Novel Dynamic Time-slotted Binary Tree Anti-collision Algorithm for UHF RFID Systems

Zhi-jun Tang[1,2], Quan Liu[1], Qi-jian Tang[2], Long Li[2], Juan Liu[2], Xi-ping Xia[2],Jian Li[2]
[1] Hunan University of Science and Technology
[2] Hunan Software Vocational Institute
Xiangtan, China
zjtang@hnust.edu.cn

***Abstract*—In view of many inquiry times and lower throughput rate of existing binary tree anti-collision algorithms in UHF RFID reader, a novel dynamic time-slotted binary tree anti-collision (DTSB) algorithm is proposed by combining ALOHA algorithm with binary tree algorithm, the stack and lock bit strategy. The simulation results show the proposed algorithm is more efficient in paging numbers, transmission delay and communications traffic, which significantly improves the communication rate between the reader and the tag. Furthermore, the throughput rate of the proposed algorithm achieves the best optimization. By introducing the proposed algorithm, the collision problem of single reader communication with multiple tags can be effectively solved in UHF RFID systems.**

Keywords—RFID; Binary tree anti-collision algorithm; Stack; Lock bit; Performance improvement

I. INTRODUCTION

Radio frequency identification (RFID) is an automatic identification technology which is developed in the 1980s. By using space coupling of RF signals and non-contact information transmission, targets are identified according to the transferred information. As the key technology of the Internet of Things (IoT), RFID has already been widely used in industrial automation, transportation, logistics, food safety, medical treatment, etc. [1-2]. The applications of RFID at low frequency, intermediate frequency and high frequency are reaching maturity. The age of the Internet of Things is quietly approaching, just as the age of internet is integrated into our lives. However, a fly in the ointment is that the development of RFID technology at UHF and microwave bands is far from enough, which means current RFID identification products are mostly used for short-distance identification. Due to some problems such as inconsistent communication protocols, inefficient of anti-collision algorithm, and high price of tag products, long-range identification of RFID system has not yet to be mature. Thus, a fast, accurate and effective anti-collision problem solution plays a vital role in the development of RFID technology.

In wireless communication technology, time division multiple access (TDMA) is most widely used for anti-collision algorithm of tags in current RFID system. It can be sectioned into two types, non-deterministic algorithm and deterministic algorithm [3]. ALOHA algorithm is a typical non-deterministic algorithm, which has tag starvation problem. Binary searching algorithm is a typical deterministic algorithm. This kind of algorithm is more complex, and it will cost more time for identification. But no tag starvation problem occurs [4]. For the need of guarantee the accuracy of identification system, the binary tree anti-collision algorithm is a good choice [5-6]. Although deterministic algorithms based on binary tree have the ability to solve tag starvation problem, but there still remains some problems such as long identification cycle, many inquiry times, and low throughput rate [7]. To solve these problems, some methods are suggested in recent years. For example, Kim S.S et al. [8] proposed an enhanced slotted binary tree algorithm with intelligent separation, which generates the request prefix so that the only existing tags according to the response in the corresponding slot, and decreases the tag identification time by reducing the overall numbers of requests. Wu H.F et al. [9] utilized BTSA algorithm to propose three passive RFID tag anti-collision protocols. Lai Y.C et al. [10] proposed a tag anti-collision algorithm of the generalized binary tree protocol, which separates the identification process into several binary tree cycles to solve the problem caused by the capture effect. Choi J.H et al. [11] proposed protocol as 16-bit random numbers aided query tree algorithm (RN16QTA),which achieves less time consumption for the tag identification than the present tag anti-collision protocols implemented in EPC Class 0, Class 1, and Class 1 Gen. 2. However, the length of RN16 QTA is actually not enough in real environments to successfully identify tags. Yang C.N et al. [12] proposed an effective RN16QTA (ERN16QTA) to really solve the tag collision in tags identification. Jia X.L et al. [13] introduced an efficient anti-collision protocol named collision tree protocol(CT), which is only dependent on the numbers of tags to be identified and not influenced by the distribution of tag IDs and other factors, and the average performance of CT for one-tag identification converges to a constant. Huang X et al. [14]improved several problems of enhanced dynamic framed

slotted ALOHA algorithm which are mentioned in literature [15], for example, tags numbers estimation, frames numbers slots numbers processing. Sunil D et al. [16] proposed an orderly reading strategy for reader in the case of dense tags according to the inefficiency problem of dynamic ALOHA algorithm in processing plenty of tags.

In consideration of problems such as many inquiry times of the reader, long identification period, lower throughput rate and high communication traffic, a novel dynamic time-slotted binary tree anti-collision algorithm is proposed by combining the existing binary tree anti-collision algorithms in this paper. The rest of the paper is organized as follows. Section 2 introduces several kinds of binary anti-collision algorithms. Section 3 elaborates the workflow and main command of dynamic time-slotted binary tree anti-collision algorithm for UHF RFID systems, and the flow diagram identification process demonstration diagram as well. Section 4 analyzes the dynamic time-slotted binary tree anti-collision algorithm from four aspects: paging time, transmission delay, throughout rate and communication traffic. In Section 5, the simulation of dynamic time-slotted binary tree anti-collision algorithm for UHF RFID systems is covered, and the compare with other typical algorithms as well. Section 6 is the conclusion part of this paper.

