A Slot Allocated Blocking Anti-Collision Algorithm for RFID Tag Identification

Yang Qing[1], Li Jiancheng[1], Wang Hongyi[1], Zeng Xianghua[1] and Zheng Liming[1]
[1] College of Electronic Science and Engineering, National University of Defense Technology, Changsha 410073, China
[e-mail: yq25171111@126.com (Y.Q.); lijc_hh@126.com (L.J.); wanghongyi2011@163.com (W.H.); 16047868@qq.com (Z.X.); 48287298@qq.com (Z.L.).]
*Corresponding author: Li Jiancheng

Received October 15, 2014; revised February 10, 2015; accepted May 12, 2015; published June 30, 2015

Abstract

In many Radio Frequency Identification (RFID) applications, the reader recognizes the tags within its scope repeatedly. For these applications, some algorithms such as the adaptive query splitting algorithm (AQS) and the novel semi-blocking AQS (SBA) were proposed. In these algorithms, a staying tag retransmits its ID to the reader to be identified, even though the ID of the tag is stored in the reader's memory. When the length of tag ID is long, the reader consumes a long time to identify the staying tags. To overcome this deficiency, we propose a slot allocated blocking anti-collision algorithm (SABA). In SABA, the reader assigns a unique slot to each tag in its range by using a slot allocation mechanism. Based on the allocated slot, each staying tag only replies a short data to the reader in the identification process. As a result, the amount of data transmitted by the staying tags is reduced greatly and the identification rate of the reader is improved effectively. The identification rate and the data amount transmitted by tags of SABA are analyzed theoretically and verified by various simulations. The simulation and analysis results show that the performance of SABA is superior to the existing algorithms significantly.

Keywords: RFID, anti-collision, blocking, slot allocation, identification rate

http://dx.doi.org/10.3837/tiis.2015.06.011 **ISSN : 1976-7277**

1. Introduction

Radio Frequency Identification (RFID) is a wireless communication technology [1-3]. The traditional bar code technology will be replaced by the RFID technology. The main components of the RFID system are a reader and tags. In the RFID systems, when multiple tags transmit data to the reader simultaneously, the collisions between these tags will result in that the reader cannot identify the tags correctly. The collision tags need to retransmit their IDs to the reader, which will increase the identification delay of the RFID system. Therefore, effective anti-collision algorithms must be proposed to solve the problem of collision.

The anti-collision algorithms in the RFID system can be classified into aloha-based [2] and tree-based [3]. In the aloha-based algorithms, a tag may not be identified by the reader for a long time which is the so-called tag starvation problem. Tree-based algorithms do not have the tag starvation problem. The principle of the tree-based algorithms is that the reader continuous divides the tags into two subsets until a subset only contains one tag. Tree-based algorithms can be further classified into binary tree (BT) based [4-8] and query tree (QT) based [9-12]. The QT based algorithms are much simpler than the BT based algorithms, and we mainly focus on the QT based algorithms in this paper.

In many RFID applications, the reader needs to identify the tags in its scope repeatedly. For example, in a supermarket, some goods may be taken away or back by consumers, the manager needs to know the real-time quantities of different things. For another instance, in an exhibition, the organizers want to get the real-time number of visitors in each showroom. For these applications, some algorithms such as the adaptive query splitting algorithm (AQS) [11] and the novel semi-blocking AQS (SBA) [12] were proposed. In these methods, all staying tags retransmit their IDs to the reader to be identified. When the length of tag ID is long, the retransmitting of the IDs will lengthen the identification delay of the reader. However, when the reader recognizes the tags in its range repeatedly, the IDs of all the staying tags are stored in its memory. The reader can verify the existence of the staying tags to realize recognition.

In this paper, we propose a slot allocated blocking anti-collision algorithm (SABA) to identify the staying tags rapidly. In SABA, the reader allocates a unique slot which can be taken as temporary ID (TID) for each tag. The length of the TID is much shorter than the length of tag ID. A staying tag only replies its TID to the reader to be identified. Thus, the identification rate of the reader is improved significantly. We analyze the identification rate and the amount of bits transmitted by tags of SABA. We also compare SABA with existing methods through various simulations. The simulation and analytic results show that SABA outperforms the existing algorithms in many situations.

The rest of the paper is organized as follows: Section 2 describes the link-timing of the RFID system. Section 3 describes the related works and their limitations. The proposed SABA is described in Section 4. Section 5 is the analysis of SABA. Simulations and discussions are contained in Section 6. Finally, Section 7 summarizes the full text.

2. The Link-Timing of the RFID System

Here, we define two terms which are round and slot. A round is the time from a reader begins to identify the tags in its scope to the reader recognizes all the tags. The *i*-th round can be written as T_i. A slot is the time from a reader begins to send a command to the ending of the following tags' response. A round consists of a few or many slots. According to the number of

tags that reply in a slot, the slot can be collision, success or idle which means more than one tags, only one tag or no tag are contained in the slot, respectively. The durations of the three types of slot are different, as shown in **Fig. 1**.

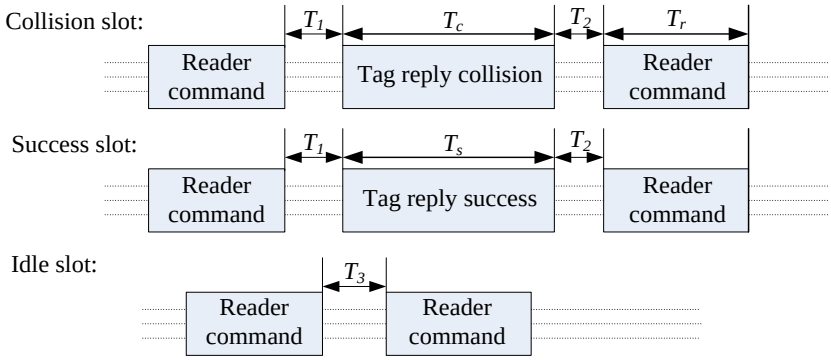


Fig. 1. The link-timing of the RFID system

In **Fig. 1**, T_1 means the time from the end of a reader command to the start of the following tag response. T_2 means the time from the end of a tag response to the start of the next reader command. T_3 means the time between the reader's two commands when no tag replies. T_r is the duration of a reader's command. T_s is the duration of a successful response of a tag. T_c is the duration of a collision response of tags. In QT, T_s and T_c are equal. According to the link-timing in **Fig. 1**, the identification rate $f(n)$ of a reader can be expressed as:

$$f(n) = \frac{n}{N_1T_1+N_2T_2+N_3T_3+N_cT_c+N_sT_s+\sum_{i=1}^{N_r} T_{ri}} \tag{1}$$

where, n means that the number of tags within the scope of the reader. N_1, N_2, N_3, N_c and N_s are the numbers of T_1, T_2, T_3, T_c and T_s, respectively. In QT-based algorithms, the length of a reader command will change with the depth of the query tree. N_r represents the total number of reader commands. The total time of all the reader commands is $\sum_{i=1}^{N_r} T_{ri}$, where T_{ri} is the duration of the i-th reader command.

In the RFID system, the length of tag ID may be very long. According to the international standard 18000-6C [1], the tag ID length can be up to 496 bits. In this case, the tags will spend a long time in transmitting their IDs. If we can reduce the data amount transmitted by tags in the identification process, the identification rate of a reader will be improved.

3. Related Works and Their Limitations

3.1 The Query Tree Algorithm (QT)

In QT [10], the reader maintains a tag ID prefix set Q which is a last in first out (LIFO) memory. The initial values of Q are {0, 1}. At the beginning of a round, the reader sends a query command with a parameter q which is popped from Q. When receiving the command, the tag whose ID matches with q will reply its ID to the reader. If the tags' responses are collision, then the reader stores two prefixes which are $q0$ and $q1$ to Q. If the tag's response is success or idle, the reader checks whether Q is empty or not, if yes, then the current round is complete, and if not, the reader transmits another query command with a parameter q which is popped from Q. This process is continued until the set Q is empty.

In the next round, the reader in QT adopts the same operations to recognize all the tags in its scope. If many tags stay in its range, the reader cannot identify the staying tags rapidly.

