M*-ary tree. However, idle leaf nodes are skipping through the time slot detection in the TSDM algorithm.

Figure 1 is the structure of the identification process of tags *T1*-*T7*. Due to the predetection of the command BQ, idle time slots are skipping in the TSDM algorithm. It reduces the average number of commands sent by the reader. Besides, Figure 1 intuitively shows the identification process, which is similar to a frame. Compared with the typical *M*-ary algorithm, the parameters carried by the commands of the TSDM algorithm are reduced. Above all, the proposed TSDM algorithm not only reduces time cost but also reduces the energy cost.

3. Theoretical Analyses

In the TSDM algorithm, the larger the m, the larger the M is since the number of groups is $M = 2^m$. Therefore, the greater the value of m, the faster the collision tag is recognized by the reader. However, the larger the value of m, the larger the mapping value. It will increase the computational complexity and increase the average number of bits between the reader and the tag. In the TSDM algorithm, the recommended value of m is 3. In this section, the number of the average paging times of the TSDM algorithm is analyzed theoretically.

In the TSDM algorithm, the number of command BQ sent by the reader is equal to the number of collision slots. Cleary, the sum number of command FQ and command SQ is equal to the number of nonempty time slots in the *M*-ary tree algorithm.

Let n be the number of tags in the broadcast domain of the reader. Let $C(n)$ be the sum of collision slots in the whole

identification process. Let the number of paging times of the TSDM algorithm be $Tt(n)$. $Tt(n)$ can be expressed as

$$Tt(n) = 2C(n) + n. \tag{1}$$

It can be seen from formula (1) that if $C(n)$ is obtain, $Tt(n)$ is obtain. Therefore, we first have to calculate what $C(n)$ is.

Theorem 1.

$$C(n) = \begin{cases} 0, & n = 01 \\ \sum_{j=0}^{7} \sum_{i=2}^{n-1} P\left(S_j = i \mid (A_0 \geq 1) \cap (A_1 \geq 1)\right) C(i), & n \geq 2 \end{cases}. \tag{2}$$

The proof of Theorem 1 is as follows:

When $n = 0$ or 1, no collision slot that exists, so $C(n) = 0$.

When $n \geq 2$, collision tags will be divided into M $(M = 2^m)$ subsets, based on the tag's m-bis ID starting from the highest collision bit. In the TSDM algorithm, the recommended value of m is 3, so we take $m = 3$ as an example. When $m = 3$, the 8 subsets are as follows: $S_0 = 000$, $S_1 = 001$, $S_2 = 010$, $S_3 = 011$, $S_4 = 100$, $S_5 = 101$, $S_6 = 110$, and $S_7 = 111$. If the 8 subsets are grouped by the highest collision bit, they will be divided into two subsets: $A_0 = [000,001,010,011]$ and $A_1 = [100,101,110,111]$.

Only if there is at least one tag in both subset A_0 and subset A_1 simultaneously, the reader can identify the highest collision bit. We can deduce that the probability of i tags falling in the slot S_j is $P(S_j = i \mid (A_0 \geq 1) \cap (A_1 \geq 1))$. Proved formula (2) using a recursive algorithm.

Lemma 1.

$$P(Sj = i|(A_0 \geq 1) \cap (A_1 \geq 1)) = \frac{\left(1 - (3/7)^{n-i}\right) \binom{i}{n} (1/8)^i (7/8)^{n-i}}{1 - 2(1/2)^n}. \tag{3}$$

Proof. From the rules of set operation, we can get

$$\begin{aligned} &P(Sj = i|(A_0 \geq 1) \cap (A_1 \geq 1)) \\ &= \frac{P\left((A_0 \geq 1) \cap (A_1 \geq 1) \middle| S_j = 1\right) \times P\left(S_j = i\right)}{P((A_0 \geq 1) \cap (A_1 \geq 1))}. \end{aligned} \tag{4}$$

If the tag's ID has a law of binomial distribution, the probability of i tags falling in subset S_0 is

$$P\left(S_j = i\right) = \binom{i}{n} \left(\frac{1}{8}\right)^i \left(\frac{7}{8}\right)^{n-i}. \tag{5}$$

Table 3: Parameters used in simulations.