II. The Related Work

Binary search (BS) algorithm is the prototype of all binary tree algorithms. Its basic principle is that after a tag is identified, then the next search begins with the top of the binary tree, until the search reaches to the leaves of the tree, and other tag is identified. In addition, data communication between the reader and tag is complete transmission every time. The average paging times, ***L***, depends on ***N***, the total number of tags within the scope of reader, that is to say

$$L(N) = \log_2 N + 1 \qquad (1)$$

This shows that although the binary search algorithm can accurately identify every tag, but paging times are increasing rapidly, and throughput rate of this system is decreasing as tag numbers grow within the range of the reader.

Compared to the traditional binary tree anti-collision algorithm, dynamic binary search (DBS) algorithm reduces the communication traffic of a reader. Each reader just sends a portion of information to tag for matching, and all tags still return to the complete sequence numbers.

Compared to the DBS algorithm, bit locking backoff (BLBO) algorithm introduces the principle of lock bit. As a result, the paging numbers of reader is reduced to a certain extent. In the DBS algorithm, a tag only response a part of the information to a reader every time, and communication traffic of the tag is reduced. Most importantly, it has changed the search mode of the traditional algorithm. Using the backoff principle in the BLBO algorithm, it is unnecessary to return to the root node of the tree that every reader identifies a tag, and it only need return to a collision node to page based on the original leaf nodes. Therefore, the paging numbers of reader are reduced significantly. Since the backoff principle greatly improves the performance of the binary tree algorithm, the improved binary tree algorithms have been applied to the principle.

III. Proposed Algorithm

From the previous analysis, although the lock bit and backoff ideas continue to produce and binary tree anti-collision algorithm is improved continuously. But the *paging numbers*, transmission delay, throughput rate and communication traffic of a reader are not yet reached the optimization so far. If there are *n* tags, the reader can recognize them by paging ***n*** times, only in this case, the transmission times, transmission delay and throughput rate are optimized maximally. In dynamic time-slotted binary tree anti-collision algorithm (DTSB) for UHF RFID, all tags are divided into two groups by collision bits, according to the particularity of binary sequence. The dynamic time-slotted strategy is introduced in the proposed algorithm. In an inquiry process of the reader, tag which serial numbers is 0 would respond directly when the reader sends signal 0. And tag which serial number is 1 would respond with delay in that case. As a result, the reader could be able to receive information from two groups of tags by using one paging. In order to deal with two sets of information respectively, the reader needs to put later received information into a pre-allocated stack. The first set of tags would be decoded first. And the reader would keep on grouping in case of collision happening, until a tag is identified. Then it would deal with information in stack by using the LIFO (Last in first out) principle of stack. In this grouping strategy, the serial numbers of tags would be divided more and more specific according to the collision bit of tags, which makes the identification of information in stack very easily. There is no need for the reader to broadcast messages, since it could identify a tag by extracting stack information once. The principle of its realization is shown in Fig.1. Variable *x* stands for a collision bit. The reader broadcasts signal 0, and it would receive two messages later. Signal 1 is put into a stack. Then the reader broadcasts signal 00, and receives two messages too. Signal 1 is put into the stack. But the signal 0 which was broadcasted by the reader before is included in these stored information. So actually, the stored information is signal 01. It can be judged that there is no collision during the identification process of this tag by decoding signal 0. Then signal 01 is extracted from the stack. Tag 010 and 011 can be identified in the same way. And information is extracted from the stack again. Tag 1 could be identified directly after decoding. As

previously analysis, in order to identify four tags, the reader only need to page for three times and lock bit page for one time. In other words, four tags can be identified by paging four times. The identification times are reduced greatly, which leads to the decrease of communication traffic and transmission delay.

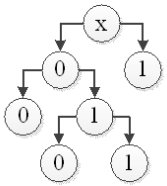


Fig.1. Stack storage identification schematic

The basic principle of dynamic time-slotted binary tree anti-collision algorithm for UHF RFID is described as follows: The reader locks collision bits by using bit locking principle. Then, there are two UID numbers in paging commands of the reader, UID0 and UID1. Tags would be compared with UID0 and UID1 according to certain rules. And data named ID0 and ID1 would be sent respectively in two time slots. The reader would analyze ID0 and ID1 one after another by using the stack [14]. The core of this algorithm is a combination of dynamic time-slotted, stack and bit locking method. By using dynamic time-slotted method, two sets of tags could respond to the reader at the same time. Stack method makes it possible for the reader to handle two sets of data from one paging process successively. And bit locking method could help tags to determine the collision bit by leaving useful information and shielding useless information, by which the communication traffic between tags and the reader is reduced.