3.2 The Adaptive Query Splitting Algorithm (AQS)

AQS [11] is improved from QT to prevent the collisions of staying tags. In AQS, the reader stores two stacks which are Q and CQ. The definition of Q is the same as in QT. The stack CQ stores the prefixes of all the success and idle slots in the previous round. In the beginning of the current round, the reader copies all the elements from CQ to Q, and then it identifies all the tags as the same operations of QT. Thus, the collisions of staying tags can be prevented and all the arriving tags can also be recognized by the reader in the current round.

AQS can avoid collisions of staying tags, but it still has two shortcomings. First, AQS cannot prevent the collisions between the staying tags and arriving tags. Second, all the staying tags retransmit their IDs to the reader in the current round.

3.3 The Novel Semi-Blocking AQS (SBA)

A SBA [12] reader stores two stacks which are Q and CQ. The definition of Q is the same as in AQS. The stack CQ only stores the prefixes of all the success slots in the previous round. The identification process of SBA can be divided into two phases. All the staying tags and a small amount of arriving tags are recognized in the first phase. The rest of arriving tags are identified in the second phase. The procedure of SBA can be described as follows. In the first phase, the reader sends the *first-phase* command with P, where P is the probability that the arriving tags reply in the first phase. When receiving the command, each staying tag sets its *isResponsible* to 1 and each arriving tag generates a random probability P_t from 0 to 1. If an arriving tag's P_t is less than P, the arriving tag sets its *isResponsible* to 1. The arriving tags with $P_t \geq P$ set their *isResponsible* to 0. Then the reader recognizes all the tags whose *isResponsible* is 1. At the end of the first phase, the reader can identify all staying tags and a few arriving tags. Based on the number of arriving tags replied in the first phase and the probability P, the reader can estimate the number of the rest arriving tags which is *NewEst*. According to the estimation value, the reader generates *NewEst* prefixes and stores them to Q. In the second phase, the reader sends the *second-phase* command. When receiving the command, the tags which are identified by the reader set their *isResponsible* to 0 and the other tags set their *isResponsible* to 1. Then the reader identifies the rest of arriving tags as the same operations of QT.

In SBA, all the staying tags retransmit their IDs to the reader to be identified even though the IDs of these tags are stored in the reader's memory. When the length of tag ID is long, the reader needs a long time to recognize the staying tags.

3.4 Other Related Works

The couple-resolution blocking protocol (CRB) and the enhanced couple-resolution blocking protocol (ECRB) [13] are two other blocking methods which can be used to solve the problem considered in this paper. In CRB and ECRB, staying tags and arriving tags reply to the reader in different phases. The reader can identify two staying tags in a slot in the two algorithms. However, when the communication channel between the reader and tags is not ideal, the reader cannot distinguish a receiving error caused by the tag collision from the noise interference. Thus, CRB and ECRB cannot be used in an error-prone environment. Besides the QT-based algorithms mentioned above, some BT-based method [6, 8, 14] were also proposed to solve the tag re-identify problem in this paper. However, all these methods still require staying tags to retransmit their IDs to the reader.

In aloha-based protocols, there are also some works on improving identification efficiency by avoiding re-identifying staying tags [15-16]. Liu *et al.* [15] proposed three algorithms which were the basic unknown tag identification protocol (BUIP), the single-pairing unknown tag identification protocol (SUIP) and the multi-pairing unknown tag identification protocol (MUIP). In BUIP, SUIP and MUIP, staying tags and arriving tags are called known tags and unknown tags, respectively. As the reader has stored all known tags' IDs, through a specific hash function and random seed, the reader can predict which slot a known tag will reply. In these protocols, each staying tag only replies a short data to the reader to be identified. Thus, the data amount transmitted by tags is reduced and the efficiency of the reader is improved. However, known tags and unknown tags may collide with each other in BUIP, SUIP and MUIP. As a result, the collisions will increase the identification delay of the reader when identifying known tags. To recognize arriving tags effectively, the filtering-based unknown tag identification (FUTI) protocol and the interactive filtering-based unknown tag identification (IFUTI) protocol were proposed in [16]. In FUTI and IFUTI, the reader first labels unknown tags with an accuracy of at least α, then the reader adopts EDFSA algorithm [17] to identify the labeled tags. FUTI and IFUTI are blocking methods, in which the collisions between known tags and unknown tags are prevented. The characteristic of FUTI and IFUTI is that the reader only identifies unknown tags with a given accuracy α, which means that the two methods cannot guarantee complete recognition of unknown tags.

The polling problem of tags is also an important aspect of RFID applications. There are many existing researches [18-20] which focus on the polling problem in RFID systems. However, the polling problem is essentially different from the problem considered in this paper. The reasons are as follows. First, in the polling protocols for tag identification, the tags covered by the reader are stable for a long time, which means no tag leaves and no tag arrives, such as [18] which studies the missing-tag problem of the RFID system. The intent of [19] is to design efficient protocols to collect sensor-produced information from tags. The work of [20] studies how to design efficient protocols to collect information from a part of tags in a large RFID system. Second, the methods [18-20] were proposed to identify active tags, thus, the energy consumption and the execution time are two important factors considered by these protocols. However, the proposed SABA is mainly intended to identify passive tags.

4. The Proposed Algorithm

We propose a slot allocated blocking anti-collision algorithm (SABA) to reduce the data amount transmitted by staying tags. In SABA, a slot allocation mechanism is adopted and each staying tag only replies a short data to the reader to be identified. To make SABA work effectively, two assumptions are adopted as follows. 1) The tag is writable and is able to keep data even if it loses power for a short while. 2) In each round, the number of leaving tags is not too large compared to the number of staying tags.

A grouping method can be used to improve the efficiency of an anti-collision algorithm [5]. We divide n arriving tags into n groups. We use the following method to estimate the number of arriving tags:

$$newest_{i+1} = (1 - z) \cdot newcount_i + z \cdot newest_i \tag{2}$$

where $newcount_i$ and $newest_i$ are the real and estimation number of arriving tags in the previous round, while 1-z and z are weight coefficients.

4.1 Slot Allocation Mechanism

When identifying the arriving tags, the SABA reader allocates a unique slot which can be taken as temporary ID (TID) to each arriving tag. A SABA tag maintains a parameter to store its TID. If an arriving tag stays in the scope of a reader in the next round, the reader can identify the tag by using the TID. When identifying the arriving tags, besides the prefix of tag ID, the reader should also take an allocated slot as one of the parameters of its query command. When receiving a query command from the reader, the arriving tag whose ID matches with the prefix will reply its ID to the reader, and then it sets its TID as the allocated slot. The SABA reader stores a 2-dimensional (2D) table QS. The first line of QS is TIDs of tags and the second line contains the IDs of tags. In the process of identification, if the reader receives an arriving tag's reply correctly, the reader stores the allocated slot and the arriving tag's ID to QS. In addition to QS, the SABA reader also maintains a stack QC. When a tag leaves from the range of its associated reader, the reader deletes the TID and ID of the tag from QS and stores the TID of the tag to QC. A SABA reader should allocate the slots (TIDs) in QC firstly to arriving tags. If QC is empty, the reader assigns slots to arriving tags incrementally.

4.2 Identification of Staying Tags

When a reader recognizes the tags in its scope repeatedly, the reader has stored the IDs of the staying tags in the previous round. Thus, the reader can verify the existence of the staying tags rather than let them return their whole IDs. When the tag ID is long, the transmission of tag ID means more time consumption. Based on the slot allocation mechanism, each staying tag stores a unique TID. The reader can send a queryrep command with an allocated slot. After receiving the command, the staying tag, whose TID matches with the command's parameter, replies its TID to the reader. When the reader receives the response, it determines that the staying tag which associates with the allocated slot is still in its range. From the identification process of staying tags, we can get that the length of data each staying tag replies to the reader is short. The amount of data transmitted by staying tags is reduced significantly. If no tag replies to the reader's queryrep command, the reader can get that the tag leaves, and then the reader deletes the TID and ID of the staying tag from QS and stores the TID to QC.

4.3 The Procedure of SABA

In SABA, the ID of a reader is rRID and each staying tag stores a tRID which is the ID of its associated reader. When a tag leaves from the reader's range, the tag resets its tRID to 0. The pseudo-code of SABA is shown in **Fig. 2**. **Fig. 2**(a) and (b) are operations of the reader and the tag, respectively.