Parameters	Value
t_1	25 us
t_2	25 us
v	160 kbps
P_t	825 MW
P_e	125 MW
L	96 bits

Next, respectively, calculate the numerator and denominator in formula (4). The calculation processes of the molecular are as follows:

$$\begin{aligned} &P\left((A_0 \geq 1) \cap (A_1 \geq 1) \middle| S_j = i\right), \\ &= 1 - P\left((A_0 = 0) \cap (A_1 = 0) \middle| S_j = i\right), \\ &= 1 - \frac{P\left(((A_0 = 0) \cup (A_1 = 0)) \cap S_j = i\right)}{P\left(S_j = i\right)}. \end{aligned} \tag{6}$$

Because the probability that a tag locates in subsets S_0 to S_7 is the same, we use subset S_0 as our example. Let $j = 0$ and take it into formula (7).

$$\begin{aligned} &P\left(((A_0 = 0) \cup (A_1 = 0)) \cap \left(S_j = i\right)\right) \\ &= P(((A_0 = 0) \cup (A_1 = 0)) \cap (S_0 = i)) \\ &= P((A_0 = 0) \cap (S_0 = i)) + P((A_1 = 0) \cap (S_0 = i)). \\ &\quad -P((A_0 = 0) \cap (A_1 = 0) \cap (S_0 = i)) \\ &= \binom{i}{n} \left(\frac{1}{8}\right)^i \left(\frac{3}{8}\right)^{n-i} \end{aligned} \tag{7}$$

If we want to compute the probability that a tag locates in subsets S_1 to S_7, the same type of calculation would be made. Thus, formula (8) can be derived from formula (7).

$$P\left(((A_0 = 0) \cup (A_1 = 0)) \cap \left(S_j = i\right)\right) = \binom{i}{n} \left(\frac{1}{8}\right)^i \left(\frac{3}{8}\right)^{n-i}. \tag{8}$$

Substituting formulas (5) and (8) into formula (6), we can get formula (9).

$$P\left((A_0 \geq 1) \cap (A_1 \geq 1) \middle| S_j = i\right) = 1 - \left(\frac{3}{7}\right)^{n-i}. \tag{9}$$

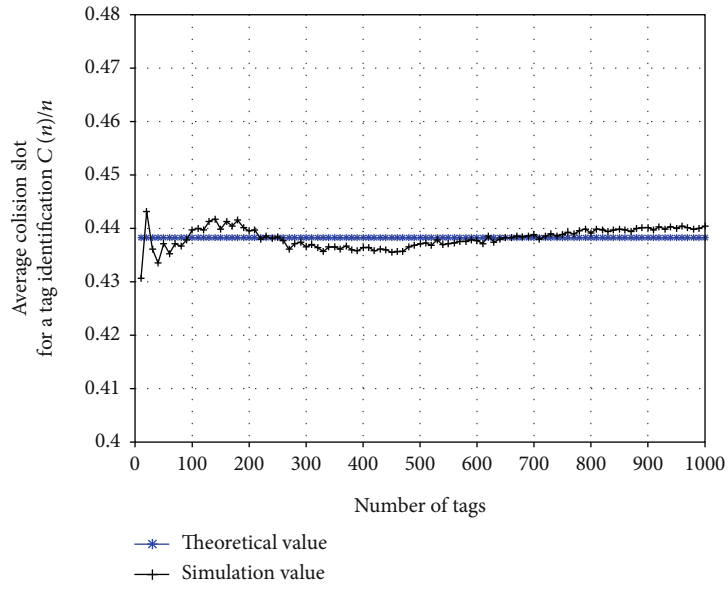

Figure 2: Theoretical and simulation results of collision slots for a tag identification.