A. Related Commands

To achieve this algorithm, a set of commands is required. It is processed by tags. In addition, each tag has a unique serial numbers. Specific commands are as follows:

REQUEST (1): Broadcast all 1 serial numbers;

SELECT (ID): Select the serial numbers of tags;

READ-DATA: Read the data;

UNSELECT: Deactivation command.

Some commands are needed to be added to this novel algorithm. These commands are: REQUEST (UID): Bit locking command. After the first time paging, UID is obtained by the reader from analysis of the result of decoding, and the collision bit sets 1, non-collision bit sets 0. This UID is broadcasted by reader. Tags in inquiry area would compare their serial numbers with the UID, load data from each bit which is set to 1 to compose a novel serial numbers, and store this novel serial numbers, use this serial numbers in this inquiry area until it is identified by the reader, then release this memory space to get ready for the next time of bit locking. At the same time, after the generation of this novel serial numbers, the tag whose first bit of novel serial numbers is 0 would send its novel serial numbers to the reader within the first time slot. And the tag whose first bit of novel serial numbers is 1 would send its novel serial numbers to the reader within the second time slot.

REQUEST(UID0,UID1): If the currently processed information is from ID0, then UID0 of last time would be combined with the highest collision bit of currently ID0 and data before it. Data which is combined with the highest collision bit 0 of ID0 is UID0, which is combined with the highest collision bit 1 of ID0 is UID1. Similarly, if the currently processed information is from ID1, then UID1 of last time would be combined with the highest collision bit of currently ID1 and data before it. Tags would compare its own serial numbers with UID 0 after receiving this command. How many bits UID0 are, how many bits tags would compare. If they are the same, then skip UID1, and send the rest serial numbers to the reader within the first time slot. If they are different, then tags would compare serial numbers with UID1. If its serial numbers is the same as UID1, then send the rest of serial numbers to the reader within the second time slot. The time slot is depended on the time needed to send message by tags. Since everytime the information sent by tags are gradually reduced, so everytime the time slot of tags responding are fluctuant. The reader would receive two signals in two time slots, set them as ID0 and ID1. ID0 code is from the response of tags toward UID0, ID1 code is from the response of tags toward UID1.

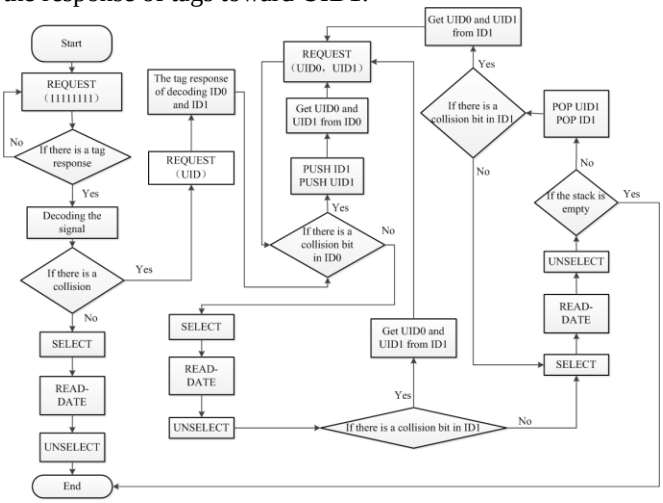


Figure 2. Flow diagram of the proposed DTSB algorithm

B. Algorithm Workflow

The flow diagram of dynamic time-slotted binary tree anti-collision algorithm is shown in Fig.2. The workflow of the proposed DTSB algorithm is described as follows:

(1) The reader sends a complete query condition Q for the first time. The length of Q is the numbers of binary bits of tags ID, ***L***. Each code of each bit is set to 1, so that each tag would return its own serial numbers. In other words, the reader broadcasts command REQUEST (1) to ask each tag in this area to return its own serial numbers.

(2) If there is no response, it indicates that there is no tag. If there is response but no collision, it indicates that there is

only one tag. The reader sends command SELECT (ID), and after the exchange is completed, sends command UNSELECT (ID), then the tag is entered deactivation state. If there is a collision, the reader decodes the obtained data. These data is set as ID. All existed collision bits can be determined according to the coding rules, the collision bit sets 1 and non-collision bit sets 0. And it forms a novel serial numbers UID. Then it goes to step (3).

(3) The reader broadcasts command REQUEST (UID). Every tag which has received this command locks its own serial numbers to form a novel serial and save it. Tag, whose first bit of this novel serial is 0, sends its serial numbers except first bit to the reader within the first time slot. Tag, whose first bit is 1, estimates the required time of sending message, and after a delay of sending time, sends its serial numbers except first bit to the reader within the second time slot. Then it goes to step (4).

(4) The serial numbers which is received by the reader in the first time slot is set as ID0, and the serial numbers which is received in the second time slot is set as ID1. ID0 is the first one to be decoded by the reader. If there is a collision, then ID1 and UID1 of the last time would be pushed into stack. If there is no valid value of UID1, it remains empty. Extract the bit before the highest collision bit of ID0, and add it to UID0 of the last time. Then add the highest collision bit. Set the highest collision bit to 0, which forms a novel UID0. If the highest collision bit is set to 1, then a novel UID1 is formed. Then it goes to step (5). If there is no collision in ID0, then the tag of this serial numbers is identified. This tag would be deactivated after the communication is completed. Then it goes to step (6).