At the start of the current round, the reader sets $newcount_{i+1}$ to 0 and estimates the number of arriving tags, as shown in lines 1-2 of **Fig. 2**(a). The identification process of the reader can be divided into two phases. The first phase identifies the staying tags as shown in lines 5-13 of **Fig. 2**(a). The number of slots that a reader consumes to identify the staying tags is the number of elements in QS, i.e. length(QS) which is stored in the parameter length_QS as shown in line 3 of **Fig. 2**(a). QS is copied to a table QS_temp before the first phase as shown in line 4 of **Fig. 2**(a). In order to identify a staying tag, the reader sends a queryrep command with rRID and Slot_S, where Slot_S is the TID of the staying tag that the reader wants to identify. When receiving the command, each tag compares its tRID to rRID, and the tag takes itself as a staying tag if the result is match, the tag is an arriving tag if the result is not match. Each staying tag compares its TID to Slot_S. If the result is match, the tag replies its TID to the reader as shown in line 4 of **Fig. 2**(b). The other tags do not reply to the queryrep command. In

the first phase, if no staying tag replies to the reader's queryrep command, which means that the staying tag leaves, the reader stores the Slot_S to QC, deletes the TID which equals Slot_S and ID of the staying tag from QS, as shown in lines 10-11 of **Fig. 2**(a).

```
/* Reader operations */
1.  newcount_{i+1} =0
2.  newest_{i+1} = (1-z)*newcount_i + z*newest_i
3.  length_QS = length(QS)
4.  QS_temp = QS
5.  for i = 1:length_QS
6.      Set Slot_S as the i-th TID of QS_temp
7.      Transmit a queryrep with rRID and Slot_S
8.      Receive tag responses and check results
9.      if no tag replies then
10.         Push(QC,Slot_S)
11.         Delete the TID and ID from QS
12.     end if
13. end for
14. QC = SetOrder(QC)
15. Slot_A = NextAllocatedSlot()
16. Q = QueryInsertion(newest_{i+1})
17. while Q != NULL do
18.     q = pop(Q)
19.     Transmit a query with rRID, Slot_A and q
20.     Receive tag responses and check results
21.     if tag collision then
22.         push(Q,q0)
23.         push(Q,q1)
24.     else
25.         if only one tag response then
26.             Store the Slot_A and ID to QS
27.             Slot_A = NextAllocatedSlot()
28.             newcount_{i+1} = newcount_{i+1} + 1
29.         end if
30.     end if
31. end while
32. Transmit a terminating with rRID
```

(a)

```
/* Tag operations */
1.  Receive a command m from the reader
2.  if m is a queryrep with rRID and Slot_S then
3.      if tRID==rRID and TID==Slot_S then
4.          Transmit TID
5.      end if
6.  end if
7.  if m is a query with rRID, Slot_A and q then
8.      if tRID!=rRID and prefix(ID) == q then
9.          Transmit ID
10.         TID = Slot_A
11.     end if
12. end if
13. if m is a terminating with rRID then
14.     tRID = rRID
15. end if
```

(b)

Fig. 2. The pseudo-code of SABA

After the reader recognizes all the staying tags, the second phase starts. When assigning slots to the arriving tags, the reader should follow the order from small to large, so the reader sorts all the elements of QC into an ascending order, as the SetOrder() function in line 14 of **Fig. 2**(a). Before identifying arriving tags, the reader determines the value of the next allocation slot which is Slot_A as shown in line 15 of **Fig. 2**(a). The value of Slot_A is determined by the function NextAllocatedSlot(). The operations of the function are as follows: The reader first checks whether QC is empty or not, if QC is not empty, then the reader pops an element from QC to Slot_A. If QC is empty, the reader judges whether QS is empty or not; if QS is not empty, the reader sets Slot_A to the maximal value of TIDs in QS adds 1, i.e. max[QS(TID)]+1; if QS is empty, the reader sets Slot_A to 1.

When identifying the arriving tags, the reader uses the estimated value $newest_{i+1}$ and QueryInsertion() function to generate $newest_{i+1}$ prefixes and stores the prefixes to Q, i.e. Q=QueryInsertion($newest_{i+1}$). If the value of $newest_{i+1}$ is 0, then QueryInsertion(0) returns {0,

1}. In order to identify the arriving tags, the reader sends a query command with rRID, Slot_A and q, where q is popped from Q. All the staying tags do not reply to the query command. If the ID of an arriving tag (tRID!=rRID) matches with q, the arriving tag replies its ID to the reader and sets its TID to Slot_A as shown in lines 9-10 of **Fig. 2**(b). In the identification process of arriving tags, if the reader detects that the responses of the arriving tags are collision, it takes the same manner as QT to solve the collision, i.e. the reader increases the current prefix q to {q0, q1} and saves them to Q, as shown in lines 22-23 of **Fig. 2**(a). If the reader receives the response of an arriving tag correctly, it stores the TID (Slot_A) and ID of the arriving tag to QS, generates a new Slot_A and adds $newcount_{i+1}$ by 1 as shown in lines 26-28 of **Fig. 2**(a). When Q becomes empty, indicating that all the arriving tags have been identified, the reader sends a terminating command with rRID. After receiving the terminating command, all the tags set their tRID to rRID, as shown in line 14 of **Fig. 2**(b).

4.4 An Example of SABA

Fig. 3 shows a specific example of SABA. **Fig. 3**(a) shows the query tree of T_{i+1}. Assume that 4 tags which are tag A, B, C and D are in the scope of the reader in round T_i. The TIDs of these tags are TID(A)=1, TID(B)=2, TID(C)=3 and TID(D)=4, respectively. The tRIDs of these tags equal rRID. In round T_{i+1}, tag C leaves, tag E and F arrive. The tRIDs of tag E and F are 0. Assume that IDs of tag E and F are 0101 and 1010, respectively.

Before the reader sends commands, it estimates the number of arriving tags. We set the weight z to 0.5. The reader can get the estimated number of arriving tags $newest_{i+1}$=0.5*4+0.5*0=2. Thus, the reader generates two prefixes {0, 1} and stores them to Q. In the following, we describe the procedure of T_{i+1} slot by slot as shown in **Fig. 3**(b).

Slot 1: The reader sends a queryrep command with rRID and 1. After receiving the command, as the tRID and TID of tag A equal rRID and 1, respectively, tag A returns its TID to the reader. The other tags do not respond to the reader. After the reader receives the response of tag A, it determines that the tag is still within its scope.

Slot 2: The reader transmits a queryrep command with rRID and 2. Upon receiving the command, tag B replies its TID to the reader. The other tags do not reply to the reader. After the reader receives the response of tag B, it can also determine that the tag is still within its range.

Slot 3: The reader sends a queryrep command with rRID and 3. Since tag C leaves, no tag replies to the reader after receiving the command. As the reader cannot detect response from the staying tag, it determines that the tag leaves. Then, the reader removes the TID which equals 3 and tag C'ID from QS and saves 3 to QC.

Slot 4: The reader sends a queryrep command with rRID and 4. Upon receiving the command, tag D replies its TID to the reader. The other tags do not reply to the reader. After the reader receives the response of tag D, it determines that the tag is still within its range.

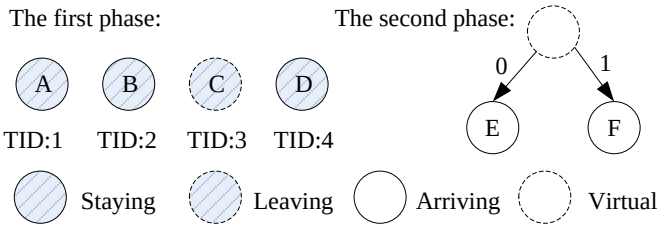
Slot 5: The reader sends a query command with rRID, 3 and "0" to identify arriving tags. After receiving this command, tag A, B and D do not reply. Because the tRID of tag E is not equal to rRID and the ID of tag E matches with "0" (the prefix in the reader's command), tag E replies its ID to the reader. The ID of tag F does not match with "0", so tag F does not reply to the reader. After sending its ID, tag E sets its TID to 3 (the slot in the reader's command). When receiving the ID of tag E, the reader stores 3 and tag E'ID to QS.

Slot 6: The reader sends a query command with rRID, 5 and "1". After receiving this command, tag F returns its ID to the reader and sets its TID to 5. The other tags do not respond. When receiving the ID of tag F, the reader stores 5 and tag F'ID to QS.

