


A bit arbitration tree anti-collision protocol in radio frequency identification systems

International Journal of Distributed Sensor Networks
2017, Vol. 13(11)
© The Author(s) 2017
DOI: 10.1177/1550147717741571
journals.sagepub.com/home/ijdsn


Yu Fu¹, Xue Wang¹, Enshu Wang² and Zhihong Qian¹

Abstract

Radio frequency identification technology has been extensively used in various practical applications, such as inventory management and logistics control. When numerous tags respond to reader simultaneously, tags-to-tag collision occurs and causes the reader to identify tags unsuccessfully. Therefore, how to reduce tag collisions has already emerged as an urgent and crucial problem to be solved for speeding up the identification operation. This article designs a characteristic-value-based grouping rule and a collision bits rule to determine transmitted bit string combinations accurately and proposes a bit arbitration tree anti-collision protocol based on these two rules to decrease the time for collecting all tag IDs. Furthermore, we consider the case that received bit string less than three bits, which occurs during the tag identification operation. Both theory and simulation analyses show that the proposed protocol can reduce the number of idle slots and total number of slots and thereby improve system efficiency.

Keywords

Anti-collision, bit arbitration, radio frequency identification, tag identification

Date received: 22 April 2017; accepted: 17 October 2017

Handling Editor: Riccardo Colella

Introduction

Radio frequency identification (RFID) has been widely used in our lives due to its low cost and ubiquitous characteristics, and it brings a revolutionary change in a wide range of applications, such as inventory control,¹ treatment,² smart environments³ and warehouse management.⁴ An RFID system is composed of one or several readers and a large number of tags. The RFID reader has powerful computation and memory capability and is mainly responsible for controlling communication among tags. Each tag in RFID systems has a unique ID that cannot be modified. When a tag is in the interrogation zone of the reader, they can communicate with each other via radio frequency. The reader can execute an identification operation to send inquiry request to ask tags to respond with their IDs. When more than one tag responds simultaneously, signal collisions will occur and cause the reader to identify tags unsuccessfully.

There are three types of collision as shown in Figure 1. These collisions can be classified as follows:⁵

Tags-to-tag collision, as illustrated in Figure 1(a), occurs when more than one tag attempts to transmit data simultaneously to a single reader within its interrogation zone. In this scenario, these tags are unable to be identified correctly by the reader.

Readers-to-tag collision, as illustrated in Figure 1(b), occurs when two or more readers try to

¹College of Communication Engineering, Jilin University, Changchun, China

²Department of Computer Science and Engineering, University at Buffalo, State University of New York, Buffalo, NY, USA

Corresponding author:

Zhihong Qian, College of Communication Engineering, Jilin University, No. 5372 Nanhu Avenue, Changchun 130012, China.
Email: dr.qzh@163.com



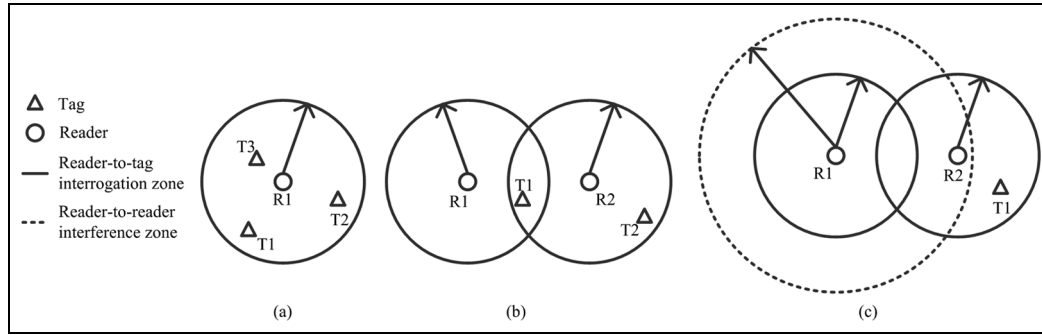


Figure 1. Three types of collision in RFID system: (a) tags-to-tag collision, (b) readers-to-tag collision and (c) readers-to-reader collision.

communicate with the same tag simultaneously in their overlap interrogation zone. In this scenario, it is not able to select a particular reader to transmit the tag ID.

Readers-to-reader collision, as illustrated in Figure 1(c), occurs when the transmission signal generated from one reader can be interfered by the communication of other readers in its interference zone. Therefore, the readers cannot read tags.

To date, there are many protocols for solving tags-to-tag collision problems that can be classified into two categories: Aloha-based probabilistic protocols^{6,7} and tree-based deterministic protocols.^{8,9} The Aloha-based protocols are mainly divided into pure Aloha (PA) protocol,¹⁰ slotted Aloha (SA) protocol,¹¹ framed slotted Aloha (FSA) protocol¹² and dynamic framed slotted Aloha (DFSA) protocol.¹³ The Aloha-based protocols estimate the quantity of unidentified tags in the interrogation zone of the reader and allocate an appropriate number of time slots to reduce probability of collision and then the tags randomly select a slot in the frame and transmit their information to the reader. If the transmission is collision, tags will repeatedly send their information in next frame until tags are identified successfully. However, Aloha-based protocols have the tag starvation problem due to the inaccurate frame size. When the number of tags is large and frame length is small, some tags may not be recognized for a long time. In contrast, the tree-based deterministic protocols such as binary search (BS) protocol,¹ collision tree (CT) protocol¹⁴ and query tree (QT) protocol¹⁵ do not cause tag starvation. In the tree-based protocols, only the tags of which IDs match the prefix will respond to the reader. If the transmission is collision, reader repeatedly divides the collided tags into two subsets until the number of tags in a group is only one, which can be identified successfully by the reader.