Finally, the denominator of formula (4) is calculated by the formula (10).

$$\begin{aligned} &P((A_0 \geq 1) \cap (A_1 \geq 1)) \\ &= 1 - P((A_0 = 0) \cup (A_1 = 0)) \\ &= 1 - P(A_0 = 0) - P(A_1 = 0) + P((A_0 = 0) \cap (A_1 = 0)). \\ &= 1 - 2\left(\frac{1}{2}\right)^n \end{aligned} \tag{10}$$

Putting formulas (5), (9), and (10) into formula (4), Lemma 1 is proved.

Incorporating Lemma 1 into Theorem 1, we can get $C(n) \approx 0.4382n$ by the iterative calculation on the computer. Putting the value of $C(n)$ into equation (1), we can get $T(n) \approx 1.876n$. The parameters in Table 3 are for simulation, and we get the simulation results by simulating the implementation process of the proposed TSDM algorithm. As shown in Figure 2, the simulated value of $C(n)$ is almost consistent with the theoretical value.

4. Simulation Results

In this section, we will start with the definition of command format and parameter values. Next, the performance of MCT, MQT, CWT, STT, and TSDM algorithms are compared by simulation. Finally, we will discuss the influence of environmental noise on the algorithm.

4.1. Simulation Configuration. In this part, a typical passive RFID system will be simulated by Matlab software, which includes a reader and several passive tags. Until the reader identifies the tag, the reader will not know the ID and the total number of tags. Tag generation obeys the typical binomial distribution, and each ID is unique in the simulation. The communication channel between the reader and the tag is ideal in Section 4.2. We will consider the environmental noise in Section 4.3.

The link sequence diagram of the TSDM algorithm is in Figure 3. The values of parameters t_1 and t_2 in Figure 3 are in Table 3. The parameter v in Table 3 represents the data rate between the reader and the tag. The last parameter L represents the length of the tag's ID. The parameters P_t and P_e are used to calculate the energy overhead, where P_t represents the reader's carrier power, and P_e represents the additional carrier power produced by the reader during the tag reply to the reader.

The proposed TSDM algorithm, MCT algorithm, MQT algorithm, CwT algorithm, and STT algorithm are simulated by Matlab software in the next part. In the TSDM, the value of m is 3. Set the variables' optimal values in the rest of the articles for the simulation validation's best performance.

4.2. Performance Comparison. Figure 2 compares the critical performance of the MQT algorithm, the MCT algorithm, the STT algorithm, and the CwT algorithm, as well as the proposed TSDM algorithm by simulation.

(1) Total communication load. The average total bits for a tag identification in Figure 4(a) represent the total communication load. It can be seen from Figure 4(a) that the average total bits for a tag identification of the TSDM algorithm are about 225 bits, which is less than MQT, MCT, STT, and CwT. Since there is no idle time slot and redundant parameters, The TSDM algorithm tremendously reduces the average bits transferred between reader and tag

(2) Communication load of the tag. The tag's average bits for a tag identification in Figure 4(b) represent the