Slot 7: As Q is empty, the reader sends a terminating command with rRID. After receiving

the command, tag A, B and D remain their tRIDs unchanged, tag E and F set their tRIDs to rRID.

(**a**) The query tree in T_{i+1}



(**b**) Slot by slot in T_{i+1}

Slot	Reader command	Tag responses	QS	QC	Q
1	Queryrep with rRID and 1	TID(A)	1 ID(A) | 2 ID(B) | 3 ID(C) | 4 ID(D)	Null	0,1
2	Queryrep with rRID and 2	TID(B)	1 ID(A) | 2 ID(B) | 3 ID(C) | 4 ID(D)	Null	0,1
3	Queryrep with rRID and 3	-	1 ID(A) | 2 ID(B) | 4 ID(D)	3	0,1
4	Queryrep with rRID and 4	TID(D)	1 ID(A) | 2 ID(B) | 4 ID(D)	3	0,1
5	Query with rRID, 3 and “0”	ID(E)	1 ID(A) | 2 ID(B) | 3 ID(E) | 4 ID(D)	Null	1
6	Query with rRID, 5 and “1”	ID(F)	1 ID(A) | 2 ID(B) | 3 ID(E) | 4 ID(D) | 5 ID(F)	Null	Null
7	Terminating with rRID	-	1 ID(A) | 2 ID(B) | 3 ID(E) | 4 ID(D) | 5 ID(F)	Null	Null

Fig. 3. An example of SABA

5. Performance Analyses

In this section, we analyze the identification rate of SABA in T_{i+1} after the completion of T_i. The number of tags in T_i is n. In T_{i+1}, the number of arriving tags and leaving tags are α and β, respectively. According to the link-timing in Section 2, we need to get the numbers of three types of slots to calculate the identification rate of SABA. In SABA, the staying tags and arriving tags are identified in the first and second phase, respectively.

Let $D_{SS}(T_{i+1}|T_i)$ and $D_{SI}(T_{i+1}|T_i)$ be the number of success and idle slots which a reader consumes to identify the staying tags, respectively. We can easily obtain that $D_{SS}(T_{i+1}|T_i)=n-\beta$ and $D_{SI}(T_{i+1}|T_i)=\beta$.

In the second phase of SABA, the reader estimates the number of arriving tags. Let $\hat{\alpha}$ be the estimation value of α, all the arriving tags are divided into $\hat{\alpha}$ groups to be identified. Let $D_A(T_{i+1}|T_i)$ represent the total number of slots consumed by the reader to identify all the arriving tags. $D^*_A(T_{i+1}|T_i)$ denotes the optimal $D_A(T_{i+1}|T_i)$.

Theorem:

$$D_A^*(T_{i+1}|T_i) = \alpha \sum_{x=0}^{\alpha} \binom{\alpha}{x} \left(\frac{1}{\alpha}\right)^x \left(1-\frac{1}{\alpha}\right)^{\alpha-x} D_{QT}(x) + 1 \quad (3)$$

where $D_{QT}(x)$ is the number of slots a reader consumes to identify x arriving tags by using QT, which is expressed as follows [11]:

$$D_{QT}(x) = 1 + \sum_{i=1}^{\infty} 2^{i+1} \left\{ 1 - \left(1 - \frac{1}{2^i}\right)^x - x \frac{1}{2^i} \left(1 - \frac{1}{2^i}\right)^{x-1} \right\} \quad (4)$$

Proof:

The reader uses the estimation value to divide all the arriving tags into $\hat{\alpha}$ groups. For a specific group, the probability that x tags select the group is expressed as follows:

$$P(x) = \binom{\alpha}{x} \left(\frac{1}{\hat{\alpha}}\right)^x \left(1 - \frac{1}{\hat{\alpha}}\right)^{\alpha - x} \quad (5)$$

The arriving tags in a group are identified by the reader as the same operations in QT. The reader sends a terminating command after identifying all the arriving tags, so the expression of $D_A(T_{i+1}|T_i)$ can be as follows:

$$D_A(T_{i+1}|T_i) = \hat{\alpha} \sum_{x=0}^{\alpha} \binom{\alpha}{x} \left(\frac{1}{\hat{\alpha}}\right)^x \left(1 - \frac{1}{\hat{\alpha}}\right)^{\alpha - x} D_{QT}(x) + 1 \quad (6)$$

If the reader can estimate the number of arriving tags accurately, the estimation value $\hat{\alpha}$ can be replaced by the real value α.

Proof ends.

To calculate the identification rate, we need to get the numbers of collision slots, success slots and idle slots when the reader identifies the arriving tags. In the second phase, the tags reply their IDs to the reader in collision or successful slots. Therefore, the success and collision slots have the same duration.

$D_{QT}(x)$ can also be expressed as:

$$D_{QT}(x) = I_{QT}(x) + C_{QT}(x) + S_{QT}(x) \quad (7)$$

where $I_{QT}(x)$, $C_{QT}(x)$ and $S_{QT}(x)$ denote the number of idle slots, collision slots and success slots, respectively. The value of $S_{QT}(x)$ is x. The value of $C_{QT}(x)$ can be expressed as:

$$C_{QT}(x) = \sum_{d=1}^{\infty} C_{QT}(x, d) \quad (8)$$

where $C_{QT}(x, d)$ represents the number of collision slots in depth d of $C_{QT}(x)$. The total number of slots in depth d is 2^d. The number of idle slots and success slots in depth d are $I_{QT}(x, d)$ and $S_{QT}(x, d)$, respectively, which can be expressed as:

$$I_{QT}(x, d) = 2^d \left(1 - \frac{1}{2^d}\right)^x \quad (9)$$

$$S_{QT}(x, d) = x \left(1 - \frac{1}{2^d}\right)^{x-1} \quad (10)$$

Then, $C_{QT}(x, d)$ can be further expressed as:

$$C_{QT}(x, d) = 2^d \left\{ 1 - \left(1 - \frac{1}{2^d}\right)^x - x \frac{1}{2^d} \left(1 - \frac{1}{2^d}\right)^{x-1} \right\} \quad (11)$$

Let $CS_{QT}(x)$ be the sum of collision slots and success slots in $D_{QT}(x)$, and then we have:

$$CS_{QT}(x) = \sum_{d=1}^{\infty} 2^d \left\{ 1 - \left(1 - \frac{1}{2^d}\right)^x - x \frac{1}{2^d} \left(1 - \frac{1}{2^d}\right)^{x-1} \right\} + x \quad (12)$$

The number of idle slots $I_{QT}(x)$ in $D_{QT}(x)$ can be shown as:

$$I_{QT}(x) = D_{QT}(x) - CS_{QT}(x) \quad (13)$$

Let $D_{AI}(T_{i+1}|T_i)$ and $D_{ACS}(T_{i+1}|T_i)$ be the number of idle slots and collision or success slots in the second phase of SABA, which can be expressed as:

$$D_{AI}(T_{i+1}|T_i) = \alpha \sum_{x=0}^{\alpha} \binom{\alpha}{x} \left(\frac{1}{\alpha}\right)^x \left(1 - \frac{1}{\alpha}\right)^{\alpha - x} I_{QT}(x) + 1 \quad (14)$$

$$D_{ACS}(T_{i+1}|T_i) = \alpha \sum_{x=0}^{\alpha} \binom{\alpha}{x} \left(\frac{1}{\alpha}\right)^x \left(1 - \frac{1}{\alpha}\right)^{\alpha - x} CS_{QT}(x) \quad (15)$$

Based on the link-timing in Section 2, the identification rate of SABA can be expressed as:

$$f_{SABA}(T_{i+1}|T_i) = \frac{n - \beta + \alpha}{\Delta_1 \cdot D_{SS}(T_{i+1}|T_i) + \Delta_2 \cdot D_{SI}(T_{i+1}|T_i) + \Delta_3 \cdot D_{AI}(T_{i+1}|T_i) + \Delta_4 \cdot D_{ACS}(T_{i+1}|T_i)} \quad (16)$$

where $\Delta_1 = T_{rs} + T_1 + T_s + T_2$, $\Delta_2 = T_{rs} + T_3$, $\Delta_3 = T_{ra} + T_3$ and $\Delta_4 = T_{ra} + T_1 + T_L + T_2$. T_{rs} denotes the time of a reader command when the reader identifies the staying tags. T_{ra}

denotes the time of a reader command when the reader identifies the arriving tags. T_s means the time a tag transmits its TID. T_L represents the time a tag transmits its ID. T_1, T_2 and T_3 have the same definitions as in **Fig. 1**.