In this article, we propose a bit arbitration tree (BAT) anti-collision protocol to minimize the total

number of slots, to speed up the identification and to increase the system efficiency in RFID systems. To the best of our knowledge, this article should be the first attempt to define two rules, characteristic-value-based grouping rule and collision bits rule, to infer bit string combinations and decrease idle slots and total number of slots. We analytically investigate the performance of the proposed protocol and compare it with existing protocols. Our simulation results demonstrate that the tag ID length is unrelated to system efficiency in the BAT protocol. And the proposed protocol performs much better in terms of system efficiency than QT, adaptive 4-ary pruning QT (A4PQT), 8-ary QT protocol and CT.

The rest of the article is organized as follows. Manchester coding and A4PQT are described in section 'Related work'. The detailed design of BAT protocol and the operation flow of BAT protocol are presented in section 'The BAT anti-collision protocol'. We analyse the experimental results and compare the proposed protocol with A4PQT, QT, 8-ary QT protocol and CT in section 'Simulation and comparison', and we make conclusions in the final section.

Related work

Manchester coding

Manchester coding is widely used for collision detection in tree-based protocols.^{14–18} The purpose of Manchester coding is to give an example to find where the collided bit is. In Manchester coding, the value of a bit is defined by the change in level within a bit window. The negative transition means a bit '0', whereas the positive transition means a bit '1'. In an RFID system, if more than two tags transmit bits with different values, then the negative and positive transition will counteract. This result is impermissible in Manchester coding during data transmission. Therefore, the reader using Manchester coding will know collisions

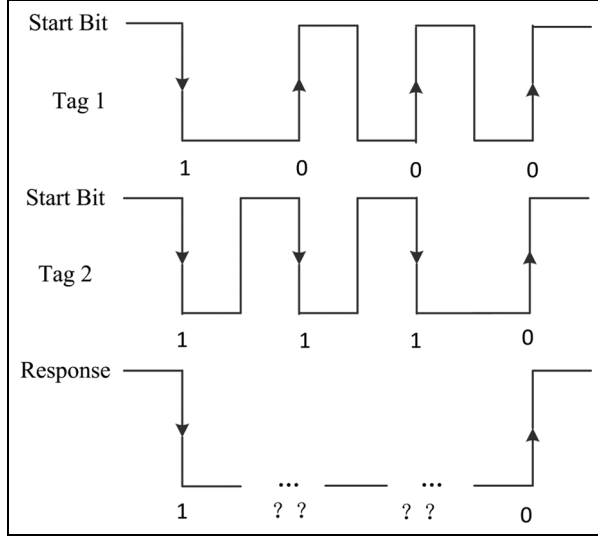


Figure 2. Response of tags with Manchester coding.

occurrence. Manchester coding makes it possible to find and identify the collision bits.

Figure 2 shows an example of Manchester coding. Assuming that tag 1, whose ID is 1000, and tag 2, whose ID is 1110. Since tag 1 and tag 2 transmit their IDs simultaneously, the decoded data from the signal received by reader is 1XX0 using Manchester coding, where ‘X’ represents a collision bit. That is, the collision bits are second and third.

Tag anti-collision protocols

Most of the existing tag anti-collision protocols can be classified into two categories. The first category is Aloha-based anti-collision protocols.^{19–23} Zheng and Kaiser¹⁹ proposed two adaptive frame size Aloha protocols, in which the following frame size was adaptively changed according to the real-time collision rate in the current frame. Su et al.²⁰ investigated how to determine the optimality of the current frame size and proposed a detected sector-based DFSA. Liu et al.²¹ designed a prefix-assisted FSA (PA-FSA) approach to improve the estimation accuracy of the traditional FSA. Solic et al.²² proposed a tag number estimate method as the slot-by-slot based on frame-by-frame (FbF) estimate. Xie et al.²³ also proposed an ensemble sampling-based method to estimate the tag size for a number of categories and achieve time efficiency.

The second category is tree-based anti-collision protocols.^{15,24,25} In QT,¹⁵ a reader sends a query including a prefix to tags. Tags respond with their IDs if the prefix is matched with their IDs. When a collision occurs, two new queries are generated by adding a ‘0’ and ‘1’ to the previous prefix. In order to alleviate tags from transmitting large number of bits, Landaluce et al.²⁴

presented an improved QT protocol, Query window Tree (QwT) protocol using a dynamic-size window. In 8-ary QT protocol,²⁵ a reader adds three bits to the collision query prefix and then generated eight query strings are sent to tags. 8-ary QT protocol reduces the collisions while increases the unnecessary inquiries. The above protocols can effectively solve the collision between tags, but they still have more unnecessary idle slots, and the performance needs to be improved.

A4PQT

A lot of researchers have proposed many protocols to improve the QT protocol. The A4PQT,¹⁸ anti-collision protocol, which adaptively prunes the idle time slot branches based on the information of collision bits, is an improved protocol of QT protocol and 4-ary QT protocol. A reader broadcasts a query string as prefix to tags within its interrogation zone. The tag whose ID matches the prefix will transmit its remaining ID except the matched prefix to the reader. When a collision occurs, the reader only receives the first collision bit and subsequent one data bit of their IDs, and then updates query string. Supposed that the reader receives information from the response of tag is $q_1q_2 \dots q_{c-1}q_cq_{c+1} \dots q_l$, where q_c is the highest collision bit detected by the reader according to Manchester coding. And l is the length of tag’s ID and Q is the set of query string. If more than one tag responds simultaneously, then new query prefixes are generated as the following principles:

1. If the q_{c+1} is 0, then the reader pushes two new prefixes, $Qq_1q_2 \dots q_{c-1}00$ and $Qq_1q_2 \dots q_{c-1}10$, onto the query stack. In this case, the reader cuts two branches of 4-ary tree by excluding $Qq_1q_2 \dots q_{c-1}01$ and $Qq_1q_2 \dots q_{c-1}11$.
2. If the q_{c+1} is 1, then the reader pushes two new prefixes, $Qq_1q_2 \dots q_{c-1}01$ and $Qq_1q_2 \dots q_{c-1}11$, onto the query stack. In this case, the reader cuts two branches of 4-ary tree by excluding $Qq_1q_2 \dots q_{c-1}00$ and $Qq_1q_2 \dots q_{c-1}10$.
3. If the q_{c+1} is a collision bit, then the reader pushes four new prefixes, what are $Qq_1q_2 \dots q_{c-1}00$, $Qq_1q_2 \dots q_{c-1}01$, $Qq_1q_2 \dots q_{c-1}10$ and $Qq_1q_2 \dots q_{c-1}11$, onto the query stack. In this case, the reader does not cut any branches of 4-ary tree.