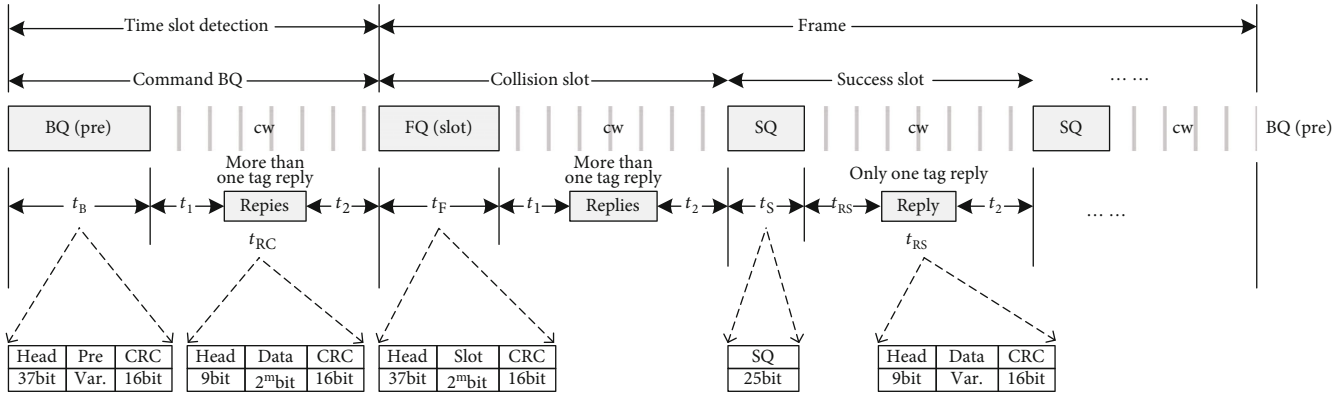

FIGURE 3: The format of commands in the TSDM algorithm.

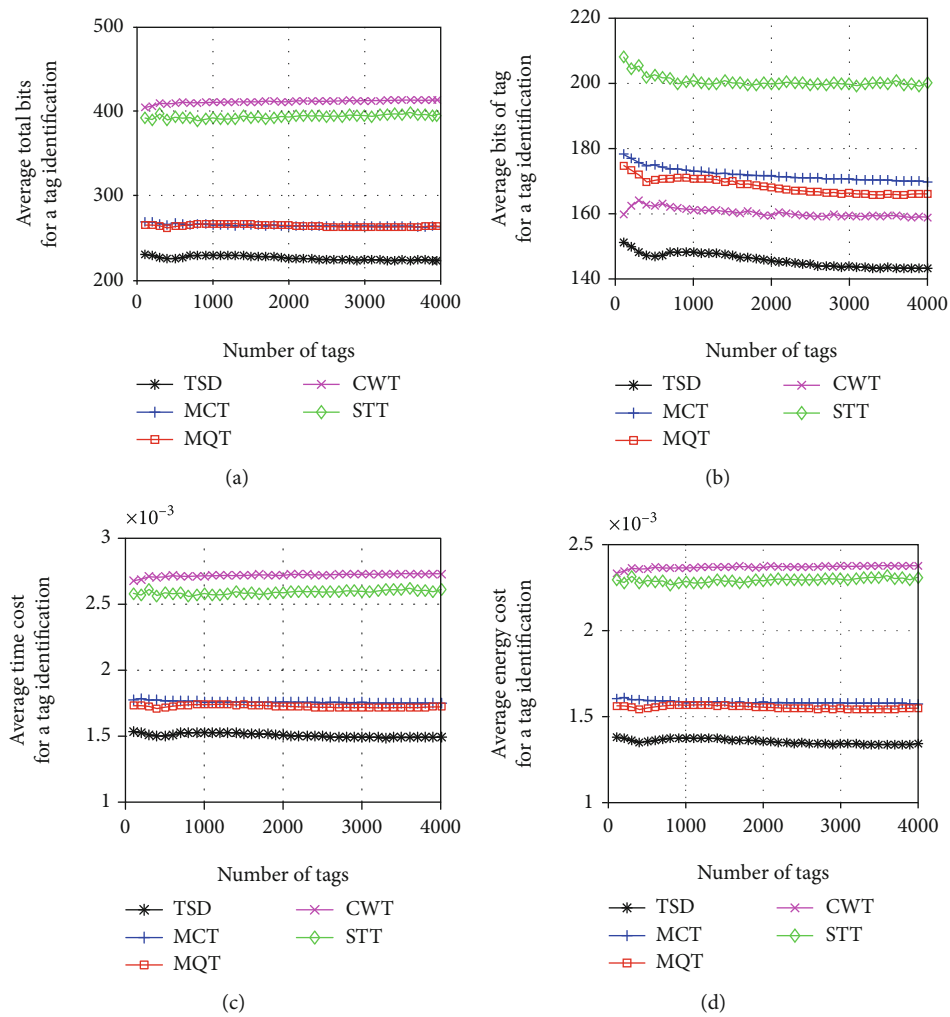

FIGURE 4: Simulation results: (a)average total bits for a tag identification, (b) average bits of tag for a tag identification, (c) average time cost for a tag identification, and (d) average energy cost for a tag identification.

tag's communication load. Most tags are passive or battery-powered In the RFID system, so the tag's communication load has a significant impact on the tag's useful lifespan. The lower the tag's communication load, the lower the algorithm's energy cost, so the longer the tag's life. As shown in Figure 4(b), the communication load of tag rank from low to high is as follows: TSDM, CwT, MQT, MCT, and STT.