According to the value of $D_{SS}(T_{i+1}|T_i)$ and $D_{ACS}(T_{i+1}|T_i)$, we can get the theoretical amount of data transmitted by tags which is $D_{SS}(T_{i+1}|T_i)\bullet L_{TID}+D_{ACS}(T_{i+1}|T_i)\bullet L_{ID}$, where L_{TID} and L_{ID} are the length of TID and ID, respectively.

6. Simulations and Discussions

In this section, we analyze the performance of SABA by comparing it with AQS [11], SBA [12] and MUIP [15]. QT is not considered, because the performance of QT is even much poorer than AQS. When QT is considered, the lines of the other algorithms will be compressed to a small space. A couple-resolution blocking technology is used in CRB and ECRB [13], which means two staying tags transmit their IDs to the reader in a slot. However, when the wireless channel is not ideal, the recognition result of CRB and ECRB becomes unreliable. Therefore, we do not compare SABA with CRB and ECRB. In aloha-based methods, MUIP has better performance than BUIP and SUIP, so we consider MUIP here. We do not consider FUTI and IFUTI [16] for two reasons. First, the two methods only identify arriving tags covered by the reader. Second, arriving tags cannot be recognized by the reader completely. Two metrics which are identification rate and data amount transmitted by tags are considered to evaluate the performance of the above algorithms. The identification rate is defined as the number of tags a reader can recognize in one second, which is the most essential performance of an anti-collision algorithm. Less amount of data transfer in tags means shorter signal transmission time, so the data amount transmitted by tags is also an important character we considered in all the simulations.

According to 18000-6C [1], both the reader command and the tag response contain a required length of preamble. The length of preamble is set to 6. In all the algorithms, the length of reader command code is set to 8. Based on 18000-6C, we set the bit rate of reader command and tag data to 120 kbps and 80 kbps, respectively. According to the tag data rate, in the link-timing in Section 2, we set T_1, T_2 and T_3 to 125 μs, 37.5 μs and 187.5 μs, respectively. Particularly, the length of reader ID is set to 8 and the length of TID is set to 16 in SABA. In all simulations, we assume that tag ID is distributed uniformly. This assumption can help us to analyze the average performance of all the algorithms, which is very common in previous studies [11-12].

In SBA, the value of the probability P is set to 0.1 or 0.2 and this setting is also adopted in [12]. For MUIP, to depict the best performance of this algorithm, we adopt the following assumptions. Firstly, the frame length is set to the sum number of arriving tags and staying tags. Secondly, all staying tags are dispersed to singleton slots. Finally, the reader can estimate the number of arriving tags perfectly when identifying arriving tags. What's more, the length of tag's short reply is set to 16 in MUIP.

In SABA, we consider the effects of different estimation errors of the arriving tags' number on the algorithm's performance. Let the estimated error be e, which can be defined as $e = (\hat{\gamma} - \gamma)/\gamma$, where, $\hat{\gamma}$ denotes the estimation number of arriving tags, γ indicates the real number of arriving tags. We consider three cases which are e=0, e=0.2 and e=-0.2 in all the simulations.

Although the algorithms which are SABA, AQS, SBA and MUIP can be applied in an error-prone environment, for simplicity, we still assume the communication channel between the reader and tags is perfect.

We first study the effects of different numbers of arriving tags and staying tags on the identification rate and data amount transmitted by tags of SABA and other methods. The numbers of arriving tags and staying tags are depicted by the arriving ratio r_a and staying ratio r_s. Assuming that the number of tags in the simulation environment is N and the number of tags in round T_i is $|T_i|$, then in round T_{i+1}, r_a is defined as the ratio of the number of arriving tags to N-$|T_i|$ and r_s is defined as the ratio of the number of staying tags to $|T_i|$. The number of tags that the reader should identify in T_{i+1} is $|T_i|\bullet r_s+(N-|T_i|)\bullet r_a$. Then, we examine the effects of changes in the tag ID length and tag size N on the performance of these algorithms. To get the average results, we repeat every simulation 200 times.

6.1 Impact of Arriving Tags

The experiment parameters are set as follows: N is 200, $|T_i|$ is set to 100, the length of tag ID is 128 bits, r_s is 1 and r_a changes from 0.1 to 1. The tag ID length is commonly used in the practical environment. **Fig. 4** shows the simulation results.

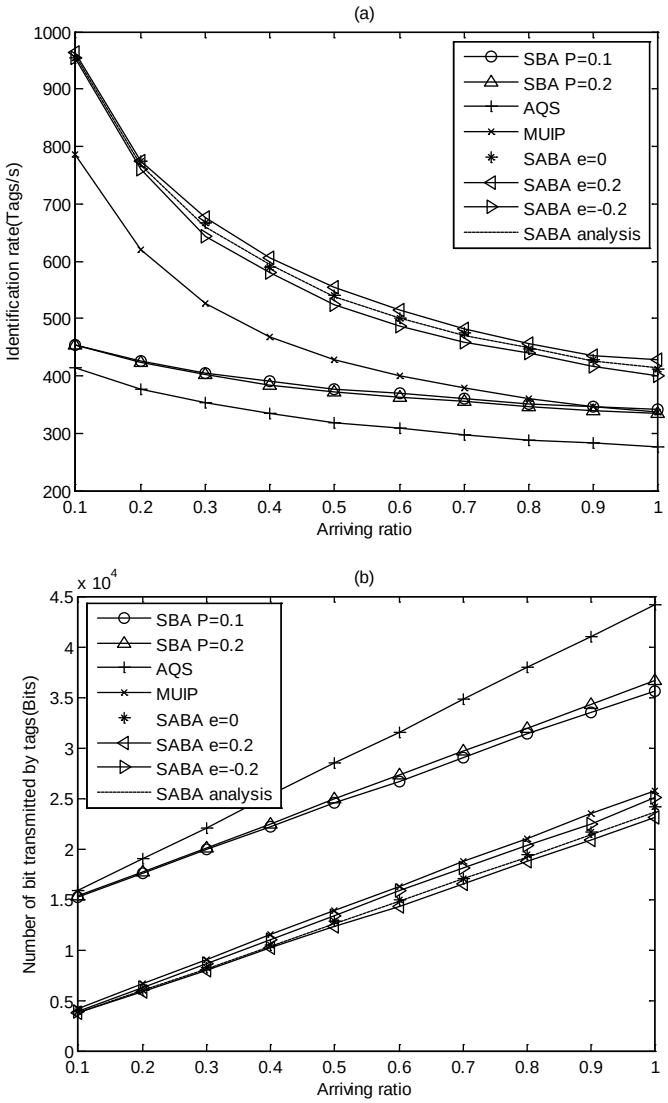


Fig. 4. Impact of r_a on the performance of algorithms

Fig. 4(a) shows the identification rate of SABA, AQS, SBA and MUIP at different values of r_a. The theoretical results of SABA are also shown in this figure. We can see from the figure that the performance of SABA is better than the other algorithms in all r_a values. The reason is that each staying tag only transmits a short data (TID) to the reader to inform its existence, which can effectively reduce the number of bits transmitted by tags. As a result, the identification rate of the reader is improved greatly. Although every staying tag in MUIP replies a short data to the reader to be identified, staying tags and arriving tags may collide with each other. Thus, the identification rate of MUIP is slower than SABA. However, the identification rate of MUIP is faster than SBA and AQS. The SBA with P=0.1 has similar performance with the SBA with P=0.2, which means that SBA is not sensitive to the probability P when r_a is increasing. All the idle slots and success slots are stored in the last round in AQS. In the current round, the idle slots may be still idle and the success slots may become collision slots. As a result, the identification rate of AQS is the lowest.