(5) The reader broadcasts command REQUEST (UID0, UID1). Tags would compare their own serial numbers with UID0 of these broadcast data after receiving this command. If these corresponding former bits are equal, then skip information after them, and send its own serial numbers to the reader within the first time slot. If they are not equal, then compare them with UID1, and send its own serial numbers to the reader within the second time slot. Then it goes to step (4).

(6) Decode signal ID1, and judge that if there is any collision in ID1. If there is a collision, extract bits before the highest collision bit, and add them to UID1 of the last time. Then add it to the highest collision bit. If the highest collision bit is 0, it forms UID0. If the highest collision bit is 1, it forms UID1. Then it goes to step (5). If there is no collision in ID1, then the tag of this serial numbers is identified by the reader. And this tag would be deactivated after the identification is completed. Determine whether the stack of the reader is empty or not. If it is empty, then the identification process is completed. If it is not, then pull to obtain UID1 and ID1. And repeat step (6).

C. Identification Process of DTSB Algorithm

TIt is assumed that there are tag A, B and C in the inquiry area of the reader. Their UID are 10111010, 01101010, and 11001010 respectively. The identification process of dynamic time-slotted binary tree anti-collision algorithm is shown in Fig.3.

(1) The reader broadcasts signal REQUEST (11111111).

(2) The reader receives the return signal (xxxx1010) of tags, the collision bit is set to 1, and non-collision bit is set to 0. The reader broadcasts bit locking signal REQUEST (11110000).

(3) Each tag locks its own UID in accordance with the bit locking signal and shields non-collision bit. Tags which the highest collision bit is 0 return other locking bits except the highest collision bit within the first time slot. Tags which the highest collision bit is 1 return other locking bits except the highest collision bit within the second time slot.

(4) In the first time slot the reader receive message "110" from tags, and "xxx" is located in the second time slot. There is no collision in information of first time slot. So add "0" to "110", and "0110" is gained. Tag B is identified.

(5) There is a collision in the decoding of second time slot information by the reader, REQUEST (10, 11);

(6) According to received information, all unidentified tags send "11" within the first time slot, "00" within the second time slot.

(7) Tag A is identified. And tag C is identified by extracting information of second time slot.

(8) The stack is empty, and the identification process is completed.

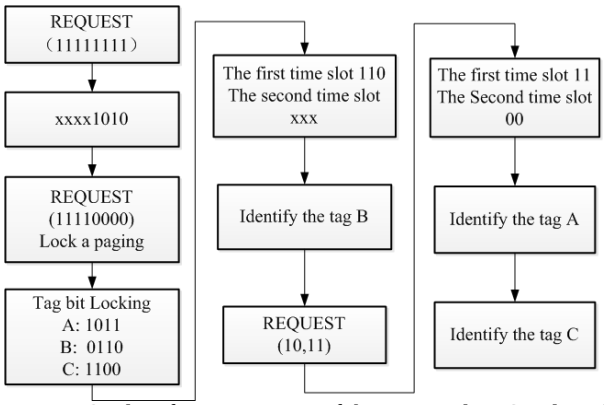


Figure 3. Identification process of the proposed DTSB algorithm

IV. PERFORMANCE ANALYSIS

A. Paging times of the reader

Let us suppose that there are n tags in working area of the reader. In order to identify those n tags, the reader has to paging $Q(n)$ times, and there are $C(n)$ times of collision during this identification process. So it can be obtained that:

$$Q(n)=C(n)+1 \quad (2)$$

$$C(n)=n-1 \quad (3)$$

$$Q(n)=n \quad (4)$$

The following part proves equation (2), (3) and (4).

Authentication: The degree of root nodes of binary tree which is generated by the proposed algorithm is 1. The numbers of all non-leaf nodes stands for paging times $Q(n)$.

The numbers of all non-leaf nodes except for root nodes (in other words, the numbers of nodes whose degree is 2) stands for the numbers of collisions C(*n*). So Q(*n*)=C(*n*)+1, that is to say equation (2) is true. The numbers of leaf nodes (the numbers of terminal nodes) is the numbers of tags ***n***. According to natures of binary tree, it can be seen that the numbers of terminal nodes equals 1 plus the numbers of nodes whose degree is 2. In other words, C(*n*)=n-1, that is, equation (3) is true. According to equation (1), (2), it can be obtained that Q(*n*)=n, i.e., the equation (4) holds.

If there is a single collision bit, the paging numbers minus two, then the paging numbers is calculated as follows:

$$\mathrm{Q(n) = n\text{-}2} \tag{5}$$

In case of a single collision comes forth *k* times, then the paging numbers of the reader is calculated as follows:

$$\mathrm{Q(n) = n\text{-}2k} \tag{6}$$

B. Transmission delay

Due to the uncertainties of a single collision, the transmission delay is not related to the processing and analysis of the case. In order to identify n tags, it requires that at least $\mathrm{Int}(\log_2 \mathrm{n})$ bits of tags ID are in different positions. Assuming there are *n* tags, each ID of each tag is at the length of *k* bits, and the numbers of collision bits is *x*, which is the numbers of collision bits determined by the reader decoding the first time received information which comes from those tags. And the range of value of ***x*** is $[\mathrm{Int}(\log_2 \mathrm{n}), \mathrm{k}]$.