Table 4: The time cost table of five methods.

Tag number Algorithm	100	1000	2000	3000	4000	Average value
MQT	1.7316 ms	1.7396 ms	1.7292 ms	1.7194 ms	1.7229 ms	1.729 ms
MCT	1.7800 ms	1.7560 ms	1.7610 ms	1.7570 ms	1.7510 ms	1.761 ms
CwT	2.6800 ms	2.7170 ms	2.7210 ms	2.7300 ms	2.7330 ms	2.716 ms
STT	2.5850 ms	2.5840 ms	2.5890 ms	2.6000 ms	2.6140 ms	2.594 ms
TSDM	1.5320 ms	1.5310 ms	1.5070 ms	1.4932 ms	1.4930 ms	1.511 ms
Decrement rate	11.53%	11.99%	12.85%	13.16%	13.34%	12.57%

Table 5: The energy cost table of five methods.

Tag number Algorithm	100	1000	2000	3000	4000	Average value
MQT	1.5651 mJ	1.5685 mJ	1.5580 mJ	1.5484 mJ	1.5511 mJ	1.558 mJ
MCT	1.6070 mJ	1.5840 mJ	1.5870 mJ	1.5830 mJ	1.5770 mJ	1.588 mJ
CwT	2.3360 mJ	2.3670 mJ	2.3700 mJ	2.3770 mJ	2.3790 mJ	2.366 mJ
STT	2.2960 mJ	2.2880 mJ	2.2920 mJ	2.3010 mJ	2.3130 mJ	2.318 mJ
TSDM	1.3820 mJ	1.3790 mJ	1.3570 mJ	1.3440 mJ	1.3440 mJ	1.361 mJ
Decrement rate	11.7%	12.08%	12.9%	13.2%	13.35%	12.65%

The TSDM algorithm proposed has the least average tag communication load

(3) Time cost. The average time cost for a tag identification represents the time cost. The lower the time cost, the more tags can be identified per unit time. Figure 4(c) shows that the TSDM algorithm has the lowest time cost. The average time to identify a tag is about 1.51 ms when using the TSDM algorithm. The average time to identify a tag is as follows: MQT is about 1.73 ms, MCT is about 1.76 ms, STT is about 2.6 ms, and CwT is about 2.72 ms. The time cost of the TSDM algorithm is reduced by at least 12.57% compared with the other algorithm

(4) Energy cost. The average energy cost for a tag identification represents the energy cost. Many RFID tags are battery-powered or passive. Therefore, the lower the power cost, the longer the lifespan of a tag. Figure 4(d) shows that the average energy cost for a tag identification of the TSDM algorithm is about 1.36 mJ. Its performance is better than other algorithms that are comparing below. The energy cost of the TSDM algorithm is reduced by at least 12.65% compared with the other algorithms

Table 4 lists the average time cost of MQT, MCT, CwT, STT, and TSDM for a tag identification. For instance, in the TSDM algorithm, the total time to recognize 100 tags is about 153.2 ms. We divide the total time by the number of tags to get the average time for a tag identification that is about 1.532 ms. As you can see in Table 4, the TSDM algorithm spends the least time for a tag identification, followed by the MQT algorithm. Therefore, we can conclude that the time cost of the TSDM algorithm is saved at least 12.57%, compared to the other four algorithms.

Table 5 lists the average energy cost of MQT, MCT, CwT, STT, and TSDM for a tag identification. The average energy cost can be obtained by referring to the calculation method of the average time. As we can see in Table 5, compared with MQT, MCT, CwT, and STT algorithm, the TSDM algorithm cut energy cost by at least 12.65%.