Therefore, the A4PQT can avoid some idle time slots. The total slots of A4PQT protocol is in the range of $(7n - 1)/6$ to $2n - 3$. However, when the IDs of tag 1 and tag 2 are 00 and 11, respectively, the reader receives the data as XX. Then, A4PQT generates four new branches instead of cutting any branches, which produces additional two idle slots.

In order to reduce some idle time slots the A4PQT brings, this article proposes a BAT anti-collision protocol, combining with the rule of collision bits to infer the transmitted bit strings. The improvement of BAT is that, by employing a designed grouping method and the collision bits, it decreases idle slots with corresponding characteristic bits to identify tags. Both theoretical and experimental results show that BAT can achieve better performance than A4PQT over system efficiency.

The BAT anti-collision protocol

Although QT, 8-ary tree protocol and A4PQT speed up the tag identification process, they still have unnecessary inquiries. To solve the problem, this article designs two rules: grouping rule based on characteristic value and the rule of collision bits, and proposes a novel method, BAT protocol. Our solution is compatible with the existing standard, EPCglobal specification. In BAT, tags are divided into two groups by grouping rule based on characteristic value, and transmitted bit string combinations are accurately determined using the rule of collision bits. Furthermore, the case that received bit string less than three bits is considered. BAT can reduce idle slots and total number of slots and thereby improve system efficiency.

Two rules definition

Binary systems use XOR, and we define \otimes as the operator of XOR, then $0 \otimes 0 = 0$, $0 \otimes 1 = 1$, $1 \otimes 1 = 0$. That is, XOR any binary number A with the binary value 1 will result in the opposite binary value of A , while XOR A with the binary value 0 will result in A itself.

If the tag ID is a fractal dimension, each of the three adjacent bits is one-dimensional (1D), that is, in a tag ID there are three successive bits $q_j q_{j+1} q_{j+2}$ from the j th bit, which can be taken as its m th-dimension, that is, $m = \lfloor (j+2)/3 \rfloor$. The m th-dimension can be indicated by D_m . Defining $q_j \otimes q_{j+1}$ is equal to its characteristic value C .

A reader has a stack, which is used to store the query prefixes using the first-in-first-out principle. Each tag owns two counters, GC and C. Since tags need battery to keep the value of those counters, our proposed BAT protocol is for active tags. Assume that a given tag i , which has $GC(i)$ and $C(i)$. It should be noted that $GC(i)$ and $C(i)$ are updated in each dimension. The functions are as follows:

1. $GC(i)$: records the group to which each tag belongs. If $GC(i) = 0$ means that the tag i belongs to the 0th group, denoted as G_0 . If $GC(i) = 1$, the tag i belongs to the 1st group, denoted as G_1 .

Table 1. Grouping example.

	G_0	G_1
$l_{j+1} l_{j+2}$	000	001
	011	010
	101	100
	110	111

2. $C(i)$: records the characteristic value, which is equal to $q_j \otimes q_{j+1}$.

Definition 1. Grouping rule based on characteristic value. If $C(i)$ is equal to q_{j+2} , tag i belongs to G_0 , that is, $GC(i) = 0$ of this tag. Otherwise, tag i belongs to G_1 , that is, $GC(i) = 1$.

Table 1 shows the specific groups based on this definition. From Table 1, we conclude that any two bits which execute an XOR operation is equal to the remaining bit in G_0 and any two bits executing an XNOR operation equals the remaining bit in G_1 .

Definition 2. Rule of collision bits. If a reader receives a bit string $q_j q_{j+1} q_{j+2}$ with two collision bits. With the only known bit, we infer the collision bits. This known bit which is equivalent to the two collision bits execute an XOR operation in G_0 or an XNOR operation in G_1 .

For example, if the reader receives a bit string as X1X in G_0 , it generates 110 and 011. If the bit string is in G_1 , it generates 010 and 111.

Protocol description

Related protocol instructions are described as follows:

REQUEST(ε): a request command. All tags, existing within the scope of a reader, automatically respond and send their IDs to the reader.

REQUEST(PRE): a request command. Within the query process, the value of PRE has to be updated. Here, PRE is a query prefix. Tags respond if their prefixes are the same as PRE after receiving the command. And then they compute GC. When RFID reader broadcasts REQUEST(PRE), the responded tags transmit their GCs and remaining ID except PRE to the reader.

REQUEST(PRE,G): a request command. Here, PRE is also a query prefix and G denotes group number, 0 or 1. When an RFID reader broadcasts REQUEST(PRE,G), the tags of which IDs' prefixes are the same as PRE respond and are divided into two groups according to definition 1, grouping rule based on characteristic value. And then tags in G send their remaining IDs except PRE to the reader.

PUSH(PRE): a read/write command. PRE, as a new prefix, is pushed to the bottom of the stack.

POP(PRE): PRE pulls from the top of the stack.

SELECT(ID): a select command to select a tag with the same ID.

READ_{DATA}: a read command to read tag IDs.

SILENCE: a silent command. The tag not responds to the newly queued queries in the tag identification process.

'||' represents a concatenation operation, such as 01 || 11 means 0111.