In the proposed algorithm, the reader has five types of nodes. Type1 are root nodes, which is the information broadcast by the reader for the first time. Their length is *k* bits. The first time tags also returns *k* bits. So Type1 nodes need 2*k*+21 clock cycles. Type2 are the bit-locking information at a length of *k* which is sent by the reader during the second time of paging. Tags send 2(*x*-1) bits information within two time slots. So Type2 nodes need *k*+2(*x*-1)+21 clock cycles. Type3 nodes are all the collision nodes except Type2 nodes, and the numbers of them is *n*-2. Paging data transmitted by the reader are composed of UID1 and UID0. Responding data returned by tags are ID0 and ID1. Tags which respond UID0 respond at the first time slot. Tags which respond UID1 respond at the second time slot. The length of the first and second time slot is equal with the time everytime tags sending information to the reader. So it needs 2*x*+21 clock cycles. Type4 nodes are nodes with pushing or pulling operation of UID1 and ID1. In case of the numbers of tags is less than 3, there is no pushing or pulling operation. If the numbers of tags is equal or greater than 3, information of UID1 and ID1 need to be pushed *b* times. Variable *b* is related to the collision times of ID0. It costs four clock cycles for pushing and pulling. So this part would cost 4*b* clock cycles in total. Type5 are all leaf nodes, and their numbers is *n*. Leaf nodes mean that there is no collision during the communication, and tags are identified successfully. Two extra clock cycles are needed for the reader to register ID information of tags. So Type5 nodes need to go through 2*n* clock cycles. The total length of all bits which are needed to be transmitted between tags and the reader during the anti-collision operation process in the algorithm is *L*.

$$\begin{aligned} L &= (2k+21)+(k+2(x-1)+21) \\ &\quad +(2x+21)(n-2)+4b+2n \\ &= 3k+4b+(2x+23)n-2x-2 \end{aligned} \tag{7}$$

Parameter *b* is the sum of the processing time for pushing ID1 and UID1 in case of collision is happened in ID0 during the identification process of the reader and the processing time for pulling ID1 and UID1 next time. Four bit extra clock cycles are needed for the reader to conduct pulling and pushing operation. Pulling and Pushing is relative. It is enough for analyzing pushing times only. In case of there are collisions in all ID0 and no collision in ID1, the reader executes the most times of pushing. The collision nodes of the binary tree which are generated at that time only have lower sub-nodes under right sub-nodes. All left sub-nodes are leaf nodes, and there are no lower sub-nodes under left sub-nodes. Any other leaf node has to be pushed and pulled once except the deepest two. So the expression of $\mathrm{b}_{\max}$ is:

$$b_{\max} = n-2 \tag{8}$$

If there is no collision in all ID0 while there are collisions in all ID1, the reader does not have to push and pull. The value of $b_{\min}$ is 0. Combining equation (8), the range of ***b*** is:

$$\mathrm{b} \in (0,\ \mathrm{n\text{-}2}) \tag{9}$$

In summary, when *n*=1, the total bits needed to be transmitted by tags and the reader during the process of anti-collision operation of the algorithm, *L*=2*k*+23. Furthermore, transmission delay depends on the total traffic of the identification process of the reader and tags, stored information of the reader and transmission delay of response of tags. In the proposed algorithm, the amount of bits needed to identify *n* tags is *L*. Assuming that the transmission rate of data is *v* bit/s, so the transmission delay of binary searching tree is:

$$\tau = \frac{L}{v} = \frac{3k+4b+(2x+23)n-2x-23}{v},\ (\mathrm{n} \geq 2) \tag{10}$$

$$\tau = \frac{L}{v} = \frac{2k+23}{v}, (n=1) \tag{11}$$

C. Throughput rate and Communications traffic

According to equation (4), the throughput rate of DTSB algorithm is:

$$s_{DTSB} = \frac{n}{Q(n)} \times 100\% = \frac{n}{n} \times 100\% = 100\% \tag{12}$$

Reader communications traffic is mainly controlled by the paging numbers of a reader. If each paging data of the reader is the length ***L*** of a tag, then the reader communications traffic $S(n)$ is calculated as:

$$S(n) = L(n-2k) \tag{13}$$

For the proposed algorithm, each paging data of the reader only contains the known data bits plus a sign bit 0. In the case of a large numbers of tags, the longest paging data is the length ***L*** of the tag ID except for the broadcast signal, while the shortest paging data may be a bit of 0. Therefore, the reader can estimate as the communications traffic of a half time $S(n)$, the reader communications traffic $S^{'}(n)$ is described as:

$$S^{'}(n) = \frac{1}{2}L(n-2k) \tag{14}$$

V. SIMULATION AND ANALYSIS

Referring to ISO18000-6 standards, excluding controlling, prefix and suffix, redundancy verifying and other expenses, the proposed algorithm is simulated under ideal channel conditions (error-free) using the RFID anti-collision simulation platform developed by ourselves. Tags ID are evenly distributed, and the length is fixed at 64bit. The numbers of tags is dynamic changing from 0 to 90, and the bit rate is 100kbit/s. This paper analyzes four performance indicators, paging times of the reader, transmission delay, throughput rate, communication traffic, and then compares with BS algorithm, DBS algorithm and BLBO algorithm.