In SABA, another phenomenon can be seen from **Fig. 4**(a) is that the identification rate is higher when e=0.2 than the identification rate when e=-0.2 or e=0. The reason is that a positive estimation error will increase the quantity of idle slots and reduce the number of collision slots. A negative estimation error will decrease the number of idle slots but increase the amount of collision slots. From the link-timing in Section 2, we can know that the length of a collision slot is far larger than an idle slot. Therefore, SABA has a higher identification rate when the estimation error is positive. We can also see from the figure that the analysis results of SABA are the closest to the simulation results when the estimation error e is 0.

Fig. 4(b) depicts the data amount transmitted by tags in the four algorithms as r_a is changing. We can obtain from the figure that SABA has the least tag data amount among these algorithms. SABA has fewer number of collision slots when the estimation error e is 0.2, so the tag data amount in this case is less than the cases when e=0 and e=-0.2. Because each staying tag only replies a short data to the reader in MUIP, the data amount transmitted by tags of MUIP is similar to the SABA with e=-0.2. In SBA, the data amount transmitted by tags when P=0.1 is almost equal to the case when P=0.2. In AQS, all the arriving tags may collide with the staying tags, so the algorithm has the maximum amount of data transmitted by tags. In **Fig. 4**(b), the simulation results of SABA match well with the analysis results when the estimation error e equals 0.

6.2 Impact of Staying Tags

Then we study the performance of the four algorithms when the staying ratio varies. The simulation conditions are set as follows: N is 200, $|T_i|$ is set to 100, the length of tag ID is 128 bits, r_a is fixed to 1 and r_s increases from 0.1 to 1.

Fig. 5(a) shows the identification rate of these algorithms with the changes of r_s. We can see from the figure that the identification rate of SABA is less than SBA when r_s<0.5. The reason is that each reader command in SABA contains two more parameters which are rRID and the allocated slot. However, when r_s≥0.5 the identification rate of SABA is faster than the other algorithms. Especially, when r_s>0.7, since each staying tag in SABA only replies a short data to the reader and does not collide with arriving tags, so the identification rate of SABA is much larger than SBA, AQS and MUIP. When the number of arriving tags is overestimated, the number of idle slots will be increased but the number of collision slots will be decreased. Therefore, SABA can get a higher identification rate when the estimation error is 0.2. We can see from the figure that the simulation results are consistent with the analysis results when e=0 of SABA. We can get from the figure that, when r_s<0.5 the SBA with P=0.2 is better than the

SBA with P=0.1, when $r_s \geq 0.5$ the SBA with P=0.1 is better than the SBA with P=0.2. The reason is as follows: in SBA, when r_s is less than 0.5, there will be more idle slots which can be occupied by arriving tags, in these cases, a larger P is better. In other words, when r_s is not less than 0.5, a smaller P is better. In AQS, along with the number of staying tags is increasing, the probability of collision between the staying tags and arriving tags is also increasing, therefore, the gap between AQS and the other algorithms is becoming more obvious. Although each staying tag only replies a short data to the reader in MUIP, the identification rate of MUIP is even slower than AQS when $r_s < 0.5$. The reason is that, when r_s is small, there will be many idle slots in MUIP, and too much idle slots will increase the identification time of the reader. However, as the number of staying tags increases, the performance of MUIP will be better than AQS.

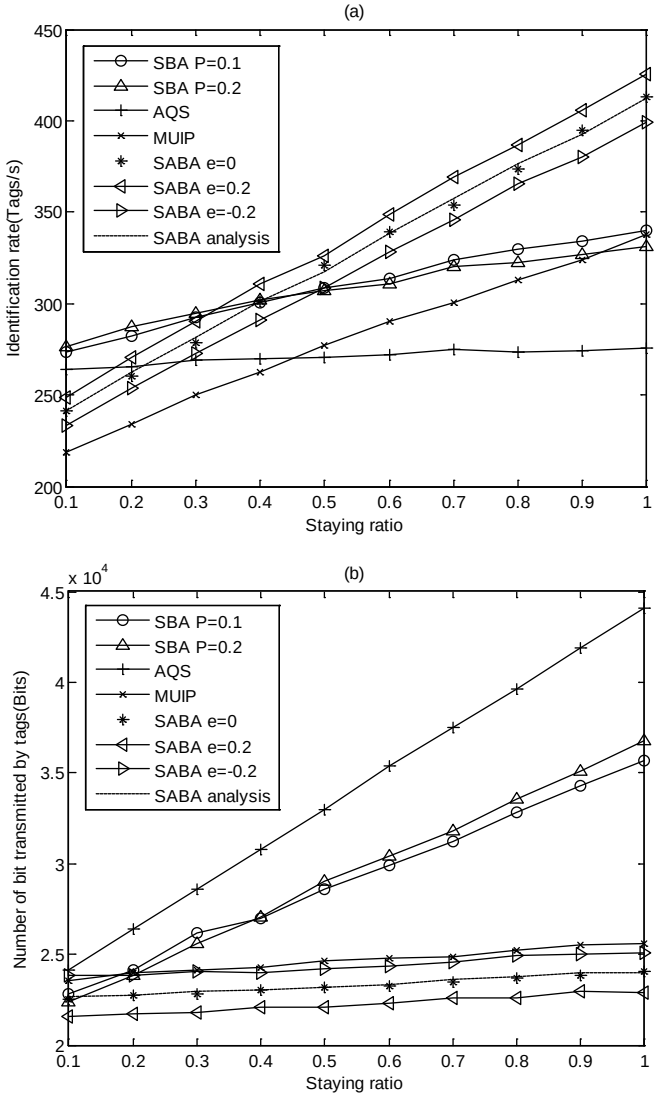


Fig. 5. Impact of r_s on the performance of algorithms

Fig. 5(b) illustrates the data amount transmitted by tags in AQS, SBA, MUIP and SABA as r_s varies. In these algorithms, SABA has the least data amount when the estimation error is 0 or

0.2. Especially, when the number of staying tags is large, the advantage of SABA will be more apparent. When e=-0.2 and r_s=0.1, the four algorithms have similar tag data traffic. When r_s>0.1, the SABA with e=-0.2 still has the least tag data amount in the four algorithms. The number of collisions between the staying tags and the arriving tags in AQS are increasing when r_s is adding, so the tag data traffic of AQS is always greater than SBA. As the short reply mechanism in MUIP, the number of bits transmitted by tags in MUIP is similar to SABA's. In addition, the simulation results of SABA match well with the analysis results.

6.3 Impact of Tag ID Length

In this simulation, we study the effects of changes in the tag ID length on the performance of these algorithms. The experimental parameters are set as follows: N is 200, $|T_i|$ is set to 100, both r_a and r_s are 1, tag ID length is in the range of 96–160 bits. The simulation results are shown in **Fig. 6**.