In this article, BAT protocol is proposed. When an RFID reader broadcasts REQUEST(PRE), these responded tags transmit their GCs and remaining ID except PRE to the reader. When no tag responds, it means that the group has zero tag, that is, an idle slot. When only one tag responds, the reader identifies the tag directly, and sends SELECT(ID) and READ_{DATA} to that tag. Once the tag has received these commands, the reader sends SILENCE to the tag to make it silent.

When two or more tags transmit signal at the same time, the reader can infer the bit string combinations based on the rule of collision bits. The reader receives decoded data and GCs of responded tags that stored in the reader's counter RGC. In the decoded data, q_c is the first collision bit, and successive two bits $q_c + 1q_c + 2$ compose a bit string $q_cq_c + 1q_c + 2$. Using the bit string and RGC, the reader makes it possible to find the collision bits and determines that there are two groups of tags or only one group of tags (i.e. G_0 or G_1) during data transmission. Assuming that the tag 1, which ID is 001, and tag 2, which ID is 111. It is concluded that $GC(1) = 1$ and $GC(2) = 1$ based on definition 1. Since these two tags should transmit their IDs and GCs simultaneously, the decoded data from the signal received by reader are XX1 and RGC is 1, using Manchester coding. After receiving the bit string $q_cq_c + 1q_c + 2$, the reader can determine one of the three conditions:

C1. The bit string $q_cq_c + 1q_c + 2$ has only one collision bit. The reader directly recognizes that the collision bit q_c to be 0 (or 1) and then sends PUSH(PRE || 0 ($q_c + 1q_c + 2$)) and PUSH(PRE || 1 ($q_c + 1q_c + 2$)).

C2. The bit string $q_cq_c + 1q_c + 2$ has two collision bits.

- If the value of RGC at the reader has no collision, this means that the responded tags are in the same group, the reader obtains two kinds of bit strings according to the rule of collision bits and then sends PUSH(PRE || bit string).
- If the value of RGC at the reader indicates a collision, this means that the responded tags are in different groups, the reader obtains two kinds of bit strings in each of the two groups, respectively,

based on the rule of collision bits, the reader then sends PUSH(PRE || bit string).

C3. The bit string $q_cq_c + 1q_c + 2$ has three collision bits, which indicates complete collision.

- If the value of RGC at the reader has no collisions, the reader obtains the corresponding four kinds of transmitting bit strings according to the group number and then sends PUSH(PRE || bit string).
- If the value of RGC at the reader indicates a collision, during the process the reader firstly broadcasts PUSH(PRE,0) and then broadcasts PUSH(PRE,1) to identify tags in different group.

Considering the case that received bit string less than three bits. When the reader only receives one bit, that is, a collision bit, this means that only two tags are collided. The reader then sends PUSH(PRE || 0) and PUSH(PRE || 1). When the reader receives two bits, the execution is similar to the mentioned operation previously under the condition that there just exists one collision bit; otherwise, PUSH(PRE || 00), PUSH(PRE || 01), PUSH(PRE || 10) and PUSH(PRE || 11) are sent by the reader.

All tags have been identified when the stack is empty and then the identification process is over. Otherwise, there is a query prefix pulling from the top of the stack, which is being used to update PRE.

Example for BAT

It is assumed that there are six tags: tag 1 (000110111), tag 2 (000110001), tag 3 (000010111), tag 4 (000100001), tag 5 (000010101) and tag 6 (101010110). The details of the identification procedure with BAT are

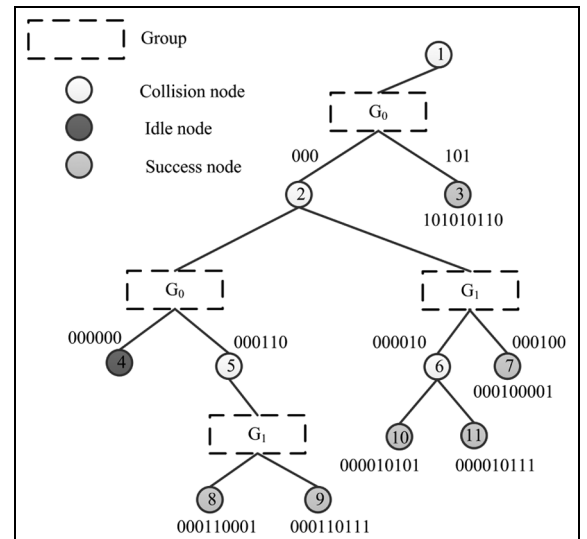


Figure 3. Example for BAT analysis.

Table 2. The identification procedure of six tags.

Slot	Request(PRE)	Tag respond	Bit string	Stack
1	ε	Tags 1, 2, 3, 4, 5, 6	Collision, X0X, RGC = 0	000, 101
2	000	Tags 1, 2, 3, 4, 5	Collision, XX0, RGC = X	101, 000000, 000110, 000010, 000100
3	101	Tag 6	Identify	000000, 000110, 000010, 000100
4	000000		Idle	000110, 000010, 000100
5	000110	Tags 1, 2	Collision, XXI, RGC = 1	000010, 000100, 00010001, 00011011
6	000010	Tags 3, 5	Collision, XI	000100, 000110001, 000110111, 000010101, 000010111
7	000100	Tag 4	Identify	000110001, 000110111, 000010101, 000010111
8	000110001	Tag 2	Identify	000110111, 000010101, 000010111
9	000110111	Tag 1	Identify	000010101, 000010111
10	000010101	Tag 5	Identify	000010111
11	000010111	Tag 3	Identify	Empty

shown in Table 2. Figure 3 shows the query tree structure using BAT. Their identification procedure can be described as follows:

First slot: At the beginning of this protocol, the reader broadcasts REQUEST(ε) to all tags, and the tags transmit their IDs and GCs of D_1 . For example, tag 1 sends 0001101110, the last bit 0 is the corresponding group. Similarly, tag 2, tag 3, tag 4, tag 5 and tag 6 sends 0001100010, 0000101110, 0001000010, 0000101010, 1010101100, respectively. Then, the reader receives X0XXX0XXX0, in which the bit string is X0X, and RGC is 0. According to the rule of collision bits, the reader can infer the bit string combinations are 000 and 101. They are pushed into the stack as candidate prefixes.