Figure 4 shows the comparison of paging times of the reader of DTSB, BS, DBS and BLBO algorithm. It can be seen from Fig.4 that the paging times of DTSB is significantly less than other three algorithms. And as the value of *n* increasing, this advantage is becoming more obvious. It can be concluded that the paging times of the reader of DTSB is less than BLBO, BS and DBS in the identification process of *n* tags. BLBO is less than BS and DBS, while BS is equals to DBS.

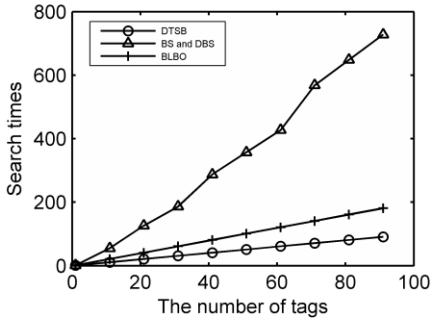


Figure 4. Paging times of the reader between DTSB, BS, DBS and BLBO

Under the condition of 100kbit/s transmission rate of data and 64bit length of tags ID, the relation among value of numbers of collision bits *x*, parameter *b*, tags numbers *n* and transmission delay of these four algorithms is shown in Fig.5. It can be seen from Fig.5 that no matter what value *x* is within the scope of $[Int(\log_2 n), k]$ and ***b*** within the scope of $[0, 2n-1]$, even if ***x*** and ***b*** is maximized, the delay of DTSB is less than BS and DBS. Although the delay of DTSB algorithm is less than BLBO, but the advantage is not so obvious, which is not unpredictable. Because these two algorithms are both using the bit locking strategy, only using the collision parts of serial numbers of tags, their traffic are equal. Therefore, there is no clearly difference between their transmission delays. As collision bits increasing, their transmission delay would become more similar. As a result, these following conclusions could be inferred: In terms of transmission delay, DTSB is minimal, and BLBO is less than DBS algorithm, while BS is maximal.

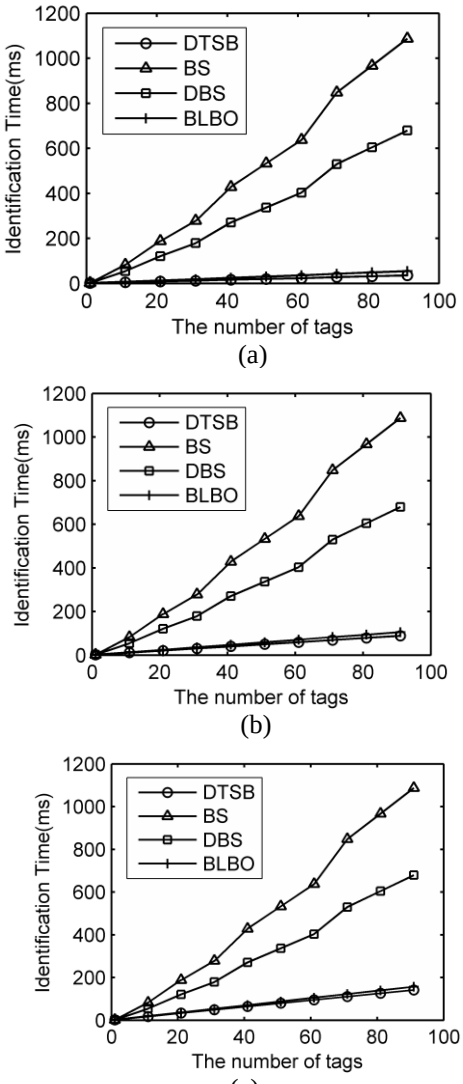
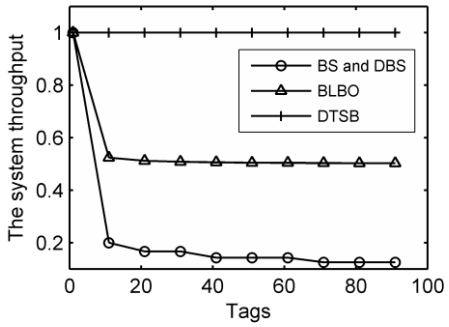


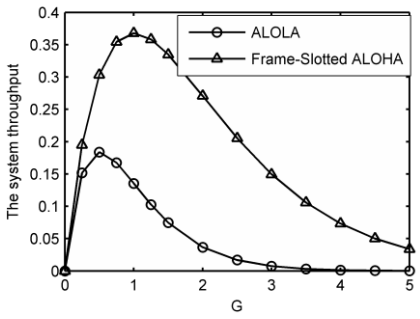
Figure 5. Comparison of transmission delay between DTSB, BS, DBS and BLBO: (a) $x = Int(\log_2 n), b = 0$;
(b) $x = (Int(\log_2 n) + k)/2, b = (n-2)/2$;
(c) $x = k, b = n-2$.