4.3. Effect of Environmental Noise. In a realistic environment, path loss and capture effects may cause randomness in the identification process, which will lead to deviations in the algorithm's performance between the realistic environment and the ideal environment. Next, the implementation process of TSDM, MQT, and MQT in the realistic environment is simulating by Matlab software.

When path loss occurs, the reader usually does not recognize the signal to which the tag is replying. Let Pd be the probability of path loss and $Pd \in [0, 0.2]$. Only the signal from the tag with the strongest signal will be detected when the capture effect occurs. Let Pc be the probability of path loss and $Pc \in [0, 0.2]$. Since the energy cost is proportional to the time cost, we are only analyzing the time cost here.

Figure 5(a) shows the change rules of the average time to identify a tag due to Pd's change when $Pc = 0.1$. When path loss occurs, a collision slot may be inaccurately considered a successful slot or an idle slot, and a successful slot may be inaccurately considered an idle slot. It will lead to uncertainty in the tag identification process, thereby increasing the identification time. Compared with MCT and MCT, the proposed

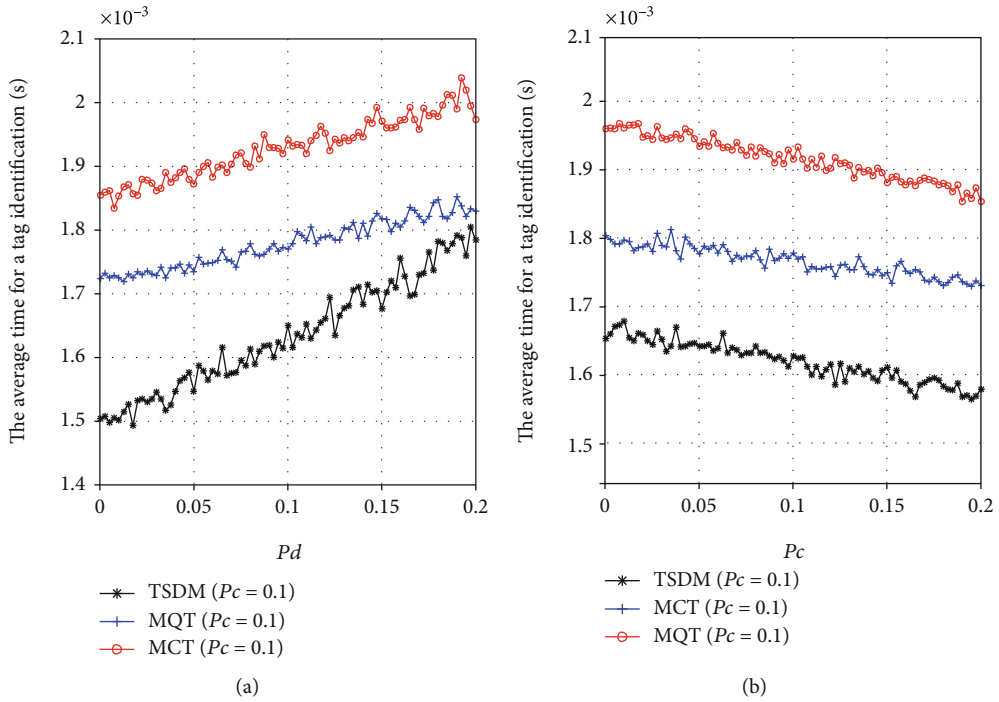

Figure 5: Average time cost for a tag identification in a realistic environment: (a) the value of Pc is 0.1, and Pd ranges from 0 to 0.2, and (b) the value of Pd is 0.1, and Pc ranges from 0 to 0.2.

TSDM algorithm still dominates the identification time when Pd changes dynamically.

Figure 5(b) shows the change rules of the average time to identify a tag due to Pc's change when $Pd = 0.1$. When the capture effect occurs, a collision slot may be inaccurately considered a success slot, which will speed up the tag identification process. As Pc increased, the average time for a tag identification is decreasing. Moreover, the identification time of TSDM is still less than MQT and MCT.