Second slot: The reader pops the prefix 000 out from the stack and broadcasts REQUEST(000). Tags 1, 2, 3, 4 and 5 reply with their remaining IDs except 000 and the reader receives XX0XX1X, in which the bit string is XX0, and RGC is X. This means there are two groups of tags, thus the bit string combinations 000, 110, 010 and 100 are inferred. Then the reader executes PUSH(000000), PUSH(000110), PUSH(000010) and PUSH(000100).

Third slot: The reader sends POP(101) and then broadcasts REQUEST(101). Only tag 6 responds and is identified directly.

Fourth slot: The reader sends POP(000000) and queries unidentified tags by 000000. No tag replies and the slot is idle.

Fifth slot: The reader sends POP(000110) and then broadcasts REQUEST(000110). Tags 1 and 2 reply and the reader receives the bit string XX1, and RGC is 1. Combining with the RGC and the bit string, the reader infers the bit string combinations, which are 001 and 111, based on the rule of collision bits. Therefore, the reader executes PUSH(000110001) and PUSH(000110111).

Sixth slot: The reader sends POP(000010) and then broadcasts REQUEST(000010). Tags 3 and 4 respond and the reader receives 1X1. Then, the bit string is X1, in which the total number of bits is two less than three. Thus 0000101 is a new PRE. The bit string combinations 01 and 11 are inferred. Then, the reader executes PUSH(000010101) and PUSH(000010111).

In the 7th, 8th, 9th, 10th and 11th slot, tags 4, 2, 1, 5 and 3 respond, respectively, and then these tags are identified. The reader terminates the identification process when no prefix exists in the stack.

Performance analysis

In an RFID system, the total number of slots for tag identification is an important parameter in the anti-collision protocol. Here, the total number of slots and system efficiency using BAT are improved. We now consider the query tree structure, as shown in Figure 4. Supposed that there are N tags in the system, the L th layer has 8^L nodes for full 8-ary tree. Each node can be divided into two groups, G_0 and G_1 , and hence there are 2×8^L groups in this layer. Each group is like a simple 8-ary tree.

Since the distribution is independent of each tag, the probability of that k tags are assigned to the same group in the L th layer follows a binomial distribution. Therefore, the probability of selecting one group from 2×8^L groups is

$$P(N, k)_L = C_N^k p^k (1-p)^{N-k} \quad (1)$$

The probability of an idle group, which means that the number of tags there is zero, is

$$P(N, 0)_L = (1-p)^N = \left[1 - \frac{1}{2 \times 8^L}\right]^N \quad (2)$$

The probability of identifying a group, which means that the group only has one tag, is

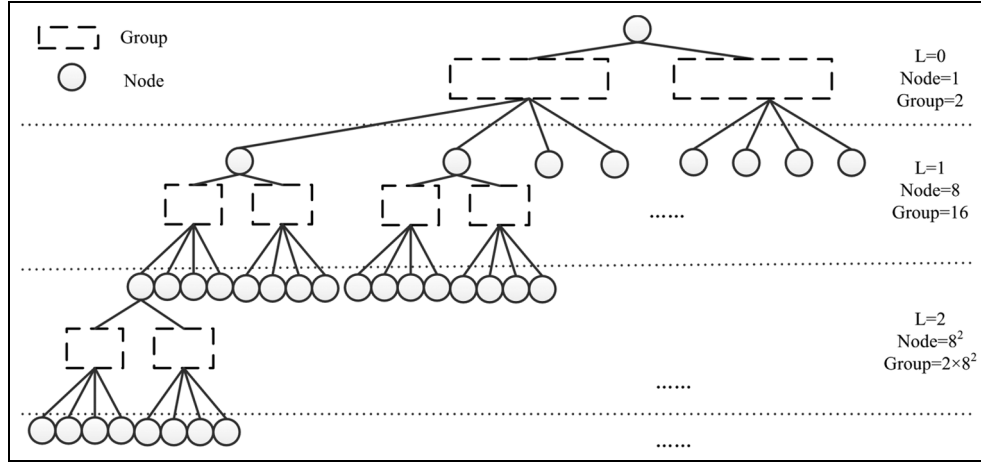


Figure 4. Query tree structure.

$$\begin{aligned}
 P(N, 1)_L &= C_N^1 p(1-p)^{N-1} \\
 &= Np \left[1 - \frac{1}{(1-2 \times 8^L)} \right]^{N-1} \quad (3)
 \end{aligned}$$

The probability of a collision group, which means that the number of tags is more than one in a group, is

$$P(N, k)_L = 1 - P(N, 0)_L - P(N, 1)_L \quad (4)$$

where $p^*(i)_L$ denotes the probability that the i th node in the L th layer is being queried. $G_0(i)_L$ and $G_1(i)_L$ are two groups which are divided by the i th node. As is known

$$p^*(i)_L = p_{G_0}^*(i)_L + p_{G_1}^*(i)_L \quad (5)$$

Here, $p_{G_0}^*(i)_L$ and $p_{G_1}^*(i)_L$ represent the probability of being queried in $G_0(i)_L$ and $G_1(i)_L$, respectively. When $L = 0$, the node is a root node which can be queried any time, so $p^*(1)_0 = 1$. If $L > 0$, other groups except the root node can be queried under the condition whereby its parent node collides, thus

$$p^*(i)_L = p_L^* = \begin{cases} 1 & L = 0 \\ \lambda_{L-1} & L > 0 \end{cases} \quad (6)$$

Thus, the probability that the j th group in the L th layer collides and that which nodes are not idle is

$$\lambda(j)_L = P(N, k)_L \times \left[1 - \left(1 - \frac{1}{4} \right)^{N_0} \right] \quad (7)$$

where $N_0 = N \times P(N, k)_L$, N_0 denotes the number of tags in the collision group.