Figure 6(a) is the comparison of throughput rate between DTSB, BLBO, BS and DBS. It can be seen from this figure that the throughput rate of DTSB is 100%. It is obviously

better than other three algorithms. And it is fixed, and not related to tags numbers. However, the throughput rate of other three algorithms is decreasing as tags numbers increasing, and approaching a constant. The throughput rate of BLBO is approaching 50%, which is half of DTSB. It is the result of using time-slotted strategy and stack strategy in DTSB. Figure 6(b) shows the comparison of throughput rate between ALOHA algorithm and frame-slotted ALOHA algorithm. It can be seen that when the frame generation rate G=0.5, the throughput rate of ALOHA is maximum, about 18.4%. When G=1, the throughput rate of frame-slotted ALOHA is maximum, about 36.8%. It is means that ALOHA and its improved algorithm all have some limitations. To achieve the best performance of these algorithms, the frame generation rate is needed to be optimized. Figure 6(a) and 6(b) shows that the throughput rate of DTSB is obviously better than other algorithm. And its throughput rate is the best state of this system.



(a)



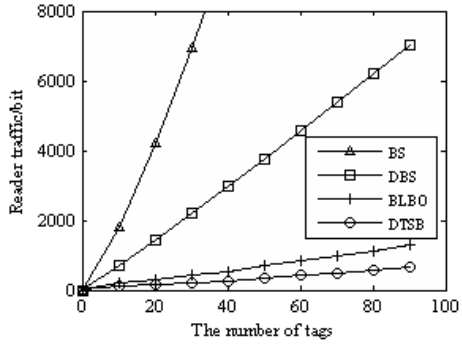
(b)

Figure 6. Comparison of throughput rate between DTSB and other algorithms: (a) DTSB and BLBO, BS, DBS; (b) ALOHA and frame-slotted ALOHA

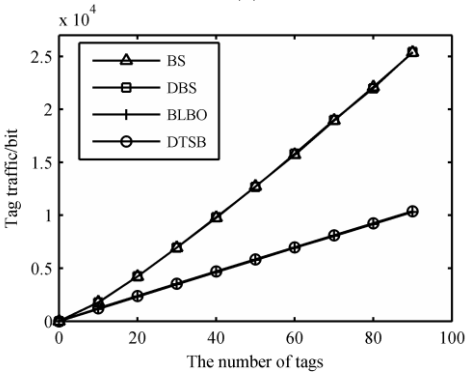
Figure 7 shows statistical analysis of communication traffic when the tag length is 64bit. Figure 7(a) shows that the comparison of reader communications traffic for DTSB, BLBO, DBS and BS algorithm. It is seen from Fig.7 (a) that DTSB has minimum value of reader communication traffic. Due to the response of tag packet, reader communication traffic of BLBO algorithm is two times of DTSB's. Because the reader paging numbers of BLBO is less than the DBS, and BLBO algorithm uses the lock-bit method, the communications traffic of BLBO algorithm is less than DBS algorithm. The amount of DBS algorithm is less than BS algorithm, due to the dynamic change of the paging instruction length for DBS algorithm. Overall, compared to other algorithms, the reader communication traffic of DTSB algorithm is reduced significantly.

The comparison of tag traffic is shown in Fig.7 (b). It is found that the tag traffic of BS algorithm is approximately equal to that of DBS algorithm, and BLBO algorithm is approximately equal to DTSB algorithm. This is mainly due to the dynamic response between the tag and the reader in the BLBO algorithm or DTSB algorithm. However, the DBS algorithm does not improve the tag response.

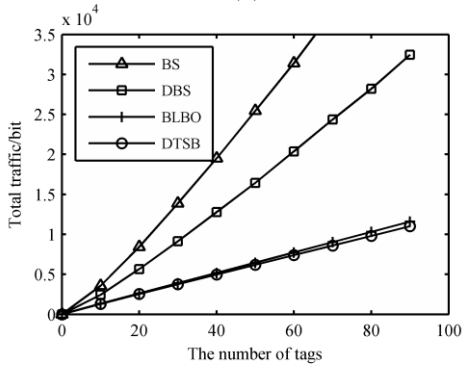
The comparison of total traffic is shown in Fig.7 (c). It can be seen that the DTSB algorithm is the least, and the BLBO algorithm is close to the DTSB algorithm. Compared with the DBS algorithm, the BLBO algorithm is reduced significantly. In addition, the DBS algorithm is about a half of the BS algorithm.



(a)



(b)



(c)

Figure 7. Statistical analysis of communication traffic: (a) reader communications traffic; (b)tag communications traffic;(c)total communications traffic.