5. Conclusions

There is predetection in the TSDM algorithm that can completely avoid the idle slots of the M-ary algorithm, thus reduce the communication data between reader and tag. It will reduce the average communication load and the average time cost, which improves the TSDM algorithm's performance. The proposed TSDM algorithm has an average communication load of about 225 bits, an average time overhead of 1.51 ms, and average power consumption of 1.36 mJ when identifying a tag with an ID length of 96 bits. Compared with the other tree-based algorithms, the TSDM algorithm reduces the time cost by at least 12.57% and the energy cost by at least 12.65%. Moreover, taking environmental noise (including path loss and capture effect) into consideration, the TSDM algorithm's performance still outperforms the other algorithms. Therefore, the TSDM algorithm is worth researching.

Data Availability

Data available is on request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is sponsored by the Science and Technology Research Program of Chongqing Municipal Education Commission of China (No. KJQN202000703), Science and Technology Research Project of the Chongqing Municipal Education Commission of China (No. KJZD-201800701), and 2018 Reliable Control and Safety Maintenance of Dynamic System (No. JDDSTD2018001).

References

[1] J. Su, Z. Sheng, A. Liu, Z. Fu, and Y. Chen, "A time and energy saving based frame adjustment strategy (TES-FAS) tag identification algorithm for UHF RFID systems," *IEEE Transactions on Wireless Communications*, vol. 19, no. 5, pp. 2974–2986, 2020.

[2] J. Su, Z. Sheng, V. C. M. Leung, and Y. Chen, "Energy-efficient tag identification algorithms for RFID: survey, motivation and new design," *IEEE Wireless Communications*, vol. 26, no. 3, pp. 118–124, 2019.

[3] L. Arjona, H. Landaluce, A. Perallos, and E. Onieva, "Fast fuzzy anti-collision protocol for the RFID standard EPC Gen-2," *Electronics Letters*, vol. 52, no. 8, pp. 663–665, 2016.

[4] J. Su, R. Xu, S. Yu, B. Wang, and J. Wang, "Idle slots skipped mechanism based tag identification algorithm with enhanced collision detection," *KSII Transactions on Internet and Information Systems*, vol. 14, no. 5, pp. 2294–2309, 2020.

[5] L. Barletta, F. Borgonovo, and M. Cesana, "A formal proof of the Optimal frame setting for dynamic-frame Aloha with known population size," *IEEE Transactions on Information Theory*, vol. 60, no. 11, pp. 7221–7230, 2014.

[6] J. Su, A. X. Liu, Z. Sheng, and Y. Chen, "A partitioning approach to RFID identification," *IEEE/ACM Transactions on Networking*, vol. 28, no. 5, pp. 2160–2173, 2020.

[7] J. Su, Z. Sheng, L. Xie, G. Li, and A. X. Liu, "Fast splitting-based tag identification algorithm for anti-collision in UHF RFID system," *IEEE Transactions on Wireless Communications*, vol. 67, no. 3, pp. 2527–2538, 2019.

[8] L. Zhang, W. Xiang, and X. Tang, "An adaptive anti-collision protocol for large-scale RFID tag identification," *IEEE Wireless Communications Letters*, vol. 3, no. 6, pp. 601–604, 2014.

[9] Y.-C. Lai, L.-Y. Hsiao, H.-J. Chen, C.-N. Lai, and J.-W. Lin, "A novel query tree protocol with bit tracking in RFID tag identification," *IEEE Transactions on Mobile Computing*, vol. 12, no. 10, pp. 2063–2075, 2013.

[10] S. Jian, I. E. E. E. Zhengguo Sheng, A. X. Liu, H. Yu, and Y. Chen, "A group-based binary splitting algorithm for UHF RFID anti-collision systems," *IEEE Transactions on Wireless Communications*, vol. 68, no. 2, pp. 998–1012, 2020.

[11] J. Su, Z. Sheng, D. Hong, and G. Wen, "An effective frame breaking policy for dynamic framed slotted Aloha in RFID," *IEEE Communications Letters*, vol. 20, no. 4, pp. 692–695, 2016.