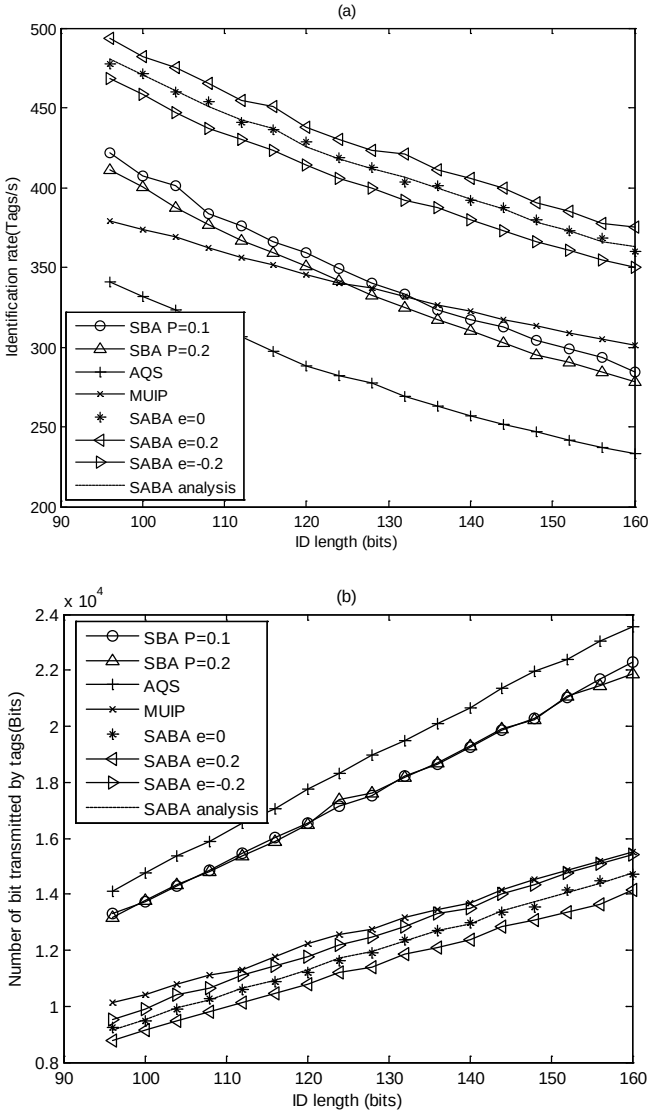


Fig. 6. Impact of tag ID length on the performance of algorithms

Fig. 6(a) describes the identification rate of the four algorithms along with the changes of the tag ID length. We can get from the figure that SABA has the fastest identification rate on all the tag ID length. The number of idle slots is increased and the number of collision slots is decreased in SABA when the estimation error e is positive, so the SABA with e=0.2 has a faster identification rate. Further, the simulation results of SABA are consistent with the analysis results when e=0. The SBA with P=0.1 or 0.2 has a higher rate than AQS. When both r_a and r_s are 1, many collisions will occur between the staying tags and arriving tags in AQS, therefore, the identification rate of AQS is the slowest. There will be many collision slots in MUIP when both r_s and r_a are 1, so the identification rate of MUIP is less than SBA when the tag ID length is shorter than 120. However, as the tag ID length is increasing, the advantage of the short reply technology in MUIP will be more obvious. Thus, the identification rate of MUIP will be faster than SBA as shown in **Fig. 6**(a).

Fig. 6(b) shows the variation of the tag data transmission amount with the changes in tag ID length. We can obtain from the figure that the tag data traffic of SABA is less than the other algorithms. We can also see that the analysis results of SABA are almost consistent with the simulation results when e=0. The number of bits transmitted by tags of MUIP is still similar to SABA's. When P=0.1 or P=0.2, SBA has closer performance. A blocking mechanism is not adopted in AQS, so the number of bits transmitted by tags of AQS is the maximal.

6.4 Impact of Tag Size

Finally, we research how the tag size N affects the performance of SABA and other methods. The simulation parameters are set as follows: |Ti| is set to 100, both rs and ra are set to 1, the tag ID length is set to 128, the tag size N varies in the range of 100-300. **Fig. 7** depicts the simulation results.

Fig. 7(a) describes the identification rate of the four methods when the tag size N is increasing. We can get that the proposed SABA has the fastest recognition rate among these methods. When the estimated number of arriving tags is larger than the real value, the number of collision slots will be reduced and the number of idle slots will be increased, *vice versa*. The length of a collision slot is much longer than the length of an idle slot. Therefore, the SABA with e=0.2 has better performance than the SABA with e=0 and e=-0.2. We can also see from this figure that the simulation results match well with the analysis results in SABA. When the tag size N is less than 200, the identification rate of MUIP is faster than SBA. However, when the tag size N is adding, the collisions between staying tags and arriving tags in MUIP will be increased considerably. Therefore, MUIP and SBA have similar identification rates when the tag size N is larger than 200. AQS doesn't adopt a blocking mechanism or a short reply technology. As a result, AQS's identification rate is the slowest.

When the tag size N is increasing, the numbers of bits transmitted by tags of these methods are shown in **Fig. 7**(b). We can obtain from this figure that the data traffic of the four algorithms are adding, but the proposed SABA always has the least number of bits transmitted by tags. What's more, the simulation results of the SABA with e=0 are consistent with the analysis results. For the short reply mechanism of staying tags, the tags in MUIP transmit a similar number of bits with the tags in SABA. As arriving tags and staying tags in MUIP may collide with each other, the number of bits of MUIP is slightly more than SABA's. In SBA and AQS, all staying tags retransmit their IDs to the reader to be identified, so the two methods' data traffic is much more than SABA and MUIP. Because all arriving tags may collide with staying tags in AQS, the tags in AQS transmit the most number of bits.

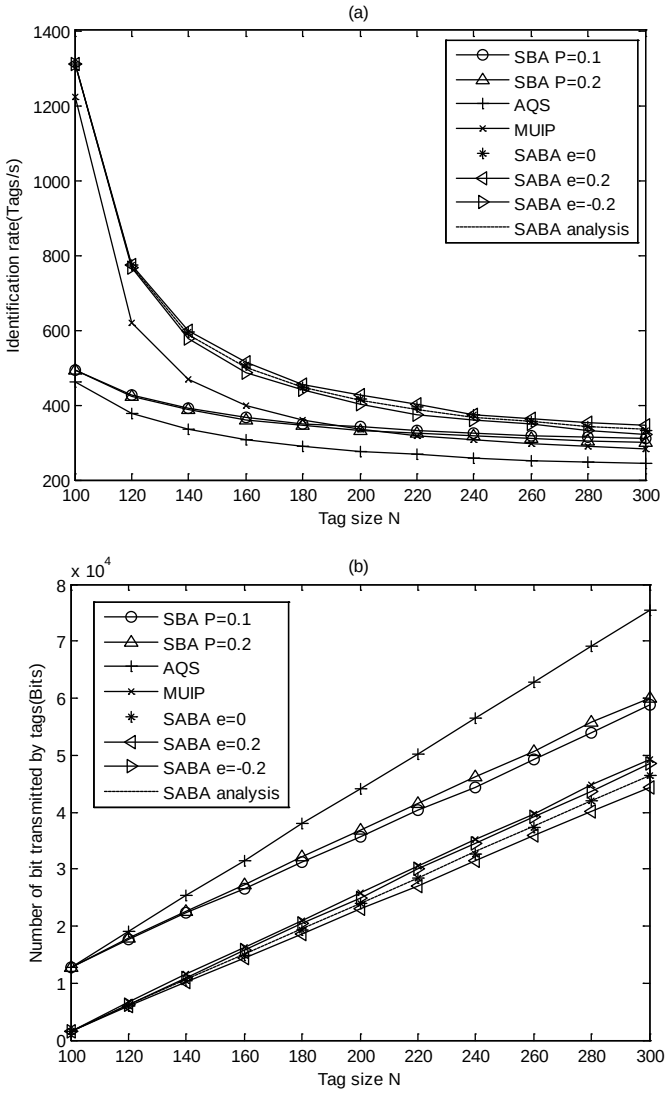


Fig. 7. Impact of tag size on the performance of algorithms

6.5 Complexity Analysis of SABA

Here, we analyze the reader complexity and tag complexity in SABA. In the case of reader complexity, a reader in SABA performs the operations of sorting and assigned slot generating which are very simple. In the case of tag complexity, a SABA tag needs to store two parameters which are tRID and TID. The typical lengths of the two parameters are 8 bits and 16 bits, respectively. In the existing commercial tags, such as the Monza 4QT chip of Impinj has a user memory of 512 bits, the total storage capacity of the Alien Higgs 3 chip is 800 bits. Therefore, the two parameters only occupy a small portion of the storage capacity of these tags. In addition to storing the two parameters, a SABA tag only performs a matching operation, so the computational complexity of the SABA tag is very light.

6.6 Discussions on the Estimation Error

In RFID tag anti-collision algorithms, the identification delay can be defined as the number of slots consumed by the reader to identify the tags in its scope [12-13]. A grouping method can effectively reduce the identification delay of the reader, such as in SBA and the proposed SABA, n arriving tags are divided into n groups. In the grouping method, an assumption is connotative, which is that a collision slot, success slot and idle slot of the RFID system have the same time duration. However, we can get from the link-time of the RFID system in **Fig. 1** that the length of an idle slot is obviously shorter than a success slot or a collision slot. When the estimation value of tags' number is slightly larger than the real value of tags' number, based on the same grouping method, the number of collision slots may be reduced and the number of idle slots may be increased. As a result, the identification time of the reader is decreased, such as the SABA with e=0.2 in the above simulations. It can be foreseen that, if the estimation value of tags' number is much larger that the real value, the identification time of the reader must be lengthened. Therefore, in the actual applications, if the number of arriving tags can be estimated correctly, to reduce the identification time of the reader, we can slightly increase the number of groups. When the time durations of three types of slots (success, collision and idle) are different, how to optimally group the arriving tags is a future topic of this research.