The probability of a collision group is equal to the probability that each group of child nodes is not idle; therefore, $\lambda_L = \lambda(j)_L$.

The average total number of slots $Q(N)$ is a summation of all $p^*(i)_L$, giving

$$\begin{aligned}
 Q(N) &= \sum_{L=0}^{\infty} \sum_{i=1}^{8^L} p^*(i)_L = 1 + \sum_{L=1}^{\infty} 8^L \lambda_{L-1} \\
 &= 1 + 8 \times \sum_{L=0}^{\infty} 8^L \lambda_L \quad (8)
 \end{aligned}$$

Then, the total number of time slots of BAT protocol is

$$\begin{aligned}
 Q(N) &= 1 + 8 \\
 &\times \sum_{L=0}^{\infty} 8^L \left\{ 1 - \left[1 - \frac{1}{(2 \times 8^L)} \right]^N - Np \left[1 - \frac{1}{(2 \times 8^L)} \right]^{N-1} \right\} \\
 &\times \left[1 - \left(1 - \frac{1}{4} \right)^{N_0} \right]^4 \quad (9)
 \end{aligned}$$

According to the total time slots of BAT, we can obtain system efficiency S , which is the ratio between the number of tags to be identified and the total time slots required to recognize them, that is, $S = N/Q(N)$. System efficiency means tag identification efficiency. The greater the system efficiency, the higher the tag identification efficiency.

Figure 5 shows the comparison between simulation and analytical results of BAT in total time slots and system efficiency. It can be seen that the performance analysis results are highly accurate and close to the simulation.

Simulation and comparison

In this section, we compare BAT protocol with QT, 8-ary QT protocol, A4PQT and CT. The reasons that we select the four protocols for performance evaluation are as follows: first, QT is one of the typical tree-based protocols. And our proposed BAT protocol is based on QT. Both the protocols need stack to store query prefixes. Second, in 8-ary QT protocol, the query prefix is

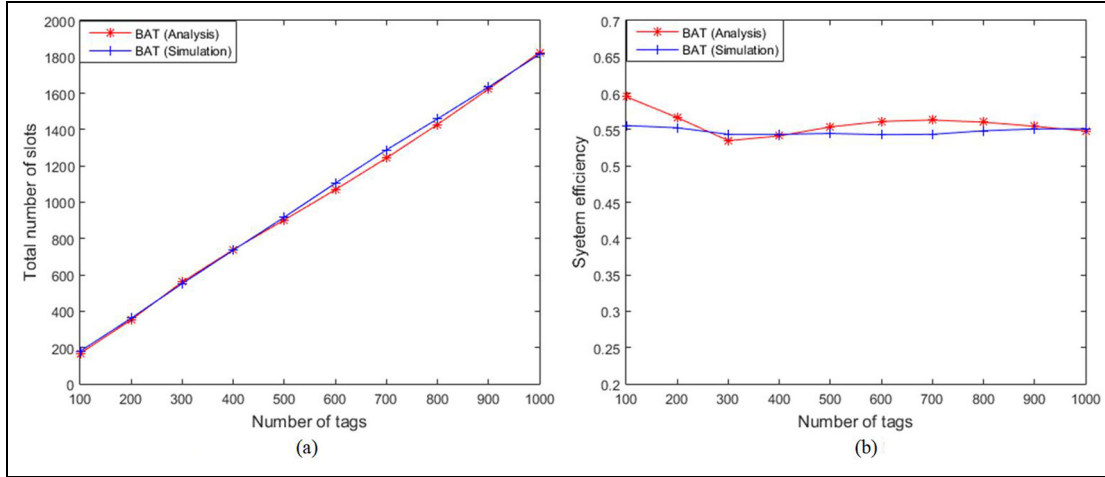


Figure 5. Comparison of analysis and simulation results for BAT protocol: (a) total number of slots and (b) system efficiency.

updated by increasing three binary bit, and BAT protocol also infers decoded data with three bits each time to speed up the identification process. Third, A4PQT is an improvement protocol of QT protocol and prunes the idle time slot branches based on the information of collision bits. If the reader receives the data that have two successive bits, A4PQT generates four new branches instead of cutting any branches, which produces additional two idle slots. To further decrease redundancy idle time slots, the BAT protocol utilizes the rule of collision bits to infer the transmitted bit strings accurately. Finally, BAT protocol with QT, 8-ary QT protocol, A4PQT and CT are not sensitive to the length of tag ID. Comparing with A4PQT, QT, 8-ary protocol and CT, the performance of our proposed BAT protocol is the best in terms of the total number of slots, the number of idle slots and system efficiency. These simulations are conducted and displayed in the following detailed description.

In our protocol, we consider an ideal transmission channel, without capture-effect and path-loss affects. The RFID system, running on a MATLAB 2012a simulation platform, consists of a single reader and numerous tags, and the number of tags is increased from 100 to 1000. To investigate the effects of tag ID length on system efficiency, Figure 6 shows the system efficiency under conditions that the tag ID lengths are 48, 96, 128, 256, 512 and 1024 bits, respectively. From Figures 7–9, the length of each tag ID is set to 96 bits.