In summary, the traffic of the tag is much greater than that of the reader. This effect is obvious in the BLBO algorithm and the DTSB algorithm. The simulation results of total traffic are similar with the simulation results of the transmission delay, and it can also prove the authenticity of the simulation data.

VI. Conclusion

In this paper, several existing typical RFID anti-collision algorithms are analyzed and compared. Their respective advantages and disadvantages are obtained. Based on this, a novel dynamic time-slotted binary tree anti-collision (DTSB) algorithm is proposed by the combination of dynamic time-slotted, stack, and bit locking. In the proposed DTSB algorithm, a bit locking command is sent by the reader. Tags would lock collision bits, divide them into two groups and return messages to the reader respectively within two time slots. After receiving these two messages, the reader saves the information of the later time slot in stack, and deals with the information of the former time slot. The result shows that there are obvious advantages of the proposed DTSB algorithm for UHF RFID in paging numbers of the reader, transmission delay, throughput rate and communication traffic over other binary tree algorithm. The proposed algorithm could reach the best performance of identification in the case of accurately, fast and long-range identifying required for the reader.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (Grant No.61875054), and by Science Research Fund Project of Hunan University of Science and Technology (Grant No.KJ1812).

References

[1] [Nikolaos K.., “An RFID Anti-collision Algorithm for Dense and Lively Environments,” IEEE Transactions on Communications, vol.60, no.2, pp.352-356,2012.

[2] Zhu L., and Yum T. P., “A critical Survey and Analysis of RFID Anti-collision Mechanisms,” IEEE Communi- cation Magazine, vol.49, no.5, pp.214-221, 2011.

[3] Amira S. E., Hania H.F., and El-Sayed E. B., “A New Proposed Anti-collision Algorithm for RFID,” the 30th National Radio Science Conference, National Telecommunication Institute, Egypt, pp.1-4, 2013.

[4] Kim S. C., Cho J. S., and Kim S.K., “Performance Improvement of Hybrid Tag Anti-collision Protocol for Radio Frequency Identification Systems,” International Journal of Communication Systems, vol.26, no.5, pp.705-719, 2013.

[5] Kim S. H., and Park P. G., “An Efficient Tree-based Tag Anti-collision Protocol for RFID Systems,” IEEE Communication Letters, vol.11, no.5, pp.449-451, 2007.

[6] Lai Y. C., and Hsiao L. Y., “General Binary Tree Protocol for Coping with the Capture Effect in RFID Tag Identification,” IEEE Communication Letters, vol.14, no.3, pp.208-210, 2010.

[7] Cui Y. H., and Zhao Y. P., “Performance Evaluation of A Multi-branch Tree Algorithm in RFID”, IEEE Transaction on Communications, vol.58, no.5, pp.1356- 1364, 2010.

[8] Kim S. S., Kim Y. H., and Ahn K., “An Enhanced Slotted Binary Tree Algorithm with Intelligent Separation in RFID Systems,” Proceedings of IEEE Symposium on Computer and Communications, Sousse, Tunisia, pp. 237-242, 2009.

[9] Wu H. F., Zeng Y., and Feng J. H., “Binary Tree Slotted ALOHA for Passive RFID Tag Anti-collision,” IEEE Transaction on Parallel and Distribution Systems, vol.24, no.1, pp.19-31, 2013.

[10] Lai Y. C., and Hsiao L. Y., “General Binary Tree Protocol for Coping with the Capture Effect in RFID Tag Identification,” IEEE communication letters, vol.14, no.3, pp.208-210, 2010.

[11] Choi J. H., Lee D., and Lee H., “Query Tree-based Reservation for Efficient RFID Tag anti-collision,” IEEE Communication Letters, vol.11, no.1, pp.85-87, 2007.

[12] Yang C. N., and He J. Y., “An Effective 16-bit Random Numbers Aided Query Tree Algorithm for RFID Tag Anti-collision,” IEEE Communication Letters, vol.15, no.5, pp.539-541, 2011.

[13] Jia X. L., Feng Q Y., and Yu L. S., “Stability Analysis of Efficient Anti-collision Protocol for RFID Tag Identification,” IEEE Transaction on Communications, vol.60, no.8, pp.2285-2294, 2012.

[14] Huang X., “An Improved ALOHA Algorithm for RFID Tag Identification,” Knowledge-based Intelligent Information and Engineering Systems Lecture Notes in Computer Science, vol.42, no.3, pp.1157-1162, 2006.

[15] Lee S. R., Joo S.D., and Lee C.W., “An Enhanced Dynamic Framed Slotted ALOHA Algorithm for RFID Tag Identification,” Proc. of the 2nd Annual International Conference on Mobile and Ubiquitous Systems, San Diego, California, USA, pp.17-21, 2005.

[16] Dhakal S., and Shin S.A., “Sequential Reading Strategy to Improve the Performance of RFID Anti-collision Algorithm in Dense Tag Environments,” The Fifth International Conference on Ubiquitous and Future Networks, Da Nang, Vietnam, pp.531-536, 2013.