7. Conclusion

In the RFID systems, a high-performance anti-collision algorithm can effectively improve the identification rate of a reader. When the reader needs to recognize the tags in its range repeatedly, the reader should identify all the staying tags quickly and spend the main time in recognizing all the arriving tags. Therefore, we propose SABA which uses a slot allocation mechanism to assign a unique slot which can be taken as a temporary ID (TID) to each tag. In SABA, the reader can quickly identify all the staying tags by using the TIDs.

In this paper, the performance of SABA is evaluated through analysis and simulations. Some important findings are summarized as follows:

1) SABA is a blocking algorithm, which not only prevents the collisions caused by staying tags, but also prohibits the collisions between the arriving tags and staying tags.

2) SABA uses a slot allocation mechanism. Each staying tag only replies a short data to the reader and the data amount transmitted by staying tags is reduced greatly.

3) When the number of arriving tags, the number of staying tags, the length of tag ID and the tag size are varying, compared to AQS, SBA and MUIP, SABA shows much better performance.

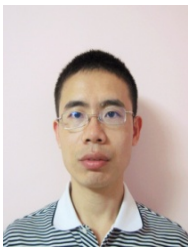
References

[1] ISO/IEC Standard 18000-6:2004/Amd. 1:(E), Information Technology, Radio Frequency Identification for Item Management, Part 6: Parameters for Air Interface Communications at 860 MHz to 960 MHz, Amendment 1: Extension with Type C and Update of Types A and B, ISO/IEC, June 2006. Article (CrossRef Link).

[2] EPC Radio-Frequency Identity Protocols Generation-2 UHF RFID Protocol for Communications at 860 MHz - 960 MHz Version 2.0.0. EPC global technical specification, 2013. Article (CrossRef Link).

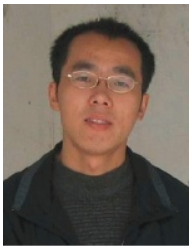
[3] K. Finkenzeller, "RFID handbook: radio-frequency identification fundamentals and applications," *in Contactless Smart Cards Identification*, 2010. Article (CrossRef Link).

[4] M. K. Yeh, J. R. Jiang and S. T. Huang, "Adaptive splitting and pre-signaling for RFID tag anti-collision," *Computer Communications*, vol. 32, no. 17, pp. 1862-1870, 2009. Article (CrossRef Link).
[5] D. H. Shih, P. L. Sun, D. C. Yen and S. M. Huang, "Taxonomy and survey of RFID anti-collision protocols," *Computer communications*, vol. 29, no. 11, pp. 2150-2166, 2006. Article (CrossRef Link).
[6] J. Myung, W. Lee, J. Srivastava and T. K. Shih, "Tag-splitting: adaptive collision arbitration protocols for RFID tag identification," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 6, pp.763-775, 2007. Article (CrossRef Link).
[7] Y. H. Cui and Y. P. Zhao, "Performance evaluation of a multi-branch tree algorithm in RFID," *IEEE Transactions on Communications*, vol. 58, no. 5, pp. 1356-1364, 2010. Article (CrossRef Link).
[8] J. S. Li and Y. M. Huo, "An efficient time-bound collision prevention scheme for RFID re-entering tags," *IEEE Transactions on Mobile Computing*, vol. 12, no. 6, pp. 1054-1064, 2013. Article (CrossRef Link).
[9] L. Pan and H. Wu, "Smart trend-traversal: a low delay and energy tag arbitration protocol for large RFID systems," *IEEE INFOCOM 2009*, pp. 2571-2575, 2009. Article (CrossRef Link).
[10] C. Law, K. Lee, and K. Y. Siu, "Efficient memoryless protocol for tag identification (extended abstract)," in *Proc. of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pp. 75-84, 2000. Article (CrossRef Link).
[11] J. Myung, W. Lee and T. K. Shih, "An adaptive memoryless protocol for RFID tag collision arbitration," *IEEE Transactions on Multimedia*, vol. 8, no. 5, pp. 1096-1101, 2006. Article (CrossRef Link).
[12] Y. C. Lai, C. C. Lin and C. L. Lee, "A semi-blocking algorithm on adaptive query splitting for RFID tag identification," *EURASIP Journal on Wireless Communications and Networking*, 231, pp. 1-14, 2013. Article (CrossRef Link).
[13] Y. C. Lai and C. C. Lin, "Two couple-resolution blocking protocols on adaptive query splitting for RFID tag identification," *IEEE Transactions on Mobile Computing*, vol. 11, no. 10, pp. 1450-1463, 2012. Article (CrossRef Link).
[14] Y. C. Lai, L. Y. Hsiao and B. S. Lin, "An RFID anti-collision algorithm with dynamic condensation and ordering binary tree," *Computer Communications*, vol. 36, no. 17, pp. 1754-1767, 2013. Article (CrossRef Link).
[15] X. Liu, B. Xiao, S. Zhang and K. Bu, "Unknown tag identification in large RFID systems: An efficient and complete solution," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, issue: 6, pp. 1775-1788, 2015. Article (CrossRef Link).
[16] X. Liu, K. Li, G. Min, K. Lin, B. Xiao, Y. Shen and W. Qu, "Efficient unknown tag identification protocols in large-scale RFID systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 12, pp. 3145-3155, 2014. Article (CrossRef Link).
[17] S. Lee, S. Joo, and C. Lee, "An enhanced dynamic framed slotted ALOHA algorithm for RFID tag identification," in *Proc. of IEEE MobiQuitous*, pp. 166-172, 2005. Article (CrossRef Link).
[18] T. Li, S. Chen, and Y. Ling, "Identifying the missing tags in a large RFID system," *Proc. of ACM Mobihoc*, 2010. Article (CrossRef Link).
[19] S. Chen, M. Zhang and B. Xiao, "Efficient information collection protocols for sensor-augmented RFID networks," *IEEE INFOCOM 2011*, pp. 3101-3109, 2011. Article (CrossRef Link).
[20] Y. Qiao, S. Chen, T. Li and S. Chen, "Energy-efficient polling protocols in RFID systems," in *Proc. of MobiHoc*, page 25, 2011. Article (CrossRef Link).

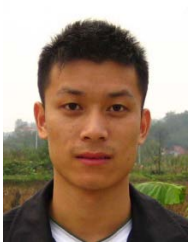


Yang Qing received the B.S. degree in electronic science and technology from Xi'dian University, Xi'an, China, in 2008, and the M.S. degree in electronic science and technology from National University of Defense Technology, Changsha, China, in 2010. He is currently working towards the Ph.D. degree at National University of Defense Technology, Changsha, China, in information and communication engineering.
In 2008, he joined ASIC R&D center, National University of Defense Technology. His research focuses on RFID technology and wireless communication protocol.

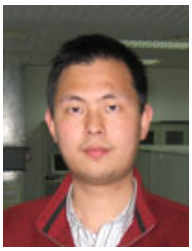
Li Jiancheng received the M.S. and Ph.D. degrees from National University of Defense Technology, Changsha, China, in 2003 and 2010, respectively. In September 2003, he joined the Satellite Navigation R&D center, National University of Defense Technology, as a member of Technical Staff. In September 2006, he joined the ASIC R&D Center, National University of Defense Technology, with responsibility for the developing of a CMOS RFIC applying to a widely-tunable multi-standard wireless communication system. He is currently the professor and director of the ASIC R&D Center, National University of Defense Technology. His research interests include ASIC design and system application, RF integrated circuits design and RFID technology.



Wang Hongyi received the B.S. degree from Beijing University of Aeronautics and Astronautics, Beijing, China, in 2001, and the M.S. and Ph.D. degrees in electronic engineering from National University of Defense Technology, Changsha, China, in 2003 and2008 respectively.
He is currently an instructor in ASIC R&D Center, National University of Defense Technology. His research interests include digital design, RFID system design, test and debug.



Zeng Xianghua received the B.S., M.S. and Ph.D. degrees from National University of Defense Technology, Changsha, China.
He is currently an instructor in ASIC R&D Center, National University of Defense Technology. His research interests include communication and internet of things.



Zheng Liming received the B.S., M.S. and Ph.D. degrees from National University of Defense Technology, Changsha, China.
He is currently an instructor in ASIC R&D Center, National University of Defense Technology. His research interests include internet & information security, internet of things.