[12] A. Abbasian and M. Safkhani, "CNCAA: A new anti-collision algorithm using both collided and non-collided parts of information," *Computer Networks*, vol. 172, pp. 1286–1389, 2020.

[13] L. Zhang, J. Zhang, and X. Tang, "Assigned tree slotted Aloha RFID tag anti-collision protocols," *IEEE Transactions on Wireless Communications*, vol. 12, no. 11, pp. 5493–5505, 2013.

[14] H. Landaluce, A. Perallos, E. Onieva, L. Arjona, and L. Bengtsson, "An energy and identification time decreasing procedure for memoryless RFID tag anti-collision protocols," *IEEE Transactions on Wireless Communications*, vol. 15, no. 6, pp. 4234–4247, 2016.

[15] L. Zhang, W. Xiang, X. Tang, Q. Li, and Q. Yan, "A time- and energy-aware collision tree protocol for efficient large-scale RFID tag identification," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 6, pp. 2406–2417, 2018.

[16] H. Landaluce, A. Perallos, E. Onieva, L. Arjona, and L. Bengtsson, "An energy and identification time decreasing procedure for memoryless RFID tag anticollision protocols," *IEEE Transactions on Wireless Communications*, vol. 15, no. 6, pp. 4234–4247, 2016.

[17] X. Jia, Q. Feng, and C. Ma, "An efficient anti-collision protocol for RFID tag identification," *IEEE Communications Letters*, vol. 14, no. 11, pp. 1014–1016, 2010.

[18] Z. Zhou, Q. M. J. Wu, Y. Yang, and X. Sun, "Region-level visual consistency verification for large-scale partial-duplicate image search," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 16, no. 2, pp. 1–25, 2020.

[19] D. G. Zhang, G. Li, Z. H. Pan, and Y. P. Liang, "A new anti-collision algorithm for RFID tag," *International Journal of Communication Systems*, vol. 27, no. 11, pp. 3312–3322, 2014.

[20] L. Zhang and W. Xiang, "An energy- and time-efficient M-ary detecting tree RFID MAC protocol," in *2015 IEEE International Conference on Communications (ICC)*, pp. 2882–2887, London, UK, 2015.

[21] J. H. Choi, D. Lee, and H. Lee, "Query tree-based reservation for efficient RFID tag anti-collision," *IEEE Communications Letters*, vol. 11, no. 1, pp. 85–87, 2007.

[22] J. Shin, B. Jeon, and D. Yang, "Multiple RFID tags identification with M-ary query tree scheme," *IEEE Communications Letters*, vol. 17, no. 3, pp. 604–607, 2013.

[23] S.-C. Tsai, Y.-M. Hu, C.-H. Chai, and J.-S. Li, "Efficient tag reading protocol for large-scale RFID systems with pre-reading," *Computer Communications*, vol. 88, pp. 73–83, 2016.

[24] Z. Zhou, Y. Mu, and Q. M. J. Wu, "Coverless image steganography using partial-duplicate image retrieval," *Soft Computing*, vol. 23, no. 13, pp. 4927–4938, 2019.

[25] J. Su, Z. Sheng, G. Wen, and V. C. M. Leung, "A time efficient tag identification algorithm using dual prefix probe scheme (DPPS)," *IEEE Signal Processing Letters*, vol. 23, no. 3, pp. 386–389, 2016.

[26] M. Shahzad and A. X. Liu, "Probabilistic optimal tree hopping for RFID identification," *IEEE/ACM Transactions on Networking*, vol. 23, no. 3, pp. 796–809, 2015.

[27] L. Pan and H. Wu, "Smart trend-traversal protocol for RFID tag arbitration," *IEEE Transactions on Wireless Communications*, vol. 10, no. 11, pp. 3565–3569, 2011.

[28] X. Yang, X. Liu, and B. Wu, "Smart batch-processing protocol for RFID multiple tag identification," *IEEE Access*, vol. 8, no. 11, pp. 159132–159142, 2020.

[29] Y. Zhu, W. Jiang, Q. Zhang, and H. Guan, "Energy-efficient identification in large-scale RFID systems with handheld reader," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 5, 2014.