Figure 6 shows the system efficiency with different length of tag IDs for BAT and QT. From Figure 6, we can see that, when the tag ID length changes, the system efficiency of QT varies between 0.34 and 0.42 and the performance of BAT is stable at close to 0.55. Both QT and BAT utilize Manchester coding, the system efficiency of QT is affected by the changing of ID length, while in BAT protocol ID length has almost no

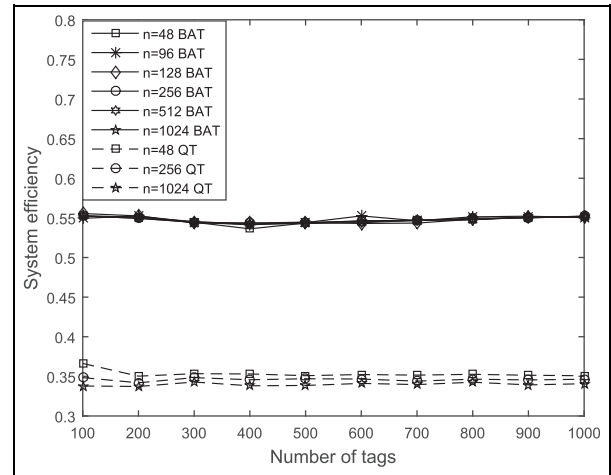


Figure 6. System efficiency with different length of tag IDs.

influence on the system efficiency. In QT, the query prefix is updated by increasing one binary bit so that QT is influenced by the ID length. In BAT protocol, the reader receives bit string from the first collision bit, and uses different cases to identify tags depending on the number of collision bits. Besides, system efficiency is the ratio between the number of tags to be identified and the total time slots, and from equation (9), the tag ID length is unrelated to the total time slots. So the system efficiency is insensitive to the length of tag ID in BAT protocol.

Figure 7 shows that the number of idle slots in BAT is less than in A4PQT, QT and 8-ary protocol. Moreover, the effect becomes more obvious with increasing of N . Note that in BAT protocol the developing tendency of the number of idle slots is slow and steady. Since the reader can more accurately infer the prefixes of responding tags than A4PQT, BAT protocol reduces the number of idle slots. Compared to other

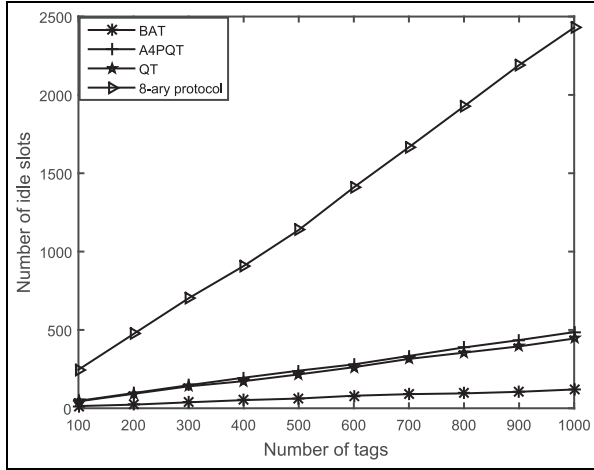


Figure 7. Number of idle slots with different protocols.

three protocols, the increasing speed of the number of idle slots with 8-ary protocol is the fastest. When the number of unidentified tags is 1000, 8-ary protocol is almost 20 times the number of idle slots with BAT protocol. Therefore, in the case of a large number of tags, BAT protocol is more excellent.

In QT, the reader inquires bit by bit, while in 8-ary QT protocol and BAT protocol the query prefix is updated by increasing three binary bits each time, which speeds up the identification process. A4PQT decreases some idle time slots compared with 8-ary protocol. And BAT protocol determines transmitted bit string combinations accurately and reduces more time slots than A4PQT. From Figure 8, it is seen that the total number of slots in BAT is much less than in A4PQT, QT, 8-ary protocol and CT. When the number of unidentified tags is 1000, the total number of slots in BAT protocol, A4PQT, QT, 8-ary protocol and CT are 1817, 2082, 2889, 3920 and 1999, respectively. Compared to the A4PQT, QT, 8-ary protocol and CT, BAT reduces the total number of slots effectively in a large-scale RFID system.

The system efficiency for identifying tags is used as the performance criterion. Figure 9 shows the system efficiency of BAT, A4PQT, QT, 8-ary protocol and CT, for identification of all tags. The system efficiency is equal to the ratio between the number of readable slots and the total number of slots. In BAT, A4PQT, QT, 8-ary protocol and CT, the number of tags to be identified is same. Therefore, the less the total number of time slots, the better the system efficiency. From Figure 9, we can see that the system efficiency of BAT is maintained at 0.55 and significantly better than A4PQT, QT, 8-ary protocol and CT. In the system efficiency, the second highest of these three protocols is CT, which tends asymptotically towards 0.5. The system efficiency of the 8-ary protocol is the lowest, at only 0.25.

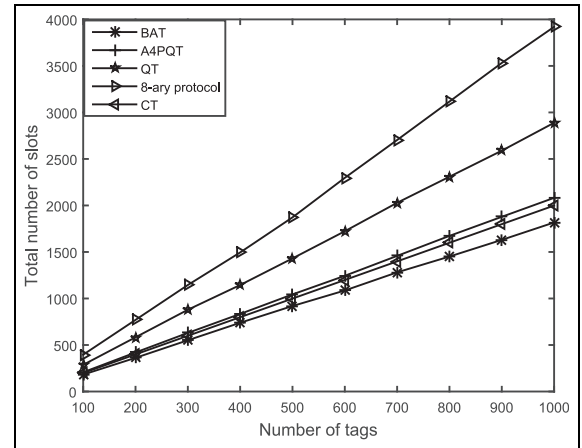


Figure 8. Total number of slots with different protocols.

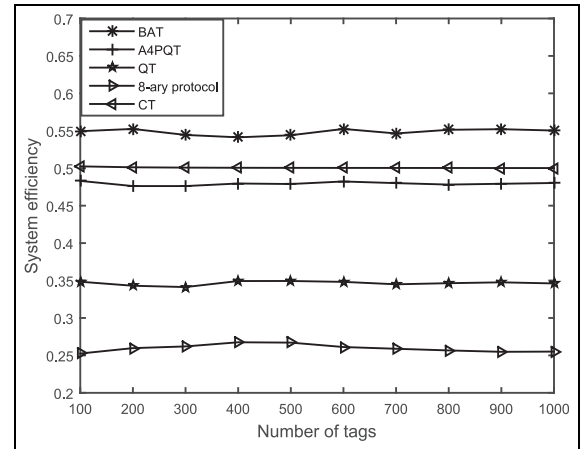


Figure 9. System efficiency with different protocols.

Conclusion

In this article, we propose a large-scale tag identification protocol, called BAT anti-collision protocol, and design a characteristic-value-based grouping rule and a collision bits rule to infer the transmitted bit string combinations. Furthermore, we also take the scenario that the bit string less than three bits in the process of tag collection into consideration. Both theory and simulation analyses show that the tag ID length is unrelated to system efficiency in our BAT protocol. Besides, our proposed BAT protocol can achieve better performance on system efficiency, the number of idle slots and total slots than A4PQT, QT and 8-ary protocol. Specifically, when compared with A4PQT, the system efficiency of our proposed BAT protocol is improved by 12%.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This research was financially supported by the Fundamental Research Funds of Jilin University (SXGJQY2017-9, 2017TD-19), the Funds of Jilin and Changchun Science and Technology Department (20150101050JC, 16SS02) and the National Natural Science Foundation of China (Nos. 61371092, 61401175, and 61771219).

References

- Klaus F. *RFID handbook: fundamentals and applications in contactless smart cards and identification*. New York: Wiley, 2003.
- Djeddou M, Khelladi R and Benssalah M. Improved RFID anti-collision algorithm. *AEU-Int J Electron C* 2013; 60(3): 756–766.
- Ullah S, Alsalihi W and Alsehim A. A review of tags anti-collision and localization protocols in RFID networks. *J Med Syst* 2012; 36(6): 4037–4050.
- Fu Y, Qian ZH, Wang X, et al. Identifying the unknown tags in a large RFID system. *China Commun* 2017; 14(1): 135–145.
- Yu J, Lee W and Du DZ. Reducing reader collision for mobile RFID. *IEEE T Consum Electr* 2011; 57(2): 574–582.
- Li M, Qian ZH, Zhang X, et al. Slot-predicting based ALOHA algorithm for RFID anti-collision. *J China Inst Commun* 2011; 32(12): 43–50.
- Zanella A. Adaptive batch resolution algorithm with deferred feedback for wireless systems. *IEEE T Wirel Commun* 2012; 11(10): 3528–3539.
- Jiang Y and Zhang R. An adaptive combination query tree protocol for tag identification in RFID systems. *IEEE Commun Lett* 2012; 16(8): 1192–1195.
- Yang CN, Hu LJ and Lai JB. Query tree algorithm for RFID tag with binary-coded decimal EPC. *IEEE Commun Lett* 2012; 16(10): 1616–1619.
- Cui Y and Zhao Y. Performance evaluation of a multi-branch tree algorithm in RFID. *IEEE T Commun* 2010; 58(5): 1356–1364.
- Liu L and Lai S. ALOHA-based anti-collision algorithms used in RFID system. In: *Proceedings of the IEEE international conference on wireless communications, networking and mobile computing*, Wuhan, China, 22–24 September 2006, pp.1–4. New York: IEEE.
- Kim SC, Cho JS and Kim SK. Performance improvement of hybrid tag anticollision protocol for radio frequency identification systems. *Int J Commun Syst* 2013; 26(6): 705–719.
- Kodialam M and Nandagopal T. Fast and reliable estimation schemes in RFID systems. In: *Proceedings of the 12th annual international conference on mobile computing and networking*, Los Angeles, CA, 23–29 September 2006, pp.322–333. New York: ACM.
- Jia X, Feng Q and Yu L. Stability analysis of an efficient anti-collision protocol for RFID tag identification. *IEEE T Commun* 2012; 60(8): 2285–2294.
- Hush DR and Wood C. Analysis of tree algorithms for RFID arbitration. In: *Proceedings of the IEEE international symposium on information theory*, Cambridge, MA, 16–21 August 1998, pp.107. New York: IEEE.
- Lai YC, Hsiao LY, Chen HJ, et al. A novel query tree protocol with bit tracking in RFID tag identification. *IEEE T Mobile Comput* 2013; 12(10): 2063–2075.
- Liu XH, Qian ZH, Zhao YH, et al. An adaptive tag anti-collision protocol in RFID wireless systems. *China Commun* 2014; 11(7): 117–127.
- Zhang W, Guo Y, Tang X, et al. An efficient adaptive anticollision algorithm based on 4-ary pruning query tree. *Int J Distrib Sens N* 2013; 9(12): 1–7.
- Zheng F and Kaiser T. Adaptive Aloha anti-collision algorithms for RFID systems. *EURASIP J Embed Syst* 2016; 2016: 7.
- Su J, Sheng ZG, Hong DF, et al. An effective frame breaking policy for dynamic framed slotted Aloha in RFID. *IEEE Commun Lett* 2016; 20(4): 692–695.
- Liu XL, Wang H, Kwok YK, et al. Exploiting the prefix information to enhance the performance of FSA-based RFID systems. *Comput Commun* 2015; 56: 108–118.
- Solic P, Radic J and Rozic N. Early frame break policy for Aloha-based RFID systems. *IEEE T Autom Sci Eng* 2016; 13(2): 876–881.
- Xie L, Han H, Li Q, et al. Efficient protocols for collecting histograms in large-scale RFID systems. *IEEE T Parallel Distr* 2016; 26(9): 2421–2433.
- Landaluce H, Perallos A and Zuazola IJG. A fast RFID identification protocol with low tag complexity. *IEEE Commun Lett* 2013; 17(9): 1704–1706.
- Yang CN, Kun YC, Chiu CY, et al. A new adaptive query tree on resolving RFID tag collision. In: *Proceedings of the IEEE international conference on RFID-technology and Applications*, Guangzhou, China, 17–19 June 2010, pp.153–158. New York: IEEE.